

# Preface

There has been a strong resurgence of AI in recent years. One core technology of AI is statistical learning, which aims to automatically “program” machines with data. While the idea can date back to the '50s of the last century, the plethora of data and inexpensive computational power allow the techniques to thrive and penetrate every aspect of our daily lives - customer behavior prediction, financial market prediction, fully automatic surveillance, self-driving vehicles, autonomous robots, and beyond.

Information theory (IT) was first introduced and developed by the great communications engineer, Claude Shannon, in the '50s of the last century. The theory was introduced in an attempt to explain the principle behind point-to-point communication and data storage. However, the technique has been incorporated into statistical learning and has inspired many underlying principles. In this book, we will explore the exciting area of statistical learning, particularly probabilistic inference, from the view of IT. It facilitates students to understand the omnipresent field of statistical learning and appreciate the widespread significance of IT.

The book will start by providing an overview of IT and statistical learning. We will then aid students to establish a solid foundation on core IT principles, including information measures, typical sequences, and source and channel coding theories. We will then apply this foundation to better understand statistical learning concepts and techniques such as cross-entropy, evidence-lower bound, and decision trees, consequently offering a bridge between IT and statistical learning. We will wrap up with a concise chapter on probabilistic inference on graphical models. There we will explain how the decoding of the low-density parity-check (LDPC) codes works. This finishes a circle of connecting IT and statistical learning by providing an example of using statistical learning (probabilistic inference) to IT since the LDPC codes are one of the most important error-correcting codes used in channel coding. To maintain a succinct manuscript, we have to make a difficult choice to exclude materials (such as rate-distortion theory and the method of type) that are often included in a more conventional information theory textbook.

The book is aimed to be self-contained. It should be accessible by undergraduate students with solid calculus and some probability background. As a solid foundation on probability is essential for a deep understanding of IT, a review of probability is included as an early chapter. Some common misconceptions such as the difference between independence and conditional independence will be highlighted. Another often used tool in IT is constrained optimization, where a reader may not have a background in. Consequently, a section on Lagrange multiplier and Karush-Kuhn-Tucker (KKT) conditions is included in the appendix. Moreover, for many real-world problems with numerical

inputs, the results can be solved with a probabilistic programming package at ease. We will describe how some of the numerical problems can be solved with the Lea probabilistic programming package in the appendix. Despite that there are many other probabilistic programming packages, we choose Lea as it is probably the easiest to use.

### How to use this book


This book is suitable for a one-semester upper-level undergraduate or first-year graduate course on Information Theory (IT) and probabilistic inference. Due to the prerequisites of solid probability knowledge, a brief review of Chapter 2 is recommended, particularly to clarify common misconceptions about independence and conditional independence.

For a graduate course or students with a strong background in calculus and probability, covering the entire book in one semester is feasible. For a concise introductory course on IT, consider covering Chapters 1-6. For a short course on probabilistic inference, cover Chapters 2-7, skipping Chapter 6.

Instructors may want to review constrained optimization (Appendix C) in class, as it is used throughout the material and may be easier to understand through lectures. Students can likely learn Python and the Lea package (Appendix A) independently. The remaining appendices can be used as reference materials.

### Some Notes on Symbols and Notation

Throughout this book, all logarithmic operations will be base 2 unless otherwise noted. That is,  $\log x \triangleq \log_2 x$ . For natural logarithms, we will use the notation  $\ln$  instead.

I have placed  icons in the sidebar near text where readers should pay extra attention. This includes conventions and results that may be counterintuitive to students first learning the subject.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Shannon's information . . . . .	7
1.1.1	Amount of information revealed . . . . .	8
1.1.2	Entropy: average information of a random variable . . . . .	8
1.2	Limitation of information theory . . . . .	9
1.3	Summary . . . . .	10
<b>2</b>	<b>Probability review</b>	<b>11</b>
2.1	Probability models and random variables . . . . .	11
2.1.1	Measure theory and Lesbegue integral . . . . .	13
2.1.2	Union bound . . . . .	15
2.2	Expectation and summary statistics . . . . .	15
2.2.1	Expectation . . . . .	15
2.2.2	Summary statistics . . . . .	16
2.3	Joint probabilities and conditional probabilities . . . . .	17
2.3.1	Joint distributions and marginal distributions . . . . .	17
2.3.2	Conditional probability, Bayes' rule, and the chain rule . . . . .	18
2.4	Independence and conditional independence . . . . .	20
2.4.1	Independent variables . . . . .	20
2.4.2	Conditionally independent variables . . . . .	22
2.4.3	Independence but not conditional independence . . . . .	23
2.4.4	Conditional independence but not independence . . . . .	24
2.5	Markov chain . . . . .	24
2.6	Probabilistic inference . . . . .	25
2.6.1	ML vs MAP vs Bayesian inference . . . . .	25
2.6.2	Conjugate prior . . . . .	28
2.7	Exercises . . . . .	30
<b>3</b>	<b>Quantify information with compression</b>	<b>33</b>
3.1	Overview of entropy . . . . .	33
3.1.1	Limitation of entropy . . . . .	34

3.2	Source coding theory . . . . .	35
3.2.1	Source coding model . . . . .	35
3.2.2	A glimpse of source coding theorem . . . . .	36
3.3	Uniquely decodable code . . . . .	37
3.3.1	Prefix-free code . . . . .	37
3.4	Quantify information by minimizing expected codeword length . . . . .	37
3.4.1	Kraft's Inequality . . . . .	38
3.4.2	A proof of Source Coding Theorem . . . . .	40
3.5	Quantify entropy using LLN . . . . .	42
3.5.1	Law of Large Number (LLN) . . . . .	42
3.5.2	Asymptotic equipartition and typical sequences . . . . .	44
3.6	Quantify entropy by construction . . . . .	46
3.6.1	Shannon-Fano-Elias code . . . . .	46
3.6.2	A constructive proof of Source Coding Theorem . . . . .	48
3.7	Exercise . . . . .	49
<b>4</b>	<b>Information measures</b>	<b>51</b>
4.1	Entropy and differential entropy . . . . .	51
4.1.1	Revisiting the entropy . . . . .	51
4.1.2	Differential entropy . . . . .	52
4.1.3	Connection between differential entropy and entropy . . . . .	54
4.1.4	Bounds of entropy and differential entropy . . . . .	55
4.1.5	Joint entropy . . . . .	58
4.2	Conditional entropy . . . . .	59
4.2.1	Conditional entropy as an average of entropy over the conditioned random variable (r.v.) . . . . .	59
4.2.2	Conditional entropy and compression with side information . . . . .	60
4.2.3	Chain rule . . . . .	60
4.2.4	Converse proofs of source coding theorems . . . . .	62
4.3	KL-Divergence . . . . .	64
4.3.1	Some applications of Kullback-Leiber divergence (KL-divergence) . . . . .	65
4.4	Mutual Information . . . . .	72
4.4.1	Properties of Mutual Information . . . . .	73
4.4.2	Mutual information for continuous variables . . . . .	75
4.5	Venn diagram for information measures . . . . .	76
4.6	Summary . . . . .	77
4.7	Exercise . . . . .	77
<b>5</b>	<b>Interlude: Some IT application examples</b>	<b>79</b>
5.1	Shannon's Perfect Secrecy . . . . .	79
5.2	Identifying Vampires . . . . .	80
5.2.1	Picking good attributes based on counting . . . . .	81
5.2.2	Information theoretic approach . . . . .	82
5.2.3	Potential Extensions in Decision Tree Analysis . . . . .	84
5.3	Exercise . . . . .	84

<b>6</b>	<b>Channel coding</b>	<b>87</b>
6.1	What is channel coding? . . . . .	87
6.1.1	Channel Coding Components . . . . .	87
6.1.2	Channel Coding Rate and Channel Capacity . . . . .	88
6.1.3	Capacities of continuous channels . . . . .	88
6.2	Communication Channel Examples . . . . .	89
6.2.1	Binary symmetric channel . . . . .	90
6.2.2	Simple Gaussian Channel . . . . .	90
6.2.3	Additive White Gaussian Noise Channel . . . . .	91
6.2.4	Gaussian Colored Noise Channels . . . . .	91
6.3	Jointly typical sequences . . . . .	93
6.3.1	What are jointly typical sequences . . . . .	94
6.3.2	Packing and Covering Lemmas . . . . .	95
6.4	Forward proof of channel coding theorem . . . . .	97
6.4.1	Setup coding scheme . . . . .	98
6.4.2	Performance analysis . . . . .	98
6.5	Converse proof of channel coding theorem . . . . .	99
6.5.1	Fano's inequality . . . . .	100
6.5.2	Converse proof . . . . .	100
6.6	Summary . . . . .	101
6.7	Exercise . . . . .	101
<b>7</b>	<b>Graphical models</b>	<b>103</b>
7.1	Bayesian networks . . . . .	103
7.1.1	D-separation in Bayesian networks . . . . .	106
7.1.2	Burglar and raccoon . . . . .	107
7.2	Undirected graphs and factor graphs . . . . .	108
7.2.1	Factor graph representation . . . . .	109
7.2.2	The moralization of Bayesian networks . . . . .	110
7.2.3	Limitations of different graphical models . . . . .	110
7.3	BP algorithm . . . . .	112
7.3.1	Low-density parity-check code . . . . .	116
7.3.2	BP and statistical physics . . . . .	118
7.4	Gaussian BP . . . . .	123
7.4.1	Manipulating multivariate Gaussian . . . . .	124
7.4.2	Gaussian BP Update . . . . .	127
7.4.3	Kalman filter and BP . . . . .	129
<b>8</b>	<b>Score, Fisher information, Cramér-Rao lower bound, and score matching</b>	<b>133</b>
8.1	Overview of Fisher information and Cramér-Rao lower bound . . . . .	133
8.2	The score function . . . . .	134
8.3	Fisher Information . . . . .	134
8.4	Cramér-Rao Lower Bound . . . . .	135
8.5	Score matching . . . . .	137
8.5.1	When MLE fails: Energy-Based Models (EBMs) . . . . .	137
8.5.2	Score Matching for energy-based model (EBM) . . . . .	137

8.5.3	Example: Score Matching for Multivariate Gaussian Distribution . . . . .	139
8.6	Exercise . . . . .	141
<b>A</b>	<b>Using Lea to solve IT problems</b>	<b>143</b>
A.1	Installation . . . . .	143
A.2	Examples . . . . .	143
A.2.1	Burglar alarm . . . . .	144
A.2.2	A fair dice and a loaded dice . . . . .	145
A.2.3	Weather on a tropical island . . . . .	147
<b>B</b>	<b>Common distributions</b>	<b>151</b>
B.1	Normal distribution . . . . .	151
B.1.1	Multivariate normal distribution . . . . .	152
B.2	Bernoulli distribution . . . . .	153
B.3	Binomial distribution . . . . .	154
B.4	Beta Distribution . . . . .	155
B.5	Multinomial distribution . . . . .	157
B.6	Dirichlet distribution . . . . .	158
B.7	Exercise . . . . .	159
<b>C</b>	<b>Lagrange multiplier and Karush-Kuhn-Tucker (KKT) conditions</b>	<b>161</b>
C.1	Constrained optimization . . . . .	161
C.2	Lagrange multiplier . . . . .	161
C.2.1	Geometric intuition . . . . .	162
C.2.2	Algebraic proof . . . . .	163
C.2.3	Physical interpretation of Lagrange multiplier . . . . .	165
C.3	Karush-Kuhn-Tucker (KKT) conditions . . . . .	165
<b>D</b>	<b>Exponential family distributions and their conjugate priors</b>	<b>167</b>
D.1	Motivation . . . . .	167
D.1.1	Gaussian distribution as an exponential family distribution . . . . .	168
D.2	Cumulant generating function . . . . .	169
D.3	Conjugate priors of exponential family distribution . . . . .	170
D.3.1	Binomial distribution as an exponential family distribution . . . . .	171
D.3.2	Conjugate prior of unit variance Gaussian distribution . . . . .	172
D.4	Exercise . . . . .	173
	<b>Index</b>	<b>177</b>

## Chapter

## 1

# Introduction

We are flooded with information every day. If you believe in the simulation hypothesis, the world may be composed entirely of information. At the same time, “information” is such an overloaded word that everyone “knows” what it means, yet most will have trouble defining it when asked.

