Machine Learning for Reliable Communication Under Coarse Quantization

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von

Maximilian Stark

aus

Parchim, Deutschland

2021

Vorsitzender des Prüfungsausschusses:
Prof. Dr.-Ing. habil. Alexander Kölpin

1. Gutachter:
Prof. Dr.-Ing. Gerhard Bauch

2. Gutachter:
Prof. Dr.-Ing. habil. Volker Kühn

3. Gutachter:
Prof. Richard D. Wesel

Tag der mündlichen Prüfung:
23.09.2021

 orcid.org/0000-0002-1750-5895

In memory of my beloved father

Acknowledgements

Right before my Ph.D. voyage kicked off, Prof. Bauch told me that the actual topic of the thesis would only be part of the adventure ahead and that the journey itself will be more exciting and challenging than I would imagine. At that time, I had no clue how right he was, which obstacles I had to surpass, and how many stunning experiences and true friends I would make around the globe.

First, I would like to thank Prof. Bauch for his continues support. Additionally, I want to thank all my colleagues at the Institute of Communications for their help, discussions and valuable inputs.

During Globecom'18 I met Prof. Wesel for the first time. His passionate interest in quantized decoding was so inspiring. I want to thank him and his Ph.D. student Linfang Wang for their tremendous support and fruitful exchange, which even could not be diminished despite the Covid pandemic stopping me from spending time at UCLA.

I met Prof. Kühn during my first weeks as a research assistant in Rostock. I want to thank him and his Ph.D. student Steffen Steiner for their intense support and the time we spend finding new applications for the information bottleneck.

The time I started my thesis also marked the beginning of the rise of deep learning in communications. I'm extremely grateful that Jakob Hoydis, Fayçal Ait Aoudia, and Matthieu Goutay gave me such a happy home at Nokia Bell Labs Paris. They paved the road for what I did in the second part of the thesis and always helped me out when my limited French skills were insufficient for the world outside Bell Labs. A big thanks also to Sebastian Cammerer, Sebastian Dörner, and Prof. ten Brink from the University of Stuttgart. They all taught me fascinating insides about deep learning in communications and the joint discussions were extremely helpful.

In particular, I would like to thank my dear friend Jan Lewandowsky for the endless hours of coding, talking, publishing, polishing, editing, laughing, etc.

Ein ganz besonderer Dank gebührt meinen Freunden und meiner Familie, die mich bis zum Ende immer auf dieser langen Reise unterstützt haben. Ihr habt mich angetrieben, mir Kraft und Zuversicht gegeben und wart auch in den schwersten Momenten immer für mich da. Dafür bin ich euch unendlich dankbar.

Hamburg, September 2021

Abstract

Error-free transmission is among the utmost goals of modern communication systems. By means of information theory, fundamental, theoretical limits for error-free transmission can be determined depending on the transmission conditions. In practice, however, application-specific limitations often prevent the use of theoretically optimum algorithms. Especially in wireless transmission, often strict requirements for latency, energy consumption and available computing power of mobile devices exist. Therefore, the information to be processed at the receiver usually has to be quantized very coarsely to enable efficient signal processing.

This work investigates the use of machine learning methods for reliable transmission despite coarsely quantized soft information at the receiver. Machine learning is often used synonymously with deep learning using neural networks. However, this thesis takes a more holistic approach and looks at respective problems from an information theoretical perspective. Based on these investigations, appropriate machine learning methods are chosen. It is shown that very powerful error correction is possible despite coarse quantization if the *relevant* information is maximized. However, this requires an interdisciplinary approach based on the interplay of information theory, machine learning, and communications engineering. A special focus is put on the information bottleneck method. The information bottleneck method is a clustering framework from the field of classical, model-based machine learning.

First, the information bottleneck method is presented in detail and compared with other approaches to quantization in information theory. New algorithms are designed which are particularly suitable for signal processing problems. In addition, two necessary extensions of the method are presented, i.e., *information bottleneck graphs* and *message alignment*.

In this thesis, the single elements of a receiver are first replaced by coarsely quantized building blocks designed with the information bottleneck method. Here, a particular focus lies on the design of modern channel decoding methods using the information bottleneck method for very reliable transmission. In particular, decoders for different variants of low-density parity-check codes are studied. Interestingly, the designed coarsely quantized signal processing units achieve almost the same performance in terms of reliability as conventional non-quantized methods.

In the second part of the thesis, the idea of mutual-information-based signal processing is extended to data-based learning methods. It is shown that by means of representation learning using autoencoders, transmitter and receiver can be optimized together to maximize the relevant information over the entire transmission

chain. In contrast to the first part of the work, mutual-information-based signal processing units are developed without detailed knowledge of the model. However, the neural networks are not considered as black boxes. Instead, the corresponding loss functions will be designed according to information-theoretical quantities to maximize the relevant information.

Contents

Chapter 1

Introduction

> "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point."
>
> *Claude E. Shannon, 1948*

This general definition by Claude E. Shannon inspired communication engineers over the last decades to build more and more evolved communications systems providing seamless connectivity for people all over the world. Shannon is often called the *father of information theory* and *founder of the information age*. Information theory enables to quantify and process information and derive fundamental limits of reliable information exchange.

The permanent exchange and immediate access to information, as well as constant availability, have become an elementary, inevitable component of our modern society. Especially the advances in *mobile* communications, i.e., broadband access to the internet everywhere at any time, induced a vast amount of new opportunities but also challenges. For the first time, the latest generation of mobile communications, i.e., the 5th generation (5G), explicitly focuses on the wireless connection of machines. This lays the foundation for the internet of things and massive machine-to-machine communication. At the same time, the rise of tiny, mobile, battery-powered devices demands more energy-efficient, resource-saving communication algorithms than ever before.

In turn, modern wireless communication engineering needs to balance three crucial parameters namely:

Latency The latency, i.e., the end-to-end delay from transmission until recovery, is not only related to the overall system throughput but also impacts the age of

information, which is especially crucial in remote-controlled applications and autonomous driving [Yat20]. Often, the computational burden of complicated baseband signal processing largely impacts the latency.

Reliability The reliability measures the success rate of the transmission. It can be controlled by appropriate choice of *modulation and channel coding* schemes. An unsuccessful transmission requires a retransmission that increases the latency. However, especially modern channel decoding to increase the reliability of the transmission is itself the most computationally demanding signal processing block.

Resources Finally, available hardware resources like battery, chip area, and computational capabilities often prohibit the implementation of the most powerful signal processing units.

This work presents a novel approach to reliable, low-complexity communication under coarse quantization based on the interplay of applied information theory, machine learning, and communications engineering. These three interdisciplinary pillars form the basis for the so-called *mutual-information based signal processing*. Table 1.1 provides a brief timeline of this journey towards mutual-information-based signal processing, which started around 2002 with the work by Ma et al. [MZY+02]. Mutual-information-based signal processing aims to design signal processing units which learn to preserve the *relevant information*. In turn, the internal resolution, i.e., the bit-width, of the signal processing units can be reduced without significantly deteriorating the system performance. This reduction in resolution, i.e., a *coarse* quantization typically reduces the implementation complexity. Hence, this approach promises an optimum trade-off between very reliable transmission and high resource and energy efficiency. The works summarized in Table 1.1 will be described in more detail in the next chapters. It will be shown that the *information bottleneck method* appears as an attractive framework to design mutual-information-based signal processing. The information bottleneck method was introduced by Tishby et al. in [TPB99]. This framework allows extracting relevant information from a high-resolution observation and representing this relevant information very compactly.

A collection of own publications is depicted in dark blue in Table 1.1 which also serves as an outline of this thesis. The major contributions of this work are threefold:

1. The information bottleneck method is investigated in the context of communications engineering. The method itself is compared in great detail to existing approaches in information theory and machine learning. Furthermore, new algorithms within the generic information bottleneck framework are presented.

TABLE 1.1 Timeline Mutual-Information-Based Signal Processing.

Year	Advances in Mutual - Information - Based Signal Processing
2002	Quantizer Design that Maximizes Mutual Information [MZY+02]
2003	Belief Propagation Decoding with Mutual Information Maximizing Operations [Tho03a; LT05]
2008	Discrete Density Evolution [KYK08]
	Relaying and Network Coding [ZKB+08]
2011	Mutual-Information Optimized Quantization for LDPC Decoder for Flash [WCS+11]
2012	Quantization of Channels with Memory [ZSK12]
2015	LDPC Decoding using Mutual-Information Maximizing Look-up Tables [MBB+15; RK15; LB15; RK16; LSB16b]
2016	General Receiver Design [LSB16a]
2017	Channel Estimation [LSB17]
	Quantizers for Higher Order Modulation [LSB17]
2018	Irregular LDPC Decoding with Message Alignment [SLB18b]
	Construction of Polar Codes [SSB18]
	Two-bit Message Passing Decoding [YSM+19]
2019	Decoding Non-Binary LDPC Codes [SBL+19]
	Decoding of Polar Codes [SSB19]
	Min-LUT Decoding of LDPC Codes [MMB20]
	Computational Domain Decoding [HCM19]
2020	Decoding Punctured Rate-Compatible LDPC Codes [SWB+20]
	Neural Information Bottleneck Decoding [SLB20]
	Trainable Communication Systems [CAD+20]

These algorithms are investigated with respect to their suitability for common problems in communications. The method itself is generalized for very complicated statistical inference tasks resulting in the so-called *information bottleneck graphs*. Furthermore, the original information bottleneck method was designed as a clustering framework. This work presents a further extension to the method, i.e., *message alignment* which enables the coarsely quantized exchange of information in irregular, random graphs, for example, distributed sensor networks or irregular channel coding schemes.

2. This work studies the problem of channel decoding under coarse quantization for very modern channel codes. In particular, many different variants of so-called low-density parity-check (LDPC) codes are considered. Please note that the presented approach is also applicable to other modern channel coding techniques like polar codes [SSB18; SSB19]. It will be shown that the presented mutual-information-based signal processing offers extremely competitive performance despite very coarse quantization and very simplified operations. Extensive numerical simulation and real hardware implementations are conducted to underline the superiority of the presented approach compared to state-of-the-art techniques.

3. The information bottleneck method itself belongs to the class of *model-based* machine learning techniques which require very accurate models. Often these models cannot capture the entire reality, or implementing the models becomes practically infeasible. Thus, in the third part of this thesis, a *data-driven* approach towards mutual-information-based signal processing is outlined. In particular, deep learning using neural networks is interpreted as an instance of the information bottleneck method that focuses solely on preserving relevant information. Detailed information-theoretical reasoning is provided, which allows adjusting the training process of neural networks resulting in capacity-achieving end-to-end communication.

In general, this thesis investigates the applicability of mutual-information based signal processing and the information bottleneck method in the broad context of reliable communication under coarse quantization.

Chapter 2 briefly reviews fundamentals of information theory, Bayesian statistics and channel coding.

This thesis aims to provide deep insides to the information bottleneck method and contributes extensions and generalization of this framework always with respect to applications in mutual-information based signal processing. Hence, the information

bottleneck method and its relations to other quantization methods are investigated in detail in Chapter 3. Here, the example of mutual-information-based channel output quantization for different channel models is used as a running example.

Chapter 4 presents an essential extension to the classical information bottleneck setup, i.e., message alignment. Within the discussion of message alignment another scenario is considered, i.e., joint quantizer design for distributed sensing. Furthermore, Chapter 4 presents the problem of coarsely quantized maximum-a-posteriori multiple symbol detection for differential modulation over phase noise channels to introduce the developed concept of information bottleneck graphs.

Afterwards, Chapter 5 presents the design of decoders for LDPC codes using the information bottleneck method. These information bottleneck decoders enable reliable transmission even under coarse quantization. Many different families of LDPC codes exist. Starting with regular LDPC codes, the decoder design is generalized to arbitrary irregular LDPC codes. In the most recent 5G communication standard, LDPC codes are used for mobile broadband. The selected LDPC codes belong to the very flexible class of rate-compatible LDPC codes. It will be shown how the design of information bottleneck decoders differs for these standardized 5G LDPC codes. Furthermore, Chapter 5 provides a detailed comparison of recent alternative LDPC decoder designs for coarse quantization which build upon the information bottleneck decoder approach. Finally, a brief analysis of hardware aspects for different hardware architectures closes Chapter 5.

A fundamentally different class of LDPC codes are so-called non-binary LDPC codes studied in Chapter 6. These error-correction codes show very strong error-correcting capabilities especially in the short blocklength regime. However, so far these codes are often ignored in practical systems due to their extremely high computational complexity. This motivates the use of the information bottleneck method to construct a low-complexity decoder which shows error-correction performance similar to state-of-the-art techniques but with much lower computational complexity.

Chapter 7 presents *data-driven* mutual-information-based signal processing design. Starting from limits of the *model-based* approach identified in Chapter 5 and Chapter 6, the neural information bottleneck decoder is presented. This decoder is a purely data-driven approach towards coarsely quantized channel decoding. A supervised and unsupervised decoder is presented which learns to preserve and maximize relevant information without explicit knowledge of the entire statistical dependencies. In the second part of Chapter 7, the entire communication chain is described by

neural networks. It is shown that so-called autoencoders inherently focus on maximizing the mutual information. Thus, to some extend, end-to-end learning is natural to mutual-information based signal processing as studied in previous chapters. As this thesis, aims to integrate and analyze different notions of mutual-information based signal processing, end-to-end learning is studied in information theoretical means and compared to the information bottleneck method.

To conclude, Chapter 8 summarizes the findings of this thesis, discusses open questions and proposes possible future research directions.

Chapter 2

Fundamentals of Probability Theory, Bayesian Statistics and Information Theory

This chapter defines the notations used throughout this thesis. Therefore, the most important theoretical concepts and definitions from statistics and information theory are reviewed. Section 2.1 introduces the concept of random variables, probabilities, and probability distributions.

The main focus of this thesis is on the design of mutual-information based signal processing units. Most mutual-information-based signal processing units can be related to high-dimensional inference problems. In decision theory, such complicated decision problems are often tackled with so-called *factor graphs*. Using factor graphs, the inference problem can be solved very efficiently using the sum-product algorithm, sometimes referred to as *message passing* in literature [KFL01]. These concepts are described in Section 2.2.

Section 2.3 recalls information-theoretical measures like entropy and mutual information. It will be shown that a more vivid and illustrative perspective on information theory termed *information geometry* proves constructive in understanding the information bottleneck method and related algorithms.

Finally, this chapter reconsiders Shannon's channel coding theorem and linear block codes in Section 2.4.

2.1 Fundamentals of Probability Theory

The definitions used in this thesis follow mainly those in [Cov06] and [PP02]. First, random variables and related concepts like sample space, joint and conditional probability distributions are defined for *discrete* and *continuous* random variables likewise.

2.1.1 Discrete and Continuous Random Variables

In general, a random variable is a function that maps the outcome of a random experiment to a real number [PP02]. This thesis denotes random variables by capital sans serif letters, for example, X or Y. The sample space of the random variable is denoted $\mathcal{X}$ and realizations of the random variable are defined by $x \in \mathcal{X}$. One distinguishes between discrete and continuous random variables.

Discrete Random Variables

A discrete random variable has a finite number of realizations given by $|\mathcal{X}|$, i.e., the cardinality of the sample space. Furthermore, $\Pr(\mathsf{X} = x)$ is the probability that a certain realization x is observed. Please note that for discrete random variables

$$\Pr(\mathsf{X} = x) > 0, \forall x \in \mathcal{X}. \tag{2.1}$$

Recording the probabilities for all $x \in \mathcal{X}$ allows us to determine the probability distribution $p_{\mathsf{X}}(x)$, also referred to as *probability mass function*. For ease of notation, the subscript X is dropped and the probability distribution of X is denoted as $p(x)$. By definition,

$$\sum_{x \in \mathcal{X}} p(x) = \sum_{x \in \mathcal{X}} \Pr(\mathsf{X} = x) = 1 \tag{2.2}$$

which is often termed the law of total probability [PP02].

Continuous Random Variables

Continuous random variables cover a broader class of random experiments than discrete random variables as the number of realizations is infinite. In turn, the sample space of continuous random variables is a subset of the real numbers. This thesis represents continuous random variables by capital sans serif letters with an additional tilde as an accent, i.e., $\tilde{\mathsf{X}}$ to distinguish between continuous and discrete random variables. To fully describe a continuous random variable a more general concept, i.e., the cumulative distribution function is required to derive the probability distribution $p(\tilde{x})$, also referred to as *probability density function*.

Let $\Pr(\tilde{\mathsf{X}} \leq \tilde{x})$ be the probability that experimental outcomes smaller or equal $\tilde{x}$ are observed. As $\Pr(\tilde{\mathsf{X}} \leq \tilde{x})$ is a function of $\tilde{x}$, $F(\tilde{x})$ defines the *cumulative distribution function* where

$$F(\tilde{x}) = \Pr(\tilde{\mathsf{X}} \leq \tilde{x}), -\infty < \tilde{x} < \infty. \tag{2.3}$$

Thus, in contrast to discrete random variables, $\Pr(\tilde{\mathsf{X}} = \tilde{x})$ is zero for continuous random variables. However, considering the limit $\Pr(\tilde{\mathsf{X}} \in \{\tilde{x}, \tilde{x} + \Delta\})$ with $\Delta \to 0$, one can derive the differential quotient

$$p(\tilde{x}) = \lim_{\Delta \to 0} \frac{F(\tilde{x} + \Delta) - F(\tilde{x})}{\Delta} \tag{2.4}$$

termed *probability density function*. Similar to the discrete case, the law of total probability from (2.2) is given by

$$\int_{-\infty}^{\infty} p(\tilde{x}) \mathrm{d}\tilde{x} = 1. \tag{2.5}$$

In general, one concludes that probability density functions and probability mass functions are closely related. Thus, the general term probability distribution is used in this thesis if no distinction is needed. Furthermore, other concepts presented in this chapter will only be introduced for discrete random variables for ease of notation as an application to continuous random variables is straightforward.

2.1.2 Joint, Conditional and Marginal Distributions

Having defined random variables and probability distributions, in this section further concepts from statistics involving multiple random variables are discussed.

Joint Probability Distribution

Given two random variables X with $x \in \mathcal{X}$ and Y with $y \in \mathcal{Y}$ the probability that x *and* y are observed together is given by $\Pr(\mathsf{X} = x, \mathsf{Y} = y)$. Recording the respective probabilities for all pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \times \mathcal{Y} = \{(x, y) | x \in \mathcal{X}, y \in \mathcal{Y}\}$ denotes the Cartesian product, yields the *joint probability distribution* $p(x, y)$. For the joint probability distribution

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) = 1 \tag{2.6}$$

holds.

Marginal Probability Distribution

Given a joint distribution $p(x, y)$ the individual *marginal* distributions $p(x)$ and $p(y)$ are found by marginalizing, i.e., summing out the other variable

$$p(x) = \sum_{y \in \mathcal{Y}} p(x, y) \tag{2.7}$$

$$p(y) = \sum_{x \in \mathcal{X}} p(x, y)\,. \tag{2.8}$$

In Section 2.2, the process of computing the marginals will also be referred to as the sum-rule of probabilities [Bis09]. Interestingly, it can be observed that applying the sum-rule is irreversible, i.e., it is possible to obtain $p(x)$ and $p(y)$ given $p(x, y)$. However, in general, it is *not* possible to derive $p(x, y)$ given $p(x)$ and $p(y)$. This is only possible in the special case that X and Y are so-called *independent*.

Conditional Probability Distribution

In the previous section, the special case of independent random variables was mentioned. This property is closely related to the question:

"How likely is a certain event given that another event was observed?"

In statistics, this relation is described using the concept of conditional probability distributions $p(y|x)$. The conditional probability distribution is defined by all probabilities $\Pr(\mathsf{Y} = y|\mathsf{X} = x), \forall (x, y) \in \mathcal{X} \times \mathcal{Y}$ indicating how likely y will be observed given x. By applying the product-rule of probabilities [Bis09], every joint distribution can be written as the product of the conditional distributions and marginals

$$p(x, y) = p(y|x)p(x) = p(x|y)p(y). \tag{2.9}$$

Given that Y and X are independent, i.e., that observing y is independent of observations of x, one obtains

$$p(x, y) = p(y|x)p(x) = p(y)p(x). \tag{2.10}$$

Information Geometry

Often it is very convenient to investigate joint and conditional distributions from a graphical perspective. This perspective is especially useful when investigating different information bottleneck algorithms in Chapter 3. In general, probability distributions can be visualized using points in a $|\mathcal{X}| - 1$-dimensional manifold [Ama16]. For

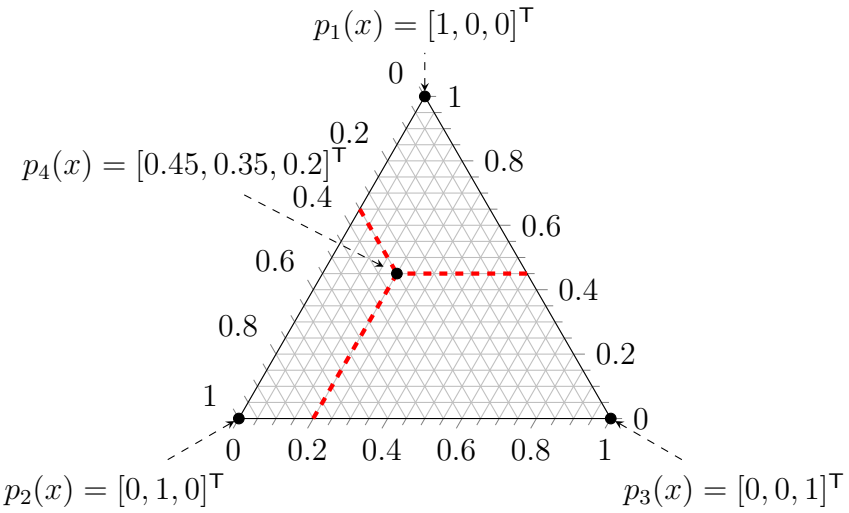


Figure 2.1: Probability simplex for ternary random variables.

discrete random variables, the probability manifold is an $|\mathcal{X}|-1$-dimensional simplex, the *probability simplex*. For a binary random variable, the respective simplex is just a line. Fig. 2.1 shows the probability simplex for a ternary random variable X. For ease of notation, let us denote the discrete probability mass function of the ternary random variable X in vector form, i.e., $p(x) = [\Pr(\mathsf{X} = 0), \Pr(\mathsf{X} = 1), \Pr(\mathsf{X} = 2)]^\mathsf{T}$ with $\mathcal{X} = \{0, 1, 2\}$. The elements of this vector can be interpreted as coordinates of the probability mass function in the simplex.

The following four probability distributions

$$
\begin{aligned}
p_1(x) &= [1, 0, 0]^\mathsf{T} \\
p_2(x) &= [0, 1, 0]^\mathsf{T} \\
p_3(x) &= [0, 0, 1]^\mathsf{T} \\
p_4(x) &= [0.45, 0.35, 0.2]^\mathsf{T}
\end{aligned}
$$

of the exemplary ternary random variables X_1, X_2, X_3 and X_4 are plotted as points in the respective simplex.

2.1.3 Moments of Random Variables

Probability distributions fully describe the properties of random variables. However, sometimes scalar quantities characterizing random variables are of interest, so-called moments. Let $f(x)$ define an arbitrary function of the random variable X. The most common moment is the expectation of $f(x)$, denoted $\mathbb{E}[f(x)]$, which defines the average value of the function $f(x)$ given the probability distribution $p(x)$ [Bis09]. For a discrete random variable X and transformation $f(x)$ the expectation is computed

as

$$\mathbb{E}\left[f(x)\right] = \sum_{x \in \mathcal{X}} f(x)p(x)\,. \tag{2.11}$$

In Chapter 7, deep learning using sample based training will be performed. In the context of data-driven machine learning it is common to approximate the expectation as

$$\mathbb{E}\left[f(x)\right] \cong \frac{1}{N}\sum_{i=1}^{N} f(x_{(i)})\,, \tag{2.12}$$

where N denotes the total number of samples $x_{(i)}$. The expectation can also be generalized to joint distributions and conditional distributions involving multiple random variables, e.g., X and Y. Here, a subscript indicates for which random variable the average is computed, i.e.,

$$\mathbb{E}_{\mathsf{X}}\left[f(x,y)\right] \tag{2.13}$$

denotes that the expectation is computed using $p(x)$ and

$$\mathbb{E}_{\mathsf{X}|\mathsf{Y}}\left[f(x)\right] = \sum_{x \in \mathsf{X}} f(x)p(x|y) \tag{2.14}$$

is the conditional expectation.

In addition to the expectation, the variance $\mathrm{var}\left[X\right]$ of a random variable X is of great interest. The variance is computed as

$$\mathrm{var}\left[X\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2\,. \tag{2.15}$$

2.1.4 Bayesian Statistics and Bayes Rule

Given the definitions from Section 2.1.2 it is possible to introduce the so-called Bayesian perspective on probabilities or *Bayesian statistics*. Bayesian statistics will play a fundamental role in all topics discussed in this thesis and is thus reviewed in more detail in this section. Bayesian statistics, named after Thomas Bayes, stands in clear contrast to the frequentist perspective on probabilities [Bis09], which is briefly reviewed first.

The frequentist perspective on probabilities only holds for very well-defined random experiments, like playing cards, rolling dices, etc. Here, the probability of an event is determined based on the relative frequency of a particular event after many trials

of the random experiment. It is assumed that in the limit, the relative frequency converges to the true probability of the event.

However, for some uncertain events like the rise of the sea level due to global warming or the spreading of a new virus, it is impossible to repeat these events many times to determine their probabilities from a frequentist perspective. Nevertheless, in general, there exists some a-priori knowledge about X, where $p(x)$ is called the *a-priori distribution* or simply prior on X. Based on new evidence Y, for instance, measurements, the so-called *a-posteriori distribution* $p(x|y)$, or posterior, can be computed, which describes the new belief on X based on the observations Y. The conditional distribution $p(y|x)$ is termed *likelihood*. These three quantities fully describe the Bayesian perspective on probabilities and form Bayes rule

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{2.16}$$

$$\text{posterior} \propto \text{likelihood} \cdot \text{prior} \tag{2.17}$$

which follows directly from the product rule of probabilities.

2.2 High-Dimensional Inference Using Factor Graphs and the Sum-Product Algorithm

So far, the concepts introduced in Section 2.1.2 were restricted to two random variables X and Y. Clearly, it is possible to extend these concepts to multivariate settings involving the multivariate random variables $\mathbf{X} = [\mathsf{X}_1, \mathsf{X}_2, \dots \mathsf{X}_N]$ and $\mathbf{Y} = [\mathsf{Y}_1, \mathsf{Y}_2, \dots \mathsf{Y}_N]$. The realizations of theses multivariate random variables are the tuples $\mathbf{x} = (x_1, x_2, \dots, x_N) \in \boldsymbol{\mathcal{X}}$ and $\mathbf{y} = (y_1, y_2, \dots, y_N) \in \boldsymbol{\mathcal{Y}}$, where $\boldsymbol{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \mathcal{X}_N$ and $\boldsymbol{\mathcal{Y}} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \dots \mathcal{Y}_N$. Similar to Eq. (2.2), Eq. (2.7) and Eq. (2.9) one obtains:

General rules of probability distributions

Total law of probabilities:

$$\sum_{\mathbf{x} \in \mathbf{X}} \sum_{\mathbf{y} \in \mathbf{Y}} p(\mathbf{x}, \mathbf{y}) = 1 \tag{2.18}$$

Marginalization (sum-rule):

$$p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_N} \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) \tag{2.19}$$

General chain rule (product-rule):

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) =& p(x_1|x_2, \ldots, x_N, \mathbf{y}) p(x_2|x_3, \ldots, x_N, \mathbf{y}) \cdots p(y_1|y_2, \ldots, y_N) \\ & \cdots p(y_{N-1}|y_N) p(y_N) \end{aligned} \tag{2.20}$$

Any joint distribution can be factorized as shown in Eq. (2.20). However, in some settings, certain random variables are *independent* or *conditionally independent*, which allows simplifying Eq. (2.20). A common visual tool to display the factorization of joint distributions and reveal independencies between random variables are *factor graphs* [KFL01].

Often it is convenient to describe mathematical concepts using graphs [Bis09]. An *undirected graph* $G = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes (sometimes called vertices) $\mathcal{V}$ and a set of edges $\mathcal{E}$ connecting these nodes, where all the edges are bidirectional. In a bipartite graph, the set of nodes can be split into two disjoint subsets of nodes. Further, edges are only allowed to connect vertices from different subsets.

Factor graphs are a particular instance of bipartite graphs. The general construction of a factor graph starts with a *global* function $g(.)$, e.g., $g(x_1, x_2, x_3, x_4)$, which is decoupled into local functions $f_i(.)$, for instance as

$$g(x_1, x_2, x_3, x_4) = f_3(x_4, x_3, x_1) f_2(x_2, x_1) f_1(x_1)\,. \tag{2.21}$$

The two disjoint subsets of nodes in a factor graph are called *variable nodes* and *factor nodes*. Variable nodes are depicted by circles and drawn for each variable in the global function. Factor nodes are represented by filled squares in the factor graph and drawn for each local function. In the last step, variable nodes and factor nodes are connected through edges if a respective variable is part of a local function. Fig. 2.2 shows the factor graph of Eq. (2.21). The following example provides a more detailed description of decomposing a global function and constructing the factor graph for a particular application.

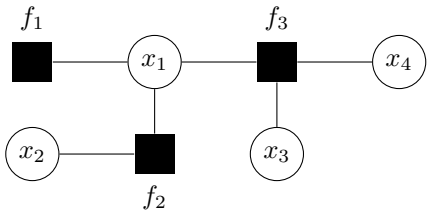


Figure 2.2: Factor graph of the decoupled global function from Eq. (2.21).

Example 2.1: Factor Graph of a Markov Chain

Let us assume the following example. During his Ph.D., a Ph.D. student travels to several IEEE conferences and other research trips. These random locations are $x'_t = \{\text{Los Angeles (LAX)}, \text{Paris (CDG)}, \text{Hamburg (HAM)}, \text{Dubai (DUB)}, \text{Shanghai (PVG)}, \text{Seoul (ICN)}\}$ defining the sample space of the random variables X'_t where $t = 1, 2, \ldots, 6$ denotes the time. Further let the random variable X_t describe the latitude associated with a respective location. The location of the student is monitored by GPS, unfortunately only noisy measurements y_t of the latitude exist, i.e., $y_t = x_t + n_t$, where n_t describes the measurement noise. Fig. 2.3 shows all possible locations x'_t and their geographical coordinates. The joint distribution $p(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = [x_1, x_2, \cdots, x_6]^\mathsf{T}$ and $\mathbf{y} = [y_1, y_2, \cdots, y_6]^\mathsf{T}$, fully defines the random experiment. Application of the general chain rule (cf. Eq. (2.20)) yields

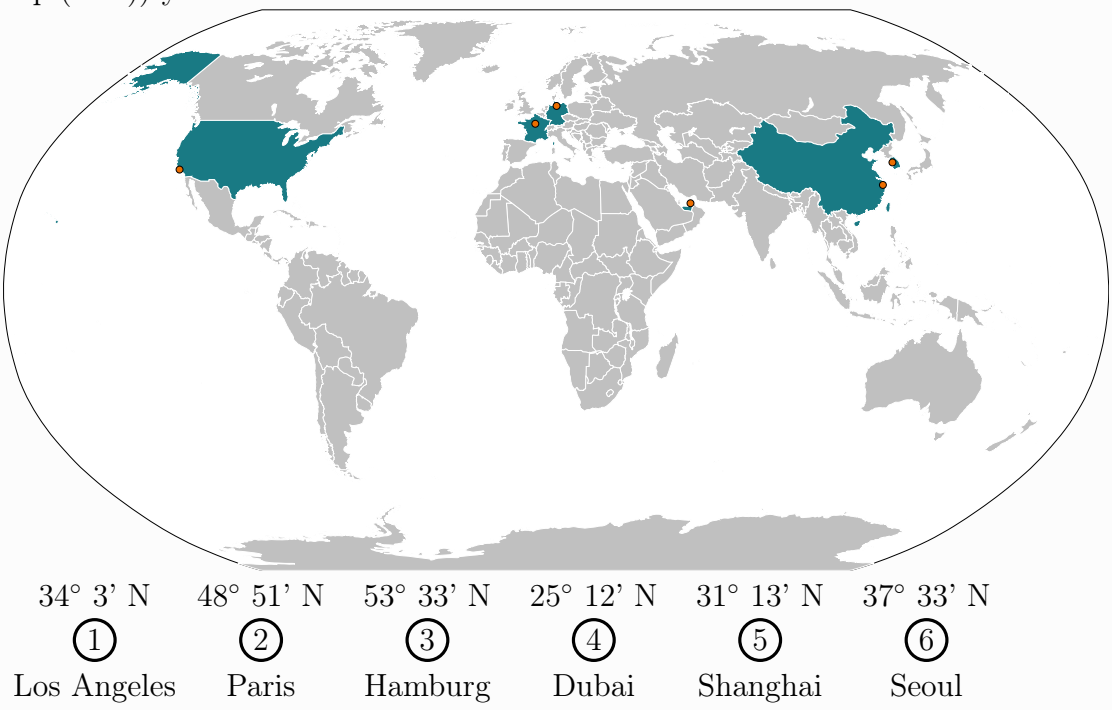


Figure 2.3: World map with six possible locations and their respective coordinates.

$$p(\mathbf{x}, \mathbf{y}) = p(x_1|x_2, \ldots, x_6, y_1, \ldots, y_6)p(x_2|x_3, \ldots, x_6, y_1, \ldots, y_6) \cdots p(y_1|y_2, \ldots, y_6) \cdots p(y_5|y_6)p(y_6)\,. \tag{2.22}$$

However, in this example, it is further assumed that the measurements at time t only depend on the location x_t. Furthermore, let us assume a first order Markov chain. Thus, the next location x_{t+1} only depends on the location x_t at time t. In turn, utilizing these conditional independencies allows to rewrite Eq. (2.22) as

$$p(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{6} p(y_t|x_t) \prod_{t=1}^{5} p(x_{t+1}|x_t)p(x_1)\,. \tag{2.23}$$

The respective factor graph is constructed as follows:

1. Draw a variable node for each variable, i.e., $x_1, x_2, \ldots, x_6, y_1, y_2, \ldots, y_6$.
2. Draw a factor node for each distribution in Eq. (2.22).
3. Draw edges connecting each variable node to a factor node if the variable is part of this factor.

and depicted in Fig. 2.4 where the prior $p(x_1)$ was intentionally excluded.

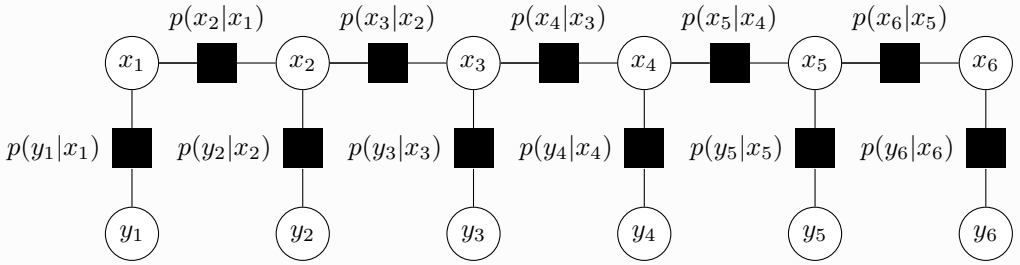


Figure 2.4: Factor graph for the distribution in Eq. (2.23).

2.2.1 Sum-Product Algorithm (Belief Propagation)

A typical inference task is to determine $\arg\max_{x_i} p(x_i|\mathbf{y}), \forall i = 1, \ldots, N$, i.e., the arg max of each marginal posterior distribution. Assuming X_i is a discrete random variable and $|\mathsf{X}_i| = |\mathsf{X}|$, a naive approach would require a marginalization over $|\mathsf{X}|^{N-1}$ terms for each i. Given a factor graph, it is possible to efficiently compute all marginals by applying the sum-product algorithm, sometimes called *belief propagation* [KFL01]. The update rules of the sum-product algorithm are reasonably simple and are different if a message or belief is passed from a variable node to a factor node $\mu_{x\to f}(x)$ or from a factor node to a variable node $\mu_{f\to x}(x)$. The variable-to-factor

message is computed as [KFL01]

$$\mu_{x\to f}(x) = \prod_{h\in N(x)\setminus f} \mu_{h\to x}(x)\,, \tag{2.24}$$

where $N(x)$ denotes the *neighborhood* of the variable node x, i.e., all factor nodes connect to x. In turn, the variable-to-factor message is the product of all incoming messages received over edges from connected factor nodes, except the message received from the factor node to which an updated belief is sent. The factor-to-variable message is computed as

$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} f(N(f)) \prod_{y\in N(f)\setminus x} \mu_{y\to f}(y) \tag{2.25}$$

where $N(f)$ denotes the neighborhood of the factor node f and $\sum_{\sim\{x\}}$ indicates that the marginalization is over all variables connected to the factor node except x. Hence, the updated belief conveyed from a factor node to a variable node is computed based on all messages, except the message from the target variable node, multiplied by the local function followed by a marginalization. The messages exchanged in the factor graph are often only proportional to valid probability distributions. Thus, typically the messages are normalized after each update. In a cycle-free factor graph, the sum-product algorithm is terminated if all variable nodes have received messages from all connected factor nodes. Then, the actual marginal distribution is proportional to the product of *all* incoming messages at the respective variable node, i.e.,

$$p(x_i) \propto \prod_{f\in N(x_i)} \mu_{f\to x}(x_i)\,. \tag{2.26}$$

The following example continues Example 2.1 and applies the sum-product algorithm to solve a trajectory-tracking task.

Example 2.2: Sum-Product Algorithm for a Hidden Markov Model

Considering Example 2.1, let us assume that the task at hand is to recover the most likely points $\hat{x}_t$ using the maximum a posteriori (MAP) estimate $\hat{x}_t = \arg\max_{x_t} p(x_t|\mathbf{y})$, $t = 1, \ldots, 6$ of the trajectory of the Ph.D. student's journey $\mathbf{x}$, given the noisy measurements $\mathbf{y} = [y_1, y_2, \cdots, y_6]^\mathsf{T}$ of the latitudes. It is assumed that the measurement noise is additive white Gaussian noise (AWGN), i.e, $y_t = x_t + n_t$, where $n_t \sim \mathcal{N}(0, \sigma_N^2)$ are independent and identically distributed (i.i.d.) samples of a zero-mean Gaussian distribution with variance σ_N^2. In this example, the variance of the measurements noise is set to $\sigma_N^2 = 9$. This is

depicted in Fig. 2.5.

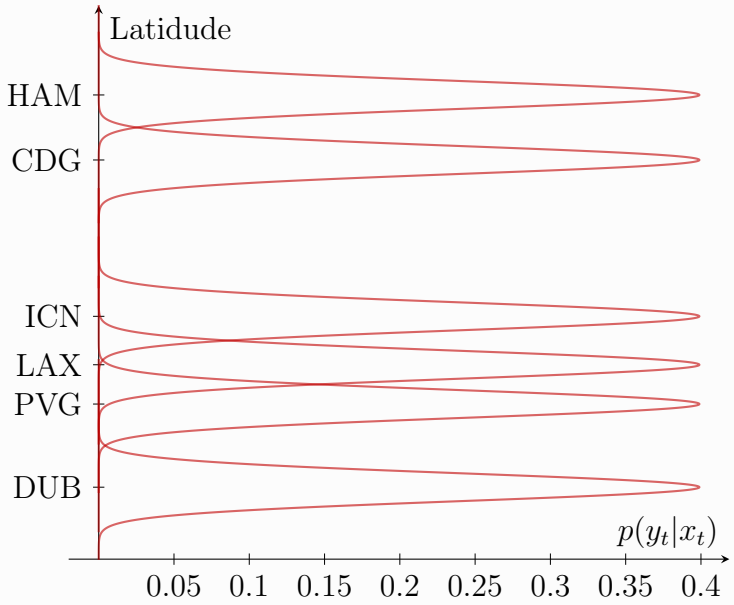


Figure 2.5: Measurement model $p(y_t|x_t)$ of the measured latitudes.

Furthermore, the transition probability $p(x_{t+1}|x_t)$, i.e., the probability of moving depends on the distance between two locations. It is modeled by an exponential distribution and it is not allowed to stay at the same location. Using the correct distances based on the coordinates from Fig. 2.3, this results in the following transition model

$$p(x_{t+1}|x_t) = \begin{array}{c} x_t \backslash x_{t+1} \\ \text{LAX} \\ \text{CDG} \\ \text{HAM} \\ \text{DUB} \\ \text{PVG} \\ \text{ICN} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{LAX} & \text{CDG} & \text{HAM} & \text{DUB} & \text{PVG} & \text{ICN} \end{array} \\ \begin{pmatrix} 0. & 0.202 & 0.203 & 0.106 & 0.165 & 0.324 \\ 0.121 & 0. & 0.423 & 0.215 & 0.118 & 0.123 \\ 0.116 & 0.406 & 0. & 0.218 & 0.127 & 0.132 \\ 0.074 & 0.251 & 0.265 & 0. & 0.210 & 0.199 \\ 0.105 & 0.125 & 0.140 & 0.191 & 0. & 0.440 \\ 0.186 & 0.118 & 0.132 & 0.163 & 0.399 & 0. \end{pmatrix} \end{array} . \tag{2.27}$$

Fig. 2.6 shows the factor graph from Example 2.1 together with arrows indicating the message passed when applying the sum-product algorithm.

The sum-product algorithm starts from the leaf nodes y_1 and y_6, resulting in a forward path and a backward path. Hence, the application of the sum-product algorithm on the considered factor graph is sometimes referred to as the forward-backward algorithm or Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm. The forward-path messages are denoted ϑ_i (cf. blue arrows) and the messages of the backward path are denoted β_i (cf. green arrows), respectively.

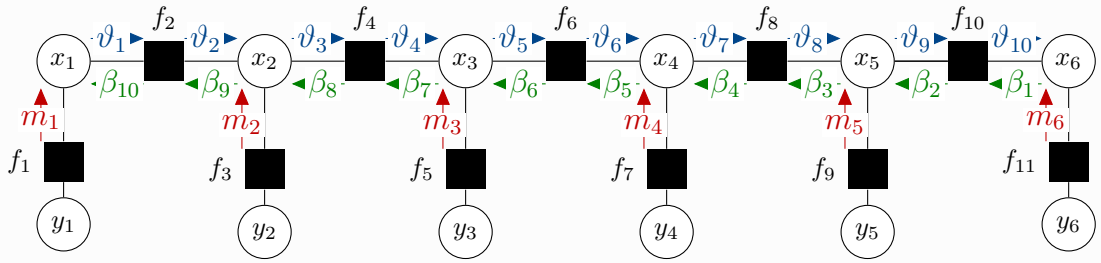


Figure 2.6: Factor graph for the distribution in Eq. (2.23).

The messages m_1, m_2, m_3, m_4, m_5, m_6 shown in red are proportional to the likelihood functions $p(y_t|x_t)$ and given as

$$m_1 = \begin{bmatrix} 0.00 \\ 0.55 \\ 0.45 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \; m_2 = \begin{bmatrix} 0.00 \\ 0.46 \\ 0.54 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \; m_3 = \begin{bmatrix} 0.13 \\ 0.00 \\ 0.00 \\ 0.38 \\ 0.48 \\ 0.01 \end{bmatrix} \; m_4 = \begin{bmatrix} 0.00 \\ 0.42 \\ 0.58 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \; m_5 = \begin{bmatrix} 0.42 \\ 0.00 \\ 0.00 \\ 0.02 \\ 0.44 \\ 0.12 \end{bmatrix} \; m_6 = \begin{bmatrix} 0.31 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.06 \\ 0.63 \end{bmatrix} \tag{2.28}$$

in the considered example.

First let us consider the forward path. The message $\vartheta_1 = \mu_{x_1 \to f_2}(x_1)$ is found by application of the variable-to-factor update rule (cf. Eq. (2.24)):

$$\mu_{x_1 \to f_2}(x_1) = \vartheta_1 = m_1 p(x_1) \propto p(x_1|y_1) \tag{2.29}$$

where it is assumed that $p(x_1)$, i.e., the a priori distribution is known and equals $p(x_1) = [0.0, 0.0, 1, 0.0, 0.0, 0.0]^\mathsf{T}$, i.e., the starting point is known to be Hamburg. Please note that the multiplication of the messages involves an element-wise multiplication followed by a normalization. In turn, $\vartheta_1 = [0.0, 0.0, 1, 0.0, 0.0, 0.0]$. The message $\vartheta_2 = \mu_{f_2 \to x_2}(x_2)$ is computed by application of the factor-to-variable rule (cf. Eq. (2.24)) as

$$\mu_{f_2 \to x_2}(x_2) = \sum_{x_1} f_2(x_1, x_2) \mu_{x_1 \to f_2}(x_1) = \vartheta_2 = \sum_{x_1} p(x_2|x_1) \vartheta_1 \tag{2.30}$$

which yields

$$\vartheta_2 = \begin{bmatrix} 0.116 \\ 0.406 \\ 0.0 \\ 0.218 \\ 0.127 \\ 0.132 \end{bmatrix}.$$

In the next step, $\vartheta_3 = \mu_{x_2 \to f_4}(x_2)$ is computed by application of the variable-to-factor rule as

$$\mu_{x_2 \to f_4}(x_2) = \mu_{f_3 \to x_2}(x_2)\mu_{f_2 \to x_2}(x_2) = \vartheta_3 = m_2\vartheta_2 \propto p(x_2|y_1, y_2). \tag{2.31}$$

It can be observed that the forward messages $\vartheta_{2t-1} = \mu_{x_t \to f_{2t}}(x_t)$ are always proportional to $p(x_t|y_1, \ldots, y_t)$, i.e., the information about x_t gained by the combination of all previous measurements. For the considered example, the messages ϑ_{2t-1} are found as

$$\vartheta_1 = \begin{bmatrix} 0.00 \\ 0.00 \\ 1.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \vartheta_3 = \begin{bmatrix} 0.00 \\ 1.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \vartheta_5 = \begin{bmatrix} 0.10 \\ 0.00 \\ 0.00 \\ 0.52 \\ 0.37 \\ 0.01 \end{bmatrix} \vartheta_7 = \begin{bmatrix} 0.00 \\ 0.41 \\ 0.59 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \vartheta_9 = \begin{bmatrix} 0.40 \\ 0.00 \\ 0.00 \\ 0.04 \\ 0.43 \\ 0.12 \end{bmatrix}. \tag{2.32}$$

Now, let us consider the backward path. The message $\beta_1 = \mu_{x_6 \to f_{10}}(x_6)$ can be found using the variable-to-factor update rule. However, it is assumed that the final location is unknown. Thus, $p(x_6) = \left[\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right]^\mathsf{T}$. In turn,

$$\mu_{x_6 \to f_{10}}(x_6) = \mu_{f_{11} \to x_6}(x_6)p(x_6) = \beta_1 \propto p(x_6|y_6)\,. \tag{2.33}$$

Technically, the message computations for the backward path are similar to the forward path, i.e, the messages $\beta_{2(N+1-t)-1} = \mu_{x_t \to f_{2(t-1)}}(x_t)$ are found using the variable-to-factor rule and the messages $\beta_{2(N-t)} = \mu_{f_{2t} \to x_t}(x_t)$ are computed based on the factor-to-variable update rule. Again it can be observed that the messages $\beta_{2(N-t)} = \mu_{f_{2t} \to x_t}(x_t)$ are proportional to $p(x_t|y_{t+1}, \ldots, y_N)$ and the messages $\beta_{2(N+1-t)-1} = \mu_{x_t \to f_{2(t-1)}}(x_t)$ are proportional to $p(x_t|y_t, \ldots, y_N)$. Given the

measurements from Eq. (2.28), one obtains

$$\beta_1 = \begin{bmatrix} 0.31 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.06 \\ 0.63 \end{bmatrix} \beta_3 = \begin{bmatrix} 0.26 \\ 0.00 \\ 0.00 \\ 0.02 \\ 0.65 \\ 0.07 \end{bmatrix} \beta_5 = \begin{bmatrix} 0.00 \\ 0.41 \\ 0.59 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \beta_7 = \begin{bmatrix} 0.10 \\ 0.00 \\ 0.00 \\ 0.52 \\ 0.38 \\ 0.01 \end{bmatrix} \beta_9 = \begin{bmatrix} 0.00 \\ 0.45 \\ 0.55 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix}. \tag{2.34}$$

The target marginal distribution $p(x_t|\mathbf{y})$ can be found by multiplication of a forward message $\mu_{x_t \to f_{2t}}(x_t)$ and a backward message $\mu_{f_{2t} \to x_t}(x_t)$, that is,

$$p(x_t|\mathbf{y}) \propto \mu_{x_t \to f_{2t}}(x_t) \cdot \mu_{f_{2t} \to x_t}(x_t) = \vartheta_{2t-1}\beta_{2(N-t)} \tag{2.35}$$

with $N = 6$ in the considered example for random measurements $\mathbf{y}$.
Application of the arg max operator yields the estimated trajectory. This trajectory is compared to the true trajectory, the maximum likelihood decisions on the measurements only, and the maximum likelihood decision on the backward path and the forward path individually. Detection errors are highlighted in dark red.

$$\text{True Path} : [\text{HAM}, \text{CDG}, \text{DUB}, \text{HAM}, \text{PVG}, \text{ICN}]^\mathsf{T} \tag{2.36}$$

$$\arg\max_{x_t} p(y_t|x_t) : [\text{CDG}, \text{HAM}, \text{PVG}, \text{HAM}, \text{PVG}, \text{ICN}]^\mathsf{T} \tag{2.37}$$

$$\text{Forward Path} : [\text{HAM}, \text{CDG}, \text{DUB}, \text{HAM}, \text{PVG}, \text{ICN}]^\mathsf{T} \tag{2.38}$$

$$\text{Backward Path} : [\text{CDG}, \text{HAM}, \text{DUB}, \text{HAM}, \text{PVG}, \text{ICN}]^\mathsf{T} \tag{2.39}$$

$$\arg\max_{x_t} p(x_t|\mathbf{y}) : [\text{HAM}, \text{CDG}, \text{DUB}, \text{HAM}, \text{PVG}, \text{ICN}]^\mathsf{T} \tag{2.40}$$

It can be observed that the maximum likelihood decision on the individual measurements performs worst, as three locations are wrongly detected. In contrast, the MAP solution derived using the sum-product algorithm detects the trajectory error-free. It is interesting to observe that already in the forward path no errors occur as the prior knowledge about the starting point helps to resolve the first locations. In contrast, the backward path has no prior knowledge about the first locations causing an error. The provided examples 2.1 and 2.2 were used to illustrate the concepts of factor graphs and the sum-product algorithm. The considered scenario will also serve as a running example in the next chapters.

2.3 Fundamentals of Information Theory

In addition to statistics and decision theory, many concepts of the broad field of information theory will be used throughout this thesis. This section reviews the most important concepts to quantify information properly. For a more detailed introduction to information theory, the interested reader is referred to [Cov06]. This section considers only discrete random variables. However, all present mathematical concepts are also applicable to continuous random variables.

2.3.1 Entropy, Joint and Conditional Entropy

As defined in Section 2.1, a random variable X is associated with a sample space $\mathcal{X}$ and each element of the sample space, i.e., the event $x \in \mathcal{X}$ appears with a certain probability summarized in the probability mass function $p(x)$. The entropy $\mathrm{H}(\mathsf{X})$ is a measure of uncertainty of the random variable X and defined as

$$\mathrm{H}(\mathsf{X}) = -\sum_{x \in \mathcal{X}} p(x) \log_2 (p(x)) \ . \tag{2.41}$$

The entropy is sometimes referred to as *expected self-information*, as the so-called self-information $\mathrm{i}(x)$ of event x is defined as

$$\mathrm{i}(x) = \log_2 \left(\frac{1}{p(x)} \right) = -\log_2 (p(x)) \ . \tag{2.42}$$

The entropy is bounded by [Cov06]

$$0 \leq \mathrm{H}(\mathsf{X}) \leq \log_2 (|\mathsf{X}|) \ . \tag{2.43}$$

If the logarithm is to the base 2, the entropy and the self-information are measured in *bits*. Please note that the entropy of a binary random variable X with equally likely outcome, i.e., $\Pr(\mathsf{X} = 0) = \Pr(\mathsf{X} = 1) = 0.5$ is exactly one bit as

$$\mathrm{H}(\mathsf{X}) = -\sum_{x \in \mathcal{X}} p(x) \log_2 (p(x)) = -0.5 \cdot \log_2 (0.5) + (-0.5 \cdot \log_2 (0.5)) = 1 \, . \tag{2.44}$$

If the base of the logarithm is e, the entropy is measured in *nats*.

The entropy can easily be computed for more than one random variable. First, let us consider the pair of random variables X and Y. Then, the *joint entropy* of this

pair of random variables is defined as

$$\mathrm{H}(\mathsf{X},\mathsf{Y}) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2\left(p(x,y)\right) \tag{2.45}$$

and likewise the *conditional entropy* is defined as

$$\mathrm{H}(\mathsf{Y}|\mathsf{X}) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2\left(p(y|x)\right)\ . \tag{2.46}$$

Combining conditional and joint entropy allows to derive the chain rule for entropy given the random variables $\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_N$ as

$$\mathrm{H}(\mathsf{X}_1,\mathsf{X}_2,\ldots,\mathsf{X}_N) = \sum_i \mathrm{H}(\mathsf{X}_i|\mathsf{X}_{i-1},\ldots,\mathsf{X}_1)\ . \tag{2.47}$$

2.3.2 Relative Entropy and Mutual Information

Based on the fundamental ideas of information theory, i.e., entropy and self-information, more enhanced concepts which are inevitable for this thesis shall be derived. Throughout this thesis, compression and clustering techniques and machine learning algorithms will be applied to derive coarsely quantized mutual-information-based signal processing units.

To determine meaningful clusters, quantifying the similarity of probability distributions is crucial for any clustering approach. Furthermore, as this thesis also focuses on machine learning algorithms exploiting sample-based training and learning, measures to evaluate how well a learned distribution $q(x)$ approximates an *unknown* distribution $p(x)$ are integral.

Here, the so-called *relative entropy* or *Kullback-Leibler divergence* $D_{\mathrm{KL}}\{p(x)||q(x)\}$ is a very useful concept defined as [Cov06]

$$D_{\mathrm{KL}}\{p(x)||q(x)\} = \sum_{x\in\mathcal{X}} p(x)\log_2\left(\frac{p(x)}{q(x)}\right)\ . \tag{2.48}$$

Given the probability distributions $p(x)$ and $q(x)$, $D_{\mathrm{KL}}\{p(x)||q(x)\} \geq 0$ which holds with equality if and only if $p(x) = q(x)$ [Cov06].

Although the Kullback-Leibler divergence is not symmetric in its argument, it is often useful to think of the Kullback-Leibler divergence as a distance measure [Cov06; Ama16].

One symmetric generalization of the Kullback-Leibler divergence is the *Jensen-Shannon divergence* $D_{\mathrm{JS}}^{\Pi}\{p(x)||q(x)\}$ defined as

$$D_{\mathrm{JS}}^{\Pi}\{p(x)||q(x)\} = \pi_1 D_{\mathrm{KL}}\{p(x)||\bar{p}(x)\} + \pi_2 D_{\mathrm{KL}}\{q(x)||\bar{p}(x)\} \tag{2.49}$$

where $\pi_1 + \pi_2 = 1$, $0 < \pi_1, \pi_2 < 1$ and $\bar{p}(x) = \pi_1 p(x) + \pi_2 q(x)$ [Slo02; Cov06]. Chapter 3 exploits the Jensen-Shannon divergence to derive cost functions for information bottleneck algorithms.

Closely related to the entropy and the Kullback-Leibler divergence, the most crucial information-theoretical measure for this thesis is Shannon's *mutual information*. In his landmark paper [Sha48], Shannon tried to answer the question

> *"Given two random variables X and Y, what is the amount of uncertainty about X removed by knowing Y and vice versa."*

This leads directly to the definition of the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ of X and Y as

$$\begin{aligned}
\mathrm{I}(\mathsf{X};\mathsf{Y}) &= \mathrm{H}(\mathsf{X}) - \mathrm{H}(\mathsf{X}|\mathsf{Y}) && (2.50)\\
&= \mathrm{H}(\mathsf{Y}) - \mathrm{H}(\mathsf{Y}|\mathsf{X}) && (2.51)\\
&= \sum_x \sum_y p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) && (2.52)\\
&= D_{\mathrm{KL}}\{p(x,y)||p(x)p(y)\} && (2.53)
\end{aligned}$$

where the entropy $\mathrm{H}(\mathsf{X})$, respectively $\mathrm{H}(\mathsf{Y})$, serves as a measure of uncertainty and the conditional entropy $\mathrm{H}(\mathsf{X}|\mathsf{Y})$ indicates the remaining uncertainty about X by knowing the other observed variable Y [Cov06].

Following [Cov06], the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is bounded by

$$0 \leq \mathrm{I}(\mathsf{X};\mathsf{Y}) \leq \min(\mathrm{H}(\mathsf{X}), \mathrm{H}(\mathsf{Y})) \tag{2.54}$$

where $\mathrm{I}(\mathsf{X};\mathsf{Y}) = 0$ if X and Y are independent.

Interestingly, Eq. (2.52) states that given two independent random variables X and Y, observing Y provides no knowledge about X.

The concept of mutual information can also be extended to more than two random variables, e.g., X and $\mathsf{Y}_1, \ldots, \mathsf{Y}_N$. In this case, the mutual information can be

rewritten using the chain rule for mutual information as

$$\mathrm{I}(\mathsf{X};\mathsf{Y}_1,\ldots,\mathsf{Y}_N)=\sum_{i=1}^{N}\mathrm{I}(\mathsf{Y}_i;\mathsf{X}|\mathsf{Y}_1,\ldots,\mathsf{Y}_{i-1}) \tag{2.55}$$

where $\mathrm{I}(\mathsf{Y}_i;\mathsf{X}|\mathsf{Y}_1\ldots,\mathsf{Y}_{i-1})$ denotes the *conditional mutual information*

$$\mathrm{I}(\mathsf{Y}_i;\mathsf{X}|\mathsf{Y}_1,\ldots,\mathsf{Y}_{i-1})=\sum_{x}\sum_{y_1}\cdots\sum_{y_N}p(x,y_i|y_1,\ldots,y_{i-1})\cdot \log_2\left(\frac{p(x,y_i|y_1,\ldots,y_{i-1})}{p(x|y_1,\ldots,y_{i-1})p(y_i|y_1,\ldots,y_{i-1})}\right). \tag{2.56}$$

In the special case that random variables form a Markov chain [Cov06], e.g.,

$$\mathsf{X}\leftrightarrow\mathsf{Y}\leftrightarrow\mathsf{T} \tag{2.57}$$

the joint distribution $p(x,y,t)$ factorizes as

$$p(x,y,t)=p(t|y)p(y|x)p(x), \tag{2.58}$$

i.e., the conditional distribution of T depends only on Y and is conditionally independent of X. Interestingly, it can be shown that no processing of Y can increase the information of Y about X, which is formalized by the data processing inequality [Cov06]. The date processing inequality states that given the Markov chain $\mathsf{X}\leftrightarrow\mathsf{Y}\leftrightarrow\mathsf{T}$ of the random variables X, Y and T

$$\mathrm{I}(\mathsf{X};\mathsf{Y})\geq\mathrm{I}(\mathsf{X};\mathsf{T})\,. \tag{2.59}$$

The data processing inequality will be essential to derive bounds on the information bottleneck setup derived in Chapter 3, as the random variables defining the information bottleneck form a Markov chain.

2.4 Channel Coding Theorem and Linear Block Codes

Based on the definition of mutual information, in his landmark paper [Sha48], Shannon derived under which conditions an error-free transmission over a noisy channel is possible. This requires advanced channel coding schemes which will be of great interest in Chapter 5 and Chapter 6.

Figure 2.7: Illustration of a transmission chain between a source and a sink.

Fig. 2.7 depicts a transmission chain for the special case of block-wise transmission, where

Uncoded message u: $\mathbf{u} = [u_1, u_2, \ldots, u_K]^\mathsf{T} \in \mathbb{F}_q^K$ denotes the uncoded message. In this thesis, the elements u_i, $i = 1, \ldots, K$ are taken from a finite field $\mathbb{F}_q$ (cf. Section 2.4.1),

Coded message c: The uncoded message is fed into the encoder. The output of the encoder is the coded message $\mathbf{c} = [c_1, c_2, \ldots, c_N]^\mathsf{T} \in \mathbb{F}_q^N$.

Transmit message x: The coded message is mapped onto complex transmit symbols $x_i \in \mathbb{C}$ by the mapper. All transmit symbols are summarized in the vector $\mathbf{x} = [x_1, x_2, \ldots, x_M]^\mathsf{T} \in \mathbb{C}^M$.

Receive message y: The transmit symbols are sent over a discrete channel defined by the transition probability $p(\mathbf{y}|\mathbf{x})$. In this thesis, most channels are discrete memoryless channels [Cov06], i.e.,

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{M} p(y_i|x_i)\,. \tag{2.60}$$

The outputs of the channel are the received symbols $y_i \in \mathbb{C}$ pooled in $\mathbf{y} = [y_1, y_2, \ldots, y_M]^\mathsf{T} \in \mathbb{C}^M$.

Decoded message û: The decoded message $\hat{\mathbf{u}} \in \mathbb{F}_q^K$ is the output of the decoder which was fed by the demapped received symbols.

Clearly, the aim of any communication system is to successfully reconstruct the uncoded message $\mathbf{u}$ at the output of the transmission chain $\hat{\mathbf{u}}$ with a negligible probability of error. A valuable quantity to determine the fundamental limits of error-free communication is Shannon's *channel capacity*. The channel capacity C for a discrete memory-less channel $p(y|x)$ is defined as [Cov06]

$$C = \max_{p(x)} \mathrm{I}(\mathsf{X};\mathsf{Y}) \tag{2.61}$$

where the maximum is taken over all possible input distributions $p(x)$. The channel capacity determines the so-called *maximum transmission rate*, measured in

bits / channel use, at which error-free transmission over the channel $p(y|x)$ is possible. This leads to Shannon's channel coding theorem stated in Theorem 1.

Theorem 1 (Channel coding theorem)**.** *For a discrete memory-less channel, all rates below the channel capacity C are achievable. Specifically, for every rate $R < C$, there exists a channel code with rate R and code word length N with a maximum error probability $p_e \to 0$ for $N \to \infty$.*

Proof. The proof of the channel coding theorem can be found in great detail in [Cov06, Section 7.7] □

The definitions of rates and achievable rates and characteristics of channel codes used in Theorem 1, will be clarified in Section 2.4.2. Beforehand, the concept of finite fields is reviewed.

2.4.1 Finite Fields

In general, the sample space of the information symbol u_i could be any discrete space $\mathcal{U}$. However, considering channel coding applications, a set of elements $\mathcal{U}$ with a well defined algebraic structure is beneficial as it allows to perform arithmetic computation with the elements of the set. Such algebraic structures are, e.g., *groups* and *finite fields* [LC01]. Let u_1, u_2, u_3 be elements of the set $\mathcal{U}$. Furthermore, $+$ and $\cdot$ denote operations [LC01].

The algebraic structure $(\mathcal{U}, +)$ forms a *group* if the following properties hold [LC01]:

1. Closure under $+$, i.e., $u_1 + u_2 \in \mathcal{U}, \forall u_1, u_2 \in \mathcal{U}$
2. Associativity, i.e., $u_1 + (u_2 + u_3) = (u_1 + u_2) + u_3, \forall u_1, u_2, u_3 \in \mathcal{U}$
3. Additive identity, i.e., $\exists 0 \in \mathcal{U} : 0 + u_1 = u_1, \forall u_1 \in \mathcal{U}$
4. Invertibility, i.e., $\exists(-u_1) \in \mathcal{U} : u_1 + (-u_1) = 0, \forall u_1 \in \mathcal{U}$

The algebraic structure $(\mathcal{U}, +)$ forms an *Abelian group* if the following additional property holds [LC01]:

5. Commutativity, $u_1 + u_2 = u_2 + u_1 \in \mathcal{U}, \forall u_1, u_2 \in \mathcal{U}$

Furthermore, the algebraic structure $(\mathcal{U}, +, \cdot)$ forms a *field* if also

6. Associativity under multiplication, i.e., $u_1 \cdot (u_2 \cdot u_3) = (u_1 \cdot u_2) \cdot u_3, \forall u_1, u_2, u_3 \in \mathcal{U}$
7. Multiplicative identity, i.e., $\exists 1 \in \mathcal{U} : 1 \cdot u_1 = u_1, \forall u_1 \in \mathcal{U}$

8. Distributive property, i.e., $u_1 \cdot (u_2 + u_3) = u_1 \cdot u_2 + u_1 \cdot u_3, \forall u_1, u_2, u_3 \in \mathcal{U}$
9. Multiplicative inverse, i.e., $\exists (u_1^{-1}) \in \mathcal{U} : u_1 \cdot u_1^{-1} = 1, \forall u_1 \in \mathcal{U} \backslash \{0\}$,

hold [LC01].

Fields with a finite number q of elements are called *finite fields* $\mathbb{F}_q$ or *Galois fields* of order q, i.e., GF(q). In the special case that the field order q can be written as p^m, where p is a prime, the Galois field GF(p^m) is an *extension field*. These extension fields will be integral in Chapter 6 where non-binary LDPC codes and their decoding are presented. This thesis mostly resorts to Galois fields which are extension fields with prime 2, i.e., GF(2^m).

Finite fields are constructed using so-called *primitive polynomials* $f(x)$. The polynomial $f(x)$ of degree m is said to be primitive if it is irreducible, i.e., it is not divisible by any polynomial over GF(2^m) of degree less than m but greater than zero, and the smallest positive integer n for which $f(x)$ divides $x^n - 1$ is $n = p^m - 1$ [LC01; CJ08].

The 2^m elements of the GF(2^m) are given as $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2}\} = \mathbb{F}_{2^m}$, where α is called *primitive element*. The following example describes the construction of the Galois fields in more detail.

Example 2.3: Construction of Galois fields GF(2^m)

Galois Field GF(2)

First, let us consider the Galois field GF(2). As the field order is a prime, one finds $\mathbb{F}_2$ directly as $\mathbb{F}_2 = \{0, 1\}$.

Galois Field GF(4)

For higher order Galois fields we require a primitive polynomial $f(x)$. For the Galois field GF(4), one possible primitive polynomial is $f(x) = x^2 + x^1 + 1$ [CJ08]. Furthermore, $\alpha^2 = \alpha + 1$ is a root of the polynomial, which yields

Primitive Element	Sum Representation	Binary Vector Representation
0	0	00
1	1	01
α	α	10
α^2	$\alpha + 1$	11

.

More details about the sum representation and the respective binary vector representation can be found in [CJ08]. Please note that for the GF(4) the binary vector representation of the field elements corresponds to the binary representation of the

decimal element index. However, in general, this is not the case as shown for the GF(8).

Galois Field GF(8)

For the Galois field with 8 elements, i.e., GF(8) = GF(2^3) one possible primitive polynomial is $f(x) = x^3 + x + 1$ [CJ08]. Furthermore, one finds $\alpha^3 = \alpha + 1$ as root of $f(x)$. With this observation we obtain the following table:

Primitive Element	Sum Representation	Binary Vector Representation
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	$\alpha + 1$	011
α^4	$\alpha^2 + \alpha$	110
α^5	$\alpha^3 + \alpha^2 = \alpha + 1 + \alpha^2$	111
α^6	$\alpha^4 + \alpha^3 = \alpha^2 + 1$	101

To derive the addition and multiplication rules in the respective GF(2^m) one uses the vector representation of the field elements or the exponent representation of the field elements respectively.

Addition The addition in GF(2^m) is simply the element-wise binary addition, i.e., the xor-operation, of the vector representations of the elements.

Multiplication The multiplication is determined by addition of the exponents of the primitive elements followed by a modulo-m operation. For example, for m=4, $\alpha^6 \cdot \alpha^{13} = \alpha^{(6+13) \mod 2^4-1} = \alpha^{(6+13) \mod 16-1} = \alpha^{19 \mod 15} = \alpha^4$.

Example 2.4: Arithmetic in Galois fields GF(2^m)

Galois Field GF(2)

First, let us reconsider the Galois field GF(2). Here, the addition and multiplication can be found as

Table 2.1: Addition GF(2)

c_i \ c_j	0	1
0	0	1
1	1	0

Table 2.2: Multiplication GF(2)

c_i \ c_j	0	1
0	0	0
1	0	1

Galois Field GF(4)

In the Galois field GF(4) the addition and multiplication becomes

Table 2.3: Addition GF(4)

c_i \ c_j	0	1	α	α^2
0	0	1	α	α^2
1	1	0	α^2	α
α	α	α^2	0	1
α^2	α^2	α	1	0

Table 2.4: Multiplication GF(4)

c_i \ c_j	0	1	α	α^2
0	0	0	0	0
1	0	1	α	α^2
α	0	α	α^2	1
α^2	0	α^2	1	α

Galois Field GF(8)

Finally, given the binary representations of the primitive elements and the primitive elements itself from Example 2.3 yields

Table 2.5: Addition GF(8)

c_i \ c_j	0	1	α	α^2	α^3	α^4	α^5	α^6
0	0	1	α	α^2	α^3	α^4	α^5	α^6
1	1	0	α^3	α^6	α	α^5	α^4	α^2
α	α	α^3	0	α^4	1	α^2	α^6	α^5
α^2	α^2	α^6	α^4	0	α^5	α	α^3	1
α^3	α^3	α	1	α^5	0	α^6	α^2	α^4
α^4	α^4	α^5	α^2	α	α^6	0	1	α^3
α^5	α^5	α^4	α^6	α^3	α^2	1	0	α
α^6	α^6	α^2	α^5	1	α^4	α^3	α	0

Table 2.6: Multiplication GF(8)

c_i \ c_j	0	1	α	α^2	α^3	α^4	α^5	α^6
0	0	0	0	0	0	0	0	0
1	0	1	α	α^2	α^3	α^4	α^5	α^6
α	0	α	α^2	α^3	α^4	α^5	α^6	1
α^2	0	α^2	α^3	α^4	α^5	α^6	1	α
α^3	0	α^3	α^4	α^5	α^6	1	α	α^2
α^4	0	α^4	α^5	α^6	1	α	α^2	α^3
α^5	0	α^5	α^6	1	α	α^2	α^3	α^4
α^6	0	α^6	1	α	α^2	α^3	α^4	α^5

2.4.2 Linear Block Codes

Linear block codes are a special class of block codes and used throughout this thesis. As the name implies, block codes partition the information sequence into blocks of information $\mathbf{u} \in \mathbb{F}_q^K$ with defined length K (cf. Fig. 2.7). The encoder maps this information block onto a codeword $\mathbf{c} \in \mathbb{F}_q^N$ consisting of N code symbols. Another essential property of the code is the code rate R_c defined as

$$R_c = \frac{\log_2\left(q^K\right)}{N} = \frac{K}{N}\log_2\left(q\right) . \tag{2.62}$$

The (K, N) block code $\mathcal{C}$ is called a *linear* block code if the $|\mathcal{C}| = q^K$ codewords form a K-dimensional subspace of $\mathbb{F}_q^N$, i.e., the addition of two codewords is again a valid codeword $\mathbf{c}_1 + \mathbf{c}_2 = \mathbf{c} \in \mathcal{C}, \forall \mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$. The code $\mathcal{C}$ is defined by the set of codewords, the so-called *codebook*. These codewords, however, can be obtained by different encoders. For block codes, it is convenient to resort to a vector-matrix description of the encoding process. Let $\mathbf{G} \in \mathbb{F}_q^{N\times K}$ denote the *generator matrix* of a respective encoder for code $\mathcal{C}$. The generator matrix consists of K independent column vectors $\mathbf{g}_i, i = 1, 2, \ldots, K$, which span the K-dimensional subspace of $\mathbb{F}_q^N$. In turn, the codewords are computed as [LC01]

$$\mathcal{C} = \left\{\mathbf{c} \in \mathbb{F}_q^N : \mathbf{c} = \mathbf{G}\mathbf{u}, \mathbf{u} \in \mathbb{F}_q^K\right\} . \tag{2.63}$$

Alternatively, the code can be defined using the parity check matrix $\mathbf{H} \in \mathbb{F}_q^{N\times(N-K)}$, which summarizes the $(N - K)$ parity check equations used to compute the parity

bits which carry redundant information. In general, the syndrome $\mathbf{s}$ has to be the all-zeros vector 0, if $\mathbf{c}$ is a valid codeword, i.e.,

$$\mathbf{s} = \mathbf{H}^{\mathsf{T}}\mathbf{c} = 0, \forall \mathbf{c} \in \mathcal{C}\,. \tag{2.64}$$

Hence,

$$\mathcal{C} = \left\{\mathbf{c} \in \mathbb{F}_q^N : \mathbf{H}^{\mathsf{T}}\mathbf{c} = 0\right\}\,. \tag{2.65}$$

As elementary operations on the columns of the generator matrix, i.e., permutations, scaling with non-zero weights, and addition do not change the code but only the encoder, it is possible to construct encoders with preferable properties. In particular, so-called *systematic encoders* are often used in practice [LC01]. In general, systematic generator matrices can be derived for any generator matrix $\mathbf{G}$ using Gaussian elimination [LC01]. Systematic encoders ensure that the information symbols $\mathbf{u}$ are an explicit part of the codeword $\mathbf{c}$. This is achieved by constructing a generator matrix which can be split in a $K \times K$ identity matrix $\mathbf{I}$ and a $(N-K) \times K$ parity matrix $\mathbf{P}$, i.e.,

$$\mathbf{G}_{\text{sys}} = \begin{bmatrix}\mathbf{I}\\ \mathbf{P}\end{bmatrix}\,. \tag{2.66}$$

Clearly, combining Eq. (2.63) and Eq. (2.66) yields

$$\mathbf{c} = \begin{bmatrix}\mathbf{u}\\ \mathbf{p}\end{bmatrix} = \begin{bmatrix}\mathbf{I}\\ \mathbf{P}\end{bmatrix}\mathbf{u} = \mathbf{G}_{\text{sys}}\mathbf{u}\,, \tag{2.67}$$

where $\mathbf{u}$ is an explicit part of the codeword $\mathbf{c}$.

2.4.3 Decoding Metrics, Error Rates and Achievable Rates

The last block in the transmission chain depicted in Fig. 2.7 is the decoder. For a given channel code $\mathcal{C}$ with generator matrix $\mathbf{G}$, which has to be known to the decoder, the decoder aims to recover $\hat{\mathbf{u}}$ based on the noisy received samples $\mathbf{y}$. In this thesis, the focus is solely on maximum a posteriori (MAP) decoders, which compute the most likely information sequence $\mathbf{u}$ based on the MAP decision criterion, i.e.,

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} p(\mathbf{u}|\mathbf{y}) \tag{2.68}$$

$$= \arg\max_{\mathbf{u}\to\mathbf{x}} p(\mathbf{x}|\mathbf{y}) \tag{2.69}$$

Furthermore, in this thesis, only soft decoding is considered, where soft-information, i.e., reliability about the received sample is fed into the decoder.

As we will see later in Chapter 5 and Chapter 6, the application of Eq. (2.69) is straightforward if the decoder works directly on the mapped symbols $x \in \mathcal{X}$. This is possible if $|\mathcal{X}| = |\mathbb{F}_q|$, i.e., the number of distinct symbols x is equivalent to the field order of the Galois field of the information symbols. In this case, *symbol-metric decoding* is performed [LC01]. Please note that if $|\mathcal{X}| > 2$, this might require *non-binary* codes, as discussed in Chapter 6.

Otherwise, the so-called *bit-metric decoding* is commonly employed if binary codes are paired with higher-order modulation schemes, i.e., $u_i, c_i \in \mathbb{F}_2$ and $|\mathcal{X}| > 2$. Here, first demapping from the symbols onto a binary representation denoted by $p(x|[c_1, \ldots, c_M])$ is required, where $M = \log_2(|\mathcal{X}|)$. This resorts to a marginalization over all transmit symbols except c_i, i.e.,

$$p(y|c_i) = \sum_{x} \sum_{\sim c_i} p(y|x)p(x|c_1, \ldots, c_i, \ldots, c_M) \tag{2.70}$$

and for discrete memory-less channels Eq. (2.69) becomes

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} p(\mathbf{u}|\mathbf{y}) \tag{2.71}$$

$$= \arg\max_{\mathbf{u}\to\mathbf{c}} \prod_{j=1}^{N} p(y_j|x_j)p(x_j|\mathbf{c}_j) \tag{2.72}$$

where $\mathbf{c}_j$ denotes the code bits mapped onto transmit symbol x_j. Bit metric decoding and demapping of higher-order modulations will be revisited in Section 4.2.3 when respective mutual-information-based signal processing units are derived. Furthermore, in Chapter 7 entire mutual-information maximizing transmission chains using autoencoders and deep learning are derived. It will be shown that for symbol-metric decoding and bit-metric decoding, the loss-function has to be chosen accordingly to achieve optimum end-to-end performance.

The choice of symbol-metric or bit-metric decoding also impacts the maximum achievable rate of the transmission chain. For symbol-metric decoding, the maximum achievable rate is determined using Shannon's channel coding theorem (cf. Theorem 1), i.e.,

$$R = \mathrm{I}(\mathsf{X};\mathsf{Y})\,. \tag{2.73}$$

However, for bit-metric decoding, the maximum achievable rate is

$$R_{\mathrm{BICM}} = \sum_{k=1}^{M} \mathrm{I}(\mathsf{C}_k;\mathsf{Y}) \tag{2.74}$$

known as bit-interleaved coded modulation (BICM) capacity derived in [CTB98].

In this thesis, the actual performance of a decoder is evaluated either using the bit error rate (BER) respectively symbol error rate (SER) defined as

$$\mathrm{BER} = \frac{1}{K} \sum_{i=1}^{K \log_2(q)} \Pr(\hat{\mathsf{B}}_i \neq \mathsf{B}_i) \tag{2.75}$$

$$\mathrm{SER} = \frac{1}{K} \sum_{i=1}^{K} \Pr(\hat{\mathsf{U}}_i \neq \mathsf{U}_i) \tag{2.76}$$

where B denotes a binary random variable and U can be any discrete random variable with $|\mathcal{U}| = q$. Alternatively, the frame error rate (FER) defined as

$$\mathrm{FER} = \Pr(\hat{\mathsf{U}} \neq \mathsf{U}) \,. \tag{2.77}$$

is considered.

Chapter 3

The Information Bottleneck Method

This chapter serves as a gentle introduction to the information bottleneck method. The information bottleneck method is a general information theoretical framework with roots in machine learning and information theory. Starting with a historical overview, this chapter aims to provide insights into which ideas and research directions in various academic societies existed and how they relate to the information bottleneck method. Furthermore, it will be sketched how these streams and ideas merged into what was recently referred to as *mutual-information-based signal processing*. A more mathematical description of the information bottleneck method itself is deferred to the second part of this chapter. This second part also reviews another framework in more detail, i.e., rate-distortion theory, which is very closely related to the information bottleneck method. Thereafter, different information bottleneck algorithms and their extensions developed in this thesis are described.

3.1 The Information Bottleneck Method and its Notions

When referring to coding in an information-theoretical sense, at least two notions exist, i.e., source coding and channel coding. Shannon's separation theorem, as described in [Cov06], directly leads to the conclusion that source coding and channel coding should be treated independently. At the same time, however, it is of crucial importance to remember Shannon's following observation [Sha59]:

> *"There is a curious and provocative duality between the properties of a source with a distortion measure and those of a channel [...] if we consider channels in which there is a "cost" associated with the different input letter. [...] This problem amounts, mathematically, to maximizing*

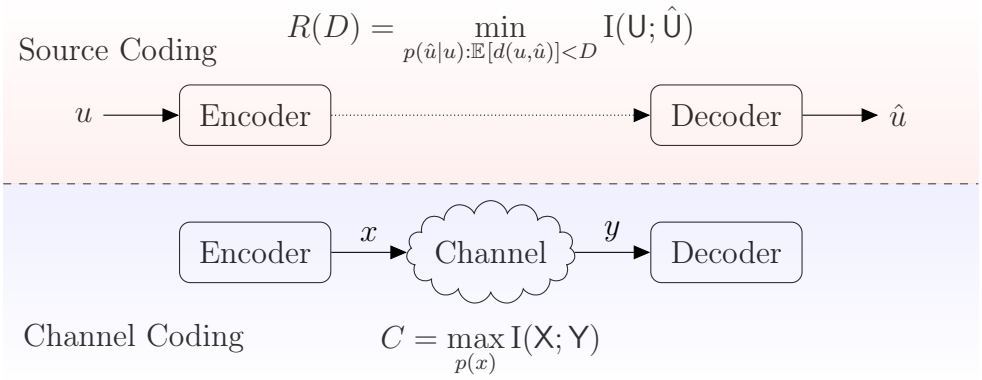


Figure 3.1: Duality between source and channel coding.

> *a mutual information [...]. In a somewhat dual way, evaluating the rate-distortion function for a source amounts, mathematically, to minimizing a mutual information [...].*" -Claude E. Shannon

Following this observation and given Fig. 3.1, one concludes that given a source, there exists a trade-off between rate R and distortion D. However, given a channel, the trade-off is between capacity C and efficiency, i.e., the gap between code rate and capacity [Cov06]. We will see later that the information bottleneck method exactly treats this min-max-optimization problem of mutual information. However, the main focus of this section is also to provide a detailed timeline of advances in information theory and machine learning which are closely related to the information bottleneck method. This timeline can be found in Table 3.1.

Somewhat natural, already in the early seventies, right after Shannon's landmark papers, many works in the information theory society related to *rate-distortion theory*, i.e., source compression, were presented. Given the theoretical bounds derived by Wyner and Ziv [WZ71], both Blahut [Bla72] and Arimoto [Ari72] proposed nearly at the same time and independently practical algorithms which achieved these bounds.

Still, with a focus on source coding, Witsenhausen, Wyner, and Ziv investigated a slightly extended rate-distortion scenario involving three random variables, namely X, Y, and Z [WW75; WZ76]. Here, the focus is on source coding of X to Z with side-information provided by Y. We will see later, that the bounds derived for this rate-distortion problem with side-information matches exactly respective bounds for the information bottleneck problem [GNT03]. However, despite the mathematically rigorous investigations and derivations, Witsenhausen and Wyner proposed no closed-form solution nor an algorithm to solve this problem.

Inline with Shannon's view on information theory and the duality of channel coding

TABLE 3.1 A brief history of the information bottleneck method.

Year	Information Theory	Machine Learning/ Pattern Recognition	Information Bottleneck
1971	Rate Distortion Theory [Ber71]		
	Bounds of Rate Distortion Theory [WZ71]		
1972	Blahut-Arimoto Algorithm [Bla72; Ari72]		
1975	Entropy Bound for Pair of Random Variables [WW75]		
1975	Rate Distortion with Side-Information [WZ76]		
⋮	⋮	⋮	⋮
1991		Generalized distance measures for partitioning [Cho91]	
1992		Minimum impurity partitions [BPK+92]	
1999		Optimal partitions for concave impurity measures [CHH99]	The Information Bottleneck Method [TPB99]
2000			Word clustering [ST00b]
2003			Connecting information bottleneck and information theory stream [GNT03]
2014		Globally optimal quantization of general binary-input channels [KY14]	
2017		Connecting information bottleneck and minimum impurity partitions [Kur17]	

and source coding, the information theory stream was eligible for proving fundamental limits and deriving achievable regimes. This often assumed a-priori knowledge of the statistics of the source. Another stream, driven by the machine learning society, focused on clustering and classification of empirical data instead of on compression of theoretical source models. Due to this fairly different emphasis on clustering data and learning models, this chapter treats this stream as independent from the information theory stream. This is also depicted in the timeline in Table 3.1.

In general, clustering describes the process of discovering groups of similar examples within a data set. In a machine learning context, this problem relates to *unsupervised* learning. Whereas in *supervised* learning a ground truth is required in addition to the data set such that a particular performance metric between the learned function and the ground truth can be optimized, *unsupervised* learning relies only on the data set itself. However, clustering, i.e., assigning similar examples within a data set to the same group, depends on the so-called *distance* measure. Possible choices of this *distance* measure, or discriminant function and the effect on the respective clustering were reviewed in [Cho91]. Chou also tried to answer the question which *distance* measure results in the optimal partitioning given an optimization criterion, e.g., the mutual information between clusters and a random variable of interest. Interestingly, Chou proposed the Kullback-Leibler divergence as a *distance* measure for this purpose. As we will see in Section 3.2.5 and as shown in [Kur17], assigning samples within a data set to clusters such that the Kullback-Leibler divergence between sample and cluster representative is minimized results in an optimal clustering in the information bottleneck sense. Please note that Chou [Cho91], Burshtein [BPK+92] and later papers in this stream use the term *minimizing impurity*, whereas in the information bottleneck literature, the term *minimizing irrelevant information* is used. Burshtein's work from 1992, i.e., [BPK+92], investigates the overall structure and shape of optimal partitions which minimize impurity. It is shown that the optimal partitioning is convex and there exist separating hyperplanes between the clusters. This fact will be used in Section 3.3 to obtain computationally-simplified information bottleneck algorithms. Despite the theoretical proof, it turned out to be fairly difficult to design algorithms that determine the globally optimum solution [CHH99]. Finally, in 2014, Kurkoski presented an efficient way to determine a globally optimal quantization of general binary-input channels [KY14].

The third stream in the timeline is the most important one for this thesis and this chapter, i.e., the information bottleneck stream. In 1999, Tishby, Pereira and Bialek proposed a generic information-theoretical framework called the *information bottleneck method*. They introduce the information bottleneck as an *unsupervised*

learning technique, i.e., a clustering framework, which considers explicitly and solely information theoretical measures, i.e., the mutual information. They argued that the main drawback of existing frameworks like rate-distortion theory is the need for an appropriately chosen distortion measure. Instead, the information bottleneck method distinguishes between *relevant* and *irrelevant* information. Thus, the information bottleneck method offers an inherent trade-off between accuracy and complexity or, in other words, a trade-off between compression and distortion. As shown in [GNT03], formally, this setting is pretty similar to the one sketched in [WW75; WZ76]. However, Tishby et al. derived an implicit solution to the information bottleneck problem and thus proposed algorithms that find locally optimal solutions. In addition, due to the neat interpretation of the involved information-theoretical quantities, they created a self-consistent generic framework described in more detail in the next section. The information bottleneck stream developed independently from the second stream including the works by Burshtein, Chou and Kurkoski. However, in 2017, again Kurkoski pointed out the relation between the algorithms and distance measures described by Chou [Cho91] and their applicability in the information bottleneck framework [Kur17].

3.2 Rate-Distortion Theory and the Information Bottleneck Setup

This section introduces rate-distortion theory and the information bottleneck method more formally.

3.2.1 Rate-Distortion Theory

Rate-distortion theory is a well known theoretical framework to derive limits and achievable rates R of compression in source coding with respect to a distortion measure D [Cov06]. As depicted in Fig. 3.1, source coding typically involves two steps. First, the process of compactly representing a source U is called *encoding* and defined by an encoding function

$$f : \mathcal{U} \mapsto \left\{1, 2, \ldots, 2^R\right\}. \tag{3.1}$$

Eq. (3.1) indicates that the source is represented using R bits. Please note that in general, source sequences U^n of length n are encoded, i.e.,

$$f_n \; : \; \mathcal{U}^n \; \mapsto \; \left\{1, 2, \ldots, 2^{nR}\right\}.$$

The second step is reconstructing the source based on the respective compact representation called *decoding* defined by

$$g : \left\{1, 2, \ldots, 2^R\right\} \mapsto \hat{\mathcal{U}}. \tag{3.2}$$

As we will see later, it is convenient to refer to the regions $f(u)$ as clusters associated with a cluster index $t \in \mathcal{T}$. The element $\hat{u} = g(t) = g(f(u))$ is called *representative* and all elements $\hat{u} = g(t), \forall t \in \mathcal{T}$ form the *codebook*. Given the encoding function, the representatives and a predefined distortion measure $d(u, \hat{u})$ the average distortion yields

$$\mathbb{E}\left[d(u, \hat{u})\right] = \sum_{u \in \mathcal{X}} d(u, \hat{u}) \underbrace{p(\hat{u}|t)}_{g(t)} \underbrace{p(t|u)}_{f(u)} p(u)\,. \tag{3.3}$$

In rate-distortion theory, the *rate* refers to the mutual information between the original source U and its reconstructed estimate $\hat{\mathsf{U}}$, i.e., $\mathrm{I}(\mathsf{U}; \hat{\mathsf{U}})$. As the reconstruction function is bijective, $\mathrm{I}(\mathsf{U}; \hat{\mathsf{U}}) = \mathrm{I}(\mathsf{U}; \mathsf{T})$, where T is called *compressed random variable*.

The aim of rate-distortion theory is to determine the smallest possible rate for a source such that the average distortion is below a given distortion D. As a consequence, this allows to define the rate-distortion function $R(D)$ as

$$R(D) = \min_{p(\hat{u}|u):\mathbb{E}[d(u,\hat{u})]<D} \mathrm{I}(\mathsf{U}; \hat{\mathsf{U}})\,. \tag{3.4}$$

3.2.2 The Information Bottleneck Method

The information bottleneck method introduced by Tishby et al. in [TPB99] treats a slightly extended setting compared to the plain rate-distortion formulation discussed in Section 3.2.1. Similar to source coding with side information [WZ76], the information bottleneck resorts to a scenario with three random variables termed

- *relevant random variable* X, $x \in \mathcal{X}$
- *observed random variable* Y, $y \in \mathcal{Y}$
- *compressed random variable* T, $t \in \mathcal{T}$.

Furthermore, the pairwise mutual information between the respective random variables is referred to as

- *original mutual information* $\mathrm{I}(\mathsf{X}; \mathsf{Y})$
- *relevant information* $\mathrm{I}(\mathsf{X}; \mathsf{T})$

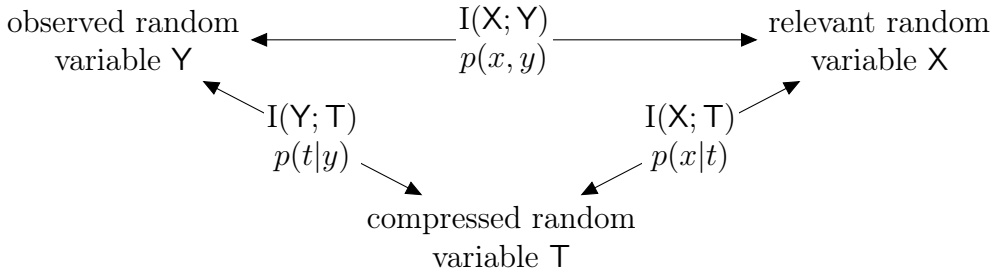


Figure 3.2: Illustration of the information bottleneck setup, where $\mathrm{I}(\mathsf{X};\mathsf{T})$ denotes the relevant information, $\mathrm{I}(\mathsf{X};\mathsf{Y})$ denotes the original mutual information and $\mathrm{I}(\mathsf{Y};\mathsf{T})$ denotes the compressed information.

- *compression information* $\mathrm{I}(\mathsf{Y};\mathsf{T})$.

The involved random variables in an information bottleneck setup form a Markov chain, i.e.,

$$\mathsf{X} \leftrightarrow \mathsf{Y} \leftrightarrow \mathsf{T}\,. \tag{3.5}$$

The three random variables and their respective mutual information form a well-defined setup as illustrated in Fig. 3.2.

In its general definition, the information bottleneck method is a generic clustering framework closely related to rate-distortion theory. However, to emphasize how this setup differs from rate-distortion theory as introduced in Section 3.2.1, let us consider the simple example of channel quantizer design as depicted in Fig. 3.3 and analyzed in more detail in Example 3.1.

Example 3.1: Channel Output Quantization as an Information Bottleneck Problem

The task of a channel quantizer is to determine a compact representation T of the possibly continuous channel output Y. Inline with classical rate-distortion theory, an optimum compression would be achieved if the found encoding, or mapping, $p(t|y)$ yields

$$\min_{p(t|y):\mathbb{E}[d(t,y)]<D} \mathrm{I}(\mathsf{Y};\mathsf{T}) \tag{3.6}$$

for a given distortion measure, e.g., the mean squared error. However, as depicted, in Fig. 3.3, an information bottleneck channel quantizer also relates to the *relevant random variable* that is, in this case, the channel input X. Thus, the side-constraint when minimizing the compression information $\mathrm{I}(\mathsf{Y};\mathsf{T})$ is the *relevant information* $\mathrm{I}(\mathsf{X};\mathsf{T})$, which should be above a particular value. Consequently, the respective

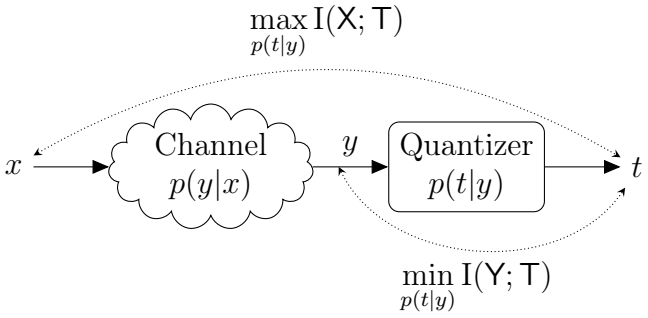


Figure 3.3: The information bottleneck setup from a channel quantization perspective.

formulation for an information bottleneck problem yields

$$\min_{p(t|y):\mathrm{I}(\mathsf{X};\mathsf{T})>\hat{D}} \mathrm{I}(\mathsf{Y};\mathsf{T})\,. \tag{3.7}$$

As discussed in Example 3.1, in contrast to rate-distortion theory, the information bottleneck method does not rely on a predefined distortion measure. Instead, the objective is inherent in the setup itself, namely to preserve a specified amount of *relevant* information $\mathrm{I}(\mathsf{X};\mathsf{T})$. Hence, closely related to the rate-distortion function (cf. Eq. (3.4)), the relevance-compression function [TPB99] can be defined as

$$\hat{R}(\hat{D}) = \min_{p(t|y):\mathrm{I}(\mathsf{X};\mathsf{T})>\hat{D}} \mathrm{I}(\mathsf{Y};\mathsf{T})\,. \tag{3.8}$$

To emphasize the close relation between rate-distortion theory and the information bottleneck method, Eq. (3.4) and Eq. (3.8) are summarized in one line as

$$R(D) = \min_{p(t|y):\,\mathbb{E}\left[d(y,t)\right] < D} \mathrm{I}(\mathsf{Y};\mathsf{T}) \qquad \hat{R}(\hat{D}) = \min_{p(t|y):\,\mathrm{I}(\mathsf{X};\mathsf{T}) > \hat{D}} \mathrm{I}(\mathsf{Y};\mathsf{T})$$

where the small but crucial difference between the objectives is highlighted in orange.

In addition to the relevance-compression function, it is often convenient to formalize Eq. (3.7) as a concave optimization problem [KY14], i.e.,

$$\mathcal{L}_{\mathrm{IB}}(p(t|y)) = \mathrm{I}(\mathsf{Y};\mathsf{T}) - \beta \mathrm{I}(\mathsf{X};\mathsf{T}) \tag{3.9}$$

where the Lagrangian multiplier β is introduced as a trade-off parameter between compression and preservation of relevant information. As derived in [Slo02], possible values for β range from $\beta = 0$ to $\beta \to \infty$. According to [Slo02], $\beta = 0$ corresponds to a pure minimization of the compression information, i.e., maximum compression,

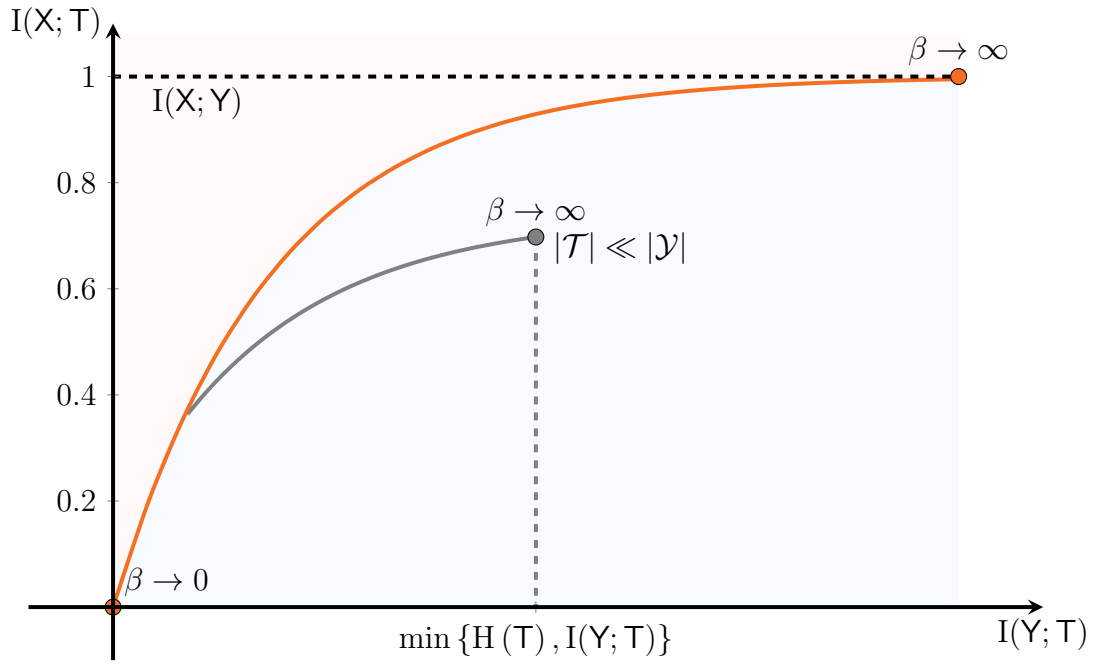


Figure 3.4: Exemplary relevance-compression curve $\hat{R}(\hat{D})$ shown in orange and an exemplary relevance-compression curve $\hat{R}(\hat{D})$ with an additional constraint on the maximum cardinality of the compressed random variable to be much smaller than the cardinality of the observed random variable, i.e., $|\mathcal{T}| \ll |\mathcal{Y}|$ (shown in gray).

whereas when $\beta \to \infty$, the focus is solely on preserving of relevant information. However, it follows from the data processing inequality that $\beta \to \infty$ does not necessarily imply $\mathrm{I}(\mathsf{X};\mathsf{Y}) = \mathrm{I}(\mathsf{X};\mathsf{T})$ as compression can be achieved by constraining the cardinality of the compressed random variable T to $|\mathcal{T}| \ll |\mathcal{Y}|$. In the extreme case of $|\mathcal{T}| = 1$, i.e., only one cluster exists, $\mathrm{I}(\mathsf{Y};\mathsf{T}) = 0$. Alternatively, when $|\mathcal{T}| = |\mathcal{Y}|$ no compression is achieved and there will be a one-to-one mapping resulting in $\mathrm{I}(\mathsf{Y};\mathsf{T}) = \mathrm{H}(\mathsf{Y})$. Thus, combining the Markov chain described by the information bottleneck setup and the data processing inequality yields

$$0 \leq \mathrm{I}(\mathsf{X};\mathsf{T}) \leq \mathrm{I}(\mathsf{X};\mathsf{Y}) \tag{3.10}$$

$$0 \leq \mathrm{I}(\mathsf{X};\mathsf{T}) \leq \min\{\mathrm{H}(\mathsf{T}), \mathrm{H}(\mathsf{X})\} \tag{3.11}$$

$$0 \leq \mathrm{I}(\mathsf{Y};\mathsf{T}) \leq \mathrm{I}(\mathsf{X};\mathsf{Y}). \tag{3.12}$$

The relevance-compression curves for these two settings are shown in Fig. 3.4. First, the orange curve depicts the classical relevance-compression curve $\hat{R}(\hat{D})$ as defined in Eq. (3.8). For $\beta = 0$, i.e., maximum compression, the relevant information approaches zero, whereas $\mathrm{I}(\mathsf{X};\mathsf{Y}) \approx \mathrm{I}(\mathsf{X};\mathsf{T})$ for $\beta \to \infty$. On the other hand, if the compressed random variable is constrained to a maximum cardinality, i.e., $|\mathcal{T}| \ll |\mathcal{Y}|$, it is shown that the maximum achievable relevant information is significantly

smaller, even as $\beta \to \infty$. As shown in the next section, for $\beta \to \infty$ the optimum mapping $p(t|y)$ becomes deterministic. Thus, $\mathrm{I}(\mathsf{Y};\mathsf{T})$ is bounded by

$$0 \leq \mathrm{I}(\mathsf{Y};\mathsf{T}) \leq \mathrm{H}(\mathsf{T}) - \underbrace{\mathrm{H}(\mathsf{T}|\mathsf{Y})}_{0 \text{ for deterministic mappings}} = \mathrm{H}(\mathsf{T}) \; . \tag{3.13}$$

Before continuing the discussion on stochastic and deterministic clustering based on the formal solution to the information bottleneck functional from (3.9), the following section determines a respective channel output quantizer derived using rate-distortion theory and a channel output quantizer that maximizes the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$.

3.2.3 Channel Output Quantizers for Binary-Input Additive White Gaussian Noise Channels

Reconsidering Fig. 3.3 and Example 3.1, a crucial problem in signal processing is the design of channel quantizers. First, a quantizer leveraging rate-distortion theory is computed. Second, an information-bottleneck-channel-output quantizer is determined. In this section, a binary-input additive white Gaussian noise (BI-AWGN) channel is considered, i.e.,

$$y = x + n, \tag{3.14}$$

where x are i.i.d. uniformly distributed samples of a binary random variable X with sample space $\mathcal{X} = \{-1, +1\}$. Furthermore, n are i.i.d. samples drawn from a normal distribution $\mathcal{N}(0, \sigma_N^2)$ with zero mean and variance σ_N^2. Consequently,

$$p(x,y) = \frac{1}{\sqrt{2\pi\sigma_N^2}} e^{-\frac{(x-y)^2}{2\sigma_N^2}} \cdot p(x) = \frac{1}{2\sqrt{2\pi\sigma_N^2}} e^{-\frac{(x-y)^2}{2\sigma_N^2}} \; . \tag{3.15}$$

The conditional distribution $p(y|x)$ for a BI-AWGN is shown in Fig. 3.5. Furthermore, in Fig. 3.5 exemplary quantization thresholds (dashed lines) and respective quantization regions are sketched that maximize the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$.

3.2.3.1 Scalar Lloyd-Max Quantizer

The rate-distortion optimal quantizer for a mean squared error distortion measure is the Lloyd-Max quantizer [Llo82]. In the scalar case, the Lloyd-Max quantizer optimizes the boundaries, so-called *thresholds*, that partition the sample space of the observed random variable, i.e., $\mathcal{Y}$ into convex sets. Thus, according to Section 3.2.1,

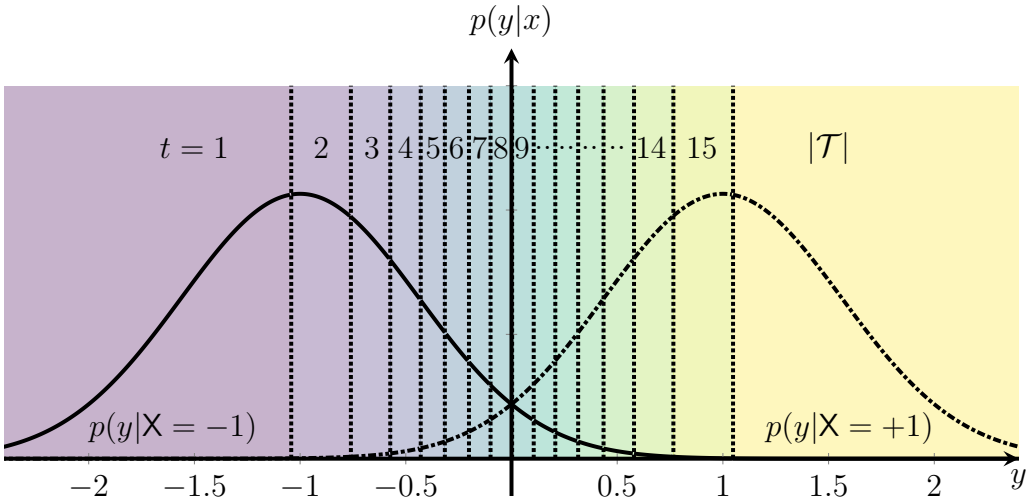


Figure 3.5: Conditional distribution of a BI-AWGN with exemplary quantization thresholds (cf. dashed lines). The numbers in each quantization region correspond to the respective cluster index t.

the mapping $t = f(y)$ maps the channel outputs into clusters $t \in \mathcal{T}$, where the elements $t \in \mathcal{T}$ called *cluster indices* are unsigned integers, i.e., $\mathcal{T} = \{1, 2, \ldots, |\mathcal{T}|\}$. The cluster index associated with a quantization region increases for increasing y as depicted in Fig. 3.5. The boundaries of each cluster t are denoted τ_t, $t = 1, 2, \ldots, |\mathcal{T}|-1$ as there exist only $|\mathcal{T}|-1$ boundaries and $\hat{y}_t$ are the cluster centroids.

In turn, the Lloyd-Max algorithm closely related to the K-Means algorithm [Bis09] resorts to two simple update rules after an initial choice of the thresholds τ_t, i.e.,

$$(1) : \hat{y}_t = \frac{\int_{\tau_t}^{\tau_{t+1}} p(y) \cdot y \, dy}{\int_{\tau_t}^{\tau_{t+1}} p(y) \, dy}, \forall t \in \mathcal{T} \tag{3.16}$$

$$(2) : \tau_t = \frac{1}{2} \left(\hat{y}_t + \hat{y}_{t+1} \right)^2, \; t = 1, 2, \ldots, |\mathcal{T}| - 1 \tag{3.17}$$

which are repeated iteratively. Here, the distortion measure is the squared Euclidean distance between $\hat{y}$ and y. Inline with rate-distortion theory, the Lloyd-Max quantizer only considers the marginal distribution $p(y)$ instead of the entire joint distribution $p(y, x)$. The rate-distortion optimal quantization boundaries and quantization regions for a BI-AWGN channel with different noise variances σ_N^2 and $|\mathcal{T}| = 16$ are shown in Fig. 3.6.

3.2.3.2 Information Bottleneck Channel Quantizer

In contrast to rate-distortion theory, the information bottleneck setup processes the entire joint distribution $p(x, y)$. In the context of *mutual-information-based signal processing*, the idea is to preserve the maximum of relevant information $\mathrm{I}(\mathsf{X}; \mathsf{T})$ given

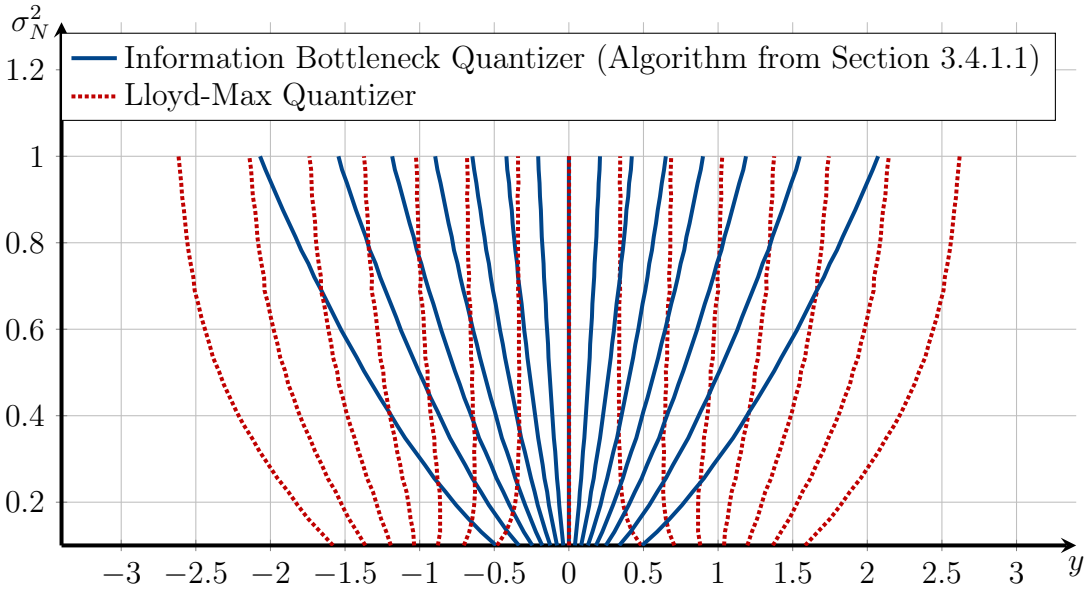


Figure 3.6: Quantization boundaries for a BI-AWGN channel for the Lloyd-Max quantizer and the information bottleneck quantizer which maximizes the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$.

a constraint for the cardinality $|\mathcal{T}|$, i.e., we choose $\beta \to \infty$. Hence, the information bottleneck setup for channel quantizer design can be reformulated as

$$\max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T})\,. \tag{3.18}$$

For this application, the so-called *linearized symmetric information bottleneck algorithm* proposed in [LB15] and discussed in more detail in Section 3.3 is applied. The quantization boundaries for this quantizer are also shown in Fig. 3.6. Interestingly, it can be observed that the quantization regions for mean-squared-error-based rate-distortion theory and the information bottleneck method look very different. The quantization boundaries of the Lloyd-Max quantizer are mainly centered around the maxima of the two Gaussians (cf. also Fig. 3.5). This observation can be explained directly considering the objective of rate-distortion, which is to minimize the average distortion. Thus, those events $y \in \mathcal{Y}$ that appear most often should be represented more precisely than very unlikely events.

In contrast, the information bottleneck quantizer places the quantization regions in the area of highest uncertainty, i.e., the area with the highest conditional entropy to maximize the relevant information. For the channel at hand, this is close to the origin as X is uniformly distributed, and the noise variance is the same for both inputs. The difference in the quantization regions also translates into different amounts of relevant information preserved by the two approaches.

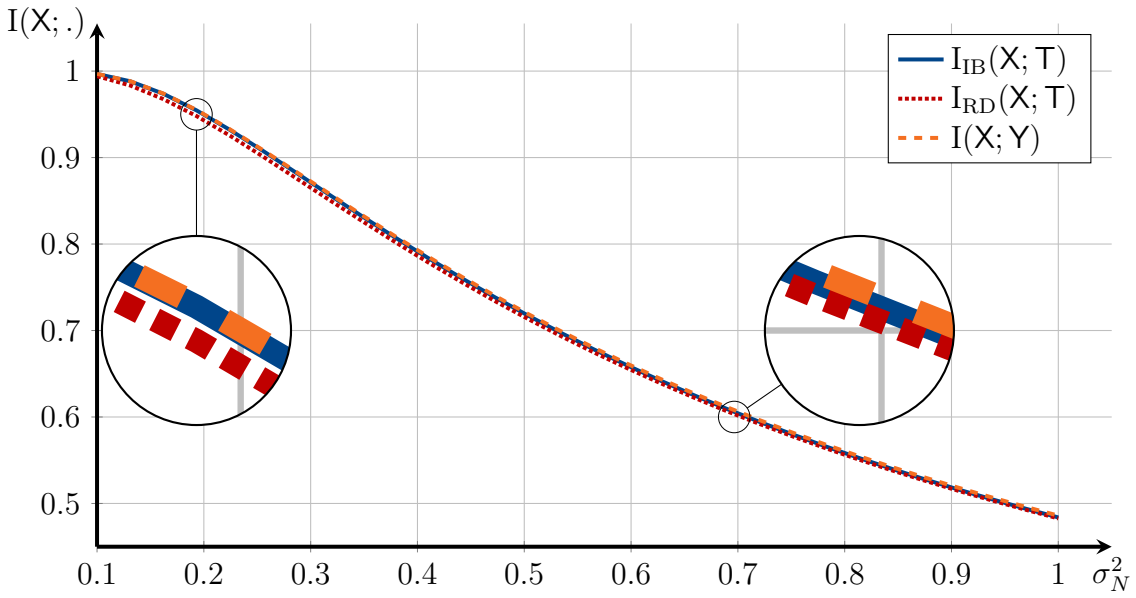
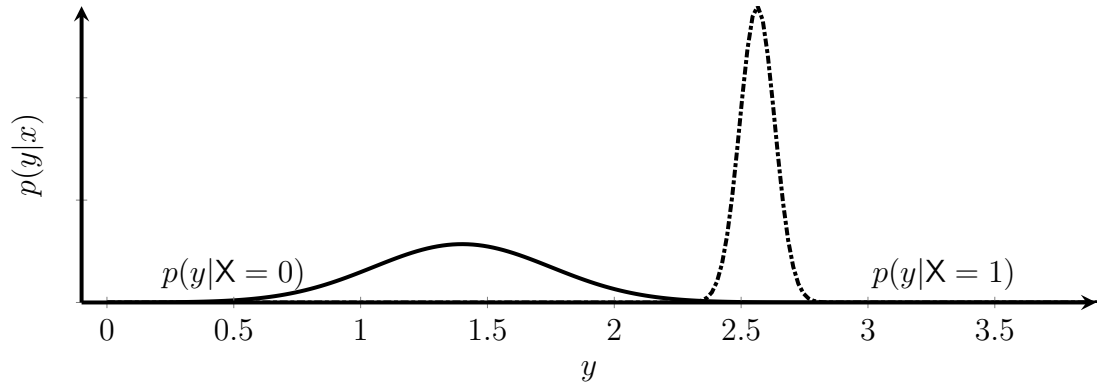


Figure 3.7: Preserved relevant information of the information bottleneck quantizer $I_{IB}(X;T)$ and the rate-distortion quantizer (Lloyd-Max) $I_{RD}(X;T)$ compared to the original input mutual information $I(X;Y)$.

Fig. 3.7 displays the relevant information preserved by the information bottleneck quantizer $I_{IB}(X;T)$ and the Lloyd-Max quantizer $I_{RD}(X;T)$ in comparison to the original input mutual information $I(X;Y)$. It can be observed that for the chosen cardinality, i.e., $|\mathcal{T}| = 16$, the information bottleneck quantizer has only a negligible loss in information for the entire range of investigated noise variances. In contrast, especially for low noise variances, the rate-distortion quantizer fails to preserve the maximum amount of relevant information, as indicated by the gap between the input mutual information curve and the $I_{RD}(X;T)$ curve (cf. left magnification class in Fig. 3.7).

Example 3.2: Channel Output Quantizer for Flash Memory Cells

Although this thesis mainly focuses on applications of mutual-information-based signal processing in modern wireless communication systems, it shall be emphasized that the derived techniques can be applied to a broad range of problems. Thus, this brief example will generalize the idea of BI-AWGN channel output quantizer design to *asymmetric* channels, as typically seen in the context of flash memory cells. NAND flash memory cells are crucial for any modern storage systems as they pair lower latency (high read/write speeds), higher reliability, and lower power consumption than classical hard disk drives [MCH19]. The information in a flash memory cell is indicated by the number of charges stored in the cell. In turn, the data can be read out by measuring the respective voltage of the cell



(a) Conditional distribution for a NAND memory cell with parameters summarized in [MCH19].

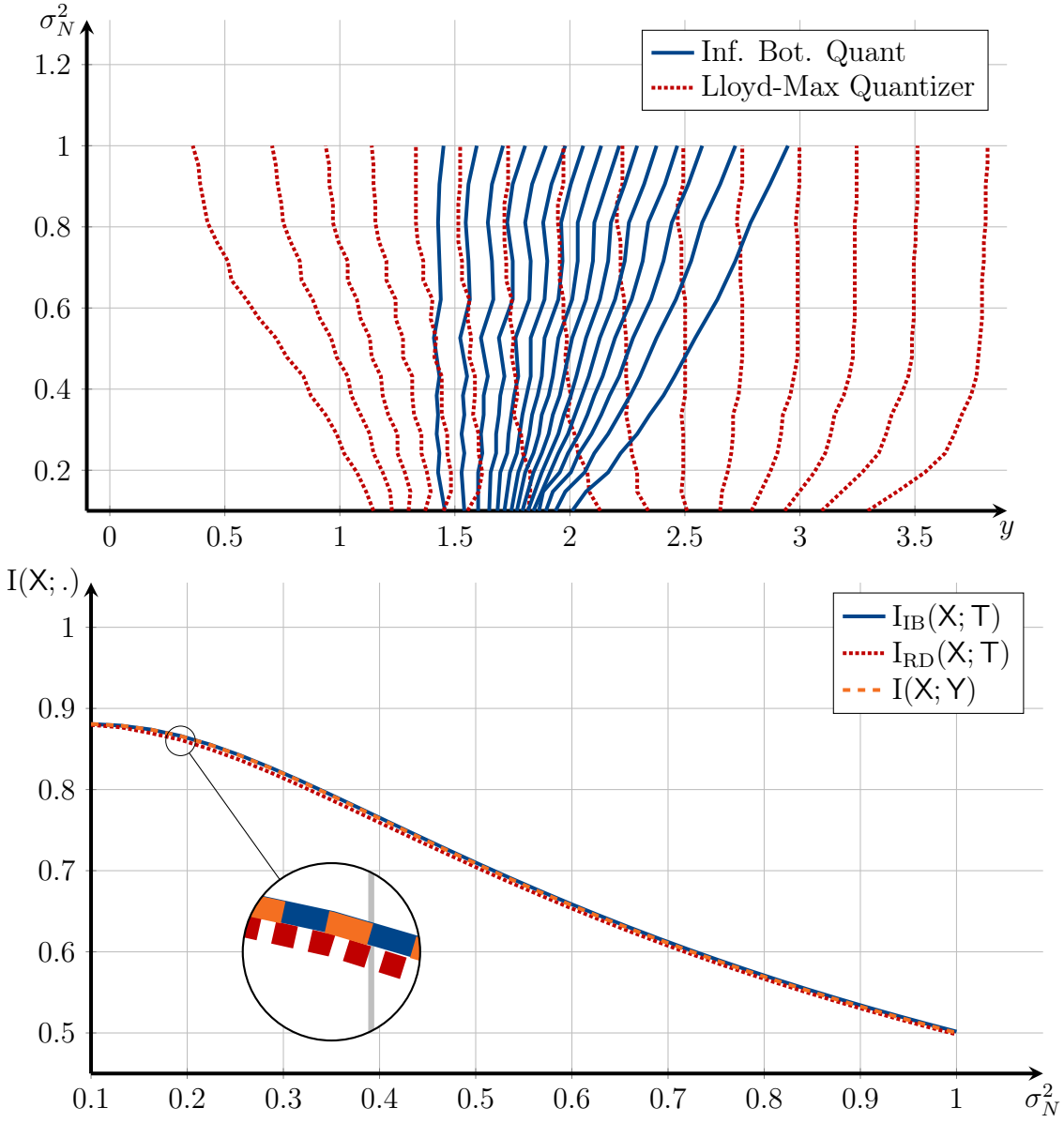


Figure 3.8: Quantization boundaries and preserved mutual information for an asymmetric channel used to model a NAND flash memory cell for the Lloyd-Max quantizer and the information bottleneck quantizer, which maximizes the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$.

and comparing it to predefined thresholds. As described in [MCH19], the flash cell voltages are affected by noise, e.g., due to programming, erasing and aging of the cell, i.e., leakage of charges. The resulting overall voltage distribution is given as a *Gaussian mixture distribution*

$$p(x, y) = p(y|\mathsf{X} = 1)\Pr(\mathsf{X} = 1) + p(y|\mathsf{X} = 0)\Pr(\mathsf{X} = 0) \tag{3.19}$$

$$p(y|\mathsf{X} = 1) \sim \mathcal{N}\left(\mu_{V_1}\,;\,\sigma^2_{N,V_1}\right) \tag{3.20}$$

$$p(y|\mathsf{X} = 0) \sim \mathcal{N}\left(\mu_{V_0}\,;\,\sigma^2_{N,V_0}\right) \tag{3.21}$$

where μ_{V_0} and μ_{V_1} are the actual reference voltages for a memory cell storing a zero respectively a one and $\sigma^2_{N,V_1}, \sigma^2_{N,V_2}$ denote the respective variances. A detailed derivation of the noise affects of NAND flash memory cells is beyond the scope of this thesis. Assuming the parameters of a realistic NAND memory cell summarized in [MCH19] yields the conditional channel probability density function shown in Fig. 3.8a. For the rest of this example we assume $\sigma^2_{N,V_1} = 7 \cdot \sigma^2_{N,V_1}$.
Again the Lloyd-Max quantizer and an information bottleneck quantizer are designed for $|\mathcal{T}| = 16$. The boundaries and preserved mutual information for each quantizer are also shown in Fig. 3.8. Again it can be observed that the information bottleneck quantizer places the boundaries between the peaks of the Gaussian distributions, whereas the Lloyd-Max quantizer places the boundaries exactly around these peaks. Furthermore, the comparison of the preserved relevant information reveals that the information bottleneck quantizer shows superior performance compared to the rate-distortion-optimal Lloyd-Max quantizer.

3.2.4 Implicit Solution of the Information Bottleneck Functional

The previous section highlighted the different objectives of rate-distortion theory and the information bottleneck method. This section focuses more on the formal solution of the information bottleneck functional defined in Eq. (3.9). Again it will be observed that the formal solution and its derivation presented in [TPB99] are closely related to the implicit solution to the rate-distortion problem, i.e., the Blahut Arimoto algorithm [Bla72].

Blahut-Arimoto Algorithm

In [Bla72], an algorithm to determine the rate-distortion function Eq. (3.4) was presented. First, Eq. (3.4) was rewritten as a Lagrangian optimization problem by the introduction of a Lagrangian multiplier $\hat{\beta}$ as

$$\mathcal{L}_{\mathrm{RD}}(p(t|y)) = \mathrm{I}(\mathsf{Y};\mathsf{T}) - \hat{\beta}\mathbb{E}\left[d(y,t)\right]. \tag{3.22}$$

In [Bla72], the solution to the rate-distortion problem was derived. For the rate-distortion functional the optimal mapping $p(t|y)$ which satisfies

$$\frac{\partial \mathcal{L}_{\mathrm{RD}}}{\partial p(t|y)} = 0 \tag{3.23}$$

is given by

$$p(t|y) = \frac{p(t)}{Z(\beta,y)} \exp\Big(-\hat{\beta} d(y,t)\Big) \tag{3.24}$$

where $Z(\beta, y)$ is a normalization function to ensure that $p(t|y)$ is a valid probability distribution [Bla72].

Based on the determined implicit solution in Eq. (3.24), [Bla72] proposed an iterative algorithm starting from a random mapping $p^{(0)}(t|y)$

$$(1): p^{(i)}(t) = \sum_{y} p^{(i-1)}(t|y)p(y) \tag{3.25}$$

$$(2): p^{(i)}(t|y) = \frac{p^{(i)}(t)}{Z(\beta,y)} \exp\Big(-\hat{\beta} d(y,t)\Big) \tag{3.26}$$

where $i = 1, 2\ldots$ denotes the iteration index. As the optimization is done over the convex sets of the normalized distributions, it was shown in [Csi84] that global convergence is assured.

Self-Consistent Information Bottleneck Equations

In contrast to rate-distortion theory, the information bottleneck method uses the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ as sort of distortion measure. Unfortunately, reusing the result obtained in rate-distortion theory is not feasible, as $\mathrm{I}(\mathsf{X};\mathsf{T})$ depends non-linear on $p(t|y)$ [Slo02] and no distortion measure is defined in advance. Hence, the information bottleneck functional from Eq. (3.9) shall be optimized. This leads directly to the self consistent information bottleneck equations.

For the information bottleneck functional the optimal mapping $p(t|y)$ which satisfies

$$\frac{\partial \mathcal{L}_{\mathrm{IB}}}{\partial p(t|y)} = 0 \tag{3.27}$$

is given by

$$p(t|y) = \frac{p(t)}{Z(\beta, y)} \exp\left(-\beta D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)\right\}\right) \tag{3.28}$$

where $Z(\beta, y)$ is a normalization function to ensure that $p(t|y)$ is a valid probability distribution [TPB99].

Thus, according to Eq. (3.28) the self-consistent information bottleneck equations are found as

$$p(t|y) = \frac{p(t)}{Z(\beta, y)} \exp\left(-\beta D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)\right\}\right) \tag{3.29}$$

$$p(t) = \sum_{y} p(t|y)p(y) \tag{3.30}$$

$$p(x|t) = \frac{1}{p(t)} \sum_{y} p(t|y)p(x, y)\,. \tag{3.31}$$

Please note that interestingly the Kullback-Leibler divergence $D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)\right\}$ in Eq. (3.28) emerged during the derivation and was not assumed as a distortion measure in advance. Furthermore, at first glance, there is a close relation between Eq. (3.28) and Eq. (3.24). Nevertheless, in contrast to Eq. (3.24), the optimization problem of the information bottleneck setup has a convex/concave structure. Thus, it cannot be guaranteed that a global optimum is found by iterating between the self-consistent equations. Hence, the presented algorithms in Section 3.3 need to be performed multiple times with different, randomly chosen, initial conditions.

The solution to the information bottleneck problem follows directly from the information bottleneck functional. However, as outlined in Section 3.1, the same self-consistent equations can also be obtained under different premises, for instance, by choosing the Kullback-Leibler divergence $D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)\right\}$ as distortion measure in a rate-distortion setting, as done for indirect source coding under logarithmic loss in [CW14; HT07].

The *Meaning* of a Cluster

Due to the Markov chain $\mathsf{X} \leftrightarrow \mathsf{Y} \leftrightarrow \mathsf{T}$, the third distribution, i.e., $p(x|t)$ appears in the self-consistent information bottleneck equations. In this thesis, this distribution is referred to as the *meaning* of a particular cluster $t \in \mathcal{T}$. The meaning can be

interpreted as a belief on the relevant variable X expressed by a particular cluster. Thus, in contrast to rate-distortion theory, where each cluster is associated with a deterministic representative, the information bottleneck provides a rather *soft* representative. It is important to highlight that the mapping $p(t|y)$ maps observations from a possibly physical domain into somehow distinguishable clusters without any natural or interpretable ordering, respectively meaning. Thus, the meaning of a cluster provided by $p(x|t)$ is crucial to perform inference on X knowing the cluster index t.

Nevertheless, as it will be shown in subsequent chapters, mutual-information-based signal processing often considers the meaning of a cluster only in the *design* and *detection* steps. In contrast, the internal processing is often done based on the compact cluster indices t alone, which reduces the computational complexity of mutual-information-based signal processing units.

3.2.5 Compression vs. Relevance - Soft Clustering vs. Hard Clustering

In information theory and signal processing, different notions of compression exist. From an information-theoretical perspective, the compression rate is given by $\mathrm{I}(\mathsf{Y};\mathsf{T})$. Thus, as

$$\mathrm{I}(\mathsf{Y};\mathsf{T}) = \mathrm{H}(\mathsf{T}) - \mathrm{H}(\mathsf{T}|\mathsf{Y}) \tag{3.32}$$

compression in a purely information theoretical sense can be already achieved if $\mathrm{H}(\mathsf{T}|\mathsf{Y}) > 0$ although $|\mathcal{T}| = |\mathcal{Y}|$ and $\mathrm{H}(\mathsf{T}) = \mathrm{H}(\mathsf{Y})$. This is easily achieved if $p(t|y)$ is a so-called *stochastic* mapping, sometimes called *soft* clustering.

In contrast, in signal processing, compression and quantization refer to the process of compactly representing an observation with fewer bits, i.e., $|\mathcal{T}| << |\mathcal{Y}|$. Furthermore, for ease of implementation, in signal processing, deterministic quantizers are preferred which map a particular observation $y \in \mathcal{Y}$ into a particular cluster $t \in \mathcal{T}$ with probability one, i.e.,

$$p(t|y) = \begin{cases} 1 & \text{if } y \in \mathcal{Y}_t \\ 0 & \text{if } y \notin \mathcal{Y}_t \end{cases} \tag{3.33}$$

where $\mathcal{Y}_t$ is a subset of $\mathcal{Y}$ and contains all observations $y \in \mathcal{Y}$ mapped into cluster t. In turn, it can be shown that

$$\mathrm{H}(\mathsf{T}|\mathsf{Y}) = -\sum_{y} p(y) \sum_{t} p(t|y) \log_2 \left(p(t|y)\right) \tag{3.34}$$

$$= 0 \tag{3.35}$$

for deterministic clusterings as

$$p(t|y)\log_2\left(p(t|y)\right) = \begin{cases} 1\cdot\log_2(1) = 0 & \text{if } y \in \mathcal{Y}_t \\ 0\cdot\log_2(0) = 0 & \text{if } y \notin \mathcal{Y}_t \end{cases}. \tag{3.36}$$

In its origin, the information bottleneck method is an information-theoretically inspired clustering framework. Hence, it primarily resorts to the information-theoretical perspective on compression where the trade-off parameter β allows to choose between relevance and compression. In mutual-information-based signal processing, the focal aim is to preserve the maximum amount of relevant information. As shown in Fig. 3.4, this is achieved by choosing $\beta \to \infty$. According to Eq. (3.28) and by application of simple calculus, it can be observed that in the limit

$$\lim_{\beta\to\infty} p(t|y) = \lim_{\beta\to\infty} \frac{p(t)\exp\left(-\beta D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)\right\}\right)}{\sum_t p(t)\exp\left(-\beta D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)\right\}\right)} \tag{3.37}$$

$$= \begin{cases} 1 & \text{for } t = \arg\min\limits_{t^*} D_{\mathrm{KL}}\left\{p(x|y)||p(x|t^*)\right\} \\ 0 & \text{else} \end{cases}. \tag{3.38}$$

Hence, for $\beta \to \infty$ the information bottleneck method yields a deterministic clustering. However, to achieve compression in the signal processing sense, $|\mathcal{T}| << |\mathcal{Y}|$.

In this thesis, the aim is the design mutual-information-based signal processing units under coarse quantization. Hence, according to the previous discussions, we restrict ourselves to $\beta \to \infty$ which leads to deterministic clusterings and compression due to a constraint on the cardinality of the compression variable, i.e., $|\mathcal{T}| << |\mathcal{Y}|$. This leads to the following consequences:

1. The original information bottleneck objective for the mapping $p(t|y)$ is reformulated as

$$p^*(t|y) = \arg\max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T}) \tag{3.39}$$

   with a constraint on $|\mathcal{T}|$.

2. According to Eq. (3.32), the compression rate is given as

$$\mathrm{I}(\mathsf{Y};\mathsf{T}) = \mathrm{H}(\mathsf{T}) = \mathrm{H}(\mathsf{Y}) - \mathrm{H}(\mathsf{Y}|\mathsf{T}) .$$

3. It is more convenient to describe the mapping $p(t|y)$ by the Kronecker-delta function

$$p(t|y) = \delta(t - f(y)) \Leftrightarrow t = f(y) \tag{3.40}$$

$\delta(t - f(y))$ where $f(y)$ is a deterministic function $f : \mathcal{Y} \mapsto \mathcal{T}$. In general, the deterministic function $f(y)$ could be implemented as lookup table (LUT). Possible alternative implementations will be presented and discussed for example in Chapter 5 and Chapter 7.

3.3 Information Bottleneck Algorithms and Devised Extensions

In the previous section, the basic concepts of the information bottleneck method were introduced. Furthermore, an implicit solution of the information bottleneck functional Eq. (3.9) was derived, resulting in Eq. (3.28). In this section, different algorithms solving the information bottleneck problem are introduced.

As outlined in [Slo02; KY14; Cov06], $\mathrm{I}(\mathsf{X};\mathsf{T})$ is itself a convex function of $p(t|y)$ for a given $p(y)$. However, as the objective for deterministic clustering is formulated as

$$p^*(t|y) = \arg\max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T}) \tag{3.41}$$

the optimization problem resorts to the class of *convex maximization* problems. As $\mathrm{I}(\mathsf{X};\mathsf{T})$ is not a convex function of $p(x|t)$, the optimization of the self-consistent information bottleneck equations is, in general, neither concave nor convex [KY14]. Hence, finding a deterministic quantizer that maximizes mutual information is typically an NP-hard problem [PR86], and most algorithms discussed in this chapter rely on heuristics. Thus, it cannot be guaranteed that a global optimum is found. Hence, these greedy algorithms need to be performed multiple times with different, randomly chosen, initial conditions to increase the probability that the found local optimum is sufficiently close to the global optimum. In the end, the found clustering which preserves the most relevant information is selected.

However, as shown later in Section 3.4.1 and proposed in [KY14], for the special case of binary-input discrete memoryless channels, the globally optimum solution can be found using dynamic programming.

In general, a large variety of information bottleneck algorithms exists [Slo02; LB18; SLB19; CGT+05]. Table 3.2 provides an overview of the algorithms discussed in

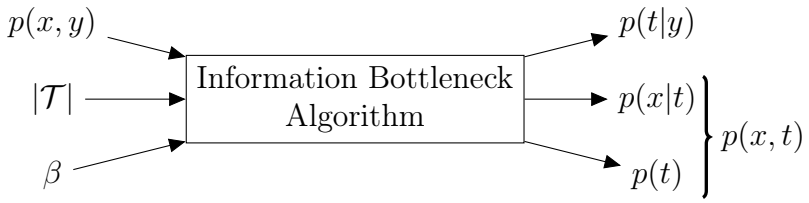


Figure 3.9: General input-output relation of an information bottleneck algorithm.

	X discrete	X continuous
T discrete	• iterative (Section 3.3.1) • agglomerative (Section 3.3.2) • sequential (Section 3.3.3) • KL-Means (Section 3.3.4)	• parametric (Section 3.4.3)
T continuous	-	• Gaussian (Section 3.4.2)

Table 3.2: Overview of selected information bottleneck algorithms for the relevant random variable, respectively compressed random variable being discrete or continuous.

this chapter. As depicted in Fig. 3.9, every information bottleneck algorithm requires the joint distribution $p(x, y)$ as input. Further inputs are the cardinality of the compressed random variable $|\mathcal{T}|$, which defines the number of clusters and, in general, also the trade-off parameter β, which is fixed to $\beta \to \infty$ if a deterministic clustering is desired. Clearly, any information bottleneck algorithm outputs the clustering $p(t|y)$. In addition, the meaning of a cluster, i.e., $p(x|t)$ and the marginal cluster probability $p(t)$ is provided.

As shown in Table 3.2, the information bottleneck algorithms can be classified according to the relevant, respectively compressed, random variable being continuous or discrete. However, algorithms optimized for continuous compressed random variables are mainly of theoretical interests as this thesis focuses solely on coarsely quantized systems where $\mathcal{T}$ is discrete.

3.3.1 Iterative Information Bottleneck Algorithm

The iterative information bottleneck algorithm was first presented in [TPB99] and followed directly from the self-consistent information bottleneck equations. The algorithm is summarized in Algorithm 1. First, the algorithm is initialized with a random mapping $p(t|y)$ and a proper choice of β and $|\mathcal{T}|$. Afterwards, the algorithm *iterates* between the self-consistent information bottleneck equations (Eqs. (3.29)

Input : $p(x, y)$, β and $|\mathcal{T}|$
Output: $p(t|y)$, $p(x|t)$ and $p(t)$

begin
  $i \leftarrow 0$;
  $p^{(i)}(t|y)$ randomly initialize mapping;
  $p^{(i)}(t) \leftarrow \sum_y p(t|y)^{(i)} p(y)$;
  $p^{(i)}(x|t) \leftarrow \frac{1}{p(t)^{(i)}} \sum_y p(t|y)^{(i)} p(x, y)$;
  while *no convergence* **do**
    $p^{(i+1)}(t|y) \leftarrow \frac{p(t)^{(i)}}{Z(\beta,y)} \exp\left(-\beta D_{\mathrm{KL}}\left\{p(x|y)||p(x|t)^{(i)}\right\}\right)$;
    $p^{(i+1)}(t) \leftarrow \sum_y p(t|y)^{(i+1)} p(y)$;
    $p^{(i+1)}(x|t) \leftarrow \frac{1}{p(t)^{(i+1)}} \sum_y p(t|y)^{(i+1)} p(x, y)$;
    $i \leftarrow i + 1$;
  end
end

Algorithm 1: Information Bottleneck Algorithm [TPB99].

to (3.31)). In each iteration i, the relevant information $\mathrm{I}(\mathsf{X}; \mathsf{T})$ can be computed for the current mapping $p^{(i)}(t|y)$. Again the superscript (i) is used to denote the distributions in iteration i. The algorithm stops either if the relevant information $\mathrm{I}(\mathsf{X}; \mathsf{T})$ stops increasing or if $p^{(i)}(t|y) = p^{(i-1)}(t|y)$, i.e., the mapping does not change anymore. As described above, convergence to the global optimum can not be guaranteed. Hence, the iterative algorithm is reinitialized with several different initial mappings and the solution which preserves the most relevant information is selected.

3.3.2 Agglomerative Information Bottleneck Algorithm

The iterative algorithm tends to become numerically unstable for the extreme case $\beta \to \infty$. Thus, different algorithms optimized for deterministic clustering have been proposed in the literature. Here, the *agglomerative* information bottleneck algorithm proposed in [ST00a] marks a straightforward greedy approach for deterministic clustering. The algorithm starts from a trivial clustering where $|\mathcal{T}|^{(0)} = |\mathcal{Y}|$, i.e., each observation is assigned to an individual cluster. In the next steps, always those two clusters t_i and t_j are merged into a new cluster $\bar{t}$ which have the smallest *merger costs*. The merger costs for the two clusters t_i and t_j are defined as

$$\Delta\mathcal{L}_{\mathrm{IB}} = p(\bar{t}) \cdot \bar{d}(t_i, t_j), \tag{3.42}$$

i.e., the loss in relevant information due to the merge for deterministic clusterings [ST00a], where $p(\bar{t}) = p(t_i) + p(t_j)$ denotes the membership probability of the possible

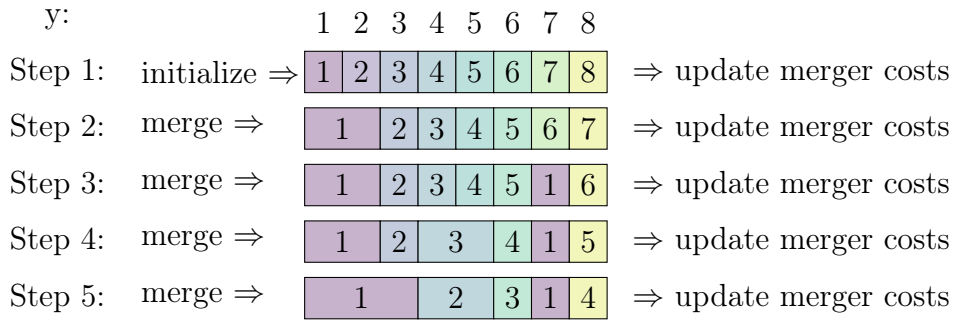


Figure 3.10: Schematic illustration of the agglomerative information bottleneck algorithm.

new class $\bar{t}$ and

$$\bar{d}(t_i, t_j) = D_{\text{JS}}^{\Pi}\left\{p(x|t_i)||p(x|t_j)\right\} + \frac{1}{\beta} D_{\text{JS}}^{\Pi}\left\{p(y|t_i)||p(y|t_j)\right\} . \tag{3.43}$$

where $D_{\text{JS}}^{\Pi}\{.||.\}$ denotes the Jensen-Shannon divergence from Eq. (2.49).

Merging is repeated until the target number of clusters $|\mathcal{T}|$ is reached. This is schematically illustrated in Fig. 3.10 for $|\mathcal{Y}| = 8$ and a target cardinality $|\mathcal{T}| = 4$. The clusters are color-coded and the number of colors reduces in each step. The algorithm is purely greedy and works only on the empirical distribution. As the computation of the merger cost is deterministic, and the algorithm does not start with a particular random clustering, it will always converge to the same solution. However, as it is known that only a local optimum solution is found, the agglomerative algorithm can get stuck easily in a bad local maximum and thus often shows the worst performance across all information bottleneck algorithms. Furthermore, especially if $|\mathcal{T}| \ll |\mathcal{Y}|$ the algorithm has to perform many sequential merging steps until the desired number of clusters is reached. Hence, the algorithm has a high runtime and also large memory requirements.

3.3.3 Sequential Information Bottleneck Algorithm

In contrast to the agglomerative information bottleneck algorithm, the sequential information bottleneck algorithm starts with an initial random clustering $p(t|y)$ [Slo02]. In an iterative procedure, always one observation y is removed from its associated cluster and put into a singleton cluster. In the next step, the merger costs of this singleton cluster to all other clusters are computed. Similar to Eq. (3.42), the merger costs are computed as

$$\Delta\mathcal{L}_{\text{IB}} = p(\bar{t}) \cdot \bar{d}(t_s, t_j) \tag{3.44}$$

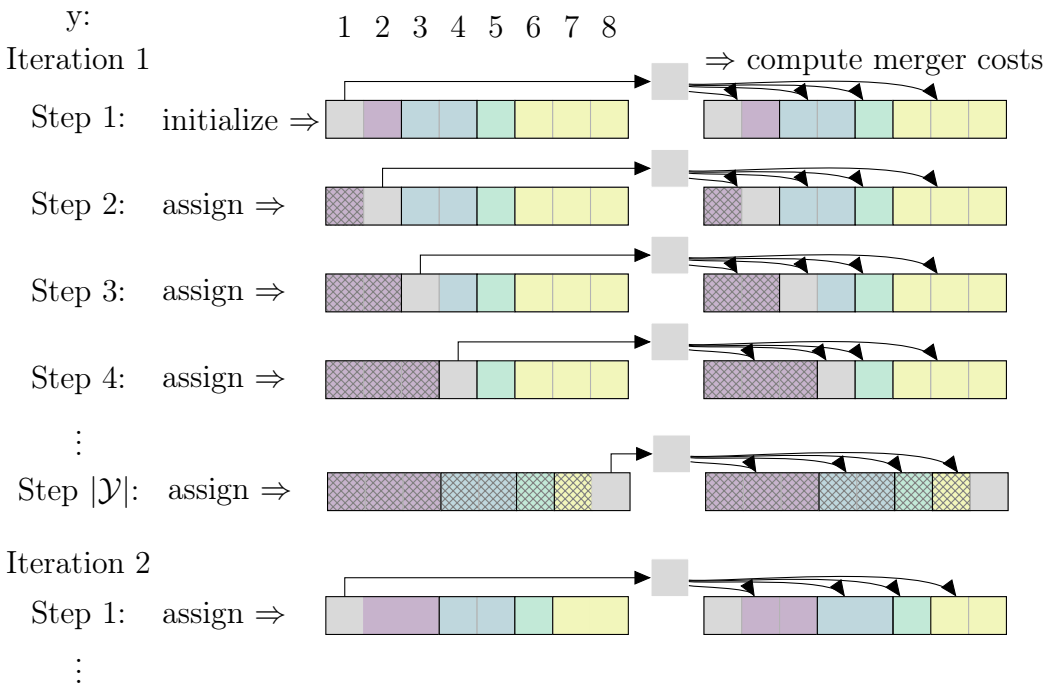


Figure 3.11: Schematic illustration of one iteration of the sequential information bottleneck algorithm which is repeated until the algorithm has converged.

where t_s denotes the singleton cluster and t_j for $j = 1, \ldots, |\mathcal{T}|$ denotes the cluster index of all existing clusters. This is schematically shown in Fig. 3.11 again for $|\mathcal{Y}| = 8$ and $|\mathcal{T}| = 4$. The singleton cluster will be assigned to the cluster with the smallest merger costs. The sequential information bottleneck algorithm requires $|\mathcal{Y}|$ merger costs calculations and assignments per iteration as each observation $y \in \mathcal{Y}$ is dragged and assigned. In contrast to the agglomerative algorithm, this assignment and computation procedure is repeated iteratively until a stable solution is found, i.e., the relevant information is not increasing anymore or the mapping remains constant over several iterations.

3.3.4 KL-Means Algorithm

The KL-Means algorithm is closely related to the K-Means algorithm well known in machine learning [Jai08] but has a wisely chosen distortion measure. The KL-Means algorithm was presented in [Kur17; ZK16] and related to the information bottleneck problem. In contrast to the previous algorithms, the derivation of the KL-Means algorithm does not start from the self-consistent equations, respectively, the information bottleneck functional. Instead, the design objective

$$\max_{p(t|y)} \mathrm{I}(\mathsf{X};\mathsf{T}) \tag{3.45}$$

serves as a starting point that is valid for deterministic clustering. This objective can be reformulated as

$$\min_{p(t|y)} \left(\mathrm{I}(\mathsf{X};\mathsf{Y}) - \mathrm{I}(\mathsf{X};\mathsf{T})\right) \tag{3.46}$$

as the aim of mutual-information-based clustering is to minimize the loss between the original mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ and the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$. Eq. (3.46) can be reformulated as

$$\min_{p(t|y)} \left(\mathrm{I}(\mathsf{X};\mathsf{Y}) - \mathrm{I}(\mathsf{X};\mathsf{T})\right) = \min_{p(t|y)} D_{\mathrm{KL}} \left\{p(x|y)||p(x|t)\right\} . \tag{3.47}$$

This reveals that the process of mutual-information-based clustering can be rewritten as a minimization of the Kullback-Leiber divergence between the posterior distribution $p(x|y)$ referred to as *backward channel* in [Kur17; ZK16; KY14] and the cluster meanings.

As a result, the KL-Means algorithm consists of two steps, an assignment step and an update step which are repeated iteratively. In the assignment step, the KL-Means algorithm assigns events to a cluster according to

$$t^* = \arg\min_{t} D_{\mathrm{KL}} \left\{p(x|y)||p(x|t)\right\}, \forall y \in \mathcal{Y}. \tag{3.48}$$

Then, the cluster meanings are updated after each assignment according to

$$p(x|t) = \frac{1}{p(t)} \sum_{y\in\mathcal{Y}} p(t|y) \cdot p(x,y), \tag{3.49}$$

which can be rewritten as

$$p(x|t) = \frac{1}{p(t)} \sum_{y\in\mathcal{Y}_t} p(x|y)\bar{p}_t(y), \tag{3.50}$$

where $\mathcal{Y}_t$ is the set of events which belong to cluster t and $\bar{p}_t(y) = \frac{p(y)}{\sum_{y\in\mathcal{Y}_t} p(y)}$ denotes the normalized event probability. This iterative update-assignment procedure is illustrated in Fig. 3.12.

The computational complexity of the KL-Means algorithm is basically affected by the required computation of the Kullback-Leibler divergence matrix to find the minimum KL-divergence between each observation and each cluster meaning. The complexity of evaluating the Kullback-Leibler divergence is affected by the cardinality of X. In most applications of the information bottleneck method in communications, the relevant variable is binary or has a small finite cardinality.

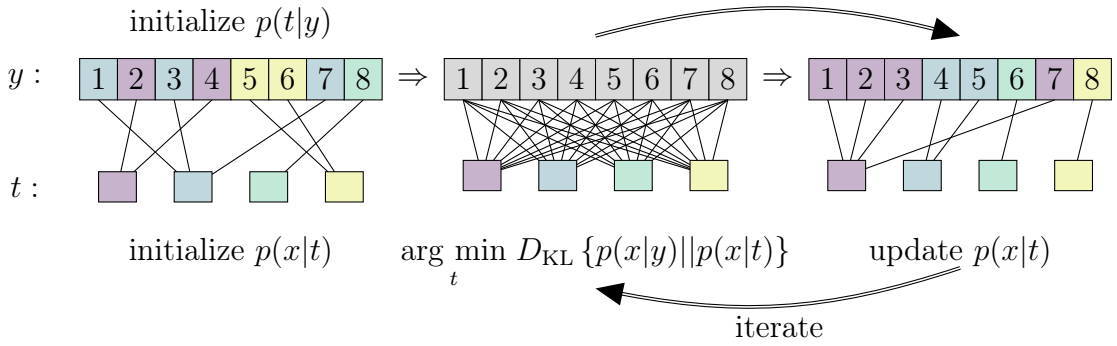


Figure 3.12: Schematic illustration of the KL-Means algorithm.

In contrast to the sequential information bottleneck algorithm, all cost computations of one iteration can be done in parallel. Thus, the runtime of the KL-Means algorithm is often smaller compared to the sequential information bottleneck algorithm.

3.3.5 A Brief Comparison of Information Bottleneck Algorithms

In general, a brute-force search for the global optimum deterministic quantizer has complexity $|\mathcal{T}|^{|\mathcal{Y}|}$, which is infeasible already for moderate $|\mathcal{Y}|$ and $|\mathcal{T}|$. As shown in [BPK+92], it is possible to exploit convexity, reducing the complexity of a brute-force search to $|\mathcal{Y}|^{|\mathcal{T}|}$. However, consider the output of an AWGN channel with fine quantization, i.e., $|\mathcal{Y}| = 1000$ which shall be clustered to $|\mathcal{T}| = 16$, i.e., 4 bit resolution. In turn, the convexity-constraint brute-force search is about $1000^{16} = 10^{48}$ possible quantizers. Thus, greedy information bottleneck algorithms are of considerable practical interest.

This section provides a brief overview and comparison of the presented information bottleneck algorithms in terms of runtime, memory complexity, and numerical stability. The total runtime depends on factors like architecture and implementation. Thus, only qualitative conclusions can be drawn. The computational complexity is measured by the number of required Kullback-Leiber divergence computations, respectively Jensen-Shannon divergence computations. Thus, the dependency on $|\mathcal{X}|$ is neglected in the overview presented in Table 3.3.

The iterative and sequential information bottleneck algorithm and the KL-Means algorithm require cost computations for each observation and with respect to each cluster, i.e., $\mathcal{O}(i \cdot |\mathcal{T}||\mathcal{Y}|)$ where i denotes the number of iterations until convergence. In contrast, the agglomerative information bottleneck algorithm starts with $|\mathcal{Y}|$ singleton clusters, thus, the first step requires $|\mathcal{Y}|^2$ merger cost computations. Due

Algorithm	Mapping	Comp. Complexity	Runtime
iterative	stochastic/deterministic	$\mathcal{O}(i \cdot \lvert\mathcal{T}\rvert\lvert\mathcal{Y}\rvert)$	fast
agglomerative	deterministic	$\mathcal{O}\left(\frac{3\lvert\mathcal{Y}\rvert^2-\lvert\mathcal{T}\rvert^2-\lvert\mathcal{Y}\rvert+\lvert\mathcal{T}\rvert}{2}\right)$	fast
sequential	deterministic	$\mathcal{O}(i \cdot \lvert\mathcal{T}\rvert\lvert\mathcal{Y}\rvert)$	slow
KL-Means	deterministic	$\mathcal{O}(i \cdot \lvert\mathcal{T}\rvert\lvert\mathcal{Y}\rvert)$	very fast

Table 3.3: Comparison of selected information bottleneck algorithms.

to the merge operations, $\sum_{k=1}^{|\mathcal{Y}|-|\mathcal{T}|} |\mathcal{Y}| - k = \frac{|\mathcal{Y}|^2-|\mathcal{T}|^2-|\mathcal{Y}|+|\mathcal{T}|}{2}$ further computations are required. Hence, in total the complexity is $\mathcal{O}\left(\frac{3|\mathcal{Y}|^2-|\mathcal{T}|^2-|\mathcal{Y}|+|\mathcal{T}|}{2}\right)$. Especially for large $|\mathcal{Y}|$ this quadratic dependency on $|\mathcal{Y}|$ is unfavorable. Furthermore, Table 3.3 summarizes the type of mapping, i.e., deterministic or stochastic and the runtime.

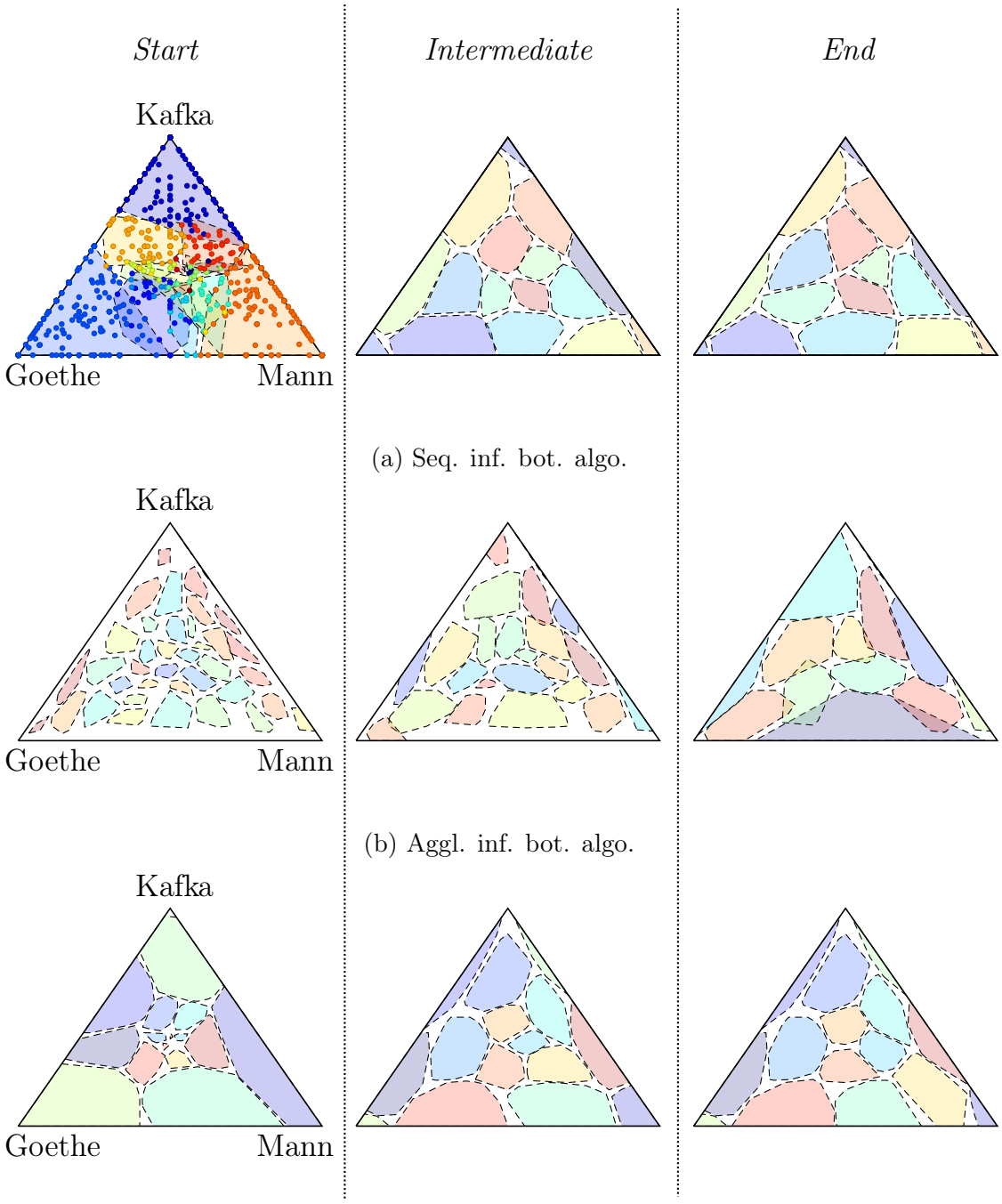
In the following example, further details and analysis of the algorithms are provided.

Example 3.3: Text Clustering using the Information Bottleneck Method

Let us consider a non-communication-related example, i.e., text clustering, respectively, author classification. Assume that one is provided with text snippets of three German authors, i.e., Johann Wolfgang Goethe, Thomas Mann and Franz Kafka. The aim is to determine very informative words in these texts which are representative for a particular author. Hence, a relevant-information-preserving clustering shall be found where the relevant random variable is the author, i.e., $x \in \{\text{Goethe}(G), \text{Kafka}(K), \text{Mann}(M)\}$. The sample space of the observed random variable Y comprises all words in the books

- Faust by Johann W. von Goethe
- Buddenbrooks by Thomas Mann
- Brief an den Vater *(Letter to His Father)* by Franz Kafka.

The words are counted and related to the author to construct $p(x, y)$, i.e., the input distribution fed into the information bottleneck algorithms. All these words are points in the probability simplex, as shown in the top left plot of Fig. 3.13. The points are shown only once as they do not change over the iterations but only the clusterings vary. Snapshots of the found mappings in different iterations for the sequential information bottleneck algorithm, agglomerative information bottleneck algorithm, and the KL-Means algorithm are shown in Fig. 3.13. The number of clusters is set to $|\mathcal{T}| = 16$. The initial mapping is always the left-most plot and the right-most plot depicts the final mapping. As the sequential algorithm works sequentially on each observation, the cluster associated with an observation is color-coded for each point in the probability simplex. Furthermore, the convex hull of the clusters is shown. It can be observed that after the initial



(a) Seq. inf. bot. algo.

(b) Aggl. inf. bot. algo.

(c) KL-Means inf. bot. algo.

Figure 3.13: Information bottleneck algorithm applied to Example 3.3 where the initial mapping is shown left and the last mapping is shown on the right and an intermediate mapping is displayed in the center.

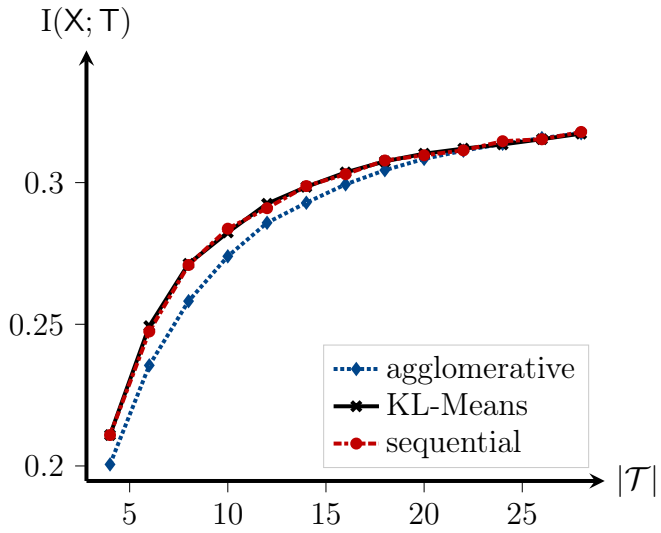


Figure 3.14: Comparison of preserved mutual information for different information bottleneck algorithms and the input distribution from Example 3.3.

iteration of the sequential information bottleneck algorithm the clusters are not necessarily convex. Nonetheless, the overall mapping changed significantly over the iterations until a local optimum is found. The agglomerative information bottleneck algorithm always merges two clusters in each step. A cluster is indicated by its colored convex hull. It can be observed in Fig. 3.13b that clusters overlap in the final clustering. This indicates that probably only a bad local optimum is found as not all events are assigned to clusters meanings $p(x|t)$ with their respective minimum Kullback-Leibler divergence. In contrast, the KL-Means algorithm finds such convex clusters right from the start but optimizes these convex cluster over the iterations. The preserved relevant information as a function of the number of clusters is displayed in Fig. 3.14. As expected, it can be observed that the sIB algorithm and the KL-Means algorithm show the smallest loss in mutual information, whereas especially for a small number of clusters, i.e., coarse quantization, the agglomerative algorithm performs worst among the investigated algorithms.
For curiosity it is also interesting to analyze the words assigned to certain clusters. For example, applying the KL-Means algorithm yields the following mapping:

cluster index	excerpt of words	meaning
1	fromm (pious), schleicht (sneak), geheimnisvoll (mysterious), Zauberei (magic), Seelen (souls), lernt (learned), Wette (bet), Menschheit (mankind)	$\begin{bmatrix} p(\mathsf{X}=G\|t)=0.982 \\ p(\mathsf{X}=K\|t)=0.014 \\ p(\mathsf{X}=M\|t)=0.004 \end{bmatrix}$
7	Erfolg (success), Ironie (irony), Tyrannei (tyranny), Rettung (rescue), verletzt (hurt), Bosheit (malice), Misstrauen (mistrust)	$\begin{bmatrix} p(\mathsf{X}=G\|t)=0 \\ p(\mathsf{X}=K\|t)=0.973 \\ p(\mathsf{X}=M\|t)=0.027 \end{bmatrix}$
16	V Villa (villa), Billards (billiards), Geschäftsfreund (business partner), Lehrer (teacher), Hausfrau (housewife), Strand (beach), Kaufmann (merchant), Kaffee (coffee), Firma (company)	$\begin{bmatrix} p(\mathsf{X}=G\|t)=0.001 \\ p(\mathsf{X}=K\|t)=0.039 \\ p(\mathsf{X}=M\|t)=0.960 \end{bmatrix}$

Interestingly, one notes that this clustering is indeed a fairly meaningful and interesting compression with respect to the considered works of the authors.

3.4 Extensions and Variants of the Information Bottleneck Setup

The information bottleneck method is a very generic information-theoretic framework. Depending on the application, different specialized streams evolved, which often exploit only a particular fraction of the method's capabilities. One such stream is mutual-information-based signal processing, as considered in this thesis, where the focus is on the design of signal processing units that preserve the maximum amount of relevant information under coarse quantization. Another stream is the so-called *Gaussian* information bottleneck. The Gaussian information bottleneck investigates analytical bounds of the information bottleneck method for the special case of jointly-Gaussian relevant and observed random variables.

In this section, a brief overview of these streams is provided and already introduced concepts are compared to these variants of the information bottleneck setup.

3.4.1 Optimal Relevant-Information-Preserving Channel Output Quantizer for Binary-Input Discrete Memoryless Channels

As described in Section 3.2.3 the design of mutual-information-maximizing channel quantization received renewed research interest after 2002 due to the several interesting works in literature [TPB99; BPK+92; MZY+02; Tho03a]. However, as discussed in Section 3.3 the information bottleneck method falls within the class of convex maximization problems which are NP-hard to solve in general and thus most greedy algorithms find only local solutions [KY14]. However, inspired by the work of Burshtein et al. [BPK+92] proposing the structure of optimal quantizers, Kurkoski proposed a dynamic programming approach for the design of quantizers for arbitrary binary-input discrete memoryless channel which guarantees to find the global optimal solution [KY14].

For clarity and a better understanding, Theorem 1 from [BPK+92] shall be restated verbatim but with renamed variables as used in the context of this thesis.

Theorem 2. *Let $\phi : \mathcal{T} \times \mathcal{X} \mapsto \mathbb{R}$ be a concave function in its second argument and let $\mathsf{V} = \mathbb{E}[\mathsf{X}|\mathsf{Y}]$ be a random variable. For any quantizer Q, there exists $Q : \mathcal{X} \mapsto \mathcal{T}$ such that $\Phi(\tilde{Q}(\mathsf{V})) \leq \Phi(Q(\mathsf{V}))$ and such that $\tilde{Q}^{-1}(t)$ is convex for all $t \in \mathcal{T}$.*

This theorem allows to reduce the search space for brute-force from $|\mathcal{T}|^{|\mathcal{Y}|}$ to $|\mathcal{Y}|^{|\mathcal{T}|}$ [KY14].

Interestingly, as described in [KY14], Kurkoski et al. showed that an optimal convex quantizer for the backward channel $p(x|y)$ of the quantizer can be found. However, this does not imply that a convex quantizer can also be found for the forward channel $p(y|x)$ [KY14]. Furthermore, it was shown that the backward channel should be ordered. For a binary-input channel, an ordering with respect to the log-likelihood ratios is possible. Given the ordered sample space, the proposed algorithm in [KY14] leverages dynamic programming to determine the optimal convex quantizer of the backward channel. The resulting algorithm is shown to have computational complexity $\mathcal{O}(|\mathcal{Y}|^3)$ which is typically much smaller than $\mathcal{O}\left(|\mathcal{Y}|^{|\mathcal{T}|}\right)$.

For an ordered event space and binary relevant variables, convex sets can be easily obtained by partitioning the sample space using clustering thresholds. The algorithm determines all combinations of boundaries of possible convex quantizers elegantly. For a binary relevant random variable, the optimization can be visualized as a Trellis diagram [KY14]. This is schematically sketched for $|\mathcal{Y}| = 5$ and $|\mathcal{T}| = 3$ in Fig. 3.15.