Information theory is a discipline that studies information through the lens of probability theory. It is one of the rare cases where the foundation of a subject was established by a single person. The father of information theory is Claude Shannon, who aimed to ignore the content of information and instead focus on how to quantify and transmit it effectively. His discoveries were published in the seminal paper “A Mathematical Theory of Communication,” in the Bell System Technical Journal in 1948.<sup>1</sup>

## 1.1 Shannon’s information

Rather than trying to precisely define what information means, Shannon focused on how to quantify information. The definition of information is rather vague, and any statement can be considered a piece of information. For example:

- Today’s high temperature is 25°C.
- Tomorrow’s average temperature will be higher than today.
- The sun is going to rise from the east tomorrow.
- John Kennedy was assassinated on November 22, 1963.

All of the above statements can be considered pieces of information. How do we quantify how much information each of them contains? This seems to be a rather tricky and subjective question. It may not be easy to even compare which statement has more information than another.

---

<sup>1</sup>Interestingly, the unit **bit**, which stands for binary digit and is commonly used in computer science and engineering, was also introduced in that paper.

### 1.1.1 Amount of information revealed

Shannon approached this problem using probability theory, treating every piece of news as a realization of a random variable. Let us explain this with a simple example: a coin toss. Given an unbiased coin, how much information is revealed if we toss the coin and get a head?

Since the unbiased coin can only have two equally likely outcomes, head (H) and tail (T), and representing two possible values requires one binary digit (bit), the amount of information obtained from knowing the coin toss result is head is 1 bit. counting bits

If we toss the coin three times and get heads for all three tosses, there are a total of 8 possible outcomes (HHH, HHT, HTH, HTT, THH, THT, TTH, TTT). We need  $\log_2 8 = 3$  bits to store one of the 8 outcomes. Therefore, the amount of information from revealing our toss outcomes (HHH) is 3 bits. From another viewpoint, each coin flip gives us 1 bit of information, so it is reasonable to get 3 bits from 3 coin tosses.

### 1.1.2 Entropy: average information of a random variable

For the coin examples, all outcomes are equally likely. But what if we have a biased coin, where the probabilities of heads and tails are different? Suppose the probability of heads is  $\frac{1}{4}$  and the probability of tails is  $\frac{3}{4}$ . Which outcome provides more information, heads or tails?

Intuitively, getting heads is more informative since it is less likely to happen. Referring back to an earlier statement, “The sun is going to rise from the east tomorrow” should contain no information since we know that it will always happen, unless the world is ending tomorrow. So, information that states something always true has no information. Conversely, if the statement is “The sun is going to rise from the **west** tomorrow” and indeed it happens, it contains an enormous amount of information. Therefore, knowing that a rare event has happened carries much more information than knowing an event that is likely to happen. rare events

Returning to the coin toss problem, how much information do we get if we know and get **count more** heads? Since the probability of getting heads is  $\frac{1}{4}$ , we might imagine that there are actually  $\frac{1}{\frac{1}{4}} = 4$  hypothetical outcomes that could have happened but didn't. Using the same argument as before, the amount of information is  $\log_2 4 = 2$  bits. In general, for an outcome with probability  $p$ , knowing that it happened will give us  $\log_2 \frac{1}{p} = -\log_2 p$  bits of information.

Now, how much average information do we gain from tossing the biased coin with a  $\frac{1}{4}$  probability of heads and  $\frac{3}{4}$  probability of tails? Since we get heads  $\frac{1}{4}$  of the time and tails  $\frac{3}{4}$  of the time, we gain  $-\log_2 \frac{1}{4}$  bits  $\frac{1}{4}$  of the time and  $-\log_2 \frac{3}{4}$  bits  $\frac{3}{4}$  of the time. The average information gain from knowing the outcome is

$$-\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4}.$$

In general, any discrete random variable  $X$  has on average entropy

$$-\sum_{x \in \mathcal{X}} p(x) \log_2 p(x)$$

bits of information. This quantity is called entropy by Shannon.<sup>2</sup> Entropy can be interpreted as the

---

<sup>2</sup>There is an interesting anecdote regarding the origin of the term “entropy.” Shannon originally wanted to name the expression “information measure.” However, after a discussion with John von Neumann, who pointed out that the term was already in use in statistical thermodynamics and would provide an edge in debates because of its



average knowledge gained from the realization of a probabilistic event. It also serves as a measure of the uncertainty or unpredictability associated with these events. We will provide a more formal argument for this expression as a way to quantify information in Chapter 3.

## 1.2 Limitation of information theory

As we see throughout this book, information theory is extremely powerful and is useful from communications, informatics, economics, and beyond. However, it has its limitations. We will try to summarize them in this short section.

**need knowledge  
of distribution**

First, information theory is based on probability and information measure such as entropy is usually defined out of the distribution of a random variable. So information measure may be computable only if the entire statistics of the variable that characterized by the distribution is known. For example, when only summary statistics such as means and variances are given, we cannot quantify the information in general unless further assumption such as the kind of the distribution of the variable is given.

**depend on  
observer**

Second, quantifying information can be subjective based on the observers as different observers could have different estimate of the distributions. For example, the statement “there are aliens in the universe” can have different amount of information based on different belief of the interpretable. One who strongly believes in alien existence will find the statement not informative and vice versa.

**can't go beyond  
probability**

Third, it is Shannon’s ingenious insight to frame information in the context of probability theory. It makes information theory extremely powerful tools for many disciplines, but at the same time information theory is essentially a subfield of applied probability and thus cannot explain things cannot be described with probability. For example, information theory is futile to quantify the amount of information of  $\pi$ . For a probabilistic view point, each digit of  $\pi$  appears to be completely random given earlier digit. At least, it is unclear how we can build a distribution of one digit given earlier digit. On the other hand, as we know from the Leibniz formula that

$$\pi = 4 \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots \right),$$

$\pi$  is deterministic and thus should contain **no** information under Shannon’s standpoint. Yet, saying it has no information ignores the amount of effort required to recover its value. A more useful terminology in this situation could be the Kolmogorov complexity, which quantifies information as the size of the smallest program that can describe it. While Kolmogorov complexity is conceptually elegant, it is almost impossible to use in practice. Therefore, we will not elaborate on it further beyond this point.

**ignore  
complexity**

Finally, information theory generally ignores the complexity of a solution. It is interested in theoretical limits such as the amount of information contained by a variable or the maximum amount of information that can be passed through a noisy channel. However, in the latter case, for example, it provides a theoretical bound on how much information can be passed through the channel but generally does not consider the amount of effort (complexity) required to achieve it.

---

ambiguous meaning, Shannon used “entropy” and “uncertainty” interchangeably in his writings.

### 1.3 Summary

We provided a concise, self-contained introduction to information theory in this chapter. We introduced Shannon's concept of information and argued in a hand-waving manner that entropy is a reasonable measure for quantifying information. We also briefly described the limitations of information theory. Since information theory is ultimately a subfield of applied probability, it requires a solid foundation in basic probability. In the next chapter, we will include a short review of probability from an applied standpoint. Readers who are familiar with probability may skip the next chapter and proceed directly to Chapter 3, where we will delve deeper into the expression of entropy from the perspective of information compression.

# Chapter 2

## Probability review

In this chapter, we offer a comprehensive and self-contained introduction to probability theory. Our coverage includes a review of fundamental concepts such as random variables, outcomes, events, sample spaces, probability distributions, expectation, and summary statistics. We also delve into the important topics of independence and conditional independence of variables. Additionally, we present an introduction to probability inference, encompassing maximum likelihood estimation, maximum a posteriori (MAP) estimation, and Bayesian inference. This chapter aims to equip readers with a solid understanding of probability theory and its applications in inference.

### 2.1 Probability models and random variables

**random variable, outcome, sample space** A probability model is used to describe random phenomenon that can have non-deterministic **outcomes**, where we call the set of all outcomes as the **sample space** and the “undetermine” object itself as a **random variable** (r.v.). If the sample space is continuous, the random variable is continuous. Otherwise, the random variable is discrete.

**probability** The **probability** of an outcome depicts the relative chance of getting that outcome. For a random variable  $X$ , we often denote the probability of  $X$  taking outcome  $a$  as  $Pr(X = a)$ . By convention, a probability is always non-negative and a probability of zero means that the outcome will never happen. On the other hand, a probability of one indicates that the outcome will certainly happen. So by definition, all possible outcomes should sum up to one as at least one of the outcome will certainly happen.

#### A word on convention

We often denote a r.v. using upper case (such as  $X$ ) and its **realization** (what was actually observed) using lower case (such as  $x$ ). Therefore,  $Pr(x = x_0)$  is a bad notation since  $x$  is not random and  $Pr(x = x_0) = 0$  in general unless  $x$  and  $x_0$  turn out to be identical.



**realization**

**Example 2.1: Coin toss**

Let's try to model a coin toss with a probability model. Let's denote the random variable as  $X$  and the outcomes **head** and **tail** as  $H$  and  $T$ , respectively. Then the sample space is  $\{H, T\}$ . The probability of the entire space should sum up to 1. Thus  $Pr(X = H) + Pr(X = T) = 1$ .

In the context of probability, an **event** is simply a set of outcomes and a subset of the sample space. For each possible event, we need a way to measure the probability of the event. So we need to have a probability function to map all possible events to a positive value since probability cannot be negative by definition. Since an event is just a set, for a discrete  $X$ , such probability measure can be intuitively constructed as

**probability mass function**

$$P(E) = \sum_{x \in E} Pr(X = x) = \sum_{x \in E} p_X(x), \quad (2.1)$$

where  $p_X(o)$  is often known as the **probability mass function**.

In principle, we can do the same thing a continuous  $X$  by just replacing summation to integration to get

$$P(E) \stackrel{(a)}{=} \lim_{\Delta x \rightarrow 0} \sum_{\substack{n \in \mathbb{Z} \\ n\Delta x \in E}} \underbrace{Pr(X \in [n\Delta x, (n+1)\Delta x])}_{p_X(n\Delta x)\Delta x} \stackrel{(b)}{=} \int_{x \in E} p_X(x) dx. \quad (2.2)$$

The expression led from (a) may look a bit obscure. But what we did is simply partition the entire real axis into segments of width  $\Delta x$ , and add the probability of the segment whenever the lower end of the segment is in  $E$ . This of course is an approximation for finite  $\Delta x$  but will be equal to  $P(E)$  as  $\Delta x$  shrinks to zero.