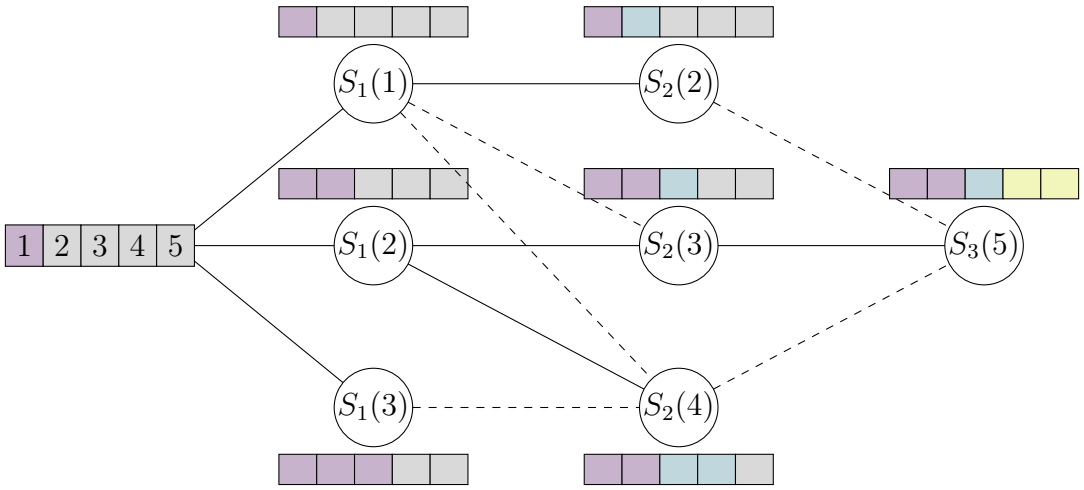


Figure 3.15: Schematic illustration of the optimal quantizer design for binary-input discrete memoryless channels.

The states $S_t(y)$ in Fig. 3.15 define the partial mutual information if the observations 1 to y are assigned to the quantization regions 1 to t. Thus, the state value can be found by recursion [KY14], i.e.,

$$S_t(y) = \max_{y'} \left(S_{t-1}(y') + \iota(y \to y') \right) \tag{3.51}$$

where $\iota(y \to y')$ denotes the partial information defined as

$$\iota(y \to y') = \sum_{x \in \mathcal{X}} \sum_{\bar{y}=y'+1}^{y} p(x, \bar{y}) \log_2 \left(\frac{\sum_{\bar{y}'=y'+1}^{y} p(\bar{y}'|x)}{\sum_{x' \in \mathcal{X}} \sum_{\bar{y}'=y'+1}^{y} p(x', \bar{y}')} \right) . \tag{3.52}$$

As shown in Fig. 3.15, several paths can arrive at a particular state. As these paths have different histories but the same future, only the incoming path which preserved most partial information will be pursued. The other path is ignored, as indicated by the dashed line in Fig. 3.15. The pseudocode of the algorithm can be found in great detail in [KY14].

3.4.1.1 Symmetric Modified Sequential Information Bottleneck Algorithm for Binary Relevant Random Variables

Also inspired by Burshtein's theorem (cf. Theorem 2), in [LB18] Lewandowsky et al. proposed a modified version of the sequential information bottleneck algorithm from Section 3.3.3 which also optimizes only over convex quantizers for a sorted sample space. In contrast to the sequential information bottleneck, its modified version is initialized with a convex cluster, i.e., only neighboring, contiguous events can be assigned to the same cluster. Afterwards, instead of computing the merger costs for

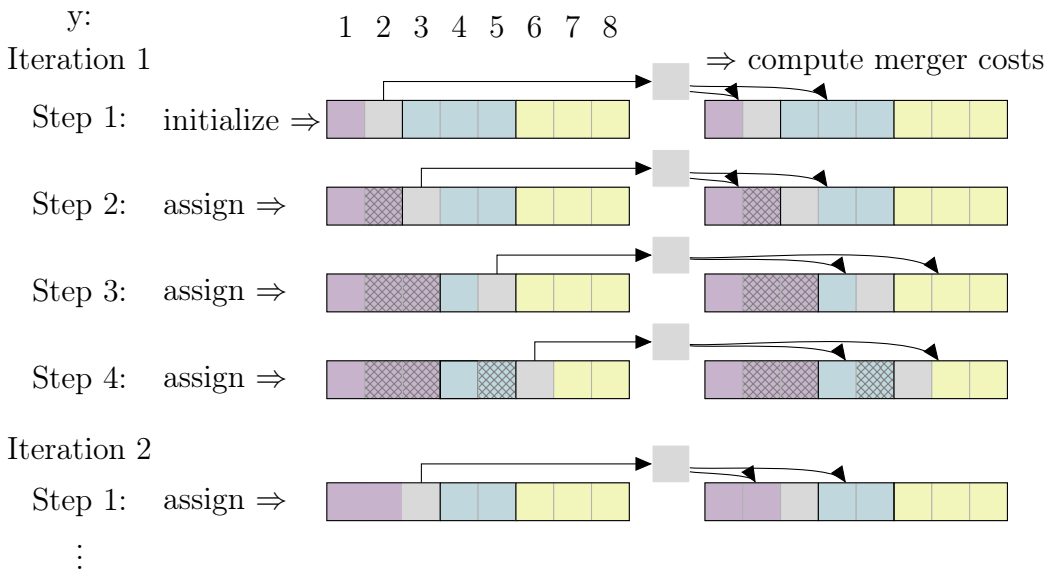


Figure 3.16: Schematic illustration of one iteration of the modified sequential information bottleneck algorithm proposed in [LB18] which is repeated until the algorithm has converged.

each element, the processing is simplified to merger cost computations *only* for the *border* elements. This is equivalent to an optimization of the quantizer boundaries. In line with the sequential structure of the original algorithm, each boundary is optimized one after the other. The procedure is repeated iteratively until a stable solution is found. Similar to the schematic illustration in Fig. 3.11 the processing of the information bottleneck algorithm from [LB18] is depicted in Fig. 3.16 . The algorithm in [LB18] is originally proposed for symmetric binary input channels, i.e., the quantizer only works on half of the sample space and mirrors the solution to the other half. However, an extension to non-symmetric input distributions is straightforward.

3.4.1.2 Modified KL-Means Algorithm for Binary Relevant Random Variables

In this thesis, also a modified version of the KL-Means algorithm is proposed and compared to the algorithms proposed in [KY14] and [LB18]. As depicted in Fig. 3.17, the KL-Means algorithm can be forced to optimize only over contiguous clusters by a proper choice of the initial clustering, respectively by a proper design of the initial meanings $p(x|t)$. If the initial meanings for binary input channels with ordered event space, i.e.,

$$\log \frac{p(\mathsf{X}=0|y)}{p(\mathsf{X}=1|y)} < \log \frac{p(\mathsf{X}=0|y')}{p(\mathsf{X}=1|y')}, \forall y < y' \tag{3.53}$$

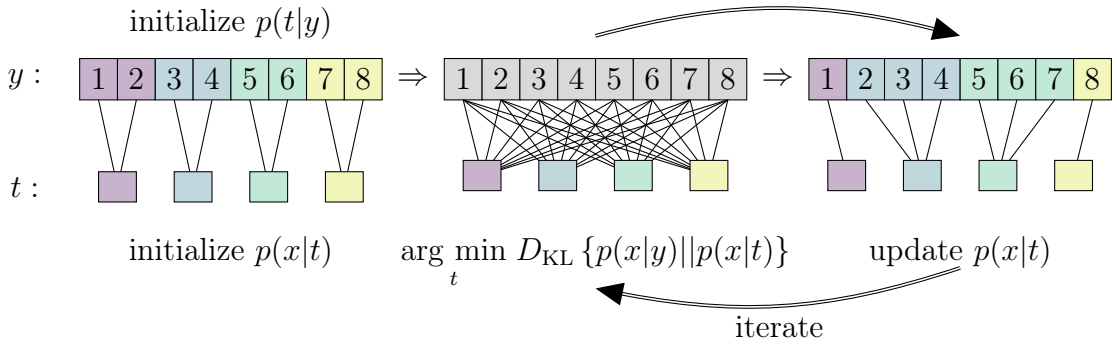


Figure 3.17: Schematic illustration of KL-Means algorithm with ordered initialization for contiguous clusters.

are chosen such that,

$$\log \frac{p(\mathsf{X}=0|t)}{p(\mathsf{X}=1|t)} < \log \frac{p(\mathsf{X}=0|t')}{p(\mathsf{X}=1|t')}, \forall t < t' \tag{3.54}$$

the optimization is restricted to contiguous clusters as the Kullback-Leibler divergence itself is convex. Furthermore, if also

$$\log \frac{p(\mathsf{X}=0|t)}{p(\mathsf{X}=1|t)} = \log \frac{p(\mathsf{X}=1|t')}{p(\mathsf{X}=0|t')}, \forall t = |\mathcal{T}| - t' \tag{3.55}$$

holds the KL-means algorithm is enforced to consider only symmetric clusters if and only if at the same time

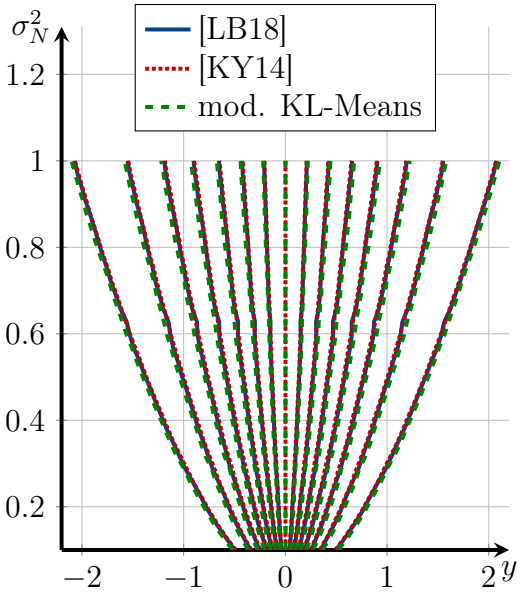
$$\log \frac{p(\mathsf{X}=0|y)}{p(\mathsf{X}=1|y)} = \log \frac{p(\mathsf{X}=1|y')}{p(\mathsf{X}=0|y')}, \forall y = |\mathcal{Y}| - y' \,. \tag{3.56}$$

In contrast to the symmetric sequential information bottleneck algorithm, the KL-means algorithm works solely on the input distribution and does not assume symmetry of $p(x|y)$. Thus, if $p(x|y)$ is not symmetric, the algorithm is not forced to resort to symmetric clusterings but inherently converges to an optimal asymmetric clustering.

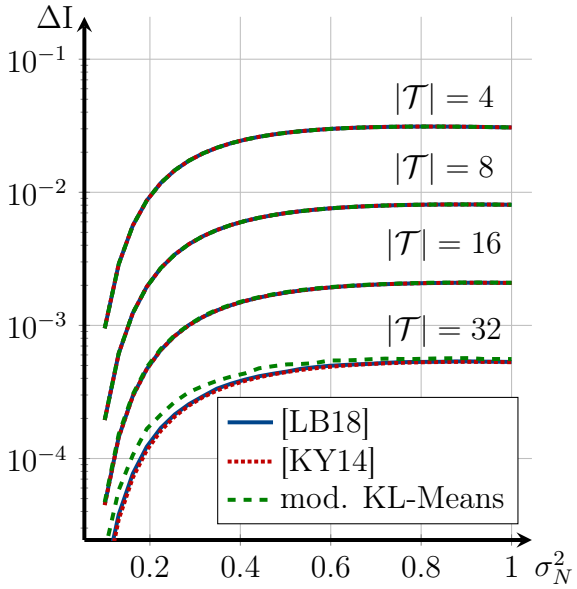
3.4.1.3 Investigation and Analysis of Binary-Input AWGN Channel Quantizers

Let us reconsider channel quantizer design for binary input channels, as discussed in Section 3.2.3, where the difference between rate-distortion and mutual-information-maximizing quantization was highlighted. In this section, presented information bottleneck algorithms are compared for the same setting.

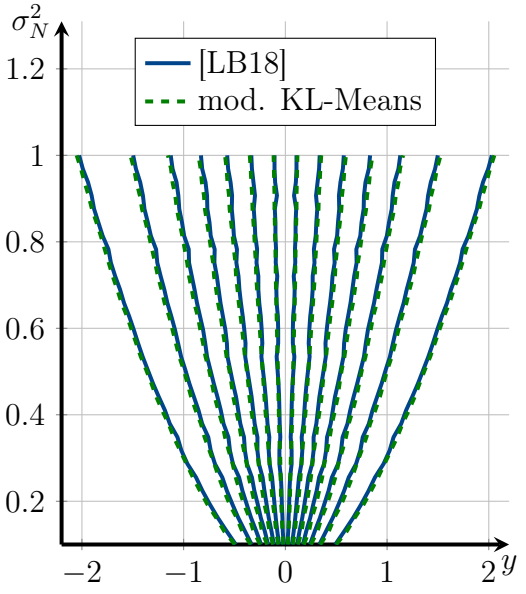
Fig. 3.18 shows the preserved mutual information and the quantization boundaries



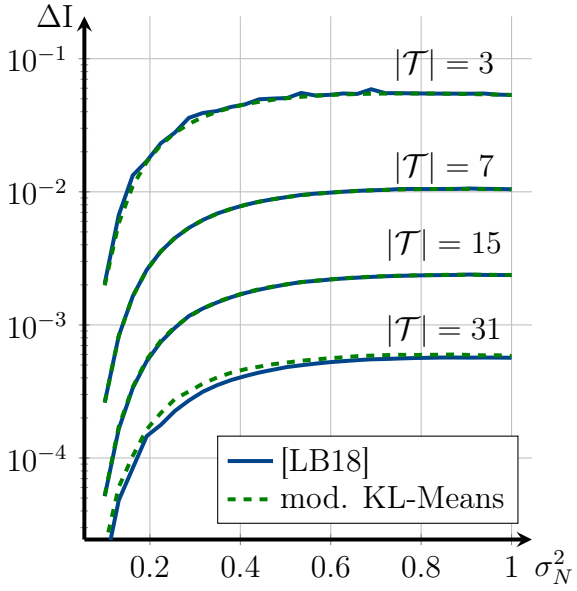
(a) Quantization boundaries for $|\mathcal{T}| = 16$.



(b) Loss in relevant information for different even numbers of clusters and algorithms.



(c) Quantization boundaries for $|\mathcal{T}| = 15$.



(d) Loss in relevant information for different odd numbers of clusters and algorithms.

Figure 3.18: Quantization boundaries and preserved mutual information for a symmetric binary-input AWGN channel for different even and odd numbers of clusters and different algorithms.

for different settings. First, let us consider an even number of clusters, as shown in the upper part of Fig. 3.18. Interestingly, it can be observed that the quantization boundaries for $|\mathcal{T}| = 16$ (cf. Fig. 3.18a) are identically for the optimal quantizer proposed in [KY14], the modified sequential information bottleneck algorithm from [LB18] and the modified KL-Means algorithm presented in Section 3.4.1.2. Similarly, the loss in mutual information, i.e.,

$$\Delta \mathrm{I} = \mathrm{I}(\mathsf{X};\mathsf{Y}) - \mathrm{I}(\mathsf{X};\mathsf{T})$$

depicted in Fig. 3.18b shows that for $|\mathcal{T}| = 4, 8, 16$ no difference in performance for the algorithms can be observed. Interestingly, for $|\mathcal{T}| = 32$ the KL-Means algorithm shows a small performance gap to the algorithm from [KY14] and [LB18]. This is also reported in literature for other applications [HWD17] where it was shown that for a large number of clusters the KL-Means algorithm is very sensitive to the random initialization. However, as this thesis targets mostly very coarsely quantized signal processing, i.e., $|\mathcal{T}| < 5$bit, all investigated algorithms can be considered equally good in terms of preserved relevant information.

In general it seems natural to use an even number of clusters as $|\mathcal{T}| = 2^{\text{bits}}$ typically where the number of bits is a positive integer. However, as shown in Fig. 3.18b, this implies that the log-likelihood ratio (LLR)=0 is not an explicit part of a cluster but the boundary instead. In some applications, however, LLR=0 plays a crucial role. Thus, *mid-step* quantizers instead of *mid-rise* quantizers can be of interest. Hence, secondly quantizers with an uneven number of clusters are considered. Fig. 3.18c and Fig. 3.18d depict the boundaries and mutual information loss for this setting. It can be observed that the boundaries are around the origin. It should be emphasized that for binary-input AWGN channel quantization the mid-step quantizer always preserves less mutual information than the mid-step quantizer for the same bit resolution. This seems intuitive as one cluster less can be used, i.e., $|\mathcal{T}| = 2^{\text{bits}} - 1$.

3.4.2 Gaussian Information Bottleneck

The Gaussian Information Bottleneck proposed in [CGT+05] is a primarily theoretical extension of the information bottleneck setup. It allows to derive an explicit, analytical, closed-form solution to the information bottleneck problem for the special case of a multivariate, jointly Gaussian relevant and observed random variable, i.e., $(\mathsf{X},\mathsf{Y}) \in \mathbb{C}^{N_x} \times \mathbb{C}^{N_y}$ with covariance matrices Σ_x, Σ_y. It is shown that in this setting, the compressed random variable is found over the linear transform

$$\mathsf{T} = \mathbf{A}\mathsf{Y} + \eta,\ \eta \sim \mathcal{N}(0, \Sigma_\eta) \tag{3.57}$$

and is also Gaussian, i.e., $\mathsf{T} \sim \mathcal{N}(0, \Sigma_t)$ with $\Sigma_t = \mathbf{A}\Sigma_y\mathbf{A}^\mathsf{T} + \Sigma_\zeta$. The transformation matrix $\mathbf{A}$ is composed of the eigenvectors of $\Sigma_{y|x}\Sigma_y^{-1}$. For a more detailed derivation and discussion the interested reader is referred to [CGT+05; ZES20].

This result has mostly theoretical implications since the compressed random variable is continuous and jointly Gaussian distributed with Y, i.e., $|\mathcal{T}|$ and $|\mathcal{Y}|$ are both infinite. Instead, only the compression information $\mathrm{I}(\mathsf{Y};\mathsf{T})$ is minimized, meeting the information-theoretical notion of compression. A *discrete* compressed random variable T with a small, finite sample space $|\mathcal{T}|$ is desirable to design relevant-information-preserving signal processing units.

Due to the nature of the solution of the Gaussian information bottleneck problem and according to [WM14], one concludes that the Gaussian information bottleneck in its pure form is of limited interest for most communication-related problems, although it has a closed-form solution. Interestingly, Winkelbauer and Matz derived the rate-information function $\mathrm{I}(R)$ for the scalar Gaussian information bottleneck problem as

$$\mathrm{I}(R) = \frac{1}{2}\log\frac{1+\rho}{1+2^{-2R}\rho} \tag{3.58}$$

where ρ denotes the signal-to-noise ratio (SNR) and showed that this expression is equivalent to the optimal rate-information trade-off achievable by rate-distortion theory [WM14]. Furthermore, Winkelbauer analyzed the performance of a Lloyd-Max quantizer with respect to these optimum bounds.

For this thesis, a novel information bottleneck algorithm was developed and proposed in [SLB19] which allows to efficiently design *discrete* clustering for Gaussian relevant random variables and Gaussian mixture distributions. This algorithm is presented in the next section.

3.4.3 Parametric Information Bottleneck Algorithms for Gaussian Relevant Random Variables and Gaussian Mixture Distributions

Most applications of the information bottleneck method are restricted to discrete decision problems due to the lack of appropriate *deterministic* information bottleneck algorithms for problems involving continuous relevant random variables. In general, one approach is to use a very fine discretization of the continuous sample space of the observed and relevant random variable. In turn, well-known information bottleneck algorithms could be applied directly but at the cost of drastically increased runtime and complexity. This motivates to develop an information bottleneck algorithm that

broadens the range of application of the information bottleneck method to continuous relevant random variables but discrete clusterings (cf. Table 3.3). Therefore, for this thesis a parametric information bottleneck algorithm suitable for relevant random variables being Gaussian or Gaussian mixtures was devised and proposed in [SLB19] which requires only the involved means and variances.

The *parametric* information bottleneck algorithm builds upon the KL-Means algorithm presented in Section 3.3.4. For the special case of a relevant variable X following a Gaussian distribution it is possible to utilize the closed-form expression [HO07]

$$D_{\mathrm{KL}}\left\{p||q\right\} = \frac{1}{2}\left(\frac{\sigma_p^2}{\sigma_q^2} + \frac{(\mu_p - \mu_q)^2}{\sigma_q^2} - 1 + \log\left(\frac{\sigma_q^2}{\sigma_p^2}\right)\right), \tag{3.59}$$

for $P \sim \mathcal{N}(\mu_p, \sigma_p^2)$ and $Q \sim \mathcal{N}(\mu_q, \sigma_q^2)$ in the assignment step of the KL-Means algorithm, i.e., Eq. (3.48). This results in the *parametric* information bottleneck algorithm. The great advantage of this algorithm is that instead of storing and processing approximated and huge numerical representations of distributions $p(x, y)$ and $p(x, t)$ to deal with continuous variables, only the respective means μ and variances σ have to be stored and processed in the algorithm. It will be shown later, that this approach has a drastic impact on the complexity.

Assuming a Gaussian likelihood $p(y|x)$, i.e., a system model $y = x + n$, $x \sim \mathcal{N}(\mu_x, \sigma_x^2), n \sim \mathcal{N}(0, \sigma_N^2)$, the posterior distribution $p(x|y)$ is also Gaussian with

$$\mu_{x|y} = \mu_x + \frac{\sigma_x^2}{\sigma_N^2 + \sigma_x^2} \cdot (y - \mu_x) \tag{3.60}$$

$$\sigma_{x|y}^2 = \frac{\sigma_x^2 \cdot \sigma_N^2}{\sigma_N^2 + \sigma_x^2}. \tag{3.61}$$

This yields the respective means and variances of $p(x|t)$ and $p(x|y)$ denoted $\mu_{x|t}$ and $\sigma_{x|t}^2$ respectively $\mu_{x|y}$ and $\sigma_{x|y}^2$ required to compute Eq. (3.59).

However, $p(x|t)$ is not Gaussian but a weighted sum of Gaussians, also called Gaussian mixture distribution with mean

$$\begin{aligned}\mu_{x|t} &= \mathbb{E}\left[\mathsf{X}|\mathsf{T} = t\right] = \sum_{y \in \mathcal{Y}_t} \bar{p}_t(y)\mathbb{E}\left[\mathsf{X}|\mathsf{Y} = y\right] \\ &= \sum_{y \in \mathcal{Y}_t} \bar{p}_t(y)\mu_{x|y} \end{aligned} \tag{3.62}$$

$$\sigma_{x|t}^2 = \mathrm{Var}\left[\mathsf{X}|\mathsf{T} = t\right] = \sum_{y \in \mathcal{Y}_t} \bar{p}_t(y) \cdot \left((\mu_{x|y} - \mu_{x|t})^2 + \sigma_{x|y}^2\right). \tag{3.63}$$

Input : Posterior mean and variance, i.e., $\mu_{x|y}$ and $\sigma^2_{x|y}$, $p(y)$, $|\mathcal{T}|$ and *tol*.
Output: Cluster mean and variance $\mu_{x|t}$ and $\sigma^2_{x|t}$, $p(t)$ and the mapping $p(t|y)$, i.e., the boundaries of the quantizer.

$p(t|y) \leftarrow$ randomly initialize convex sets with variable size
$\bar{p}_t(y) \leftarrow \frac{p(y)}{\sum_y p(y)}, \forall y \in \mathcal{Y}_t$
$\mu_{x|t} \leftarrow \sum_{y\in\mathcal{Y}_t} \bar{p}_t(y)\mu_{x|y}$ $\sigma^2_{x|t} \leftarrow \sum_{y\in\mathcal{Y}_t} \bar{p}_t(y) \cdot ((\mu_{x|y} - \mu_{x|t})^2 + \sigma^2_{x|y})$
$k \leftarrow 0$
$\mathrm{I}^k(\mathsf{X};\mathsf{T}) \leftarrow h(\mathsf{X}) - h(\mathsf{X}|\mathsf{T})$
while $I^k(X;T) - I^{k-1}(X;T) > tol$ **do**
  $k \leftarrow k + 1$
  $t^* \leftarrow \arg\min_t D_{KL}\{p(x|y)||p(x|t)\})$; ▷*Assign*
  $\bar{p}_t(y) \leftarrow \frac{p(y)}{\sum_y p(y)}, \forall y \in \mathcal{Y}_t$; ▷*Update*
  $\mu_{x|t} \leftarrow \sum_{y\in\mathcal{Y}_t} \bar{p}_t(y)\mu_{x|y}$ $\sigma^2_{x|t} \leftarrow \sum_{y\in\mathcal{Y}_t} \bar{p}_t(y) \cdot ((\mu_{x|y} - \mu_{x|t})^2 + \sigma^2_{x|y})$
  $\mathrm{I}^k(\mathsf{X};\mathsf{T}) \leftarrow h(\mathsf{X}) - h(\mathsf{X}|\mathsf{T})$
end

Algorithm 2: Parametric information bottleneck algorithm for Gaussian relevant random variables.

Following [HO07], the Gaussian mixture is unimodal, i.e., it is sufficient to approximate the Gaussian mixture as Gaussian distribution with mean $\mu_{x|t}$ and variance $\sigma^2_{x|t}$. Thus, one can resort to Eq. (3.59). The resulting parametric information bottleneck algorithm for Gaussian random variables is summarized in Algorithm 2.

The algorithm stops if the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ does not increase in further iterations. Interestingly, also the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ can be computed in closed form as

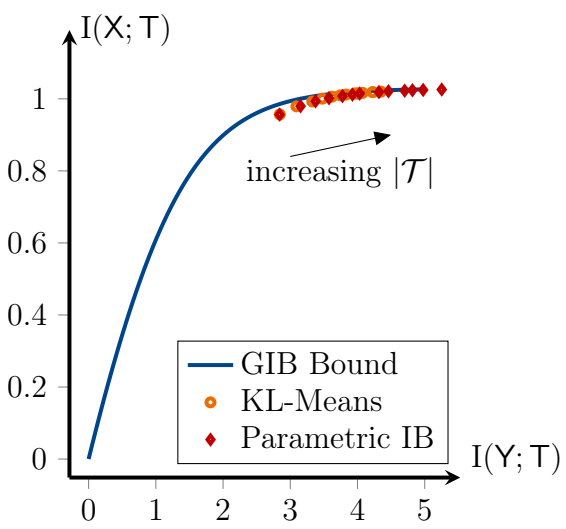
$$\mathrm{I}(\mathsf{X};\mathsf{T}) = h(\mathsf{X}) - h(\mathsf{X}|\mathsf{T})\,. \tag{3.64}$$

where

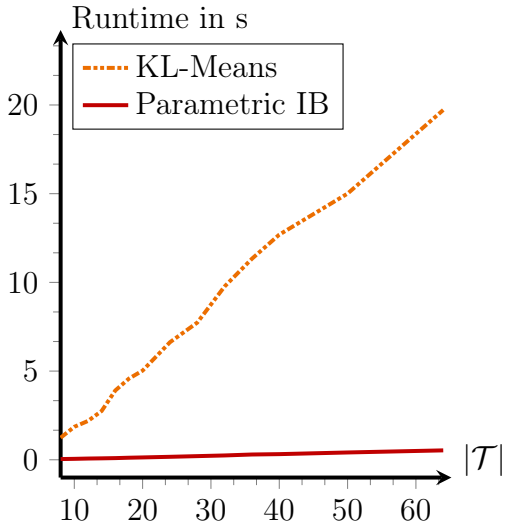
$$h(\mathsf{X}) = -\int_{-\infty}^{\infty} p(x)\log p(x)dx = \frac{1}{2}\log_2(2\pi e\sigma_x^2) \tag{3.65}$$

denotes the differential entropy and $h(\mathsf{X}|\mathsf{T}) = \frac{1}{2}\log_2(2\pi e\sigma^2_{x|t})$.

The performance of the developed algorithm is shown in Fig. 3.19a by investigating the achieved points in the relevance-compression plane. For comparison, the performance of the original KL-Means algorithm is shown. Since well-known information bottleneck algorithms, like the KL-Means algorithm, are not applicable to continuous relevant and observed random variables, the sample spaces are finely binned, i.e., $|\mathcal{X}| = |\mathcal{Y}| = 4000$ when fed into the KL-Means algorithm. Furthermore, the theoretical Gaussian information bottleneck (GIB) bound, i.e., Eq. (3.58), is included



(a) Achieved points in the relevance-compression plane including the theoretically achievable Gaussian information bottleneck (GIB) bound.



(b) Comparison of the algorithm runtimes for a varying number of clusters $|\mathcal{T}|$.

Figure 3.19: Investigation of the Parametric information bottleneck algorithm compared to the Gaussian information bottleneck bound and the finely binned KL-Means algorithm.

in Fig. 3.19a.

It can be observed that the parametric information bottleneck algorithm handles the trade-off between compression and maximum preservation of relevant information very well as all points lie very close to the theoretical bound. Please note that the remaining negligible gap between the Gaussian information bottleneck bound and the points achieved with our algorithm arises from the fact that the compression variable T is *not* continuous but discrete and has a small, finite event space.

Interestingly, not only the parametric information algorithm but also the KL-Means algorithm using a discretized input distribution operates very closely to the bound. However, Fig. 3.19b reveals the superiority of the parametric information bottleneck algorithm as it has a much smaller runtime compared to the KL-Means algorithm and shows only a moderate linear increase in runtime for an increasing number of clusters.

In the previous section, relevant random variables with Gaussian distribution were considered. A much broader class of distributions is Gaussian mixture distributions. Especially in statistical learning, these Gaussian mixture models play an essential role, e.g., in the Expectation-Maximization-algorithm, speaker recognition, or support vector machines.

As defined earlier a Gaussian mixture is the weighted sum of normal distributions

$$p(x|\mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2) = \sum_i w_i \mathcal{N}(\mu_i, \sigma_i^2), \tag{3.66}$$

where the vector $\mathbf{w}$ pools all weights w_i, with $\sum_i w_i = 1$, the vector $\boldsymbol{\mu}$ pools all component means μ_i and the vector $\boldsymbol{\sigma}^2$ pools all component variances σ_i^2.

To extend the parametric information bottleneck algorithm to Gaussian mixture models a parametric form of the posterior $p(x|y)$ is required. Again we assume a Gaussian likelihood $p(y|x)$ but this time a Gaussian mixture as prior $p(x)$. Using Bayes rule, it can be shown that a mixture prior results in a mixture of posteriors, i.e., $p(x|y) = \sum_i p_i(x|y)\tilde{w}_i$, where $p_i(x|y)$ denotes the posteriors of the components, $p_i(y)$ the marginals of the components and $\tilde{w}_i$ are their new weights found by $\tilde{w}_i = \frac{p_i(y) \cdot w_i}{p(y)}$.

According to Eq. (3.60) and Eq. (3.61), a Gaussian likelihood and different Gaussian priors the posterior $p_i(x|y)$ results in a Gaussian distribution with parameters

$$\tilde{\mu}_{i,x|y} = \mu_i + \frac{\sigma_i^2}{\sigma_N^2 + \sigma_i^2} \cdot (y - \mu_i) \tag{3.67}$$

$$\tilde{\sigma}_{i,x|y}^2 = \frac{\sigma_i^2 \cdot \sigma_N^2}{\sigma_N^2 + \sigma_i^2}\,. \tag{3.68}$$

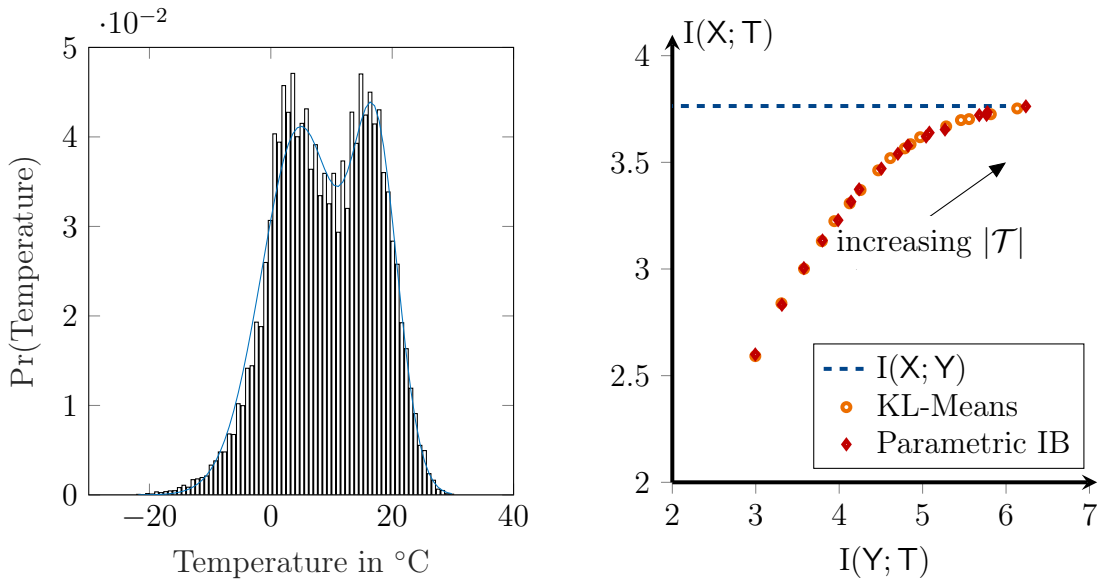
Hence, in this setting $p(x|y)$ and $p(x|t)$ are Gaussian mixture distributions that need to be considered when computing the Kullback-Leibler divergence to assign events to clusters. Since $p(x|t)$ and $p(x|y)$ are Gaussian mixtures, an approximation of the Kullback-Leibler divergence as proposed in [HO07] is needed. For two Gaussian mixtures $f = \sum_a w_a f_a$ and $g = \sum_b w_b g_b$ this approximation can be written as

$$D_{KL,\mathrm{GMM}}\{f||g\} \approx \sum_a w_a \log \frac{\sum_{a'} w_{a'} \cdot \exp(-D_{\mathrm{KL}}\{f_a||f_{a'}\})}{\sum_b w_b \cdot \exp(-D_{\mathrm{KL}}\{f_a||g_b\})}\,, \tag{3.69}$$

where f_a and $f_{a'}$ denote different Gaussian components from the mixture f. Replacing the Kullback-Leibler divergence computation in Algorithm 2 with its closed form expression yields a parametric information bottleneck algorithm for relevant variables with a Gaussian mixture distribution.

An illustrative example of the parametric information bottleneck algorithm for Gaussian mixtures is provided in Example 3.4

To summarize, in contrast to classical information bottleneck algorithms the parametric information bottleneck algorithm requires only the posterior means, variances and the marginals $p(y)$ and $p(x)$ which are both Gaussian. The cluster means can



(a) (b)

Figure 3.20: (a) Temperature distribution in Berlin since 1766 approximated by a Gaussian mixture with two components. (b) Achieved points in the relevance-compression plane of the considered algorithms including the upper bound $\mathrm{I}(\mathsf{X};\mathsf{Y})$ using a Gaussian mixture prior.

be computed and used to determine the Kullback-Leibler divergence between $p(x|y)$ and $p(x|t)$ based on the posterior means and variances. Due to the closed-form expression in Eq. (3.59), evaluating this term is very fast.

Example 3.4: Coarsely Quantized Temperature Sensors

Let us consider the design of a mutual-information based temperature sensor with coarse quantization. Here, the temperature is the relevant variable X and the observation Y is the temperature plus additive Gaussian noise. Please note that such a simple system could hardly be solved with existing information bottleneck algorithms directly, due to the continuous nature of the relevant variable, i.e., the temperature. As a starting point, one requires the marginal distribution of the relevant random variable, i.e., the temperature. The histogram in Figure 3.20a, shows the distribution of the average temperature in Berlin since 1766 [KWJ+16]. As illustrated in Figure 3.20a, the temperature distribution can be approximated quite well using a Gaussian mixture with two components and the parameters $\boldsymbol{\mu} = [17.30, 4.85]^{\mathrm{T}}$, $\mathbf{w} = [0.34, 0.66]^{\mathrm{T}}$ and $\boldsymbol{\sigma}^2 = [13.7, 40.70]^{\mathrm{T}}$.
In the next step, it is assumed that due to additive Gaussian measurement noise the continuous, sensed value equals

$$y = x + n, n \sim \mathcal{N}(0, \sigma_N^2)\,. \tag{3.70}$$

By application of the parametric information bottleneck algorithm the aim is to efficiently determine the quantizer which maximizes the relevant information. In contrast to the purely Gaussian case, a theoretical bound for Gaussian mixtures cannot be computed in closed form. Hence, the comparison of the achieved points in the relevance-compression curve only illustrates the original mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ for the considered example. The number of clusters was varied in the range from 8 to 64 clusters.
The presented simulation results show that the achieved points of the parametric information bottleneck proposed algorithm are nearly the same as the ones obtained by using the KL-Means algorithm with a finely discretized input distribution. Furthermore, the achieved points for a sufficiently large $|\mathcal{T}|$ are very close to the maximum obtainable mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$, i.e., nearly all relevant information is preserved while compressing. In turn, also in this example, the parametric information bottleneck algorithm yields a significantly shorter runtime than the KL-Means algorithm.

3.4.4 Deterministic Information Bottleneck

Another variant of the information bottleneck is the *deterministic* information bottleneck (detIB) proposed by Strouse et al. [SS17], which can be easily confused with the information bottleneck objective for deterministic clusterings (cf. Section 3.2.5). However, the detIB approach does not necessarily consider the extreme case $\beta \to \infty$. Instead the information bottleneck functional in [SS17] is reformulated as

$$\mathcal{L}_{\mathrm{detIB}}(p(t|y)) = \mathrm{H}(\mathsf{T}) - \beta \mathrm{I}(\mathsf{X};\mathsf{T}) \tag{3.71}$$

to achieve a deterministic clustering but to keep the trade-off parameter β. Please note that, Eq. (3.71) follows directly from Eq. (3.9) as

$$\begin{aligned}
\mathcal{L}_{\mathrm{detIB}}(p(t|y)) &= \mathrm{I}(\mathsf{Y};\mathsf{T}) - \beta \mathrm{I}(\mathsf{X};\mathsf{T}) && (3.72)\\
&= \mathrm{H}(\mathsf{T}) - \underbrace{\mathrm{H}(\mathsf{T}|\mathsf{Y})}_{\text{0 for deterministic mappings}} - \beta \mathrm{I}(\mathsf{X};\mathsf{T}) && (3.73)\\
&\Rightarrow \mathcal{L}_{\mathrm{detIB}}(p(t|y))\,. && (3.74)
\end{aligned}$$

In turn, according to Eq. (3.71) the trade-off in the detIB setting is not between compression information and relevant information but between the entropy of the

compressed random variable and the relevant information.

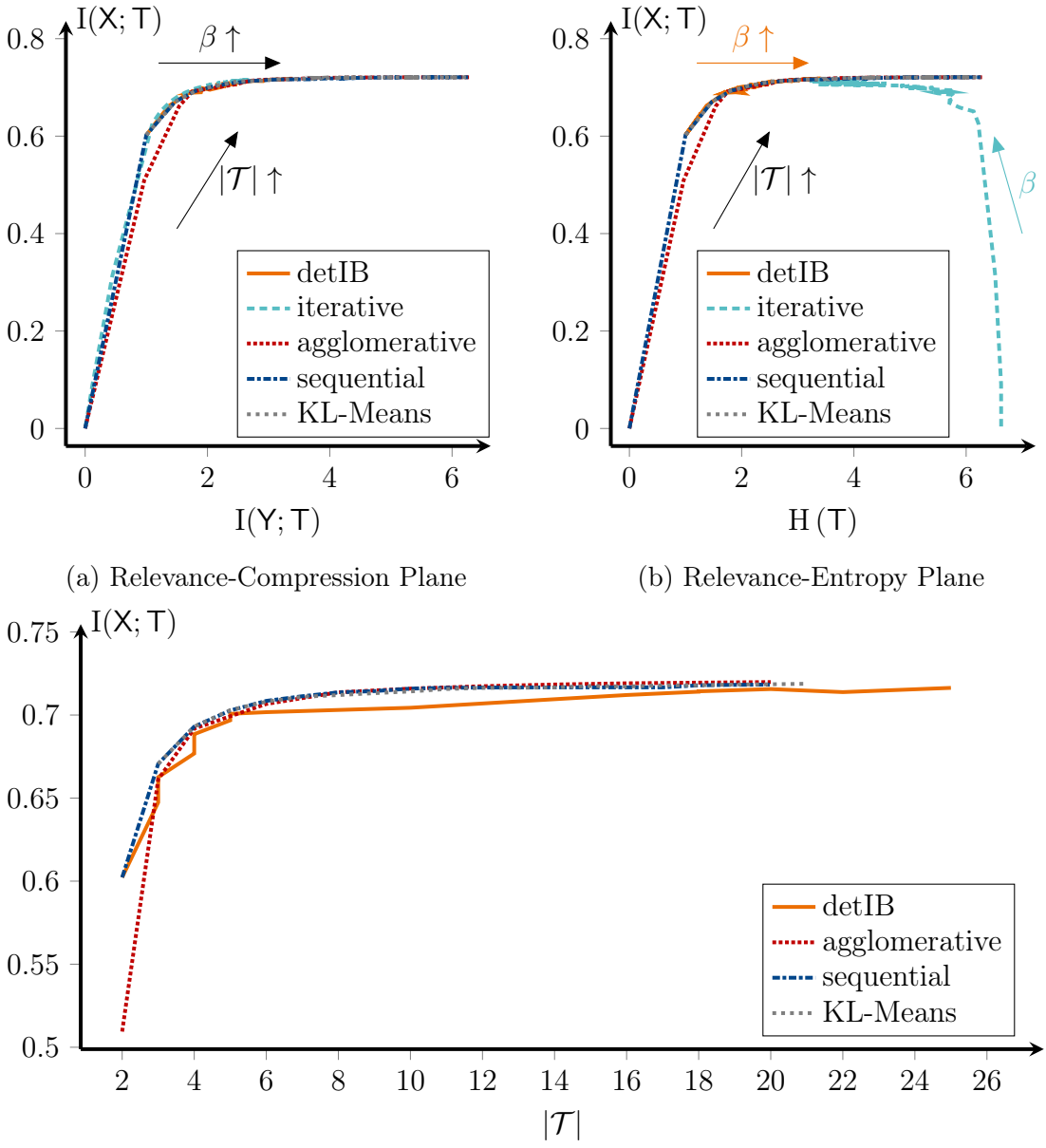
Strouse et al. motivate their modified objective with the observation in the literature that the iterative information bottleneck algorithm converges to uniformly distributed clusters, i.e., $\mathrm{H}(\mathsf{T})$ is maximized, for small β [SS17]. This is in line with the information-theoretic perspective on compression, as a stochastic mapping can reduce the compression information. Thus, $\mathrm{I}(\mathsf{Y};\mathsf{T})$ can be reduced without reducing $\mathrm{H}(\mathsf{T})$ directly. Please note that the reformulation in [SS17] does not necessarily imply that the number of clusters is reduced, i.e., a compression in the signal processing sense is achieved. Instead, only if the entropy of the compressed random variable suffice $\mathrm{H}(\mathsf{T}) < \frac{1}{|\mathcal{T}|}$ additional source coding can be applied which in turn might reduce $|\mathcal{T}|$.

Interestingly, in [SS17], only the iterative information bottleneck algorithm is considered which is known to produce stochastic clustering for finite β (cf. Section 3.2.5). In this section, a brief further comparison of the detIB approach to the deterministic information bottleneck algorithms discussed in Sections 3.3.2 to 3.3.4 is provided.

Fig. 3.21 depicts the relevance-compression curve and the entropy-compression curve used in [SS17] for the example of a channel output quantizer for a binary-input AWGN channel as considered in Section 3.4.1.3. As outlined in [SS17], it can be observed that the iterative information bottleneck algorithm and the detIB approach show the same performance in terms of the relevance-compression trade-off (cf. Fig. 3.21a) but at the expense of a higher entropy of the compressed random variable for the iterative information bottleneck algorithm (cf. Fig. 3.21b).

However, if also the deterministic information bottleneck algorithms discussed in Sections 3.3.2 to 3.3.4 are taken into consideration, it can be observed that also the entropy of the compressed random variable is similar for a particular preserved relevant information (cf. Fig. 3.21b). Please note that for the detIB and the iterative information bottleneck algorithm β is changed (cf. blue and orange arrow in Fig. 3.21b), whereas for the deterministic information bottleneck algorithms from Sections 3.3.2 to 3.3.4, $|\mathcal{T}|$ is varied as $\beta \to \infty$ for these algorithms (cf. black arrow in Fig. 3.21b).

This thesis focuses on the design of coarsely quantized signal processing units. Thus, it is more important to consider the number of required clusters to achieve a particular amount of relevant information instead of the entropy and additional source coding. Fig. 3.21c shows the number of non-empty clusters, i.e., clusters t with $p(T = t) > 0$ for different information bottleneck algorithms with respect to the associated relevant information. It can be observed that the detIB always requires



(a) Relevance-Compression Plane

(b) Relevance-Entropy Plane

(c) Number of required clusters for a particular relevant information

Figure 3.21: Comparison of the deterministic information bottleneck (detIB) approach to the classical information bottleneck setting. In subfigure (c) the iterative information bottleneck algorithm is not shown as it yields stochastic mappings for finite β.

more clusters to achieve the same relevant information preserved by the information bottleneck algorithms discussed in Sections 3.3.2 to 3.3.4. In turn, the provided investigation reveals that the detIB approach proposed in [SS17] allows reducing the number of clusters needed for a particular relevant information compared to the iterative information bottleneck algorithm. In contrast, the analysis also reveals that the deterministic algorithms discussed in Sections 3.3.2 to 3.3.4 are preferable for the objectives of this thesis as for the same $|\mathcal{T}|$ more relevant information can be preserved.

3.5 Summary

This chapter served as a gentle introduction to the area of mutual-information based signal processing under coarse quantization. The information bottleneck method was reviewed and analyzed in great detail. Furthermore, the information bottleneck approach was compared to different quantization approaches related to rate-distortion theory. Especially the close relation and the main difference between rate-distortion theory and the information bottleneck method were presented in detail. Variants and notions of the information bottleneck method were discussed, including a broad overview of advances and proposals in different research communities.

This chapter presented detailed graphical and mathematical explanations and discussed many information bottleneck algorithms able to solve the information bottleneck problem. These algorithms will be used in the remainder of this thesis to construct powerful mutual-information-based signal processing units. It was shown that deterministic information bottleneck algorithms are the preferred choice for the applications discussed in this thesis. In particular, the sequential information bottleneck algorithm and the KL-Means algorithm will be leveraged as efficient implementations exist, which yield a short runtime.

Furthermore, leveraging the illustrative example of channel quantization for binary-input AWGN channels, different algorithms, and techniques were compared and investigated. Many, also non-communication-related, examples were provided to vividly illustrate the idea of preservation of relevant information and clustering. It was shown that compared to rate-distortion theory the information bottleneck method is able to preserve a higher amount of relevant information.

New information bottleneck algorithms were devised and proposed for symmetric and non-symmetric binary-input channels and enhanced scenarios like continuous relevant random variables. The parametric information bottleneck algorithm was presented with versatile applications in settings where the relevant and observed

random variable are jointly Gaussian, or the relevant random variable is a Gaussian mixture distribution.

Finally, this chapter introduced and reviewed extensions of the information bottleneck setting like the deterministic information bottleneck setting introduced by [SS17] and contributed a detailed comparison to other information bottleneck algorithms. It was revealed that by the application of deterministic information bottleneck algorithms designed for the original information bottleneck setup, the extension proposed in [SS17] contributes no further advantages for the design of mutual-information-based signal processing units and is thus not considered in the following chapters.

Chapter 4

Information Bottleneck Graphs and the Message Alignment Problem

As emphasized in Chapter 3, the information bottleneck method is a compelling framework to design coarsely quantized, relevant-information-preserving signal processing units. However, especially if an entire communication chain or more complex signal processing consisting of many subproblems shall be developed, capturing the multivariate relations between possibly different relevant and observed random variables could be tedious. Hence, a visual framework closely related to factor graphs introduced in Section 2.2 was developed in the scope of this thesis and also published in [LSB16a]. This framework is called *information bottleneck graphs*. Information bottleneck graphs allow us to control the *flow* of relevant information in a coarsely quantized signal processing design using the information bottleneck method. The first section of this chapter introduces information bottleneck graphs in more detail.

In the second part of this chapter we devise the so-called *message alignment problem*. The close connection of the message alignment problem to the information bottleneck setup will be highlighted. The solution to the message alignment problem is identified as a crucial enabler for mutual-information-based signal processing in random graphs. Applications of message alignment in such random graphs include sensor networks and channel output quantization of higher-order modulation which we published [SLB18c] and [LSB17], respectively. Furthermore, message alignment is an integral part of the design of information bottleneck decoders for irregular LDPC codes [SLB18b] as discussed in Chapter 5. Furthermore, we used message alignment in the design of information bottleneck list decoders for polar codes [SSB19].

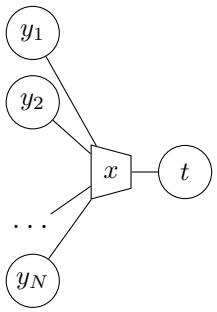


Figure 4.1: Illustration of a simple information bottleneck graph.

4.1 Information Bottleneck Graphs

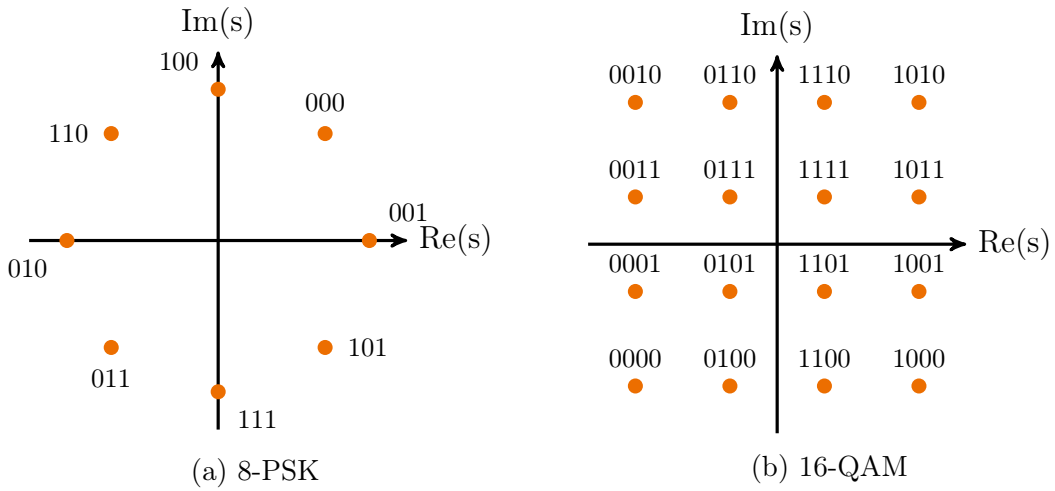
As discussed in Chapter 3, the information bottleneck method aims to preserve the maximum relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ if the observed random variable is squeezed through a compact bottleneck, i.e., the observation is represented by the compressed random variable T. This requires access to the joint distribution between the relevant and observed random variable. Especially in more advanced communication units, the observed random variable is often multivariate, i.e., $\mathbf{\mathsf{Y}} = [\mathsf{Y}_1, \mathsf{Y}_2, \dots \mathsf{Y}_N]$ (cf. Section 2.2). In such scenarios solving the information bottleneck problem can become tedious as the computational complexity of most information bottleneck algorithms depends largely on $|\mathbf{\mathsf{Y}}|$ (cf. Section 3.3.5). In statistical inference, factor graphs as introduced in Section 2.2 are a powerful framework which exploits conditional independences between random variables to determine simpler subproblems.

Inspired by the concept of factor graphs, so-called information bottleneck graphs were developed in the scope of this thesis and first published in [LSB16a]. Similar to factor graphs, information bottleneck graphs are bi-partied graphs consisting of

1. variable nodes
2. information bottleneck nodes.

Here the information bottleneck node illustrates compression mappings $p(t|\mathbf{y})$, which were designed with the information bottleneck method to preserve information about X. The information bottleneck node is represented by a trapezoid, where a vector of observations $\mathbf{y} = [y_1, y_2, \dots, y_N]^{\mathrm{T}}$ is connected to all but the shortest sides and the compression variable is connected to the shortest side. The relevant variable is written in the center of this node. This is depicted in Fig. 4.1.

The transformation of a factor graph into an associated information bottleneck graph is described more vividly in Example 4.1.



(a) 8-PSK (b) 16-QAM

Figure 4.2: Constellations of higher order modulation for 8-PSK and 16-QAM with Gray labeling.

Example 4.1: Higher-Order Modulation Symbol Quantizer Design

Let us consider the problem of channel output quantization for an AWGN channel and higher-order modulation. Higher-order modulation schemes are characterized by M constellation points in the complex plane, the so-called transmit symbols. Each symbol carries $\log_2(M)$ bits. Two common constellations, i.e., 16-quadrature amplitude modulation (QAM) and 8-phase-shift keying (PSK) are depicted in Fig. 4.2 with respective Gray labeling.

In contrast to binary phase-shift keying (BPSK) modulation, the symbols and thus also the received samples are complex. Hence, one can model the complex AWGN channel as

$$y = y_{\text{re}} + jy_{\text{im}} = s + n = s_{\text{re}} + js_{\text{im}} + n\,, \tag{4.1}$$

where n are i.i.d. samples from a complex zero-mean Gaussian distribution with variance σ_N^2.

In this example, the joint distribution fed into the information bottleneck algorithm is $p(s, y_{\text{re}}, y_{\text{im}})$. However, one observes that for any M-QAM the real and imaginary parts are independent as exploited in [LSB17], i.e.,

$$p(s, y_{\text{re}}, y_{\text{im}}) = p(y_{\text{re}}|s_{\text{re}})p(y_{\text{im}}|s_{\text{im}})p(s). \tag{4.2}$$

From Eq. (4.2) and the corresponding factor graph in Fig. 4.3, one observes that for 16-QAM, two independent information bottleneck quantizers, namely one for the real part $p(t_{\text{re}}|y_{\text{re}})$ and one for the imaginary part $p(t_{\text{im}}|y_{\text{im}})$, can be designed

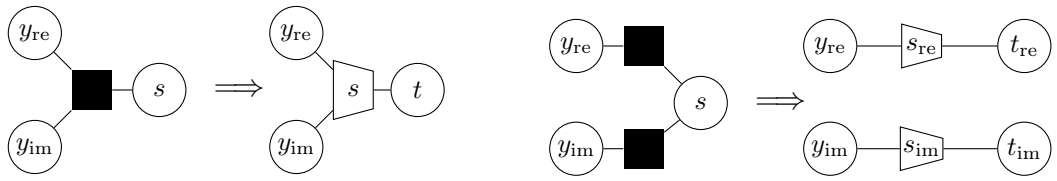


Figure 4.3: Transformation of the respective factor graph into an information bottleneck graph for (a) 8-PSK and (b) 16-QAM.

without any performance degradation. One calls this process *opening of the information bottleneck node*.

Please note that such a simple decomposition is not possible for the 8-PSK modulation. Here, a vector quantizer $p(t|y_{\mathrm{im}}, y_{\mathrm{re}})$ has to be designed using an information bottleneck algorithm.

Fig. 4.3a and Fig. 4.3b also illustrate the transformation from a factor graph into an information bottleneck graph. In Fig. 4.3a, it can be observed that the variable node for the relevant random variable S vanishes and instead, a variable node for the compressed random variable T is drawn. However, the information bottleneck node is labeled with the respective relevant random variable to indicate that the particular clustering was designed to maximize $\mathrm{I}(\mathsf{X};\mathsf{T})$.

The trapezoid symbol which represents the information bottleneck node can also be interpreted as a pointer indicating the direction of the *flow of relevant information*.

Fig. 4.4 shows the loss in mutual information due to quantization for different noise variances and different $|\mathcal{T}|$ for 16 QAM. As expected, opening the information bottleneck node, i.e., designing an independent quantizer $p(t_{\mathrm{re}}|y_{\mathrm{re}})$ for the real part and $p(t_{\mathrm{im}}|y_{\mathrm{im}})$ for the imaginary part did not result in a performance degradation compared to the closed information bottleneck node, i.e., $p(t|y_{\mathrm{im}}, y_{\mathrm{re}})$. In addition, the mutual-information loss $\Delta\mathrm{I}_{\mathrm{RD}}(\mathsf{X};\mathsf{T})$ for the Lloyd max quantizer is shown for comparison, which is always larger, i.e., the rate-distortion quantizer looses more mutual information.

Example 4.1 illustrates a case where it was possible to *open*, i.e., decompose, the information bottleneck node without any performance degradation to reduce the computational complexity. However, in some applications, for example, the sum-product algorithm, it is sometimes necessary to open the information bottleneck node to achieve manageable computational complexity but concatenate several lossy information bottleneck clusterings. In such cases, the preserved mutual information of the opened node will be smaller than the mutual information of the closed node.

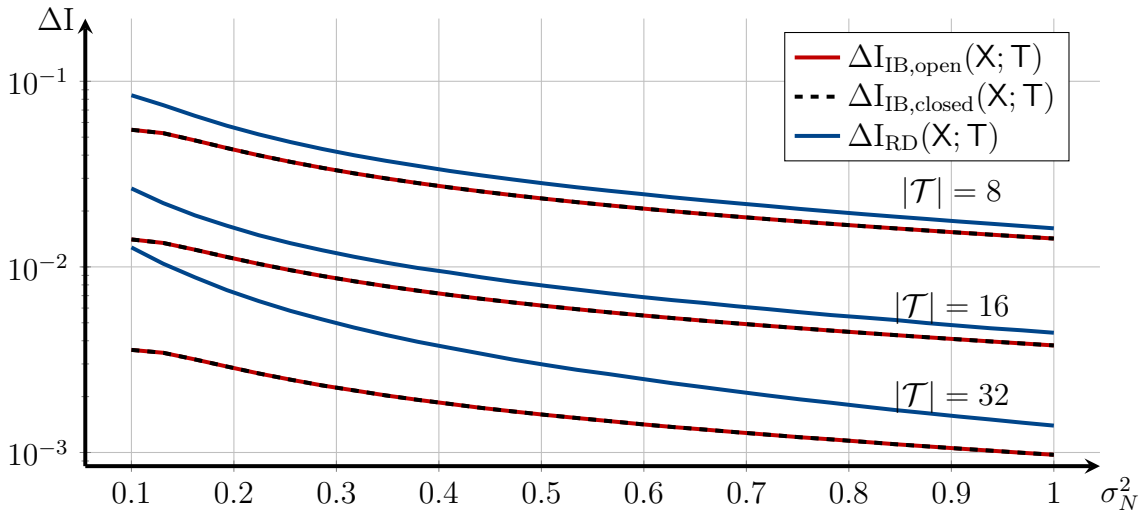


Figure 4.4: Loss in mutual information for a 16-QAM quantizer for the opened and closed information bottleneck node compared to the Lloyd Max quantizer.

Nevertheless, it turns out that it is often possible to design highly meaningful *flows of relevant information* using information bottleneck graphs that are implementable in practice and preserve a significant amount of relevant information.

This is discussed in more detail in the next section, where a coarsely quantized multiple-symbol detector for differential modulation is presented, which implements mutual-information-based message passing similar to the forward-backward algorithm introduced in Section 2.2, also termed BCJR algorithm [BCJ+74].

4.1.1 Maximum A-Posterior Multiple Symbol Detection for Phase Noise Receivers

In this section, a more complex mutual-information based signal-processing unit with coarse quantization is presented, i.e., a maximum a-posterior multiple symbol detector for differential modulation under phase noise.

The M-PSK and M-QAM constellations, as shown in Fig. 4.2, encode the conveyed information in the phase and amplitude. So far, only AWGN channels have been considered. However, in practice, additional channel perturbations are observed which are typically modeled by complex channel coefficients $h = |h|e^{j\phi_h} \in \mathbb{C}$, i.e.,

$$y = hs + n\,. \tag{4.3}$$

The complex coefficient h causes a phase rotation and an amplitude change of the original transmit symbol s.

In communications engineering, two common approaches exist to cope with these random and unknown phase rotations and amplitude changes. These approaches are called *coherent* and *non-coherent* detection [Gol05]. Coherent demodulation requires a precise channel estimation to compensate for the channel impact. However, this estimate might be difficult to obtain or notably increases the complexity of the receiver. In the scope of this thesis, coarsely quantized channel estimators were designed using the information bottleneck method which have a manageable implementation complexity and ensure near-optimum detection performance. These developed channel estimators use information bottleneck graphs [LSB16a; LSM+17].

In contrast, non-coherent demodulation does not require a coherent phase reference with respect to the transmitted signal [Gol05]. Instead, non-coherent modulation and demodulation relies on *differential* encoding. In differential modulation, the information is not conveyed by a constellation point s as shown in Fig. 4.2 directly but encoded by the transition between two subsequent constellation symbols s_k and s_{k-1} where k denotes the discrete-time index. This technique is commonly used for M-PSK modulation where the information is solely expressed by the phase, i.e., $s_k = e^{j\phi_{s,k}}$. Differential modulation is done as follows. First, the information is mapped onto the PSK symbol $u_k = e^{j\phi_{u,k}}$. In the next step, this current symbol is multiplied with the previous modulation symbol, i.e.,

$$s_k = u_k s_{k-1} = e^{j(\phi_{u,k}+\phi_{s,k-1})} = e^{j(\phi_{s,k})} \,. \tag{4.4}$$

Assuming $h_k \approx h_{k-1}$ at the receiver, the demodulator computes

$$\frac{y_k}{y_{k-1}} = \frac{h_k e^{j(\phi_{s,k})} + n_k}{h_{k-1} e^{j(\phi_{s,k-1})} + n_{k-1}} = e^{j(\phi_{s,k}-\phi_{s,k-1})} = e^{j(\phi_{u,k})} \tag{4.5}$$

and recovers u_k without knowledge of the complex channel coefficient h_k for the noise-free case. However, it can be observed that differential modulation is prone to error-propagation as one wrongly recovered phase can lead to two erroneous information symbols [Gol05]. This performance gap to coherent detection can be reduced using a multiple-symbol detector which considers a sequence of modulation symbols instead of only two subsequent symbols. Let $\mathbf{s} = [s_0, s_1, \ldots, s_k, \ldots, s_N]^\mathsf{T}$ denote the transmit vector of N subsequent samples and the initial reference symbol s_0. Likewise, $\mathbf{y} = [y_0, y_1, \ldots, y_k, \ldots, y_N]^\mathsf{T}$ denotes the receive vector and $\mathbf{u} = [u_1, u_2, \ldots, u_k, \ldots, u_N]^\mathsf{T}$ contains the information symbols which shall be transmitted. Hence, the modulation symbol s_k can be computed as

$$s_k = e^{\sum_{i=1}^{k} \phi_{u,k}+\phi_{s,0}} \,. \tag{4.6}$$

For ease of notation the phase of the initial modulation symbol is set to $\phi_{s,0} = 0$.

Non-coherent detection and thus differential modulation is of large interest in low-cost Internet-of-Things (IoT) devices which have only limited computational and energy resources [NML+19]. Thus, non-coherent detection is also an interesting application for coarsely quantized mutual-information-based signal processing design, as considered in this thesis. Furthermore, multiple symbol detection shall serve as a further illustrative example to introduce the concept of information bottleneck graphs.

As discussed in [NML+19], low-cost devices often suffer from additional phase noise due to cheap oscillators. This phase noise θ_k can be modeled as random-walk using the Wiener model [NML+19], i.e.,

$$\theta_k = \theta_{k-1} + \Delta\theta_k \tag{4.7}$$

where $\Delta\theta_k \sim \mathcal{N}(0, \sigma_\Delta^2)$. In turn, the phase of the received symbol is found as

$$\arg y_k = \arg\left\{e^{\phi_{s,k}} e^{\theta_k} + n_k\right\} = \arg\left\{e^{\psi_k} + n_k\right\} \tag{4.8}$$

where $n_k \sim \mathcal{N}(0, \sigma_N^2)$ and $h_k = 1$. For ease of notation, we introduce the auxiliary variable $\psi_k = \phi_{s,k} + \theta_k$. The multiple symbol detector aims to determine

$$\hat{u}_k = \arg\max_{u_k} p(u_k|\mathbf{y})\,. \tag{4.9}$$

The corresponding factor graph is shown in Fig. 4.5 and is fairly similar to the factor graph in Example 2.1. Furthermore, the schedule of the sum-product algorithm is also similar to the processing described in Example 2.2. Thus, the working principle of the forward-backward algorithm to solve Eq. (4.9) shall not be reviewed but can be found in great detail also in [KFL01]. In later comparisons, this approach is referred to as *conventional detector*.

The respective *information bottleneck detector* design requires the transformation of the factor graph from Fig. 4.5 into an information bottleneck graph. Without loss of generality, only the design of the forward path is presented as the backward path can be designed analogously.

Let us first consider the factor nodes f_{2k+1} at the bottom of Fig. 4.5 representing $p(y_k|\psi_k)$. These factor nodes can be transformed into an information-bottleneck quantizer which preserves the maximum amount of relevant information about ψ_k

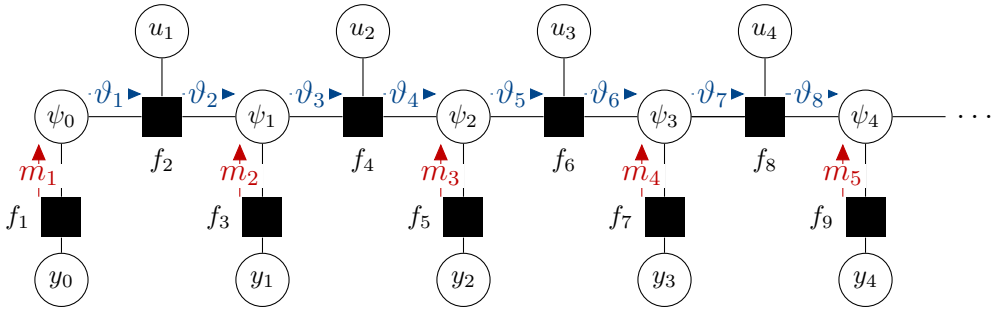


Figure 4.5: Factor graph of a MAP multiple differential symbol detector. The message passing schedule for the forward path is indicated by arrows.

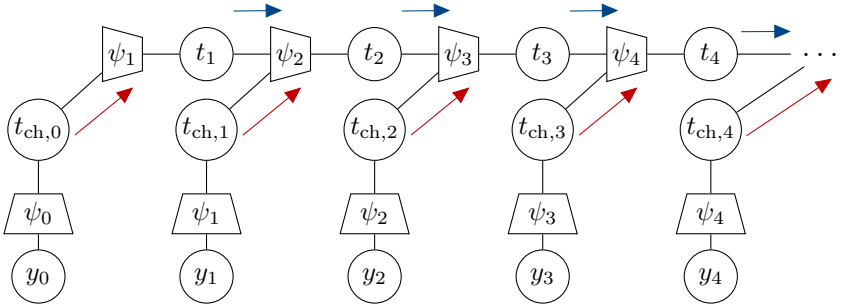


Figure 4.6: Information bottleneck graph of a MAP multiple differential symbol detector for the forward path.

(cf. Fig. 4.6). At the first glance, it might seem counter-intuitive to chose a relevant random variable which contains the actual phase plus undesired phase noise. However, the phase noise can only be eliminated by combining forward and backward path together. Thus, when considering the forward path, the relevant information about $\psi_k = \phi_{s,k} + \theta_k$ has to be preserved. Assuming an uniform prior on ψ_k, the joint distribution $p(y_k, \psi_k)$ can be computed directly. Please note that in this application, the intermediate relevant random variable is actually continuous. However, as discussed in [DS90] discretizing ψ_k to $\psi_k \in \left[0, \frac{2\pi}{L}, \ldots, 2\pi\right]^{\mathsf{T}}$ with $L = 8 \cdot |\mathsf{U}_k|$ does not harm the performance of the detection algorithm. In general, every information bottleneck algorithm discussed in Section 3.3 can be used to find the clustering $p(t_{\mathrm{ch},k}|y_k)$. In addition to $p(t_{\mathrm{ch},k}|y_k)$, the information bottleneck algorithm provides $p(\psi_k|t_{\mathrm{ch},k})$, i.e., the meaning of each cluster. Please note that this meaning will be required in the design of the next step. However, once all mutual-information-maximizing mappings were found, it is sufficient to pass only the cluster indices instead of the entire distribution. In turn, the actual processing can be implemented with a limited precision defined by the cardinality of compressed random variable T. As these indices can be nicely represented as integers, mutual-information-preserving message passing designed using the information bottleneck

is also termed *integer-based* message passing [LSB16b]. We discussed the close relation between the sum-product algorithm and integer-based message passing more formally in [LSB16a; LSB16b].

As described in Section 4.1, the shape of the information bottleneck node already indicates the direction of the flow of relevant information. However, this flow is emphasized by the blue and red arrows in Fig. 4.6. Here, the red arrow highlights the relevant information preserved by the quantizer and gathered in the current timestep. In M-PSK the amplitude of the received sample encodes no information, thus, only the phase is considered and quantized which simplifies the receiver design. Following [Gol05] and with further mathematical reformulation, the impact of AWGN on the phase of the received sample is computed as

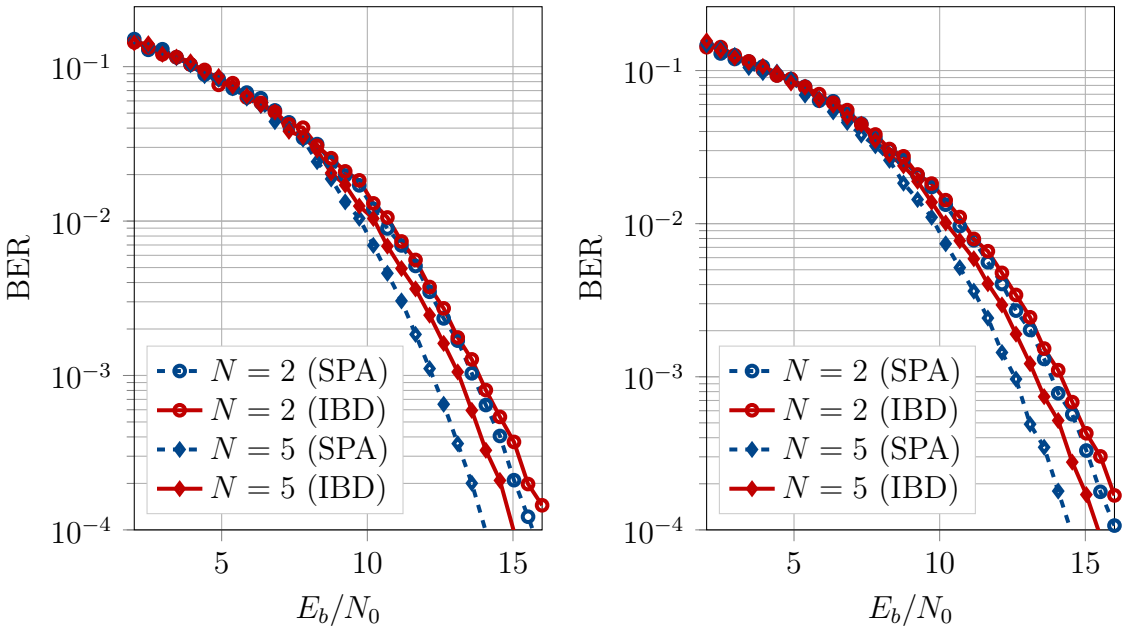
$$p(y_k, \psi_k) = \frac{1}{\pi} e^{-\frac{\cos(\psi_k)\sin(\psi_k)^2}{\sigma_N^2}} \cdot \frac{1}{2} e^{-\left(\frac{\cos(\psi_k)}{2\sigma_N^2}\right)^2} p(\psi_k) \tag{4.10}$$

by coordinate transformation. This joint distribution is fed into the information bottleneck algorithm to design the mutual information preserving channel output quantizer.

In contrast, the blue arrow indicates the relevant information about the current state gathered in the past. Extracting relevant information based on past observations, is the task of the information bottleneck nodes at the top of Fig. 4.6. This node fuses $p(\psi_{k-1}|t_{\mathrm{ch},k-1})$ and $p(\psi_{k-1}|t_{k-1})$ to obtain $p(\psi_k|t_k)$. Here, $p(\psi_k|t_k)$ is closely related to the analog distribution $p(\psi_k|y_0, \ldots, y_k)$ in the forward-backward algorithm (cf. Section 2.2 and Example 2.2). Internally, this information bottleneck also contains the transition distribution $p(\psi_k|\psi_{k-1}, u_k)$, i.e., except for $k = 0$ the overall joint distribution processed in the information bottleneck node yields

$$p(\psi_k, t_{k-1}, t_{\mathrm{ch},k-1}) = \sum_{\psi_{k-1}} \sum_{u_k} p(\psi_k|\psi_{k-1}, u_k) p(\psi_{k-1}|t_{\mathrm{ch},k-1}) p(\psi_{k-1}|t_{k-1}) p(u_k) \tag{4.11}$$

where in the forward path $p(u_k)$ is assumed to be unknown, i.e., uniformly distributed. Following the schedule of the sum-product algorithm, all factor nodes in Fig. 4.5 are replaced by information bottleneck nodes in Fig. 4.6, i.e., all arithmetic operations in the nodes are replaced by relevant-information preserving mappings. These mappings could be implemented, for instance, as lookup tables.



(a) $\sigma_\Delta^2 = 1°$ (b) $\sigma_\Delta^2 = 2°$

Figure 4.7: BER performance of the sum-product algorithm (SPA) and the information bottleneck detector (IBD) for MAP multiple-symbol detection for phase noise channels and differential 8-PSK modulation. The internal message representation of the coarsely quantized information bottleneck detector was 6 bits.

The computation of the joint distributions $p(\psi_{k-1}, t_k, t_{\mathrm{ch},k})$ to find relevant-information preserving mappings for the backward path is similar to Eq. (4.11), i.e.,

$$p(\psi_{k-1}, t_k, t_{\mathrm{ch},k}) = \sum_{\psi_k} \sum_{u_k} p(\psi_{k-1}|\psi_k, u_k) p(\psi_k|t_{\mathrm{ch},k}) p(\psi_k|t_k) p(u_k)\,. \tag{4.12}$$

In the following analysis of the presented application, 8-PSK and phase noise with $\sigma_\Delta^2 = 1°$ respectively $\sigma_\Delta^2 = 2°$ is considered as proposed in [DS90; NML+17; NML+19; Col12]. Furthermore, the length of the receive sequence was set to $N = 2$, i.e., classical differential demodulation and $N = 5$, i.e., MAP multiple-symbol detection. The uncoded BER over the AWGN variance σ_N^2 is plotted in Fig. 4.7. Please remember that the mutual-information-based, coarsely-quantized message passing algorithm (termed information bottleneck symbol detector (IBD) in the legend of Fig. 4.7) passes only integers and performs lookup operations. To construct the message mappings the KL-means algorithm was used. For comparison, the performance of the double-precision sum-product algorithm (SPA) is shown. In both cases, the received samples were quantized with 6 bits by the information bottleneck channel output quantizer. The information bottleneck symbol detector (IBD) passes only one 6 bit integers per message, whereas the beliefs in the sum-product algorithm were

represented by a probability vector $\mu \in \mathbb{R}^L$ with $L = 64$ double-precision entries. Interestingly and despite this remarkably coarse quantization of the exchanged messages from $64 \cdot 64 = 4096$ bits to 6 bits per message and the replacement of arithmetical node operations with table lookups, the presented integer-based detector shows only a fairly small performance gap of less than 0.5 dB at a BER of 10^{-3}. This gap could be reduced even further by increasing $|\mathcal{T}|$. Please note that as $\mu \in \mathbb{R}^L$, a comparable implementation would either require to represent each entry in μ with $6/L = 6/64 = 0.09$ bit, which is impossible or to perform hard-decision detection. As a remark, it should be noted that as expected, the multiple-symbol detector with $N = 5$ outperforms the classical differential demodulator which considers solely two adjacent symbols by around 2 dB.

4.2 Message Alignment

So far, we restricted ourselves to factor graphs where all exchanged beliefs were easily tractable. However, in communications, often so-called random graphs are needed to appropriately describe a certain scenario. Such settings often involve distributed wireless sensor networks [SLB18c; SK19], iterative turbo detection and decoding with interleavers but also channel decoding of the so-called irregular low-density parity check codes that are discussed in detail in Chapter 5. In general, the sum-product algorithm can be applied to these settings without further modifications. However, this work reveals that for the design of the respective coarsely quantized mutual-information-maximizing signal processing, an intermediate design step is integral. This intermediate step termed *message alignment* and the respective message alignment problem is discussed in detail in this section. We also published parts of this section in [SLB18c; LSB17; SLB18b].

4.2.1 Design of Coarsely Quantized Distributed Sensor Nodes using Message Alignment

Let us first motivate the message alignment problem considering a distributed wireless sensor network, as depicted in Fig. 4.8 and proposed in [SLB18c]. The sensors s_1 to s_K gather noisy observations of the same binary relevant random variable X. The measurement noise $\sigma_{N,k}^2$ of each sensor k differs. In general, the measurement model suffice

$$y_k = x + n_k \tag{4.13}$$

where $n_k \sim \mathcal{N}(0, \sigma_{N,k}^2)$. For each sensor, an independent information bottleneck quantizer is designed, described by $p_k(t|y)$, which preserves the maximum relevant

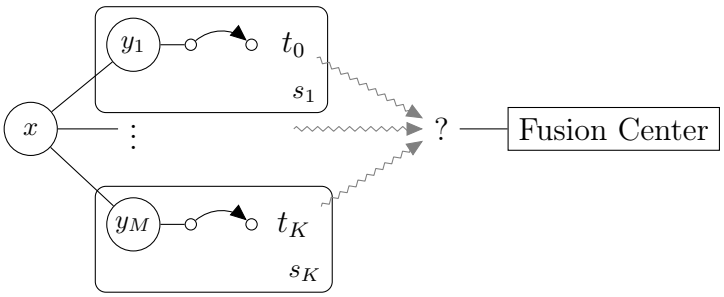


Figure 4.8: Several distributed nodes $s_1, ..., s_N$ measure the same quantity X and forward their compressed belief over a network to the fusion center, where the relevant quantity X is recaptured from the incoming beliefs.

information $\mathrm{I}(\mathsf{X};\mathsf{T}_k)$. The subscript k is introduced, indicating that the mapping $p_k(t|y)$ and the respective joint distributions $p_k(x,y)$ is associated with sensor k. These compressed observations are forwarded to a fusion center. The links are assumed to be error-free for a particular rate but the channel access of the senors is random and the transmitting sensor is unknown to the fusion center as depicted in Fig. 4.8. As the mappings $p_k(t|y)$ are different for each sensor, due to the different $\sigma^2_{N,k}$, also the meaning of the clusters $p_k(x|t)$ differ. This is exemplarily shown in Fig. 4.9a for three sensors with $\sigma^2_{N,1} = 2$, $\sigma^2_{N,2} = 0.5$ and $\sigma^2_{N,2} = 0.25$ and $|\mathcal{T}| = 16$ where the LLRs, i.e., $L_k(x|t) = \log \frac{p_k(\mathsf{X}=0|t)}{p_k(\mathsf{X}=1|t)}$ are plotted.

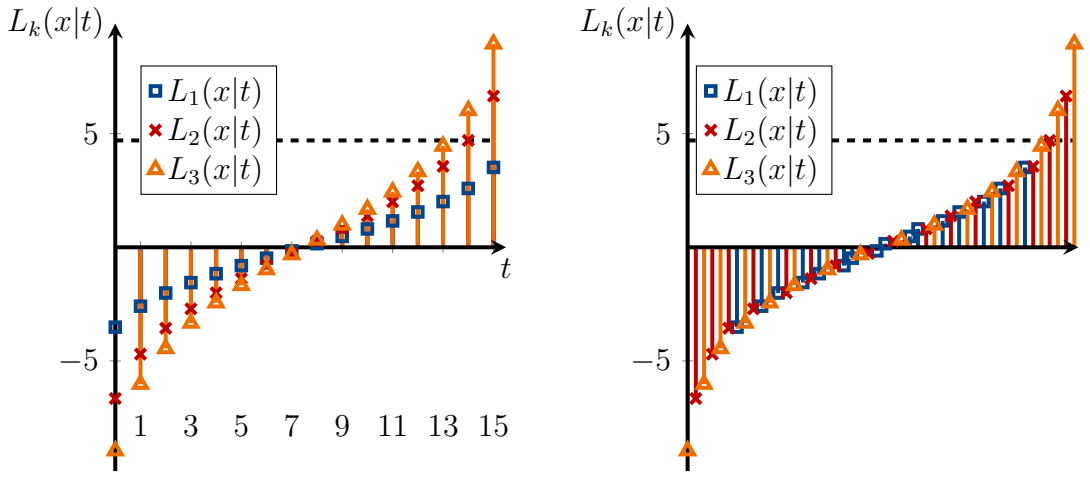
As the transmitting sensor is unknown to the fusion center, the sensor index k shall from now on be seen as a realization of the random variable K. Hence, for ease of notation, it is more convenient to rewrite the distributions $p_k(t|y)$, $p_k(x|t)$ and $p_k(x,y)$ as $p(t|y,k)$, $p(x|t,k)$ and $p(x,y|k)$, respectively and $\Pr(\mathsf{K}=k)$ is the probability that sensor k transmits a message to the fusion center. Not knowing the conveying sensor equals a marginalization over k, i.e.,

$$p(x|t) = \sum_k p(x|t,k)p(k)\,. \tag{4.14}$$

In turn, also the overall mutual information $\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{K})$ is reduced to $\mathrm{I}(\mathsf{X};\mathsf{T})$ due to the marginalization. Application of the chain rule yields

$$\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{K}) = \mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T}) + \mathrm{I}(\mathsf{X};\mathsf{T}) \tag{4.15}$$

assuming that $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T}) > 0$, where $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T})$ can be interpreted as the extra information gained over the relevant variable X by knowing K if the quantization index T is already known. In the absence of any further design steps, the respective mutual-information-based signal processing system would always loose $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T})$.



(a) LLR cluster meanings of the different sensors for the same cluster index t.

(b) LLRs of all sensors and clusters sorted.

Figure 4.9: Log-likelihood ratios of coarsely quantized sensor outputs for $\sigma^2_{N,1} = 2$, $\sigma^2_{N,2} = 0.5$ and $\sigma^2_{N,2} = 0.25$ and $|\mathcal{T}| = 16$.

Vividly, this loss in information can also be seen in Fig. 4.9a, where it can be observed that each sensor k maps a different LLRs onto the same cluster index t which is not surprising as all quantizers used the same sample space $\mathcal{T}$ but faced different input distributions. Interestingly, it can be observed that similar LLRs can be found in different clusters at different sensors. This is highlighted by the dashed horizontal line in Fig. 4.9a where one finds that the red cross marker for tuple $(k = 2, t = 14)$ and the orange triangle marker for $(k = 3, t = 13)$ have a smaller Euclidean distance than the red cross marker for tuple $(k = 2, t = 14)$ and the orange triangle marker for $(k = 3, t = 14)$. This observation also holds for other pairs of cluster indices as depicted in Fig. 4.9b where the LLRs of all sensors are sorted with respect to their magnitude. Hence, the meanings of the clusters are said to be *not aligned*.

In the following proposition, the difference between log-likelihood ratios is related to a more information-theoretic measure of similarity of meanings $p(x|t_i, k_j)$, i.e., the similarity of probability distributions, namely the Kullback-Leiber divergence.

Proposition 4.2.1. *Let us assume two meanings $p(x|t_i, k_j)$ and $p(x|t_n, k_m)$ where $i, n \in \{1, \ldots, |\mathcal{T}|\}$ and $j, m \in \{1, \ldots, N\}$. For a binary relevant random variable X,*

the Kullback-Leibler divergence $D_{KL}\left\{p(x|t_i,k_j)||p(x|t_n,k_m)\right\}$ *can be approximated as*

$$\approx \begin{cases} (1+p_0)\left(|L_m(x|t_n)|-|L_j(x|t_i)|\right), & L_j(x|t_i)\leq 0, L_m(x|t_n)\leq 0 \\ (2-p_0)\,|L_m(x|t_n)|-(p_0-1)\,|L_j(x|t_i)|\,, & L_j(x|t_i)\leq 0, L_m(x|t_n)> 0 \\ (1+p_0)\,|L_m(x|t_n)|-(2-p_0)\,|L_j(x|t_i)|\,, & L_j(x|t_i)> 0, L_m(x|t_n)\leq 0 \\ (2-p_0)\left(|L_m(x|t_n)|-|L_j(x|t_i)|\right), & L_j(x|t_i)> 0, L_m(x|t_n)> 0 \end{cases} \tag{4.16}$$

where $p_0 = p(\mathsf{X}=0|t_i,k_j)$, $L_m(x|t_n) = L(x|t_n,k_m)$, $L_j(x|t_i) = L(x|t_i,k_j)$.

Proof. See Appendix A.1 □

Hence, from Proposition 4.2.1, one concludes that approximately those meanings shall be assigned to similar clusters that have the smallest pairwise absolute difference considering their respective log-likelihood ratios. This observation leads directly to the *message alignment problem* [LSB17]. Starting from this graphical perspective on the message alignment problem and the connection between LLRs and the Kullback-Leibler divergence, it is also possible to investigate the message alignment problem from a more information-theoretical perspective.

In Eq. (4.15), it was shown that $\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{K})$ is the mutual information on X conveying the index of the transmitting sensor node *and* the cluster index together. From an information-theoretic point of view, $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T}) = 0$ implies that knowing the sensor node index K in addition to the cluster index yields no information gain about X. In this case, the meanings would be *perfectly aligned*. Thus, to solve the message alignment problem, a novel message mapping $p(z|t,k)$ for each sensor node is required, which minimizes $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T})$ such that exchanging the sensor index k *in addition* to the cluster index T yields approximately no information gain.

Reformulation of $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T})$ yields

$$\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{T}) = \sum_{t\in\mathcal{T}}\sum_{k} p(t,k) D_{\mathrm{KL}}\left\{p(x|t,k)||p(x|t)\right\} \tag{4.17}$$

$$= \mathbb{E}_{t,k}\left[D_{\mathrm{KL}}\left\{p(x|t,k)||p(x|t)\right\}\right]. \tag{4.18}$$

Introducing a mapping $p(z|t,k)$ can be interpreted as a reordering strategy that yields $p(x|z,k)$, i.e., the mapping of the aligned clusters. Please note that due to the introduction of the aligned random variable Z, $p(z|t,k)$ should be determined as

$$\min_{p(z|t,k)} \mathbb{E}_{z,k}\left[D_{\mathrm{KL}}\left\{p(x|z,k)||p(x|z)\right\}\right], \forall k \tag{4.19}$$

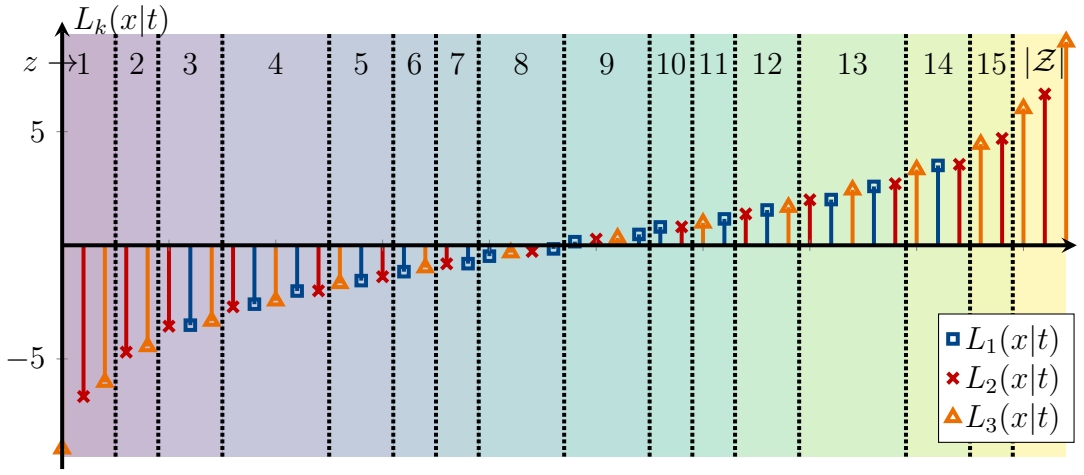


Figure 4.10: Illustration of a message alignment mapping $p(z|t,k)$.

to inherently achieve $\mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{Z}) \approx 0$ and thus

$$\mathrm{I}(\mathsf{X};\mathsf{K},\mathsf{Z}) = \mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{Z}) + \mathrm{I}(\mathsf{X};\mathsf{Z}) \approx \mathrm{I}(\mathsf{X};\mathsf{Z})\,. \tag{4.20}$$

As discussed in [SLB18b], Eq. (4.19) can be solved in an iterative manner using an algorithm closely related to the KL-Means algorithm, which consists of the following two update equations

$$z^* = \arg\min_z D_{\mathrm{KL}}\left\{p(x|t,k)||p(x|z)\right\},\ \forall (t,k) \in \mathcal{T}\times\mathcal{K} \tag{4.21}$$

$$p(x|z) = \frac{1}{p(z)} \sum_{(t,k)\in\mathcal{T}\times\mathcal{K}} p(z|t,k)\cdot p(x,t,k)\,. \tag{4.22}$$

The respective message alignment for the example depicted in Fig. 4.9b is shown in Fig. 4.10 where also the new aligned cluster labels z associated with the tuple (t,k) are highlighted.

Concluding the example and underlining the importance of message alignment, Fig. 4.11 shows the detection performance of the fusion center in the considered distributed scenario with random channel access. In Fig. 4.11, the detection error rate is plotted versus the averaged SNR $1/\bar{\sigma}_N^2$. Please note that the averaged SNR in this example is assumed as $\frac{1}{\bar{\sigma}_N^2} = \frac{1}{1/K\sum_k \sigma_{N,k}^2}$. The fusion center receives quantized observations from all sensors but in a random, unknown order. The optimum benchmark is the MAP-detector where the observations are not quantized and thus no alignment is needed (cf. green curve). The continuous MAP-detector in the fusion

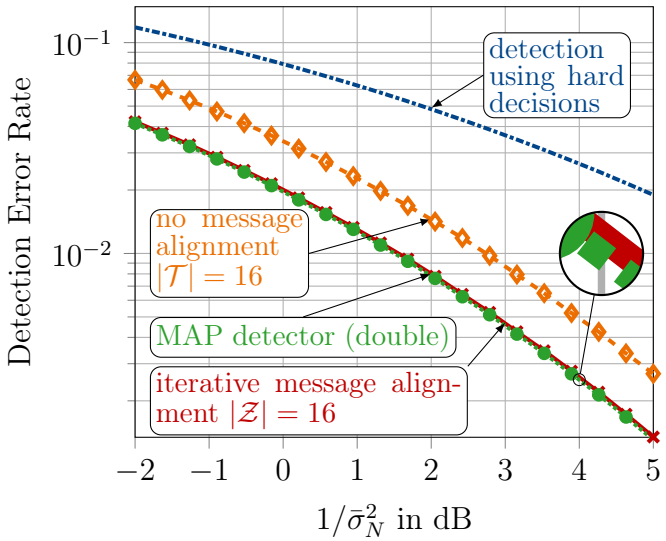


Figure 4.11: Detection error rate for two conventional benchmark systems and mutual-information-based signal processing units designed with and without message alignment.

center solves

$$x^* = \arg\max_x p(x|y_1, \ldots, y_k)\,. \tag{4.23}$$

This requires exact knowledge about k and also $\sigma_{N,k}^2$ at the fusion center. The second benchmark system relies on hard decisions, i.e., the observations are quantized with 1 bit (cf. blue curve). Two relevant-information-preserving systems with coarse quantization are evaluated. The first system is designed with message alignment and the second system is designed without message alignment. Here, one assumes that the fusion center solves

$$x^* = \arg\max_x p(x|t_1, \ldots, t_k) \tag{4.24}$$

where the actual processing is replaced by a lookup table. The red curve shows the performance of the system with message alignment where $|\mathcal{T}| = |\mathcal{Z}| = 16$. Interestingly, only a very small performance degradation compared to the double-precision benchmark system is observed. Further, the big gap between the system with and without message alignment reveals that the presented message alignment problem needs to be taken care of in the design of coarsely quantized mutual-information-based signal processing units to avoid significant performance degradation in random graphs. Further, the presented results show that the solution to the message alignment problem devised during this thesis can successfully align mismatched meanings of independently designed quantizers.

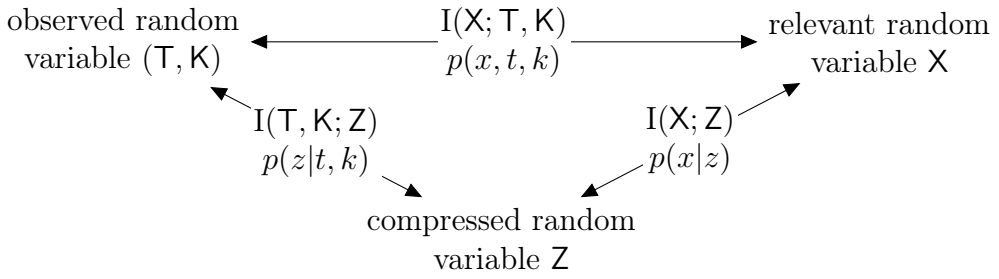


Figure 4.12: Illustration of the message problem as information bottleneck setup, where $\mathrm{I}(\mathsf{X};\mathsf{Z})$ denotes the relevant information, $\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{K})$ denotes the original mutual information and $\mathrm{I}(\mathsf{T},\mathsf{K};\mathsf{Z})$ denotes the compressed information.

4.2.2 Message Alignment as Information Bottleneck Problem

As introduced in the previous section, the message alignment problem arises naturally in many applications of coarsely-quantized mutual-information-based signal processing units. So far, message alignment has been regarded as an independent problem and has been described in information theoretic terms in Eq. (4.17). However, the message alignment objective from Eq. (4.20) can also be formulated as

$$\mathrm{I}(\mathsf{X};\mathsf{K},\mathsf{T}) = \mathrm{I}(\mathsf{X};\mathsf{K},\mathsf{Z}) = \mathrm{I}(\mathsf{X};\mathsf{K}|\mathsf{Z}) + \mathrm{I}(\mathsf{X};\mathsf{Z}) \overset{!}{\approx} \mathrm{I}(\mathsf{X};\mathsf{Z})\,. \tag{4.25}$$

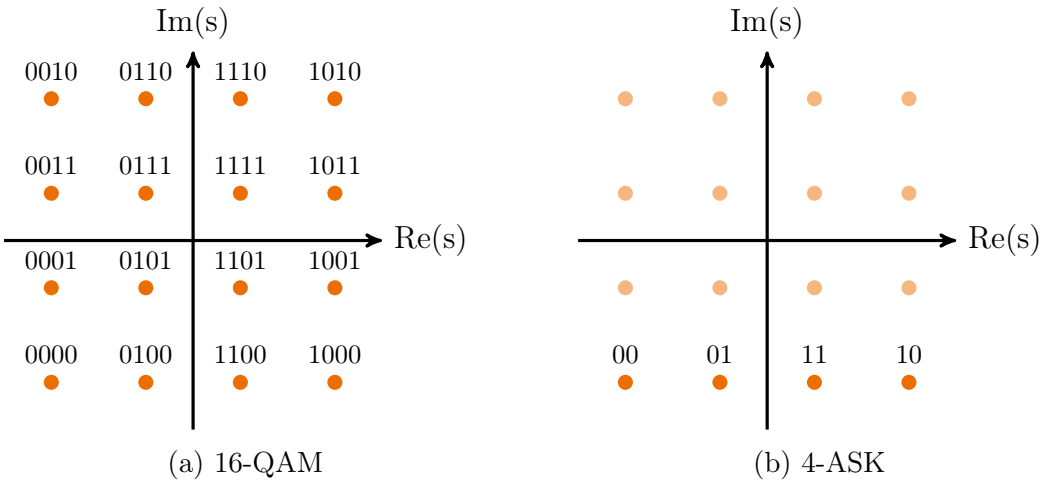
under the assumption that $p(z|t,k)$ describes a deterministic mapping. Thus, the optimal deterministic mapping $p(z|t,k)$ satisfies

$$\min_{p(z|t,k)} \mathrm{I}(\mathsf{X};\mathsf{K},\mathsf{T}) - \mathrm{I}(\mathsf{X};\mathsf{Z}) \tag{4.26}$$

which equals

$$\max_{p(z|t,k)} \mathrm{I}(\mathsf{X};\mathsf{Z})\,. \tag{4.27}$$

Interestingly, this formulation is inherently in line with the design objective of mutual-information-based signal processing units using the information bottleneck method (cf. Eq. (3.39)). Further, Eq. (4.27) reveals that the message alignment problem can be related to a multivariate information bottleneck problem, where the tuple (T,K) forms the multivariate observed random variable, X is the relevant random variable, and Z takes the role of the compressed random variable. Hence, in the notion of an information bottleneck problem, message alignment aims to preserve the maximum relevant information about X if the combination of cluster index *and* sensor index can only be represented by a single scalar random variable. This is also depicted in Fig. 4.12.



(a) 16-QAM (b) 4-ASK

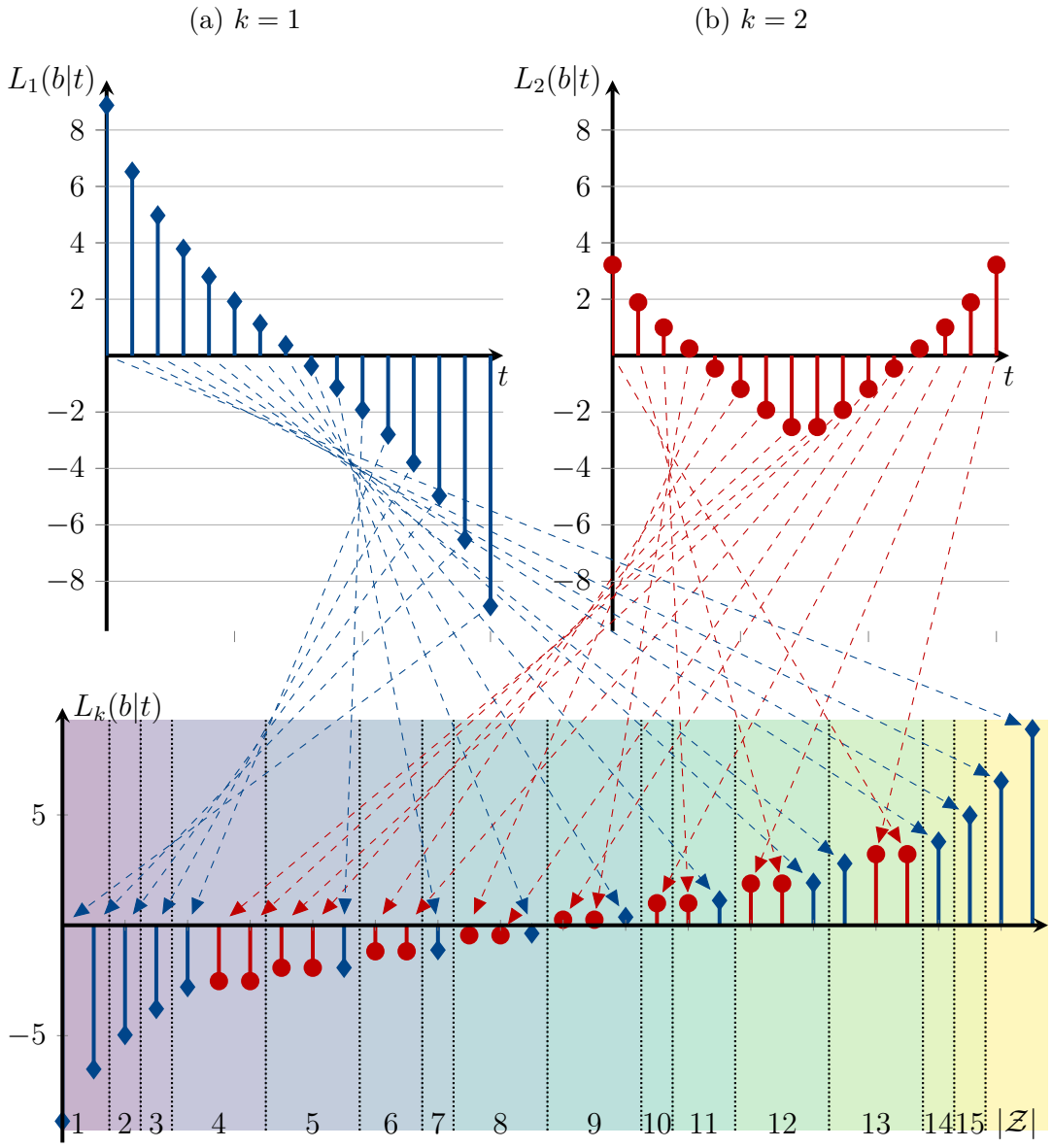
Figure 4.13: Illustration of a 16-QAM constellation which can be interpreted as two orthogonal 4-ASK constellations, i.e., one for the real and one for imaginary dimension.

This relation between the information bottleneck and message alignment is of large practical interest as it allows to apply the powerful tools and algorithm of the information bottleneck framework also to the message alignment problem.

4.2.3 Design of Bitwise-Mutual-Information-Based Channel Quantizer for Higher Order Modulation

The application in Section 4.2.1 introduced the message alignment problem in random factor graphs. Further, in Example 4.1, the design of relevant-information-preserving channel quantizers using the information bottleneck method was presented. In this section, it is shown that the message alignment problem often appears exactly at this interface between channel quantization and message-passing over factor graphs. Especially if bit-metric decoding with coarse quantization, as discussed in the next chapter, is paired with higher-order modulation schemes to increase the spectral efficiency of the transmission, the message alignment problem can appear.

Without loss of generality, let us consider the 16-QAM constellation depicted in Fig. 4.13a. Please note that any M-QAM constellation with Gray coding can also be treated as two orthogonal $\sqrt{M}$-ASK constellations, as shown in Fig. 4.13b [Gol05]. Thus, we restrict our further investigations to 4-ASK constellations. In bit-metric decoding, soft information about the bits and not about the symbols is fed into the subsequent channel decoder (cf. Section 2.4.3). In Example 4.1, a channel quantizer design that preserves the relevant information about the constellation symbol was presented. However, the demapping of the symbol onto the respective bit sequence



(c) All LLRs of the first bit and second bit ordered with respect to their magnitude.

Figure 4.14: Illustration of the message alignment problem for the design of a channel output quantizer for higher order modulation. The arrows indicate the optimum reordering of the meanings found using message alignment to ensure $\mathrm{I}(\mathsf{B};\mathsf{T},\mathsf{K}) \approx \mathrm{I}(\mathsf{B};\mathsf{Z})$.

reveals that the meaning conveyed by a cluster index differs for each information bit. For example, consider the two constellation points in the middle of the 4-ASK constellation in Fig. 4.13b encoded by the bit sequence $\mathbf{b}_1 = [b_1, \ldots, b_k, b_{\log_2(\sqrt{M})}] = [0, 1]$ and $\mathbf{b}_2 = [1, 1]$. These two bit sequences differ only in one bit, i.e., for the same received sample also the respective bit reliabilities of the two bits differ.

Fig. 4.14 shows the reliability of a particular bit associated with a cluster index

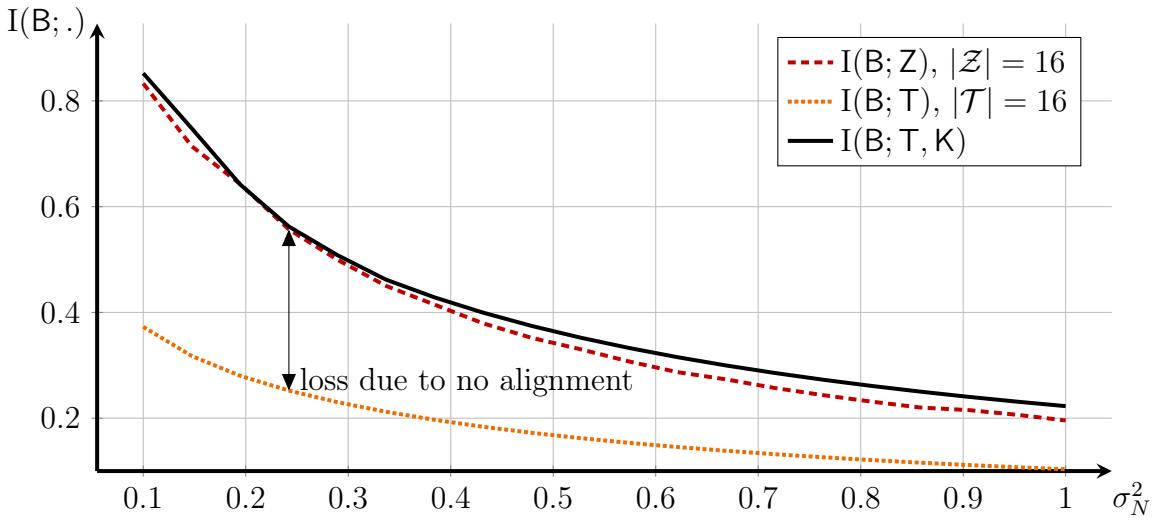


Figure 4.15: Loss in relevant information about the bit of a 4-ASK quantizer with (cf. $\mathrm{I}(\mathsf{B};\mathsf{Z})$) and without message alignment (cf. $\mathrm{I}(\mathsf{B};\mathsf{T})$) for $|\mathcal{Z}| = |\mathcal{T}| = 16$.

for the first bit (cf. Fig. 4.14a) and the second bit, respectively (cf. Fig. 4.14b). Designing an integer-based message-passing algorithm for this setting would be cumbersome as the input distribution used to design the lookup tables depends on the bit position in the symbol. Hence, the bit position has to be taken into account resulting in a largely increased complexity of the lookup table design. However, message alignment can also be used in this setting to harmonize the cluster meanings. Resorting to the observation from Section 4.2.2 that any message alignment problem can be treated as a multivariate information bottleneck problem similar to Fig. 4.12, in this application, the cluster index T and the bit index K serve as observed random variable and B is the relevant random variable.

Fig. 4.14c shows the equivalent joint optimization problem where all combinations of meanings, i.e., their respective LLRs, are plotted in ascending order. Further, Fig. 4.14c shows the reordered meanings after application of message alignment, also highlighted by the arrows connecting a cluster meaning before and after message alignment. Please note that in this scenario, an application of message alignment aims $\mathrm{I}(\mathsf{B};\mathsf{Z}) \approx \mathrm{I}(\mathsf{B};\mathsf{T},\mathsf{K})$. This is shown for transmission over an AWGN channel and different noise variances in Fig. 4.15. In Fig. 4.15, $\mathrm{I}(\mathsf{B};\mathsf{T},\mathsf{K})$ is shown as upper bound. It can be observed that a straightforward marginalization without message alignment would destroy nearly all relevant information about the bits preserved by the mutual-information-based symbol channel output quantizer. In turn, with message alignment nearly all relevant information can be preserved. The impact of message alignment on the performance of a subsequent channel decoder will be

investigated in more detail in the next chapter.

4.3 Summary

In this chapter, two integral tools for the mutual-information-based signal processing design devised in the scope of this thesis were presented. First, information bottleneck graphs were introduced. Exemplary, using a practically relevant application, i.e., maximum a-posterior multiple symbol detection for differential modulation over phase noise channels, the concept of the flow of relevant information was introduced. Further, it was outlined how information bottleneck graphs can be used to decompose complex untraceable inference tasks into smaller sub-problems, which can be turned into relevant-information-preserving building blocks using the information bottleneck method. The resulting message passing algorithm exchanged only integer values cluster indices, and the actual arithmetic operation degenerate to simple lookup operations. Provided numerical simulations showed that this system suffers only from a small performance degradation compared to double precision benchmark systems despite the very coarse quantization. It shall be emphasized that the information bottleneck graphs themselves are a very generic and universal tool with roots in factor graphs and high-dimensional inference. Throughout this thesis, information bottleneck graphs will be utilized in various applications but are not limited to those applications. The concept of information bottleneck which we first published in [LSB16a] was also reused by authors in literature for example in [KK17] or [ZK16] which proposed further interesting applications of information bottleneck graphs.

Second, message alignment was presented as another crucial tool. Starting from a scenario including distributed sensors nodes, an information-theoretic description of the message alignment problem was provided. Later, message alignment was connected to a multivariate information bottleneck problem, which is an integral finding of this chapter and will be exploited extensively in the next chapters. It was presented that message alignment can be interpreted as an additional mapping that reorders the clusters indices of independently designed information bottleneck mappings in a relevant-mutual-information-preserving manner. As a second practical problem, a coarsely quantized channel output quantizer design for higher-order modulation was considered. It was shown that to enable bit-metric decoding the meanings of clusters of different bits have to be aligned to prevent a significant loss in relevant information.

Chapter 5

Decoding Binary Low-Density Parity-Check Codes Using the Information Bottleneck Method

The previous chapters already sketched the potential of mutual-information-based signal processing design in terms of coarse quantization paired with negligible performance degradation. In this chapter, this design approach is extended to the most inevitable but also most resource-consuming part in baseband signal processing in modern communication systems, i.e., channel decoding. In particular so-called LDPC codes have received considerable attention and became part of several modern standards for instance IEEE 802.11 (WLAN) [IEE16], 802.16e (WiMAX)[IEE18], DVB-S2 [Bro14] and 5G [3GP18] due to their stunning error-correction capabilities [RSU01]. Although Gallager already proposed LDPC codes in [Gal62], it should last until the 1990s when MacKay [Mac99] rediscovered them. Due to the improved computing power compared to the 1960s, the decoding of LDPC codes became implementable and practically relevant. Significantly boosted by the pioneering work by Richardson et al. [RSU01; CFR+01], where it was shown that so-called irregular LDPC codes could approach Shannon's postulated achievable channel capacity within 0.0045 dB, LDPC codes turned into an inevitable building block of modern communication systems. First, this chapter reviews state-of-the-art decoding of LDPC codes using the sum-product algorithm and the so-called *Tanner* graph. Further, it will be outlined that message-passing decoding of LDPC codes is computationally very complex. To fully exploit the error-correction capabilities of these codes with conventional decoders, high precision belief propagation is required.

However, increased data rates and support for low-latency communication place

tight constraints on the computational complexity of channel decoders especially in the most current 5G communications standard. Here, the classical decoding approach comes with two main challenges. First, the messages that carry the soft information are exchanged iteratively between the check nodes and variable nodes in the Tanner graph and require a high resolution to convey the belief on a codeword bit precisely. As a result, the message transfer becomes a major bottleneck as the block length increases [KHW18]. Second, realizing the correct check node operation, i.e., the box-plus operation requires several computationally complex operations.

Thus, practical LDPC decoder implementations use message-passing decoding with finite precision and approximated node operations to reduce the computational burden compared to double-precision belief propagation decoding [JDE+05]. Often, the precision has to become very coarse as complexity is more severely constrained. In turn, also the performance of finite-precision state-of-the-art LDPC decoders degrades as the precision becomes more coarse.

Pursing the information bottleneck method and the idea of mutual-information-based design, this chapter presents the so-called *information bottleneck decoder*. This decoding approach allows us to coarsely quantize the exchanged messages and replace the complex arithmetic node operations with simple lookup operations in optimized message mappings. In more detail, this chapter contains the following main contributions:

1. Review of the information bottleneck decoder design for *regular* LDPC codes proposed by Lewandowsky et al. in [LB18] (Section 5.2).

2. Extension of the decoder construction framework from [LB18] to arbitrary irregular LDPC codes (Section 5.3).

   - Derivation of underlying information-theoretic problem formulation and explanation how the intermediate optimization technique called message alignment can be incorporated.

   - Detailed investigation of the designed decoders for a large variety of code-rates, different bit-width and code length from different standards.

   - The presented decoders are shown to perform very close, i.e., within 0.1 dB, to double-precision belief propagation despite the coarse quantization of the messages and the node operations being simple lookup operations.

3. Extension of the information bottleneck decoder design to enhanced LDPC code structures including puncturing and rate-compatibility, in particular for

protograph-based raptor-like (PBRL) LDPC codes, which is inevitable for error correction codes in modern communication standards (Section 5.4).

- Reformulation of message alignment as an information bottleneck problem, facilitating designs for irregular LDPC codes. The new interpretation of message alignment allows reuse of tables across the entire rate range allowing a compact rate-compatible IB decoder for an entire PBRL code family.
- Investigation of several message alignment implementations and their effect on the decoder performances.
- Detailed investigation of the designed decoders where 4-bit information bottleneck decoders for a PBRL code family designed using the devised construction approach outperforms a 6-bit normalized-min-sum decoder and performs very close to double-precision belief propagation decoding.

4. Investigation of the hardware complexity of the proposed information bottleneck decoder (Section 5.6).
   - Optimization of the lookup table scheduling and introduction of a novel tree-like lookup pattern. In turn, the relation between the number of lookup operations required per iteration and the node degree changes from linear to logarithmic, which is important for LDPC codes with highly irregular node degree distributions.
   - Derivation and investigation of alternative implementations of the mutual-information-maximizing lookup tables.

Interestingly, the idea to design mutual-information based channel decoders dates back to Thorpe [Tho03a], where only a rough sketch of this decoding approach was presented without generic algorithms or enhanced design concepts. However, several streams towards a coarsely quantized LDPC decoder were pursued with versatile assumptions and approximations always tailored to specific needs. A collection of such approaches from the literature is schematically illustrated in a decision tree in Fig. 5.1. Furthermore, the actual branch pursued in this thesis is additionally highlighted in Fig. 5.1. At the end of this chapter, the very generic design-concept using the information bottleneck method presented in this thesis is used as a starting point and considered as baseline to discuss, evaluate and relate the other decoding approaches summarized in Fig. 5.1. Please note that although not explicitly shown in Fig. 5.1, the finite-alphabet iterative decoding (FAID) approach presented in [PDD+11; CZD+14; DVP+13; PDD+13] which also deals with lookup table based

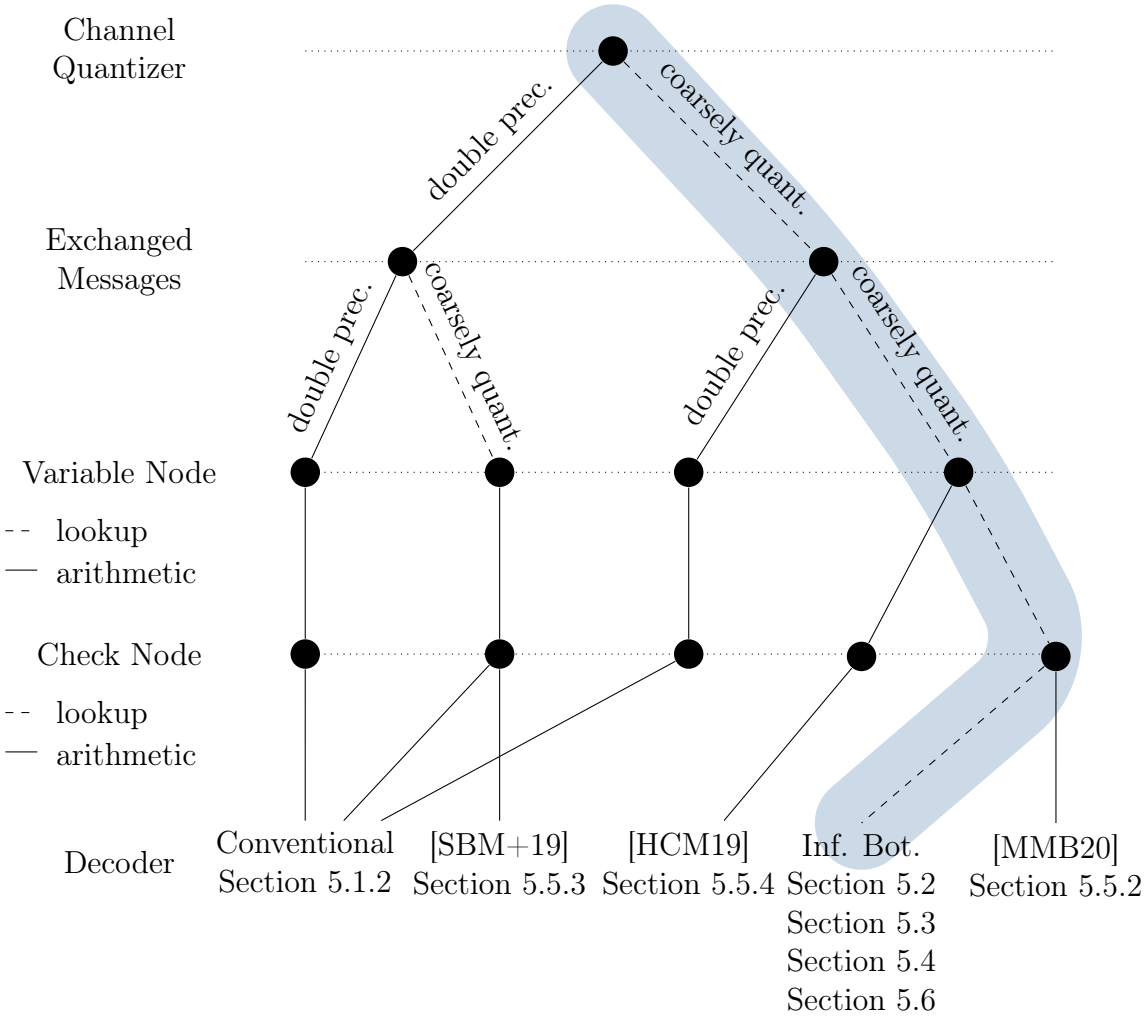


Figure 5.1: Tree of a selection of available LDPC decoders in literature depending on the choice of node operations, channel quantizer resolution and precision of the exchanged messages which also serves as an illustration of the structure of this chapter. The information bottleneck decoding approach devised in this thesis is highlighted.

LDPC decoding for regular LDPC codes, will be reviewed in the context of regular information bottleneck decoders in Section 5.2 and Section 5.5.1.

5.1 Preliminaries on Binary Low-Density Parity-Check Codes

In general, binary LDPC codes belong to the class of linear block codes as described in Section 2.4. In particular, LDPC codes are defined by a *sparse* $N_c \times N_v$ parity check matrix $\mathbf{H}$, where N_v denotes the number of so-called *variable nodes* and N_c denotes the number of *check nodes*, respectively. These terms follow directly from

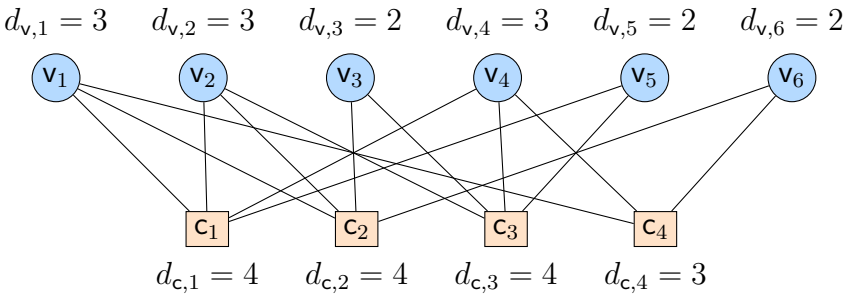


Figure 5.2: Tanner graph for an irregular LDPC code.

a graphical perspective on LDPC codes employing Tanner graphs [Joh10]. Tanner graphs are bipartite graphs with N_v variable nodes, N_c factor nodes, which are called check nodes in the context of Tanner graphs. Such a Tanner graph is exemplarily shown in Fig. 5.2 for the exemplary parity check matrix

$$\mathbf{H}^{\mathsf{T}} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{5.1}$$

An edge connecting a variable node v_j and a check node c_i exists, if the parity-check matrix element $h_{i,j} = 1$. The number of edges associated with a respective node defines its *degree*, i.e., $d_{\mathsf{v},j}$ and $d_{\mathsf{c},i}$ are the node degrees of v_j and c_i, respectively.

5.1.1 LDPC Code Ensembles

As shown later, for the asymptotic analyses of LDPC codes and the design of information bottleneck decoders, it is often convenient to resort to the so-called code ensembles [Gal62]. Instead of considering a particular LDPC code with parity check matrix $\mathbf{H}^{\mathsf{T}}$, the code ensembles comprise all realizations of LDPC codes with one specific check node degree distribution and variable node degree distribution [Joh10].

In literature, one distinguishes between a node and edge perspective on the ensemble characteristics. However, as shown later, it is more appropriate to focus on the *degree distribution from an edge perspective*, referred to as edge-degree distribution in this thesis. Thus, the connections between the two sets of nodes are characterized probabilistically by the polynomials [RL09]

$$\lambda(\zeta) = \sum_{j=1}^{d_{\mathsf{v}}^{\max}} \lambda_j \zeta^{j-1} \qquad \rho(\zeta) = \sum_{i=2}^{d_{\mathsf{c}}^{\max}} \rho_i \zeta^{i-1}, \tag{5.2}$$

where λ_j denotes the fraction of edges connected to variable nodes with degree j and ρ_i denotes the fraction of edges connected to check nodes with degree i. Further, $d_c^{\max}$ and $d_v^{\max}$ denote the maximum check node degree and variable node degree respectively.

The code rate can be computed based on the edge-degree distributions as [Joh10]

$$R = \frac{K}{N} = 1 - \frac{\sum_{i=2}^{d_{\mathsf{c}}^{\max}} \rho_i / i}{\sum_{j=1}^{d_{\mathsf{v}}^{\max}} \lambda_j / j}\,. \tag{5.3}$$

Example 5.1: Analysis of an Irregular LDPC Code

For the exemplary LDPC code with the parity-check matrix given in Eq. (5.1), the coefficients λ_j of $\lambda(\zeta)$ are determined as

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_{\max}]^{\mathsf{T}} = \left[0, \frac{6}{15}, \frac{9}{15}\right]^{\mathsf{T}} \tag{5.4}$$

as highlighted in Fig. 5.2. Similarly, for the check nodes, the coefficients ρ_i of $\rho(\zeta)$ are found as

$$\boldsymbol{\rho} = [\rho_2, \rho_3, \ldots, \rho_{\max}]^{\mathsf{T}} = \left[0, \frac{3}{15}, \frac{12}{15}\right]^{\mathsf{T}}\,. \tag{5.5}$$

Using, Eq. (5.3) the respective code rate of the LDPC code is determined as

$$R = \frac{K}{N} = 1 - \frac{\sum_{i=2}^{d_{\mathsf{c}}^{\max}} \rho_i / i}{\sum_{j=1}^{d_{\mathsf{v}}^{\max}} \lambda_j / j} \tag{5.6}$$

$$= 1 - \frac{0 \cdot \frac{1}{2} + \frac{3}{15} \cdot \frac{1}{3} + \frac{12}{15} \cdot \frac{1}{4}}{0 \cdot \frac{1}{1} + \frac{6}{15} \cdot \frac{1}{2} + \frac{9}{15} \cdot \frac{1}{3}} \tag{5.7}$$

$$= 1 - \frac{\frac{4}{15}}{\frac{6}{15}} = 1 - \frac{4}{6} \tag{5.8}$$

$$= \frac{1}{3}\,. \tag{5.9}$$

If the node degree is d_{v} for all variable nodes and d_{c} for all check nodes, the LDPC code is called *regular* and otherwise *irregular*. For a regular LDPC code Eq. (5.3) simplifies to [Joh10]

$$R = 1 - \frac{d_{\mathsf{v}}}{d_{\mathsf{c}}}\,. \tag{5.10}$$

Another important property of a particular LDPC code within the code ensemble is the length of the shortest cycle in the Tanner graph, i.e., the so-called *girth* [Joh10]. As explained later in Section 5.1.2 and Section 5.1.3, to analyze and decode LDPC

codes it is assumed that the exchanged messages are independent. For a large number of iterations, this assumption is typically violated due to cycles in the Tanner graph. Thus, it is beneficial to avoid short cycles in the code design resulting in a large girth [Joh10].

5.1.1.1 Structured LDPC Code Ensembles

If the code ensemble is only described by the edge-degree distributions and no further constrains are employed, the ensemble is said do be *unstructured*. Thus the entire Tanner graph is constructed to satisfy certain code characteristics like the degree distribution. To ensure a good error correction performance under iterative message passing decoding, a large girth in inevitable. It is hard to guarantee a sufficiently large girth in the unstructured scenario as the construction methods are based on randomness. Thus, Thorpe [Tho03b; DDJ+09] introduced LDPC codes constructed from a protograph. A protograph is a small Tanner graph that describes the connectivity of the overall LDPC code Tanner graph and serves as a blueprint [Tho03b]. Based on this protograph, the performance of the overall LDPC code is easily analyzed. A copy and permute operation applied to the protograph termed *lifting* obtains the full LDPC parity check matrix [Tho03b]. In turn, there exists some kind of *structure* in the parity check matrix.

Another practical purpose of structured ensembles relates to the encoding process. For arbitrary unstructured LDPC codes, it is often difficult to derive efficient encoders. One common approach is to apply Gaussian elimination to determine a systematic generator matrix that can be used for encoding. However, in contrast to the parity check matrix, this generator matrix is often not sparse but relatively dense, which increases the computational complexity of the encoding process. Hence, it is beneficial to impose more structure on an LDPC code to allow efficient encoding. Mainly, so-called *repeat-accumulate codes* constitute an essential class of practical LDPC codes due to their low complexity encoding, which are also part of many practical communication standards like WLAN and DVB-S2 [Joh10].

In Section 5.3, coarsely quantized LDPC decoders designed using the information bottleneck for both structured and arbitrary unstructured LDPC code ensembles are proposed.

Protograph-Based-Raptor-Like (PBRL) LDPC Codes

In the 5G communication standard, so-called PBRL LDPC codes first proposed in [CVD+15], are considered for enhanced mobile broadband. This family of codes

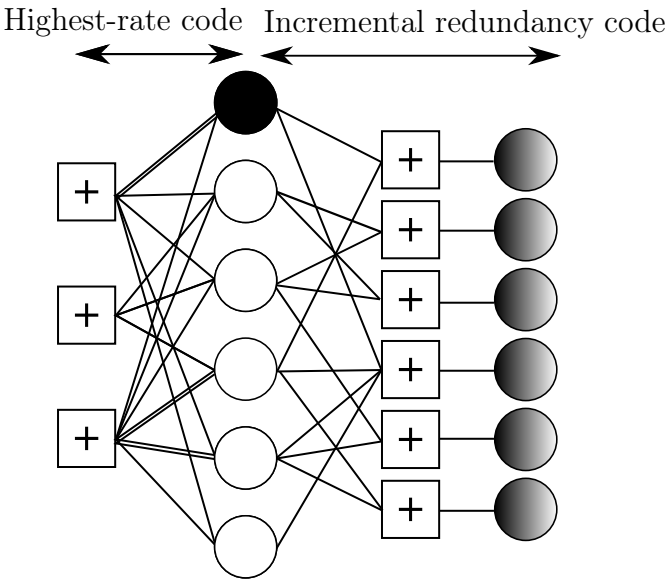


Figure 5.3: Protograph of a PBRL LDPC code where the shaded node depicts a punctured variable node in the highest-rate code and the partial shade indicates that degree-one variable nodes can be punctured to adapt the rate.

builds upon the protograph concept but is tailored for another integral characteristic of modern channel coding schemes, i.e., *rate-compatibility*. Such rate-compatible LDPC codes can easily be paired with hybrid automated repeat request for efficient retransmission and support a large range of code rates. Starting from a common base matrix which defines the *highest-rate* code (HRC) protograph, an *incremental-redundancy* code (IRC) protograph is derived. Fig. 5.3 shows the protograph structure of a PBRL code as described in [CVD+15; RDW19]. In general, the protomatrix of the protograph shown in Fig. 5.3 is given as

$$\mathbf{H}^{\mathsf{T}} = \begin{bmatrix} \mathbf{H}_{\mathrm{HRC}}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{H}_{\mathrm{IRC}}^{\mathsf{T}} & \mathbf{I} \end{bmatrix}, \tag{5.11}$$

where $\mathbf{H}_{\mathrm{HRC}}$ denotes the parity check matrix of the highest rate code, $\mathbf{H}_{\mathrm{IRC}}$ denotes the parity check matrix of the incremental redundancy code and $\mathbf{0}$ and $\mathbf{I}$ represent the all-zeros matrix and the identity matrix, respectively. Hence, the overall PBRL protograph can be interpreted as the concatenation of the highest-rate LDPC code and a low-density generator matrix (LDGM) code having the systematic generator matrix [CVD+15]

$$\mathbf{G}_{\mathrm{LDGM}} = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}_{\mathrm{IRC}} \end{bmatrix}.$$

In general, a greedy procedure determines the rows of $\begin{bmatrix} \mathbf{H}_{\mathrm{IRC}}^{\mathsf{T}} & \mathbf{I} \end{bmatrix}$ corresponding to the *incremental-redundancy* code such that the decoding threshold for the respective

code rate is minimized. As a design principle of PBRL codes, one or two variable nodes in the HRC remain punctured for all supported code rates [CVD+15], which is indicated by the HRC variable node filled in black in Fig. 5.3. For the highest code rate, all of the IRC variable nodes are punctured. The IRC provides lower rates as more of its variable nodes are transmitted, starting from the top, i.e., degree-one variable nodes are added to the protograph as the rate is lowered. Hence, a degree-one variable node might be punctured depending on the code rate, as indicated by the partial shade of the degree-one variable nodes.

Often, it is convenient to identify the HRC for a family of PBRL codes by the triplet

$$(n_v, n_c, n_p)$$

where n_v denotes number of variable nodes, n_c denotes the number of check nodes and finally n_p denotes the number of punctured nodes in the highest rate code. Given this triplet and assuming $\mathbf{H}_{\text{HRC}}$ is full rank, the supported rates of the PBRL code are found as

$$R_c = \frac{n_v - n_c}{n_v - n_p + i} \tag{5.12}$$

for $i > 0$ indicating the number of unpunctured degree-one variables nodes of the incremental redundancy code.

A more detailed introduction to PBRL LDPC codes can be found in [CVD+15; RDW19].

Encoding of PBRL codes is similar to Raptor codes and resorts to a very efficient two-step encoding process. First, the information bits $\mathbf{u}$ are mapped onto so-called *precoded* symbols as

$$\mathbf{c}' = \mathbf{G}_{\text{HRC}}\mathbf{u} \tag{5.13}$$

where $\mathbf{G}_{\text{HRC}}$ denotes the generator matrix of the precode, which is derived from the parity check matrix of the highest-rate code, i.e., $\mathbf{H}_{\text{HRC}}$. Also the second encoding step, involves only exclusive-or operations to compute

$$\mathbf{c} = \mathbf{G}_{\text{LDGM}}\mathbf{c}' = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}_{\text{IRC}} \end{bmatrix} \mathbf{c}' = \begin{bmatrix} \mathbf{c}' \\ \mathbf{H}_{\text{IRC}}\mathbf{c}' \end{bmatrix}. \tag{5.14}$$

Section 5.4 addresses the issue of designing information bottleneck decoders that accommodate the puncturing that is inherent to PBRL code families.

5.1.2 Belief-Propagation Decoding

Decoding LDPC codes resorts to employing the sum-product algorithm on the Tanner graph representation of LDPC codes. However, in contrast to previous applications of the sum-product, as discussed in Section 2.2.1, Example 4.1 and Section 4.1.1, decoding of LDPC codes is an instance of *loopy belief propagation* as the graph is not cycle-free [Bis09].

As introduced in Section 2.4.3 for bit-metric decoding, BPSK transmission, and discrete memory-less channels, the decoding problem can be formalized as

$$\hat{b_j} = \arg\max_{b_j} p(b_j|\mathbf{y})\,. \tag{5.15}$$

Following the factor-graph notation from Section 2.2 each code bit is represented by a variable node, and the factor nodes are termed check nodes. The name *check* node already suggests the local function executed at this node, i.e., evaluating a single-parity check equation described by the respective rows of $\mathbf{H}^{\mathsf{T}}$.

In an iterative decoder, the extrinsic information is exchanged iteratively, and variable node and check node updates are performed, starting with a check node update. This iterative procedure ends either if all check node operations are satisfied or a maximum number of decoding iterations $i_{\max}$ is reached. As practically LDPC codes typically do not have cycle-free graphs, the found posterior distribution is only an approximation of the true distribution. Hence, a convergence of the iterative sum-product algorithm to the correct solution is not guaranteed.

Please remember that the exchanged messages in the original sum-product algorithm are proportional to probability distributions also termed beliefs. Hence, first belief propagation decoding is introduced in the *probability domain*. This domain is vital for the generalization of belief-propagation decoding to non-binary LDPC codes discussed in Chapter 6. Second, the probability domain is often the starting point to derive approximations and simplifications of binary LDPC code decoding as presented in Section 5.5.4.

Let us denote the message from a variable node v_j to check node c_i representing a belief on code bit b_j as $m_{\mathsf{v}_j\to\mathsf{c}_i}(b_j)$. Likewise, let $m_{\mathsf{c}_i\to\mathsf{v}_j}(b_j)$ denote the message from a check node c_i to variable node v_j and thus the belief about b_j conveyed by check node c_i. Again, $N(\mathsf{c}_i)$ denotes the neighborhood of check node c_i, i.e., all connected variables nodes to the check node c_i. Likewise, $N(\mathsf{v}_j)$ denotes the neighborhood of variable node v_j respectively.

5.1.2.1 Conventional Belief-Propagation Variable Node Operation

The variable node update follows directly from the sum-product algorithm and can be written as

$$m_{\mathsf{v}_j \to \mathsf{c}_i}(b_j) = \prod_{\mathsf{c}_l \in N(\mathsf{v}_j)\setminus\{\mathsf{c}_i\}} m_{\mathsf{c}_l \to \mathsf{v}_j}(b_j)\,. \tag{5.16}$$

In state-of-the-art belief propagation decoding of binary LDPC codes, the soft-information is represented by LLRs. At a variable node v_j, two types of LLR exist, the *channel* log-likelihood ratio $L_{\mathrm{ch},j}(b_j|y)$ and the exchanged log-likelihood ratios $L_{\mathsf{c}_l \to \mathsf{v}_j}(b_j)$. For ease of notation, we drop the argument and denote the log-likelihood ratios as $L_{\mathrm{ch},j}$ and $L_{\mathsf{c}_i \to \mathsf{v}_j}$, respectively. The log-likelihood ratios are determined as

$$L_{\mathrm{ch},j} = \log\left(\frac{p(\mathsf{B}_j = 0|y_j)}{p(\mathsf{B}_j = 1|y_j)}\right) \qquad L_{\mathsf{c}_i \to \mathsf{v}_j} = \log\left(\frac{m_{\mathsf{c}_i \to \mathsf{v}_j}(\mathsf{B}_j = 0)}{m_{\mathsf{c}_i \to \mathsf{v}_j}(\mathsf{B}_j = 1)}\right)\,. \tag{5.17}$$

In the *log-domain*, Eq. (5.16) equals a summation of LLRs. That is, all incoming LLRs are summed up except the message received over the edge for which *extrinsic information* is to be generated. Thus, Eq. (5.16) in the log-domain is formalized as

$$L_{\mathrm{ext},j} = L_{\mathsf{v}_j \to \mathsf{c}_i} = L_{\mathrm{ch},j} + \sum_{\mathsf{c}_l \in N(\mathsf{v}_j)\setminus\{\mathsf{c}_i\}} L_{\mathsf{c}_l \to \mathsf{v}_j}\,. \tag{5.18}$$

Further, for the final decision on the underlying code bit, i.e., to compute the a posteriori estimate $L_{\mathrm{app},j}$ all messages are considered, which yields

$$L_{\mathrm{app},j} = L_{\mathrm{ch},j} + \sum_{\mathsf{c}_l \in N(\mathsf{v}_j)} L_{\mathsf{c}_l \to \mathsf{v}_j}\,. \tag{5.19}$$

5.1.2.2 Conventional Belief-Propagation Check Node Operation

For a check node c_i with degree d_c, the *outgoing* message to variable node v_j is computed as [Joh10]

$$\begin{aligned} m_{\mathsf{c}_i \to \mathsf{v}_j}(b_j) &= \sum_{\mathsf{v}_1,\mathsf{v}_2,\ldots,\mathsf{v}_{d_c} \in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} f(\mathsf{v}_1,\mathsf{v}_2,\ldots,\mathsf{v}_{d_c}) \prod_{\mathsf{v}_l \in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} m_{\mathsf{v}_l \to \mathsf{c}_i}(b) \\ &= \sum_{\mathsf{v}_1,\mathsf{v}_2,\ldots,\mathsf{v}_{d_c} \in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} \delta(b_1 \oplus b_2 \oplus \cdots \oplus b_{d_c}) \prod_{\mathsf{v}_l \in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} m_{\mathsf{v}_l \to \mathsf{c}_i}(b) \\ &= \frac{1}{2} + (-1)^{b_j}\frac{1}{2} \prod_{\mathsf{v}_l \in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} 1 - 2 \cdot m_{\mathsf{v}_l \to \mathsf{c}_i}(\mathsf{B} = 1) \end{aligned} \tag{5.20}$$

where the notation $N(\mathsf{c}_i)\backslash\{\mathsf{v}_j\}$ underlines that only *extrinsic information* shall be generated in the node update. That is, to compute the outgoing edge, all information conveyed by incoming messages received over all edges, except the edge for which the outgoing message is generated, is considered. Further, the local function $f(\mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_{d_c})$ is replaced by the Kronecker delta function $\delta(b_1 \oplus b_2 \oplus \cdots \oplus b_{d_c})$ which is only one if the parity check equation is satisfied and zero otherwise.

Again, it is more common for binary LDPC codes to work in the log-domain, i.e., all exchanged messages are represented by a scalar double-precision representative, the LLR. We denote the extrinsic LLR for check node c_i forwarded to variable node v_i as $L_{\text{ext},i}$ which is computed as [Joh10]

$$\begin{aligned} L_{\text{ext},i} = L_{\mathsf{c}_i \to \mathsf{v}_j} &= \log\left(\frac{m_{\mathsf{c}_i \to \mathsf{v}_j}(\mathsf{B}_j = 0)}{m_{\mathsf{c}_i \to \mathsf{v}_j}(\mathsf{B}_j = 1)}\right) \\ &= \log\left(\frac{\frac{1}{2} + \frac{1}{2}\prod_{\mathsf{v}_l \in N(\mathsf{c}_i)\backslash\{\mathsf{v}_j\}} 1 - 2 \cdot m_{\mathsf{v}_l \to \mathsf{c}_i}(\mathsf{B} = 1)}{\frac{1}{2} - \frac{1}{2}\prod_{\mathsf{v}_l \in N(\mathsf{c}_i)\backslash\{\mathsf{v}_j\}} 1 - 2 \cdot m_{\mathsf{v}_l \to \mathsf{c}_i}(\mathsf{B} = 1)}\right) \\ &= 2\text{atanh}\left(\prod_{\mathsf{v}_l \in N(\mathsf{c}_i)\backslash\{\mathsf{v}_j\}} \tanh\left(\frac{L_{\mathsf{v}_l \to \mathsf{c}_i}}{2}\right)\right) . \end{aligned} \tag{5.21}$$

This can be further reformulated using the box-plus operation [HOP96]. The box-plus operation $\boxplus$ of two LLRs L_1 and L_2 is defined as

$$L_1 \boxplus L_2 = \log \frac{e^{L_1 + L_2} + 1}{e^{L_1} + e^{L_2}} . \tag{5.22}$$

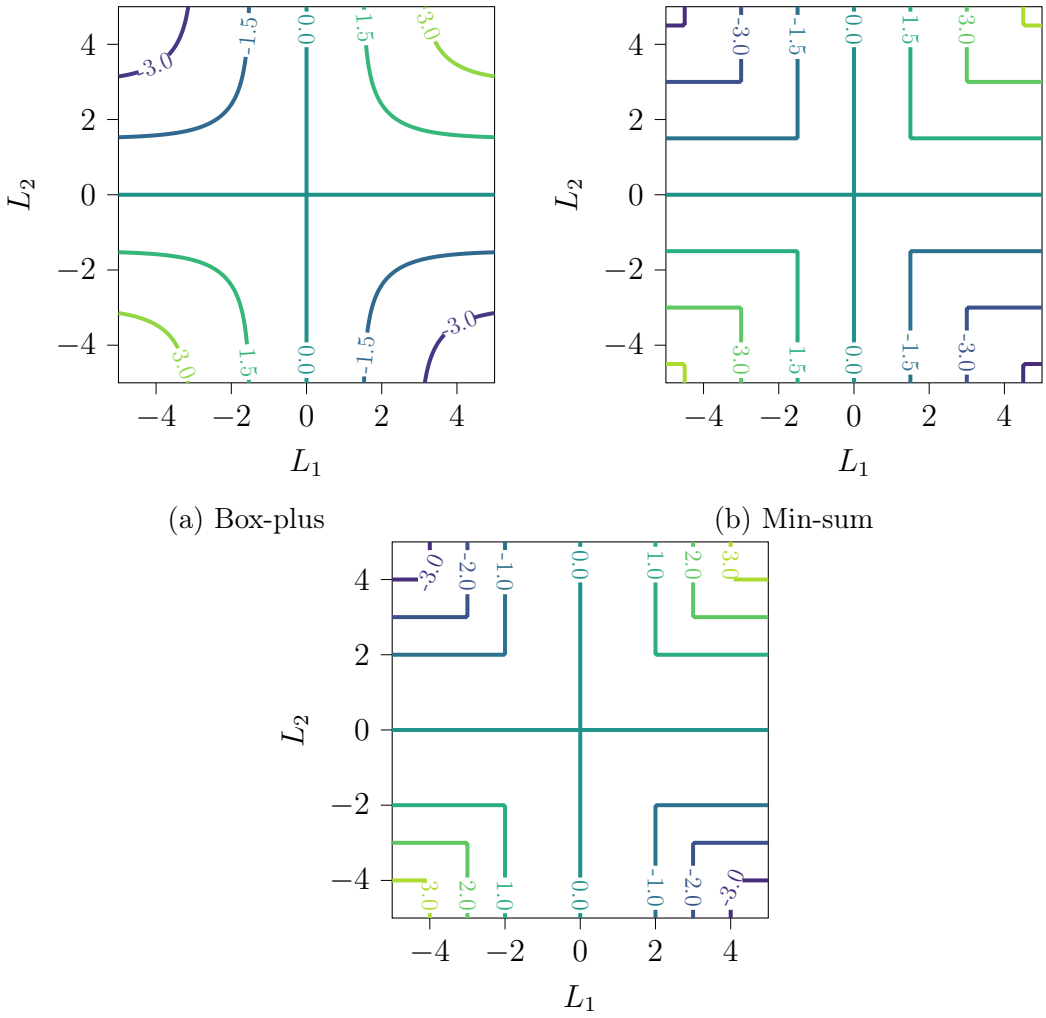
In turn, the outgoing message of a check node can also be written as

$$L_{\text{ext},i} = L_{\mathsf{c}_i \to \mathsf{v}_j} = \sum_{\mathsf{v} \in N(\mathsf{c}_i)\backslash\{\mathsf{v}_j\}}\!\!\boxplus\; L_{\mathsf{v} \to \mathsf{c}_i} , \tag{5.23}$$

i.e., the box-plus sum of the incoming log-likelihood ratios.

The input-output relation of the box-plus operation is sketched in Fig. 5.4a. The axes display the possible input values and the contour shows the respective output value. The evaluation of the exponential and logarithmic functions in the box-plus operations is extremely numerically complex and requires enhanced arithmetical units resulting in impractically high implementation complexity. A common approximation avoiding this high computational burden is the so-called min-sum approximation. Applying the Jacobian logarithm [Vit98; JDE+05; HEA+01]

$$\log\left(e^x + e^y\right) = \max^*(x, y) = \max(x, y) + \log\left(1 + e^{-|x-y|}\right) , \tag{5.24}$$



(a) Box-plus

(b) Min-sum

(c) Offset Min-sum

Figure 5.4: Input-output relation of (a) the box-plus operation, (b) min-sum operation and (c) offset min-sum operation.

Eq. (5.22) can be written as

$$
\begin{aligned}
L_1 \boxplus L_2 &= \log \frac{e^{L_1} e^{L_2} + 1}{e^{L_1} + e^{L_2}} \\
&= \max(L_1 + L_2, 0) + \log\left(1 + e^{-|L_1 + L_2|}\right) \\
&\quad - \left(\max(L_1, L_2) - \log\left(1 + e^{-|L_1 - L_2|}\right)\right) && (5.25) \\
&= \operatorname{sgn}(L_1) \cdot \operatorname{sgn}(L_2) \cdot \min(|L_1|, |L_2|) + \log\left(1 + e^{-|L_1 + L_2|}\right) \\
&\quad + \log\left(1 + e^{-|L_1 - L_2|}\right) && (5.26) \\
&\approx \operatorname{sgn}(L_1) \cdot \operatorname{sgn}(L_2) \cdot \min(|L_1|, |L_2|) \ . && (5.27)
\end{aligned}
$$

The input-output relation of the min-sum operation is sketched in Fig. 5.4b. Obviously, the non-linearities of the box-plus operation result in bent contour plots, whereas the min-sum operation cannot capture these curves and produces an edged

shape instead.

In [JDE+05], two versions of the min-sum operation were proposed. First, the normalized min-sum decoder weights the minimum LLR by a factor υ which yields

$$L_1 \boxplus L_2 \approx \operatorname{sgn}(L_1) \cdot \operatorname{sgn}(L_2) \cdot \frac{\min(|L_1|, |L_2|)}{\upsilon} . \tag{5.28}$$

This scaling of the LLRs by $1/\upsilon$ can vastly improve the performance compared to pure min-sum decoding [JDE+05]. The proper choice of $1/\upsilon$ can be determined using density evolution (cf. Section 5.1.3).

Second, instead of multiplying with a constant $1/\upsilon$ the offset-min-sum decoder subtracts a predetermined constant τ depending on the smallest magnitude of the respective LLRs, i.e.,

$$L_1 \boxplus L_2 \approx \operatorname{sgn}(L_1) \cdot \operatorname{sgn}(L_2) \cdot \max\left(\left(\min(|L_1|, |L_2|) - \tau\right), 0\right) . \tag{5.29}$$

As pointed out in [JDE+05], in contrast to normalized min-sum operation, the constant τ will set LLRs with small magnitudes to zero, i.e., the contribution of the conveyed check node messages in the next variable node update vanishes. In Fig. 5.4c, the input-output relation for an offset-min-sum check node is sketched. In Fig. 5.4c, the values of τ were set to

$$\tau = \begin{cases} 0, & \text{for } \min(|L_1|, |L_2|) < 1 \\ 1, & \text{for } 1 \leq \min(|L_1|, |L_2|) < 6 \\ 2, & \text{for } 6 \leq \min(|L_1|, |L_2|) \end{cases} . \tag{5.30}$$

5.1.3 Density Evolution, Extrinsic Transfer Charts and Asymptotic Decoding Threshold Analysis

Despite a possible performance degradation due to cycles in practical Tanner graphs, it is common to base the asymptotic decoding analysis of a particular code on code ensembles. Thus, it is assumed that for an infinite number of iterations $i \rightarrow \infty$, the graph is cycle-free [Joh10]. Here, the *concentration theorem* is exploited, which states that nearly all codes randomly chosen from an ensemble will have an iterative decoding performance close to the ensemble average performance [Joh10]. In addition, assuming also a symmetric input channel model, e.g., $p(y|\mathsf{X} = +1) = p(-y|\mathsf{X} = -1)$ for binary-input channels, the so-called *density evolution* can be performed to predict the reliability of exchanged messages while decoding using the *all-zeros* codeword.

Based on this evolution of the belief and, thus, the error probability $p_e^{(i)}$ in iteration i, the so-called decoding threshold can be determined. The decoding threshold determines the channel conditions under which error-free transmission is possible, i.e.,

$$\lim_{l\to\infty} \int_{-\infty}^{0} p_e^{(i)}(\tau) d\tau \to 0\,. \tag{5.31}$$

Hence, let us define the decoding threshold as [Joh10]

$$\alpha^* = \sup\left\{\alpha : \lim_{l\to\infty} \int_{-\infty}^{0} p_e^{(i)}(\tau) d\tau \to 0\right\}. \tag{5.32}$$

The actual derivation of the involved distributions is deferred to Section 5.2 where *discrete density evolution* is discussed, which is of integral importance for the mutual-information-based channel decoders derived in this thesis.

Instead of the density evolution, another very powerful analysis tool tracks the actual extrinsic information passed in the decoding process [tBKA04]. This tool is termed Extrinsic Information Transfer (EXIT) charts. EXIT charts belong to the class of parametric analysis tools as a single parameter, i.e., the extrinsic information is tracked instead of the entire distribution. Therefore, let us define the mutual information [tBKA04]

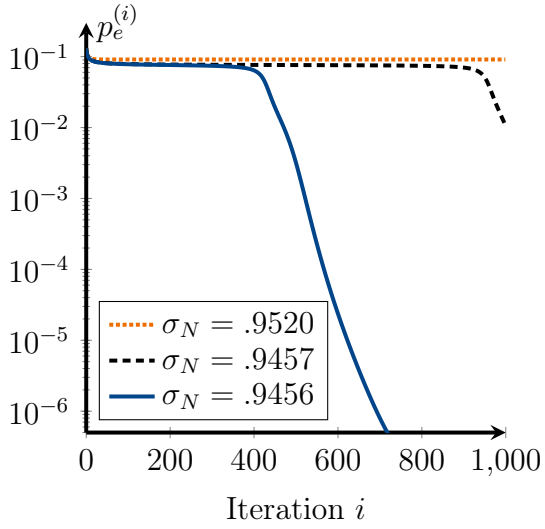
$$\mathrm{I}_{\mathsf{v}\to\mathsf{c}} = \mathrm{I}(\mathsf{V}; L_{\mathsf{v}\to\mathsf{c}}) \tag{5.33}$$

$$\mathrm{I}_{\mathsf{c}\to\mathsf{v}} = \mathrm{I}(\mathsf{V}; L_{\mathsf{c}\to\mathsf{v}}) \tag{5.34}$$

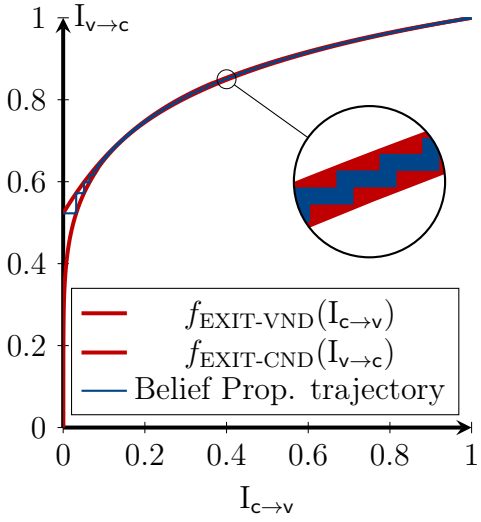
$$\mathrm{I}_{\mathrm{ch}} = \mathrm{I}(\mathsf{V}; L_{\mathrm{ch}}) \tag{5.35}$$

as the relevant extrinsic mutual information that is exchanged. As derived in [tBKA04], the EXIT analysis aims to determine a function $f_{\text{EXIT-VND}}(\mathrm{I}_{\mathsf{c}\to\mathsf{v}}, \mathrm{I}_{\mathrm{ch}})$ for the variable node update and $f_{\text{EXIT-CND}}(\mathrm{I}_{\mathsf{v}\to\mathsf{c}})$ for the check node update that predicts the outgoing extrinsic information based on the respective input mutual information.

In general, it is very challenging to compute these functions in closed form for arbitrary channels. Also for the BI-AWGN, which will be used throughout this thesis, deriving $f_{\text{EXIT-VND}}(\mathrm{I}_{\mathsf{c}\to\mathsf{v}}, \mathrm{I}_{\mathrm{ch}})$ and $f_{\text{EXIT-CND}}(\mathrm{I}_{\mathsf{v}\to\mathsf{c}})$ in closed form is impossible. However, ten Brink et al. [tBKA04] proposed an elegant approximation which assumes that the incoming and i.i.d. messages $L_{\mathsf{c}\to\mathsf{v}}$ respectively $L_{\mathsf{v}\to\mathsf{c}}$ are Gaussian distributed. Besides, also the outgoing messages are assumed Gaussian distributed with variance $\sigma^2_{L_{\mathsf{c}\to\mathsf{v}}}$ and mean $\mu_{L_{\mathsf{c}\to\mathsf{v}}} = 2\sigma^2_{L_{\mathsf{c}\to\mathsf{v}}}$, or $\sigma^2_{L_{\mathsf{v}\to\mathsf{c}}}$ and mean $\mu_{L_{\mathsf{v}\to\mathsf{c}}} = 2\sigma^2_{L_{\mathsf{v}\to\mathsf{c}}}$



(a) Bit error probability $p_e^{(i)}$ for the irregular LDPC code from Example 5.2 and transmission over a BI-AWGN with different noise variances.



(b) EXIT chart for for the irregular LDPC code from Example 5.2 and $\sigma_N = .9456$.

Figure 5.5: Analysis of the asymptotic decoding performance of the irregular LDPC code from Example 5.2.

respectively. Further, defining

$$J(\sigma_L) = \mathrm{I}(\mathsf{X};\mathsf{L}) = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_L^2}} e^{-\frac{(x-\sigma_L^2)^2}{2\sigma_L^2}} \log_2\left(1+e^{-z}\right) dz \tag{5.36}$$

as the mutual information of a BI-AWGN channel one obtains the following EXIT function for an BI-AWGN channel and for arbitrary *irregular* LDPC codes [tBKA04]

$$f_{\text{EXIT-VND}}(\mathrm{I}_{\mathsf{c}\to\mathsf{v}}) = \sum_{d_v=1}^{d_{\mathsf{v}}^{\max}} \lambda_{d_v} \cdot J\left(\sqrt{(d_v-1)J^{-1}(\mathrm{I}_{\mathsf{c}\to\mathsf{v}})^2 + \sigma_N^2}\right) \tag{5.37}$$

$$f_{\text{EXIT-CND}}(\mathrm{I}_{\mathsf{v}\to\mathsf{c}}) = \sum_{d_c=2}^{d_{\mathsf{c}}^{\max}} \rho_{d_c} \cdot \left\{1 - J\left(\sqrt{(d_c-1)J^{-1}(1-\mathrm{I}_{\mathsf{v}\to\mathsf{c}})}\right)\right\}. \tag{5.38}$$

Both considered analysis tools are applied in Example 5.2 to derive the decoding threshold of the irregular LDPC code from the DVB-S2 standard. Although both approaches were initially proposed for code construction and decoding threshold analysis, the next sections will utilize density evolution to design and evaluate coarsely quantized mutual-information based channel decoders for a given code ensemble.

Example 5.2: Continuous Density Evolution and EXIT chart analysis of an irregular LDPC code from the DVB-S2 standard

The DVB-S2 standard [Bro14] includes, among others, a standardized irregular LDPC code with rate $R_c = 0.5$ codeword length $N = 64000$ and the degree distributions

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_{\max}]^{\mathsf{T}} = [0, 0.2875, 0.25714, 0, 0, 0, 0, 0.4571]^{\mathsf{T}} \tag{5.39}$$

$$\boldsymbol{\rho} = [\rho_2, \rho_3, \ldots, \rho_{\max}]^{\mathsf{T}} = [0, 0, 0, 0, 0, 0.0000264, 0.999735]^{\mathsf{T}} \, . \tag{5.40}$$

Based on this degree distributions, density evolution can be performed to project the bit error probability after a maximum number of $i_{\max} = 1000$ iterations. This is shown in Fig. 5.5a. It can be observed that only a very tiny difference in the noise variance decides if the analyzed code either achieves error free transmission (in this example error free transmission is declared if $p_e \approx 10^{-6}$ is reached) or does not converge even for a very large number of iterations. Thus, the decoding threshold for this code can be found as $\alpha^* = \sigma_N* = 0.9456$ which corresponds to an $E_b/N_0 = 0.485565$ dB. Fig. 5.5b shows the EXIT chart for exactly this channel condition. The functions $f_{\text{EXIT-VND}}(\mathrm{I}_{\mathsf{c}\to\mathsf{v}})$ and $f_{\text{EXIT-CND}}(\mathrm{I}_{\mathsf{v}\to\mathsf{c}})$ are plotted in red, which bound the actual decoding trajectory of the belief propagation decoder depicted in blue. It can be observed that the variable node and check node function build a very narrow tunnel. However, this decoding tunnel is wide enough for the belief propagation decoder to pass and reach a final extrinsic information of approximately one. In cases where no convergence is observed, these two functions might intersect, which stops the iterative decoder from improving the reliabilities.

5.2 Decoding Binary Regular LDPC Codes Using the Information Bottleneck Method

The previous section discussed the asymptotic analysis of LDPC using density evolution. For particular channel models like BSC, BEC and AWGN channels closed-form solutions or elegant approximations for the density evolution exist.

However, Kurkoski et al. discussed a practically more relevant setting in [KYK08], i.e., density evolution for arbitrary discrete memory-less channels with coarsely quantized channel outputs. As example, Kurkoski et al. studied an AWGN channel with channel output quantization as discussed in Section 3.2.3. Interestingly, for this scenario deriving closed-form solutions of the tracked distributions is intractable as

discrete random variables X with arbitrary probability distributions $p(x)$ are considered where parametric approaches, e.g., Gaussian approximation are not applicable.

Instead, [KYK08] proposed introducing a mutual-information-based compression that allows restricting the otherwise exponentially growing sample space of the involved random variables but still allows tracing the evolution of the relevant extrinsic information for asymptotic code ensemble analysis. In the following section, the respective discrete probability distributions for a variable node and a check node in *discrete density evolution* are designed as published in [LSB16b] in the scope of this thesis. Further, the idea to reuse the relevant-information-preserving clusterings to replace the arithmetic node operations in the LDPC decoder is sketched, resulting in the so-called *information bottleneck decoder* [LB18].

5.2.1 Mutual-Information-Based Discrete Density Evolution

As discussed in Section 5.1.3, density evolution leverages the node update operations from belief propagation decoding for a particular channel output distribution, i.e., knowing $p(x|t_{\mathrm{ch}})$ to predict the asymptotic decoding performance of a specific LDPC code ensemble with known degree distributions. In discrete density evolution, it is assumed that T_{ch}, i.e., the observed random variable describing the quantized channel output is discrete, i.e., $|\mathcal{T}_{ch}|$ is finite.

Discrete Variable Node Update In [LSB16b], we derived the probability mass function $p(x, \mathbf{t}^{\mathrm{in}})$ from the sum-product algorithm similar to [KYK08]. Here, x denotes the respective relevant code bit and the vector $\mathbf{t}^{\mathrm{in}} = [t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}, \ldots, t_{d_v-1}^{\mathrm{in}}]^{\mathsf{T}}$ comprises the incoming *discrete* messages. Let us also assume that $t_{\mathrm{ch}}^{\mathrm{in}}$ represents the cluster index of the channel output quantizer derived using the information bottleneck method. The messages t^{in} are some cluster indices representing compact beliefs conveyed by the check nodes.

Without loss of generality, let us restrict to a $d_v = 2$ variable node and $|\mathcal{T}_{ch}| = |\mathcal{T}|$. By application of the factor-graph rules one obtains [LSB16b]

$$\begin{aligned} p(x, \mathbf{t}^{\mathrm{in}}) &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, \mathbf{t}^{\mathrm{in}}) \\ &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}) \\ &= \sum_{b_1} \sum_{b_2} p(x_1|b_1, b_2) p(b_1, b_2, t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}) \end{aligned}$$

$$= \sum_{b_1} \sum_{b_2} \underbrace{p(x_1|b_1,b_2)}_{\delta(b_1-x)\delta(b_2-x)} p(t_{\text{ch}}^{\text{in}}|b_1) p(t_1^{\text{in}}|b_2) \underbrace{p(b_1|b_2)}_{\delta(b_1-b_2)} p(b_2)$$
$$= \sum_{\substack{(b_1,b_2):\\ b_1=b_2=x_1}} p(t_{\text{ch}}^{\text{in}}|b_1) p(t_1^{\text{in}}, b_2)\,. \tag{5.41}$$

where the equality constraint at a variable node is exploited such that $x = b_1 = b_2$. Please note that in contrast to Eq. (5.21) and Eq. (5.20), where only conditional distributions were required to compute the respective LLRs, for mutual-information-based signal processing and discrete density evolution, it is essential to process the joint distributions.

Discrete Check Node Update Similar to the variable node, also the probability mass function $p(x, \mathbf{t}^{\text{in}})$ for the check node can be derived. Here, x denotes the respective result of the modulo-2 sum of the incoming messages and $\mathbf{t}^{\text{in}} = [t_1^{\text{in}}, \ldots, t_{d_c-1}^{\text{in}}]^{\mathsf{T}}$ as no additional channel message is processed at a check node.

Without loss of generality, let us consider a $d_c = 3$ check node. In turn, the joint probability mass function for a discrete check node update is computed as

$$\begin{aligned} p(x, \mathbf{t}^{\text{in}}) &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, \mathbf{t}^{\text{in}}) \\ &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, y_1, y_2) \\ &= \sum_{b_1} \sum_{b_2} p(x_1|b_1,b_2) p(b_1, b_2, y_1, y_2) \\ &= \sum_{b_1} \sum_{b_2} \underbrace{p(x_1|b_1,b_2)}_{\delta(b_1 \oplus b_2 \oplus x_1)} p(y_1|b_1) p(y_2|b_2) \underbrace{p(b_1|b_2)}_{p(b_1)} p(b_2) \\ &= \sum_{\substack{(b_1,b_2):\\ b_1 \oplus b_2 = x_1}} p(y_1, b_1) p(y_2, b_2)\,. \end{aligned} \tag{5.42}$$

5.2.2 Towards Information Bottleneck Decoding

The previous sections presented the computation of the joint distributions exchanged in an LDPC code. Interestingly, these distributions can be used to find deterministic mappings $f(\mathbf{t}^{\text{in}})$, preserving the maximum amount of relevant information and replacing the conventional computationally demanding node operations in belief propagation decoding [KYK08; LB15]. In addition, those mutual-information-based decoders are designed to take a very coarse quantization of exchanged messages into account already by design. However, deriving those relevant-information-preserving

mappings is not trivial. Early approaches started with hand-optimized lookup tables for only very specific channel models resulting in a so-called FAID decoder [PDD+13; DVP+13]. A more generic framework to derive these mappings is the information bottleneck method resulting in so-called information bottleneck decoders. The actual design processes of these decoders are reviewed in the next sections.