Another remark is that in general it is not meaningful to write  $Pr(X = x)$  when  $X$  is continuous. Because this value is likely to be zero. For example, for uniformly distributed  $X$  between 0 to 1, the probability of getting exactly  $X = 0.5$  will be zero. After all, even if we have  $X = 0.5 + 10^{-10}$ , it is still not equal to 0.5. Instead, it is reasonable to only ask the probability of  $X$  falling into a particular range such as  $Pr(X \in [x, x + \Delta x])$ . However, we can often define a **probability density function** roughly translate to how probable we get a sample  $X$  near the function argument. More precisely, we can define the pdf as

**probability density function**

$$p_X(x) = \lim_{\Delta x \rightarrow 0} \frac{Pr(X \in [x, x + \Delta x])}{\Delta x} \quad (2.3)$$

and that is how we get the expression under the curly bracket inside the expression after (a). With the pdf defined, we finally get the succinct expression after (b).

And note that from the definition above in (2.3), we have

$$Pr(a \leq X \leq b) = \int_a^b p_X(x) dx$$

where the integral is just the area underneath  $p_X(x)$  between  $a$  and  $b$ . Moreover, since  $X$  has to

take some value in the real axis,

$$Pr(-\infty \leq X \leq \infty) = \int_{-\infty}^{\infty} p_X(x) dx = 1 \quad (2.4)$$

Note that since we can interpret probability density function (PDF) just as a normalized probability function,  $p_X(x) \geq 0$  just as the original definition of the probability.

### A remark on notation

In probability textbook, it often use  $f_X(x)$  to denote the PDF of  $X$ . However, we will follow the classic text of Cover and Thomas and use  $p_X(x)$  directly to denote the PDF as there should be little confusion on notation. And we can free up the common symbol  $f(\cdot)$  for other places. We just need to remind ourselves that when  $X$  is continuous,  $p_X(x) \neq Pr(X = x)$ .

### Example 2.2: PDF of the uniform distribution

Let's take a r.v.  $X$  that is uniformly distributed between 0 and 1 as an example. The PDF is simply

$$p_X(x) = \begin{cases} 1, & 0 \leq x \leq 1, \\ 0, & \text{otherwise} \end{cases}$$

Note that the area underneath  $p_X(x)$  is 1 as expected.

## 2.1.1 Measure theory and Lebesgue integral

While the above discussion is intuitive, it is not applicable to all situations. Given a uniformly distributed  $X$  between 0 and 1, consider the question of asking the probability of  $X$  to be a rational number. We cannot answer the question as in (2.2) even we know the pdf, which is just 1 when  $0 \leq x \leq 1$ . We can try to write out the probability as in (2.2) as

$$P(E) = \int_{\substack{0 \leq x \leq 1 \\ x \in \mathbb{Q}}} dx, \quad (2.5)$$

where  $E = \{X \in \mathbb{Q}\}$  and  $\mathbb{Q}$  denotes the rational numbers. The question is, what should be the value of the integral?

The problem is more fundamental than probability but on how we interpret integration in calculus. From our first class in calculus, we don't actually have tools to find integral as in (2.5). The integration that we learned in our introductory calculus class, i.e., the Riemann integration, essentially sums the product of the function and an infinitely small partition of  $x$ -axis. The problem is that if the function or the region of integration are extremely discontinuous as in (2.5), then the Riemann integral will not be well-defined.

To resolve this, we need measure theory and extend Riemann integral to Lebesgue integral. Those are more advanced topics that will not be covered fully in this book. Luckily, we are not losing much in practice since the situation where we need to ask something like the earlier questions are rare in real-world setting. Having said that, let's also give a very high level explanation of measure



theory and Lebesgue integral through our earlier example. The idea to resolve the dilemma of the Riemann integral is that we define a measure  $\mu$  for all plausible measurable sets<sup>1</sup>. It sounds like **measure and measurable sets** an overarching task to create such a measure. However, there are some reasonable rules for the measure so that we can extend it without the need of defining it for all measurable sets.

$$\mu(\phi) = 0 \tag{2.6}$$

$$\mu(A \cup B) = \mu(A) + \mu(B) \text{ if } A \cap B = \phi \tag{2.7}$$

Given the measure  $\mu$ , let's consider a simple function  $f(x) = a \cdot I(x \in A) + b \cdot I(x \in B)$  such that  $A$  and  $B$  are measurable, the Lebesgue integral of  $f$  with measure  $\mu$  will simply be **Lebesgue integral**

$$\int f(x)d\mu = \int a \cdot I(x \in A) + b \cdot I(x \in B)d\mu \tag{2.8}$$

$$= a \cdot \mu(A) + b \cdot \mu(B). \tag{2.9}$$

The Lebesgue integral depends on the definition of the measure. And we should choose the measure such that the integral coincide with classic results from Riemann integral. Consequently, it is reasonable to have

$$\mu([x, y]) = y - x. \tag{2.10}$$

Now, back to the integration in (2.5), note that we have  $[0, 1] = ([0, 1] \cap \mathbb{Q}) \cup ([0, 1] \cap (\mathbb{R} \setminus \mathbb{Q}))$ , where  $[0, 1] \cap \mathbb{Q}$  is just the set of all rational numbers between 0 and 1 and  $[0, 1] \cap (\mathbb{R} \setminus \mathbb{Q})$  is the set of all irrational numbers between the same range. From (2.7) and (2.10), we have  $1 = \mu([0, 1]) = \mu([0, 1] \cap \mathbb{Q}) + \mu([0, 1] \cap (\mathbb{R} \setminus \mathbb{Q}))$ . Since there are infinitely many irrational number comparing to rational number, we should have  $\mu([0, 1] \cap \mathbb{Q}) = 0$  and  $\mu([0, 1] \cap (\mathbb{R} \setminus \mathbb{Q})) = 1$ . Consequently,

$$P(E) = \int_{\{x|x \in [0,1] \cap \mathbb{Q}\}} dx = \mu([0, 1] \cap \mathbb{Q}) = 0 \tag{2.11}$$

as one would expect.

When applying measure theory to probability theory, the approach is analogous to the Lebesgue integral example described earlier. The primary difference lies in the probability measure having more flexibility, without being constrained by (2.10), as we aimed to reconcile the results of Lebesgue and Riemann integrals in the previous example. However, we require the probability measure **probability measure**  $p(A) \geq 0$  for all  $A$ , ensuring that probability values are always non-negative.

---

<sup>1</sup>Formally, all measurable set forms a so-called  $\sigma$ -algebra, which satisfies the following: the empty set is measurable, and countably unions and intersections are measurable as well.

### 2.1.2 Union bound

Let's recap the three fundamental conditions of a probability measure  $p$  introduced in the previous section:

$$p(A) \geq 0, \quad \text{for any event } A \quad (2.12)$$

$$p(\phi) = 0 \quad (2.13)$$

$$p(A \cup B) = p(A) + p(B), \quad \text{if } A \cap B = \phi \quad (2.14)$$

The first condition ensures that probability values are non-negative. The second and third conditions are required because a probability measure is also a Lebesgue measure, as discussed earlier.

Notably, the third condition can be derived from the elementary probability definition without using measure theory. For instance, when  $A$  and  $B$  are disjoint events, we have

$$p(A \cup B) = \sum_{x \in A \cup B} p(x) = \sum_{x \in A} p(x) + \sum_{x \in B} p(x) = p(A) + p(B) \quad (2.15)$$

and

$$p(A \cup B) = \int_{x \in A \cup B} p(x) dx = \int_{x \in A} p(x) dx + \int_{x \in B} p(x) dx = p(A) + p(B) \quad (2.16)$$

for both discrete and continuous cases.

If  $A$  is a superset of  $B$ , we can write  $A = B \cup (A \setminus B)$  and note that  $B \cap (A \setminus B) = \phi$ . Therefore, we have

$$p(A) = p(B) + p(A \setminus B) \geq p(B), \quad (2.17)$$

which probably is expected from the readers.

A useful extension of this result is the union bound, which states that

$$p(A \cup B) \leq p(A) + p(B). \quad (2.18)$$

The proof is straightforward and is left as an exercise.

## 2.2 Expectation and summary statistics

Expectation is a fundamental concept in probability theory, serving as a crucial building block for defining essential summary statistics, including the mean and variance. These statistics provide valuable insights into the central tendency and dispersion of random variables, enabling us to better understand and analyze probability distributions.

### 2.2.1 Expectation

Consider a r.v.  $X$  and a deterministic function  $g(\cdot)$ .  $g(X)$  is a r.v. as well. Each sample of  $g(X)$  will be different. However, as we sample  $g(X)$  multiple times, the empirical average will converge

as we obtain more and more samples. The converged value is called the expectation of  $g(X)$ , and it is denoted by  $E[g(X)]$ . Mathematically,

**expectation**

$$E[g(X)] = \frac{1}{N} \lim_{N \rightarrow \infty} (g(x_1) + g(x_2) + \cdots + g(x_N)), \quad (2.19)$$

where  $x_i$  is the  $i$ th sample of  $X$ .

For discrete r.v.s, for any outcome  $x$  from the sample space  $\mathcal{X}$ , there will be a fraction  $p(x)$  of time that  $x$  occurs. Therefore, we can rewrite (2.19) as

$$E[g(X)] = \sum_{x \in \mathcal{X}} p_X(x)g(x). \quad (2.20)$$

For continuous r.v.s, note that the PDF does not equal the probability explicitly, as explained in the last section. Consequently, the expression becomes an integral instead. That is,

$$E[g(X)] = \int p_X(x)g(x) dx, \quad (2.21)$$

since

$$E[g(X)] = \lim_{\Delta \rightarrow 0^+} \sum_{n=-\infty}^{\infty} \underbrace{[p_X(n\Delta)\Delta]}_{\Pr(n\Delta \leq X \leq (n+1)\Delta)} g(n\Delta), \quad (2.22)$$

which is just the definition of  $\int p_X(x)g(x) dx$ .

### Expectation is linear

One of the most important properties of expectation is that  $E[\cdot]$  as an operation is linear. This means that for any two r.v.s  $X$  and  $Y$  and two constants  $a$  and  $b$ , we have

$$E[aX + bY] = aE[X] + bE[Y]. \quad (2.23)$$

The above result can be readily verified because, as we see from (2.20) and (2.21), the definitions of expectation for both discrete and continuous r.v.s just involve either a sum or an integral, and both of these operations are linear. Similarly, for any r.v.  $X$  and constant  $C$ ,

$$E[X + C] = E[X] + C. \quad (2.24)$$

We will leave the proofs of the above results as exercises.

### 2.2.2 Summary statistics

With a different function  $g(\cdot)$ , we can compute  $E[g(X)]$  as a summary description of the distribution  $p_X(\cdot)$ . Such a description is known as a **summary statistic**. The most common summary statistics are the **mean** and the **variance**.



**Mean**

The mean of a r.v.  $X$  is simply the expected value of  $X$  itself, denoted by  $E[X]$ . Since we are taking the expectation of  $X$  directly, we expect that the variable  $X$  is numerical rather than categorical. For example, it wouldn't make much sense to compute the mean of a coin toss unless we map the outcomes of heads and tails to some numerical values.

**mean**

From (2.19), we see that the empirical average of samples of  $X$  should converge to the mean. That is, given samples  $x_1, x_2, \dots, x_N$ ,

$$\frac{1}{N}(x_1 + x_2 + \dots + x_N) \rightarrow E[X] \quad (2.25)$$

as  $N$  goes to infinity.

**Variance**

The variance of  $X$  describes how much  $X$  fluctuates from its mean. It is defined as  $E[(X - \bar{X})^2]$ , where  $\bar{X} \triangleq E[X]$  is the mean of  $X$ . Note that the mean  $\bar{X}$  is a constant despite  $X$  being a r.v.. Expanding  $E[(X - \bar{X})^2]$ , we have

**variance**

$$E[(X - \bar{X})^2] = E[X^2 - 2\bar{X}X + \bar{X}^2] \quad (2.26)$$

$$\stackrel{(a)}{=} E[X^2] - 2\bar{X}E[X] + \bar{X}^2 \quad (2.27)$$

$$= E[X^2] - 2\bar{X}\bar{X} + \bar{X}^2 \quad (2.28)$$

$$= E[X^2] - \bar{X}^2, \quad (2.29)$$

where it is often more convenient to compute the variance of  $X$  using the last expression, and (a) comes from the linear property of expectation.

**2.3 Joint probabilities and conditional probabilities**

Up to now we only consider a single scalar r.v. at a time. Let's consider multiple r.v.s and how they interact with one another in this section.

**2.3.1 Joint distributions and marginal distributions****joint  
distribution  
marginalization**

Given two discrete r.v.s  $X$  and  $Y$ , the joint probability mass function (PMF)  $p_{X,Y}(x, y) \triangleq \Pr(X = x, Y = y)$  provides all the statistical information with respect to  $X$  and  $Y$ . Moreover, we can compute the probability of one variable regardless of the value of the other. For example,

$$p_X(x) = \sum_{y \in \mathcal{Y}} p_{X,Y}(x, y). \quad (2.30)$$

The above procedure of summing out all the dummy variables from the joint probability is known as marginalization, and the resulting probability  $p_X(x)$  is known as a marginal distribution.

For continuous variables, the marginalization step is similar, except the summation is replaced by an integral. For example, for continuous r.v.s  $X$  and  $Y$ ,

$$p_X(x) = \int p_{X,Y}(x, y) dy. \quad (2.31)$$

### Example 2.3: Weather forecast

Let's denote  $P$  and  $W$  as the predicted weather and actual weather tomorrow, where both variables can take the outcomes of *sunny* or *rainy*. Assume the joint probability  $p_{P,W}(\cdot, \cdot)$  is tabulated as below:

$P \backslash W$	Sunny	Rainy
Sunny	0.6	0.06
Rainy	0.04	0.3

As a sanity check, first note that the joint probability should sum up to one ( $0.6 + 0.06 + 0.04 + 0.3 = 1$ ). The probability of sunny weather tomorrow is

$$p_{P,W}(\text{sunny}, \text{sunny}) + p_{P,W}(\text{rainy}, \text{sunny}) = 0.6 + 0.04 = 0.64,$$

and the probability of rainy weather tomorrow is

$$p_{P,W}(\text{sunny}, \text{rainy}) + p_{P,W}(\text{rainy}, \text{rainy}) = 0.06 + 0.3 = 0.36,$$

which, of course, is just equal to 1 minus the probability of sunny weather tomorrow:  $1 - 0.64 = 0.36$ .

### 2.3.2 Conditional probability, Bayes' rule, and the chain rule

The joint probability gives us the probability of all variables with the desired outcomes. For example,  $p_{P,W}(\text{sunny}, \text{sunny})$  in the example of the last subsection gives us the probability that both the predicted and actual weather will be sunny tomorrow. Often, we are interested in finding the probability when some variables are already fixed and known. For example, if we have already predicted that the weather will be sunny tomorrow, what is the probability that the actual weather will also be sunny?

Since only  $p_{P,W}(\text{sunny}, \text{sunny})$  and  $p_{P,W}(\text{sunny}, \text{rainy})$  correspond to a sunny prediction, and among them, we are interested in the case where the actual weather is also sunny, the probability should be

$$\frac{p_{P,W}(\text{sunny}, \text{sunny})}{p_{P,W}(\text{sunny}, \text{sunny}) + p_{P,W}(\text{sunny}, \text{rainy})}, \quad (2.32)$$

**conditional  
probability**

which is known as the **conditional probability** of the weather being sunny given that the prediction is sunny, and is often denoted as  $p_{W|P}(\text{sunny}|\text{sunny})$ .

Note that by the marginalization rule described in the last subsection, the denominator in (2.32)

is just  $p_P(\text{sunny})$ , and so

$$p_{W|P}(\text{sunny}|\text{sunny}) = \frac{p_{P,W}(\text{sunny}, \text{sunny})}{p_P(\text{sunny})}. \quad (2.33)$$

In general, we have

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x, y)}{p_Y(y)}. \quad (2.34)$$

Therefore, we have  $p_{X,Y}(x, y) = p_Y(y)p_{X|Y}(x|y)$ . By the same token,  $p_{X,Y}(x, y) = p_X(x)p_{Y|X}(y|x)$ .

**Bayes' rule**

Thus,  $p_X(x)p_{Y|X}(y|x) = p_Y(y)p_{X|Y}(x|y)$  and  $p_{Y|X}(y|x) = \frac{p_Y(y)p_{X|Y}(x|y)}{p_X(x)}$ , or simply

$$p(y|x) = \frac{p(y)p(x|y)}{p(x)}. \quad (2.35)$$

This last expression is the famous Bayes' rule, but it is just a straightforward result of (2.34).

Despite the fame of Bayes' rule, the following chain rule is much more useful and general in practice. Note that we can rewrite (2.34) as

$$p(x, y) = p(y)p(x|y), \quad (2.36)$$

and this can be generalized to the case even when the right-hand side is conditioned. For example,

$$p(x, y|z) \stackrel{(a)}{=} \frac{p(x, y, z)}{p(z)} \quad (2.37)$$

$$\stackrel{(b)}{=} \frac{p(x, z)p(y|x, z)}{p(z)} \quad (2.38)$$

$$= \frac{p(x, z)}{p(z)}p(y|x, z) \quad (2.39)$$

$$\stackrel{(c)}{=} p(x|z)p(y|x, z), \quad (2.40)$$

where (a) is from (2.34) treating  $(x, y)$  as a single variable, (b) is from (2.36) treating  $(x, z)$  as a single variable, and (c) is from (2.34) again with  $y$  replaced by  $z$ .

**chain rule**

Combining (2.36) and (2.40), we have

$$\begin{aligned} p(x_1, x_2, \dots, x_N) &= p(x_1)p(x_2, \dots, x_N|x_1) \\ &= p(x_1)p(x_2|x_1)p(x_3, \dots, x_N|x_1, x_2) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4, \dots, x_N|x_1, x_2, x_3) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_N|x_1, \dots, x_{N-1}), \end{aligned}$$

where we will simply refer to as the chain rule.

### A shorthand notation

It is a good place to introduce another shorthand used throughout this book. For a list of variables,  $x_k, x_{k+1}, x_{k+2}, \dots, x_N$ , we may shorthand them as  $x_k^N$ . And when  $k = 1$ , we may shorthand it further to  $x^N$ . For example, we can rewrite the above chain rule as

$$p(x^N) = p(x_1)p(x_2|x_1)p(x_3|x^2) \cdots p(x_N|x^{N-1}). \quad (2.41)$$



## 2.4 Independence and conditional independence

As the name suggests, we say two r.v.s are independent if they have no effect on one another. For example, the outcomes of tossing two different dice should be independent. On the other hand, the forecast variable  $P$  should depend on the actual weather  $W$  in our earlier example.

Conditional independence is a similar concept, but it considers whether two variables may have an effect on one another given that some third variable is known. It may be surprising to those who first encounter these concepts, but independence and conditional independence are “independent” concepts. One property does not imply the other and vice versa.

### 2.4.1 Independent variables

Consider the joint probability of two r.v.s  $X$  and  $Y$ . Given a value  $x$ , we can consider the conditional probability  $p_{Y|X}(y|x)$  as a function of  $y$  parameterized by  $x$ . If  $p_{Y|X}(y|x_1) = p_{Y|X}(y|x_2)$  for all  $x_1, x_2 \in \mathcal{X}$ , then  $X$  and  $Y$  must be independent because the conditional distribution does not change regardless of the value  $X$  takes. Moreover, if  $X$  and  $Y$  are independent, then

**independence**

$$\begin{aligned} p_Y(y) &= \sum_{x \in \mathcal{X}} p_{X,Y}(x,y) = \sum_{x \in \mathcal{X}} p_X(x)p_{Y|X}(y|x) \\ &\stackrel{(a)}{=} p_{Y|X}(y|x) \sum_{x \in \mathcal{X}} p_X(x) \\ &\stackrel{(b)}{=} p_{Y|X}(y|x), \end{aligned} \quad (2.42)$$

where (a) holds because  $p_{Y|X}(y|x)$  is the same for all  $x$  and (b) follows from  $\sum_{x \in \mathcal{X}} p_X(x) = 1$ .

Furthermore, we have

$$p_{X,Y}(x,y) \stackrel{(a)}{=} p_X(x)p_{Y|X}(y|x) \stackrel{(b)}{=} p_X(x)p_Y(y), \quad (2.43)$$

where (a) is from the chain rule and (b) is from (2.42). Note that (2.43) is usually used as the formal “definition” of independence in most probability textbooks. However, I believe that (2.42) is more natural and easier to understand.

**Example 2.4: Tossing two coins**

Let's denote  $X_1$  and  $X_2$  as the outcomes of tossing two coins. Let's also assume that the probabilities of getting a head for  $X_1$  and  $X_2$  are  $p_1$  and  $p_2$ , respectively. Unless the coins interact in some mysterious way, it is safe to assume that their outcomes are independent. The probability of getting both heads will be  $p_1 \cdot p_2$ . We can tabulate the joint probability as shown below.

$X_1 \backslash X_2$	Head	Tail
Head	$p_1 \cdot p_2$	$p_1(1 - p_2)$
Tail	$(1 - p_1)p_2$	$(1 - p_1)(1 - p_2)$

Let's also try to tabulate the conditional probability distribution  $p_{X_2|X_1}$ . Since  $p_{X_2|X_1}(x_2|x_1) = \frac{p(x_1, x_2)}{p(x_1)}$ , we can create a table of  $p_{X_2|X_1}$  by simply dividing each row of the above table by the respective  $p(x_1)$ . That is,  $p_1$  for the first row, and  $1 - p_1$  for the second row. And so we get  $p_{X_2|X_1}$  given by

$X_1 \backslash X_2$	Head	Tail
Head	$p_2$	$1 - p_2$
Tail	$p_2$	$1 - p_2$

Note that each row of the above table is the same. That means that  $p_{X_2|X_1}(\cdot|x_1)$  does not change with different  $x_1$ . Therefore,  $X_1$  and  $X_2$  are indeed independent.