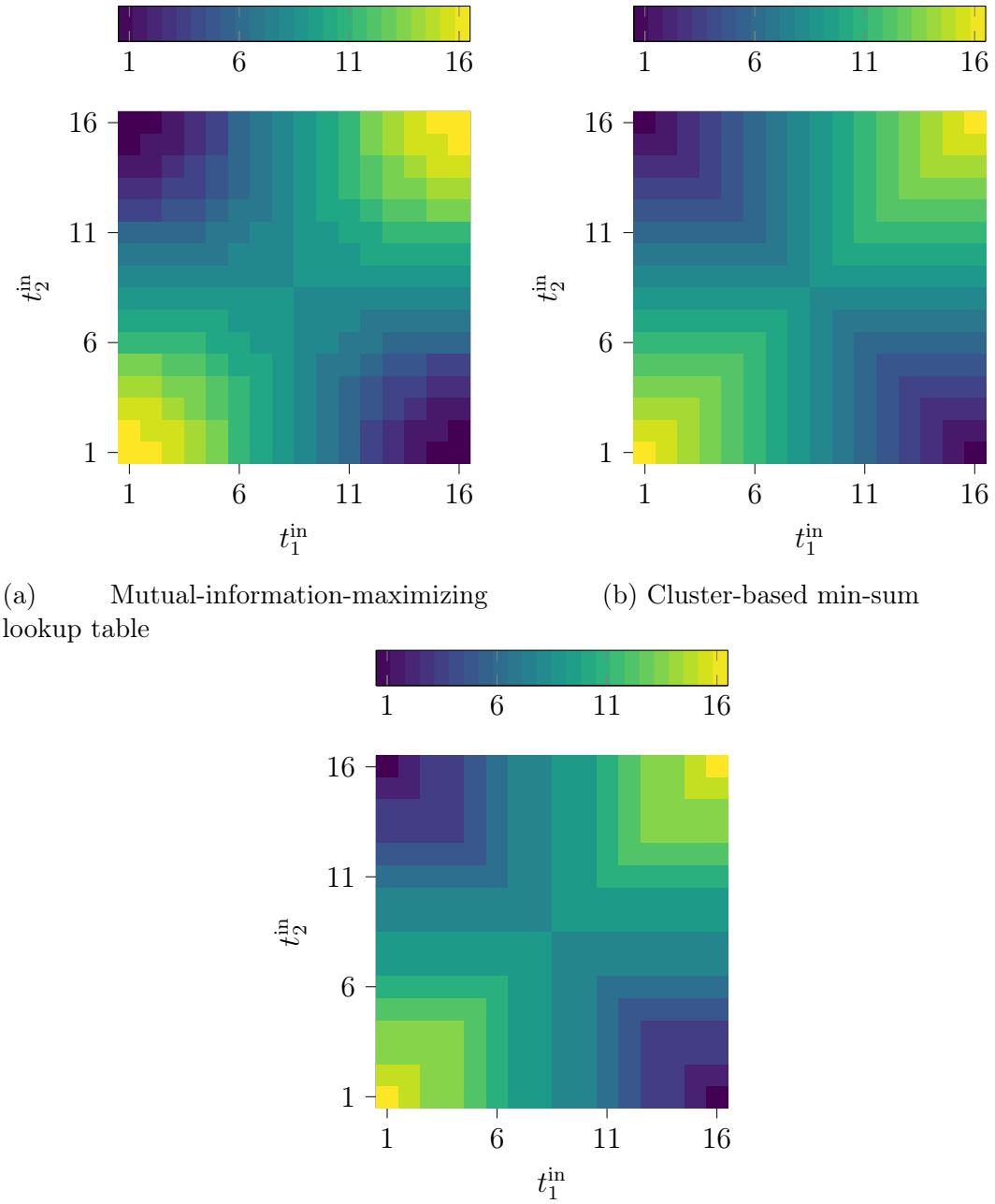
5.2.2.1 Mutual-Information-Based Variable Node Operation

In mutual-information-based signal processing, the task is to determine a deterministic function which maps an input vector $\mathbf{t}^{\text{in}} = [t_1^{\text{in}}, \ldots, t_M^{\text{in}}]^{\mathsf{T}}$ with M incoming discrete, integer-valued messages t_i^{in}, $i = 1, \ldots, M$ onto a discrete output t^{out}. In this case, the relevant variable X is the codeword bit represented by the variable node. The clustering is done such that the mutual information between the compressed observation t^{out} and the relevant codeword bit is maximized. Consequently, the actual decoding simplifies to an exchange of cluster indices and discrete mappings or lookup operations. Thus, there is no need to exchange real-valued LLRs and to perform arithmetic operations. Instead, the challenge is to obtain $p(x, \mathbf{t}^{\text{in}})$ such that the information bottleneck algorithm can find the optimal assignment $p(t^{\text{out}}|\mathbf{t}^{\text{in}})$ or $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$. The distribution $p(x, \mathbf{t}^{\text{in}})$ for a discrete variable node was derived in Eq. (5.41).

5.2.2.2 Mutual-Information-Based Check Node Operation

To determine the relevant-information-preserving mapping at a check node, the joint distribution $p(x, \mathbf{t}^{\text{in}})$ from Eq. (5.42) of the input vector $\mathbf{t}^{\text{in}}$ pooling discrete, integer valued messages and the relevant quantity is required. For generation of extrinsic information at a check node, the relevant variable X is the modulo 2 sum of $d_c - 1$ bits connected to the check node as discussed in Eq. (5.42). Applying the information bottleneck algorithm yields a discrete input-output mapping $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$ as depicted in Fig. 5.6a for $\mathbf{t}^{\text{in}} = [t_1^{\text{in}}, t_2^{\text{in}}]^{\mathsf{T}}$. Here, the clusters t_1^{in} and t_2^{in} are sorted according to their respective LLR. Although not intended to approximate the box plus operation, the mapping found by the information bottleneck represents the bended contours of the box plus operations (cf. Fig. 5.4a) much better than the min-sum operation. Furthermore, one observes that the symmetric properties of the box plus operations are preserved which allows to reduce the memory need when storing the function $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$ as lookup table.

Interestingly, as pointed out in [MBB+15], when the variable node operations are replaced by mutual-information-maximizing lookup tables, an application of the min-sum approximation is straightforward if the incoming messages are discrete



(a) Mutual-information-maximizing lookup table

(b) Cluster-based min-sum

(c) Cluster-based offset min-sum

Figure 5.6: Input-output relation of (a) the check node lookup table designed using the information bottleneck method, (b) the min-sum operation using cluster indices and (c) the offset min-sum operation using cluster indices.

cluster indices t_1^{in} and t_2^{in} and not LLRs L_1 and L_2. This is possible if the natural ordering of the cluster indices t_i^{in} represents the ordering of the LLRs L_i associated with t_i^{in}, where $i \in \{1, \ldots, M\}$ and M denotes the number of processed messages. This relation can be formalized as

$$t^{\text{out}} = \operatorname{sgn}\left(t_1^{\text{in}}\right) \operatorname{sgn}\left(t_2^{\text{in}}\right) \min\left\{|t_1^{\text{in}}||t_2^{\text{in}}|\right\} \tag{5.43}$$

where

$$\operatorname{sgn}\left(t_i^{\text{in}}\right) = \begin{cases} -1 & , t_i^{\text{in}} \leq |\mathcal{T}|/2 \\ +1 & , t_i^{\text{in}} > |\mathcal{T}|/2 \end{cases} \tag{5.44}$$

and

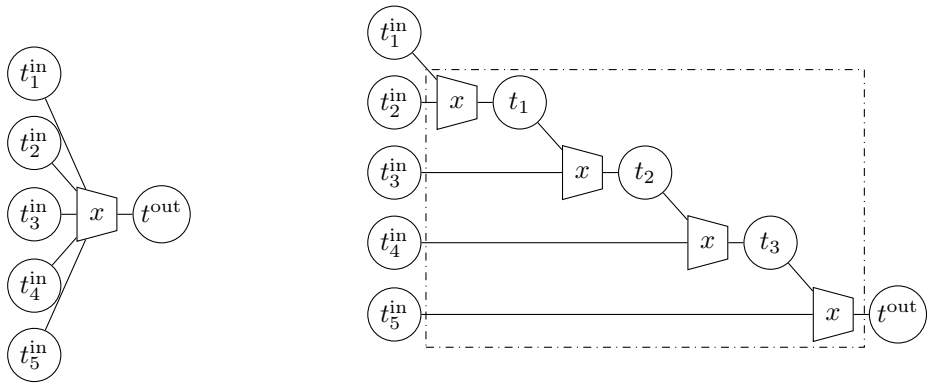
$$|t_i^{\text{in}}| = \begin{cases} |\mathcal{T}|/2 + 1 - t_i^{\text{in}} & , t_i^{\text{in}} < |\mathcal{T}|/2 \\ t_i^{\text{in}} - |\mathcal{T}|/2 & , t_i^{\text{in}} \geq |\mathcal{T}|/2 \end{cases}. \tag{5.45}$$

The respective input-output relations for min-sum and offset min-sum using clusters are shown in Fig. 5.6b and Fig. 5.6c, respectively.

5.2.2.3 The Curse of Dimensionality of Nodes with Large Node Degrees

In general, Eq. (5.41) and Eq. (5.42) can be generalized to any node degree. However, it shall be noted that the sample space and cardinality of the multivariate random variable for the incoming messages can be found as $\boldsymbol{\mathcal{T}}^{\text{in}} = \mathcal{T}_1^{\text{in}} \times \cdots \times \mathcal{T}_M^{\text{in}}$ and $|\boldsymbol{\mathcal{T}}^{\text{in}}| = |\mathcal{T}^{\text{in}}|^M$ where $M = d_v$ for a variable node and $M = d_c - 1$ for a check node. Clearly, $|\boldsymbol{\mathcal{T}}^{\text{in}}|$ grows exponentially with the number of inputs. Especially for irregular LDPC codes discussed in Section 5.3, this dependency can result in an extremely large sample space. In turn, also the mapping $f(\mathbf{t}^{\text{in}})$ becomes impractical if implemented as a lookup table. Hence, an idea already proposed in [KYK08] was to split the global function into a concatenation of simpler local two-input functions. This is shown in Fig. 5.7 for a check node with degree $d_c = 6$. For ease of notation, let us introduce the vector $\mathbf{t}_l$ where l indicates the *level* in the computation tree which contains the inputs to the intermediate node operations. Hence, $\mathbf{t}_1 = [t_1^{\text{in}}, t_2^{\text{in}}]^{\mathsf{T}}$ for the first level and $\mathbf{t}_l = [t_{l-1}, t_{l+1}^{\text{in}}]^{\mathsf{T}}$ for $l > 1$. The computation of the intermediate joint distributions $p(x, \mathbf{t}_l)$ is equivalent to Eq. (5.41) and Eq. (5.42).

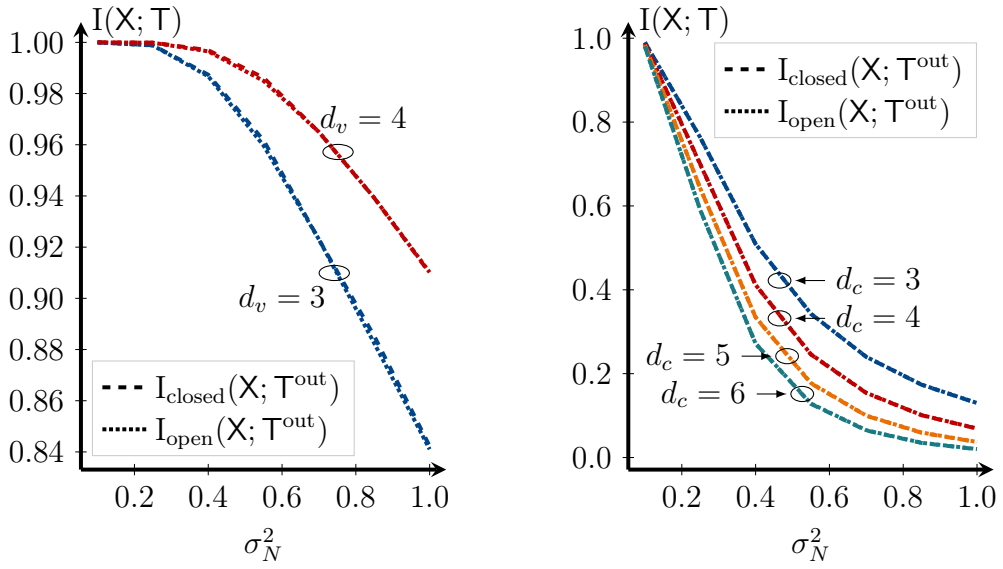
In contrast to classical algebra, where due to the associative law, computing intermediate results does not impact the result, opening the node can rather be interpreted as a concatenation of lossy compression steps due to the application of the information bottleneck, which looses a very small amount of relevant information in each step.



(a) Closed check node mapping.

(b) Check node mapping as concatenation of two-input mappings.

Figure 5.7: Comparison of different implementations of an exemplary $d_c = 6$ check node.

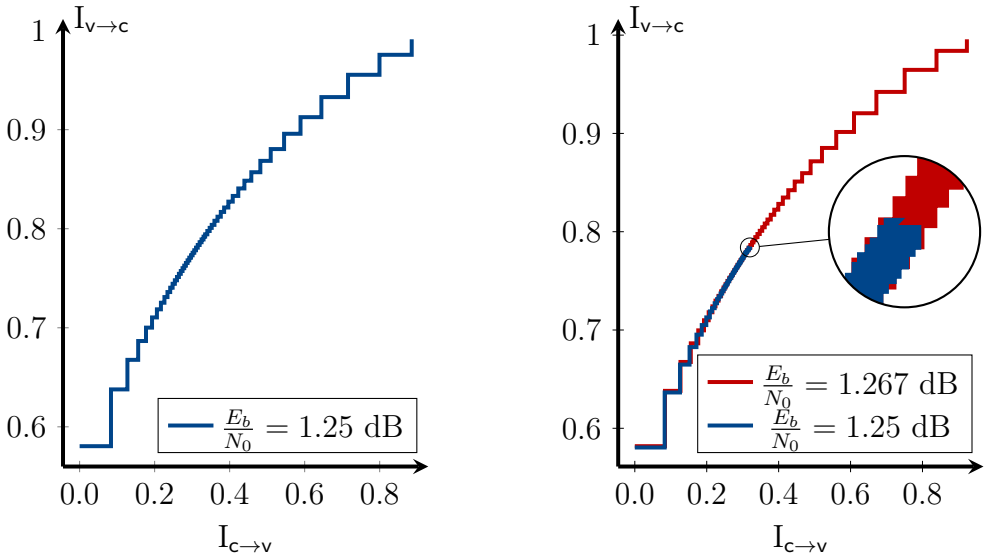


(a) Preserved relevant information $\mathrm{I}_{\text{closed}}(\mathsf{X};\mathsf{T}^{\text{out}})$ and $\mathrm{I}_{\text{open}}(\mathsf{X};\mathsf{T}^{\text{out}})$ for different variable node degrees.

(b) Preserved relevant information $\mathrm{I}_{\text{closed}}(\mathsf{X};\mathsf{T}^{\text{out}})$ and $\mathrm{I}_{\text{open}}(\mathsf{X};\mathsf{T}^{\text{out}})$ for different check node degrees.

Figure 5.8: Comparison of preserved relevant information for different node structures, node types and node degrees.

Fig. 5.8 shows the preserved mutual information for different check node degrees and variable node degrees for both the closed and the opened node. It can be observed that the loss seems negligible. However, as observed in Section 5.1.3, the decoding process is highly non-linear. Thus, the impact on the actual *iterative* decoding performance cannot be directly inferred from the results presented in Fig. 5.8. Hence, a



(a) EXIT trajectory for a (3,6) regular LDPC code information bottleneck decoder with a closed node structure.

(b) EXIT chart for a (3,6) regular LDPC code information bottleneck decoder with an opened node structure.

Figure 5.9: Comparison of EXIT charts for a regular (3,6) regular LDPC code ensemble with different information bottleneck decoder node structures.

discrete EXIT chart analysis is provided to investigate the impact of the additional compression loss in a two-input setting. This is shown in Fig. 5.9 for a (3,6) regular code ensemble for $i_{\max} = 50$ and corresponding design $E_b/N_0 = 1.25$ dB. Fig. 5.9a shows the EXIT trajectory for the closed node and Fig. 5.9b shows the EXIT chart for the opened node, respectively. Interestingly, it can be observed that the information bottleneck decoder with an opened node structure gets stuck before a mutual information of one is reached for the design $E_b/N_0 = 1.25$ dB. Instead, the actual decoding threshold of this information bottleneck decoder is $E_b/N_0 = 1.267$ dB. Nevertheless, this loss is negligible in most practical applications. Thus, due to the negligible loss but the largely reduced complexity of the two-input approach, all decoders analyzed in this thesis are constructed with this node structure if no other structure is explicitly mentioned.

5.2.3 Simulation Results and Evaluation

In this section, the performance of the devised information bottleneck decoder is analyzed for two different regular LDPC codes and compared to state-of-the-art systems. In this section, the LDPC codes are taken from the public code database [Mac20]. More precisely, the following codes are considered

Table 5.1: Simulation parameters of decoder and reference system compared in Fig. 5.10a for Code 1.

decoder	node operation (check / var.)	precision exchanged messages	precision check node	precision variable node	channel quantizer
belief-propagation (belief-prop.)	box-plus/ addition	64 bit	64 bit	64 bit	4 bit
min-sum	Eq. (5.27)/ addition	4 bit	4 bit	6 bit	4 bit
information bottleneck decoder (IB)	lookup table/ lookup table	4 bit	4 bit	4 bit	4 bit

Table 5.2: Simulation parameters of decoder and reference system compared in Fig. 5.10b for Code 2.

decoder	node operation (check / var.)	precision exchanged messages	precision check node	precision variable node	channel quantizer
belief-propagation (belief-prop.)	box-plus/ addition	64 bit	64 bit	64 bit	64 bit
min-sum	Eq. (5.27)/ addition	4 bit	4 bit	6 bit	4 bit
information bottleneck decoder (IB)	lookup table/ lookup table	3-5 bit	3-5 bit	3-5 bit	3-5 bit

1. Code 1: (3,6) - LDPC code with identifier 8000.4000.3.483 from [Mac20] with code length $N = 8000$ and code rate $R_c = 0.5$.

2. Code 2:(4,24) - LDPC code with identifier 4161.731.4.356 from [Mac20] with code length $N = 4161$ and code rate $R_c = 5/6 \approx 0.833$.

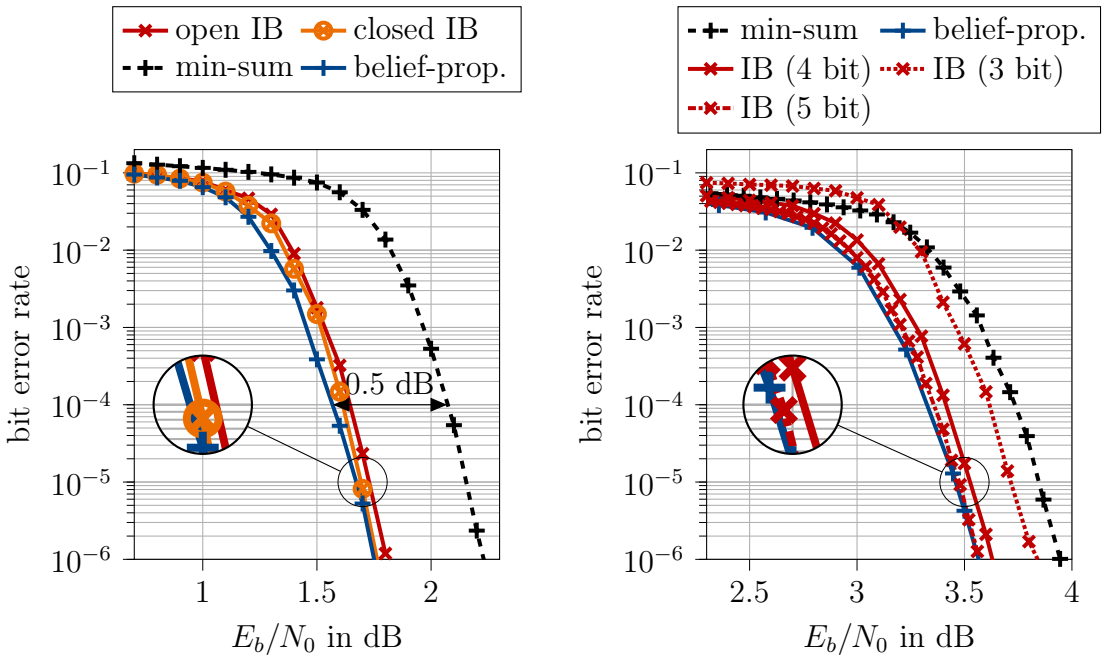
The BER simulation results are shown in Fig. 5.10a and Fig. 5.10b, respectively. In this section, only the belief propagation decoder and min-sum decoder without any modification (cf. Eq. (5.27)) are used for comparison. For the first code presented in Fig. 5.10a, the channel output fed into all decoders is coarsely quantized with 4 bit for all investigated decoders. The simulation parameters for the decoders decoding code 1 depicted in Fig. 5.10a are summarized in Table 5.1. Both possible implementations of the internal node structure for the information bottleneck decoder, i.e., an opened node (open IB) and a closed node structure (closed IB), are analyzed in Fig. 5.10a

in terms of the respective bit error rate performance. Similar to the theoretical EXIT chart investigation presented in Section 5.2.2.3, it can be observed that the performance loss caused by opening the nodes in a practical information bottleneck decoder results in a performance degradation of less than 0.1 dB. For both decoders, the design E_b/N_0 values determined using the EXIT chart analysis were chosen, i.e., a design $E_b/N_0 = 1.25$ dB for the closed node structure and a design $E_b/N_0 = 1.267$ dB for the opened node. Again, this underlines that the practical benefits of an opened node structure predominates the very small performance degradation.

Compared to the reference systems, the opened-node information bottleneck decoder operates only around 0.1 dB away from the belief-propagation decoder (dark blue curve in Fig. 5.10a), which requires high-precision internal messages and very complicated arithmetical node operations, whereas the information bottleneck decoder had an internal resolution of only 4 bits and all node operations where replaced by simple lookup tables. Also, the second benchmark system, i.e., the min-sum decoder (black curve in Fig. 5.10a), works with double-precision internal messages and (approximated) arithmetical node operations but works significantly worse than the information bottleneck decoder. In detail, the gap shown in Fig. 5.10a between belief-propagation decoder and the min-sum decoder is around 0.5 dB.

Furthermore, the investigations for the second code with a significantly higher rate and decoder parameters summarized in Table 5.2 yield similar observations. The plot in Fig. 5.10b shows the performance of the reference schemes (please note that the belief-propagation decoder is *not* paired with a channel output quantizer) and the information bottleneck decodes which have an internal resolution and a channel output quantizer resolution ranging from 3 bit to 5 bit. Interestingly, it can be observed that a 5 bit information bottleneck decoder performs extremely close (less than 0.01 dB) to a not-quantized double-precision belief-propagation decoder. Furthermore, it is shown that a 3 bit information bottleneck decoder outperforms a 4 bit min-sum decoder by a fair margin. Please note that all information bottleneck decoders for code 2 require an open internal node structure as the check node degree $d_c = 24$ prohibits the design of closed nodes. The determined design E_b/N_0 values are 3.065 dB (5 bit), 3.115 dB (4 bit) and 3.275 dB (3 bit).

All information bottleneck decoders are only generated once for the particular design channel conditions and used mismatched in the simulation, i.e., also the channel quantizer is not adjusted to the actual channel conditions. Thus, only one set of lookup tables and one channel quantizer is required, and the decoder is also robust against erroneous SNR estimation.



(a) Bit error rate simulations for length 8000 $(3, 6)$ regular LDPC code with $i_{\max} = 50$ decoding iterations over AWGN channel with BPSK from [Mac20] with identifier 8000.4000.3.483.

(b) Bit error rate simulations for length 4161 $(4, 24)$ regular LDPC code with $i_{\max} = 50$ decoding iterations over AWGN channel with BPSK from [Mac20] with identifier 4161.731.4.356.

Figure 5.10: Comparison of bit-error simulation results for two different regular LDPC codes and different decoders.

In total, all these observations motivate to extend the concept of information bottleneck decoders to more enhanced code structures, including irregular LDPC codes and punctured, rate-compatible ensembles. These extended decoders were developed in the scope of this thesis and are discussed in the next sections.

5.3 Decoding Binary Irregular LDPC Codes Using the Information Bottleneck Method

In contrast to regular LDPC codes, irregular LDPC codes are characterized by nodes with varying degrees. As introduced in Section 5.1, the edge-degree distributions $\lambda(\zeta)$ and $\rho(\zeta)$ are used. For irregular LDPC codes, the outgoing messages depend on the node degree of the conveying node. Thus, in discrete density evolution for irregular LDPC codes also the respective joint distributions $p(x, \mathbf{t}^{\text{in}}|d)$ depend on the node degree d. Consequently, it is not sufficient to design message mappings only for variable nodes or check nodes but for variable nodes or check nodes *considering* the individual node degrees.

However, in density evolution a code ensemble is considered instead of a particular irregular LDPC code with a certain parity check matrix. Hence, the connectivity between variable and check nodes is only known on average and defined by the degree distribution.

To construct the required input joint distributions $p(x, \mathbf{t}^{\text{in}}|d)$, discrete density evolution from [KYK08; LB18] needs to be extended to consider the degree distribution of the code ensemble. This section will derive this generalization of the information bottleneck decoder to arbitrary irregular LDPC codes, which we first published in [SLB18a; SLB18b].

5.3.1 Mutual-Information-Based Discrete Density Evolution for Binary Irregular LDPC Codes

Discrete Variable Node Update In Section 5.2.1, we derived the probability mass function $p(x, \mathbf{t}^{\text{in}})$ for discrete density evolution in regular LDPC codes to design an information bottleneck decoder. These distributions still build the starting point for the generalization of information bottleneck decoders to irregular LDPC codes. First, degree-dependent joint distributions $p(x, \mathbf{t}^{\text{in}}|d)$ are computed for each variable node degree present in the code ensemble according to Eq. (5.41). Please note that assuming a trivial, straightforward generalization of information bottleneck decoders to irregular LDPC codes, one would stop at this point. In this case, lookups in node-degree-specific tables $p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d)$ would replace the node operations and only integer valued cluster indices are exchanged. However, this approach is practically infeasible as each path of an exchanged message through the Tanner graph has to be tracked independently and no *generic* decoder design for a code *ensemble* is possible.

Hence, to incorporate the degree distribution of a code ensemble, one has to average over all possible degrees resulting in the marginal distribution $p(x, \mathbf{t}^{\text{in}})$, i.e.,

$$p(x, \mathbf{t}^{\text{in}}) = \sum_{d=1}^{d_{\text{v}}^{\max}} \lambda_d p(x, \mathbf{t}^{\text{in}}|d)\,. \tag{5.46}$$

In *discrete* density evolution, $p(x, t^{\text{out}})$ has to be tracked instead of $p(x, \mathbf{t}^{\text{in}})$. We define the marginal distribution $p(x, t^{\text{out}})$ as

$$\begin{aligned} p(x, t^{\text{out}}) &= \sum_{d=1}^{d_{\text{v}}^{\max}} \lambda_d p(x, t^{\text{out}}|d) \\ &= \sum_{d=1}^{d_{\text{v}}^{\max}} \lambda_d \sum_{\mathbf{t}^{\text{in}} \in \mathcal{T}^{\text{vec}}} p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d) p(x, \mathbf{t}^{\text{in}}|d)\,, \end{aligned} \tag{5.47}$$

where $\mathcal{T}^{\text{vec}}$ denotes the set of all possible combinations of $\mathbf{t}^{\text{in}}$ for a node with degree d. As it will be shown later, this straightforward marginalization is unfavorable for the mutual-information-maximizing design principle.

Discrete Check Node Update Similar to the variable node, also the degree-depend probability mass function $p(x, \mathbf{t}^{\text{in}}|d)$ for the check node can be derived using Eq. (5.42). Based on the degree distribution, the marginal distribution $p(x, \mathbf{t}^{\text{in}})$ for a check node is found as

$$p(x, \mathbf{t}^{\text{in}}) = \sum_{d=2}^{d_{\text{c}}^{\max}} \rho_d p(x, \mathbf{t}^{\text{in}}|d)\,. \tag{5.48}$$

Again, as we target *discrete* density evolution where the distribution $p(x, \mathbf{t}^{\text{in}}|d)$ is fed into an information bottleneck algorithm to find a very compact representation of the exchanged relevant information, $p(x, t^{\text{out}})$ has to be computed:

$$\begin{aligned} p(x, t^{\text{out}}) &= \sum_{d=2}^{d_{\text{c}}^{\max}} \rho_d p(x, t^{\text{out}}|d) \\ &= \sum_{d=2}^{d_{\text{c}}^{\max}} \rho_d \sum_{\mathbf{t}^{\text{in}} \in \mathcal{T}^{\text{vec}}} p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d) p(x, \mathbf{t}^{\text{in}}|d)\,. \end{aligned} \tag{5.49}$$

In [SKW06], it was first described that discretized min-sum decoders require a particular degree distribution $\lambda(\zeta)$, respectively $\rho(\zeta)$, in order to not suffer from a large gap between the decoding threshold of the belief-propagation decoder and that decoding threshold of the discretized min-sum decoder. Note that the sample space $\mathcal{T} = \{1, \ldots, |\mathcal{T}|\}$ is *independent* of the node degree d, but in contrast, the particular meaning $p(x|t^{\text{out}}, d)$ *depends* on the node degree. Thus, from the cluster indices alone, the check node cannot resolve if a message originates from a variable with a high or a low degree. As the variety of node degrees increases, the dynamic range of the respective reliabilities also increases. Thus, especially irregular LDPC codes with high irregularity suffer from large performance degradation when decoded with coarse quantization and conventional LLR-based decoding, e.g., using min-sum decoding. However, in the next subsection, a pre-processing step to improve the performance of the information bottleneck decoder based on message alignment is presented.

5.3.2 The Message Alignment Problem in Discrete Density Evolution for Irregular LDPC Codes

This section reviews message alignment as introduced in [SLB18b]. In addition to the approach from [SLB18b], an alternative realization is proposed, which is closely related to [MMB20].

5.3.2.1 Explicit Message Alignment

As the event space $\mathcal{T}$ is *independent* of the node degree d, but in contrast $p(x|t^{\text{out}}, d)$ *depends* on the node degree, the marginalization as in Eq. (5.47) averages misaligned beliefs [SLB18a]. In turn, this marginalization causes an additional loss of relevant information, i.e., $\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{D}) > \mathrm{I}(\mathsf{X};\mathsf{T})$. Fig. 5.11a depicts the exemplary distributions $p(x|t^{\text{out}}, d_v = 2)$, $p(x|t^{\text{out}}, d_v = 3)$ and $p(x|t^{\text{out}}, d_v = 8)$ in terms of their LLR for an irregular LDPC code taken from the DVB-S2 standard (cf. Example 5.2). The LLRs of the distribution $p(x|t^{\text{out}})$ resulting from a straightforward marginalization is depicted in Fig. 5.11b. It can be observed that especially the more reliable messages suffer from the marginalization of non-aligned beliefs. Interestingly, this problem is closely related to the example of bitwise channel output quantizer design presented in Section 4.2.3, which was solved using message alignment. Hence, it is beneficial to tackle this problem from an information-theoretical perspective instead of performing Eq. (5.47) directly.

For the node-degree-dependent information bottleneck design, the information bottleneck setting involves the random variables $\mathsf{T}, \mathsf{D}, \mathsf{X}$, and Z as depicted in Fig. 5.12. As visualized in Fig. 5.13, given $p(x, t^{\text{out}}|d)$ and λ_d in the considered example, the joint distribution of these random variables can be found as $p(x, t^{\text{out}}, d)$, with mutual information $\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{D})$. As the outgoing message shall be restricted to $\mathcal{Z}$, the task is to find a mapping $p(z|t^{\text{out}}, d)$ such that $\mathrm{I}(\mathsf{X};\mathsf{Z})$ is maximized. Here, it is assumed that $\mathcal{Z} = \mathcal{T}$. As the mapping $p(z|t^{\text{out}}, d)$ can be decomposed and embedded in the node design, such that the lookup table becomes $p(z|\mathbf{t}^{\text{in}}, d)$ instead of $p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d)$, as introduced in Chapter 4, this technique is called *message alignment* as it ensures that messages with the same index capture the same belief. This is illustrated in Fig. 5.11c. Since the alignment relies explicitly on the degree-dependent mappings, $p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d)$ is referred to as *explicit* message alignment. The resulting *aligned* belief $p(x, z)$ is computed as

$$p(x,z) = \sum_{d=1}^{d_{\text{v}}^{\max}} \lambda_d \sum_{\mathbf{t}^{\text{in}} \in \mathcal{T}^{\text{vec}}} p(z|t^{\text{out}}, d) p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d) p(x, \mathbf{t}^{\text{in}}|d)\,, \tag{5.50}$$

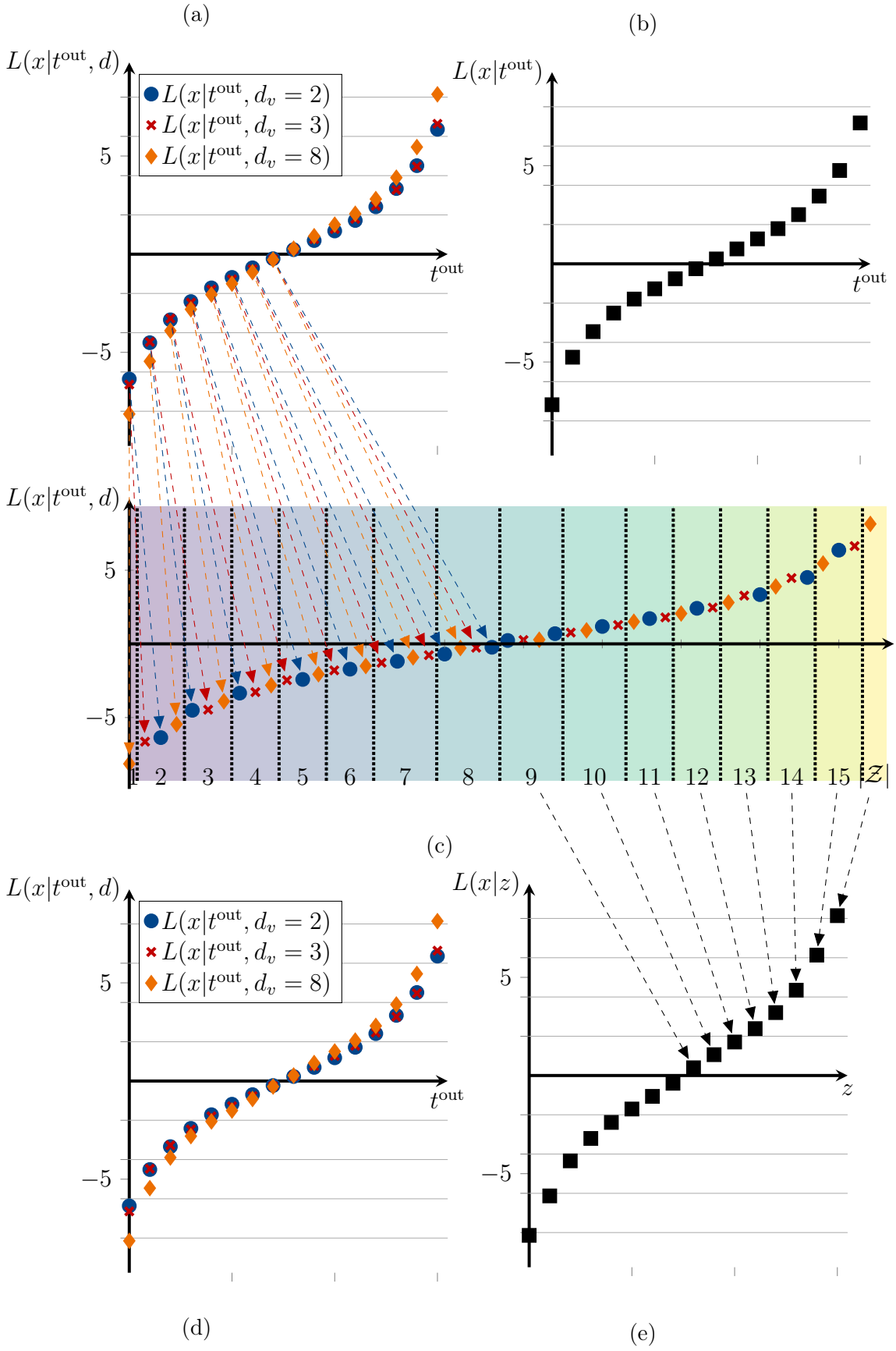


Figure 5.11: Illustration of the message alignment problem for the design of irregular variable nodes.

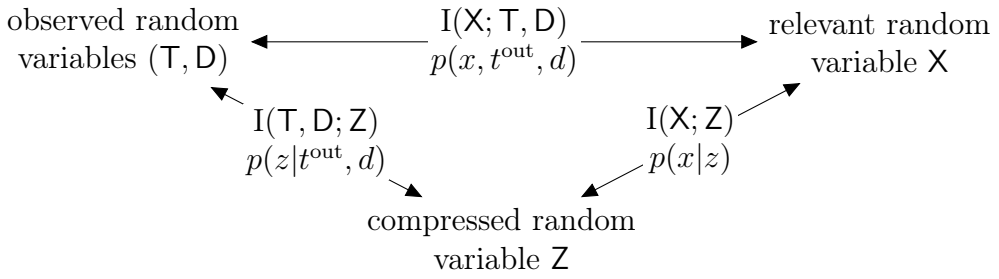


Figure 5.12: Message alignment formulated as an information bottleneck, where $\mathsf{I}(\mathsf{X};\mathsf{Z})$ is the relevant information, $\mathsf{I}(\mathsf{X};\mathsf{T},\mathsf{D})$ is the original mutual information and $\mathsf{I}(\mathsf{T},\mathsf{D};\mathsf{Z})$ is the compressed information.

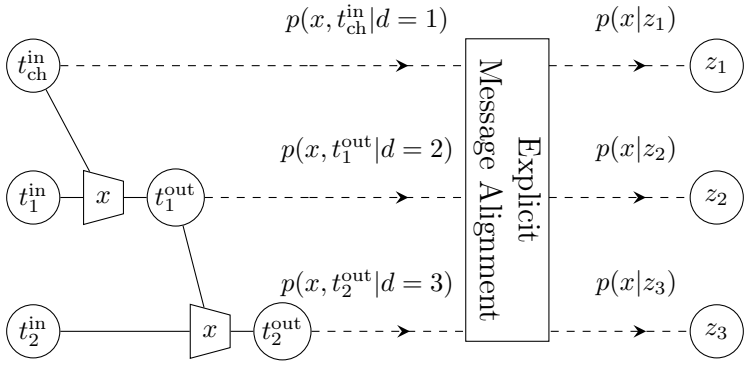


Figure 5.13: Designing explicit message alignment for a variable node: The joint distribution $p(x, t^{\text{out}}, d)$ used for alignment is composed of the individual output distributions $p(x, t^{\text{out}}|d)$ weighted by the edge-degree distribution.

for a variable node and as

$$p(x,z) = \sum_{d=2}^{d_{\text{c}}^{\max}} \rho_d \sum_{\mathbf{t}^{\text{in}} \in \mathcal{T}^{\text{vec}}} p(z|t^{\text{out}},d)p(t^{\text{out}}|\mathbf{t}^{\text{in}},d)p(x,\mathbf{t}^{\text{in}}|d)\,, \tag{5.51}$$

for a check node, respectively. For the considered example depicted in Fig. 5.11, the LLRs of resulting *aligned* distribution $p(x|z)$ are shown in Fig. 5.11e. In contrast to the LLRs of $p(x|t^{\text{out}})$, i.e., without alignment (cf. Fig. 5.11b), it can be observed that the actual reliabilities encoded by the node-depended cluster indices are no longer significantly changed due to the marginalization. This also impacts the amount of preserved relevant information of the exchanged messages.

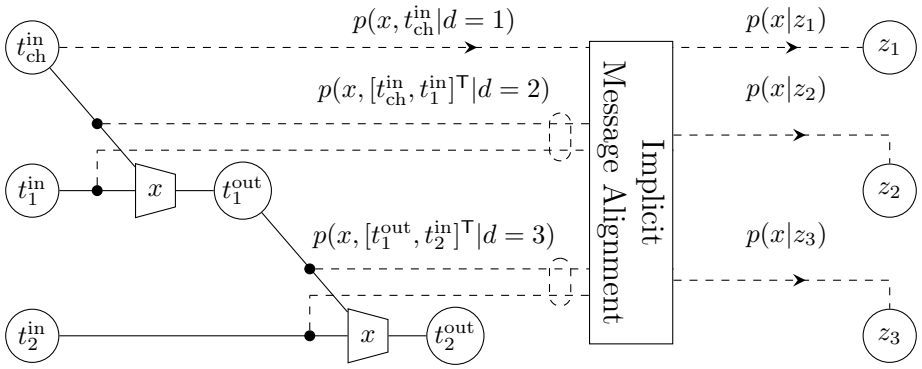


Figure 5.14: Designing implicit message alignment: The joint distribution $p(x, \mathbf{t}^{\text{in}}, d)$ used for alignment is now composed of the individual input distributions $p(x, \mathbf{t}^{\text{in}}|d)$ of the information bottleneck algorithm weighted by the edge-degree distribution.

Example 5.3: Message Alignment for an Irregular Variable Node

Let us assume an information bottleneck variable node with degree $d_v = 4$, depicted in Fig. 5.13 as a concatenation of two-input-lookup tables. In Fig. 5.13, each lookup table is depicted as an information bottleneck node with the input vector $\mathbf{t}^{\text{in}} = [t_{\text{ch}}^{\text{in}}, t_1^{\text{in}}]^{\mathsf{T}}$ or $\mathbf{t}^{\text{in}} = [t_{i-1}^{\text{out}}, t_i^{\text{in}}]^{\mathsf{T}}$, where $i = 2, \ldots, d_v - 1$ and output t_i^{out}. As illustrated in Fig. 5.21, the lookup tables for all node degrees $d_v < 4$ are implicitly constructed as they serve as intermediate results for the $d_v = 4$ variable node. Thus, the overall number of lookup tables depends only on the largest node degree and not on the variety of node degrees. In turn, the intermediate mappings $p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d)$ are fed into the message alignment unit to explicitly construct a node-degree-independent belief $p(x, z)$.

5.3.2.2 Implicit Message Alignment

Instead of treating message alignment as a post-processing step, it can also be included as a design objective immediately in the lookup table design. Here, the mappings $p(z|\mathbf{t}^{\text{in}})$ are designed using $p(x, \mathbf{t}^{\text{in}}|d)$ directly instead of using $p(x, t^{\text{out}}|d)$. This is depicted in Fig. 5.14. Analog to the message alignment setup from Fig. 5.12, the random variables $\mathsf{T}, \mathsf{X}, \mathsf{D}$ and Z serve as a starting point. Thus, instead of two subsequent optimizations, i.e., first to find $p(t^{\text{out}}|\mathbf{t}^{\text{in}}, d)$ and then $p(z|t^{\text{out}}, d)$, as in the explicit message alignment setting, $p(z|\mathbf{t}^{\text{in}}, d)$ is found in one shot. In analogy to Eq. (5.50) and Eq. (5.51) the *implicitly* aligned distributions are found as [SWB+20]

$$p(x, z) = \sum_{d=1}^{d_{\text{v}}^{\max}} \lambda_d \sum_{\mathbf{t}^{\text{in}} \in \mathcal{T}^{\text{vec}}} p(z|\mathbf{t}^{\text{in}}, d) p(x, \mathbf{t}^{\text{in}}|d)\,, \tag{5.52}$$

for a variable node and as

$$p(x,z) = \sum_{d=2}^{d_{\mathrm{c}}^{\max}} \rho_d \sum_{\mathbf{t}^{\mathrm{in}} \in \mathcal{T}^{\mathrm{vec}}} p(z|\mathbf{t}^{\mathrm{in}}, d) p(x, \mathbf{t}^{\mathrm{in}}|d)\,, \tag{5.53}$$

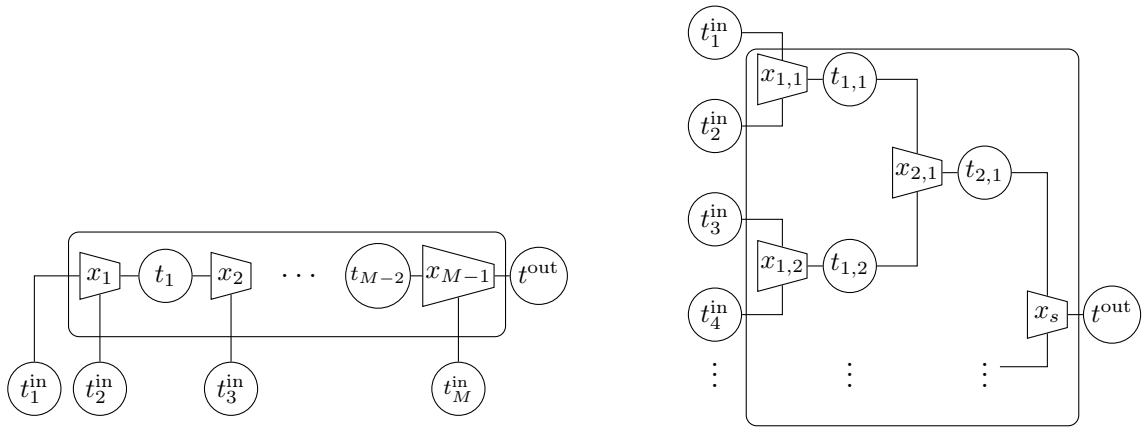
for a check node respectively.

5.3.3 Optimized Node Structures Operations for Large Node Degrees

Especially very powerful irregular LDPC codes often require very high node degrees to achieve medium to high code rates [RSU01]. For the presented two-input decomposition discussed in Section 5.2.2.3, the number of required lookup tables and lookup operations depends linearly on the node degree. However, especially for high node degrees, a more efficient tree-like decomposition exists, which we published first in [SLB18a]. In contrast to the approach presented in Section 5.2.2.3, the tree-decomposition discussed in this section enables more efficient decoding in terms of space complexity and latency.

First, let us review the sequential decomposition from Section 5.2.2.3 in terms of memory requirements and the total number of *distinct* lookup tables. As illustrated in Fig. 5.15a, $M-1$ different tables are required in each iteration and for each node type due to the sequential concatenation of lookup tables. Since only two incoming discrete messages are used at a time, the size of the lookup table is $|\mathcal{T}^{\mathrm{in}}|^2$. Thus, instead of one large table with $|\mathcal{T}^{\mathrm{in}}|^M$ entries, in total, $(M-1)\cdot|\mathcal{T}^{\mathrm{in}}|^2$ entries need to be constructed. In turn, during decoding using the sequential design, $M-1$ lookup operations have to be performed to determine one outgoing message. Especially in applications where high code rates are required, the check node degrees are very large. Hence, in these settings, the sequential design is very inefficient in terms of memory demand and latency due to the large number of required lookup tables.

We presented an alternative decomposition in [SLB18a], which resorts to a tree-like splitting of the node operation, as shown in Fig. 5.15b. We note that when using a tree-like pattern, the depth of the information bottleneck graph is reduced from $\mathcal{O}(M-1)$ to $\mathcal{O}(2\cdot\lfloor\log_2(M)\rfloor)$. In case M is not a power of two, at most $2\cdot\lfloor\log_2(M)\rfloor$ lookup stages are needed, see Appendix A.2 for a detailed derivation. Table 5.3 contains an overview of the required memory depending on the chosen node structure.



(a) (b)

Figure 5.15: Illustration of an opened check node, where all M incoming messages $t_1^{\text{in}}, ...t_M^{\text{in}}$ are clustered sequentially (Fig. 5.15a) or in a tree-like manner (Fig. 5.15b).

Table 5.3: Overview of maximum required lookup tables and their sizes depending on the node structure.

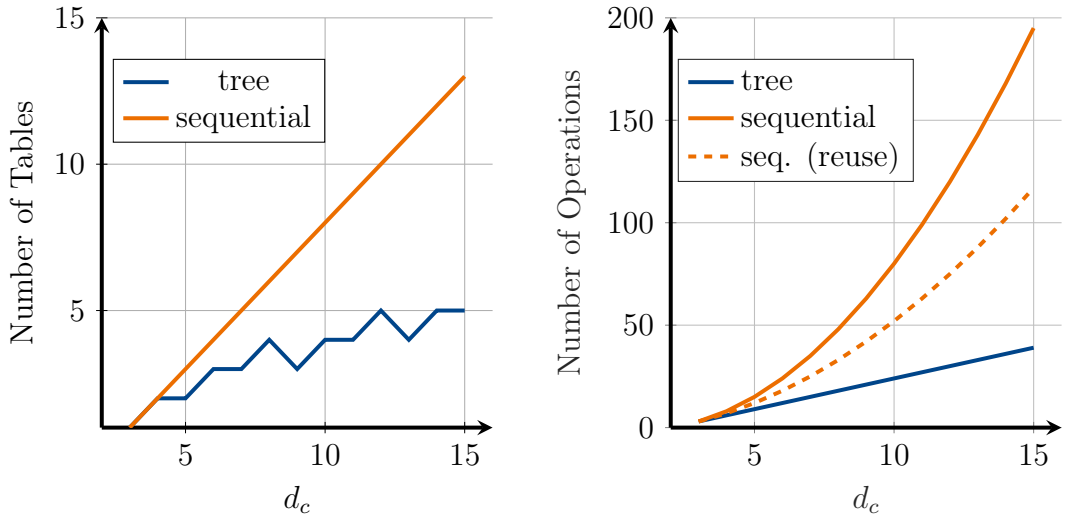
Node Structure	Entries per Table	Look-up Tables	Total Memory Demand
closed	$\lvert\mathcal{T}^{\text{in}}\rvert^M$	1	$\lvert\mathcal{T}^{\text{in}}\rvert^M$
open (sequential)	$\lvert\mathcal{T}^{\text{in}}\rvert^2$	$M-1$	$(M-1)\cdot\lvert\mathcal{T}^{\text{in}}\rvert^2$
open (tree)	$\lvert\mathcal{T}^{\text{in}}\rvert^2$	$\leq 2\cdot\lfloor\log_2(M)\rfloor$	$\leq 2\cdot\lfloor\log_2(M)\rfloor\cdot\lvert\mathcal{T}^{\text{in}}\rvert^2$

Reusing Intermediate Results In information-bottleneck decoders, the actual decoding simplifies to lookup operations in the offline-generated tables. To compute an outgoing message, all incoming messages, except the one received over the edge connected to the node we generate the outgoing message for, are considered. Hence, all outgoing messages are computed using a slightly different input vector $\mathbf{t}^{\text{in}}$. In a sequential node decomposition of a check node, this would result in

$$d_c \cdot (d_c - 2) = d_c^2 - 2d_c \tag{5.54}$$

lookup operations.

However, parts of the input vector do not change for several outgoing messages. Thus, the total number of lookup operations per node can be reduced by a reuse of intermediate results. Assuming a sequential node structure, we note that, for example, if only t_M^{in} is changed, all t_i for $i = 1, \ldots, M-2$ previous results could be reused. Thus, the number of total operations for all outgoing edges for a node with



(a) Number of lookup tables for different node architectures and (check) node degrees.

(b) Number of lookup operations required to compute *all* outgoing messages for different node architectures and (check) node degrees.

Figure 5.16: Evaluation of number of operations and lookup tables for different internal node structures and reuse patterns.

M incoming messages can be found as

$$d_c \cdot (d_c - 2) - \sum_{i=3}^{d_c-1} (i-2) = d_c \cdot (d_c - 2) - \sum_{i=1}^{d_c-3} i = \frac{(d_c-2)(d_c+3)}{2} . \tag{5.55}$$

By exploiting the proposed tree structure, the number of lookup operations per node can be reduced even further compared to the sequential approach. Fig. 5.16 depicts the increase in lookup tables and lookup operations for an increasing check node degree. It can be observed that the tree-like node opening is superior to the sequential node opening in both the number of operations and lookup tables, especially for larger node degrees. The actual reuse potential depends largely on the internal structure of the tree. However, in the worst-case $\mathcal{O}(M\lfloor \log_2 M \rfloor)$ operations per node are required [SLB18a]. Such a reuse in a tree-like structure is illustrated in Fig. 5.17 for an exemplary $d_c = 6$ check node.

5.3.4 Simulation Results and Evaluation

In this section, the devised information bottleneck decoder design for arbitrary irregular LDPC code ensembles is evaluated. Therefore, the following two codes from different standards are considered:

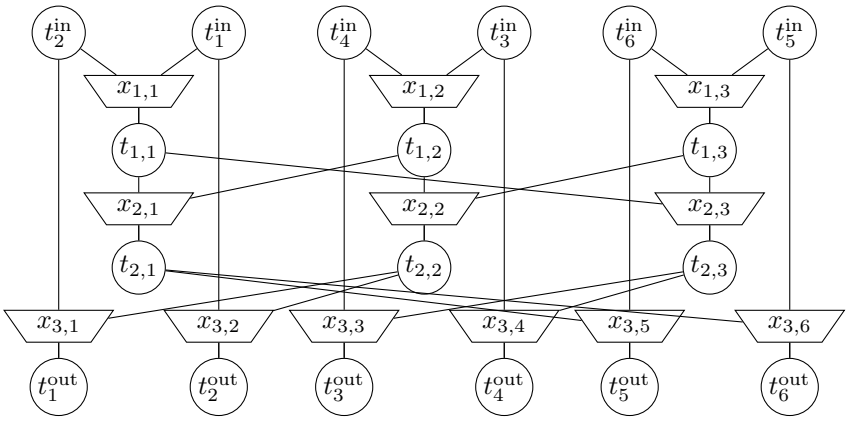


Figure 5.17: Reuse of intermediate lookup results in a tree-like structure to minimize the total number of lookup operations.

1. Code 3: $R_c = 0.5$, $N = 64000$ LDPC code from the DVB-S2 standard with edge degree distribution:

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_8]^{\mathsf{T}} = [0, 0.2875, 0.25714, 0, 0, 0, 0, 0.4571]^{\mathsf{T}} \tag{5.56}$$

$$\boldsymbol{\rho} = [\rho_2, \rho_3, \ldots, \rho_7]^{\mathsf{T}} = [0, 0, 0, 0, 0, 0.0000264, 0.999735]^{\mathsf{T}} \, . \tag{5.57}$$

2. Code 4: $R_c = 0.5$, $N = 1056$ LDPC code from the WiMAX standard with edge degree distribution:

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_6]^{\mathsf{T}} = [0, 0.28947368, 0.31578947, 0, 0, 0.39473684]^{\mathsf{T}} \tag{5.58}$$

$$\boldsymbol{\rho} = [\rho_2, \rho_3, \ldots, \rho_7]^{\mathsf{T}} = [0, 0, 0, 0, 0, 0.63157895, 0.36842105]^{\mathsf{T}} \, . \tag{5.59}$$

In this section, only information bottleneck decoders with $|\mathcal{T}| = 16$, i.e., 4 bit quantization, are considered for ease of brevity. A variation of the bit width and code rate is presented in the next section. Similar to the analysis of information bottleneck decoders for regular LDPC codes (cf. Section 5.3), a belief-propagation decoder and a min-sum decoder with parameters presented in Table 5.4 are considered as benchmark systems.

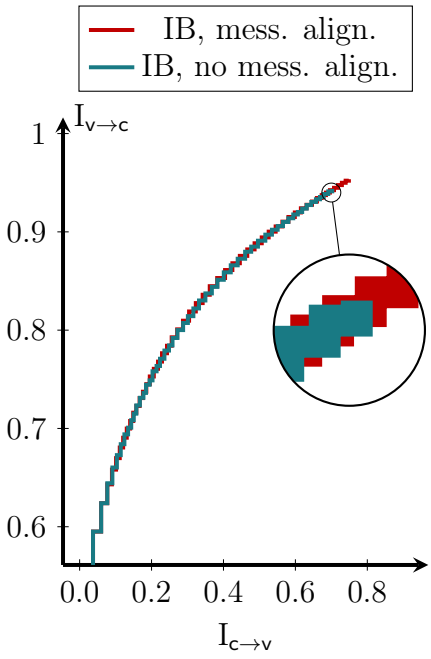
Impact of Message Alignment in the Decoder Design on the Decoding Performance First, the impact of the presented message alignment scheme on the decoding performance shall be analyzed. This investigation utilizes EXIT charts, i.e., an asymptotic performance analysis tool and actual bit error rate simulations for code 3, i.e., from the DVB-S2 standard. Fig. 5.18a depicts the EXIT trajectories for an information bottleneck decoder with message alignment (cf. red curve

Table 5.4: Simulation parameters of decoder and reference system compared in Fig. 5.18b for Code 3 and in Fig. 5.19 for Code 4.

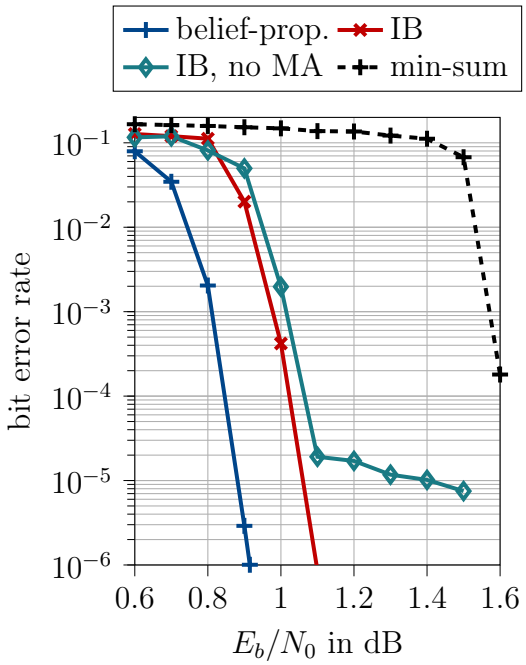
decoder	node operation (check / var.)	precision exchanged messages	precision check node	precision variable node	channel quantizer
belief-propagation (belief-prop.)	box-plus/ addition	64 bit	64 bit	64 bit	4 bit
min-sum	Eq. (5.27)/ addition	4 bit	4 bit	6 bit	4 bit
information bottleneck decoder (IB)	lookup table/ lookup table	4 bit	4 bit	4 bit	4 bit

in Fig. 5.18a) and without message alignment (cf. dark-green curve in Fig. 5.18a) for a design $E_b/N_0 = 0.8$ dB. It can be observed that the decoder without message alignment can not generate sufficiently large extrinsic information. Instead, the marginalization of misaligned beliefs in the computation of the joint distributions (cf. Eq. (5.49) and Eq. (5.47)) results in an additional loss of relevant information, which harms the overall decoder performance. Interestingly, these observations from the EXIT chart analysis translate directly into the BER results shown in Fig. 5.18b. It can be observed that the information bottleneck decoder without message alignment shows an undesirably high error floor at a bit error rate of around 10^{-5}. In contrast, the presented information bottleneck decoder with message alignment achieves higher extrinsic information in the EXIT chart and shows no error floor in the considered bit error rate range, as shown in Fig. 5.18a and Fig. 5.18b. Furthermore, comparison to the benchmark decoders reveals that the information bottleneck decoder can outperform a min-sum decoder by a considerable margin and operates within a 0.15-0.2 dB gap to double-precision belief-propagation decoding.

Impact of the Design E_b/N_0 on the Decoding Performance As the decoder mappings are generated once offline and used for the entire SNR range, the design E_b/N_0 is a crucial design parameter. Fig. 5.19 depicts the bit error rate performance for code 4 of several information bottleneck decoders with message alignment for different design E_b/N_0 values. It can be observed that there exists a trade-off between the decoding cliff, i.e., the start of the waterfall region, and the error floor of the decoder. As the design E_b/N_0 is increased, the decoder *overestimates* the channel conditions and thus overestimates the reliabilities of received values resulting in a deteriorated decoding cliff. In contrast, if the design E_b/N_0 is decreased, the decoder *underestimates* the channel conditions in the high SNR regime resulting in an early



(a) EXIT chart for an information bottleneck decoder with and without message alignment for design $E_b/N_0 = 0.8$ dB for code 3 from the DVB-S2 standard.



(b) BER simulations for the presented information bottleneck decoder and other benchmark decoder.

Figure 5.18: EXIT charts and BER results for an irregular $R_c = 0.5$ LDPC code from the DVB-S2 standard (Code 3).

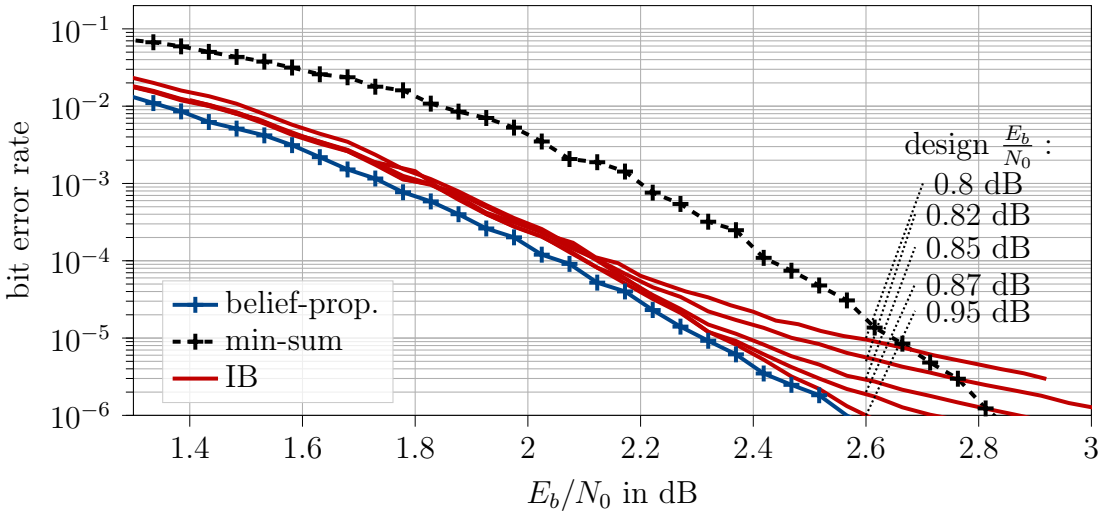


Figure 5.19: Investigation of the impact of design-E_b/N_0 on the decoder performance for the irregular LDPC code from the WiMAX standard (Code 4).

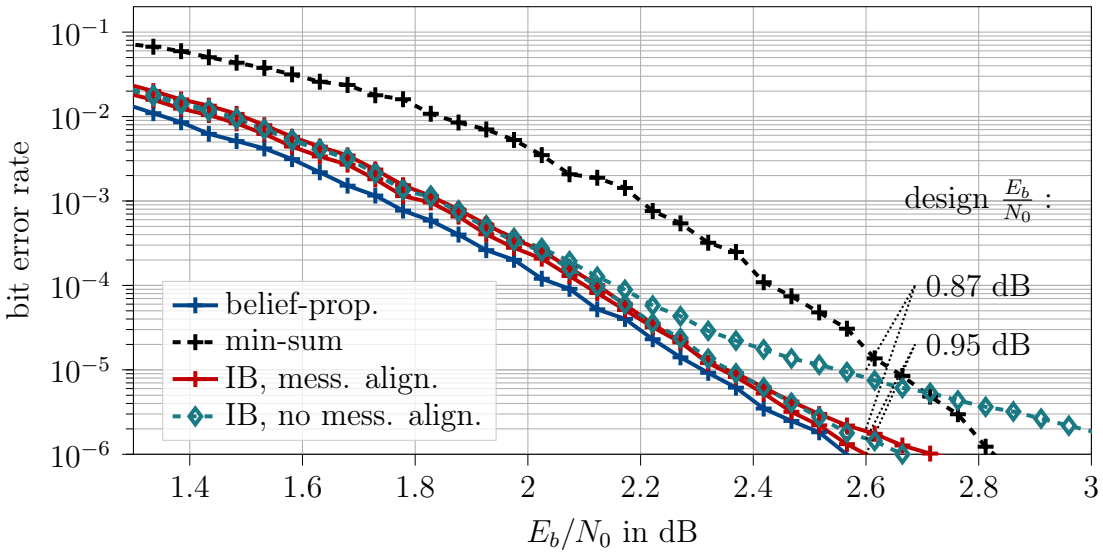


Figure 5.20: Investigation of the impact of message alignment on the decoder performance for the irregular LDPC code from the WiMAX standard (Code 4) and different design-E_b/N_0.

error floor as, during the decoder design, very reliable messages were not expected. In addition, Fig. 5.20 again underlines the importance of message alignment in the decoder design. It can be observed that without message alignment, the decoding threshold is deteriorated by around 0.08 dB just due to an unfavorable decoder design technique. In turn, with message alignment, the performance of the decoder can be improved notably. However, it should be noted that the derivation of the proper design E_b/N_0 is a delicate task and requires a bisection search to yield the best decoding performance.

5.4 Information Bottleneck Decoders for PBRL LDPC Codes Capturing Rate Compatible Design and Puncturing

As outlined in Section 5.1.1.1, modern communication systems resort to structured LDPC code ensembles, which support puncturing and inherent rate-compatibility to adjust to varying transmission conditions quickly. In the most recent 5G standard, LDPC codes from the protograph-based raptor-like (PBRL) family are considered.

As these PBRL codes rely heavily on puncturing, also the information bottleneck decoder design has to be generalized to handle this puncturing in the high-redundancy code. Additionally, all of the incremental redundancy variable nodes are punctured

for the highest code rate, but degree-one variable nodes are added to the protograph as the rate is lowered. Thus, a degree-one variable node might be punctured depending on the code rate. The information bottleneck decoder must be able to adapt to the induced changes in the degree distributions and the associated changes in the probability distributions of message reliabilities that occur as the rate is lowered. Thus, the respective information bottleneck decoder must support puncturing to decode PBRL LDPC codes. Puncturing denotes the process of *not* transmitting code bits. As a result, the number of information bits per codeword bits, which is the code rate, can be easily and gradually changed. At the receiver side in a conventional LLR-based decoder, the punctured bits are represented by LLRs equal to zero, which are fed into the decoder. Although puncturing itself is a fairly easy problem for conventional decoders, it is not straightforward for information bottleneck decoders. In the following sections, incorporating punctured nodes in the decoder design is explained. Therefore, the implications of puncturing on density evolution are facilitated as a message alignment problem. Furthermore, the next sections focus on the different notions of puncturing faced in PBRL codes and provides detailed examples and simulation results. Finally, a rate-compatible decoder design is devised, which allows reusing lookup tables across a broad range of code rates. We published parts of this section in [SWB+20; SBW+20].

5.4.1 Puncturing PBRL LDPC Codes as Message Alignment Problem

As shown in [LB18; RK16; KYK08], to achieve the best performance with mutual-information-based lookup tables, symmetric input distributions are optimum if the channel is symmetric. As a result, the LLR zero is originally not covered in information bottleneck decoders. An additional cluster might be introduced to tackle this problem, which explicitly corresponds to the LLR zero. This approach results in an uneven number of clusters.

This section shows that by using message alignment and the structure of PBRL codes, mutual-information maximizing lookup tables that support puncturing can also be designed with an even number of clusters that show close-to-optimum decoding performance. First, the effects of puncturing at the variable nodes and check nodes on the computation of the input joint distributions for the information bottleneck method are investigated.

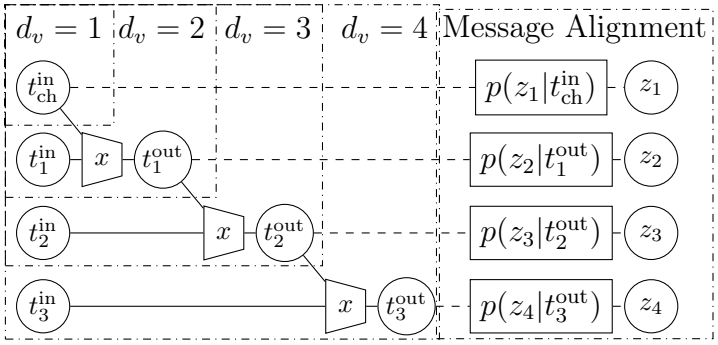


Figure 5.21: Processing in a concatenated lookup table for $d_{\mathrm{v}} = 4$ with message alignment.

5.4.1.1 Puncturing from a Variable Node Perspective

A variable node can encounter puncturing in two ways. First, the channel message which is connected to the variable node can be punctured. Second, a message from a check node can be essentially punctured, if the respective check node was connected to a punctured degree-one variable node. We will refer to the first type as channel-induced puncturing and the second type as check-node-induced puncturing. Irrespective of the origin, a punctured message cannot contribute any *relevant* information. As a result, there is no need to process this message.

Example 5.4: Puncturing from a Variable Node Perspective

Let us assume again an information bottleneck variable node with degree $d_v = 4$, depicted in Fig. 5.21 as a concatenation of two-input-lookup tables. Please remember that in the unpunctured case, the number of incoming messages was $M = 4$, since three messages received over edges connected to check nodes plus the channel message are processed. If the channel message is punctured, the effective degree is reduced by one. Thus, the respective input distribution is $p(x, t_1^{\mathrm{in}}, t_2^{\mathrm{in}}, t_3^{\mathrm{in}})$. Also, if the message from a check node is punctured, the effective degree is reduced by one. Thus, the respective input distribution is, for example, $p(x, t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}, t_2^{\mathrm{in}})$ if t_3^{in} is punctured. Please note that for this joint distribution, it does not matter which of the messages conveyed by check nodes is punctured, as due to density evolution and message alignment, all individual distributions $p(x, t_i^{\mathrm{in}})$ are the same in each iteration. In turn, only the number of punctured messages conveyed from the check nodes matters.

Example 5.4 illustrates that two notions of puncturing exist at a variable node. First, we introduce the random variable P with event space $\varrho \in \{\text{true, false}\}$ indicating if a node is punctured or not. In this context, the puncturing rate equals the fraction of variable nodes with degree $d > 1$ that are punctured.

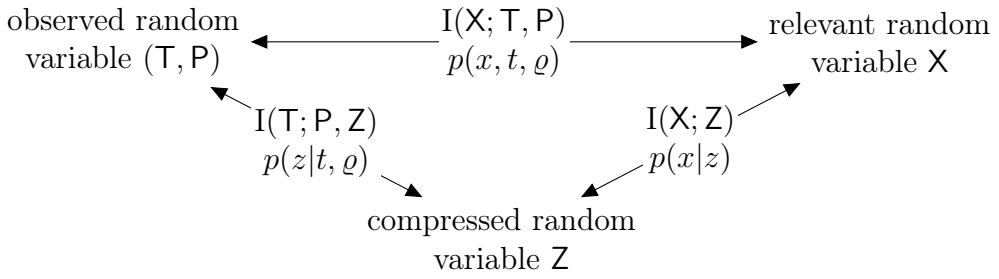


Figure 5.22: Considering puncturing as message alignment problem, where $\mathrm{I}(\mathsf{X};\mathsf{Z})$ is the relevant information, $\mathrm{I}(\mathsf{X};\mathsf{T},\mathsf{P})$ is the original mutual information and $\mathrm{I}(\mathsf{T};\mathsf{P},\mathsf{Z})$ is the compressed information.

Channel-Induced Puncturing

As depicted in Fig. 5.21, the channel message is processed in the first stage, for which the input vector is $\mathbf{t}^{\mathrm{in}} = [t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}]^{\mathsf{T}}$. The resulting joint distribution equals

$$p(x, [t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}]^{\mathsf{T}}) = \frac{1}{p(x)} p(x, t_{\mathrm{ch}}^{\mathrm{in}}) p(x, t_1^{\mathrm{in}})\,. \tag{5.60}$$

In the next step, $p(x, [t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}]^{\mathsf{T}})$ and thus also $p(x|t_1^{\mathrm{out}})$ are used which depend on the statistics of the quantized channel output (cf. Section 3.4.1.3). When incorporating puncturing, $p(x, t_{\mathrm{ch}}^{\mathrm{in}})$ differs if the channel message is punctured or not. As a result, we rewrite (5.60) as

$$p(x, [t_{\mathrm{ch}}^{\mathrm{in}}, t_1^{\mathrm{in}}]^{\mathsf{T}}|\varrho) = \frac{1}{p(x)} p(x, t_{\mathrm{ch}}^{\mathrm{in}}|\varrho) p(x, t_1^{\mathrm{in}})\,. \tag{5.61}$$

Due to the concatenation of lookup tables, as shown in Fig. 5.21, all subsequent tables depend on P. Consequently, in a straightforward implementation, the number of required lookup tables will increase drastically to account for all possible combinations of punctured and non-punctured variable nodes and their respective degrees. Hence, we propose to make use of the message alignment technique to prohibit such an increase in the number of lookup tables. The corresponding information bottleneck setting is shown in Fig. 5.22. By applying message alignment, one creates the mapping $p(z|t, \varrho)$ and the meaning $p(x|z)$ such that all subsequently constructed tables do not depend any longer on the node being punctured or not. This approach ensures that the number of lookup tables is not increased as compared to an unpunctured information bottleneck decoder.

Check-Node-Induced Puncturing

Besides the puncturing of the channel message, also a message received from a check node can be punctured, e.g., if the respective check node is connected to a punctured

degree-one variable node. This is explained in more detail in Section 5.4.1.2. As a result, the variable node degree is reduced, i.e., fewer lookup tables need to be constructed (cf. Fig. 5.21). However, the computation of the joint distributions remains

$$p(x, [t_{\text{ch}}^{\text{in}}, t_1^{\text{in}}]^{\mathsf{T}}) = \frac{1}{p(x)} p(x, t_{\text{ch}}^{\text{in}}) p(x, t_1^{\text{in}}) \tag{5.62}$$

for the first lookup table and

$$p(x, [t_{i-1}^{\text{out}}, t_i^{\text{in}}]^{\mathsf{T}}) = \frac{1}{p(x)} p(x, t_{i-1}^{\text{out}}) p(x, t_i^{\text{in}}) \tag{5.63}$$

for all subsequent lookup tables $i = 2, \ldots, d_{v,\max} - 1$. However, the overall *effective* edge-degree distribution $\boldsymbol{\lambda}_{\text{eff}}$ will be changed. In turn, this effective edge-degree distribution has to be considered in message alignment when computing the overall outgoing aligned belief.

5.4.1.2 Puncturing from a Check Node Perspective

Check nodes are only implicitly affected by puncturing if they are connected to a punctured degree-one variable node, or in the first iteration if the incoming message is a punctured channel message. If one incoming message is punctured, i.e., the relevant information is zero, all outgoing messages will also be punctured, i.e., they convey no information. Thus, the respective check node is effectively *deactivated* in the Tanner graph. This changes the effective edge-degree distribution for the check nodes $\boldsymbol{\rho}_{\text{eff}}$, which has to be considered in the message alignment for the lookup table construction.

5.4.1.3 Effective Degree Distributions

As discussed in the previous section, puncturing effects the effective degrees of both the variable nodes and the check nodes. Thus, in contrast to classical density evolution where the code ensemble is considered, when designing information bottleneck decoders, the Tanner graph needs to be known to determine the effective edges in PBRL codes and the corresponding effective degree distributions $\boldsymbol{\rho}_{\text{eff}} \neq \boldsymbol{\rho}$ and $\boldsymbol{\lambda}_{\text{eff}} \neq \boldsymbol{\lambda}$.

5.4.2 Constructing Rate-Compatible Information Bottleneck Decoders

Rate-compatible codes which allow to efficiently adapt the code rate according to the channel conditions are a crucial and inevitable part of modern communication

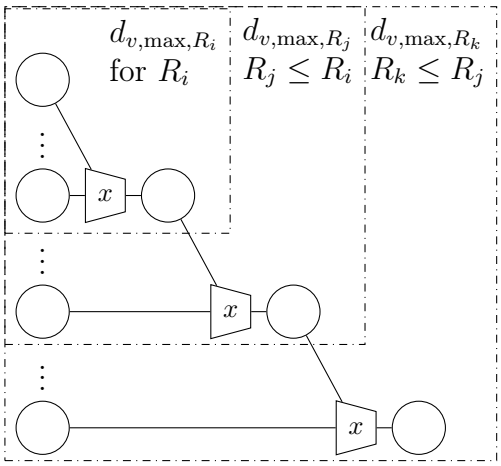


Figure 5.23: Schematic sketch of reuse of lookup tables for all rates based on the lookup tables for codes with higher rates.

systems. In contrast to state-of-the-art message-passing decoders, their mutual-information-based counterparts are not rate-compatible as the set of mutual information preserving mappings is matched to a specific rate. In this section, we devise an approach to reuse the mappings across several rates. In Section 5.4.3.1, the devised decoder which incorporates puncturing is evaluated. Afterwards, in Section 5.4.3.2, the impact of the bit resolution on the performance is analyzed. Section 5.4.3.3 discusses the impact of explicit and implicit message alignment (cf. Section 5.3.2) on the frame error rate performance. Simulation results for the proposed table reuse strategy from Section 5.4.2 are shown in Section 5.4.3.4. Finally, in Section 5.4.3.5, alternative implementations of the lookup table approach, as proposed in [MBB+15] are applied to the proposed design approach. A more detailed discussion of implementation aspects of the information bottleneck mappings is deferred to Section 5.5 and Section 5.6.

5.4.2.1 Reusing Tables of Information Bottleneck Decoders for Multiple Rates

As summarized in Section 5.1.1.1, PBRL codes consist of a high-rate code (HRC) and an incremental redundancy code (IRC). The set of possible rates R_i of a PBRL LDPC code with full rank $\mathbf{H}_{HRC}$ was given in Eq. (5.12). As discussed in the previous section, puncturing degree-one variable nodes has an impact on the effective degree-distributions $\boldsymbol{\lambda}_{\text{eff}}, \boldsymbol{\rho}_{\text{eff}}$ and, thus, also the maximum node degree $\lambda_{\text{eff,max}}$ depends on the number of punctured degree-one variable nodes. At the lowest rate, no degree-one variable node is punctured. Thus, one will observe the largest values $\lambda_{\text{eff,max}}$ across all rates R_i for $R_{i_{\max}}$. On the other hand, $\lambda_{\text{eff,max}}$ will be smallest for R_0.

Proposition 5.4.1. *For a fixed PBRL code with full rank* $\mathbf{H}_{HRC}$, $\lambda_{\mathit{eff},max,R_i} \leq \lambda_{\mathit{eff},max,R_j}, \forall i,j = 0,\ldots,i_{max}$ *if* $R_i > R_j$.

Proof. As the IRC adds more redundancy by activating parts in the Tanner graph, this is equivalent to augmenting $\mathbf{H}_{HRC}$. Hence, the node degree of the variable nodes can only be increased and not decreased. □

Please note that the number of needed lookup tables depends on the node degree (cf. Fig. 5.21). According to Proposition 5.4.1, designing an information bottleneck decoder for the highest code rate supported by a PBRL code yields variable nodes with the smallest number of lookup tables. Thus, similar to the table reuse in an irregular LDPC code, where a node with a larger degree is obtained by stacking new tables on top of a node with a lower degree, we propose to use the lookup tables for the highest rate as a starting point for the design of the lookup tables for lower rates. This is depicted in Fig. 5.23.

Example 5.5: Table Reuse in Information Bottleneck Decoders for Several Code Rates

Let us consider a fixed PBRL code with rates $R_i = 2/3, R_j = 1/2, R_k = 1/3$ and $d_{v,\max,R_i} = 9$, $d_{v,\max,R_j} = 15$, $d_{v,\max,R_k} = 27$. Please note that $d_{c,\max} = 19$ and is independent of the chosen rate. Without table reuse, mutual-information-maximizing mappings are designed for each rate for a fixed design-E_b/N_0. Simulation results for such a setting are shown later in Section 5.4.3.1. With the table reuse, first, the mappings for rate $R_i = 2/3$ with $d_{v,\max,R_i} = 9$ are designed for a fixed design-E_b/N_0 optimized for this rate. In the second step, the mappings derived for $R_i = 2/3$ are reused for $R_j = 1/2$ with $d_{v,\max,R_j} = 15$. As $d_{v,\max,R_i} - 1$ mappings could be reused, only $d_{v,\max,R_j} - d_{v,\max,R_i} = 6$ new mappings are designed and appended, as shown in Fig. 5.23. These new mappings are designed for a new design-E_b/N_0 optimized for $R_j = 1/2$. Thus, a subset of mappings is used mismatched, i.e., designed for another rate and also different channel conditions. However, it will be shown in Section 5.4.3 that only a small performance degradation is observed. This is due to the fact that message alignment is adapted to the degree distribution and compensates for the slight imperfections of the reused mappings. In the next step, the mappings for R_k are designed with a new design-E_b/N_0 optimized for $R_k = 1/3$ but reusing the $d_{v,\max,R_i} - 1$ mappings for $R_i = 2/3$ and the $d_{v,\max,R_j} - d_{v,\max,R_i}$ mappings optimized for $R_j = 1/2$. Please note that message alignment is not shown in Fig. 5.23 explicitly, but it is done for every code rate successively. In addition, also the mappings in the check nodes remain unchanged, and only message alignment is updated if needed.

5.4.3 Simulation Results and Evaluation

In this section, we present and discuss results obtained performing frame error rate simulations for an exemplary PBRL LDPC code from [CVD+15] which is referred to as Code 5 in this thesis:

- Code 5: Code from [CVD+15] with $K = 1032$ information bits code rates R_c ranging from $R_c = 1/3$ up to $R_c = 4/5$.

All involved lookup tables are constructed just once for a fixed design-E_b/N_0, which is optimized for each rate. The constructed lookup tables are then stored and applied for the entire E_b/N_0-range. Hence, the lookup table construction needs to be done only once and offline.

To evaluate different fascets within the construction of information bottleneck decoders for PBRL LDPC codes, this section is split into five subsections. Section 5.4.3.1 discusses the frame error rate performance of the PBRL decoder for particular code rates, i.e., only puncturing but not a rate-compatible design is covered. Still discarding rate-compatibility Section 5.4.3.2 investigates the impact of the bit resolution on the decoding performance. Afterwards, Section 5.4.3.3 compares information bottleneck decoders designed with the two different presented message alignments schemes, i.e., implicit and explicit message alignment. Finally, Section 5.4.3.4 combines all design steps proposed in the previous sections to analyze a rate-compatible PBRL LDPC information bottleneck decoder. Furthermore, Section 5.4.3.5 investigates alternative implementations to the lookup-table approach of the relevant-information-preserving mappings.

In general, we consider three reference schemes to compare the performance of our decoder. The decoding of a codeword is stopped after a maximum number of 100 decoding iterations or earlier if the syndrome check is successful. First, we consider a double-precision belief propagation decoder with a flooding schedule. The received samples are *not* coarsely quantized but represented with double precision, and the internal operations are additions at the variable node and box-plus at the check node. Second, we use the normalized min-sum algorithm (NMS) [JDE+05] with 6 bit resolution for the outgoing check node message and 6 bit for the outgoing variable node message. The internal precision of the node is slightly higher to prevent overflow. Again the inputs to the decoder, i.e., the channel outputs, are *not* coarsely quantized but represented with double precision. The operations here are additions at the variable nodes, but the normalized min-sum approximation is used at the check nodes (cf. Eq. (5.28)). Third, we use the offset-min-sum (OMS) decoder with only 4 bit resolution at the check node and offsets according to Eq. (5.30) and

Table 5.5: Simulation parameters of decoder and reference system for error rate simulations for the investigated PBRL LDPC code (Code 5).

decoder	node operation (check / var.)	precision exchanged messages	precision check node	precision variable node	channel quantizer
belief-propagation (belief-prop.)	box-plus/ addition	64 bit	64 bit	64 bit	64 bit
offset min-sum (OMS)	Eq. (5.29)/ addition	4 bit	4 bit	6 bit	4 bit
normalized min-sum (NMS) [JDE+05]	Eq. (5.28)/ addition	6 bit	6 bit	6 bit	64 bit
information bottleneck decoder (IB)	lookup table/ lookup table	4 bit	4 bit	4 bit	4 bit

6 bit at the variable node to prevent an overflow when adding the 4 bit messages received from the channel quantizer. Finally, we designed our proposed information bottleneck decoder for fully 4 bit integer architecture. This means, starting from the channel quantizer, which outputs 4 bit integers, the internal messages require only 4 bit and only *lookup* operations are performed. These lookups do not mimic any arithmetic function but realize the relevant-information preserving mappings found using the information bottleneck method.