**Example 2.5: Weather forecast (con't)**

Let's continue with the example from Section 2.3. Recall that the joint probability  $p_{P,W}(\cdot, \cdot)$  is tabulated as below

$P \backslash W$	Sunny	Rainy
Sunny	0.6	0.06
Rainy	0.04	0.3

Let's try to tabulate the conditional probability  $p_{W|P}$  instead. Just as in the coin toss example earlier, we should divide each row of the above table by the respective marginal probabilities ( $0.6 + 0.06 = 0.66$  and  $0.04 + 0.3 = 0.34$ ). Therefore, we have:

$P \backslash W$	Sunny	Rainy
Sunny	0.91	0.09
Rainy	0.12	0.88

Note that the two rows are very different, meaning that  $p_{W|P}(\cdot|p)$  changes significantly with different  $p$ . Therefore,  $W$  and  $P$  must depend on one another.

### Notation for independence

Note that we often denote  $X \perp Y$  when  $X$  and  $Y$  are independent, i.e., when (2.42) and (2.43) are satisfied.



## 2.4.2 Conditionally independent variables

Now, let us consider three variables  $X$ ,  $Y$ , and  $Z$ . From now on, we simplify the notation by removing the subscript of  $p$ . For example,  $p(x|y, z) \equiv p_{X|Y,Z}(x|y, z)$ . We say that  $X$  and  $Y$  are conditionally independent given  $Z$  if

$$p(x|y_1, z) = p(x|y_2, z) \quad (2.44)$$

**conditional independence**

for any  $z$ ,  $y_1$ , and  $y_2$ .

The condition should be self-evident. It states that given a fixed  $z$ , the conditional distribution of  $X$  given  $Y$  and  $z$  does not depend on the choice of  $Y$ . So, given  $z$ ,  $X$  and  $Y$  will be independent.

Moreover, we have

$$p(x|z) = \sum_{y \in \mathcal{Y}} p(x, y|z) \quad (2.45)$$

$$\stackrel{(a)}{=} \sum_{y \in \mathcal{Y}} p(y|z)p(x|y, z) \quad (2.46)$$

$$\stackrel{(b)}{=} p(x|y, z) \sum_{y \in \mathcal{Y}} p(y|z) \quad (2.47)$$

$$\stackrel{(c)}{=} p(x|y, z), \quad (2.48)$$

where (a) comes from the chain rule (cf. (2.41)), (b) is from (2.44), and (c) is due to the probabilities summing up to 1.

Note that the conditional joint probability  $p(x, y|z)$  can now be expanded as

$$p(x, y|z) = p(y|z)p(x|y, z) = p(y|z)p(x|z), \quad (2.49)$$

where the last equality is due to (2.48). Like (2.43), (2.49) is often written as the “definition” of conditional independence in many probability textbooks, even though (2.44) is more natural and easier to interpret and understand.

### Example 2.6: Naïve Bayes classifier

Naïve Bayes is a simple machine learning algorithm used to classify an object based on some given features. A major assumption of Naïve Bayes is that the features are conditionally independent given the object class. Suppose  $O$  denotes the object and  $c(O)$  denotes its corresponding class. Let  $f_1(O), f_2(O), \dots, f_K(O)$  denote the  $K$  features of the object. For simplicity, let's rewrite  $c(O)$  as  $C$  and  $f_i(O)$  as  $F_i$ . It is important to realize that the “randomness” of  $c(O)$  and  $f_i(O)$  originates from  $O$ .

**Naïve Bayes classifier**

$$\begin{aligned}
 p(c|f_1, \dots, f_K) &= \frac{p(c, f_1, \dots, f_K)}{p(f_1, \dots, f_K)} \\
 &= \frac{p(c)p(f_1, \dots, f_K|c)}{p(f_1, \dots, f_K)} \\
 &= \frac{p(c)p(f_1|c) \cdots p(f_K|c)}{p(f_1, \dots, f_K)} \\
 &\propto p(c)p(f_1|c) \cdots p(f_K|c)
 \end{aligned}$$

To classify an object, we compute the product  $p(c)p(f_1|c) \cdots p(f_K|c)$  for each class  $c$ , and the output class should be the one with the maximum value.

### Notation for conditional independence

To conclude this section, we would like to note that when  $X$  and  $Y$  are conditionally independent given  $Z$ , this is often denoted by  $X \perp Y|Z$ . This notation indicates that (2.48) and (2.49) are satisfied.

### 2.4.3 Independence but not conditional independence

As we mentioned at the beginning of this section, a common mistake beginners make is to assume that independence implies conditional independence or vice versa. It turns out that the two properties are entirely “independent.”

Let’s map the outcomes of two coin tosses to zero (tail) and one (head) and denote them as  $X_1$  and  $X_2$ . Without any magical correlation,  $X_1$  and  $X_2$  should be independent. Let’s say the probability of heads for both  $X_1$  and  $X_2$  is  $p$ . The joint probability is given by:

	$X_2$	1	0
$X_1$			
1		$p^2$	$p(1-p)$
0		$p(1-p)$	$(1-p)^2$

One can verify that  $p(x_1, x_2) = p(x_1)p(x_2)$  for all combinations above, satisfying the independence condition given by (2.43).

Now, let’s define  $Y = X_1 \oplus X_2$ , where  $\oplus$  is the exclusive-or operator. Note that while  $X_1 \perp X_2$ ,  $X_1 \perp X_2|Y$  does not hold. In fact, note that  $Y = X_1 \oplus X_2$  implies  $X_2 = X_1 \oplus Y$ , so given  $Y$ , we can compute  $X_2$  deterministically from  $X_1$ . There is no way  $X_1$  and  $X_2$  are independent given  $Y$ .

To gain further insight, let’s tabulate the conditional distributions  $p_{X_2|X_1, Y}(\cdot|x_1, y)$  below

$Y = 0$

	$X_2$	1	0
$X_1$			
1		1	0
0		0	1

$Y = 1$

	$X_2$	1	0
$X_1$			
1		0	1
0		1	0



**independence  
but not  
conditional  
independence**

For  $X_1 \perp X_2 | Y$ , the rows in each table should be identical ( $p(x_2|x_1, y) = p(x_2|y)$ ). The rows being so different suggests that  $X_1$  and  $X_2$  are highly correlated given  $Y$ .

### 2.4.4 Conditional independence but not independence

Let's consider two noisy observations  $Y_1$  and  $Y_2$  of a r.v.  $X$ . For simplicity, let's assume all three **conditional** variables are binary. The noises  $Z_1 = X \oplus Y_1$  and  $Z_2 = X \oplus Y_2$  are independently generated from **independence** a binary symmetric source with a probability  $p$  of being 1. Let's also assume that the probability **but not** of  $X = 1$  is  $q$ . **independence**

Since  $Y_1$  and  $Y_2$  are independent observations of  $X$ , we would expect that they will be independent given  $X$ . On the other hand, it is reasonable to assume that  $Y_1$  and  $Y_2$  won't be independent (actually, they should be very correlated). Let's first show  $Y_1 \perp Y_2 | X$ .

Let's tabulate the joint probability  $p(y_1, y_2, x)$  as below

		$X = 0$		$X = 1$	
		$Y_2$		$Y_2$	
$Y_1$		1	0	1	0
1		$(1-q)p^2$	$(1-q)(1-p)p$	$q(1-p)^2$	$qp(1-p)$
0		$(1-q)(1-p)p$	$(1-q)(1-p)^2$	$qp(1-p)$	$qp^2$

From the tables above, let's tabulate the conditional probability  $p(y_2|y_1, x) = \frac{p(y_2, y_1, x)}{p(y_1, x)}$  below as

		$X = 0$		$X = 1$	
		$Y_2$		$Y_2$	
$Y_1$		1	0	1	0
1		$p$	$1-p$	$1-p$	$p$
0		$p$	$1-p$	$1-p$	$p$

Note that both rows in each table are the same. This means that  $p(y_2|y_1, x) = p(y_2|x)$ , and thus  $Y_1 \perp Y_2 | X$ .

On the other hand, let's tabulate the joint probability  $p(y_1, y_2)$  as follows:

		$Y_2$	
		1	0
$Y_1$			
1		$(1-q)p^2 + q(1-p)^2$	$(1-q)(1-p)p + qp(1-p)$
0		$(1-q)(1-p)p + qp(1-p)$	$(1-q)(1-p)^2 + qp^2$

It can be readily verified that, in general, we won't have  $p(y_1, y_2) = p(y_1)p(y_2)$ . Therefore,  $Y_1$  and  $Y_2$  are not independent.

## 2.5 Markov chain

Many sequential random variables have relatively local influence on each other. For example, if we consider the price of a stock each day as a sequence of r.v.s, the stock price today is probably more correlated with the price yesterday than the price last month. To an extreme, we may assume that



all historical information regarding today's price is summarized completely by yesterday's price. Even though it is not entirely true, it would be a good approximation to start with, and we say these price variables form a Markov chain.

**Markov chain**

Mathematically, let  $X_1, \dots, X_N$  be the sequence of price variables. We say the variables form a Markov chain if, for any  $k$  and  $l$  ( $l < k - 1$ ),  $X_k$  is conditionally independent of  $X_l$  given  $X_{k-1}$ . We often denote the chain by  $X_1 \leftrightarrow X_2 \leftrightarrow \dots \leftrightarrow X_N$ . Some textbooks also use a one-directional arrow for the notation. However, we prefer to use a double-sided arrow to indicate that the definition is indeed symmetric. That is, if we have a chain  $X_1 \leftrightarrow X_2 \leftrightarrow \dots \leftrightarrow X_N$ , we have  $X_N \leftrightarrow X_{N-1} \leftrightarrow \dots \leftrightarrow X_1$ . Note that the Markov property implies that we can express the joint probability as

$$\begin{aligned} p(x^N) &= p(x_1)p(x_2|x_1)p(x_3|x_2) \cdots p(x_N|x_{N-1}) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_2) \cdots p(x_N|x_{N-1}). \end{aligned} \quad (2.50)$$

## 2.6 Probabilistic inference

One of the most common problems we encounter in probability is estimating some latent variables based on observations. These latent variables, in turn, will depend on the model we choose, which is often determined by some model parameters.

### 2.6.1 ML vs MAP vs Bayesian inference

**MAP**

Let  $o$ ,  $\theta$ , and  $z$  be the observed variable, the model parameter, and the latent variable, respectively. Given the model parameter  $\theta$  and the observation  $o$ , it is natural to estimate  $z$  as

**MAP**

$$\hat{z} = \arg \max_z p(z|\theta, o). \quad (2.51)$$

Note that we can rewrite  $p(z|\theta, o) = \frac{p(z, o|\theta)}{p(o|\theta)} = \frac{p(o|z, \theta)p(z|\theta)}{p(o|\theta)}$ . Since the denominator does not depend on  $z$ , (2.51) can be rewritten as

$$\hat{z} = \arg \max_z p(o|z, \theta)p(z|\theta). \quad (2.52)$$

Often, the observation does not depend on the model parameter once the latent variable  $z$  is given. Therefore, we can simplify (2.52) to

$$\hat{z} = \arg \max_z \underbrace{p(o|z)}_{\text{likelihood}} \underbrace{p(z|\theta)}_{\text{prior}}, \quad (2.53)$$

where  $p(o|z)$  is known as the likelihood function and  $p(z|\theta)$  is the prior. Equations (2.51) through (2.53) describes the so-called Maximum a-posteriori (MAP) estimator.