5.4.3.1 Puncturing Using Message Alignment

In this subsection, we investigate the proposed generalized decoder design to cover punctured variable nodes. Here, Code 5 was used. Furthermore, in this subsection, the decoder mappings were designed for each code rate individually with an individual design-E_b/N_0, i.e., the table reuse from Section 5.4.2 is not applied.

The most important parameters of the applied decoders are summarized in Table 5.5 for a quick overview. First, we consider a decoder designed for a fixed rate of $R_c = 0.5$. The results are shown in Fig. 5.24a. As expected, the belief-propagation (BP) algorithm (●-marker) achieves the best frame error rate performance, but at the same time, has the highest computational complexity (cf. Table 5.5). Although all applied operations in the information bottleneck decoder (⧫-marker) are simple lookups, the decoder performs only less than 0.2 dB worse than the benchmark. The results are even more remarkable when considering the tremendous gap to the offset-min-sum and normalized min-sum decoders with an even slightly higher resolution.

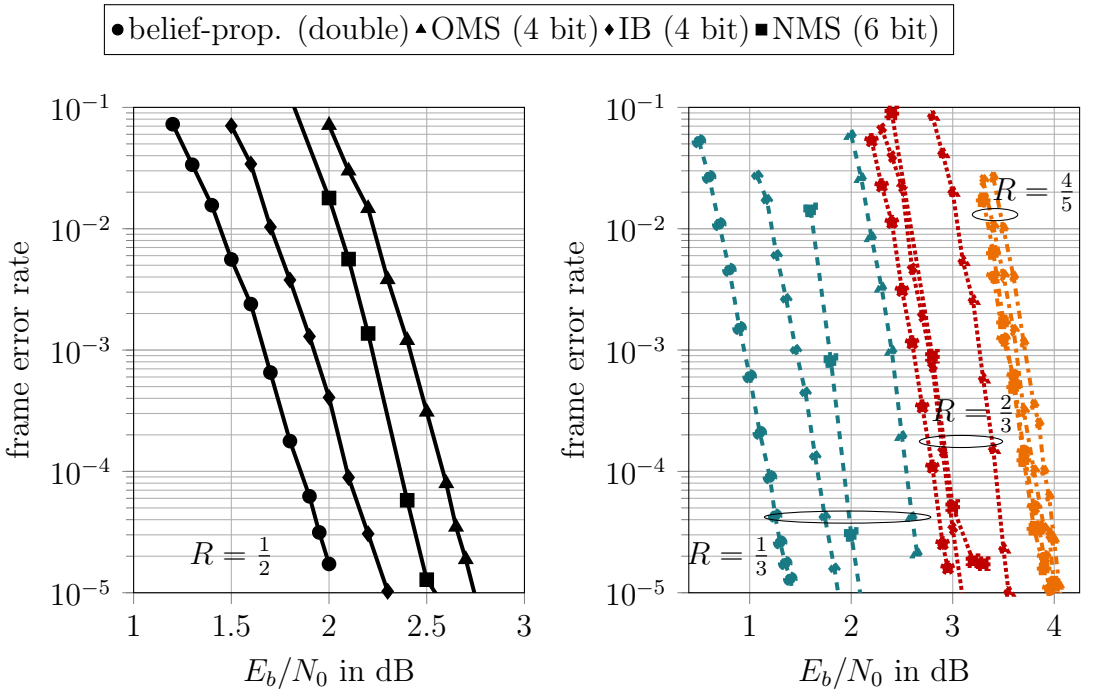


Figure 5.24: Frame error rates for the presented information bottleneck decoder, and the reference schemes summarized in Table 5.5 for the considered PBRL LDPC code with code rate (a) $R_c = 1/2$ (black, solid), (b) $R_c = 1/3$ (blue, dashed), 2/3 (dark red, dotted), 4/5 (dark orange, dash dot).

Please note that PBRL codes typically have variable nodes with very large degrees. The gap of 0.75 dB noticed in Fig. 5.24a reveals that a conventional offset-min-sum decoder, which exchanges only 4 bit messages, cannot be used for PBRL codes with such a coarse quantization since the dynamic range of the LLRs cannot be captured appropriately (cf. Section 5.5.2). As indicated by the frame error rate curve for the 6 bit NMS decoder, choosing a finer resolution can reduce this gap. However, with the generalized design for information bottleneck decoders, both challenges, i.e., puncturing and rate-compatible design, can be efficiently tackled to enable fully 4 bit decoders for PBRL codes. Fig. 5.24b shows the results for various other rates. The belief propagation decoder with double-precision resolution and no channel quantizer achieves the best performance for all considered rates. However, again we observe that the proposed information bottleneck decoder operates very close to this benchmark. Interestingly, the proposed schemes outperform the 4 bit offset min-sum decoder and the 6 bit NMS decoder for all investigated rates.

Table 5.6: Simulation parameters of investigated information bottleneck decoders.

exchanged messages	check node	variable node	channel quantizer
3 bit	3 bit	3 bit	3 bit
4 bit	4 bit	4 bit	4 bit
5 bit	5 bit	5 bit	5 bit

5.4.3.2 Impact of the Bit Resolution on the Decoder Performance

For the considered Code 5, this subsection investigates the impact of the chosen bit resolution on the performance. The respective bit resolutions used are summarized in Table 5.6. The results are shown in Fig. 5.25. For the sake of clarity, only the results for $R_c = 1/3$ are shown and the reference systems are limited to the offset min-sum decoder and the belief propagation decoder with the parameters from Table 5.5. However, similar results were obtained for all other code rates. Interestingly, it can be observed that for a 5 bit information bottleneck decoder, the performance gap to double-precision belief propagation decoding nearly vanishes. Furthermore, it can be observed that the proposed 3 bit information bottleneck decoder shows the same performance as the offset min-sum decoder, which uses 4 bit for the channel quantizer, 4 bit for the exchanged messages and 6 bit for the variable node operation (cf. Table 5.5).

5.4.3.3 Impact of Different Implementations of Message Alignment

As proposed in Section 5.3.2, the message alignment approach can be realized either based on $p(x, t^{\text{out}}|d)$ termed explicit message alignment or based on $p(x, \mathbf{t}^{\text{in}}|d)$ referred to as implicit message alignment. In Fig. 5.26, the impact of the selected message alignment approach on the frame error rate performance is investigated. It is shown that the performance gain achieved by the implicit approach is 0-0.1 dB over the explicit message alignment approach. The slight performance degradation of explicit message alignment is caused by using the compressed representation t^{out} of $\mathbf{t}^{\text{in}}$ in the alignment step instead of $\mathbf{t}^{\text{in}}$. However, when considering the concatenated scheme with reuse, the implicit message alignment approach has slightly higher memory complexity. The implicit alignment mapping has $|\mathcal{T}|^2$ input combinations, whereas explicit message alignment works on the compressed random variable directly and has only $|\mathcal{T}|$ input combinations (cf. Fig. 5.13 and Fig. 5.14).

5.4.3.4 Memory Considerations and Table Reuse

Besides supporting puncturing, the proposed generalized decoder design also enables the reuse of lookup tables across several rates. In contrast to state-of-the-art

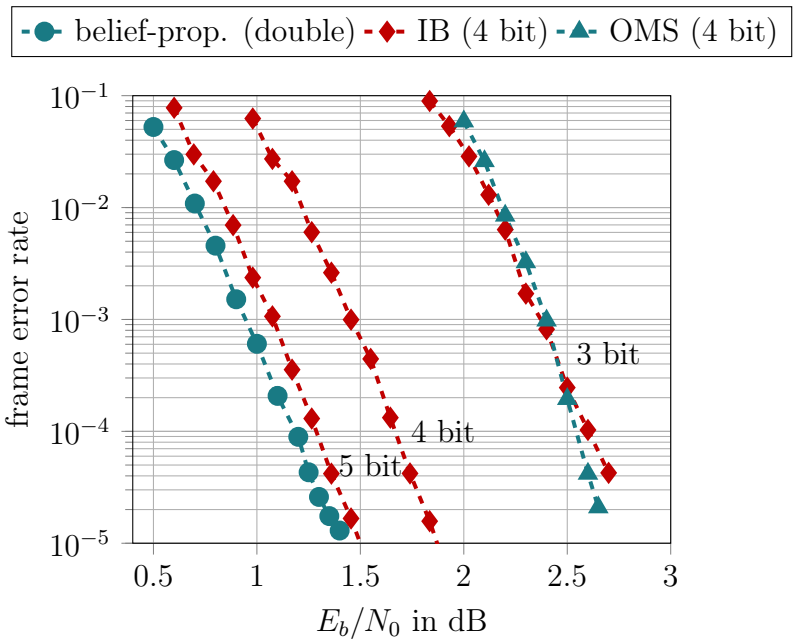


Figure 5.25: Frame error rate simulations for the proposed decoder for 3 bit, 4bit and 5 bit resolution for code rate $R_c = 1/3$. Only belief propagation decoder and the offset min-sum decoder are shown as reference, with parameters according to Table 5.5.

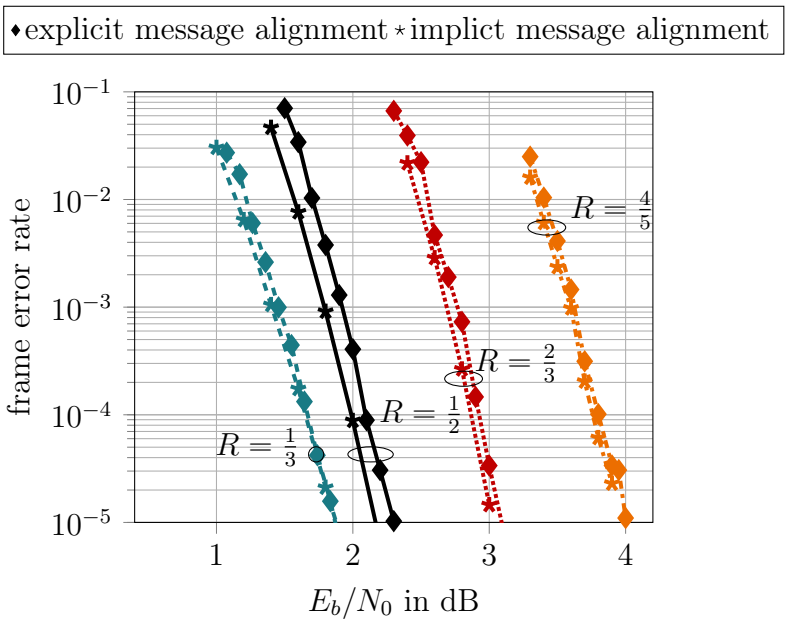


Figure 5.26: Frame error rate simulations for implicit and explicit message alignment.

information bottleneck decoders where one set of tables was designed for only one particular rate, the devised decoder using the design technique from Section 5.4.2

Table 5.7: Memory requirements per iteration for information bottleneck decoders with and without table reuse for a 4 bit decoder.

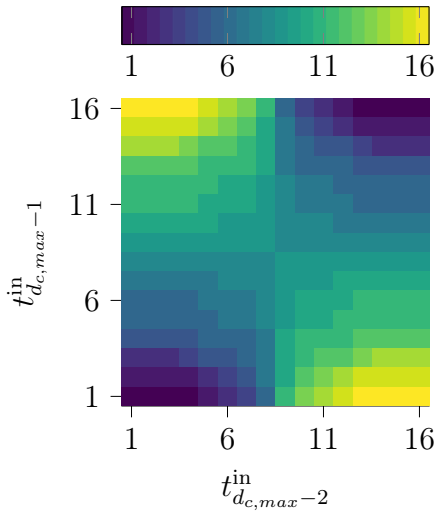
R_c	$d_{c,max}$	check node memory		$d_{v,max}$	variable node memory	
		no reuse	reuse		no reuse	reuse
4/5	19	8.7 kB	8.7 kB	6	2.6 kB	2.6 kB
2/3	19	8.7 kB	-	9	4.1 kB	1.5 kB
1/2	19	8.7 kB	-	15	7.2 kB	3.1 kB
1/3	19	8.7 kB	-	27	13.3 kB	6.1 kB
Total		34.8 kB	8.7 kB		27.1 kB	13.3 kB

requires only one set of tables for all rates. Fig. 5.28 shows the simulation results, where the information bottleneck decoder optimized for each rate proposed in the previous section is included as a reference. As described in Section 5.4.2, the decoder construction starts with the highest code rate, i.e., $R_c = 4/5$. Thus, no difference between the decoders can be observed for this rate. The lookup tables for all lower code rates are built on top of the lookup tables from the code with a higher rate. Here, we observe a small performance degradation below 0.1 dB due to the mismatched table reuse.

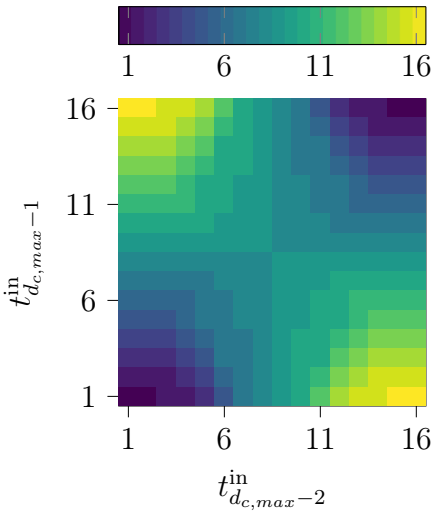
According to Section 5.3.3, the memory of one two-input lookup table is given as $\frac{|\mathcal{T}|^2 \cdot |\mathcal{T}|}{8}$ byte if $|\mathcal{T}_{ch}| = |\mathcal{T}|$. Table 5.7 summarizes the overall required memory demand. It can be observed that for the considered PBRL code, the memory per iteration can be reduced by a factor 3 for the check nodes and by a factor of approximately 1.8 for the variable nodes. Besides the reduction in memory, the table reuse allows for more efficient implementations as the same set of lookup tables can also be used for multiple code rates. Please note that the memory requirements given in Table 5.7 hold only for decoder implementations on a digital signal processor or software-defined radio where the lookup tables are stored in memory. In general, the mappings could also be efficiently implemented as a static logic synthesis on an FPGA or ASIC [GBM+18]. A more detailed discussion concerning hardware aspects of information bottleneck decoders is provided in Section 5.6.

5.4.3.5 Implementing the Lookup Tables

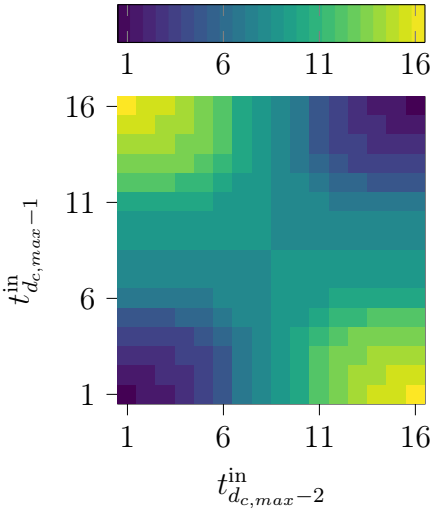
As described in Chapter 3, the general aim of the information bottleneck is to obtain a mapping $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$, which preserves the relevant information. Typically, these mappings depend on the code rate and iteration. Fig. 5.27 shows the check node lookup tables in the last step of the cascaded structure of two-input tables for a code rate $R_c = 0.5$ and different iterations, i.e., iteration 1, 50 and 100. It can be observed that tables change over the iterations.



(a) Check node mapping at iteration 1



(b) Check node mapping at iteration 50



(c) Check node mapping at iteration 100

Figure 5.27: Input-output relation of the check node lookup table designed using the information bottleneck method at different iterations.

In general, the lookup table implementation is just one way to realize the mapping. However, the general design concepts discussed in previous sections are crucial for any implementation of the learned function $f(\mathbf{t}^{\text{in}})$. In the literature, alternative implementations were proposed, which are discussed in detail in the next section. One possible alternative is the MIN-LUT decoder proposed in [MBB+15], where the min-sum operation at the check node is performed using the integer-valued cluster indices as described in Section 5.2.2.2. Simulation results for such a hybrid approach where the lookup tables replace the variable node operation but the check node performs the min-sum operation (cf. Fig. 5.6b) are shown in Fig. 5.29. In turn, the mapping at the check node is fixed for all iterations and does not change. However, the variable node mappings are still adapted to the evolving densities and,

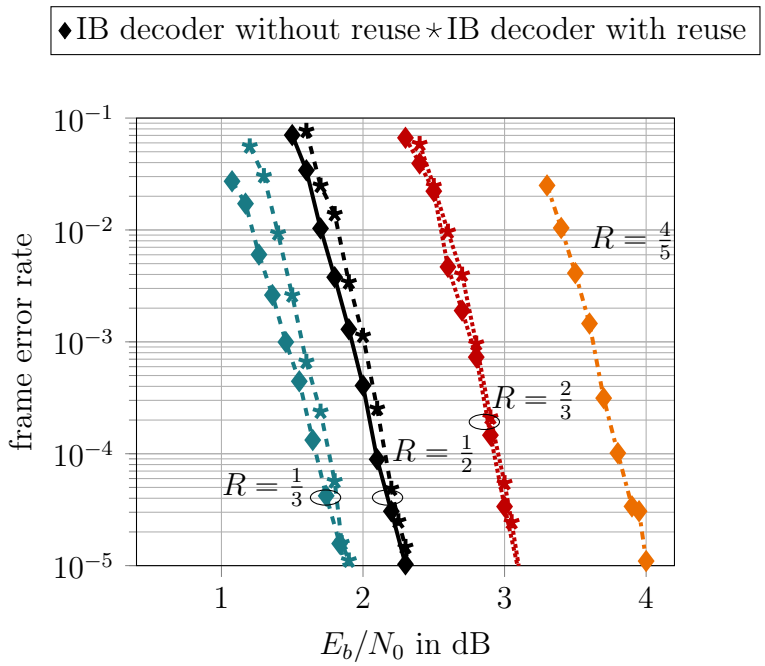


Figure 5.28: Frame error rate simulations where one static set of lookup tables is used for all rates.

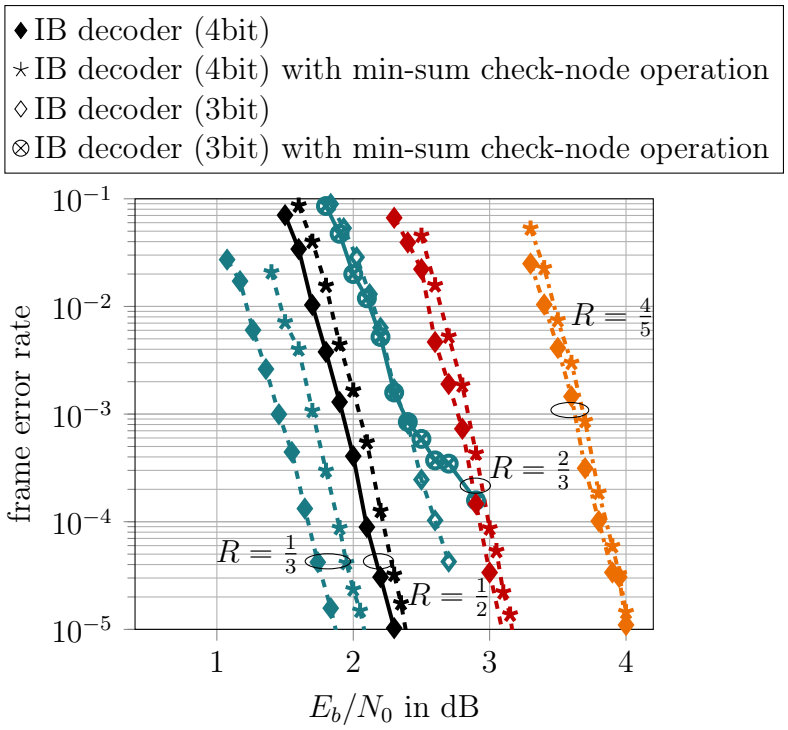


Figure 5.29: Frame error rate simulations where the min-sum update rule replaces all lookup tables in the check nodes (cf. Fig. 5.6b).

thus, change in each iteration. Again, only the results for the proposed decoder with the parameters from Table 5.5 are shown as reference for the sake of clarity. It can be observed that the performance of the proposed decoder with the min-sum update rule of Fig. 5.6b is much better than the state-of-the-art offset min-sum decoder (cf. Fig. 5.24a and Fig. 5.24b), especially for the lowest rate, i.e., $R_c = 1/3$. For example, at a FER of 10^{-4} the offset min-sum decoder shown in Fig. 5.24b is outperformed by 0.6 dB. This is an interesting observation as the 4 bit min-sum decoders are typically known to work fairly bad for low code rates. Only if the resolution is reduced further, e.g., down to 3 bit, the hybrid approach with the min-sum operation at the check node shows an early error floor as depicted in Fig. 5.29 for code rate $R_c = 1/3$.

5.5 Brief Overview and Comparison of Coarsely Quantized LDPC Decoders without the Information Bottleneck Method

In the previous section, the devised information bottleneck decoder was presented for versatile settings and families of LDPC codes. In literature, following the original idea of mutual-information-based decoder design from [LT05], different approaches to quantized decoding developed as summarized in Fig. 5.1 were proposed. This section focuses only on the most recent advances with a clear connection to the information bottleneck decoding approach. The main focus of this section is not to provide a detailed review of these alternative approaches but highlight the different assumptions in the decoder design.

5.5.1 Finite-Alphabet Iterative Decoding (FAID)

One of the first proposed ways of dealing with coarse quantization is the finite-alphabet iterative decoding (FAID) approach [PDD+11; PDD+13; DVP+13]. This approach is conceptually related to the information bottleneck decoder. In the FAID approach, hand-optimized lookup tables are designed which replace the conventional node operations. The FAID approach was shown to achieve very competitive performance on a binary symmetric channel with regular LDPC codes, despite coarse quantization. While similar in operation to the lookup tables developed for the FAID approach [PDD+11; PDD+13; DVP+13], the tables used in information bottleneck decoders are designed analytically using density evolution. Furthermore, to the best of our knowledge, the originally proposed FAID decoders are restricted to regular

LDPC codes and very specific channel models like binary symmetric channels and do not support irregular codes, puncturing, or rate-compatibility and are thus not considered as a benchmark system.

5.5.2 Optimized MIN-LUT Decoders

Starting with regular LDPC codes [MBB+15], Meidlinger et al. developed coarsely quantized LDPC decoders, which can be interpreted as a hybrid approach between the information bottleneck decoder and the conventional min-sum decoder. Assuming for cluster indices that are sorted with respect to their associated LLR values, it was proposed to replace the relevant-information-preserving mapping $p(t^{\text{out}}|\mathbf{t}^{\text{in}})$ with a static arithmetic operation, i.e., Fig. 5.6b. However, as discussed in [MMB20], this approach can not be applied to arbitrary code ensembles but instead requires optimized codes that avoid certain stopping set conditions. These problems are well-known for coarsely quantized min-sum decoding and discussed in great detail in [SKW06]. Please note that this stands in clear contrast to the information bottleneck decoder design with message alignment at the variable *and* check nodes presented in this thesis, which works for arbitrary degree distributions and, hence, irregular LDPC codes. A comparison of the bit error rate performance of both decoding approaches is given in Fig. 5.30. The left plot in Fig. 5.30 shows the bit error rates for the rate $R_c = 0.5$ from the DVB-S2 standard considered in previous sections. It can be observed that the MIN-LUT approach shows an early error floor. For the optimized LDPC code according to [MMB20] (cf. right plot in Fig. 5.30), the error floor can not be observed, instead only a small performance degradation due to the approximated check node operation in the MIN-LUT decoder remains compared to the information bottleneck decoder. For the simulations, a quantization of 4 bit was used for the information bottleneck decoder and the MIN-LUT decoder.

5.5.3 One and Two Bit Message Passing Decoding

A very extreme case of coarse quantization was investigated by Steiner et al. in [SBM+19]. However, in contrast to information bottleneck decoders, these decoders assume a double-precision channel log-likelihood ratio, i.e., no channel output quantizer. Further, also the node operations perform the conventional sum-product decoding node updates with double-precision internally. Hence, the focus is solely on the quantization of the exchanged messages. However, the quantization of these messages follows a mutual-information-preserving design objective with close relations to the work by Wang et al. [WCS+11].

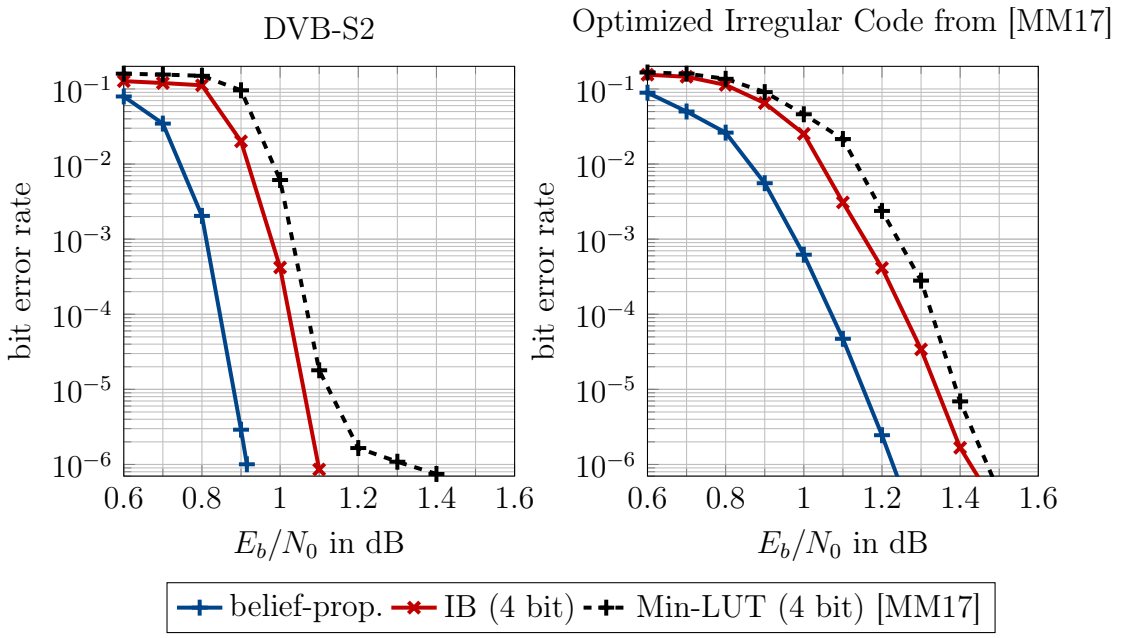


Figure 5.30: Comparison of the information bottleneck decoder to the MIN-LUT decoder presented in [MM17] for an standardized LDPC code and an LDPC code optimized for MIN-LUT decoding.

5.5.4 Computational Domain Decoding and Reconstruction Function Decoding

Pointing into a somewhat similar direction as [SBM+19], He et al. proposed to replace the mutual-information-based mappings, e.g., implemented as lookup tables, by simple high-precision arithmetical operations [HCM19]. Here, in contrast to the actual information bottleneck decoder presented in this thesis, arithmetical operations are not replaced by simple relevant-information-preserving mappings. Instead, mutual-information-preserving arithmetical operations are designed. The idea is to convert the cluster t_i^{in} indices by a non-linear transformation $\phi_c(t_i^{\text{in}})$ into a so-called *computational domain*. Within this computational domain, arithmetical operations related to the belief propagation update are computed. This computational domain typically has a resolution of 8-10 bits. After performing the respective node operations in the computational domain, the result is quantized to the desired resolution of the exchanged messages. The determined node operations in the computational domain are solely based on additions. This is natural for the variable node as the converted cluster indices in the computational domain can be interpreted as LLRs. However, the check node operation in the computational domain decoder from [HCM19] follows from an approximation of $\tanh\left(\frac{L_{\mathrm{v}\to\mathrm{c}_i}}{2}\right)$ in Eq. (5.21) as

$$\prod_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} \tanh\left(\frac{L_{\mathsf{v}_{j'}\to\mathsf{c}_i}}{2}\right) = \prod_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} p(\mathsf{B}_{\mathsf{v}_{j'}}=0|y) - p(\mathsf{B}_{\mathsf{v}_{j'}}=1|y) \tag{5.64}$$

$$= \left[\prod_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} \operatorname{sgn}\Big(p(\mathsf{B}_{\mathsf{v}_{j'}}=0|y) - p(\mathsf{B}_{\mathsf{v}_{j'}}=1|y)\Big)\right] \cdot \exp\left(\sum_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} \log|p(\mathsf{B}_{\mathsf{v}_{j'}}=0|y) - p(\mathsf{B}_{\mathsf{v}_{j'}}=1|y)|\right) \tag{5.65}$$

$$= \prod_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} \operatorname{sgn}\left(\phi_c^*(t_{j'}^{\text{in}})\right) \exp\left(\sum_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} -|\phi_c^*(t_{j'}^{\text{in}})|\right) \tag{5.66}$$

$$\approx \left[\prod_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} \operatorname{sgn}\left(\phi_c(t_{j'}^{\text{in}})\right)\right] \sum_{\mathsf{v}_{j'}\in N(\mathsf{c}_i)\setminus\{\mathsf{v}_j\}} |\phi_c(t_{j'}^{\text{in}})| \tag{5.67}$$

where it is argued in [HCM19] that $|\phi_c^*(t_{j'}^{\text{in}})| = -\log|p(\mathsf{B}_{\mathsf{v}_{j'}}=0|y) - p(\mathsf{B}_{\mathsf{v}_{j'}}=1|y)| \approx \frac{1}{|L_{\mathsf{v}_{j'}\to\mathsf{c}_i}|} = |\phi_c(t_{j'}^{\text{in}})|$ and that the exponential function can be dropped as due to the subsequent quantization only the relative values instead of the absolute values matter. The work of [HCM19] is mainly limited to regular LDPC codes. In [WSW+20], we investigated other reconstruction functions, i.e., different arithmetical operations within the computational domain, and also applied the approach to irregular LDPC codes. The information bottleneck decoder and the computational domain decoder are compared in Fig. 5.31 for the regular (3,6) LDPC code considered in Section 5.2. It can be observed that the computational domain decoder operates in the small gap (0.02 dB) between the closed and the opened information bottleneck decoder. As the closed information bottleneck decoder, the computational domain decoder does not lose additional relevant information due to the concatenation of two-input compression steps. However, the computational domain decoder does lose relevant information due to the imperfect transformation in the computational domain, restricted to 10 bit resolution, and the approximated node operation at the check node. In contrast to the information bottleneck decoder, the computational decoder employs a lookup table for the transformation *and* a high-precision arithmetical unit which computes a summation. The exchanged messages in both decoders are restricted to 4 bit. The simulation parameters of the decoders are again summarized in Table 5.8. It can be concluded that the computational domain decoder provides

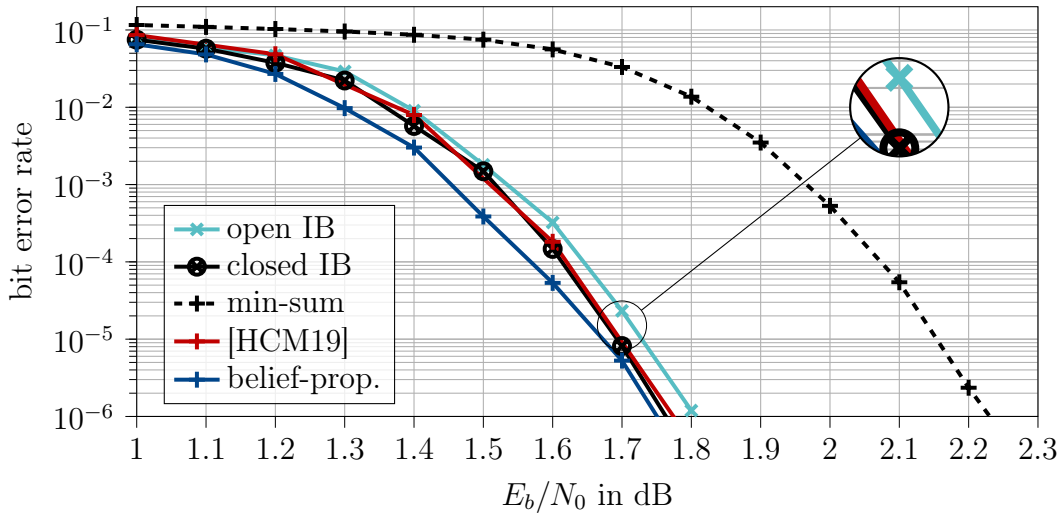


Figure 5.31: Bit error rate simulations for length $N = 8000$ $(3, 6)$ regular LDPC code with $i_{\max} = 50$ decoding iterations over AWGN channel with BPSK from [Mac20] with identifier 8000.4000.3.483 including the computational domain decoder from [HCM19].

another, hybrid approach towards mutual-information-based decoder design, which pairs coarse quantization of the exchanged message but high-resolution arithmetical operations of the internal node operations as in [SBM+19].

5.6 Hardware Aspects of Information Bottleneck Decoders for LDPC Codes

The review of alternative decoder designs in Section 5.5 already outlined that the choice of a particular implementation is often closely related to the actual hardware architecture at hand and the respective hardware capabilities. In this section, favorable implementations of the learned mappings $p(t^{\text{out}}|\mathbf{t}^{\text{in}})$ for different hardware architecture are presented and analyzed. These investigations were conducted in close cooperation with the Fraunhofer Institute FKIE in Wachtberg [LBT+18] and the Microelectronic Systems Design (EMS) department at TU Kaiserslautern.

5.6.1 Digital Signal Processor (DSP)

A digital signal processor is the fundamental building block of modern software-defined radio applications. In contrast to application-specific chip design (cf. Section 5.6.2), software-defined radios are a general-purpose signal processing platform to adjust the units in a communication chain, e.g., channel decoding and modulation, by simple software updates. The performance of information bottleneck decoders on

Table 5.8: Simulation parameters of decoders compared in Fig. 5.31 for Code 1.

decoder	node operation (check / var.)	precision exchanged messages	precision check node	precision var. node	channel quantizer
belief-propagation (belief-prop.)	box-plus/ addition	64 bit	64 bit	64 bit	4 bit
min-sum	Eq. (5.27)/ addition	4 bit	4 bit	6 bit	4 bit
information bottleneck decoder (IB)	lookup table/ lookup table	4 bit	4 bit	4 bit	4 bit
comp. domain [HCM19]	LUT+addition/ LUT+addition	4 bit	10 bit	10 bit	4 bit

a DSP was studied in [LBT+18]. On a DSP, the relevant-information-preserving-mappings are stored in memory. Thus, the conventional node operation simplifies to a computation of the respective memory address and a memory read. Assuming the lookup tables $\mathbf{lut}_m^i$ for all iterations i, $i = 1, \ldots, i_{\max}$ and two-input decompositions m are stored in a single vector

$$\mathbf{LUT} = \left[\mathbf{lut}_0^1, \mathbf{lut}_1^1, \ldots, \mathbf{lut}_M^1, \ldots, \mathbf{lut}_M^{i_{\max}}\right] \tag{5.68}$$

for each node type, the respective memory address of the cluster index t or t^{out} based on $\mathbf{t}^{\text{in}} = [t_1^{\text{in}}, t_2^{\text{in}}]^{\mathsf{T}}$ is found as [LBT+18]:

$$t = \mathbf{LUT}\left[(i-1)M|\mathcal{T}|^2 + (m-1)|\mathcal{T}|^2 + t_1^{\text{in}}|\mathcal{T}|^2 + t_2^{\text{in}}\right] \tag{5.69}$$

if $|\mathcal{T}| = |\mathcal{T}_{ch}|$.

The overall throughput is mainly affected by the duration of the memory access. As proposed in [LBT+18], with efficient use of the cache, a superior performance of the information bottleneck decoder compared to a min-sum decoder by 20%-40% was achieved in terms of the net throughput.

5.6.2 Field Programmable Gate Array (FPGA)

The first approach to more application-specific hardware implementation, i.e., field programmable gate array (FPGA) or ASIC, of information bottleneck decoder was presented in [GBM+18]. For a regular LDPC code and the MIN-LUT decoding approach from [MBB+15], a fully-parallel unrolled decoder implementation was

Table 5.9: Exemplary truth table for an information-bottleneck-variable-node-two-input mapping with $|T_c| = 4$.

$[t_1^{\text{in}}, t_2^{\text{in}}]^\mathsf{T}$	$[\text{bin}(t_1^{\text{in}}), \text{bin}(t_2^{\text{in}})]^\mathsf{T}$	$\text{bin}(t^{\text{out}})$	t^{out}
$[0,0]^\mathsf{T}$	$[0,0,0,0]^\mathsf{T}$	$[0,0]^\mathsf{T}$	0
$[0,1]^\mathsf{T}$	$[0,0,0,1]^\mathsf{T}$	$[0,0]^\mathsf{T}$	0
$[0,2]^\mathsf{T}$	$[0,0,1,0]^\mathsf{T}$	$[0,0]^\mathsf{T}$	0
$[0,3]^\mathsf{T}$	$[0,0,1,1]^\mathsf{T}$	$[0,1]^\mathsf{T}$	1
$[1,0]^\mathsf{T}$	$[0,1,0,0]^\mathsf{T}$	$[0,0]^\mathsf{T}$	0
$[1,1]^\mathsf{T}$	$[0,1,0,1]^\mathsf{T}$	$[0,1]^\mathsf{T}$	1
$[1,2]^\mathsf{T}$	$[0,1,1,0]^\mathsf{T}$	$[0,1]^\mathsf{T}$	1
$[1,3]^\mathsf{T}$	$[0,1,1,1]^\mathsf{T}$	$[1,0]^\mathsf{T}$	2
$[2,0]^\mathsf{T}$	$[1,0,0,0]^\mathsf{T}$	$[0,1]^\mathsf{T}$	1
$[2,1]^\mathsf{T}$	$[1,0,0,1]^\mathsf{T}$	$[1,0]^\mathsf{T}$	2
$[2,2]^\mathsf{T}$	$[1,0,1,0]^\mathsf{T}$	$[1,0]^\mathsf{T}$	2
$[2,3]^\mathsf{T}$	$[1,0,1,1]^\mathsf{T}$	$[1,1]^\mathsf{T}$	3
$[3,0]^\mathsf{T}$	$[1,1,0,0]^\mathsf{T}$	$[1,0]^\mathsf{T}$	2
$[3,1]^\mathsf{T}$	$[1,1,0,1]^\mathsf{T}$	$[1,1]^\mathsf{T}$	3
$[3,2]^\mathsf{T}$	$[1,1,1,0]^\mathsf{T}$	$[1,1]^\mathsf{T}$	3
$[3,3]^\mathsf{T}$	$[1,1,1,1]^\mathsf{T}$	$[1,1]^\mathsf{T}$	3

proposed, which was 3.1 times more area efficient and two times more energy-efficient compared to a min-sum decoder with serial message-transfer architecture [GBM+18].

For an FPGA implementation of the information bottleneck decoder, the mappings $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$ are not treated as a lookup table, which is stored in memory. Instead, a hard-wired logic circuit is generated by a synthesis tool which realizes $f(\mathbf{t}^{\text{in}})$ using Boolean algebra and respective operations. Starting from a large truth table, as exemplarily shown in Table 5.9, an optimized Boolean expression is derived. As this optimization is typically NP-hard, several heuristic optimization techniques exist [BHM+84]. In this thesis, the *Espresso* multi-valued PLA minimization tool from the UC Berkely was used for optimization. Please note that in Table 5.9, $\mathcal{T} = \{0, 1, 2, |\mathcal{T}| - 1\}$ to enable a direct conversion into the binary representation of t^{in}, denoted $\text{bin}(t^{\text{in}})$. Fig. 5.32 depicts the optimized logic circuit to represent the truth table using NAND and NOR gates.

For the synthesis of the entire decoder in cooperation with TU Kaiserslautern, Code 4, i.e., the code from the WiMAX standard is considered with $i_{\max} = 5$. A comparison of the chip area of the information bottleneck decoder compared to the offset min-sum decoder is provided in Table 5.10. Please note that in contrast to [GBM+18] these results were obtained for an irregular LDPC code and, thus, an information bottleneck decoder designed using message alignment. The synthesis

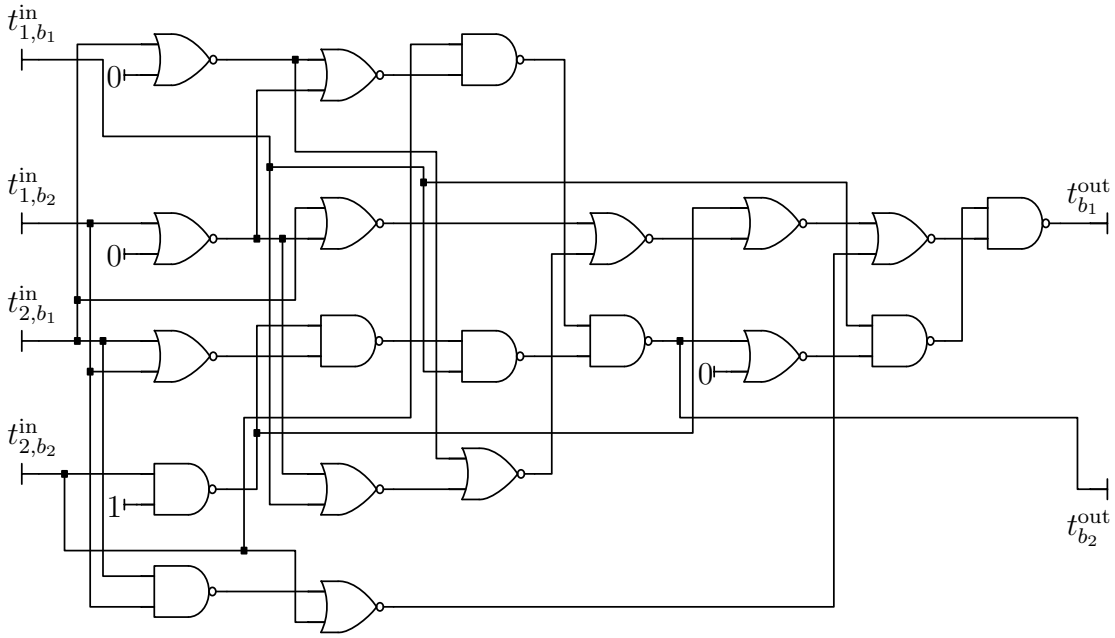


Figure 5.32: Logic circuit comprised of NAND and NOR gates of the optimized truth table in Table 5.9.

results in Table 5.10 are shown for a dummy codeword size of 24 bit. Furthermore, Table 5.10 shows that the information bottleneck decoder for 3 bit is generally smaller and faster than a 6 bit offset min-sum decoder. In particular, the 3 bit information bottleneck decoder with the static min-LUT check node operation is approximately 43% smaller and 14% faster.

Fig. 5.33 depicts the bit error rate performance for different decoders for code 4 and $i_{\max} = 5$. As the focus is on the comparison of practically implementable decoders, the performance of double-precision belief propagation decoding is not shown in Fig. 5.33. In turn, it can be observed that the information-bottleneck decoder with the static check node mapping (IB/min) and an internal 4 bit resolution performs best. This decoder outperforms a 4 bit min-sum decoder significantly and is also superior to the 6 bit min-sum decoder. However, from the synthesis report summarized in Table 5.10 we conclude that this decoder has a larger core area. Nevertheless, the information bottleneck decoder with 3 bit internal resolution performs similar to the 6 bit min-sum decoder. Please remember that the 3 bit information bottleneck decoder with min-LUT check node operation was 43% smaller and 14% faster. Furthermore, Fig. 5.33 also underlines that for a 3 bit min-sum decoder the quantization is too coarse resulting in a significantly deteriorated performance. Thus, one concludes that especially for LDPC decoder implementations which require a very

Table 5.10: Comparison of hardware complexity of a min-sum decoder and the information bottleneck decoder for Code 4 and $i_{\max} = 5$.

Architecture	Offset Min-Sum (6 bit)	IB (4 bit)	IB/min (4 bit)	IB (3 bit)	IB/min (3 bit)
Frequency (MHz)	571	559	500	671	662
Throughput (Gbps)	13.7	13.4	12.0	16.1	15.9
Core Area (μm^2)	**122567**	**294032**	**179396**	**92034**	**70222**

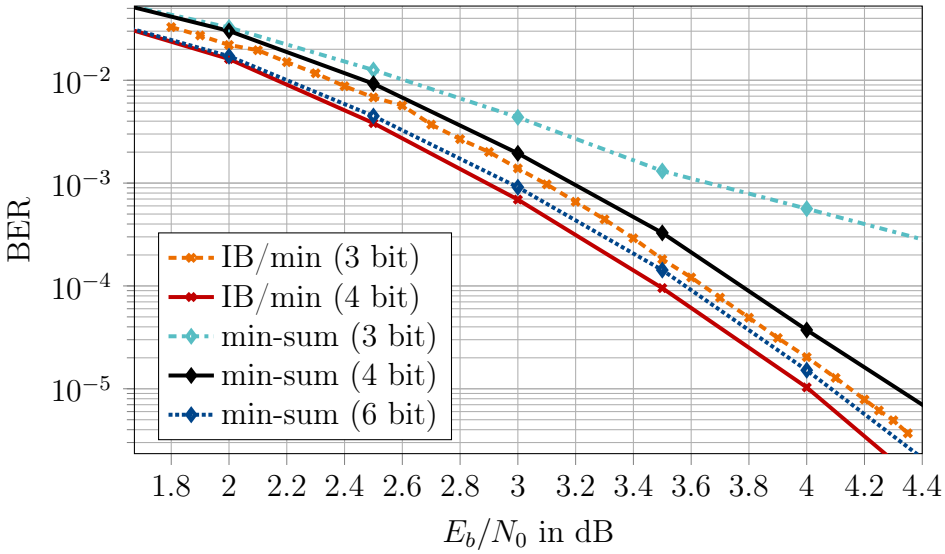


Figure 5.33: Bit error rate performance for Code 4 and $i_{\max} = 5$.

coarse quantization, either due to routing and wiring constraints or latency requirements, the presented 3 bit information bottleneck decoder appears as an appealing decoding approach.

5.7 Summary

In this chapter, the so-called information bottleneck decoder was presented for a large variety of LDPC code families. Starting with regular LDPC codes, the concept of discrete density evolution was introduced to track the evolution of discrete probability mass functions while decoding iteratively. These distributions served as input to the information bottleneck method returning highly informative compression mappings which preserve the maximum amount of relevant information. The

mappings were used to replace the arithmetical node operations in conventional decoders. As these mappings are only generated once and offline for a fixed design E_b/N_0, it was described how this crucial design parameter is obtained utilizing EXIT charts. In turn, an LDPC decoder was designed, which exchanges only integer-valued cluster indices representing the exchanged soft information in an effective manner. Provided simulation results revealed that practically relevant conventional decoders, e.g, a min-sum decoder with 4 bit precision, were outperformed even with a fully 3 bit information bottleneck decoder. Also, the gap to the double-precision benchmark system, i.e., a belief-propagation decoder, was very small ($\approx$ 0.1 dB for a 4 bit information bottleneck decoder).

The size of the mappings of the information bottleneck decoders increases exponentially with the node degree. Therefore, different node decompositions were proposed and compared to the so-called closed node.

It was identified that the design of coarsely quantized decoders for irregular LDPC codes is itself a very fundamental and challenging problem. Generalizing the information bottleneck decoder design principle to irregular LDPC codes, a connection to the message alignment problem in the decoder design was established. It was shown that with the application of message alignment, the decoder performance could be significantly improved.

In the next step, the problem of puncturing and rate compatible design was tackled. Considering a particular LDPC code family, i.e., PBRL code, which is also considered in the 5G standard, the information bottleneck decoder design was further generalized. It was revealed that different notions of puncturing have to be treated differently but again can be interpreted as an instance of the message alignment problem. Exploiting the incremental structure of PBRL codes, a very efficient reuse pattern of mappings across code rates was presented which reduces the number of distinct lookup tables to be stored significantly.

Afterwards, it was shown that in literature, many decoding approaches exist which build upon the universal decoder design framework devised in this chapter that leverages the relevant-information-maximizing design objective. Most of these approaches employ approximations or modifications of the theoretical information-optimum design presented in this chapter to satisfy very application-specific assumptions and constraints. Nonetheless, bit error rate simulations proved that the presented information bottleneck decoding design principle is theoretically superior to all analyzed approaches.

Finally, hardware aspects of the information bottleneck decoders were studied. Here, two different architectures, i.e., a DSP and FPGA, were considered. As a result, it was shown that for both architectures information bottleneck decoders could be designed with a reduced implementation complexity and increased throughput compared to state-of-the-art decoder implementations.

Chapter 6

Decoding Non-Binary Low-Density Parity-Check Codes Using the Information Bottleneck Method

In the previous chapter, we applied the information bottleneck to the design of coarsely-quantized channel decoders, which preserve the maximum amount of relevant information. Different types of channel codes, i.e., regular, irregular, and rate-compatible LDPC codes, were studied under bit-metric decoding, i.e., also if higher-order modulation schemes would be used for transmission as shown in [LSB17] the actual decoder operates on the information bits rather than on the information symbols. Here, so-called non-binary LDPC codes are of large interest.

The generalization of binary LDPC codes to non-binary symbol alphabets over higher-order Galois fields with field order q was proposed right after the rediscovery of binary LDPC codes by MacKay [Mac99]. However, the decoding of these codes using sum-product decoding is computationally much more expensive than decoding their binary counterparts, as explained in the next section.

Despite many very important works [Sav08; DF07; DF05; WSM04], the development of efficient decoding methods for non-binary LDPC codes continues to be an interesting subject of current research for practical purposes as non-binary LDPC codes have better error correction properties for short block lengths than binary LDPC codes. The latter unfold their capacity approaching behavior only for very large codeword lengths [RL09]. Therefore, especially in 5G related scenarios such as massive-machine-type communications and ultra-reliable low latency communication (uRLLC), non-binary LDPC codes could be promising candidates if decoders with affordable complexity were available.

In this chapter, the design of information bottleneck decoders for symbol-metric decoding using non-binary LDPC codes is presented, which was devised in the scope of this thesis. We published parts of this chapter in [SBL+19]. First, preliminaries on non-binary LDPC are reviewed, and the differences to binary LDPC codes are highlighted. Afterwards, the design of information bottleneck decoders for non-binary LDPC codes is presented. It will be shown that different design objectives can be pursued based on the available memory complexity at the receiver. In more detail, the chapter contains the following contributions:

- A so-called *channel combiner* is devised which allows to fuse the relevant information spread across multiple channel uses.
- Relevant-information-preserving variable and check node operations using the information bottleneck method resulting in an information bottleneck decoder for non-binary LDPC codes are presented.
- A fully symbol-metric decoding receiver is presented, which pairs higher-order modulation schemes and the respectice non-binary LDPC decoder.

6.1 Preliminaries on Non-Binary Low-Density Parity-Check Codes

Closely related to binary low-density parity-check codes, non-binary LDPC codes are defined using a sparse parity-check matrix $\mathbf{H}$ with dimension $N_\mathrm{c} \times N_\mathrm{v}$. However, in contrast to binary LDPC codes, the entries h_k of the parity check matrix are not only zero and one but can be any field element from $\mathrm{GF}(q)$. Still, each row of $\mathbf{H}$ represents a parity-check equation. Such an equation has the form

$$\sum_{k=0}^{d_\mathrm{c}-1} \underbrace{h_k c_k}_{c'_k} = 0, \tag{6.1}$$

where c_k are the corresponding codeword symbols and d_c denotes the node degree. As c_k and h_k are elements from $\mathrm{GF}(q)$, the addition and multiplication according to the respective field arithmetic have to be applied. The arithmetic in Galois fields was discussed in detail in Section 2.4.1. Thus, in contrast to the binary case, the check node operations do *not* equal a mod 2 sum but require addition *and* multiplications. In this chapter, we restrict ourselves to extension fields, i.e., $\mathrm{GF}(q) = \mathrm{GF}(2^m)$. A Tanner graph similar to Fig. 5.2 is depicted in Fig. 6.1 for the non-binary regime, where crossed circles illustrate the additional multiplications by weights h_k.

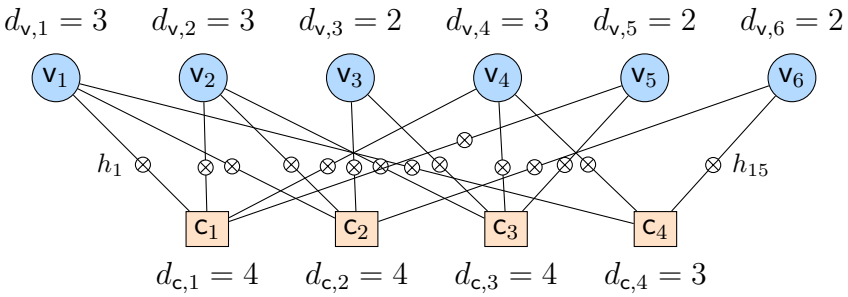


Figure 6.1: Tanner graph for an non-binary LDPC code.

6.1.1 Sum-Product Decoding of Non-Binary LDPC Codes

Sum-product decoding relies on the exchange of soft-information. In non-binary decoding, this soft-information can be represented either in the probability domain or in a corresponding log-domain [WSM04]. In the probability domain, the exchanged messages equal probability mass functions and are represented as probability vectors

$$\mathbf{m}_{\mathsf{c}_i \to \mathsf{v}_j}(c_j) = \left[\Pr(\mathsf{C}=0), \Pr(\mathsf{C}=1), \Pr(\mathsf{C}=\alpha), \ldots, \Pr(\mathsf{C}=\alpha^{q-2})\right] . \tag{6.2}$$

An alternative representation is the log-likelihood vector (LLR-vector)

$$\mathbf{L}_{\mathsf{c}_i \to \mathsf{v}_j} = \left[\log \frac{\Pr(\mathsf{C}=1)}{\Pr(\mathsf{C}=0)}, \log \frac{\Pr(\mathsf{C}=\alpha)}{\Pr(\mathsf{C}=0)}, \ldots, \log \frac{\Pr(\mathsf{C}=\alpha^{q-2})}{\Pr(\mathsf{C}=0)}\right] \tag{6.3}$$

where all entries are normalized with respect to $\Pr(\mathsf{C}=0)$. Hence, the dimension of the LLR-vector is $q-1$. Furthermore, please allow the slight abuse of notation as $\mathbf{L}_{\mathsf{c}_i \to \mathsf{v}_j}$ denotes a vector and not a matrix.

6.1.1.1 Non-Binary Check Node Operations in Sum-Product Decoding

In sum-product decoding of non-binary LDPC codes, the symbol probabilities are passed to the check nodes either in the probability domain or the log-domain. Each check node performs three tasks according to its parity-check equation (cf. Eq. (6.1)). This is exemplarily depicted in Fig. 6.2 for a $d_\mathrm{c} = 5$ check node.

1) Multiplication by Edge Weights $c'_k = h_k c_k$

First, the incoming probability vectors for the incoming symbols c_k are transformed into the probability vectors for the products c'_k incorporating the appropriate edge weight h_k. According to the multiplication rules described in Section 2.4.1, this corresponds to a cyclic shift of the last $2^m - 1$ entries in the probability vectors [CJ08].

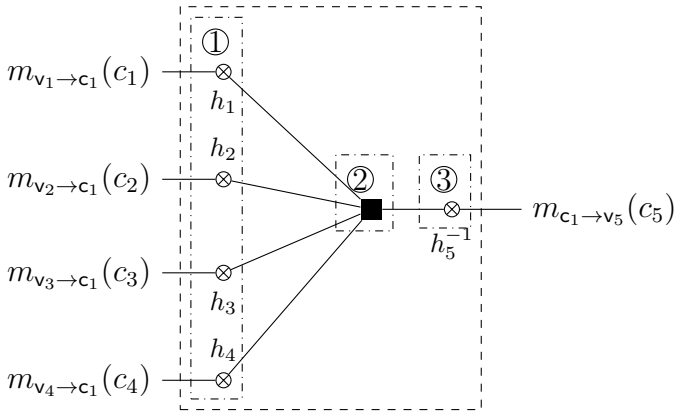


Figure 6.2: Illustration of a non-binary check node operation for $d_\mathrm{c} = 5$.

2) Summation

Once all entries of $p(c'_k)$ are obtained, the check node computes the convolution of $d_\mathrm{c} - 1$ probability vectors $p(c'_k)$ to account for the summation of the involved c'_k in $c'_i = \sum_{k \neq i} c'_k$ which follows from Eq. (6.1). This convolution is usually implemented as a fast convolution using FHT, resulting in the complexity $\mathcal{O}(d_\mathrm{c} 2^m \log_2 2^m)$.

3) Multiplication by Inverse Edge Weights $c_i = h_i^{-1} c'_i$

In the last step of the check node update, the outgoing message passed to a connected variable node is again found by a cyclic shift of the last $2^m - 1$ entries of $p(c'_i)$ according to the inverse edge weight h_i^{-1}.

Log-Max Approximation The main computational burden of the decoding of non-binary LDPC codes is the required convolution of probability distributions at the check nodes. As described above, one possible reduced-complexity implementation in the probability domain is the fast Walsh-Hadamard transform (FHT). Another alternative is the log-max decoder proposed in [WSM04], which can be seen as a generalization of the min-sum decoder. This idea was pursued further in [DF05] and proposed as *extended min-sum decoder*. Let us consider the two messages $\mathbf{L}_{\mathsf{v}_k\to\mathsf{c}_i}$ and $\mathbf{L}_{\mathsf{v}_{k'}\to\mathsf{c}_i}$ with weights h_k and $h_{k'}$.

In the log-domain, each vector entry m of outgoing message $\mathbf{L}_{\mathsf{c}_i\to\mathsf{v}_j,m}$ is found as

$$\mathbf{L}_{\mathsf{c}_i\to\mathsf{v}_j,m} = \log\Bigg(e^{\mathbf{L}_{\mathsf{v}_k\to\mathsf{c}_i,h_k^{-1}\alpha_m}} + e^{\mathbf{L}_{\mathsf{v}_{k'}\to\mathsf{c}_i,h_{k'}^{-1}\alpha_m}} + \sum_{x\in GF(q)\setminus\{h_k^{-1}\alpha_m\}} e^{\mathbf{L}_{\mathsf{v}_k\to\mathsf{c}_i,x}+\mathbf{L}_{\mathsf{v}_{k'}\to\mathsf{c}_i,h_{k'}^{-1}(\alpha_m+xh_k)}} \Bigg)$$

$$- \log \left(1 + \sum_{x \in GF(q)} e^{\mathbf{L}_{\mathsf{v}_k \to \mathsf{c}_i, x} + \mathbf{L}_{\mathsf{v}_{k'} \to \mathsf{c}_i, h_{k'}^{-1} h_k x}} \right) . \tag{6.4}$$

The log-max decoder simplifies this expression applying the Jacobian logarithm from Eq. (5.24) as [WSM04]

$$\begin{aligned} \mathbf{L}_{\mathsf{c}_i \to \mathsf{v}_j, m} = &\max \left(\mathbf{L}_{\mathsf{v}_k \to \mathsf{c}_i, h_k^{-1} \alpha_m}, \max \left(\mathbf{L}_{\mathsf{v}_{k'} \to \mathsf{c}_i, h_{k'}^{-1} \alpha_m}, \max \left(\ldots \right) \right) \right) \\ &- \max \left(0, \max \left(\mathbf{L}_{\mathsf{v}_k \to \mathsf{c}_i, x} + \mathbf{L}_{\mathsf{v}_{k'} \to \mathsf{c}_i, h_{k'}^{-1} h_k x}, \max \left(\ldots \right) \right) \right) , \end{aligned} \tag{6.5}$$

which requires recursion, resulting in a complexity of $\mathcal{O}(q^2 d_c)$. However, in the log-max decoders, only simple max-operations and no multiplications are required. In addition, the extended min-sum decoder [DF05] neglects the least reliable $q - l$ entries in the LLR-vector. This allows reducing the complexity to $\mathcal{O}(q l d_c)$.

6.1.1.2 Non-Binary Variable Node Operations in Sum-Product Decoding

In sum-product decoding of non-binary LDPC codes, each variable node receives d_v probability vectors from its connected check nodes, where d_v is the degree of the variable node. To generate extrinsic information that is passed back to the check nodes during decoding, $d_\mathrm{v} - 1$ messages from the check nodes and the channel message (cf. Eq. (6.6)) are multiplied element-wise followed by a normalization to ensure that the exchanged message is a valid probability mass function. This results from the equality constraint of a variable node, i.e., all incoming messages are probability vectors for the same codeword symbol. In the log-domain, the variable node update equals an element-wise addition of the LLR-vectors.

6.1.2 The Channel Combiner for Higher-Order Galois Fields

Naturally, non-binary LDPC codes are paired with higher-order modulation schemes. In these scenarios, the information bottleneck channel output quantizer from Example 4.1 can be employed as the first building block in a coarsely-quantized, mutual-information-based, non-binary LDPC decoder.

However, in the literature, it is more common to resort to BPSK transmission of the codeword symbols c_k [CJ08]. Hence, in the first part of this chapter, we consider a non-binary LDPC encoded transmission over a quantized output, symmetric additive white Gaussian noise (AWGN) channel with binary phase shift keying modulation (BPSK). In the applied scheme, m BPSK symbols are transmitted for each

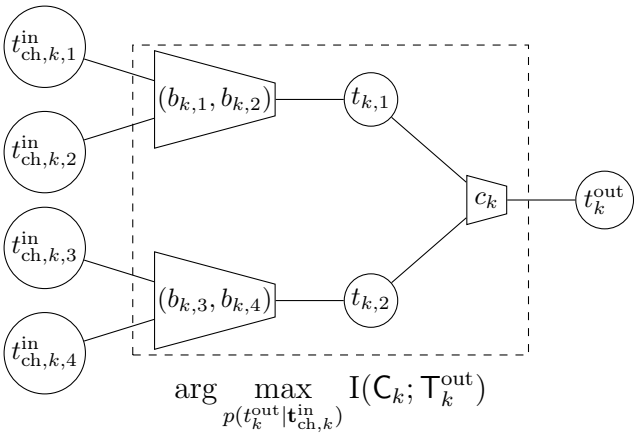


Figure 6.3: Information bottleneck graph of lookup table $p(t^{\text{out}}_k|\mathbf{t}^{\text{in}}_{\text{ch},k})$.

codeword symbol c_k. At the receiver, the received signal is first quantized. The quantizer delivers m outputs $\mathbf{t}^{\text{in}}_{\text{ch},k} = [t^{\text{in}}_{\text{ch},k,1}, t^{\text{in}}_{\text{ch},k,2}, \ldots, t^{\text{in}}_{\text{ch},k,m}]^{\mathsf{T}}$ for each codeword symbol c_k. The bit width of the applied quantizer is denoted w, such that the outputs $t^{\text{in}}_{\text{ch},k,j}$ are from the alphabet $\{1, 2, \ldots, 2^w = |\mathcal{T}_{\text{ch}}|\}$.

The first step in the conventional sum-product decoder of non-binary LDPC codes is the calculation of the symbol probabilities

$$p(c_k|\mathbf{t}^{\text{in}}_{\text{ch},k}) = \frac{p(c_k)}{p(\mathbf{t}^{\text{in}}_{\text{ch},k})} \prod_{j=1}^{m} p(t^{\text{in}}_{\text{ch},k,j}|b_{k,j}), \tag{6.6}$$

where $b_{k,j}$ denotes the bits in the binary representation of c_k. For each symbol, this corresponds to a probability vector which is used as channel knowledge for sum-product decoding.

In contrast, the proposed information bottleneck decoder does not use any probability vector but processes a single quantization index $t^{\text{out}}_k \in \{1, \ldots, |\mathcal{T}^{\text{out}}|\}$ instead. Intuitively, this quantization index should be highly informative about c_k. Such an index t^{out}_k can be obtained from $\mathbf{t}^{\text{in}}_{\text{ch},k}$ using a mutual-information-maximizing mapping $p(t^{\text{out}}_k|\mathbf{t}^{\text{in}}_k)$, which is constructed with the information bottleneck method. The required joint distribution $p(c_k, \mathbf{t}^{\text{in}}_{\text{ch},k})$ to construct the table follows directly from Eq. (6.6). As a by-product, we obtain $p(c_k, t^{\text{out}}_k)$, which will be used for the construction of subsequent lookup tables. The size of the lookup table can be reduced by using a decomposition into two-input lookup tables, as exemplified in Fig. 6.3 for $m = 4$ inputs. As this unit fuses the information spread across several BPSK symbols, we call this unit *channel combiner*.

Interestingly, we observe that in contrast to previous instances of mutual-information-bottleneck units, the inputs to the channel combiner do not carry redundant information about the same relevant random variable, which can be compressed. Instead, the channel combiner gathers independent partial information. Thus, the original mutual information $\mathrm{I}(\mathsf{C}_k; \mathsf{T}^{\mathrm{in}}_{\mathrm{ch},k})$ yields

$$\mathrm{I}(\mathsf{C}_k; \mathsf{T}^{\mathrm{in}}_{\mathrm{ch},k}) = \sum_m \mathrm{I}(\mathsf{B}_{k,m}; \mathsf{T}^{\mathrm{in}}_{\mathrm{ch},k,m}) \tag{6.7}$$

by application of the chain rule of mutual information. Table 6.1 summarizes the preserved relevant information for different intermediate cardinalities $|\mathcal{T}|$, channel output cardinalities $|\mathcal{T}_{ch}|$ and output cardinalities $|\mathcal{T}^{\mathrm{out}}|$ for $c_k \in \mathrm{GF}(16)$ as depicted in Fig. 6.3 and also for $c_k \in \mathrm{GF}(4)$ and $c_k \in \mathrm{GF}(8)$. It can be observed that due to the independent partial information spread across the individually transmitted bits, the channel combiner looses a substantial fraction of relevant information if $\log_2(|\mathcal{T}^{\mathrm{out}}|) \ll q \log_2(|\mathcal{T}_{ch}|)$, especially for large field orders. Thus, it is more common in the literature to measure the quantization of the soft information in non-binary LDPC decoding in the pseudo-unit $\frac{\mathrm{bits}}{\mathrm{field\ element}} = \frac{\mathrm{bits}}{\mathrm{FE}}$ [WSM04]. Hence, the overall number of bits needed to represent an exchanged message depends largely on the field order. Especially for large field orders like GF(256), an extremely coarse quantization of 1 $\frac{\mathrm{bit}}{\mathrm{FE}}$ results in 256 bits per message. Furthermore, as all information bottleneck algorithms presented so far require access to the entire joint distributions, in this thesis, only LDPC codes over GF(4) and GF(16) are considered in the next sections to maintain a manageable memory and implementation complexity.

6.2 Decoding Non-Binary LDPC Codes Using the Information Bottleneck Method

In the previous chapter on binary LDPC codes, the relevant variable X for a check node is the modulo 2 sum of the bits connected to the check node. Thus, the mutual-information-maximizing lookup table serves as an integer-based replacement for the well-known box-plus operation for log-likelihood ratios. This section presents all the required steps to generalize the information bottleneck decoder construction from binary to non-binary LDPC codes. This generalization is not straightforward, and the challenges are versatile. The main reason is the much more sophisticated arithmetic in higher-order Galois fields.

In this section, it is described how a lookup table based decoder for non-binary LDPC codes is built. In particular, the relevant-information-preserving mappings replacing

Table 6.1: Selection of different channel output cardinalities, channel combiner cardinalities for various Galois fields and the associated ratio of relevant and original mutual information.

GF(q)	$\lvert\mathcal{T}_{ch}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{T}^{\text{out}}\rvert$	$\frac{\text{bits}}{\text{FE}}$	$\frac{\mathrm{I}(\mathsf{C}_k;\mathsf{T}_k^{\text{out}})}{\mathrm{I}(\mathsf{C}_k;\mathsf{T}_{\text{ch},k}^{\text{in}})}$
GF(4)	8	-	8	0.75	0.91
GF(4)	8	-	32	1.25	0.98
GF(4)	16	-	16	1.0	0.96
GF(4)	16	-	64	1.5	0.99
GF(8)	8	16	64	0.75	0.96
GF(8)	8	32	128	0.875	0.98
GF(8)	16	32	128	0.875	0.97
GF(8)	16	64	256	1.0	0.99
GF(16)	8	16	64	0.375	0.90
GF(16)	8	128	512	0.5625	0.97
GF(16)	16	32	128	0.4375	0.94
GF(16)	16	64	1024	0.625	0.98

check and variable node operations in the sum-product algorithm are devised.

6.2.1 Non-Binary Check Node Operations from the Information Bottleneck Method

Similar to Section 6.1.1.1, the non-binary check node operation in a respective information bottleneck decoder is comprised of three steps, corresponding to multiplication, summation, and inverse multiplication. As in the binary case (cf. Chapter 5), the information bottleneck method requires access to the correct joint distributions capturing the stochastic input-output relations.

This section aims to replace all of the operations from Section 6.1.1.1 with mutual-information-maximizing lookup tables to tackle the impractically high implementation complexity of conventional non-binary LDPC decoder. Furthermore, the exchanged soft-information shall be represented using a single scalar cluster index instead of a double-precision probability vector or LLR vector. The entire workflow of the check node design with the information bottleneck method is exemplified in Fig. 6.4 for a degree $d_{\text{c}} = 5$ check node. This check node processes $d_{\text{c}} - 1 = 4$ incoming quantization indices t_k^{in} to determine one outgoing quantization index t_1^{out} which is passed back to the variable node replacing the probability vector $p(c_1)$ in the sum-product algorithm. Please note that the message t_1^{in} for c_1 is excluded since extrinsic information on c_1 shall be generated. Message generation has to be carried out using an equivalent structure for all other c_j.

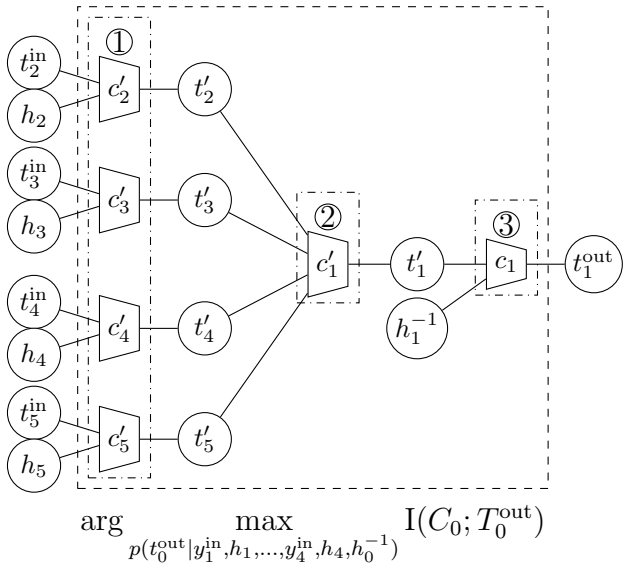


Figure 6.4: Information bottleneck graph of the relevant-information-preserving mapping $p(t_1^{\text{out}}|t_2^{\text{in}},h_2,\ldots,t_5^{\text{in}},h_5,h_1^{-1})$ for $d_{\text{c}}=5$.

We provide a step-by-step derivation of the joint distributions required as inputs for the information bottleneck algorithms to generate the respective mutual-information-maximizing mappings.

1) Multiplication by Edge Weights $c_k' = h_k c_k$

In Fig. 6.4, the multiplication equivalent mapping is depicted in the box labeled ①. Obviously, since all incoming quantization indices t_k^{in} are just unsigned integers, no shift of any probability vector is possible. However, this is not required since we are only interested in preserving the information on the relevant random variable C_k' given the input tuple (h_k, t_k^{in}). Therefore, we need to determine the joint distribution $p(c_k', h_k, t_k^{\text{in}})$ to design a mutual-information-maximizing mapping with the information bottleneck method. According to the general chain rule of probabilities and given the independence of t_k^{in} and h_k,

$$p(c_k', h_k, t_k^{\text{in}}) = \sum_{c_k \in \mathrm{GF}(2^m)} p(c_k'|h_k, c_k) p(c_k, t_k^{\text{in}}) p(h_k)\,. \tag{6.8}$$

In Eq. (6.8), $p(c_k'|h_k,c_k)$ reflects the multiplication arithmetic $c_k' = h_k c_k$ in $\mathrm{GF}(2^m)$. Mathematically, $p(c_k'|h_k,c_k) = \delta(c_k' + h_k c_k)$, i.e., it is 1 if $c_k' = h_k c_k$ and 0 otherwise. In the first decoding iteration, $p(c_k, t_k^{\text{in}})$ is given by $p(c_k, t_{\text{ch},k}^{\text{in}})$ with $t_k^{\text{in}} = t_{\text{ch},k}^{\text{in}}$ since all incoming t_k^{in} are obtained directly from the channel combiner (cf. Section 6.1.2). Feeding the joint distribution from Eq. (6.8) to an information bottleneck algorithm with output cardinality $|\mathcal{T}_{\text{mult}}|$ delivers the clustering $p(t_k'|h_k, t_k^{\text{in}})$,

where $t'_k \in \{1, 2, \ldots, |\mathcal{T}_{\text{mult}}|\}$ and $\mathrm{I}(\mathsf{C}'_k; \mathsf{T}'_k) \to \max$ for the given cardinality $|\mathcal{T}_{\text{mult}}|$.

2) Summation

To account for the summation $c'_1 = \sum_{k \neq 1} c'_k$, in Fig. 6.4, the convolution equivalent lookup table is depicted in the box labeled ②. Again since only unsigned integers t'_k are processed instead of probability vectors, a new t'_1 given $(t'_2, t'_3 \ldots, t'_{d_c})$ has to be generated which is highly informative about $c'_1 = \sum_{k \neq 1} c'_k$. Therefore, the joint distribution $p(c'_1, t'_2, t'_3 \ldots, t'_{d_c})$ is required. Similarly as in (6.8) one finds

$$p(c'_1, t'_2, t'_3, \ldots, t'_{d_c}) = \sum_{c'_2, c'_3, \ldots, c'_{\mathrm{d_c}}} p(c'_1 | c'_2, c'_3, \ldots, c'_{d_c}) \prod_{k=2}^{d_c} p(t'_k, c'_k)\,. \tag{6.9}$$

In Eq. (6.9), $p(c'_1 | c'_2, c'_3, \ldots, c'_{d_c})$ reflects the sum arithmetic $c'_1 = \sum_{k \neq 1} c'_k$ in $\mathrm{GF}(2^m)$. Mathematically, $p(c'_1 | c'_2, c'_3, \ldots, c'_{d_c}) = \delta(c'_1 + \sum_{k \neq 1} c'_k)$. Feeding the joint distribution (Eq. (6.9)) to an information bottleneck algorithm with output cardinality $|\mathcal{T}_{\text{conv}}|$ delivers a mapping $p(t'_1 | t'_2, t'_3, \ldots, t'_{\mathrm{d_c}})$, where $t'_1 \in \{1, 2, \ldots, |\mathcal{T}_{\text{conv}}|\}$ and $\mathrm{I}(\mathsf{C}'_1; \mathsf{T}'_1) \to \max$ for the given cardinality $|\mathcal{T}_{\text{conv}}|$.

Similar as in Section 5.2.2.3, we note that a two-input decomposition of lookup tables can be applied to reduce the size of the lookup table $p(t'_1 | t'_2, t'_3, \ldots, t'_{\mathrm{d_c}})$.

3) Multiplication by Inverse Edge Weights $c_1 = h_1^{-1} c'_1$

The multiplication equivalent by the inverse edge label h_1^{-1} is also implemented as a mutual-information-maximizing mapping $p(t_1^{\text{out}} | h_1^{-1}, t'_1)$ and depicted in the box labeled ③ in Fig. 6.4. The joint distribution $p(c_1, h_1^{-1}, t'_1)$ for designing the Finvolved lookup table can be obtained equivalently as explained for the multiplication equivalent by h_1 using Eq. (6.8). The final output $t_1^{\text{out}} \in \{1, 2, \ldots, |\mathcal{T}_{\text{prod}}|\}$ is passed to a connected variable node.

6.2.2 Non-Binary Variable Node Operations from the Information Bottleneck Method

In the following, we consider an arbitrary node that belongs to a codeword symbol c. Here, again the idea is to replace the described variable node operation with a relevant-information-preserving mapping. This mapping is depicted in Fig. 6.5 and it processes $d_{\text{v}} - 1$ incoming quantization indices t_k^{in} received from the check nodes and a channel index $t_{\text{ch}}^{\text{in}}$ from the channel output quantizer to determine one outgoing quantization index t_1^{out} which is passed back to a check node. Please note that the

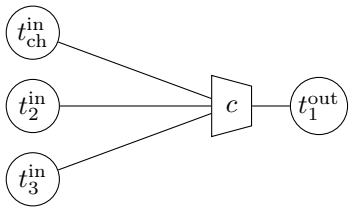


Figure 6.5: Information bottleneck graph of lookup table $p(t_1^{\text{out}}|t_{\text{ch}}^{\text{in}}, t_2^{\text{in}}, t_3^{\text{in}}, \ldots, t_{d_{\text{v}}}^{\text{in}})$ for $d_{\text{v}} = 3$.

message t_1^{in} is excluded at the input on the left since extrinsic information shall be generated. Message generation has to be carried out with the same structure for all other connected edges. The joint input distribution to design the depicted mapping in Fig. 6.5 is given by

$$p(c, t_{\text{ch}}^{\text{in}}, t_2^{\text{in}}, t_3^{\text{in}}, \ldots, t_{d_{\text{v}}}^{\text{in}}) = p(c)p(t_{\text{ch}}^{\text{in}}|c) \prod_{l=2}^{d_{\text{v}}} p(t_l^{\text{in}}|c). \tag{6.10}$$

This joint distribution reflects the aforementioned equality constraint of the variable node. Feeding the joint distribution from Eq. (6.10) to an information bottleneck algorithm with output cardinality $|\mathcal{T}_{\text{var}}|$ yields $p(t_1^{\text{out}}|t_{\text{ch}}^{\text{in}}, t_2^{\text{in}}, t_3^{\text{in}}, \ldots, t_{d_{\text{v}}}^{\text{in}})$, where $t_1^{\text{out}} \in \{1, 2, \ldots, |\mathcal{T}_{\text{var}}|\}$ and $\mathrm{I}(\mathsf{C}; \mathsf{T}_1^{\text{out}}) \rightarrow \max$. The unsigned integer t_1^{out} is passed back to the connected check node on the target edge in the next decoding iteration.

Finally, one notes that a two-input decomposition of lookup tables can be applied to reduce the variable node lookup table size.

6.2.3 Discrete Density Evolution for Non-Binary Codes and Fixed Lookup Tables

It is important to remember that the distributions of the exchanged messages evolve over the iterations. Therefore, to cope with this evolution, it is appropriate to design updated lookup tables for each decoding iteration using the appropriate distributions. These joint distributions correspond to the by-products $p(c, t^{\text{out}})$ of the applied information bottleneck algorithm. Using these output distributions as inputs of the next applied information bottleneck to construct lookup tables, we inherently track the evolution of these joint input distributions. This is entirely analogous to the discrete density evolution scheme for binary LDPC codes described in Chapter 5. As an interesting consequence, the decoding performance for a considered regular ensemble under the proposed lookup table based decoding scheme can be investigated. We note that performing efficient density evolution for non-binary LDPC codes is an open problem that is inherently tackled by the proposed lookup table

construction scheme. However, further investigations of this interesting finding and implications on the code design of non-binary LDPC codes are beyond the scope of this thesis.

Finally, all involved lookup tables are constructed just once for a fixed design-E_b/N_0. The created lookup tables are then stored and applied for all E_b/N_0. Hence, the lookup table construction has to be done only once and offline.

6.3 Simulation Results and Evaluation

In this section, results from bit error rate simulations, respectively symbol error rate simulations, are presented and discussed for two different settings involving two different non-binary LDPC codes, i.e.,

1. Setting 1: BPSK transmission combined with the respective channel combiner from Section 6.1.2 is paired with a non-binary LDPC code over Galois field GF(4). The code was taken from [Mac20] and has length $N_\mathrm{v} = 816$, code rate $R_\mathrm{c} = 0.5$, variable node degree $d_\mathrm{v} = 3$ and check node degree $d_\mathrm{c} = 6$ and identifier 816.3.174.

2. Setting 2: 16 QAM transmission combined with the respective information bottleneck channel output quantizer from Example 4.1. Here, the non-binary code is an ultra-sparse LDPC code with an optimized girth, $N = 38$, $R_\mathrm{c} = 0.5$, variable node degree $d_\mathrm{v} = 2$ and check node degree $d_\mathrm{c} = 4$ over GF(16) proposed in [VDP08].

Setting 1: BPSK Transmision and Non-Binary Code over GF(4)

The obtained bit error rates for sum-product decoding using FHT [DF07] with check node complexity $\mathcal{O}(d_c q(\log_2 q + d_c))$, log-max decoding [WSM04] with check node complexity $\mathcal{O}(d_c q^2)$, and the information bottleneck based decoding with check node complexity $\mathcal{O}(d_c)$ are depicted in Fig. 6.6. The channel quantizer described in Section 6.1.2 was used with an output cardinality $|\mathcal{T}_\mathrm{chan}| = 128$ corresponding to 7 bit quantization, i.e., 1.75 $\frac{\text{bits}}{\text{FE}}$. For the belief-propagation decoder and the log-max decoder, the symbol probabilities $p(c_k, t^{\text{in}}_{\text{ch},k})$ were used for decoding. In contrast, the information bottleneck decoder worked directly on the quantization indices $t^{\text{in}}_{\text{ch},k}$.

All decoders performed a maximum of $i_\mathrm{max} = 40$ iterations. The information bottleneck decoder was constructed for a design-E_b/N_0 of 1.5 dB, which was found by

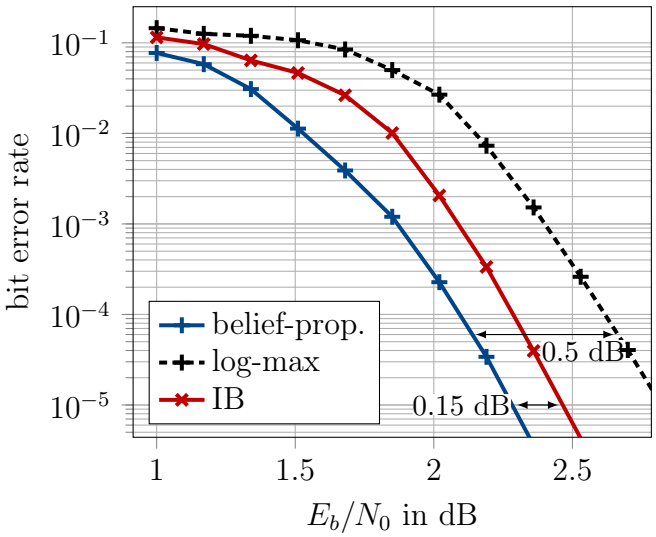


Figure 6.6: Bit error rate performance of our proposed decoder and reference systems with properties summarized in Table 6.2, Setting 1 and $i_{\max} = 40$.

a bisection search as optimum design parameter. The most important parameters of the applied decoders are summarized in Table 6.2 for a quick overview.

In the information bottleneck decoder, only integer-valued indices from the sets $\mathcal{T}_{\text{mult}}$ and $\mathcal{T}_{\text{var}}$ are used as messages instead of probability vectors. Thus, the cardinalities summarized in Table 6.3, correspond to a very coarse quantization of 2-2.25 bits per field element (cf. Table 6.2).

In Fig. 6.6, the belief-propagation algorithm serves as a benchmark with the best bit error rate performance, but at the same time, it has the highest computational complexity (cf. Table 6.2). Although all applied operations in the information bottleneck decoder are simple lookups, the decoder performs only 0.15 dB worse than the benchmark. Although the log-max decoder uses conventional arithmetic and double-precision message representation, it is clearly outperformed by the proposed information bottleneck decoder. In summary, the proposed decoder for non-binary codes achieves a similar result in terms of bit error rate performance as for the binary counterpart from the previous chapter. It shall be emphasized that the applied lookup tables for the non-binary case completely replace all arithmetical operations such as convolution of probability vectors and multiplication. The processing of probability vectors simplifies to lookups of scalar integers in pre-generated tables. For the considered GF(4) LDPC code, the amount of memory required to store the lookup tables is provided in Table 6.3. It can be seen that for the considered decoder, 215.00 kB are needed per iteration in a respective DSP implementation. The vast savings in computational complexity can justify this amount of memory required.

Table 6.2: Simulation parameters of decoder and reference system compared in Fig. 6.6 for Setting 1.

decoder	node operation (check / var)	precision exchanged messages	precision check node	precision variable node	channel quantizer
belief-propagation (belief-prop.)	FHT+ multiplication / multiplication	64 $\frac{\text{bits}}{\text{FE}}$	64 $\frac{\text{bits}}{\text{FE}}$	64 $\frac{\text{bits}}{\text{FE}}$	1.75 $\frac{\text{bits}}{\text{FE}}$
log-max [WSM04]	$\max^*()$/ addition	64 $\frac{\text{bits}}{\text{FE}}$	64 $\frac{\text{bits}}{\text{FE}}$	64 $\frac{\text{bits}}{\text{FE}}$	1.75 $\frac{\text{bits}}{\text{FE}}$
information bottleneck decoder (IB)	lookup table/ lookup table	2.25 $\frac{\text{bits}}{\text{FE}}$	2.25 $\frac{\text{bits}}{\text{FE}}$	2.25 $\frac{\text{bits}}{\text{FE}}$	1.75 $\frac{\text{bits}}{\text{FE}}$

Table 6.3: Total memory amount of lookup tables in the information bottleneck decoder per iteration.

lookup table	cardinality	table size
check node ①, ③	$\lvert\mathcal{T}_{\text{mult}}\rvert = 256 = 2\frac{\text{bits}}{\text{FE}}$	3.04 kB
check node ②	$\lvert\mathcal{T}_{\text{conv}}\rvert = 512 = 2.25\frac{\text{bits}}{\text{FE}}$	129.02 kB
variable node	$\lvert\mathcal{T}_{\text{var}}\rvert = 512 = 2.25\frac{\text{bits}}{\text{FE}}$	82.94 kB
total		215.00 kB

Setting 2: 16 QAM Transmision and Non-Binary Code over GF(16)

In the previous setting, the non-binary field elements were transmitted as BPSK symbols. However, in practice, non-binary LDPC codes are often coupled with higher-order modulation schemes. In such application, it is more appropriate to match the field order of the Galois field and the modulation order. In this second example, we consider non-binary codes over GF(16) and $i_{\max} = 10$. Furthermore, a 16 QAM modulation as discussed in Example 4.1 is considered. Here, we use an information bottleneck channel output quantizer with 4 bit per dimension, i.e., 8 bit resolution. In turn, $|\mathcal{T}_{\text{chan}}| = 256$ which equals a theoretical resolution of $0.5\frac{\text{bits}}{\text{FE}}$.

Again, let us consider a belief-propagation decoder and the log-max decoder as reference. The simulation parameters are summarized in Table 6.4. As discussed in Section 6.1.2, the required total number of bits required per message increases with the field order. For ease of implementation let us restrict the internal resolution

Table 6.4: Simulation parameters of decoder and reference system compared in Fig. 6.7 for Setting 1.

decoder	node operation (check / var)	precision exchanged messages	precision check node	precision variable node	channel quantizer
belief-propagation (belief-prop.)	FHT+ multiplication / multiplication	64 $\frac{\text{bits}}{\text{FE}}$	64 $\frac{\text{bits}}{\text{FE}}$	64 $\frac{\text{bits}}{\text{FE}}$	0.5 $\frac{\text{bits}}{\text{FE}}$
log-max [WSM04]	max*()/ addition	1 $\frac{\text{bits}}{\text{FE}}$	1 $\frac{\text{bits}}{\text{FE}}$	1 $\frac{\text{bits}}{\text{FE}}$	0.5 $\frac{\text{bits}}{\text{FE}}$
information bottleneck decoder (IB)	lookup table/ lookup table	0.5 $\frac{\text{bits}}{\text{FE}}$	0.5 $\frac{\text{bits}}{\text{FE}}$	0.5 $\frac{\text{bits}}{\text{FE}}$	0.5 $\frac{\text{bits}}{\text{FE}}$

of the information bottleneck quantizer to $|\mathcal{T}_{\text{mult}}| = |\mathcal{T}_{\text{var}}| = |\mathcal{T}_{\text{conv}}| = 256 = 8$ bit, i.e., again 0.5 $\frac{\text{bits}}{\text{FE}}$. Only since the information bottleneck decoder represents the entire vector by a single index, it is possible to achieve a quantization being a fraction of a bit per field element. Thus, the information bottleneck decoder learns a compression across the vector elements, which is an interesting observation. In a conventional decoder, this is not possible, as all elements have to be represented with at least 1 $\frac{\text{bit}}{\text{FE}}$. Thus, for a more appropriate comparison, we limit the internal resolution of the log-max decoder to exactly 1 $\frac{\text{bit}}{\text{FE}}$. The performance in terms of symbol error rate is depicted in Fig. 6.7. Despite the extremely coarse quantization of the information bottleneck decoder, the performance is only 1 dB worse than the double-precision belief propagation decoder, which exchanges entire probability vectors and performs complicated arithmetical operations in the nodes. Comparing the log-max decoder with 1 $\frac{\text{bit}}{\text{FE}}$ message resolution and the information bottleneck decoder with 0.5 $\frac{\text{bits}}{\text{FE}}$ message resolution is even more important. It can be observed that the log-max decoder loses 9 dB compared to the belief-propagation benchmark and 8 dB compared to the information bottleneck decoder. Thus, we concluded that the log-max decoder is impractical for such coarse quantization, whereas the information bottleneck decoder still works reasonably well.

Comparing Setting 1 and Setting 2 it can be observed that the loss in decoding performance of the coarsely quantized decoding schemes is noticeably larger. This is caused by two main factors. First, the overall quantization is much coarser quantization in Setting 2 compared so Setting 1. Second, the field order of the non-binary code is much higher in Setting 2 compared to Setting 1. Thus, the design of the

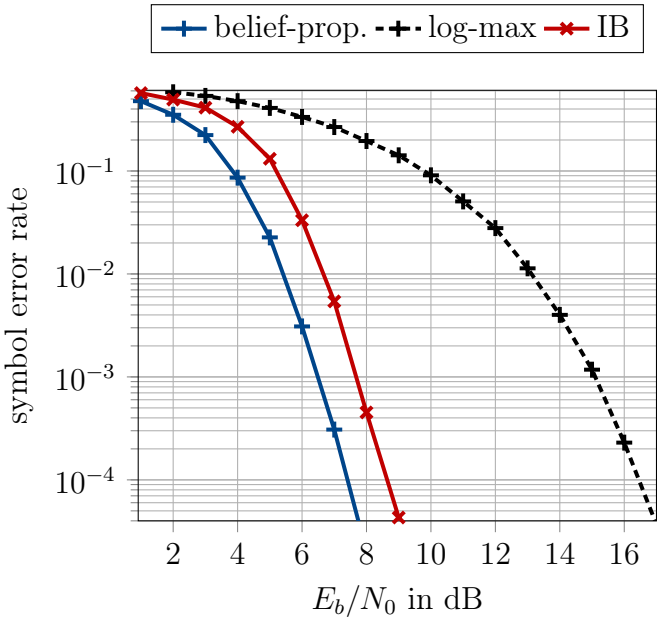


Figure 6.7: Bit error rate performance of our proposed decoder and reference systems with properties summarized in Table 6.4, Setting 2 and $i_{\max} = 10$.

mutual-information-maximizing node operations is more sensitive to a coarse quantization which yields a larger gap in decoding performance compared to a double-precision benchmark system.

6.4 Summary

Especially in massive machine-type communication, reliable and low-energy signal processing is required. Theoretically, non-binary LDPC codes are compelling channel coding schemes also for very short packages. However, the decoding complexity of state-of-the-art decoders for non-binary LDPC codes often prohibits practical applications.

This chapter leveraged the information bottleneck's ability to preserve relevant information to overcome the main computational burdens of non-binary LDPC decoding. Motivated by the results for binary LDPC codes presented in Chapter 5, a complete framework to design check node and variable node operations which replace all arithmetic operations using only lookups in non-binary LDPC decoders was presented. A step-by-step conversion of the conventional sum-product algorithm resulting in relevant-information-preserving mappings yields an information bottleneck decoder, which performs only 0.15 dB worse than the sum-product algorithm and outperforms the log-max algorithm for a BPSK transmission and GF(4). Furthermore,

we combined the higher-order modulation quantizer from Chapter 4 with an information bottleneck decoder for non-binary LDPC codes over GF(16). It was shown that the presented decoder is superior to state-of-the-art decoders with comparable quantization.

It was observed that the presented discrete density evolution scheme might be used for code construction of non-binary LDPC, which is an active field of research. However, as code construction is not the primary focus of this thesis, possible investigations are left for further research.

Chapter 7

Data-Driven Mutual-Information-Based Signal Processing

Recently, *deep learning* received considerable attention in academia but also in industry. The tremendous research interest in neural networks is mainly induced by an ongoing shift from classical, model-based machine learning towards data-driven machine learning. This shift is mainly caused by several observations in different disciplines where deriving analytical, closed-form models to capture realistic real-world processes is cumbersome or often even infeasible. Hence, prevalent, robust, and mathematically well-understood machine-learning techniques like support vector machines, Gaussian processes, or naive Bayes classifiers are more-and-more replaced by a very generic and universal function approximator called *neural network*. However, it shall be emphasized that besides the outstanding success in computer science, image processing, etc., the hope that neural networks can achieve stunning results without any domain knowledge or fundamental understanding of the functional relation to be modeled turned out to be unrealistic [OH17]. Instead, many applications of neural networks, especially in communications, showed that a solid domain knowledge is crucial to tweak the architecture of neural networks accordingly to yield an optimum performance [OH17; DCH+18; AH18; NBB16; SDW17].

In [SDW17; CAD+20], detailed examples are sketched where domain knowledge helps to turn known communication system designs into a neural network equivalent. A common approach is unfolding [BS19]. Like in an unrolled LDPC decoder implementation, an iterative algorithm is unfolded and turned into a sequential

structure. Here, the neural network replaces the state-of-the-art operations. Especially if the classical operations are derived based on mismatched models, which do not capture the reality well, the neural network might learn more appropriate operations. Further benefits and applications are discussed in great detail in [BS19].

This chapter leverages an information-theoretical perspective on neural networks as proposed in [Sim18]. Especially for communications engineering problems that solve a classification problem, it is possible to investigate the neural networks using well-known information-theoretical quantities like the mutual information or the Kullback-Leibler divergence. Interestingly, as pointed out in [TZ15], it is also possible to measure the mutual information inside the neural network. Besides, [TZ15] revealed close relations to the information bottleneck method. However, in [SBD+19], the promising findings from [TZ15] could not be confirmed, and there is an ongoing debate about the role of the information bottleneck method to interpret neural networks. Nonetheless, it turned out that the information bottleneck functional itself is a compelling and robust loss function in representation learning [AFD+19].

All previous chapters resort to the classical perspective on the information bottleneck method, which we, from now on, term a *model-based* perspective. The model-based perspective assumes that all distributions fed into the information bottleneck are traceable, nicely described, and can be modeled perfectly.

However, in some applications, the distribution might be unknown or changing over time. Alternatively, like for non-binary LDPC codes with large field orders, the distribution becomes impractically large and cannot be represented easily.

Thus, this chapter presents a *data-driven* perspective on the information bottleneck method. Here, the idea is to learn the distribution based on empirical training data and employ a trainable function, e.g., a neural network representing the mapping $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$. Despite this change in perspective, this chapter still aims to design signal processing units which maximize the relevant information, i.e.,

$$\max_{t^{\text{out}}=f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}})} \mathrm{I}(\mathsf{X};\mathsf{T})\,, \tag{7.1}$$

where $f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}})$ indicates that we resort to trainable functions $f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}})$ with trainable parameters $\boldsymbol{\theta}$.

In this chapter, two different scenarios are considered. First, the *neural information bottleneck decoder* is presented. This decoder tackles the problem of infeasible large message mappings that are hard to obtain using the model-based perspective in

information bottleneck decoders for non-binary codes with large field orders and binary LDPC codes with closed nodes and high node degrees.

The second part of this chapter considers so-called *capacity-achieving* end-to-end learning. Here, an entire communication chain, including transmitter and receiver, is trained to maximize the relevant information and achieve the channel capacity. Interestingly, this is in line with the information bottleneck objective. It is shown that a solid information-theoretic understanding is required to derive a meaningful loss function such that the relevant information is maximized. Different applications of end-to-end learning are presented. A so-called *autoencoder* will be derived, which learns capacity-achieving transmit symbol constellations for arbitrary channels using constellation and probabilistic shaping. Therefore, well-known loss functions in deep learning are interpreted from an information-theoretical perspective. This idea is applied to both *symbol-wise* and *bit-wise* learning. The latter reveals that the proposed information-theory-inspired learning yields capacity-achieving bit labelings, which are generally very difficult to find in practice.

We published parts of this chapter in [SAH19; CAD+20]. First, this chapter provides a brief introduction to the broad field of neural networks.

7.1 Preliminaries on Neural Networks

In general, the concept of neural networks is old. It was presented already in 1958 in [Ros58] and [Ros61]. Also, the training algorithms to optimize neural networks are known for quite some time [RHW86]. However, the actual breakthrough was observed in 2010 with the so-called ImageNet moment where a neural network classified real-world images with extremely high precision. This breakthrough was mainly driven by a vast increase in available computational resources in the last decades. To train a neural network, one needs a considerable amount of training data and sufficient computational power to compute the large number of gradients backpropagating through the network while training. It can be observed that the increase in computational power and storage capacity directly correlates with the proposed size and expressive power of neural network architectures. The training of neural networks itself is described in more detail in Section 7.1.1.

To introduce neural networks and data-driven machine learning more formally the definition from [Mit05] is recalled:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at

tasks in T, as measured by P, improves with experience E."

In other words, a neural network tries to approximate an *unknown* function $f(x)$ sufficiently close by a trainable function $f_{\boldsymbol{\theta}}(x)$ with trainable parameters $\boldsymbol{\theta}$. The quality of this approximation is evaluated by a performance measure termed *loss*, and should improve through experience, i.e., if more training data have been observed.

The Task

In general, many kinds of tasks exist. Examples range from regression, translation to density estimation. However, as many communication tasks fall into the category of so-called *classification* problems, this task will be considered in more detail.

In a classification problem the training samples $\mathbf{y} \in \mathbb{R}^n$, holding the input features, are mapped onto one out of k classes or onto the probability of the class $p(\mathsf{K} = k|\,\mathbf{y})$. Thus,

$$f:\ \mathbb{R}^n \mapsto \{1, \ldots, k\} \tag{7.2}$$

or

$$f:\ \mathbb{R}^n \mapsto [p(\mathsf{K}=1|\mathbf{y}), \ldots, p(\mathsf{K}=k|\mathbf{y})]^{\mathsf{T}} \tag{7.3}$$

for classification problems. Especially, the mapping onto probabilities will turn out to be very useful later when deriving an information-theoretical perspective on neural networks.

The Experience

The actual machine learning algorithm makes different experiences during the learning process. On the one hand, this is due to the observed, actual training data describing the input-output relation to be learned. On the other hand, experience also relates to the type of dataset used for learning. Broadly, one can distinguish between supervised learning and unsupervised learning.

Supervised Learning In a mathematical sense, supervised learning aims to estimate $p(x|y)$, where y is the observed sample and x is the target or label. Thus, the dataset $\mathcal{D}$ always contains the features (or observations) y *and* the label x, which are drawn i.i.d. from the joint distribution $p(x, y)$, i.e.,

$$\mathcal{D} = (x, y) \underset{i.i.d}{\sim} p(x, y)\,. \tag{7.4}$$

The term *supervised* implies that a teacher builds the dataset by assigning the correct labels to each possible observation. However, as we will see later, if one

has access to the true data generating distribution $p(x, y)$, the dataset can be generated without manual labeling. Nonetheless, in most applications, $p(x, y)$ is unknown, and thus, constructing a correctly labeled dataset is challenging. The most common neural network architecture to tackle supervised learning problems is a feed-forward neural network described in more detail in Section 7.1.2.

Unsupervised Learning In contrast to supervised learning, in unsupervised learning the dataset $\mathcal{D}$ only contains the features y, i.e.,

$$\mathcal{D} = y \underset{i.i.d}{\sim} p(y)\,. \tag{7.5}$$

In turn, the goal is to learn useful characteristics of the structure of the dataset. Often in deep learning, this equals estimating the feature distribution $p(y)$ implicitly or explicitly. A common field of application for unsupervised learning is dimensionality reduction and clustering, for example, using the K-means algorithm [Llo82]. Another example is the expectation-maximization algorithm [Mit05]. Furthermore, unsupervised learning can also be used to train neural networks to mimic $p(y)$. Recently, this was successfully shown using generative adversarial networks (GANs) [GPM+14].

In brief, unsupervised learning considers only observed samples and tries to find meaningful representations, for example, by clustering or feature extraction. Supervised learning instead involves observing the pair of observation and associated label. Nonetheless, supervised and unsupervised learning are not completely distinct concepts. Especially, when targeting so-called *representation learning* a proper categorization is difficult. Representation learning is a very modern field within machine learning and covers a wider range of problems as deep learning, including so-called *autoencoders* [GBC16].

In literature, the information bottleneck method is often termed an unsupervised learning framework as it is formally a clustering tool. Whereas rate-distortion clearly falls into the category of unsupervised learning as usually only $p(y)$ is considered, the information bottleneck is more closely related to representation learning. Thus an explicit connection to supervised and/or unsupervised learning is difficult as the information bottleneck method requires access to $p(x, y)$.

7.1.1 Loss Functions and Training a Neural Network

To evaluate the performance of a neural network, an appropriate performance measure according to the task has to be chosen. One distinguishes between *point estimates* whereby one aims to obtain a specific value $\hat{x} = f_\theta(y)$ and *distributional estimates* trying to learn the entire probability mass function $p(x|y)$. In the latter case, the neuronal network $f_\theta(y)$ can be interpreted as a trainable approximation of $p_{\boldsymbol{\theta}}(x|y) \approx p(x|y)$.

If a detector shall provide soft-information for a subsequent soft-input channel decoder, the symbol error rate alone is not the best performance criterion. Thus, instead of optimizing a point estimate, the optimization is over distribution the $\hat{p}(x|y)$ itself. As discussed in 2.3.2, a useful divergence measure to determine the similarity of distributions is the Kullback-Leibler divergence $D_{\mathrm{KL}}\{p_{\boldsymbol{\theta}}(x|y)||p(x|y)\}$. It is shown in [Sim18] that this expression can be related to the so-called *cross-entropy loss*

$$H(x|y) = \mathbb{E}_{(x,y)\sim p(x,y)}\left[-\log(p_{\boldsymbol{\theta}}(x|y))\right] \tag{7.6}$$

that is of integral importance for training neural networks which solve a classification task.

Considering the standard formulation of supervised learning, it is assumed that instead of having access to the *true* data generating distribution $p(x, y)$ directly, one only has access to a set of samples $(x_{(i)}, y_{(i)})$, $i = 1, \ldots, N$ i.i.d. drawn from $p(x, y)$. Usually, these samples are said to form the training set ($\mathbf{x}^{\text{train}} = [x_{(1)}^{\text{train}}, x_{(2)}^{\text{train}}, \ldots]^\mathsf{T}$, $\mathbf{y}^{\text{train}} = [y_{(1)}^{\text{train}}, y_{(2)}^{\text{train}}, \ldots]^\mathsf{T}$) and test set ($\mathbf{x}^{\text{test}} = [x_{(1)}^{\text{test}}, x_{(2)}^{\text{test}}, \ldots]^\mathsf{T}$, $\mathbf{y}^{\text{test}} = [y_{(1)}^{\text{test}}, y_{(2)}^{\text{test}}, \ldots]^\mathsf{T}$). Depending on the application, this strict distinction might not be required.

Offline Learning Offline learning uses a static data set with a finite number of samples. This can lead to underfitting or overfitting as the network might learn only to reproduce precisely the trained data but does not generalize well to unseen data [Sim18]. Thus, the data set is split into samples used for training, i.e., the training set, and samples used for testing, i.e., the test set. In general, it is also preferable to have a third set, i.e., the validation set [GBC16].

Online Learning Online learning does not use a static data set. Instead, it has direct access to the unknown, data generating distribution $p(x, y)$ to obtain new samples processed in a streaming fashion continually. As the number of samples available is infinite, it is not needed to partition the samples into sets.

Instead, it is sufficient to generate samples during the training phase and then use newly generated samples for testing.

As the true distribution is unknown, the average loss $\mathbb{E}_{x\sim p(x|y)}\left[l(x, f_{\boldsymbol{\theta}}(y)\right]$ becomes the so-called *empirical risk* or *empirical loss* $J(\boldsymbol{\theta})$ of the training set [Sim18]

$$J(\boldsymbol{\theta}, \mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}}) = \frac{1}{N}\sum_{i=1}^{N} l(x_{(i)}^{\text{train}}, f_{\boldsymbol{\theta}}(y_{(i)}^{\text{train}}))\,, \tag{7.7}$$

where $l(.)$ denotes an arbitrary sample loss function.

Please note that all applications considered in this thesis allow online learning.

7.1.1.1 (Stochastic) Gradient Descent Optimization

As defined above, a neural network can be seen as a trainable function with trainable parameters $\boldsymbol{\theta}$, which minimizes a particular objection or loss function in the broader sense. The most common optimization technique to adjust the trainable parameters and thereby minimize the objective function is the *gradient descent algorithm* [GBC16]. As the optimization relies on training data, the empirical loss from Eq. (7.7) is commonly used as an objective. Thus, the task of the optimization algorithm is to determine

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}})\,. \tag{7.8}$$

Gradient descent is a relatively old and well-known first-order iterative optimization algorithm [Cau47] to find a local minimum of a differentiable function. Starting with a random initial guess $\boldsymbol{\theta}_0$, each iteration updates the solution by moving proportionally in the direction opposite to the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}})$, i.e.,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}_k} J(\boldsymbol{\theta}_k, \mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}}) \tag{7.9}$$

where $\epsilon > 0$ is the *learning rate* and k is the iteration index. The learning rate is a crucial hyper-parameter controlling the step size. On the one hand, the larger the step size, the faster the algorithm converges. On the other hand, choosing a too large step size prohibits to hit the local minimum as the solution is oscillating around the local minimum instead. Many adaptive algorithms were proposed to optimize the learning rate. The most common choice is the so-called Adam approach [KB14], which leverages an adaptive learning rate update and a momentum technique. A detailed discussion of algorithms to update the learning rate is beyond the scope of this chapter but can be found in great detail in [GBC16].

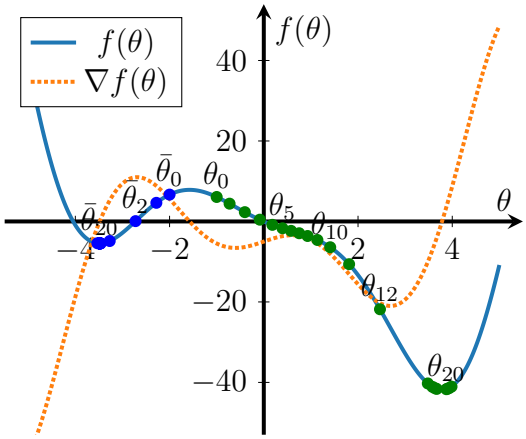


Figure 7.1: Illustration of the gradient descent algorithm to find the minimum of the function $f(x)$ for two different initial random starting points θ_0 and $\bar{\theta}_0$.

Equation (7.9) is repeated until a particular stopping criterion is met, e.g., if the loss is no longer significantly reduced between two iterations. This procedure is shown in Fig. 7.1 for an exemplary function $f(\theta)$ and two random initial starting points θ_0 and $\bar{\theta}_0$.

As shown in Fig. 7.1, gradient descent is very sensitive to the starting point and can easily end up in a bad local minimum. Thus, in the past, gradient descent was not considered as a suitable approach for nonconvex optimization problems [GBC16]. Nonetheless, neural networks where shown to be well trained using gradient descent. Although sometimes not even a local minimum is found, gradient descent achieves very low values of the loss function in a computationally efficient manner. One such efficient implementation is the back-propagation algorithm proposed in [RHW86].

As shown in Eq. (7.7), the actual loss is composed as the sum of the per-sample loss for all training samples. Clearly, the computational costs of this operation depends on the number of training samples N. Easily, a proper data set might contain billions of training data. Thus, already the computation of a single update step might be impractically long.

Hence, an extension of gradient descent called *stochastic gradient descent* is a more suitable algorithm to train neural networks with large data sets. In contrast to computing the actual gradient, stochastic gradient descent computes just an estimated gradient using a small subset of the training data. This subset is referred to as *minibatch* $\mathbb{B}$ [GBC16]. The minibatch is randomly drawn for each gradient step. In turn, using Eq. (7.7), the gradient is approximated as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}}) = \frac{1}{|\mathbb{B}|} \sum_{i \in \mathbb{B}} \nabla_{\boldsymbol{\theta}} l(x_{(i)}^{\text{train}}, f_{\boldsymbol{\theta}}(y_{(i)}^{\text{train}})) \tag{7.10}$$

where $|\mathbb{B}|$ is the size of the minibatch. According to [GBC16] the following general statements hold:

- The larger the minibatch size, the more accurate is the estimate of the gradient.
- The smaller the minibatch size, the larger is the variance of the estimate of the gradient. Thus, a smaller learning rate is required to ensure stable training. In turn, many iterations are needed, and the runtime is high. However, due to the high variance of the estimate, the entire parameter space can be explored faster and bad local optima can be avoided.

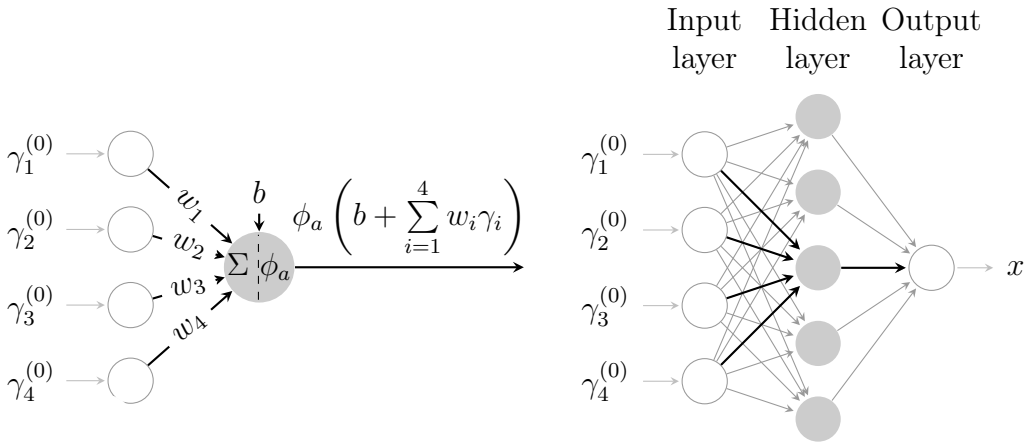
The minibatch size choice is an individual field of research which will not be discussed in detail here. For a more detailed discussion, the interested reader is referred to [GBC16].

Stochastic gradient descent optimizes the trainable parameters $\boldsymbol{\theta}$ of the neural network. However, to efficiently compute the gradients, the so-called *back-propagation algorithm* is used [RHW86; GBC16]. The back-propagation algorithm uses the chain rule of calculus to recursively calculate the partial derivatives of all trainable parameters [GBC16]. A more detailed discussion of the back-propagation algorithm can be found in [Bis09; GBC16; Sim18].

7.1.2 Feedforward Neural Networks (Multilayer Perceptrons)

Following the early ideas presented in [Ros58], the multilayer perceptron (MLP) also called *feed-forward neural network* is the most natural neural network architecture and maybe also the simplest class of neural networks in general. Formally, the feed forward neural network implements a mapping $f_\theta(\boldsymbol{\gamma}^0) : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ of an input vector $\boldsymbol{\gamma}^{(0)} \in \mathbb{R}^{N_0}$ to an output vector $\boldsymbol{\gamma}^{(L)} \in \mathbb{R}^{N_L}$, where L defines the *depth* of the neural network.

An MLP consists of an input layer, an output layer, and at least one hidden layer [GBC16]. Such an architecture is shown in Fig. 7.2b. The special case of a neural network with only one hidden layer is very useful to prove the universal approximation theorem of neural networks [GBC16].



(a) Single neuron (b) Feed-forward neural network

Figure 7.2: Illustration of a single neuron (a) which is embedded in a larger neural network with one hidden layer.

Each layer consists of several *neurons* [Ros58], sometimes termed *units* [GBC16]. In Fig. 7.2a, a single neuron is shown with four inputs, and the thick lines in Fig. 7.2b illustrate how the neuron from Fig. 7.2a is incorporated in the neural network.

Each neuron computes the weighted sum of its inputs plus the so-called *bias* followed by an *activation* function $\phi_a(\varkappa)$. Thus, the output of a single neuron is

$$\phi_a\left(b+\sum_{i=1}^{N} w_i\gamma_i\right), \tag{7.11}$$

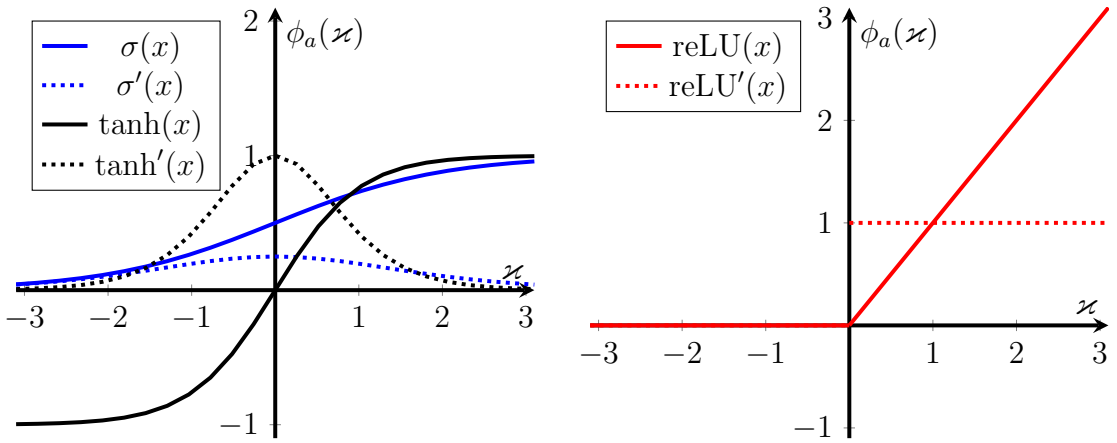
where w_i denotes the trainable weights and b is the trainable bias. The activation function of the input layer and the output layer is usually linear, i.e, $\phi_a(\varkappa)=\varkappa$. However, the activation function of the neurons in the hidden layers is generally non-linear, which is important for the expressive power of the neural network, as it allows to easily learn non-linear models. Common non-linear activation functions are [GBC16]:

Sigmoid function $\sigma(\varkappa)=\frac{1}{1+e^{-\varkappa}}$

Hyperbolic tangent $\tanh(\varkappa)=\frac{e^{\varkappa}-e^{-\varkappa}}{e^{\varkappa}+e^{-\varkappa}}$

Rectified linear unit (ReLU) $\mathrm{reLU}(\varkappa)=\max(0,\varkappa)$.

These activation functions and their first-order derivatives are shown in Fig. 7.3. The hyperbolic tangent and sigmoid function are highly non-linear. However, their first-order derivatives are close to zero for large input values. This results in the so-called *vanishing gradient problem*, which is unfavorable for gradient descent optimization techniques [GBC16]. Thus, despite its only small non-linearity, the rectified linear



(a) Sigmoid function and hyperbolic tangent

(b) Rectified linear unit (ReLU)

Figure 7.3: Common non-linear activation functions in neural networks.

unit (ReLU) activation turned out to be the most common activation function as it allows for better training using stochastic gradient descent. First, the ReLU activation function suffers less from the vanishing gradient problem as the gradient does not saturate for large positive inputs. Second, it implements a *sparse* activation as only positive inputs have a non-zero output. Third, the computation is very efficient compared to evaluating the hyperbolic tangent and the sigmoid function as only addition, multiplications, and comparisons are required.

It is often more convenient to use the vector notation to describe the output of an entire layer instead of a particular neuron. The MLP is called a feed-forward neural network as the information flows from one layer to the next layer [GBC16]. Hence, the function $f_\theta(\boldsymbol{\gamma}^{(0)})$ is implemented through L iterative processing steps

$$\boldsymbol{\gamma}^{(l)} = f_{\theta_l}^{(l)}(\boldsymbol{\gamma}^{l-1}), l = 1, \dots L\,. \tag{7.12}$$

Let $\boldsymbol{\gamma}^{(l)}$ denote the output of layer l. This output is comprised of the previous layer's output $\boldsymbol{\gamma}^{(l-1)}$, multiplied by a weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$, added to a bias vector $\mathbf{b}^{(l)} \in \mathbb{R}^{N_l}$, and then fed into the activation function. The set of parameters for this layer is $\boldsymbol{\theta}_l = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$. In turn, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_L\}$ is the set of all trainable parameters of the neural network. Thus, the output can be written as

$$\boldsymbol{\gamma}^{(l)} = \phi_a^{(l)}(\mathbf{W}^{(l)}\boldsymbol{\gamma}^{(l-1)} + \mathbf{b}^{(l)})\,. \tag{7.13}$$

The depth L, the *width* of each layer N_l, i.e., the number of units, and the activation functions $\phi_a^{(l)}$ are said to form the *hyperparameters* of the neural network.

Input Normalization and One Hot Vectors

The MLP takes the feature vector $\boldsymbol{\gamma}^{(0)} \in \mathbb{R}^{N_0}$ as input. In a communications scenario, these input features might be received noisy channel output symbols, noisy distance measures or channel coefficients. In some applications, the neural network inputs are discrete cluster indices obtained by preprocessing the training data. For instance, the discrete inputs are described by the indices $y_i \in \{1, 2, \dots |\mathcal{Y}|\}$ and span the space $\mathcal{Y}^{N_0}$. These labelings are arbitrary, and a meaningful mapping $\mathcal{Y}^{N_0} \mapsto \mathbb{R}^{N_0}$ might be hard to find. In turn, it is more common to use so-called *one-hot encoding* [GBC16]. One-hot encoding represents the decimal indices by a sparse vector which contains only one non-zero entry indicating the decimal number, i.e., if $|\mathcal{Y}| = 4$

$$1 \mapsto \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \; 2 \mapsto \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \; 3 \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \; 4 \mapsto \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} . \tag{7.14}$$

Due to the sparsity, the vector-matrix product in the input layer "selects" only a particular column of weights, which is then processed in the next layers.

Activation Functions for the Output Layer

The activation function of the output layer depends on the task. Furthermore, the choice of the activation function of the output layer is closely related to the selected loss function. As described above, the cross-entropy loss is a common loss function for classification problems. However, the cardinality of the output variable X also impacts the choice of the output layer.

Sigmoid Activation for Binary Variables: For a binary variable X, it is sufficient if the neural network outputs $\Pr(\mathsf{X} = 1|\mathbf{y})$, where $\mathbf{y} = \boldsymbol{\gamma}^0$. Hence, the neural network output must lie in the interval $[0, 1]$ to be a valid probability. As shown in Fig. 7.3a, this can be directly achieved by choosing the sigmoid activation function for the output units. The input to the activation function is $\varkappa = \mathbf{w}^{(L)\mathsf{T}}\boldsymbol{\gamma}^{(L-1)} + \mathbf{b}^{(L)}$. Please note that $\varkappa$ is referred to as *logit* in the machine learning literature [GBC16]. Using the definition of the sigmoid function $\sigma(\varkappa) = \frac{1}{1+e^{-\varkappa}}$ it can be shown that for binary variables X, i.e., the logit is closely related to the LLR,

$$\Pr(\mathsf{X} = 1|\mathbf{y}) = \sigma(\varkappa) \tag{7.15}$$

$$\Pr(\mathsf{X} = 1|\mathbf{y}) = \frac{1}{1 + e^{-\varkappa}} \tag{7.16}$$

$$\Pr(\mathsf{X}=1|\mathbf{y})\left(1+e^{-\varkappa}\right)=1 \tag{7.17}$$

$$\Pr(\mathsf{X}=1|\mathbf{y})\left(1+e^{-\varkappa}\right)=\Pr(\mathsf{X}=1|\mathbf{y})+\Pr(\mathsf{X}=0|\mathbf{y}) \tag{7.18}$$

$$\Pr(\mathsf{X}=1|\mathbf{y})e^{-\varkappa}=\Pr(\mathsf{X}=0|\mathbf{y}) \tag{7.19}$$

$$e^{-\varkappa}=\frac{\Pr(\mathsf{X}=0|\mathbf{y})}{\Pr(\mathsf{X}=1|\mathbf{y})} \tag{7.20}$$

$$\varkappa=-\log\frac{\Pr(\mathsf{X}=0|\mathbf{y})}{\Pr(\mathsf{X}=1|\mathbf{y})}\,. \tag{7.21}$$

Softmax Activation for Categorial Variables: For discrete random variables X with n possible values, a generalization of the sigmoid activation is required. This is the so-called soft max($\boldsymbol{\varkappa}$) function. The input to the soft max is the unnormalized vector of logits $\boldsymbol{\varkappa}=\mathbf{W}^{(L)\mathsf{T}}\boldsymbol{\gamma}^{(L-1)}+\mathbf{b}^{(L)}$, i.e., $\boldsymbol{\varkappa}=[\varkappa_1,\varkappa_2,\ldots,\varkappa_{N_L}]$. Mathematically,

$$\text{soft}\max(\boldsymbol{\varkappa})=\left[\frac{\exp(\varkappa_1)}{\sum_{j=1}^{N_L}\exp(\varkappa_j)},\frac{\exp(\varkappa_2)}{\sum_{j=1}^{N_L}\exp(\varkappa_j)},\ldots\frac{\exp(\varkappa_{N_L})}{\sum_{j=1}^{N_L}\exp(\varkappa_j)}\right], \tag{7.22}$$

i.e., the soft max function translates the unnormalized log-probabilities into a proper probability vector $p(x|\mathbf{y})$. As shown in [GBC16], the soft max is a soft version of the arg max function. In turn, after training, the most-likely output $\hat{x}$ can be obtained by computing $\hat{x}=\arg\max_i \varkappa_i$, which is computationally very efficient as it works on the unnormalized logits directly.

7.2 Neural Information Bottleneck Decoding

The ongoing discussion about the future communication standard 6G is dominated by innovations considering deep learning using neural networks in the communication system design. In this section neural networks shall be used as universal function approximator to implement known message mappings $t^{\text{out}}=f(\mathbf{t}^{\text{in}})$. As many popular deep learning applications drive the development of dedicated deep learning hardware, dedicated deep learning processors can be expected to be available in many future communication devices. Therefore, channel decoders tailored to such dedicated deep learning circuits might become of great practical interest.

Thus, the approach presented in this section, i.e., *neural* information bottleneck decoding, is not intended as a competing approach to unrolled information bottleneck decoder architectures from Section 5.6 but rather meant to accompany these

approaches and support novel architectures.

We first published the idea of neural information bottleneck decoding in [SLB20]. Neural information bottleneck decoding aims to combat the curse of dimensionality discussed in Section 5.2.2.3 faced by the exponential increase of the lookup tables if the node degree increases and no decomposition is performed. Therefore, the idea is to represent the mapping $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$ as a trainable function $f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}})$, which can be efficiently implemented using a neural network with parameters $\boldsymbol{\theta}$. The neural network is trained based on samples and learns both its approximate design distribution $p(x, \mathbf{t}^{\text{in}})$ and the mapping $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$ during training. With this approach, even long input vectors $\mathbf{t}^{\text{in}}$, which hold a huge number of N incoming messages, can be processed in one step [SLB20]. Also, especially if not an unrolled decoding architecture is leveraged, the slight changes in the lookup tables over the iterations might result in large memory requirements for a very large number of decoding iterations $i_{\max}$. Thus, the neural network is trained to output all extrinsic messages generated by a check/variable node in one step, rather than excluding each respective incoming message from the target edge separately. Further, an *iteration-aware* neural network can be designed to take the iteration as additional input and replace *all* lookup tables that vary for different iterations in the information bottleneck decoders presented so far. Please note that the neural network is not only used to learn to decode the channel code as, e.g., in [NBB16]. Instead, the focus is to learn message-passing decoding with *coarsely quantized* messages. This section proposes two different approaches to neural information bottleneck decoding:

Supervised Neural Information Bottleneck Decoder First, we train neural networks, which output the outgoing messages t^{out} given the input vectors $\mathbf{t}^{\text{in}}$ in an information bottleneck decoder. This requires mappings for the variable and the check nodes designed using discrete density evolution but without node decompositions, as described in Section 5.2. Once constructed, we analyze the overall resulting N-input 1-output mappings $t^{\text{out}} = f(\mathbf{t}^{\text{in}})$ for each node type and each iteration i, which have prohibitive complexity for a one-shot implementation in a lookup table. Then, we train neural networks $f_{\boldsymbol{\theta}}^{(i)}(\mathbf{t}^{\text{in}})$ that process all inputs $\mathbf{t}^{\text{in}}$ of a node in iteration i in one shot for each decoding iterations i. These networks mimic and replace the iteration depending lookup tables. As generalizations, we also train one network $f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}}, i)$ for the variable nodes and another one for the check nodes, which take the iteration index i as additional inputs. Furthermore, these networks implement N-input N-output mappings that generate all outgoing messages of a variable/check node at once. In turn, the supervised neural information bottleneck decoder learns to exploit redundancies and connections among lookup tables for different iterations

and the computation of the outgoing message within a node. Please note that such a design-objective is practically infeasible with the model-based approach presented in previous chapters.

Unsupervised Neural Information Bottleneck Decoder In a second variant of the neural information bottleneck decoder, the mapping is not known in advance through discrete density evolution. Instead, the *unsupervised* neural information bottleneck decoder learns the coarsely quantized message mappings based on the training data considering the maximization of relevant information as a training objective.

7.2.1 Supervised Neural Information Bottleneck Decoder

The supervised neural information bottleneck decoder considers a neural network with ReLU activation for the hidden layer and a linear activation for the input and the output layer. The input vector $\mathbf{t}^{\text{in}}$ and the output cluster t^{out} are represented by their one-hot equivalents.

Clearly, after one-hot encoding the input to the neural network is very sparse. In turn, the vector-matrix product in the first hidden layer $\mathbf{W}^{(1)}\mathbf{t}^{\text{in}}_{(i)}$ can be interpreted as a selection of columns of the weight matrix $\mathbf{W}^{(1)}$ of the hidden layer. The final output is

$$p(t^{\text{out}}|\mathbf{t}^{\text{in}}) = \text{soft}\max\left(\mathbf{W}^{(L)}\max\left(0, \mathbf{W}^{(1)}\mathbf{t}^{\text{in}}\right)\right) = \text{soft}\max\left(\mathbf{W}^{(L)}\text{reLU}\left(\mathbf{W}^{(1)}\mathbf{t}^{\text{in}}\right)\right) \tag{7.23}$$

during training with the soft max activation function as only one hidden layer is used, i.e., $L = 2$. This operation is replaced by arg max during decoding. Please note, that applying arg max yields the integer t^{out}. Thus, in total, the proposed architecture consists of one selection step, one addition step and a final arg max operation.

As a second approach, we consider the binary representation of the cluster index instead of its one-hot encoding. In this case, the exemplary input-output pair ($\mathbf{t}^{\text{in}} = [0, 2, 3, 0]^{\mathsf{T}}, t^{\text{out}} = 1$) with $t^{\text{in}}, t^{\text{out}} \in \{0, 1 \ldots, 3\}$ is represented as:

$$0 \mapsto \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; 2 \mapsto \begin{bmatrix} 0 \\ 1 \end{bmatrix} \; 3 \mapsto \begin{bmatrix} 1 \\ 1 \end{bmatrix} \; 0 \mapsto \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad t^{\text{out}} \mapsto \begin{bmatrix} 1 \\ 0 \end{bmatrix} . \tag{7.24}$$

Although this input is less sparse, the input dimensions are reduced, and, in turn, the number of neurons and the complexity of the neuronal network is reduced. Furthermore, this input encoding is more common in signal processing and communications than the one-hot encoding and might be of larger practical interest as it is easier to integrate in conventional signal processing units.

For training, samples are generated using known lookup tables similar to the design proposed in [LB18], i.e., the lookup tables designed using an information bottleneck algorithm and discrete density evolution. As a result, there is no fixed data set, which is split into a training set and a test set. Instead, samples $(t^{\text{out}}_{(i)}, \mathbf{t}^{\text{in}}_{(i)})$ are constantly generated by sampling from $p(\mathbf{t}^{\text{in}}|x)$ and feeding the resulting $\mathbf{t}^{\text{in}}_{(i)}$ to the target network. In general, the design of information bottleneck decoders employs a decomposition of the node operation into a concatenation of two-input lookup tables. Nonetheless, the concatenation of lossy compression results in a very small overall additional loss in relevant information compared to the non-decomposed lookup table design (cf. Section 5.2.2.3). This section shows that a trainable function can be learned to approximate the non-decomposed lookup table even with fewer parameters as required by a decomposed design and better performance than the decomposed design.

As we face only discrete random variables, the neural network $f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}})$ solves a classification problem. Hence, we choose the cross-entropy loss

$$\mathrm{H}\left(t^{\text{out}}|\mathbf{t}^{\text{in}}\right) = \mathbb{E}_{(t^{\text{out}}_{(i)}, \mathbf{t}^{\text{in}}_{(i)}) \sim p(t^{\text{out}}, \mathbf{t}^{\text{in}})}\left[-\log(p_{\boldsymbol{\theta}}(t^{\text{out}}|\mathbf{t}^{\text{in}}))\right] \tag{7.25}$$

as target function to be minimized during training.

7.2.1.1 Multiple-Output Nodes and Iteration-Aware Neural Network

The architecture discussed in the previous section allows computing *one* particular outgoing message t^{out} from the incoming messages $\mathbf{t}^{\text{in}}$ carrying the extrinsic information. However, in message-passing decoding messages *for each* connected target edge of a node need to be computed. In information bottleneck decoding, this is usually achieved by copying the concatenated structure or reusing the respective structure sequentially with a different removed incoming message received from the target edge. This either increases the required space or the decoding latency. The first extension of the proposed approach is a neural network that outputs *all* outgoing messages in one shot from *all* incoming messages. In turn, the neural network learns to consider only certain incoming messages to generate the extrinsic information and reuse possible intermediate results. Again we consider only one hidden layer with

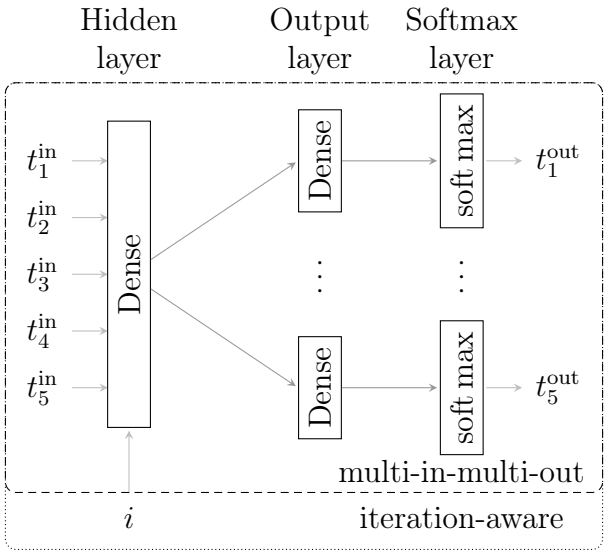


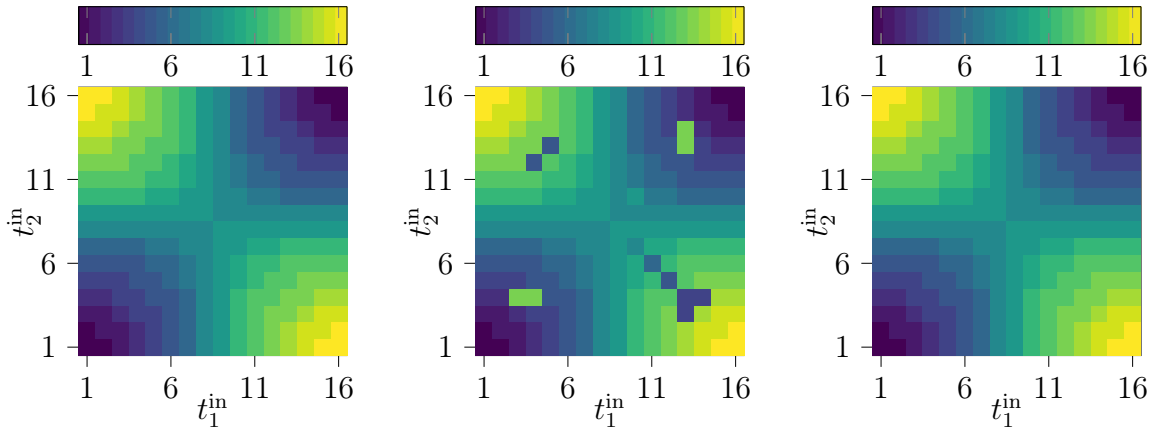
Figure 7.4: Neural network architecture able to output all messages at once (multi-in-multi-out) and the decoding iteration as additional input (iteration-aware).

ReLU activation. Such an architecture is shown in Fig. 7.4 for an exemplary degree five check node indicated by the densely dashed area.

In general, the optimum mappings $f^{(i)}(\mathbf{t}^{\text{in}})$ for a variable node and for a check node, respectively, depend on the iteration i. In an unrolled architecture, this iteration-dependency can be dealt with easily. In some applications, e.g., on a digital signal processor, it is important that the same mapping is used for all iterations to enable an efficient implementation. Therefore, an extended neural architecture is considered, which takes the iteration i as an additional input. Thus, the neural network represents $f_{\boldsymbol{\theta}}(\mathbf{t}^{\text{in}}, i)$ where i denotes the iteration. Several architecture were investigate in the scope of this thesis and the best found architecture is shown in Fig. 7.4 indicated by the densely dotted area. There the neural network is additionally fed with the iteration index i, which is again encoded as a one-hot vector.

7.2.1.2 Evaluation

In this section, the performances of the trained neural networks are evaluated in a two-step process. First, we analyze the neural network's ability to represent the node operations designed using the information bottleneck method as in [LB18]. The network's task is to represent these mappings as accurately as possible in all decoder iterations to achieve the same performance as the conventional information bottleneck decoders from [LB18]. We analyze how many parameters $\boldsymbol{\theta}$ are required for an adequate representation and aim for much fewer parameters than lookup table entries in the original decoders. If adequate representation is possible with



(a) Original two-input mapping designed using an information bottleneck algorithm

(b) Mapping learned by a neural network with 8 hidden nodes

(c) Mapping learned by a neural network with 16 hidden nodes

Figure 7.5: Comparison of a two-input lookup table designed with an information bottleneck algorithm and two learned mappings, which implement their respective mappings using neural networks with 8 and 16 hidden nodes.

fewer parameters, the designed decoders should have the same performance as the lookup table based decoders, as they implement comparable mappings. Therefore, in a second step, we validate this assumption by analyzing the bit error rates of the respective decoders. In detail, we consider two architectures all with one hidden layer. First, a neural network with $N \cdot 2^{q+1} + i_{\max}$ hidden neurons and one-hot input (one-hot in. NN) is considered. Second, we implement a neural network with $N \cdot q + i_{\max}$ hidden neurons and binary input (binary in. NN) where $q = \log_2(|\mathcal{T}_{ch}|)$.

Analysis of the Learned Mappings In order to evaluate if neural networks can learn to represent information bottleneck decoding mappings, we consider an illustrative example of a mapping $t^{\text{out}} = f(t_1^{\text{in}}, t_2^{\text{in}})$ with only two input messages. Fig. 7.5 shows a comparison of an exemplary two-input mapping from the check nodes of a $q = 4$ bit information bottleneck decoder (cf. Fig. 7.5a). The figure also shows learned mappings using neural networks, which used the lookup table from the information bottleneck approach as training data (cf. Fig. 7.5b and Fig. 7.5c).

The neural network consists of only one hidden layer. However, the number of neurons in the hidden layer was 8 in Fig. 7.5b and 16 in Fig. 7.5c. Vividly, both networks can implement the lookup tables, quite well, but the accuracy of the representation depends on the number of parameters used. Please note that the number N of incoming messages t_n^{in} to be processed by a check node during decoding is determined

Table 7.1: Number of trainable parameters of the neural network compared to the number of lookup table entries for a regular LDPC decoder with, $|\mathcal{T}| = 16$, $d_c = 6$, $d_v = 3$ and $i_{\max} = 50$ decoding iterations.

decoder	No. parameters check node	No. parameters var. node
information bottleneck decoder - opened node (open inf. bot.)	$(d_c - 2)\lvert\mathcal{T}\rvert^2 \cdot d_c \cdot i_{\max}$ $= 307,200$	$(d_v - 1)\lvert\mathcal{T}\rvert^2 \cdot d_v \cdot i_{\max}$ $= 76,800$
information bottleneck decoder - closed node (closed inf. bot.)	$\lvert\mathcal{T}\rvert^{d_c-1} \cdot d_c \cdot i_{\max}$ $= 314,572,800$	$\lvert\mathcal{T}\rvert^{d_v} \cdot d_v \cdot i_{\max}$ $= 614,400$
neural information bottleneck decoder - one-hot input (one-hot NN)	58,900	20,292
neural information bottleneck decoder - binary input (binary-in NN)	16,854	8,710

by its degree d_c. Table 7.1 shows the number of trainable parameters for check node degree $d_c = 6$ and variable node degree $d_v = 3$ and different architectures. Please remember that we devised a multi-in-multi-out setting and iteration awareness. In turn, a single neural network replaces the check nodes respectively variable nodes, that was trained to output all outgoing messages in one step from all incoming messages and also the iteration index. The architecture was shown in Fig. 7.4. For ease of brevity, we call the number of lookup table entries of the conventional information bottleneck decoder also parameters in Table 7.1. As shown in Table 7.1, the number of parameters required by the neural network is much smaller than a one-shot lookup table, which shows an exponential increase in m. Interestingly, considering that $i_{\max} = 50$ decoding iterations are performed, the overall number of parameters for the neural network is even smaller than for the decomposed node operation. Thus, the neural network is shown to mimic the information bottleneck lookup tables accurately with fewer parameters. Similar observations can be made for the variable nodes. However, it should be noted that the parameters in an information bottleneck decoder are coarsely quantized and represented by unsigned integers, whereas the parameters in a neural inform

To investigate the performance of the trained *iteration-aware* neural network, Code 1 with $(d_v, d_c) = (3, 6)$ and $N = 8,000$ as in Chapter 5 and $q = 4$ bit decoding is considered. A maximum of $i_{\max} = 50$ decoding iterations was performed for all decoders.

Fig. 7.6 shows bit error rate results for LDPC encoded data transmission over an

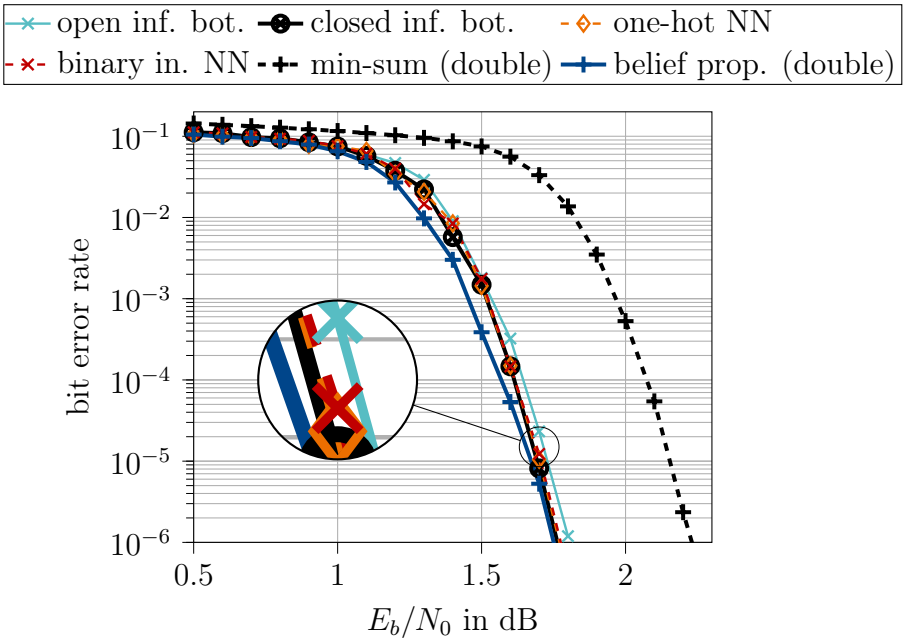


Figure 7.6: Bit error rate simulations for Code 1 from Chapter 5 with $i_{\max} = 50$ decoding iterations over AWGN channel with BPSK.

AWGN channel with BPSK modulation. Results are shown for the information bottleneck decoder proposed in [LB18] which harnesses decomposed node operations to ensure a reasonable memory complexity (open inf. bot). Furthermore, results for the one-shot information bottleneck algorithm are shown, where one impractically large lookup table is used to replace the arithmetical node operations. Please note that the memory requirements would prohibit a practical implementation of this decoder, but it is provided as a benchmark. It is shown that this decoder shows a slightly better bit-error-rate performance compared to the information bottleneck decoder with concatenated lossy lookup operations. Nonetheless, this decoder is used to generate the training data for the neural information bottleneck decoder. Provided computer simulations reveal that the neural information bottleneck decoder is able to mimic the one-shot information bottleneck decoder perfectly but with practical complexity and much fewer parameters. It is shown that the performance is superior to the error-correction capability of the practically relevant sequential information bottleneck decoder (open inf. bot.). This holds likewise for the decoder with one-hot input and binary input. Moreover, a double-precision belief-propagation decoder and a min-sum decoder serve as references. Interestingly, it can be observed that the proposed neural information bottleneck decoders outperform the min-sum decoder and approaches the performance of the double-precision belief propagation decoder (belief prop.) nearly as close as the lookup table based information bottleneck decoder, i.e., up to 0.05 dB over E_b/N_0. This holds although fewer parameters are needed, single iteration-aware neural networks are used for the check, and the variable nodes

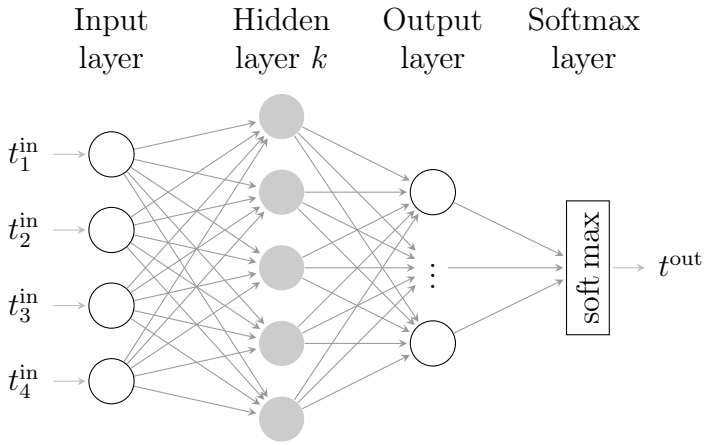


Figure 7.7: Neural network architecture of an unsupervised neural information bottleneck decoder mapping.

and all outgoing messages of a node are computed in one step during decoding.

7.2.2 Unsupervised Neural Information Bottleneck Decoder

In the previous section, the supervised neural information bottleneck decoder built upon mappings designed using discrete density evolution. Thus, the neural network was trained in a supervised manner to exploit redundancy in tables across various iterations and implement a one-shot mapping that returns all outgoing messages at once with manageable implementation complexity.

This section extends this idea to unsupervised learning, i.e., no ground-truth lookup tables exist. Instead, the neural network learns to maximize the relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ directly. It was described in Chapter 3 that deterministic mappings preserved the highest amount of relevant information. However, the training of neural networks requires differentiable functions. Thus, it is impossible to embed the arg max operation in the training process. Hence, the soft max operation is used as a differentiable alternative. However, this function returns stochastic mappings $p(t^{\text{out}}|\mathbf{t}^{\text{in}})$ in the information bottleneck sense. Thus, it is important to include the trade-off parameter β in the training process to ensure that the mapping is as little stochastic as possible. In turn, we require the loss function

$$J(\boldsymbol{\theta}) = -\mathrm{I}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\text{out}}) + \frac{1}{\beta}\mathrm{I}_{\boldsymbol{\theta}}(\mathsf{T}^{\text{in}};\mathsf{T}^{\text{out}}) \tag{7.26}$$

where we compute the relevant information $\mathrm{I}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\mathrm{out}})$ as

$$\mathrm{I}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\mathrm{out}}) = \sum_{x}\sum_{t^{\mathrm{out}}} p_{\boldsymbol{\theta}}(x,t^{\mathrm{out}}) \log_2\left(\frac{p_{\boldsymbol{\theta}}(x|t^{\mathrm{out}})}{p(x)}\right) \tag{7.27}$$

with

$$\begin{aligned} p_{\boldsymbol{\theta}}(x,t^{\mathrm{out}}) &= \sum_{\mathbf{t}^{\mathrm{in}}} p_{\boldsymbol{\theta}}(t^{\mathrm{out}}|\mathbf{t}^{\mathrm{in}}) p(x,\mathbf{t}^{\mathrm{in}}) && (7.28)\\ &= \sum_{\mathbf{t}^{\mathrm{in}}} p(\mathbf{t}^{\mathrm{in}}) p_{\boldsymbol{\theta}}(t^{\mathrm{out}}|\mathbf{t}^{\mathrm{in}}) p(x|\mathbf{t}^{\mathrm{in}}) && (7.29)\\ &= \mathbb{E}_{p(\mathbf{t}^{\mathrm{in}})}\left[p_{\boldsymbol{\theta}}(t^{\mathrm{out}}|\mathbf{t}^{\mathrm{in}}) p(x|\mathbf{t}^{\mathrm{in}})\right] . && (7.30) \end{aligned}$$

Based on the empirical training data, the empirical mutual information estimate $\tilde{\mathrm{I}}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\mathrm{out}})$ for sufficiently large batch size equals

$$\tilde{\mathrm{I}}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\mathrm{out}}) = \sum_{x}\sum_{t^{\mathrm{out}}} \tilde{p}_{\boldsymbol{\theta}}(x,t^{\mathrm{out}}) \log_2\left(\frac{\tilde{p}_{\boldsymbol{\theta}}(x|t^{\mathrm{out}})}{p(x)}\right) \tag{7.31}$$

with

$$\tilde{p}_{\boldsymbol{\theta}}(x,t^{\mathrm{out}}) = \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} p_{\boldsymbol{\theta}}(t^{\mathrm{out}}_{(i)}|\mathbf{t}^{\mathrm{in}}_{(i)})) p(x|\mathbf{t}^{\mathrm{in}}_{(i)}) . \tag{7.32}$$

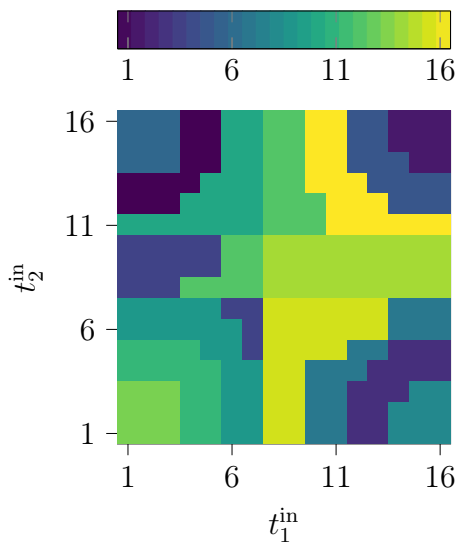
Thus, the empirical loss function yields

$$\tilde{J}(\boldsymbol{\theta}) = -\tilde{\mathrm{I}}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\mathrm{out}}) + \frac{1}{\beta}\tilde{\mathrm{I}}_{\boldsymbol{\theta}}(\mathsf{T}^{\mathrm{in}};\mathsf{T}^{\mathrm{out}}) . \tag{7.33}$$

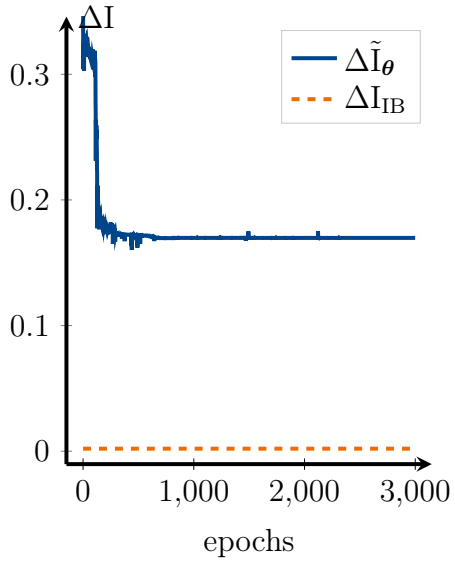
Another possible approach is to estimate the mutual information using the mutual information neural estimator (MINE) from [BBR+18].

7.2.2.1 Evaluation

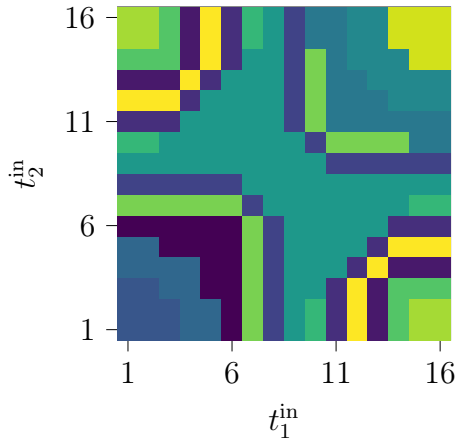
Having defined a proper loss function, we investigate the mappings vividly and also in term of preserved relevant information. Therefore, we measure the difference $\Delta\tilde{\mathrm{I}}_{\boldsymbol{\theta}} = \mathrm{I}(\mathsf{X};\mathsf{T}^{\mathrm{in}}) - \tilde{\mathrm{I}}_{\boldsymbol{\theta}}(\mathsf{X};\mathsf{T}^{\mathrm{out}})$ over the training process and also compare the result to the preserved mutual information of the model-based approach as described in Chapter 5 and termed $\Delta\mathrm{I}_{\mathrm{IB}} = \mathrm{I}(\mathsf{X};\mathsf{T}^{\mathrm{in}}) - \mathrm{I}_{\mathrm{IB}}(\mathsf{X};\mathsf{T}^{\mathrm{out}})$ in Fig. 7.8.



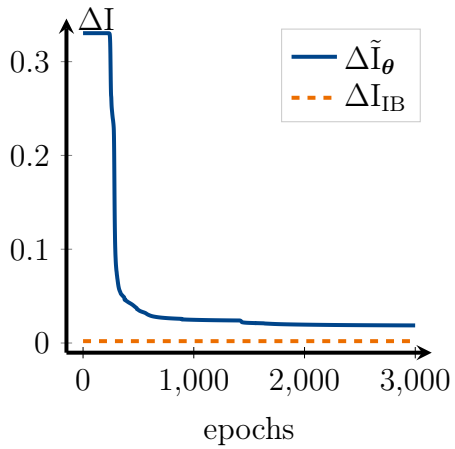
(a) Mapping for $\beta = 5$



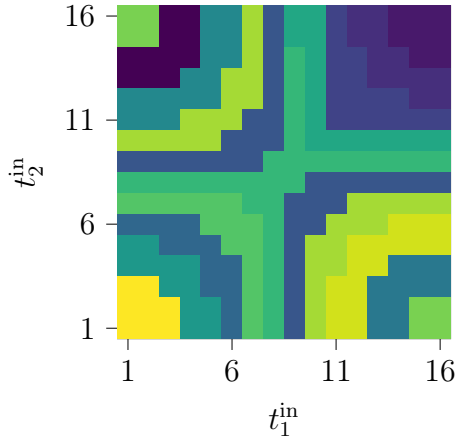
(b) Evolution for $\beta = 5$



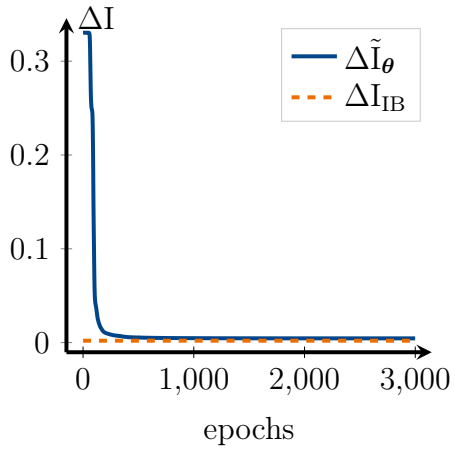
(c) Mapping for $\beta = 10$



(d) Evolution for $\beta = 10$



(e) Mapping for $\beta = 100$



(f) Evolution for $\beta = 100$

Figure 7.8: Learned check node mappings $p(t^{\text{out}}|t_1^{\text{in}}, t_2^{\text{in}})$ for different β and the evolution of the loss in preserved relevant information $\Delta\tilde{\text{I}}_{\boldsymbol{\theta}}$ over the training epochs compared to the loss in relevant information $\Delta\text{I}_{\text{IB}}$ preserved with the model-based information bottleneck approach.

In Fig. 7.8, three different values for β are considered, i.e., $\beta = 5, 10, 100$. As β increases, the objective changes from compression to the preservation of relevant information. It can be observed in Fig. 7.8a that the number of used clusters stays below the possible cardinality $|\mathcal{T}^{\text{out}}| = 16$. Furthermore, for small β the unsupervised neural information bottleneck mappings lose a considerable amount of relevant information as the focus is on compression and not only on the maximization of relevant information. However, for a sufficiently large β we observe that the learned mapping tends to become deterministic and the gap between the model-based information bottleneck mapping from Eq. (5.42) and the unsupervised neural information bottleneck mapping vanishes. In turn, it can be concluded that the presented neural network architecture is able to learn relevant-information-preserving mappings solely based on empirical training data. In this chapter, we restrict our investigations to binary check node operations for clarity. However, an extension to non-binary information bottleneck node operations is straightforward.

7.3 Deep Learning of Mutual-Information-Based End-to-End Communication

It is common in communications engineering to split the communication chain into individual sub-blocks. However, following Shannon's view on communication, the universal design goal is to maximize the end-to-end performance of a communication system in terms of mutual information, which theoretically requires solely two blocks, i.e., a transmitter and a receiver as depicted in Fig. 7.9.

Thus, the mutual-information-based end-to-end objective can be formalized as

$$\max_{p(x|s)} \mathrm{I}(\hat{\mathsf{S}}; \mathsf{X}) \tag{7.34}$$

which clearly relates to the information bottleneck setup considered throughout this thesis. Here, $\hat{\mathsf{S}}$ represents the relevant random variable and the physical constraint on the cardinality of the modulation scheme, i.e., $|\mathcal{X}|$, defines the latent dimension. Thus, the constellation symbols x resort to realizations of the compressed random variable X in an analog information bottleneck setup.

Furthermore, assuming a deterministic mapping $p(x|s)$, Eq. (7.34) can be directly related to the channel capacity

$$C = \max_{p(x)} \mathrm{I}(\mathsf{X}; \mathsf{Y}), \tag{7.35}$$

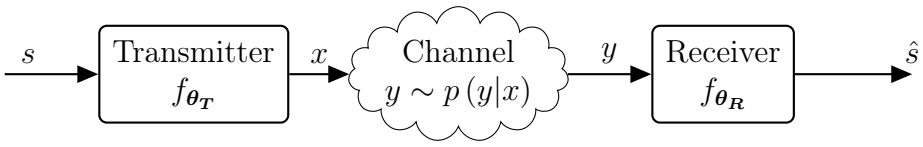


Figure 7.9: A universal transmission chain as an autoencoder architecture.

which underlines that symbol-wise end-to-end communication aims to be capacity-achieving. In general, finding $p(x)$ maximizing $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is a very difficult problem for several reasons. First, analytical optimization would require knowledge of the statistics of the transmission channel described by the conditional distribution $p(y|x)$. However, even if $p(y|x)$ is known, the respective integrals are often intractable. Considering the problem of mapping bits onto transmit symbols and the respective demapping, two approaches exist to find capacity-achieving constellations. Either the location of the constellation points in the complex plane itself can be optimized, i.e., the sample space of X, or the frequency with which certain constellation points are sent, i.e., $p(x)$. These techniques are referred to as *geometric shaping* or *probabilistic shaping*, respectively.

Recently, the concept of autoencoders received considerable interest in the research community to facilitate *end-to-end learning* of capacity-achieving communication. End-to-end learning of communication systems implements the transmitter, channel, and receiver as a single neural network, referred to as an autoencoder. The autoencoder is than trained to reproduce its input at its output [OH17]. This approach enables joint optimization of the transmitter and receiver for a specific channel model without extensive mathematical analysis. This idea was pioneered in [OH17] and has led to many extensions towards channel coding [NBB16], orthogonal frequency-division multiplexing (OFDM) [FCD+18; AH20], multiple-input multiple-output (MIMO) [KHH+20] and joint source-channel coding [BBG19]. Moreover, a handful of techniques were proposed to enable training of the end-to-end systems without a channel model. In [YLJ+18; ORW19; DHC+20; YLL+20], it was proposed to leverage generative adversarial neural networks to learn a differentiable model of the channel and then to train the end-to-end system using the learned model. Autoencoder-based communication systems have subsequently been extended towards other settings, such as optical fiber [KCT+18], optical wireless [ZZC+19], and molecular communications [MDJ+19].

This section derives an information-theoretical perspective on autoencoders and relates the commonly used symbol-wise categorical cross-entropy to known information-theoretic quantities [SAH19]. It is derived that this loss function is indeed equivalent to maximizing the mutual information between the channel input and output. Based

on this observation, an end-to-end learning system that learns capacity-achieving probabilistic shaping and geometric shaping is devised.

However, practical systems usually rely on bit-interleaved coded modulation and bit-metric decoding at the receiver because of its reasonable complexity. Thus, the constellation points forming the channel input constellation are labeled by bit vectors, and bit levels are treated independently. Therefore, the points that form a constellation learned by an autoencoder optimized on the symbol-wise categorical cross entropy must be labeled by bit vectors to be used in a practical bit-metric decoding-based system. This is typically a challenging task, even for low modulation orders. More importantly, the mutual information between channel input and output is known to *not* be a rate achievable by bit-metric decoding [BSS15]. This suggests that $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is not necessarily an appropriate metric for trainable communication systems leveraging bit-metric decoding. Thus, in the second part of this section, we generalize the information-theoretic findings related to symbol-wise autoencoders also to bit-wise autoencoders. In this section, the learned constellation and the associated bit-labeling are investigated. The superior performance of the learned end-to-end system over conventional systems in terms of coded bit error rates is presented.

In general, autoencoders unfold their full potential in combination with unknown channel models and real over-the-air measurements. We published those investigations in cooperation with Nokia Bell Labs France and the University Stuttgart in [CAD+20] but excluded a detailed discussion for ease of brevity. The investigations showed that the performance improvements observed in numerical computer simulations can be reproduced in real-word experiments.

7.3.1 Symbol-Wise Autoencoder

Fig. 7.10 shows a symbol-wise autoencoder. It consists of the following main parts:

Transmitter The incoming bit sequence is denoted by $\mathbf{b} = [b_1, \ldots, b_N]^\mathsf{T}$ and is of size N and mapped onto hypersymbols $s \in \mathcal{S}$. Here, $\mathcal{S}$ is the finite, discrete eventspace of the random variable S. The sequence of hypersymbols denoted by $\mathbf{s} = [s_1, \ldots, s_M]^\mathsf{T}$ of size M is fed into a symbol mapper which maps $\mathbf{s}$ onto a sequence of complex transmit symbols $\mathbf{x} \in \mathbb{C}^M$ according to $p(\mathbf{x}|\mathbf{s})$, where M can be seen as the number of channel uses. To perform probabilistic shaping, the trainable distribution $p_{\boldsymbol{\theta}_S}(s)$ with parameters $\boldsymbol{\theta}_S$ ensures that symbols $s \in \mathcal{S}$ appear such that $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is maximized. The actual probabilistic shaping, respectively, the learning procedure to determine $p_{\boldsymbol{\theta}_S}(s)$ is described in Section 7.3.1.1. When considering

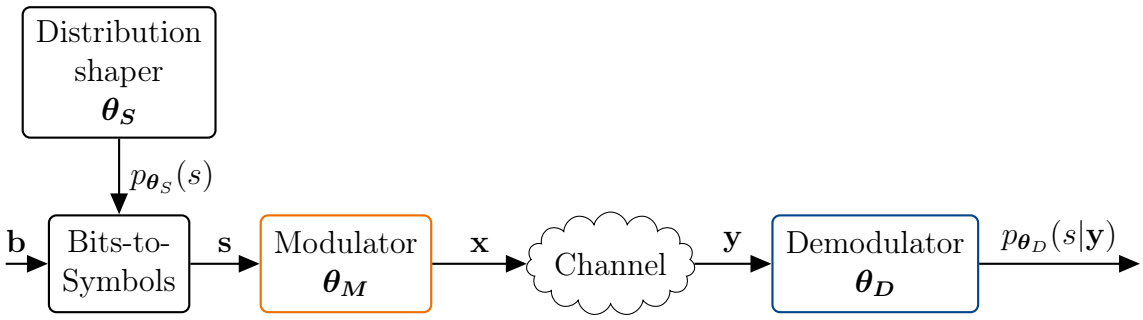


Figure 7.10: Trainable end-to-end communication system for joint geometric and probabilistic shaping.

geometric shaping, also the mapper is implemented as a neural network $f_{\boldsymbol{\theta}_M}$ with trainable parameters $\boldsymbol{\theta}_M$. Both, the mapper and the distribution could be trainable, resulting in joint probabilistic-geometric shaping. Again, more details regarding the training of $\boldsymbol{\theta}_M$ are deferred to Section 7.3.1.2.

Channel The channel models the physical transmission medium. It takes the complex transmit symbols $\mathbf{x} \in \mathbb{C}^M$ and outputs M received samples pooled in $\mathbf{y} \in \mathbb{C}^M$. The possibly unknown transition probability $p(\mathbf{y}|\mathbf{x})$ describes the input-output relation.

Receiver The receiver is comprised of a demapper. The demapper is also implemented as a neural network with trainable parameters $\boldsymbol{\theta}_D$, which maps each received sample $\mathbf{y} \in \mathbb{C}$ to a probability vector over the set of symbols $\mathcal{S}$. The mapping defined by the demapper is denoted by $\tilde{p}_{\boldsymbol{\theta}_D}(s|\mathbf{y})$, and defines an approximation of the true posterior distribution $p(s|\mathbf{y})$.

7.3.1.1 Joint Geometric and Probabilistic Shaping

Let us assume a bits-to-symbols mapper which maps the bits from $\mathbf{b}$ to symbols s such that these symbols follow the distribution $p_{\boldsymbol{\theta}_S}(s)$. This can be done, e.g., using the algorithm presented in [SB16]. Therefore, we consider a transmitter that transmits symbols according to $p_{\boldsymbol{\theta}_S}(s)$, and a receiver which aims to reconstruct the transmitted symbols by approximating the posterior distribution $p(s|\mathbf{y})$. Because the mapper $f_{\boldsymbol{\theta}_M}$ maps a symbol $s \in \mathcal{S}$ to a channel symbol vector $\mathbf{x} \in \mathbb{C}$ in a deterministic manner, the distribution over the channel inputs x equals

$$p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x) = \sum_{s=1}^{M} \delta\left(x - f_{\boldsymbol{\theta}_M}(s)\right) p_{\boldsymbol{\theta}_S}(s)\,. \tag{7.36}$$

Please recall that the target of probabilistic shaping is to find $p_{\boldsymbol{\theta}_S}(s)$, such that $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is maximized.