**ML**

It is sometimes impossible to determine the prior  $p(z|\theta)$ . In that case, the best we can do is to ignore  $p(z|\theta)$  and assume it to be a constant. Hence, (2.53) becomes

$$\hat{z} = \arg \max_z p(o|z), \quad (2.54)$$

**Maximum likelihood**

and this is known as maximum likelihood (ML) estimation.

**Bayesian estimation**

In both MAP and ML, we estimate  $z$  from the mode of some functions ( $p(z|o, \theta)$  and  $p(o|z)$ ). However, it is often a waste to discard all model information except for the mode. Bayesian estimation is more conservative and tries to leverage all possible  $z$  by computing a weighted sum of them as the estimate. More precisely, we have

**Bayesian estimation**

$$\hat{z} = \sum_{z \in \mathcal{Z}} z p(z|\theta, o). \quad (2.55)$$

Often, we are not fundamentally interested in  $z$  but rather in some function  $f$  that depends on  $z$ . In this case, we create an estimate as

$$\sum_{z \in \mathcal{Z}} f(z) p(z|\theta, o). \quad (2.56)$$

In contrast, for MAP and ML, we simply output  $f(\hat{z})$ , where  $\hat{z}$  is the latent variable that maximizes the a posteriori or the likelihood function.

To consolidate the idea, let's consider a simple toy example below.

**Example 2.7: Three types of coins**

Let's say we have three types of identically looking coins, but only the first type is fair. The second type is heavily biased towards heads with  $p(\text{Head}) = 0.8$  and the third type is biased towards tails with  $p(\text{Head}) = 0.4$ .

We have an unknown number of these coins put into a jar. Then, we randomly draw a coin from the jar and toss the coin three times. Let's say we got two tails for the first two tosses. What is the probability of getting a head on the last toss?

Our estimated result heavily relies on the estimation method we use. Let's tackle the problem separately using ML, MAP, and Bayesian inference.

**Example 2.8: Solving the coin problem with ML**

Let's denote  $Z \in \{1, 2, 3\}$  as the type of the coin that was actually picked. Let  $x(z)$  be the

probability of getting a head when a type- $z$  coin is picked. Then,

$$x(z) = \begin{cases} 0.5, & z = 1, \\ 0.8, & z = 2, \\ 0.4, & z = 3 \end{cases}$$

For ML, we assume no prior knowledge of  $Z$ , and the best estimate of  $Z$  is simply

$$\hat{z} = \arg \max_{z \in \{1,2,3\}} p(o|z),$$

where the observation  $o$  is  $(T)ail, (T)ail$ . Thus,

$$p(o|z) = \begin{cases} 0.5 \cdot 0.5 = 0.25, & z = 1, \\ 0.2 \cdot 0.2 = 0.04, & z = 2, \\ 0.6 \cdot 0.6 = 0.36, & z = 3. \end{cases}$$

Since  $p(o|z)$  is largest for  $z = 3$ , we will estimate  $\hat{z} = 3$ , and the predicted probability of heads for the last toss is 0.4.

#### Example 2.9: Solving the coin problem with MAP

When using ML, we do not assume any prior knowledge of  $Z$ . Let's assume that there are two type-1 coins, seven type-2 coins, and only one type-3 coin in the jar. Thus, we have

$$p(z) = \begin{cases} 0.2, & z = 1, \\ 0.7, & z = 2, \\ 0.1, & z = 3. \end{cases}$$

For MAP, we compute the best estimate of  $z$  as

$$\hat{z} = \arg \max_{z \in \{1,2,3\}} p(z|o) \stackrel{(a)}{=} \arg \max_{z \in \{1,2,3\}} \frac{p(z)p(o|z)}{p(o)} \stackrel{(b)}{=} \arg \max_{z \in \{1,2,3\}} p(z)p(o|z),$$

where (a) is due to Bayes' rule and (b) is because  $p(o)$  is a constant with respect to  $z$ . Since

$$p(z)p(o|z) = \begin{cases} 0.2 \cdot 0.25 = 0.05, & z = 1, \\ 0.7 \cdot 0.04 = 0.028, & z = 2, \\ 0.1 \cdot 0.36 = 0.036, & z = 3, \end{cases}$$

the best estimate is  $\hat{z} = 1$ , and so the predicted probability of heads for the last toss is 0.5.

**Example 2.10: Solving the coin problem with Bayesian estimation**

Rather than picking a single best model parameter as in MAP, Bayesian estimation leverages all models and makes a prediction as the weighted average of estimates from all models. That is, we estimate  $x$  as

$$\hat{x} = \sum_{z \in \{1,2,3\}} x(z)p(z|o).$$

Note that  $p(z|o) \propto p(z)p(o|z)$  and should normalize to 1. Therefore, we can compute  $p(z|o)$  as

$$p(z|o) = \begin{cases} \frac{0.05}{0.05+0.028+0.036} = 0.4386, & z = 1, \\ \frac{0.028}{0.05+0.028+0.036} = 0.2456, & z = 2, \\ \frac{0.036}{0.05+0.028+0.036} = 0.3158, & z = 3. \end{cases}$$

Therefore,

$$\hat{x} = 0.4386 \cdot 0.5 + 0.2456 \cdot 0.8 + 0.3158 \cdot 0.4 = 0.5421.$$

**2.6.2 Conjugate prior**

In the example given in the last section, we have exactly three types of coins and we know precisely the probability of heads for each type. In many real problems, the prior knowledge can be more vague. What if we don't know the exact probability of heads for the coin but tend to believe that we are more likely to have a fair coin (probability of heads close to 0.5) than an unfair coin? In this case, we may impose a prior similar to the one shown in Figure 2.1.

There are many ways we can parametrize a prior, as shown in Figure 2.1. The real challenge is which function we should choose. Note that the likelihood function  $p(o|x)$  is given by  $(1-x)^2$ . More generally, if we have  $u$  heads and  $v$  tails in  $u+v$  tosses,

$$p(o|x) = x^u(1-x)^v. \quad (2.57)$$

To estimate  $x$  with MAP, we want

$$\hat{x} = \arg \max_x p(o|x)p(x) = \arg \max_x x^u(1-x)^v p(x).$$

Different people may have different opinions on the choice of  $p(x)$ . However, if we select  $p(x)$  of a form

$$p(x) \propto x^{u'}(1-x)^{v'},$$

then the resulting posterior distribution has the same form as before. This choice is often made **conjugate prior** for practical purposes, and a prior with the same "form" as its likelihood (and thus posterior) is known as the conjugate prior.

It turns out that the conjugate prior with  $p(x) \propto x^{u'}(1-x)^{v'}$  is the Beta distribution. Here,

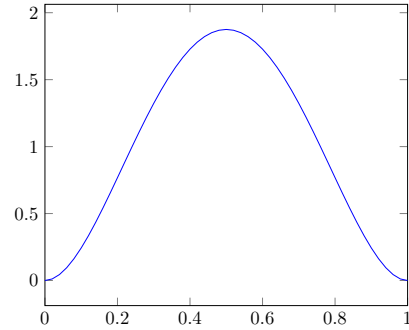


Figure 2.1: A continuous prior: Beta( $x$ ; 3, 3)

we will just state a few facts useful for our discussion. A reader may find more details of the Beta distribution in the Appendix. A Beta distribution has two parameters  $(a, b)$  and its PDF is denoted by  $\text{Beta}(x; a, b) \propto x^{a-1}(1-x)^{b-1}$ . The mode of its PDF is given by

### Beta distribution

$$\begin{cases} \frac{a-1}{a+b-2}, & a, b > 1, \\ [0, 1], & a = b = 1, \\ 0 \text{ or } 1, & \text{otherwise.} \end{cases} \quad (2.58)$$

And the mean of the Beta distribution is given by  $\frac{a}{a+b}$ . If a prior  $\text{Beta}(a, b)$  is chosen, and  $u$  heads and  $v$  tails are observed when tossing a coin  $u+v$  times, the posterior probability is given by

$$p(x|o) = K_1 \cdot \text{Beta}(x; a, b)p(o|x) \quad (2.59)$$

$$= K_1 \cdot \text{Beta}(x; a, b)x^u(1-x)^v \quad (2.60)$$

$$= K_2 \cdot K_1 \cdot x^{a-1}(1-x)^{b-1}x^u(1-x)^v \quad (2.61)$$

$$= K_2 \cdot K_1 \cdot x^{u+a-1}(1-x)^{v+b-1} \quad (2.62)$$

$$= \underbrace{K_3 \cdot K_2 \cdot K_1}_1 \text{Beta}(x; u+a, v+b), \quad (2.63)$$

where  $K_1, K_2$ , and  $K_3$  are some normalization factors, and note that the product  $K_1K_2K_3 = 1$  since both  $p(x|o)$  and  $\text{Beta}(x; u+a, v+b)$  normalize to 1 as we integrate over all  $x$ .

In summary, the posterior probability after observing  $u$  heads and  $v$  tails is simply another Beta distribution but with parameters changed as follows,

$$a \leftarrow u + a, \quad (2.64)$$

$$b \leftarrow v + b. \quad (2.65)$$

#### Example 2.11: Revisit coin problem with $\text{Beta}(3, 3)$ prior

Let's revisit the coin tossing problem but without restricting to any specific types of coin. Instead, we will simply assume the prior probability of heads is  $\text{Beta}(3, 3)$ , which is the one actually shown in Figure 2.1.

After observing two tails, the posterior probability has a distribution  $\text{Beta}(3, 3+2) = \text{Beta}(3, 5)$ .

If we are going to estimate the probability of heads with MAP, we should pick the mode of  $\text{Beta}(x; 3, 5)$ , which will be  $\frac{3-1}{3+5-2} = \frac{1}{3}$ .

If we try to estimate the probability using Bayesian inference, the estimate should be

$$\int_x x p(x|o) dx = \int_x x \text{Beta}(x; 3, 5) dx = \frac{3}{3+5} = \frac{3}{8}.$$

In the above example, we assumed that a prior shown in Figure 2.1 was used. What if we don't have any prior knowledge? It seems reasonable to use a uniform prior, i.e., a constant everywhere. Recall that  $\text{Beta}(x; a, b) \propto x^{a-1}(1-x)^{b-1}$ . Thus, we have a uniform prior if we pick  $a = 1$  and

$b = 1$ .

**Example 2.12: Revisit coin problem with uniform (Beta(1, 1)) prior**

Let's repeat the last example but just pick a uniform prior Beta(1, 1). Thus after observing two tails, the posterior probability has a distribution Beta(1, 1 + 2) = Beta(1, 3).

If we are going to estimate the probability of heads with MAP, we should pick the mode of Beta( $x$ ; 1, 3), which will be  $\frac{1-1}{1+3-2} = 0$ . Note that this result is rather extreme as it essentially rules out the possibility of getting a head for the next toss.

Note that since the prior Beta(1, 1) is really a constant, the MAP estimation with such a prior is actually just the ML estimate.

Instead, if we try to estimate the probability using Bayesian inference, the estimate will be the mean of Beta( $x$ ; 1, 3), which is

$$\int_x x p(x|o) dx = \int_x x \text{Beta}(x; 1, 3) dx = \frac{1}{1+3} = \frac{1}{4}.$$

It may seem surprising at first that the ML estimation (or MAP with a uniform prior) result is so extreme. But without additional information, the best guess of the probability is from statistically counting, and the estimate of zero head probability seems reasonable from that perspective, as none out of two historical tosses were heads.