Training is performed considering the demodulation as a classification task and using the categorical cross-entropy as loss function:

$$J(\boldsymbol{\theta}_S, \boldsymbol{\theta}_M, \boldsymbol{\theta}_D) \triangleq \mathbb{E}_{s,y}\left[-\log \tilde{p}_{\boldsymbol{\theta}_D}(s|y)\right] = -\sum_{s=1}^{N} p_{\boldsymbol{\theta}_S}(s) \int_y p\left(y|f_{\boldsymbol{\theta}_M}(s)\right) \log \tilde{p}_{\boldsymbol{\theta}_D}(s|y) dy\,. \tag{7.37}$$

Rewriting the loss function yields

$$\begin{aligned} J(\boldsymbol{\theta}_S, \boldsymbol{\theta}_M, \boldsymbol{\theta}_D) &= -\int_x p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x) \int_y p(y|x) \log \tilde{p}_{\boldsymbol{\theta}_D}(x|y) dy dx \\ &= -\int_x \int_y p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x,y) \log \tilde{p}_{\boldsymbol{\theta}_D}(x|y) dy dx \\ &= -\int_x p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x) \log p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x) dx \\ &\quad - \int_x \int_y p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x,y) \log \frac{\tilde{p}_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M,\boldsymbol{\theta}_D}(x,y)}{p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(y) p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x)} dy dx\,. \end{aligned} \tag{7.38}$$

It is important to notice that $p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x,y)$ is the true joint distribution of X and Y, whereas $\tilde{p}_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M,\boldsymbol{\theta}_D}(x,y)$ is the joint distribution computed from the posterior approximated by the demapper $\tilde{p}_{\boldsymbol{\theta}_D}(x|y)$ where

$$p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x,y) = p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x) p(y|x)\,, \tag{7.39}$$

$$p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(y) = \int_x p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x,y)\,, \tag{7.40}$$

$$\tilde{p}_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M,\boldsymbol{\theta}_D}(x,y) = \tilde{p}_{\boldsymbol{\theta}_D}(x|y) p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(y)\,. \tag{7.41}$$

In a next step one finds

$$J(\boldsymbol{\theta}_S, \boldsymbol{\theta}_M, \boldsymbol{\theta}_D) = \mathrm{H}_{\boldsymbol{\theta}_S}(\mathsf{S}) - \mathrm{I}_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(\mathsf{X};\mathsf{Y}) + \mathbb{E}_y\left[D_{\mathrm{KL}}\left\{p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x|y)||\tilde{p}_{\boldsymbol{\theta}_D}(x|y)\right\}\right]\,, \tag{7.42}$$

where D_{KL} is the Kullback-Leibler divergence. This is a crucial finding of this section as it allows to isolate the mutual information in the loss function. Especially

if probabilistic shaping is performed, often a collapse of the constellation was observed, i.e., multiple symbols where mapped onto the same constellation point, as the minimization of the cross-entropy loss in Eq. (7.42) also implies a minimization of the entropy $\mathrm{H}_{\boldsymbol{\theta}_S}(\mathsf{S})$. Hence, a more appropriate loss $\widehat{J}$ becomes

$$\widehat{J}(\boldsymbol{\theta}_S, \boldsymbol{\theta}_M, \boldsymbol{\theta}_D) \triangleq J(\boldsymbol{\theta}_S, \boldsymbol{\theta}_M, \boldsymbol{\theta}_D) - \mathrm{H}_{\boldsymbol{\theta}_S}(\mathsf{S}) \tag{7.43}$$

which ensures a maximization of the relevant information $\mathrm{I}_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(\mathsf{X};\mathsf{Y})$.

Training the end-to-end system by minimizing $\widehat{J}$ corresponds to maximizing the mutual information of the channel inputs X and outputs Y while minimizing the Kullback-Leibler divergence between the true posterior distribution $p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x|y)$ and the one learned by the receiver $\tilde{p}_{\boldsymbol{\theta}_D}(x|y)$. Moreover, the neural network implementing the receiver should approximate the posterior distribution $p_{\boldsymbol{\theta}_S,\boldsymbol{\theta}_M}(x|y)$ of a constellation maximizing the mutual information with high precision. This is important to avoid learning a constellation for which the posterior distribution is well approximated, but which does not maximize the mutual information.

The challenge of performing probabilistic shaping with machine-learning-based algorithms comes from the difficulty of training a sampling mechanism for symbols s drawn from the finite set $\mathcal{S}$. Let us address this issue exploiting the Gumbel-Softmax trick [JGP17], an extension of the Gumbel-Max trick [HJ12]. The Gumbel-Max trick provides a convenient way to sample a discrete distribution $p_{\boldsymbol{\theta}_S}(s)$ by computing the samples as follows:

$$s = \underset{i=1,\dots,S}{\arg\max}\ (g_i + \log p_{\boldsymbol{\theta}_S}(i)) \tag{7.44}$$

where g_i are i.i.d. samples drawn from a standard Gumbel distribution. Because the arg max operator is not differentiable, one cannot train $p_{\boldsymbol{\theta}_S}(s)$ using usual stochastic gradient descent methods. The key idea of the Gumbel-Softmax trick is to use the softmax function as a differentiable approximation to arg max. More precisely, one generates a vector of dimension $|\mathcal{S}|$, denoted by $\tilde{\mathbf{s}}$, with components

$$\tilde{s}_i = \frac{\exp g_i + \log p_{\boldsymbol{\theta}_S}(i)/\tau}{\sum_{j=1}^{S} \exp g_j + \log p_{\boldsymbol{\theta}_S}(j)/\tau},\ i = 1, \dots, |\mathcal{S}| \tag{7.45}$$

where τ is a positive parameter called the *temperature*. The vector $\tilde{\mathbf{s}}$ is a probability vector where $\arg\max_i\ \tilde{s}_i = s$. This is an approximation of the *one-hot* representation of s denoted by $\mathbf{s}$. As the temperature goes to zero, samples generated by the Gumbel-Softmax method become closer to one-hot vectors, and their distribution becomes closer to $p_{\boldsymbol{\theta}_S}(s)$ [JGP17].

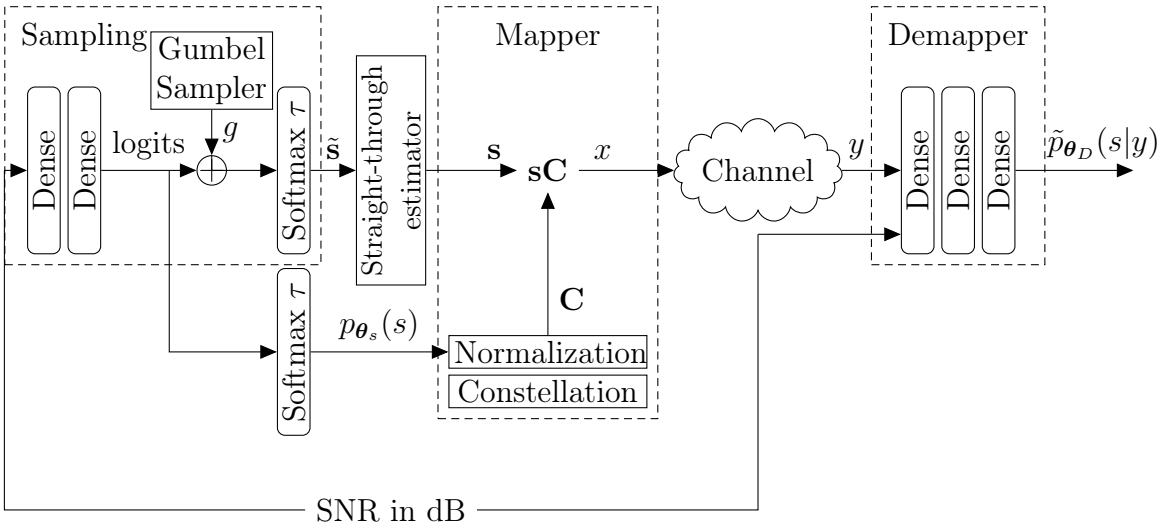


Figure 7.11: End-to-end architecture for geometric and probabilistic shaping .

Fig. 7.11 shows the architecture of the sampling mechanism. The optimal probabilistic shaping depends on the SNR of the channel over which we transmit, which therefore must be *a priori* known by the transmitter [FAB+16]. The SNR in dB is fed to a neural network with trainable parameters $\boldsymbol{\theta}_S$. The neural network is made of two dense layers and generates the *logits* of the symbols distribution $p_{\boldsymbol{\theta}_S}(s)$. The logits are the unnormalized log probabilities, and the distribution $p_{\boldsymbol{\theta}_S}(s)$ can be retrieved by applying a softmax activation to the logits. The first dense layer is made of 128 units with ReLU activations and the second layer of S units with linear activations. The number of units was determined empirically. By tuning the neural network parameters $\boldsymbol{\theta}_S$ one, therefore, optimizes the distribution $p_{\boldsymbol{\theta}_S}(s)$.

The mapper is made of a matrix of dimension $|\mathcal{S}| \times 2$ followed by a normalization layer, as shown in Fig. 7.11. The two columns of constellation matrix are used to represent the real and imaginary part of the complex constellation points separately. The matrix consists of the unnormalized constellation point locations. The normalized constellation is denoted by $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_s, \ldots, \mathbf{c}_{|\mathcal{S}|}]^\mathsf{T}$ where $\mathbf{c}_s \in \mathbb{R}^2,\ s = 1, \ldots, |\mathcal{S}|$. By taking the product of a one-hot vector $\mathbf{s}$ with the sth element set to one with $\mathbf{C}$, one selects a constellation point $\mathbf{x} = \mathbf{c}_s$. Normalization is performed to ensure the average energy per constellation point equals unity, i.e.,

$$\sum_{s \in \mathcal{S}} p_{\boldsymbol{\theta}_S}(s)\, |\mathbf{x}_s|_2^2 = 1\,. \tag{7.46}$$

If only probabilistic shaping is performed, the constellation is not trained, and some fixed constellation, e.g., QAM, is used. When geometric shaping is performed, the

constellation is trainable and $\boldsymbol{\theta}_M = \mathbf{C}$.

A drawback of the Gumbel-Softmax trick is that the generated vector $\tilde{\mathbf{s}}$ is only approximating a true one-hot vector $\mathbf{s}$. Consequently, taking the product of $\tilde{\mathbf{s}}$ and $\mathbf{C}$ results in a linear combination of multiple constellation points $\mathbf{c}_s$. To avoid this issue, we take advantage of the straight-through estimator [BLC13], which uses the true one-hot vector $\mathbf{s}$ when performing the forward path and the approximate one-hot vector $\tilde{\mathbf{s}}$ for the backward path at training.

The trainable demapper consists of three dense layers, as shown in Fig. 7.11. The first two layers are made of 128 units with ReLU activations, while the last layer is made of S units with softmax acitvation to output a probability vector over the set of symbols $\mathcal{S}$. Also, the demapper takes the SNR in dB as input. This was motivated by the observation that the posterior distribution depends on the SNR, and it was found experimentally to be crucial to achieve the best performance.

7.3.1.2 Simulation Results Symbol-Wise Autoencoder

In this section, the mutual information of the channel input and output achieved for the symbol-wise autoencoder is compared to state-of-the-art modulation schemes considering an AWGN channel. Simulation results for Rayleigh fading channels were published in [SAH19]. Training of the end-to-end system introduced in the previous section is performed with respect to the loss function $\widehat{J}$ defined in Eq. (7.43), as opposed to previous works, which train autoencoders with respect to the usual cross-entropy J defined in Eq. (7.42). Using $\widehat{J}$ as loss function is crucial to enable probabilistic shaping, as using J would rather lead to a minimization of the source entropy $H_{\boldsymbol{\theta}_S}(S)$ instead of maximization of the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$. The training was performed with the Adam stochastic gradient descent (SGD) variant, with batch sizes progressively increasing from 100 to 10,000 and learning rates progressively decreasing from 10^{-3} to 10^{-5}. When probabilistic shaping was performed, the temperature in (7.45) was set to 10. All these parameters were found empirically through intensive computer simulations and heuristic optimizations. First, the results that were obtained when both the locations and probabilities of the constellation points are optimized are shown, i.e., joint geometric and probabilistic shaping. The considered modulation orders N are 16, 64, 256, and 1024.

In this section, both the probability distribution and the geometry are assumed to be trainable. Pure probabilistic shaping of QAM is studied in great detail in [FAB+16]. It is well known that for an AWGN channel, distributions $p(s)$ from the Maxwell-Boltzmann family maximize the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ and, thus, achieve the

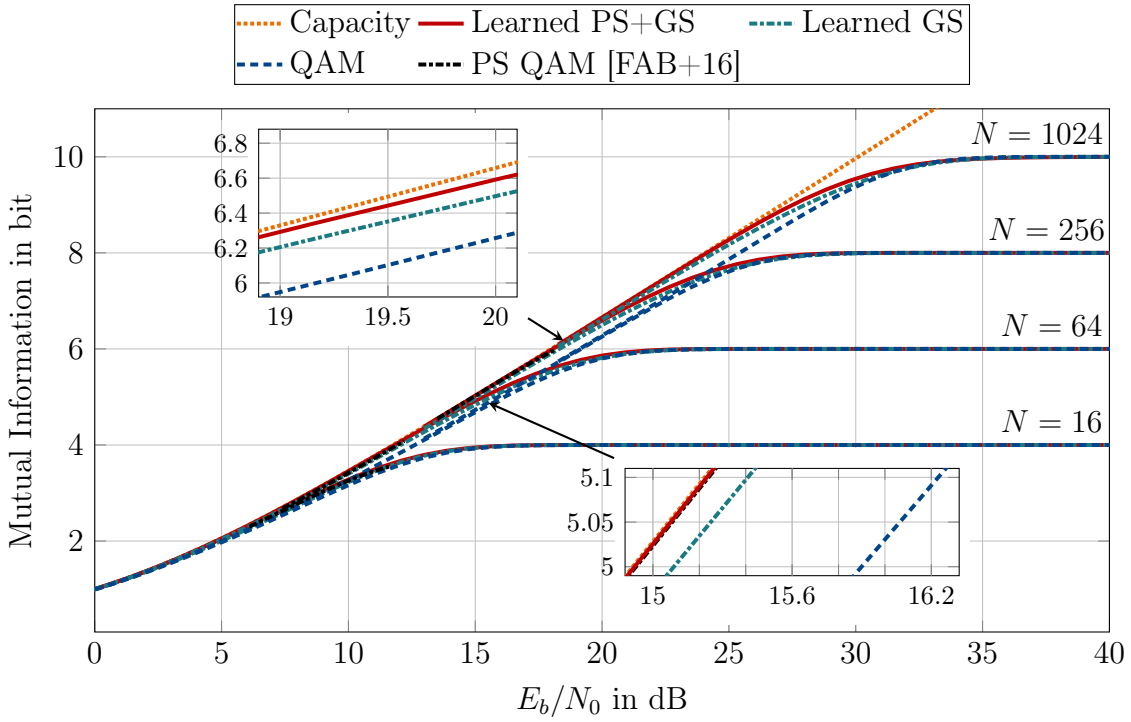


Figure 7.12: Mutual information achieved by the reference schemes and the learned joint probabilistic and geometric shaping on the AWGN channel. Magnification is done for $N = 256$.

capacity in a certain regime [FAB+16]. However, when training a symbol-wise autoencoder, we do not enforce the learning of distributions from this family. The following modulation schemes are compared: QAM with no probabilistic shaping (QAM), trainable geometric shaping (Learned GS), trainable joint probabilistic and geometric shaping (Learned PS+GS) both optimized as described in the previous section, and QAM with Maxwell-Boltzmann distribution for probabilistic shaping as in [FAB+16]. QAM with Maxwell-Boltzmann from [FAB+16] is presented only for modulation orders of 16, 64, and 256, and the distributions are optimized for specific SNR values. In Fig. 7.12, the mutual information $\mathrm{I}(\mathsf{X};\mathsf{Y})$ achieved by the proposed joint shaping scheme (cf. red solid curve) is compared to several reference schemes. It is shown that geometric and probabilistic shaping is superior to all reference schemes. It is observed that already geometric shaping alone significantly outperforms QAM. Furthermore, it can be seen that the proposed neural network learns a joint probabilistic and geometric shaping, which operates very close to capacity for a wide range of SNRs. From Fig. 7.12, one observes that the achievable gains enabled by constellation shaping are more significant the higher the modulation order gets. Notice that the proposed approach benefits from training a neural network that computes optimized shaping distributions over a wide range of SNRs. In contrast, the Maxwell-Boltzmann distribution from [FAB+16] is only optimized

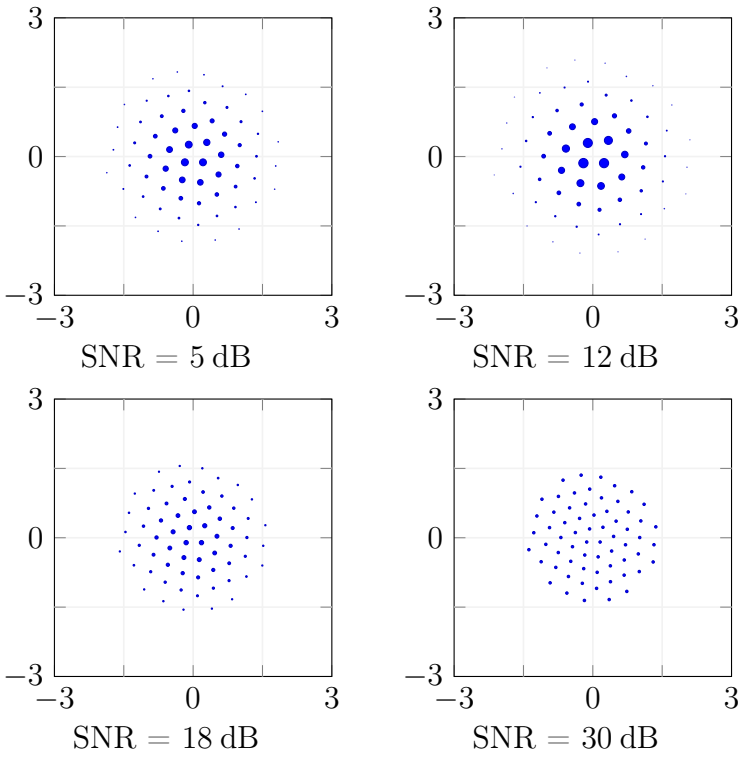


Figure 7.13: Learned joint shaping for $N = 64$. The size of the points is proportional to their probabilities of occurrence.

for a limited SNR range. This is highlighted in the second magnification for the SNR range from 19 to 20 dB. There, the black curve for Maxwell-Boltzmann distribution is not shown as in this regime the achievable rate for a 256-ary constellation already deviates from the theoretical channel capacity curve for a continuous input alphabet. However, the presented end-to-end learning approach operates extremely close to the continuous capacity reference and significantly outperforms a QAM constellation. For this evaluation, the sampling mechanism was trained for SNR values ranging from -2 dB to 40 dB. Notice that the actual channel model was not known to the autoencoder to achieve this result.

Fig. 7.13 shows the joint probabilistic and geometric constellations for various SNR values and for $M = 64$. It can be observed that the learned shaping is similar to a two-dimensional Gaussian distribution. For lower SNRs, the learned shaping favors constellation points closer to the origin. Due to the normalization, which ensures that $\mathbb{E}\{|x|^2\} = 1$, the less frequently transmitted outer points are placed further apart from the origin. As the SNR increases, the distribution becomes uniform.

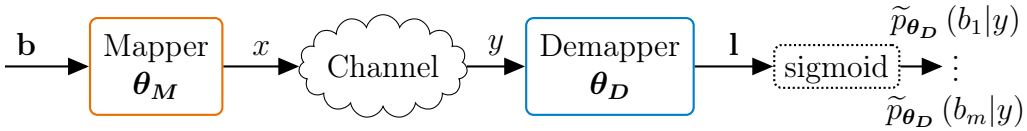


Figure 7.14: Bit-wise autoencoder.

7.3.2 Bit-Wise Autoencoder

Although it appears as a trivial modification at first glance, the transition from symbol-wise to bit-wise autoencoder-based communication systems turns out to require a carefully adjusted framework. The main difference in the objective is to minimize the BER instead of the SER. This immediately leads to the question of how to find the optimal labeling scheme for the learned constellations, which is not part of previous autoencoder implementations. The constellation obtained by training on the symbol-wise cross-entropy needs to be labeled to be used in a conventional BICM system. However, since the learned constellation points do not usually form a grid (cf, Fig. 7.13), as in a conventional QAM, finding the optimal labeling is a combinatorial problem with $2^m!$ possibilities (neglecting symmetries). Therefore, we can only rely on sub-optimal heuristics to label the constellation points *after* the training process, as, e.g., done in [KCT+18].

Fig. 7.14 depicts a bit-wise autoencoder. In the bit-wise autoencoder, the mapper takes as input a bit vector of length m, and the demapper outputs m logits (one per bit) which form a vector denoted by $\mathbf{l} \in \mathbb{R}^m$. Probabilities over the m bits, denoted by $\widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y),\ j = 1, \ldots, m$, are obtained by element-wisel application of the sigmoid function.

An autoencoder-based communication system is typically trained by minimizing the categorical cross-entropy between the true posterior distribution $p_{\boldsymbol{\theta_M}}(s|y)$ and the one learned by the receiver $\widetilde{p}_{\boldsymbol{\theta_D}}(s|y)$, averaged over all the possible channel outputs y, i.e.,

$$\mathbb{E}_y\left[\mathrm{H}\left(p_{\boldsymbol{\theta_M}}(s|y), \widetilde{p}_{\boldsymbol{\theta_D}}(s|y)\right)\right] = -\mathbb{E}_y\left[\sum_{s=0}^{2^m-1} p_{\boldsymbol{\theta_M}}(s|y) \log\left(\widetilde{p}_{\boldsymbol{\theta_D}}(s|y)\right)\right]. \tag{7.47}$$

As shown in the previous section, this is equivalent to maximizing the mutual information between the channel input X and output Y. For practical use of a constellation learned by an autoencoder trained on Eq. (7.47) with bit-metric decoding, the constellation needs to be labeled, i.e., each constellation point needs to be mapped to a unique bit vector. This task is non-trivial and becomes quickly intractable even for small modulation orders. Moreover, $\mathrm{I}(\mathsf{X};\mathsf{Y})$ is not an achievable rate by

bit-metric decoding [BSS15]. Therefore, even if optimal labeling is assumed (e.g., found by exhaustive search), a constellation learned by minimization of Eq. (7.47) is not necessarily optimal for practical bit-metric decoding receivers.

An achievable rate by bit-metric decoding is the bit-wise mutual information [Böc18]

$$R \coloneqq \mathrm{H}\left(\mathsf{B}\right) - \sum_{j=1}^{m} \mathrm{H}\left(\mathsf{B}_j|\mathsf{Y}\right) \leq \mathrm{I}(\mathsf{X};\mathsf{Y}) \tag{7.48}$$

where B denotes the multivariate random variable associated with the input bit vector $\mathbf{b}$ of length m.

Optimization of the bit-wise autoencoder is done by minimizing the total binary cross entropy

$$J(\boldsymbol{\theta_M}, \boldsymbol{\theta_D}) = \sum_{j=1}^{m} \mathbb{E}_y \left[\mathrm{H}\left(p_{\boldsymbol{\theta_M}}(b_j|y), \widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)\right)\right] \tag{7.49}$$

$$= \sum_{j=1}^{m} \mathbb{E}_{y,b_j} \left[-\log\left(\widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)\right)\right] \tag{7.50}$$

which is closely related to the bit-wise mutual information

$$J(\boldsymbol{\theta_M}, \boldsymbol{\theta_D}) = H(\mathbf{B}) - R + \sum_{j=1}^{m} \mathbb{E}_y \left[D_{\mathrm{KL}}\left\{p_{\boldsymbol{\theta_M}}(b_j|y)||\widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)\right\}\right] \tag{7.51}$$

where $p_{\boldsymbol{\theta_M}}(b_j|y),\ j = 1, \ldots, m$ are the true posterior distributions. Interestingly, according to [Böc18], (7.49) itself is an achievable rate for an imperfect receiver. Rewriting (7.49) as (7.51) allows to connect the actual bit-wise mutual information to the achievable rate for imperfect receivers. In turn, it becomes clear how trainable communication systems can easily outperform conventional systems with imperfect receivers. In more detail, the first term on the right-hand side of (7.51), i.e., $H(\mathbf{B})$ is the entropy of the bit vector generated by the source, which is typically constant and equals the number of bits m mapped to one symbol (assuming uniformly distributed bits). The second term, i.e., R, is the bit-wise mutual information, and the third term is the sum of the Kullback-Leibler (KL) divergences between the true posterior distributions $p_{\boldsymbol{\theta_M}}(b_j|y)$ and the ones learned by the demapper $\widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)$. The KL divergences accounts for the sub-optimality of the receiver. Training of the end-to-end system by minimizing J, therefore, corresponds to maximizing R, which is suited to bit-metric decoding, while minimizing the KL divergence between the optimal demapper and the one learned at the receiver. Moreover, because the mapper assigns each bit vector to a constellation point, joint geometric shaping and

bit labeling is performed when minimizing J. The neural network implementing the demapper should approximate the posterior distributions $p_{\boldsymbol{\theta}_M}(b_j|y)$ of a constellation maximizing the bit-wise mutual information with high precision. This avoids learning a constellation where the posterior distributions are well-approximated, but the bit-wise mutual information is not maximized.

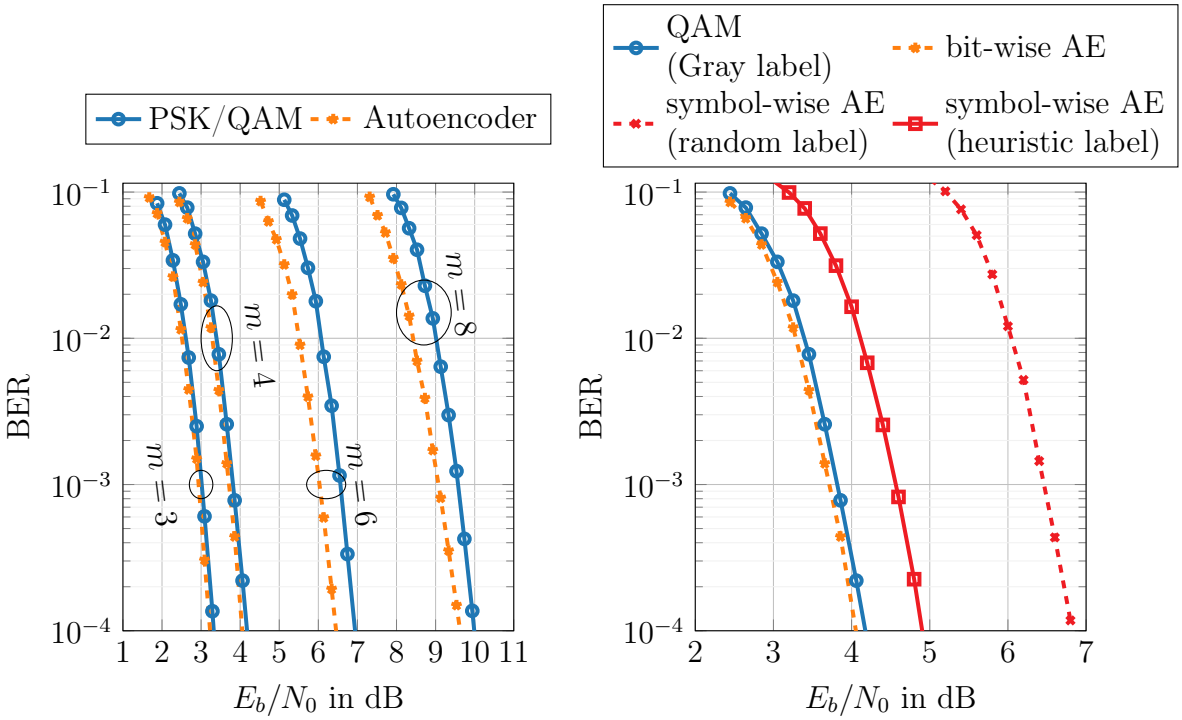
7.3.2.1 Simulation Results Bit-Wise Autoencoder

This section evaluates the bit-wise autoencoder on an AWGN channel. We demonstrate that training on the bit-wise mutual information allows seamless integration with practical bit-metric decoding receivers, as well as joint optimization of constellation shaping and labeling. Therefore, a bitstream is fed to an LDPC encoder which generates codewords $\mathbf{c}$ of length n. The number of bits per channel use is m and it is assumed that n is a multiple of m. Each codeword is broken apart into $s = \frac{n}{m}$ bit vectors $\mathbf{b}$ of length m, i.e., $\mathbf{c} = \left[\mathbf{b}_1^\mathsf{T}, \ldots, \mathbf{b}_s^\mathsf{T}\right]^\mathsf{T}$. Each bit vector $\mathbf{b}_i, i = 1, \ldots, s$, is mapped into a complex baseband symbol $x_i \in \mathbb{C}$, and is sent over the channel. On the receiver side, the demapper processes each received sample $y_i \in \mathbb{C}$ and generates LLRs $\mathbf{l}_i \in \mathbb{R}^m$. Finally, the LLRs of the entire codeword $\mathbf{l} = \left[\mathbf{l}^{(1)\mathsf{T}}, \ldots, \mathbf{l}^{(s)\mathsf{T}}\right]^\mathsf{T}$ are fed into a belief-propagation decoder.

Like the symbol-wise autoencoder, the mapper in Fig. 7.14 includes a neural network that generates a continuum of constellations $\mathcal{C} \in \mathbb{C}^{2^m}$ that depends on the SNR. The neural network included in the mapper is made of two dense layers with ReLU activations and one dense layer with linear activations. Please note again that the architecture of the mapper was determine empirically. The remaining internal structure of the mapper and the internal structure of the demapper are similar to the symbol-wise autoencoder sketched in Fig. 7.11. Please remember that the learned constellation set is SNR dependent, i.e., mapper and demapper know the SNR. However, this is just a generalization of the case when re-training per SNR is performed as for trainable systems, the SNR is always implicitly part of the training data. In case that feedback of the SNR is not possible, the same setup can be trained with fixed SNR input at the price of a slightly reduced BER performance. Recall that in *classical* communication systems, e.g., in the 5G standard, one typically defines the modulation and coding scheme based on the SNR.

An irregular LDPC code from the 802.11n standard [IEE16] was considered, with rate $R_c = 0.5$ and codeword length $N = 1296$ bit.

The batch-size $|\mathcal{B}|$ for training was set to 500, whereas the learning rate was progressively decreased from 10^{-3} to 10^{-5}. For training, the SNR was randomly and



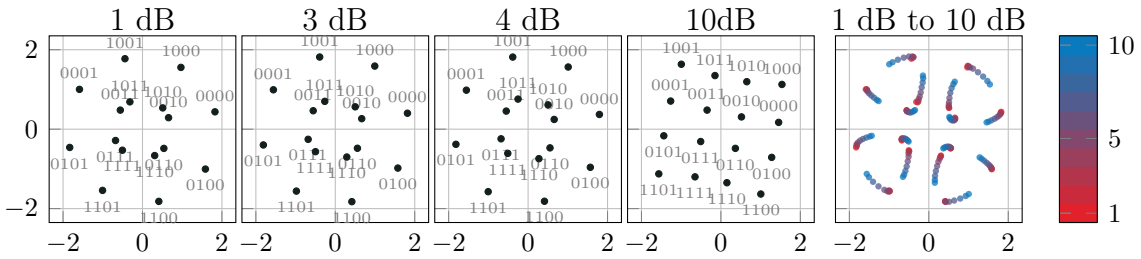
(a) Coded BER achieved by PSK/QAM constellations and the bit-wise autoencoder for $m = 3,\ 4,\ 6$ and 8. The baseline is PSK for $m = 3$, and QAM otherwise.

(b) Achieved coded BER with $m = 4$ bit for baseline QAM, bit-wise autoencoder and symbol-wise autoencoder (with random and heuristic labeling).

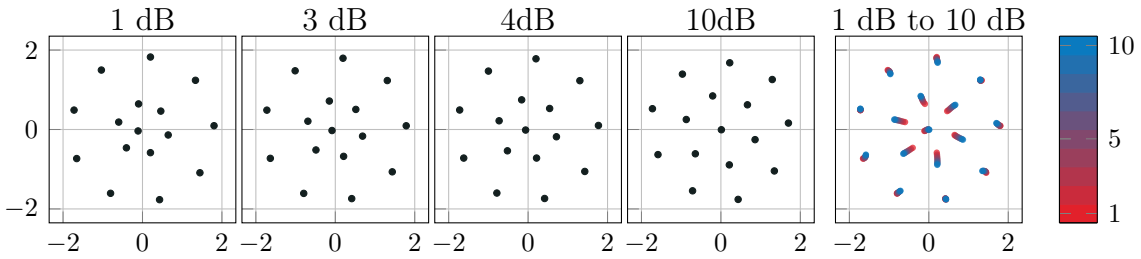
Figure 7.15: Coded BERs for different bit-wise autoencoders and different reference schemes.

uniformly selected for each example from a 4 dB range centered on the waterfall region of the code. Note that, at training, the channel encoder and decoder are not needed, as the loss function Eq. (7.49) is based on the output of the demapper. For evaluation, the number of iterations performed by the belief-propagation decoder was set to 40.

The (coded) BER achieved by the bit-wise autoencoder was compared to the BER of PSK for $m = 3$ and to QAM for $m = 4$, 6, and 8. For the considered baselines, maximum-likelihood demapping was used. Fig. 7.15a shows the BER achieved by the compared approaches with the setup presented above. It can be seen that the schemes learned by the bit-wise autoencoder outperform the baselines. Moreover, the gains achieved by the learning schemes increase with the modulation order, reaching 0.8 dB for $m = 8$, compared to 0.3 dB for $m = 3$.



(a) Bit-wise



(b) Symbol-wise

Figure 7.16: Constellations learned for $m = 4$ bit at varying SNR. The learned labeling of the bit-wise autoencoder is shown in gray.

To get insight into the learned constellation geometries, Fig. 7.16 shows the constellations learned for $m = 4$, when leveraging the bit-wise and symbol-wise autoencoder, respectively. It can be seen that training on the bit-wise mutual information (Fig. 7.16a) leads to a constellation that differs significantly from the one obtained from training on the symbol-wise cross-entropy loss (cf. Fig. 7.16b). Recall that using the constellation learned with the symbol-wise autoencoder (cf. Fig. 7.16b) in a bit-metric decoding-based system requires an additional heuristic labeling step, which is typically not trivial.

Besides the labeling, the optimal position of the constellation points can differ under different metrics. An intuitive example is depicted in Fig. 7.16, where, for the low SNR region, the bit-wise optimized autoencoder (Fig. 7.16a) *clusters* constellation points into groups that only differ in one bit position. This effectively weakens the reliability of this bit position while improving the other positions and, thereby, optimizes the overall achievable information rate. However, in the alternative symbol-metric approach, an ambiguity is unavoidable between neighboring symbols and not taking into consideration during the training. Therefore, a degraded bit-wise performance can be observed if the training is done for the symbol-metric approach. When looking at the corresponding symbol-metric optimized constellation in Fig. 7.16b,

we see that such a clustering does not occur for the exact same channel parameters after training. While a symbol-wise trained constellation improves the SER, it degrades the achievable BER as the process of finding optimal bit labels is not part of the training objective. Fig. 7.15b shows the coded BER performance of a symbol-wise trained autoencoder system with randomly chosen labels and heuristic labels compared with a QAM baseline with Gray labeling and a bit-wise optimized autoencoder. It should be emphasized that the heuristic labeling is not necessarily optimal but underlines the difficulties in finding such labeling. As it can be seen, the BER performance of the symbol-wise optimized autoencoder with heuristic labels is ≈ 0.7 dB worse than the conventional QAM baseline and even ≈ 0.8 dB worse than the bit-wise optimized autoencoder. These results suggest that one should train on the bit-wise mutual information when using bit-metric decoding. Moreover, in the case of the bit-wise autoencoder, the learned labeling is also shown in Fig. 7.16a. One can see that a form of Gray labeling was learned, where two points close to each other are assigned labels that differ in only one bit.

7.4 Summary

This chapter addressed the recently enforcing shift from model-based to data-driven system design. We highlighted that mutual-information-based system design is natural in deep learning and often done implicitly. The opportunities of a data-driven mutual-information-based system design over a classical model-driven design were outlined based on two distinct applications. In detail, this chapter discussed the neural information bottleneck decoder and capacity-achieving end-to-end learning. The findings in this chapter should be considered as a starting point for intensive future research as the presented scenarios might only exploit a fraction of the entire potential of the sketched approaches. In the following, the main findings of this chapter are summarized.

Previous chapters revealed the excellent performance of information bottleneck decoders despite coarse quantization. However, the practical implementation of the derived mappings depends mainly on the available hardware and associated resources. Currently, many popular deep learning applications drive the development of dedicated deep learning hardware. As a result, dedicated deep learning processors can be expected to be available in many future communication devices. Therefore, channel decoders tailored to such dedicated deep learning circuits are of great practical interest. This chapter presented the neural information bottleneck decoder in a supervised and in an unsupervised manner. The supervised neural information bottleneck decoder was trained on existing lookup tables but learned very complex

input-output relations for a multiple-input-multiple-output scenario. Here, a single iteration-aware neural network was trained, which learns to exploit redundancies in the computation of the extrinsic messages and also across tables for different iterations. The devised neural network outperforms the open-node information bottleneck decoder from Chapter 5 although much fewer parameters where used. However, the trainable parameters in the neural network were stored with high precision, and the operations in the neural network rely on multiplications. Therefore, future research on the supervised neural information bottleneck decoder should focus on the impact of quantization on the performance as well as an efficient implementation of multiplications. However, these issues are not dedicated to the neural information bottleneck decoder but are of large practical interest for any application of neural networks.

In a second step, the unsupervised neural information bottleneck decoder was presented. This approach can be seen as an instance of a sample-based information bottleneck. Here, the relevant-information-preserving mapping $p(t^{\text{out}}|\mathbf{t}^{\text{in}})$ is learned based on small sample batches during training. It was shown that similar mappings as for the model-based information bottleneck were found, which had access to the exact joint distribution in advance. These observations motivate the application of the unsupervised neural information bottleneck decoder in applications where the computation of the joint distribution is intractable, e.g., in density evolution for non-binary LDPC codes with high field orders. Furthermore, in future research, further constraints on the mapping, e.g., symmetry, might be included in the training objective to avoid local optima and reduce the network complexity.

In the last part of this chapter, end-to-end learning using autoencoders was presented as a generic example of mutual-information-based communication system design. Instead of optimizing individual blocks, the transmitter and receiver were jointly optimized to maximize the relevant information given a constraint on the modulation alphabet's size, respectively the channel quality. By considering constellation shaping as an instance of end-to-end learning, different shaping variants, i.e., geometric and probabilistic shaping, were investigated. Based on an information-theoretic analysis of common loss functions, the sample-based training in neural networks is interpreted as a mutual-information-maximization task. Starting from a symbol-wise autoencoder, which was shown to outperform existing shaping techniques on an AWGN channel, the bit-wise autoencoder was addressed. The training on bit-wise mutual information enabled seamless integration with bit-metric decoding receivers, widely used in practice. The performance improvements over classic QAM baselines result from geometric constellation shaping as well as from learning

of the optimal demapper. Furthermore, it was shown that the bit-wise autoencoder could solve the labeling problem in constellation design, which is known to be very difficult. Thus, learning-based optimization of the full physical layer for a point-to-point link could be completely automated and may be one of the key ingredients of next-generation communication systems. In future work, an extension to multiuser communications, i.e., multiple access and broadcast channels, should be investigated. Please note that this chapter mainly focused on the transmission over an AWGN channel, which can be easily modeled. However, significant gains in the end-to-end performance were observed already for this channel model compared to standardized schemes. Nonetheless, the data-driven approaches typically unfold their full potential if the channel is unknown or difficult to model. However, this requires access to actual measurement data, for example, of non-linear power amplifies or harsh, real-world transmission environments. This was beyond the scope of this thesis but is an interesting field of future research.

Chapter 8

Conclusion

This dissertation studied machine learning methods for reliable transmission under coarse quantization. In particular, the information bottleneck method was utilized in the first part to devise different signal processing units that aim to maximize the relevant information.

In great detail, the information bottleneck method was connected to different quantization techniques in literature with a primary focus on rate-distortion theory. In its origin, the information bottleneck method was treated as a clustering tool, for example, for document classification. Based on several illustrative examples, the transition of this abstract clustering framework into the concept of mutual-information-based signal processing was introduced (Chapter 3).

However, to unfold the full potential of mutual-information-based signal processing, this dissertation enriched the information bottleneck framework with two novel and essential extensions, i.e., information bottleneck graphs and message alignment (Chapter 4). These tools are among the most powerful and crucial contributions of this thesis as they allow to design remarkably enhanced signal processing units for arbitrary irregular graphical models.

Build upon this theoretical foundation, the decoding of LDPC codes under coarse quantization was studied in great detail. Error correction codes are the integral backbone of modern communication systems to ensure reliable transmission. However, the decoding of such error-correcting codes requires the processing of soft-information. Especially if this soft-information cannot be represented with high precision, the decoding performance deteriorates drastically. However, this thesis showed that the so-called information bottleneck decoder could handle the trade-of between coarse quantization, i.e., a low bit width of the exchange messages, and

near-optimum performance extremely well. In addition, the mutual-information-based design approach allows to replace computationally demanding arithmetical operations with simple relevant-information-preserving mappings. These mappings can be implemented in various ways, for example, as lookup tables, static logic circuits, or neural networks, depending on the available hardware architecture. As further contributions, this dissertation presented a very general decoder design suitable for regular, irregular, punctured, or rate-compatible LDPC codes. In turn, the presented decoders are applicable in many modern communication standards like 5G, Wifi, DVB-S2, etc. (Chapter 5).

In general, LDPC codes can be constructed for arbitrary finite fields. However, in practice, the decoding of such non-binary LDPC codes is extremely complex. In this thesis, the design of non-binary LDPC decoders was investigated for field orders up to 16. Furthermore, higher-order modulation schemes were combined with respective non-binary LDPC codes. Using simulations, it was shown that the presented information bottleneck decoder largely outperforms state-of-the-art decoders with a comparable quantization (Chapter 6).

The second part of this work extended the generic information bottleneck framework to data-driven machine learning, including deep learning using neural networks and autoencoders. The discrete density evolution approach used to construct information bottleneck decoders can become tedious, especially for high field orders. If the involved probability distributions cannot be handled by the information bottleneck algorithms, a data-driven approach is required that relies on training data instead of traceable models. This approach is the main contribution of the second part of this dissertation. Using the information bottleneck decoder as a starting point, the neural information bottleneck decoder is presented, which can be utilized in a supervised or unsupervised manner. This approach might also enable new fields of research that are beyond the scope of this thesis, e.g., non-binary LDPC decoder design for large field orders. In addition, Chapter 7 addresses a more holistic perspective on mutual-information-based signal processing, i.e., end-to-end learning using autoencoders. Leveraging information-theoretic quantities, optimized loss functions for data-driven training were derived to build a theoretically capacity-achieving transmission chain. Especially the effect of geometric and probabilistic shaping on an AWGN channel was studied. The promising results motivate further research, including transmission over unknown channel models, non-linear hardware imperfections, multiple access schemes, broadcasting, etc.

In summary, one concludes that both proposed approaches, i.e., the model-based

and data-driven approach, enable a paradigm shift towards mutual-information-based signal processing where the emphasis is on preserving relevant information as a holistic design objective. All signal processing blocks presented in the first part of this dissertation within the broad concept of mutual-information-based signal processing process unsigned integer indices, which require only a few bits in hardware. Despite this coarse quantization, the devised blocks learn to extract the relevant information without significant losses compared to the optimum conventional signal processing algorithms with high precision. Furthermore, the presented concept was generalized to end-to-end learning using autoencoders. Here, the transmitter and receiver are implemented as two individual neural networks that jointly learn to maximize the achievable information rate over a possibly unknown channel.

Appendix A

Proofs and Derivations

A.1 Proof of Proposition 4.2.1

Let us assume two meanings $p(x|t_i, k_j)$ and $p(x|t_n, k_m)$ where $i, n \in \{1, \ldots, |\mathcal{T}|\}$ and $j, m \in \{1, \ldots, N\}$. For binary relevant random variable X, the Kullback-Leibler divergence $D_{\mathrm{KL}}\{p(x|t_i, k_j)||p(x|t_n, k_m)\}$ can be written as

$$D_{\mathrm{KL}}\{p(x|t_i, k_j)||p(x|t_n, k_m)\} \tag{A.1}$$

$$= p(\mathsf{X}=0|t_i, k_j) \log \frac{p(\mathsf{X}=0|t_i, k_j)}{p(\mathsf{X}=0|t_n, k_m)} + p(\mathsf{X}=1|t_i, k_j) \log \frac{p(\mathsf{X}=1|t_i, k_j)}{p(\mathsf{X}=1|t_n, k_m)} \tag{A.2}$$

$$= p(\mathsf{X}=0|t_i, k_j) \log \frac{p(\mathsf{X}=0|t_i, k_j)}{p(\mathsf{X}=0|t_n, k_m)} + (1 - p(\mathsf{X}=0|t_i, k_j)) \log \frac{p(\mathsf{X}=1|t_i, k_j)}{p(\mathsf{X}=1|t_n, k_m)} \tag{A.3}$$

$$= p(\mathsf{X}=0|t_i, k_j) \cdot \left(\log \frac{p(\mathsf{X}=0|t_i, k_j)}{p(\mathsf{X}=0|t_n, k_m)} - \log \frac{p(\mathsf{X}=1|t_i, k_j)}{p(\mathsf{X}=1|t_n, k_m)} \right) + \log \frac{p(\mathsf{X}=1|t_i, k_j)}{p(\mathsf{X}=1|t_n, k_m)} \tag{A.4}$$

$$= p(\mathsf{X}=0|t_i, k_j) \left(L(x|t_i, k_j) - L(x|t_n, k_m)\right) + \log \frac{p(\mathsf{X}=1|t_i, k_j)}{p(\mathsf{X}=1|t_n, k_m)} \tag{A.5}$$

$$= p(\mathsf{X}=0|t_i, k_j) \left(L(x|t_i, k_j) - L(x|t_n, k_m)\right) + \left(\log 1 + e^{L(x|t_n, k_m)} - \log 1 + e^{L(x|t_i, k_j)}\right) \tag{A.6}$$

$$= p(\mathsf{X}=0|t_i, k_j) \left(L(x|t_i, k_j) - L(x|t_n, k_m)\right) + \max(0, |L(x|t_n, k_m)|) - \max(0, |L(x|t_i, k_j)|) + \log 1 + e^{|L(x|t_n, k_m)|} - \log 1 + e^{|L(x|t_i, k_j|)} \tag{A.7}$$

$$= p(\mathsf{X}=0|t_i,k_j)\left(L(x|t_i,k_j)-L(x|t_n,k_m)\right)+\frac{|L(x|t_n,k_m)|}{2}+\frac{L(x|t_n,k_m)}{2} - \frac{|L(x|t_i,k_j)|}{2}-\frac{L(x|t_i,k_j)}{2}+\log 1+e^{|L(x|t_n,k_m)|}-\log 1+e^{|L(x|t_i,k_j|)} \tag{A.8}$$

$$= p(\mathsf{X}=0|t_i,k_j)\left(L(x|t_i,k_j)-L(x|t_n,k_m)\right)+\frac{|L(x|t_n,k_m)|}{2}+\frac{L(x|t_n,k_m)}{2} - \frac{|L(x|t_i,k_j)|}{2}-\frac{L(x|t_i,k_j)}{2}+\log 1+e^{|L(x|t_n,k_m)|}-\log 1+e^{|L(x|t_i,k_j|)} \tag{A.9}$$

$$= \left(p(\mathsf{X}=0|t_i,k_j)-\frac{1}{2}\right)\left(|L(x|t_i,k_j)|-\operatorname{sgn}\left(L(x|t_i,k_j)\right)\operatorname{sgn}\left(L(x|t_n,k_m)\right)|L(x|t_n,k_m)|\right) - 0.5\left(|L(x|t_i,k_j)|-|L(x|t_n,k_m)|\right)+\log 1+e^{|L(x|t_n,k_m)|}-\log 1+e^{|L(x|t_i,k_j|)} \tag{A.10}$$

$$= \left|p(\mathsf{X}=0|t_i,k_j)-\frac{1}{2}\right|\left(|L(x|t_i,k_j)|-\operatorname{sgn}\left(L(x|t_i,k_j)\right)\operatorname{sgn}\left(L(x|t_n,k_m)\right)|L(x|t_n,k_m)|\right) - 0.5\left(|L(x|t_i,k_j)|-|L(x|t_n,k_m)|\right)+\log 1+e^{|L(x|t_n,k_m)|}-\log 1+e^{|L(x|t_i,k_j|)} \tag{A.11}$$

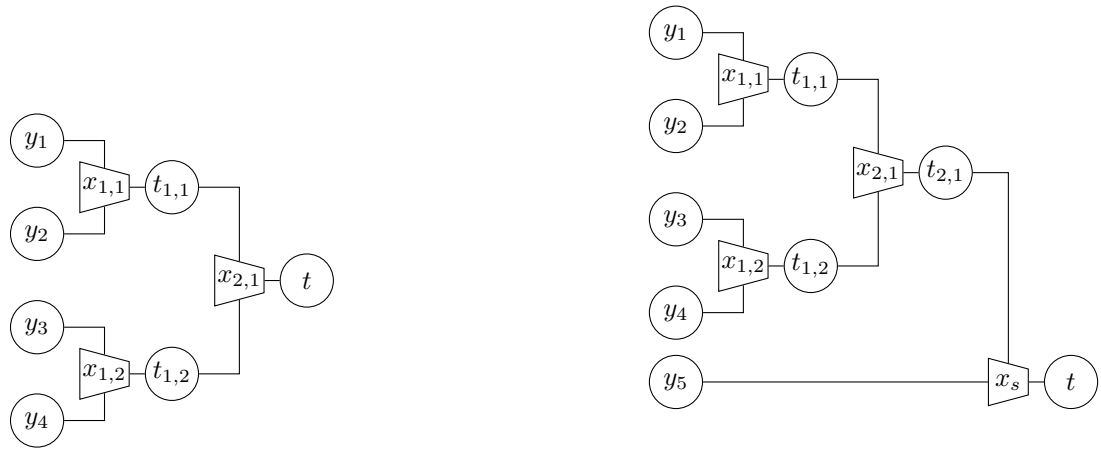
Which yields $D_{\mathrm{KL}}\left\{p(x|t_i,k_j)||p(x|t_n,k_m)\right\}$

$$= \begin{cases} p_0\left(|L_m(x|t_n)|-|L_j(x|t_i)|\right)+\log\frac{1+e^{|L_m(x|t_n)|}}{1+e^{|L_j(x|t_i)|}}, & L_j(x|t_i)\le 0, L_m(x|t_n)\le 0 \\ (1-p_0)\,|L_m(x|t_n)|-p_0\,|L_j(x|t_i)|+\log\frac{1+e^{|L_m(x|t_n)|}}{1+e^{|L_j(x|t_i)|}}, & L_j(x|t_i)\le 0, L_m(x|t_n)> 0 \\ p_0\,|L_m(x|t_n)|-(1-p_0)\,|L_j(x|t_i)|+\log\frac{1+e^{|L_m(x|t_n)|}}{1+e^{|L_j(x|t_i)|}}, & L_j(x|t_i)> 0, L_m(x|t_n)\le 0 \\ (1-p_0)\left(|L_m(x|t_n)|-|L_j(x|t_i)|\right)+\log\frac{1+e^{|L_m(x|t_n)|}}{1+e^{|L_j(x|t_i)|}}, & L_j(x|t_i)> 0, L_m(x|t_n)> 0 \end{cases} \tag{A.12}$$

where $p_0 = p(\mathsf{X}=0|t_i,k_j)$, $L_m(x|t_n) = L(x|t_n,k_m)$, $L_j(x|t_i) = L(x|t_i,k_j)$ and approximately

$$\approx \begin{cases} (1+p_0)\left(|L_m(x|t_n)|-|L_j(x|t_i)|\right), & L_j(x|t_i)\le 0, L_m(x|t_n)\le 0 \\ (2-p_0)\,|L_m(x|t_n)|-(p_0-1)\,|L_j(x|t_i)|, & L_j(x|t_i)\le 0, L_m(x|t_n)> 0 \\ (1+p_0)\,|L_m(x|t_n)|-(2-p_0)\,|L_j(x|t_i)|, & L_j(x|t_i)> 0, L_m(x|t_n)\le 0 \\ (2-p_0)\left(|L_m(x|t_n)|-|L_j(x|t_i)|\right), & L_j(x|t_i)> 0, L_m(x|t_n)> 0 \end{cases} \tag{A.13}$$

for sufficiently large $|L_m(x|t_n)|$ and $|L_j(x|t_i)|$.



(a) (b)

Figure A.1: Illustration of a tree-like structure if M (a) is a power of two, (b) not a power of two.

A.2 Proof of Tree-Like Node Decomposition

A.2.1 Number of Look-Up Stages

The tree-like information bottleneck graph exploits the fact that the look-up tables are equivalent if the probability distributions of the inputs are equivalent. Hence, the vector $\mathbf{y} = [y_1, \ldots, y_M]^{\mathrm{T}}$ has to be split such that a balanced tree is obtained. This tree has the minimum depth, i.e. the minimum number of distinct look-up tables are required. If M, i.e., the number of incoming discrete messages is a power of two, clearly an optimum split will result in $\log_2 M$ stages (cf. Fig. A.1a). The number of levels required if M is no power of two can be obtained from the binary representation of M, i.e., $\mathbf{b}_M = [b_{msb}, \ldots, b_{lsb}]^{\mathrm{T}}$. Here, b_{msb} denotes the most significant bit and b_{lsb} is the least significant bit. Clearly, $\log_2 b_{M,\mathrm{msb}} = \lfloor \log_2 M \rfloor$ gives a lower bound on the required stages. The approach to determine the number of stages s is similar to the decimal-to-binary conversion algorithm. In the first step, M is divided by two, since always to inputs shall be combined. The remainder is the least-significant bit indicating if an extra look-up table is needed, e.g., if M is odd. The quotient is again divided by two; its remainder becomes the next bit again indicating if an extra look-up table is needed. This process repeats until a quotient of one is reached. The number of extra tables, which is equivalent to the number of ones following the most significant bit, is added to $\lfloor \log_2 M \rfloor$. Thus, at most $2 \cdot \lfloor \log_2 M \rfloor$ stages are needed if M is not a power of two.

List of Figures

List of Tables

Bibliography

[3GP18] 3GPP Technical Report 138.212 V15.2.0, *3rd Generation Partnership Project; Technical Specification –5G, NR, Multiplexing and Channel Coding*, Jul. 2018.

[AFD+19] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep Variational Information Bottleneck," Oct. 23, 2019. arXiv: 1612.00410.

[AH18] F. A. Aoudia and J. Hoydis, "End-to-End Learning of Communications Systems Without a Channel Model," in *Proc. of the 52nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, USA, Oct. 2018, pp. 298–303.

[AH20] F. A. Aoudia and J. Hoydis, "End-to-end Learning for OFDM: From Neural Receivers to Pilotless Communication," Oct. 13, 2020. arXiv: 2009.05261.

[Ama16] S.-i. Amari, *Information Geometry and Its Applications*. Tokyo: Springer Japan, Jan. 1, 2016, vol. 194, 376 pp.

[Ari72] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, Jan. 1972.

[BBG19] E. Bourtsoulatze, D. Burth Kurka, and D. Gunduz, "Deep Joint Source-Channel Coding for Wireless Image Transmission," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 567–579, Sep. 2019.

[BBR+18] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *Proc. of the International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 531–540.

[BCJ+74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[Ber71] T. Berger, "Rate distortion theory and data compression," in *Advances in Source Coding*, ser. International Centre for Mechanical Sciences, T. Berger and L. D. Davisson, Eds., Vienna, Austria: Springer Vienna, 1971, pp. 1–39.

[BHM+84] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Springer Science & Business Media, 1984, vol. 2.

[Bis09] C. M. Bishop, *Pattern Recognition and Machine Learning*, Corrected at 8. printing 2009, ser. Information Science and Statistics. New York, NY: Springer, Jan. 1, 2009, 738 pp.

[Bla72] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, Jul. 1972.

[BLC13] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," Aug. 15, 2013. arXiv: 1308.3432.

[Böc18] G. Böcherer, "Achievable Rates for Probabilistic Shaping," May 22, 2018. arXiv: 1707.01134 [cs, math].

[BPK+92] D. Burshtein, V. D. Pietra, D. Kanevsky, and A. Nadas, "Minimum Impurity Partitions," *The Annals of Statistics*, vol. 20, pp. 1637–1646, Sep. 1992.

[Bro14] D. V. Broadcasting (DVB), "Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," *Final Draft ETSI EN*, vol. 302, pp. 2005–01, 2014.

[BS19] A. Balatsoukas-Stimming and C. Studer, "Deep Unfolding for Communications Systems: A Survey and Some New Directions," Jun. 13, 2019. arXiv: 1906.05774.

[BSS15] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4651–4665, Dec. 2015.

[CAD+20] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. T. Brink, "Trainable Communication Systems: Concepts and Prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5489–5503, 2020.

[Cau47] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[CFR+01] S.-Y. Chung, G. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[CGT+05] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, "Information bottleneck for Gaussian variables," *Journal of Machine Learning Research*, vol. 6, p. 165 188, Jan. 1, 2005.

[CHH99] D. Coppersmith, S. J. Hong, and J. R. Hosking, "Partitioning nominal attributes in decision trees," *Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 197–217, 1999.

[Cho91] P. A. Chou, "Optimal partitioning for classification and regression trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 340–354, Apr. 1991.

[CJ08] R. A. Carrasco and M. Johnston, *Non-Binary Error Control Coding for Wireless Communication and Data Storage*. Chichester, UK: John Wiley & Sons, Ltd, Jan. 1, 2008.

[Col12] G. Colavolpe, "Communications over phase-noise channels: A tutorial review," in *Proc. of the 6th Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC)*, Baiona, Spain, Sep. 2012, pp. 316–327.

[Cov06] T. M. Cover, *Elements of Information Theory*, Second edition, J. A. Thomas, Ed. Hoboken, NJ: Wiley-Interscience, 2006.

[Csi84] I. Csiszár, "Information geometry and alternating minimization procedures," 1984.

[CTB98] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[CVD+15] T.-Y. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-Based Raptor-Like LDPC Codes," *IEEE Transactions on Communications*, vol. 63, no. 5, pp. 1522–1532, Jan. 1, 2015.

[CW14] T. A. Courtade and T. Weissman, "Multiterminal Source Coding Under Logarithmic Loss," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 740–761, Jan. 2014.

[CZD+14] F. Cai, X. Zhang, D. Declercq, S. K. Planjery, and B. Vasić, "Finite Alphabet Iterative Decoders for LDPC Codes: Optimization, Architecture and Analysis," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 5, pp. 1366–1375, May 2014.

[DCH+18] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep Learning Based Communication Over the Air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, Feb. 2018.

[DDJ+09] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[DF05] D. Declercq and M. Fossorier, "Extended minsum algorithm for decoding LDPC codes over GF(/sub q/," in *Proc. of the 2005 International Symposium on Information Theory (ISIT)*, Adelaide, SA, Australia, Sep. 2005, pp. 464–468.

[DF07] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over GF(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, Jan. 1, 2007.

[DHC+20] S. Dörner, M. Henninger, S. Cammerer, and S. ten Brink, "WGAN-based Autoencoder Training Over-the-air," in *Proc. of the 2020 IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Atlanta, GA, USA, May 2020, pp. 1–5.

[DS90] D. Divsalar and M. Simon, "Multiple-symbol differential detection of MPSK," *IEEE Transactions on Communications*, vol. 38, no. 3, pp. 300–308, Mar. 1990.

[DVP+13] D. Declercq, B. Vasic, S. K. Planjery, and E. Li, "Finite Alphabet Iterative Decoders—Part II: Towards Guaranteed Error Correction of LDPC Codes via Iterative Decoder Diversity," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4046–4057, Oct. 2013.

[FAB+16] T. Fehenberger, A. Alvarado, G. Böcherer, and N. Hanik, "On Probabilistic Shaping of Quadrature Amplitude Modulation for the Nonlinear Fiber Channel," *Journal of Lightwave Technology*, vol. 34, no. 21, pp. 5063–5073, Nov. 2016.

[FCD+18] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. Ten Brink, "OFDM-Autoencoder for End-to-End Learning of Communications Systems," in *Proc. of the 2018 IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, Greece, Jun. 2018, pp. 1–5.

[Gal62] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[GBM+18] R. Ghanaatian, A. Balatsoukas-Stimming, T. C. Muller, M. Meidlinger, G. Matz, A. Teman, and A. Burg, "A 588-Gb/s LDPC Decoder Based on Finite-Alphabet Message Passing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 329–340, Jan. 1, 2018.

[GNT03] R. Gilad-Bachrach, A. Navot, and N. Tishby, "An Information Theoretic Tradeoff between Complexity and Accuracy," G. Goos, J. Hartmanis, J. van Leeuwen, B. Schölkopf, and M. K. Warmuth, Eds., pp. 595–609, Jan. 1, 2003.

[Gol05] A. Goldsmith, *Wireless Communications*. Cambridge university press, 2005.

[GPM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of the Advances in Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 2672–2680.

[HCM19] X. He, K. Cai, and Z. Mei, "On Mutual Information-Maximizing Quantized Belief Propagation Decoding of LDPC Codes," in *Proc. of the 2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[HEA+01] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. of the 2001 IEEE Global Communications Conference (GLOBECOM)*, vol. 2, San Antonio, TX, USA, Nov. 2001, 1036–1036E vol.2.

[HJ12] T. Hazan and T. Jaakkola, "On the Partition Function and Random Maximum A-Posteriori Perturbations," in *Proc. of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12, Edinburgh, Scotland: Omnipress, 2012, pp. 1667–1674.

[HO07] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models," in *Proc. of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP*, Honolulu, HI, USA: IEEE, Jan. 1, 2007, pp. -317–320.

[HOP96] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[HT07] P. Harremoes and N. Tishby, "The Information Bottleneck Revisited or How to Choose a Good Distortion Measure," in *Proc. of the 2007 IEEE International Symposium on Information Theory*, Nice, France: IEEE, Jun. 2007, pp. 566–570.

[HWD17] S. Hassanpour, D. Wuebben, and A. Dekorsy, "Overview and Investigation of Algorithms for the Information Bottleneck Method," in *Proc. of the 11th International ITG Conference on Systems, Communications and Coding*, Hamburg, Germany: VDE, Jan. 1, 2017.

[IEE16] IEEE Std 802.11-2016, *IEEE Standard for Information technology —Telecommunications and information exchange between systems Local and metropolitan area networks —Specific requirements - Part 11*, Dec. 2016.

[IEE18] IEEE Std 802.16-2017, *IEEE Standard for Air Interface for Broadband Wireless Access Systems*, Mar. 2018.

[Jai08] A. K. Jain, "Data Clustering: 50 Years Beyond K-means," Lecture Notes in Computer Science, W. Daelemans, B. Goethals, and K. Morik, Eds., pp. 3–4, 2008.

[JDE+05] Jinghu Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and Xiao-Yu Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[JGP17] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017. arXiv: `1611.01144 [stat.ML]`.

[Joh10] S. J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*. Cambridge university press, 2010.

[KB14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. arXiv: `1412.6980`.

[KCT+18] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen, "End-to-End Deep Learning of Optical Fiber Communications," *Journal of Lightwave Technology*, vol. 36, no. 20, pp. 4843–4855, Oct. 15, 2018.

[KFL01] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Jan. 1, 2001.

[KHH+20] D. Korpi, M. Honkala, J. M. J. Huttunen, and V. Starck, "Deeprx mimo: Convolutional mimo detection with learned multiplicative transformations," 2020. arXiv: 2010.16283 [eess.SP].

[KHW18] C. Kestel, M. Herrmann, and N. Wehn, "When Channel Coding Hits the Implementation Wall," in *Proc. of the 2018 IEEE International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, Hong Kong, Hong Kong: IEEE, Dec. 2018, pp. 1–6.

[KK17] D. Kern and V. Kuehn, "On Information Bottleneck Graphs to design compress and forward quantizers with side information for multicarrier transmission," in *Proc. of the 2017 IEEE International Conference on Communications (ICC)*, Paris, France: IEEE, May 2017, pp. 1–6.

[Kur17] B. M. Kurkoski, "On the Relationship Between the KL Means Algorithm and the Information Bottleneck Method," in *Proc. of the 11th International ITG Conference on Systems, Communications and Coding SCC*, Hamburg, Germany: VDE, Feb. 2017, pp. 1–6.

[KWJ+16] C. Kadow, B. Wentzel, I. Jaekel, M. Bahlo, and U. Cubasch, "Berlin Climate Record - Daily Mean Temperature - Inner City - 1766-1934," in, PANGAEA, Jan. 1, 2016.

[KY14] B. M. Kurkoski and H. Yagi, "Quantization of Binary-Input Discrete Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4544–4552, Aug. 2014.

[KYK08] B. M. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Noise Thresholds for Discrete LDPC Decoding Mappings," in *Proc. of the 2008 IEEE Global Telecommunications Conference IEEE GLOBECOM*, New Orleans, LO, USA: IEEE, Nov. 2008, pp. 1–5.

[LB15] J. Lewandowsky and G. Bauch, "Trellis based node operations for LDPC decoders from the Information Bottleneck method," in *Proc. of the 2015 International Conference on Signal Processing and Communication Systems (ICSPCS)*, Cairns, Australia: IEEE, Dec. 2015, pp. 1–10.

[LB18] J. Lewandowsky and G. Bauch, "Information-Optimum LDPC Decoders Based on the Information Bottleneck Method," *IEEE Access*, vol. 6, pp. 4054–4071, 2018.

[LBT+18] J. Lewandowsky, G. Bauch, M. Tschauner, and P. Oppermann, "Design and Evaluation of Information Bottleneck LDPC Decoders for

Software Defined Radios," in *Proc. of the 2018 International Conference on Signal Processing and Communication Systems (ICSPCS)*, Cairns, Australia, Dec. 2018, pp. 1–9.

[LC01] S. Lin and D. J. Costello, *Error Control Coding*. Prentice hall, 2001, vol. 2.

[Llo82] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[LSB16a] J. Lewandowsky, M. Stark, and G. Bauch, "Information Bottleneck Graphs for receiver design," in *Proc. of the 2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain: IEEE, Jan. 1, 2016, pp. 2888–2892.

[LSB16b] J. Lewandowsky, M. Stark, and G. Bauch, "Optimum message mapping LDPC decoders derived from the sum-product algorithm," in *Proc. of the 2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia: IEEE, 2016, pp. 1–6.

[LSB17] J. Lewandowsky, M. Stark, and G. Bauch, "Message alignment for discrete LDPC decoders with quadrature amplitude modulation," in *Proc. of the 2017 IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany: IEEE, Jun. 2017, pp. 2925–2929.

[LSM+17] J. Lewandowsky, M. Stark, R. Mendrzik, and G. Bauch, "Discrete Channel Estimation by Integer Passing in Information Bottleneck Graphs," in *Proc. of the 11th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany: VDE, Jan. 1, 2017, pp. 1–6.

[LT05] J. K. Lee and J. Thorpe, "Memory-efficient decoding of LDPC codes," in *Proc. of the 2005 International Symposium on Information Theory (ISIT)*, Adelaide, SA, Australia: IEEE, Sep. 2005, pp. 459–463.

[Mac20] D. MacKay. (2020). Encyclopedia of Sparse Graph Codes, [Online]. Available: `http://www.inference.org.uk/mackay/codes/data.html` (visited on 10/15/2020).

[Mac99] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[MBB+15] M. Meidlinger, A. Balatsoukas-Stimming, A. Burg, and G. Matz, "Quantized message passing for LDPC codes," in *Proc. of the 49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA: IEEE, 2015, pp. 1606–1610.

[MCH19] Z. Mei, K. Cai, and X. He, "Deep learning-aided dynamic read thresholds design for multi-level-cell flash memories," 2019. arXiv: 1907.03938 [cs.IT].

[MDJ+19] S. Mohamed, J. Dong, A. R. Junejo, and D. C. Zuo, "Model-Based: End-to-End Molecular Communication System Through Deep Reinforcement Learning Auto Encoder," *IEEE Access*, vol. 7, pp. 70 279–70 286, 2019.

[Mit05] T. M. Mitchell, *Machine Learning*, International ed., [Nachdr.], ser. McGraw-Hill Series in Computer Science. McGraw-Hill, 2005.

[MM17] M. Meidlinger and G. Matz, "On irregular LDPC codes with quantized message passing decoding," in *Proc. of the 2017 IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sapporo, Japan: IEEE, Jul. 2017, pp. 1–5.

[MMB20] M. Meidlinger, G. Matz, and A. Burg, "Design and Decoding of Irregular LDPC Codes Based on Discrete Message Passing," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1329–1343, Mar. 2020.

[MZY+02] X. Ma, X. Zhang, H. Yu, and A. Kavcic, "Optimal quantization for soft-decision decoding revisited," in *Proc. of the 2002 IEEE International Symposium on Information Theory and its Applications (ISITA)*, Xi'an, China: IEEE, Jul. 2002, p. 3.

[NBB16] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. of the 2016 Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, USA, Sep. 2016, pp. 341–346.

[NML+17] T. Ninacs, B. Matuz, G. Liva, and G. Colavolpe, "Non-binary LDPC coded DPSK modulation for phase noise channels," in *Proc. of the 2017 IEEE International Conference on Communications (ICC)*, Paris, France: IEEE, May 2017, pp. 1–6.

[NML+19] T. Ninacs, B. Matuz, G. Liva, and G. Colavolpe, "Short Non-Binary Low-Density Parity-Check Codes for Phase Noise Channels," *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 4575–4584, Jul. 2019.

[OH17] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[ORW19] T. J. O'Shea, T. Roy, and N. West, "Approximating the Void: Learning Stochastic Channel Models from Observation with Variational Generative Adversarial Networks," in *Proc. of the 2019 IEEE International*

Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA: IEEE, Feb. 2019, pp. 681–686.

[PDD+11] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders," *IEEE Electronics Letters*, vol. 47, no. 16, pp. 919–921, Aug. 2011.

[PDD+13] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite Alphabet Iterative Decoders—Part I: Decoding Beyond Belief Propagation on the Binary Symmetric Channel," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4033–4045, Oct. 2013.

[PP02] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. Boston: McGraw-Hill, 2002, 852 pp.

[PR86] P. M. Pardalos and J. B. Rosen, "Methods for global concave minimization: A bibliographic survey," *Siam Review*, vol. 28, no. 3, pp. 367–379, 1986.

[RDW19] S. V. S. Ranganathan, D. Divsalar, and R. D. Wesel, "Quasi-Cyclic Protograph-Based Raptor-Like LDPC Codes for Short Block-Lengths," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3758–3777, Jun. 2019.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[RK15] F. J. C. Romero and B. M. Kurkoski, "Decoding LDPC codes with mutual information-maximizing lookup tables," in *Proc. of the 2015 IEEE International Symposium on Information Theory (ISIT)*, Hong Kong, China: IEEE, 2015, pp. 426–430.

[RK16] F. J. C. Romero and B. M. Kurkoski, "LDPC Decoding Mappings That Maximize Mutual Information," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2391–2401, Sep. 2016.

[RL09] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge: Cambridge University Press, 2009.

[Ros58] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[Ros61] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[RSU01] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[SAH19] M. Stark, F. A. Aoudia, and J. Hoydis, "Joint learning of geometric and probabilistic constellation shaping," in *Proc. of the 2019 IEEE Globecom Workshops (GC Wkshps)*, Waikoloa, HI, USA: IEEE, Dec. 2019, pp. 1–6.

[Sav08] V. Savin, "Min-Max decoding for non binary LDPC codes," in *Proc. of the 2008 IEEE International Symposium on Information Theory (ISIT)*, Piscataway, NJ: IEEE, Jan. 1, 2008, pp. 960–964.

[SB16] P. Schulte and G. Böcherer, "Constant Composition Distribution Matching," *IEEE Transactions on Information Theory*, vol. 62, no. 1, pp. 430–434, Jan. 2016.

[SBD+19] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124 020, 2019.

[SBL+19] M. Stark, G. Bauch, J. Lewandowsky, and S. Saha, "Decoding of Non-Binary LDPC Codes using the Information Bottleneck Method," in *Proc. of the 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China: IEEE, May 2019, pp. 1–6.

[SBM+19] F. Steiner, E. Ben Yacoub, B. Matuz, G. Liva, and A. G. i Amat, "One and Two Bit Message Passing for SC-LDPC Codes With Higher-Order Modulation," *Journal of Lightwave Technology*, vol. 37, no. 23, pp. 5914–5925, Dec. 2019.

[SBW+20] M. Stark, G. Bauch, L. Wang, and R. D. Wesel, "Information bottleneck decoding of rate-compatible 5g-ldpc codes," in *Proc. of the 2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland: IEEE, Jun. 2020, pp. 1–6.

[SDW17] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *Proc. of the 2017 IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Sapporo, Japan: IEEE, 2017, pp. 1–5.

[Sha48] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[Sha59] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Conv. Rec*, vol. 4, no. 142-163, p. 1, 1959.

[Sim18] O. Simeone, "A brief introduction to machine learning for engineers," 2018. arXiv: 1709.02840 [cs.LG].

[SK19] S. Steiner and V. Kuehn, "Optimization Of Distributed Quantizers Using An Alternating Information Bottleneck Approach," in *Proc. of the 2019 International ITG Workshop on Smart Antennas (WSA)*, Hamburg, Germany: VDE, Apr. 2019, pp. 1–6.

[SKW06] B. Smith, F. Kschischang, and Wei Yu, "Low-Density Parity-Check Codes for Discretized Min-Sum Decoding," in *Proc. of the 2006 Biennial Symposium on Communications*, Kingston, ON, Canada: IEEE, 2006, pp. 14–17.

[SLB18a] M. Stark, J. Lewandowsky, and G. Bauch, "Information-Bottleneck Decoding of High-Rate Irregular LDPC Codes for Optical Communication Using Message Alignment," *Applied Sciences*, vol. 8, no. 10, p. 1884, Jan. 1, 2018.

[SLB18b] M. Stark, J. Lewandowsky, and G. Bauch, "Information-Optimum LDPC Decoders with Message Alignment for Irregular Codes," in *Proc. of the 2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates: IEEE, Jan. 1, 2018, pp. 1–6.

[SLB18c] M. Stark, J. Lewandowsky, and G. Bauch, "Iterative Message Alignment for Quantized Message Passing between Distributed Sensor Nodes," in *Proc. of the 2018 IEEE Vehicular Technology Conference (IEEE VTC)*, Porto, Portugal: IEEE, Jan. 1, 2018, pp. 1–5.

[SLB19] M. Stark, J. Lewandowsky, and G. Bauch, "A Parametric Information Bottleneck Algorithm for Gaussian Random Variables and Gaussian Mixtures," in *Proc. of the 12th International ITG Conference on Systems, Communications and Coding (SCC)*, Rostock, Germany: VDE, Jan. 1, 2019.

[SLB20] M. Stark, J. Lewandowsky, and G. Bauch, "Neural Information Bottleneck Decoding," in *Proc. of the 2020 International Conference on Signal Processing and Communication Systems (ICSPCS)*, Adelaide, Australia: IEEE, Dec. 2020.

[Slo02] N. Slonim, "The Information Bottleneck Theory and Applications," Thesis, Jan. 1, 2002, 157 pp.

[SS17] D. J. Strouse and D. J. Schwab, "The Deterministic Information Bottleneck," *Neural computation*, vol. 29, no. 6, pp. 1611–1630, Jan. 1, 2017.

[SSB18] M. Stark, S. A. A. Shah, and G. Bauch, "Polar Code Construction using the Information Bottleneck Method," in *Proc. of the 2018*

IEEE Wireless Communications and Networking Conference Workshops (WCNCW): Polar Coding for Future Networks: Theory and Practice (IEEE WCNCW PCFN), Barcelona, Spain: IEEE, Jan. 1, 2018, pp. 7–12.

[SSB19] S. A. A. Shah, M. Stark, and G. Bauch, "Coarsely Quantized Decoding and Construction of Polar Codes Using the Information Bottleneck Method," *Algorithms*, vol. 12, no. 9, p. 192, Sep. 2019.

[ST00a] N. Slonim and N. Tishby, "Agglomerative information bottleneck," in *Proc. of the 2000 Advances in Neural Information Processing Systems*, Denver, CO, USA: MIT Press, 2000, pp. 617–623.

[ST00b] N. Slonim and N. Tishby, "Document Clustering Using Word Clusters via the Information Bottleneck Method," in *Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece: ACM, 2000, pp. 208–215.

[SWB+20] M. Stark, L. Wang, G. Bauch, and R. D. Wesel, "Decoding Rate-Compatible 5G-LDPC Codes With Coarse Quantization Using the Information Bottleneck Method," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 646–660, 2020.

[tBKA04] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[Tho03a] J. Thorpe, "Low-complexity approximations to belief propagation for LDPC codes," in *Proc. of the 2003 IEEE International Symposium on Information Theory (ISIT)*, Yokohama, Japan: IEEE, 2003.

[Tho03b] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *IPN progress report*, vol. 42, no. 154, pp. 42–154, 2003.

[TPB99] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in *Proc. of the 37th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, Sep. 1, 1999, pp. 368–377.

[TZ15] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proc. of the 2015 IEEE Information Theory Workshop (ITW)*, Jerusalem, Israel: IEEE, 2015, pp. 1–5.

[VDP08] A. Venkiah, D. Declercq, and C. Poulliat, "Design of cages with a randomized progressive edge-growth algorithm," *IEEE Communications Letters*, vol. 12, no. 4, pp. 301–303, Apr. 2008.

[Vit98] A. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE Journal on*

Selected Areas in Communications, vol. 16, no. 2, pp. 260–264, Feb. 1998.

[WCS+11] J. Wang, T. Courtade, H. Shankar, and R. D. Wesel, "Soft Information for LDPC Decoding in Flash: Mutual-Information Optimized Quantization," in *Proc. of the 2011 IEEE Global Telecommunications Conference (GLOBECOM)*, Houston, TX, USA: IEEE, Dec. 2011, pp. 1–6.

[WM14] A. Winkelbauer and G. Matz, "Rate-information-optimal Gaussian channel output compression," in *Proc. of the 2014 Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, USA: IEEE, Jan. 1, 2014, pp. 1–5.

[WSM04] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. of the 2004 IEEE International Conference on Communications (ICC)*, vol. 2, Paris, France: IEEE, Jun. 2004, 772–776 Vol.2.

[WSW+20] L. Wang, M. Stark, R. Wesel, and G. Bauch, "A Reconstruction-Computation-Quantization (RCQ) approach to node operations in LDPC decoding," in *Proc. of the 2020 IEEE Global Communications Conference: Communication Theory (GLOBECOM)*, Taipei, Taiwan: IEEE, Dec. 2020.

[WW75] H. Witsenhausen and A. Wyner, "A conditional entropy bound for a pair of discrete random variables," *IEEE Transactions on Information Theory*, vol. 21, no. 5, pp. 493–501, Sep. 1975.

[WZ71] A. Wyner and J. Ziv, "Bounds on the rate-distortion function for stationary sources with memory," *IEEE Transactions on Information Theory*, vol. 17, no. 5, pp. 508–513, Sep. 1971.

[WZ76] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, Jan. 1976.

[Yat20] R. D. Yates, "The Age of Information in Networks: Moments, Distributions, and Sampling," *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5712–5728, Sep. 2020.

[YLJ+18] H. Ye, G. Y. Li, B. F. Juang, and K. Sivanesan, "Channel Agnostic End-to-End Learning Based Communication Systems with Conditional GAN," in *Proc. of the 2018 IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates: IEEE, Dec. 2018, pp. 1–5.

[YLL+20] H. Ye, L. Liang, G. Y. Li, and B. Juang, "Deep Learning-Based End-to-End Wireless Communication Systems With Conditional GANs as Unknown Channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3133–3143, May 2020.

[YSM+19] E. B. Yacoub, F. Steiner, B. Matuz, and G. Liva, "Protograph-Based LDPC Code Design for Ternary Message Passing Decoding," in *Proc. of the 12th International ITG Conference on Systems, Communications and Coding (SCC)*, Rostock, Germany: VDE, Feb. 2019, pp. 1–6.

[ZES20] A. Zaidi, I. Estella-Aguerri, and S. Shamai (Shitz), "On the Information Bottleneck Problems: Models, Connections, Applications and Information Theoretic Views," *Entropy*, vol. 22, no. 2, p. 151, 2 Feb. 2020.

[ZK16] J. A. Zhang and B. M. Kurkoski, "Low-complexity quantization of discrete memoryless channels," in *Proc. of the 2016 International Symposium on Information Theory and Its Applications (ISITA)*, Monterey, CA, USA: IEEE, Oct. 2016, pp. 448–452.

[ZKB+08] G. Zeitler, R. Koetter, G. Bauch, and J. Widmer, "Design of network coding functions in multihop relay networks," in *Proc. of the 2008 International Symposium on Turbo Codes and Related Topics*, Lausanne, Switzerland: IEEE, Sep. 2008, pp. 249–254.

[ZSK12] G. Zeitler, A. C. Singer, and G. Kramer, "Low-Precision A/D Conversion for Maximum Information Rate in Channels with Memory," *IEEE Transactions on Communications*, vol. 60, no. 9, pp. 2511–2521, Sep. 2012.

[ZZC+19] Z. Zhu, J. Zhang, R. Chen, and H. Yu, "Autoencoder-Based Transceiver Design for OWC Systems in Log-Normal Fading Channel," *IEEE Photonics Journal*, vol. 11, no. 5, pp. 1–12, Oct. 2019.