When we impose a non-uniform prior such as Beta(3, 3) as in our example, it introduces a “regularization” effect that makes the estimate less extreme. Just by inspection, we can see that **Bayesian** the Beta(3, 3) prior can be interpreted as if some prior experiment had been performed before our **estimation offers** observations. In particular, the prior experiment included  $4 = (3 + 3 - 2)$  tosses, out of which **free**  $2 = (3 - 1)$  were heads. Even though for Beta( $u, v$ ) with non-integer  $u$  and  $v$ , it would be much **regularization** more difficult to interpret the physical meaning of such a prior.

Another interesting observation from the above example is that Bayesian inference includes some free regularization even when a non-informative uniform prior is used. The estimated probability of heads is  $\frac{1}{4}$  rather than 0 as in MAP or ML. The averaging effect over many model parameters creates a less extreme estimate and thus offers some regularization effect.

## 2.7 Exercises

1. Show the union bound  $p(A \cup B) \leq p(A) + p(B)$ . Hint: note that we can write  $A \cup B = A \cup (B \setminus A)$  such that  $A$  and  $B \setminus A$  are disjoint.
2. Show that for a r.v.  $X$ ,  $E[X + C] = E[X] + C$  when  $C$  is a constant, regardless of whether  $X$  is continuous or discrete.
3. Show that if  $X$  and  $Y$  are independent, then  $X$  and  $Y$  are uncorrelated. That is,  $E[XY] = E[X]E[Y]$ .
4. Show that if  $X$  and  $Y$  are independent, for any deterministic function  $f(\cdot)$  and  $g(\cdot)$ ,  $f(X)$  and  $g(Y)$  are independent as well.

5. Let  $X$  be a continuous random variable uniformly distributed in the interval  $[-1, 1]$ . Let  $Y = X^2$ .

- (a) Show that  $X$  and  $Y$  are uncorrelated, i.e.,  $E[XY] = E[X]E[Y]$ .  
 (b) Consider deterministic functions  $f(\cdot)$  and  $g(\cdot)$  such that  $f(x) = x^2$  and  $g(y) = y$ . Show that  $f(X)$  and  $g(Y)$  are **not** uncorrelated.

This example illustrated that  $X$  and  $Y$  being uncorrelated does not imply  $f(X)$  and  $g(Y)$  being uncorrelated in general.

6. For  $X$  and  $Y$  that are uncorrelated (i.e.,  $E[XY] = E[X]E[Y]$ ), show that

$$\begin{aligned}\text{Var}(X + Y) &= E[(X + Y - E[X + Y])^2] \\ &= E[(X - E[X])^2] + E[(Y - E[Y])^2] = \text{Var}(X) + \text{Var}(Y).\end{aligned}$$

7. Show that  $X_1 \leftrightarrow X_2 \leftrightarrow \cdots \leftrightarrow X_N$  implies  $X_N \leftrightarrow X_{N-1} \leftrightarrow \cdots \leftrightarrow X_1$ . That is, if

$$p(x^N) = p(x_1)p(x_2|x_1)p(x_3|x_2) \cdots p(x_N|x_{N-1}),$$

we have

$$p(x^N) = p(x_N)p(x_{N-1}|x_N)p(x_{N-2}|x_{N-1}) \cdots p(x_1|x_2)$$

as well.





# Chapter 3

## Quantify information with compression

You should call it entropy for two reasons: first because that is what the formula is in statistical mechanics but second and more important, as nobody knows what entropy is, whenever you use the term you will always be at an advantage

John von Neuman

In this chapter, we will quantify the amount of information in a random variable, which ends up being the entropy of the r.v.. This concept is typically introduced through the source coding theorem. We will present multiple proofs of the theorem, each providing further insights into information theory. Notably, the second proof will introduce the concept of typical sequences, a special case of the law of large numbers, which serves as a crucial tool in information theory.

### 3.1 Overview of entropy

Information is an abstract concept and is challenging to define and quantify. Comparing “the sun will rise from the east tomorrow” and “it is going to rain this coming Wednesday,” which one should have more information? We will argue that the latter is more *informative* since, unless you are from another planet, everyone knows that the former is *always* true.

On the other hand, if a comet is going to hit Earth so hard tomorrow that its rotation flips, the statement “the sun will rise from the *west* the day after tomorrow” is very informative to all of us as it is not something we expect every day. Sadly, we probably wouldn’t survive to see it if it really happens.

How should we quantify the amount of information of an event then? By using probability! We will formally show later in this chapter that we should value an outcome with probability  $p$  with  $-\log p$  bits, as argued in Chapter 1, if the outcome indeed happened. For example, say the probability that the sun will rise from the east tomorrow is 0.9999999 (hopefully, it should be larger than that in reality), and someone tells us that this *will* happen. The “value” of this

rare events have more information but on average are not

piece of information will then be  $-\log_2 0.9999999 \approx 0.00000014$  bits. What will be the amount of information if the sun rises from the west tomorrow?<sup>1</sup> The value of this piece of information is  $-\log_2(1 - 0.9999999) \approx 23.25$  bits. However, this piece of information has no value unless the outcome actually happens.

Now, say if someone can accurately predict if the sun rises from the east the day after, what is the average value of his prediction? It would be  $0.9999999 \cdot 0.00000014 + 0.0000001 \cdot 23.25 \approx 0.0000025$  bits. This may be surprising to some as it does not contain a lot of “value”.

The average value we just computed is known as the entropy. For any r.v., the entropy of the r.v., which only depends on the distribution of the r.v., can be interpreted as the amount of uncertainty of that variable. If we view the r.v. as an information source through repeatedly sampling the r.v., the number of bits on average needed to store the outcome of the r.v. is precisely the entropy as well. This fact is known as the source coding<sup>2</sup> theorem. Essentially, the Source Coding Theorem not only states that on average we can compress an output from a discrete memoryless source to its entropy, but it also states that, on average, we can’t compress a source beyond that theoretical limit.

In this chapter, we will present three different forward proofs of the Source Coding Theorem. The first proof is based on optimization and is likely the most intuitive and natural one for most readers. The second proof is based on the law of large numbers and typical sequences, which may appear more abstract to some but probably is the most elegant. Finally, we will provide a more constructive proof using the Shannon-Fano-Elias (SFE) code.

In the next chapter, we will provide a converse proof of the source coding theorem after we learn more information measures beyond entropy.

### 3.1.1 Limitation of entropy

As discussed, entropy quantifies the average number of bits required to represent a r.v.. However, **entropy can’t estimate economic values** this only computes how much storage is needed on average to keep the information. It does not evaluate the economic value of the information. For example, the entropy of a winning lottery ticket will be less than 100 bits, which is less than the exact counts of different species of insects in my backyard. However, the latter piece of information is likely to have very little economic value, perhaps only satisfying my own curiosity.

Even if we understand that entropy only quantifies the “amount” of information in a variable, **entropy can be counterintuitive** this interpretation can still be counterintuitive and confusing at times. Note that a more random (more uniform) r.v. will have higher entropy than a less random (more skewed) r.v.. Intuitively, we might think the reverse should be true; something less random should have more information as it is more likely to be artificially created. For example, a randomly generated piece of text will have higher entropy than an encyclopedia article given the same character count. But while the latter probably contains valuable information, no one would likely agree that the former contains any useful information at all.

Finally, we have assumed so far that we have access to the distribution of a r.v.. We never **entropy requires knowledge of distribution** questioned how and where we got the distribution. Getting the distribution may be easy for some problems but can be very hard for others. For example, as in our earlier example, how can we estimate the probability that the sun will rise from the east tomorrow? How about the probability of having alien life forms in the universe? Obtaining the distribution itself is beyond the scope of

<sup>1</sup>For simplicity, we assume the sun will always rise and either from the west or the east.

<sup>2</sup>Source coding is just a fancy name for compression among information theorists.

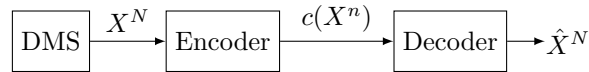


Figure 3.1: Source coding model

information theory. Like most information theory literature, we will treat the distribution of the r.v. as a given truth for the rest of the book, something akin to an axiom in mathematics that we must accept before proceeding. Note that even for problems often treated with probability, such as estimating the probability of drawing an Ace of Spades from a deck of cards, the probability model will only reflect reality if no cheating is involved. Your probability model (assuming no cheating) can be very different from that of your opponent, who cheats and takes that into account.

When in doubt, it is always useful to step back and remember what entropy fundamentally represents.

### What is Entropy?

Given a r.v., its entropy simply quantifies the average number of bits required to represent the r.v..

It is convenient to interpret entropy as a way to quantify the amount of information from a random source, but it is important to be aware of the caveats mentioned above.

## 3.2 Source coding theory

Before stating exactly the Source Coding Theorem, we need to define the source coding model.

### 3.2.1 Source coding model

We will begin by introducing the concept of a discrete memoryless source (DMS), which is generated by repeatedly sampling from a discrete r.v.. The source is considered memoryless because we assume that samples obtained at different times are independent.

Consider a sequence  $X_1, X_2, \dots, X_N$  sampled from a DMS. The problem of source coding is to determine, on average, how many bits are needed to represent each symbol  $X_i$  *losslessly*. The sequence will be compressed by an encoder and later decompressed by a decoder, as shown in Figure 3.1. In plain English, source coding is just the problem of *lossless* compression. It is important that the compression considered here is lossless, meaning each  $X_i$  should be perfectly reconstructed later on.

#### Encoder

The **encoder** compresses a sequence  $x^N = x_1, x_2, \dots, x_N$  from the DMS into a representation  $c(x^N)$  of a "smaller" size. We will assume for the moment that each symbol  $x_i$  will be mapped separately into a binary sequence  $c(x_i)$  and  $c(x^N)$  will simply be a concatenation of  $c(x_1)c(x_2) \dots c(x_N)$ .

discrete  
memoryless  
source

encoder

Generally, we will call  $\mathbf{c}(x^N)$  a **codeword**, and the collection of all codewords a **codebook**, **codeword and** which can be imagined as a table storing each codeword  $\mathbf{c}(x^N)$  for each entry  $x^N$ . By the way, since **codebook**  $c(x)$  is just a special case of  $\mathbf{c}(x^N)$  for  $N = 1$ , we will call  $c(x)$  a codeword as well.

Mapping each symbol independently seems to be a major constraint of the model. But this is not the case in reality since we can always group some symbols together to form a super-symbol and treat each super-symbol independently instead. For example, we may group two symbols and **symbol group** treat the pair as one unit. Then, we will have  $\mathbf{c}(x^N) = c(x_1, x_2)c(x_3, x_4) \cdots c(x_{N-1}, x_N)$  instead. **trick** We call this the **symbol grouping trick** and will need to use it later. One thing to be aware of is that symbol grouping allows us to achieve fractional code rates. In principle, if we allow grouping of  $M$  symbols, we can achieve any fractional code rate as  $M$  goes to infinity.

Now, back to the case of treating each symbol separately, say we have lengths of  $c(x_i)$  to be  $l(x_i)$ . Then, our goal will be to simply minimize the expected length of the symbol  $E[l(X)]$ .

### Decoder

Given a binary sequence (the output of the encoder  $\mathbf{c}(x^N) = c(x_1)c(x_2) \cdots c(x_N)$ ), the **decoder decoder**  $\mathfrak{d}(\cdot) = \mathbf{c}^{-1}(\cdot)$  simply tries to reverse the operation and find  $x^N$ . Note that since the compression is lossless, we should have  $\mathfrak{d}(\mathbf{c}(x^N)) = x^N$ .

At an abstract level, we can consider that the codebook is available for both the encoder and the decoder. The decoder could recover  $x^N$  through a simple table lookup. However, this is usually not computationally feasible, and other techniques are involved. But that is typically beyond the scope of information theory and our discussion here. Yet, in the last section of this chapter, we will discuss the SFE code, which provides a glimpse of what real compression systems might look like.

### Source coding rate

The **source coding rate** is defined as the average number of bits per symbol required to encode **source coding rate** a source sequence. For our model, it is simply

$$R = \frac{1}{N} E[\text{length}(\mathbf{c}(X^N))] = E[l(X)],$$

where  $l(x)$  denotes the length of  $c(x)$ .

### 3.2.2 A glimpse of source coding theorem

Now that we have established the necessary terminology, we can describe the source coding theorem. **source coding theorem**

#### Source Coding Theorem

For a DMS created by a r.v.  $X$ , we can find a lossless encoder-decoder pair if the coding rate is at least  $H(X) \triangleq E[-\log p(X)]$ .  
Moreover, if the coding rate is less than  $H(X)$ , lossless reconstruction will not be possible.

Showing the existence of an encoder-decoder pair is usually known as the **forward proof**, while **forward and** demonstrating that no such pair exists when the coding rate is less than the entropy is known as **converse proofs** the **converse proof**. In the rest of the chapter, we will present the forward proof in three different ways.

### 3.3 Uniquely decodable code

**uniquely  
decodable code**

For lossless compression, all input sequences should map to different compressed outputs. Otherwise, if we have  $\mathbf{c}(\mathbf{x}) = \mathbf{c}(\mathbf{x}')$  for some  $\mathbf{x} \neq \mathbf{x}'$ , there is no way for the decoder to determine whether the original input was  $\mathbf{x}$  or  $\mathbf{x}'$ . Therefore, we need  $\mathbf{c}(\mathbf{x}) \neq \mathbf{c}(\mathbf{x}')$  if  $\mathbf{x} \neq \mathbf{x}'$ , meaning  $\mathbf{c}(\cdot)$  must be injective. A code that satisfies this property is known to be **uniquely decodable**.

Recall that  $\mathbf{c}(x^N) = c(x_1)c(x_2)\cdots c(x_N)$ . When a code is uniquely decodable,  $c(\cdot)$  must be injective, meaning each  $c(x)$  must be distinct for different  $x$ . However, the opposite is not true; an injective  $c(\cdot)$  does not guarantee that the code is uniquely decodable. Consider a simple example where  $X$  has only four outcomes, say  $\mathcal{X} = \{\alpha, \beta, \gamma, \delta\}$ . Let  $c(\alpha) = 1$ ,  $c(\beta) = 0$ ,  $c(\gamma) = 10$ , and  $c(\delta) = 01$ . While  $c(\cdot)$  is injective, the resulting  $\mathbf{c}$  is not uniquely decodable. For example, a decoder receiving 10 cannot determine whether the original input was  $\gamma$  or  $\alpha\beta$ .

#### 3.3.1 Prefix-free code

For practical purposes, we would like to be able to decode a symbol as soon as it becomes available. Consider a code with the following mapping:

$$\alpha \mapsto 10, \beta \mapsto 00, \gamma \mapsto 11, \delta \mapsto 110. \quad (3.1)$$

One can show that this code is uniquely decodable, and we will leave this as an exercise.

Now, consider an input sequence  $\gamma\beta\beta\beta$  that maps to 11000000. Note that when the decoder reads the first 3 bits, it is not able to tell if the first input symbol is  $\gamma$  or  $\delta$ . Actually, it will not be until the decoder reads the last bit that it will be able to confirm that the first input symbol is  $\gamma$ . This is definitely not desirable.

**prefix-free code**

Instead, let's change the code for  $\delta$  from 110 to 011. We can argue that we can always decode a symbol as soon as it becomes available. We call a code with such a property an instantaneous code. Why don't we have the problem of mixing up symbols anymore? In the original code,  $\gamma$  could be mixed up with  $\delta$  since  $\gamma$  is a prefix of  $\delta$ , i.e., 11. However, in the new code, none of the codewords can be a prefix of another, so no such confusion is possible. Therefore, an instantaneous code is also sometimes known as a **prefix-free code**.

Besides its "instant" decoding property, another nice property of a prefix-free code is that it is very easy to verify if a code is prefix-free or not by simply ensuring that none of the codewords can be a prefix of another. And when the code is prefix-free, it is apparent that it will be uniquely decodable. In contrast, it is quite difficult to verify if a code is uniquely decodable if it is not prefix-free, as we see from our earlier example.

### 3.4 Quantify information by minimizing expected codeword length

Now, let's return to the question of quantifying the amount of information in a DMS. Namely, on average, what is the minimum number of bits needed to represent a source symbol losslessly?

Recall that  $\mathbf{c}(x^N) = c(x_1)c(x_2)\cdots c(x_N)$  and  $l(x_i)$  is the length of  $c(x_i)$ . The expected length of the code per symbol is  $E[l(X)]$ . Our objective is to make  $E[l(X)]$  as small as possible for some allowable length profile  $l(x), x \in \mathcal{X}$ .

Note that  $l(x)$  cannot be chosen arbitrarily, as it is simply impossible to have a uniquely decodable code (and hence lossless compression) for some **length profiles**. For example, take **length profile**  $\mathcal{X} = \{\alpha, \beta, \gamma, \delta\}$  and  $l(\alpha) = l(\beta) = 1$ ,  $l(\gamma) = l(\delta) = 2$ . One example could be  $c(\alpha) = 1$ ,  $c(\beta) = 0$ ,  $c(\gamma) = 10$ ,  $c(\delta) = 01$ . It should be apparent that we can never have a uniquely decodable code for this length profile because  $c(\gamma)$  or  $c(\delta)$  would be mixed up with a combination of  $c(\alpha)$  and  $c(\beta)$ .

While it is easy to verify if a code is prefix-free when we see one, how can we determine a length profile that can facilitate a uniquely decodable code? It turns out that we can verify this very easily with a simple condition known as Kraft's Inequality.

### 3.4.1 Kraft's Inequality

Kraft stated a magical condition that whenever the condition is satisfied by a length profile, we can find a uniquely decodable code with that length profile. Otherwise, no uniquely decodable code with the given length profile is possible.

More precisely, consider a length profile  $l_1, l_2, \dots, l_K$ . If

**Kraft's  
inequality**

$$\sum_{k=1}^K 2^{-l_k} \leq 1 \quad (3.2)$$

then there exists a uniquely decodable code with the given profile. That is, we have  $l(x_1) = l_1$ ,  $l(x_2) = l_2, \dots, l(x_K) = l_K$  for symbols  $x_1, x_2, \dots, x_K$ . Otherwise, no such uniquely decodable code is possible.

#### Intuition

Let's get some intuition on where Kraft's inequality in (3.2) comes from. We can represent every codeword of a code by a node in a binary tree. As shown in Fig. 3.2, a left branch split corresponds to a zero, and a right branch split corresponds to a one. So, the codeword 000 will correspond to the leftmost leaf node in the tree, as in Fig. 3.2.

Note that if a codeword is a prefix of another one, its corresponding node will be an ancestor of the latter. Therefore, if a code is prefix-free, the corresponding nodes of all codewords can only be leaf nodes of the tree. Moreover, the descendants of distinct codeword nodes must not overlap!

Let  $l_{\max}$  be the maximum length of a coded symbol, that is,  $l_{\max} = \max_{x \in \mathcal{X}} l(x)$ . For a length- $l$  codeword, note that the number of its descendants at the  $l_{\max}$ -level is simply  $2^{l_{\max}-l}$ . If a code is prefix-free, the descendant sets of all codewords must be disjoint. Therefore, the total number of all length- $l_{\max}$  descendants must be less than or equal to all possible length- $l_{\max}$  codewords, i.e.,

$$\sum_{k=1}^K 2^{l_{\max}-l_k} \leq 2^{l_{\max}} \Rightarrow \sum_{k=1}^K 2^{-l_k} \leq 1.$$

Conversely, if Kraft's inequality is violated, the descendant sets must not be disjoint. Therefore, two codewords must share a descendant, and consequently, one must be a prefix of the other. One can easily verify the latter, and we will leave it as an exercise. As a result, if the descendant sets of codewords are not disjoint, the code is not prefix-free.

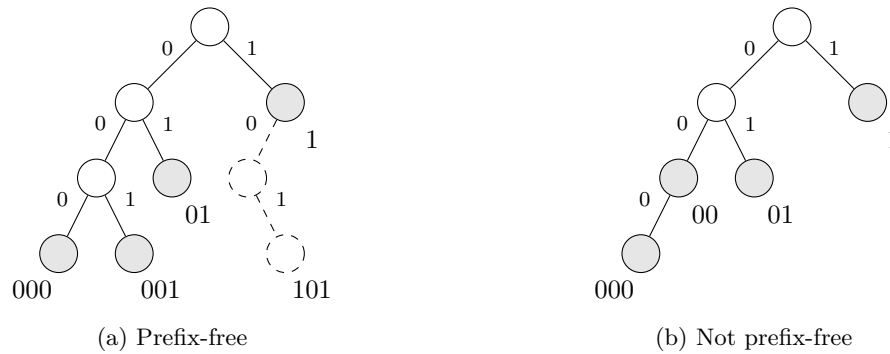


Figure 3.2: Understanding Kraft’s inequality: The left tree corresponds to a prefix-free code, while the right one does not. Note that the grey nodes are the codewords. For a prefix-free code, any descendant of a codeword cannot overlap with another codeword and its descendants. For example, in the left figure, the codeword 101, which is a descendant of 1, cannot overlap with any other codeword and its descendants. On the other hand, in the right figure, the descendant of 00, which is 000, overlaps with another codeword 00, illustrating a violation of the prefix-free property.

### Forward Proof of Kraft’s Inequality

**prefix-code exists for length profile that satisfies Kraft’s inequality**

Here we will show that as long as Kraft’s inequality is satisfied, we will be able to find a prefix-free code and hence a uniquely decodable code with the given profile.

Given  $l_1, l_2, \dots, l_K$  that satisfy  $\sum_{k=1}^K 2^{-l_k} \leq 1$ , we can assign codewords to nodes on a tree and ensure all nodes have disjoint descendants as follows. First, assign one codeword at a time starting from the smallest index and assign it to the highest available node at the  $l_i$ -level. Once a node is assigned, cross out the assigned node and all its descendants, which will become unavailable for future selection. Repeat this until all codewords are assigned.

From our earlier discussion, as long as Kraft’s inequality is satisfied, we know that there are sufficient tree nodes to be assigned. Thus, the corresponding code is apparently prefix-free and, therefore, uniquely decodable.

### Converse Proof of Kraft’s Inequality

**code with length profile that violates Kraft’s inequality is not uniquely-decodable**

From our discussion near the end of the “Intuition” subsection, we see that no prefix-free code can have a length profile that violates Kraft’s inequality. However, one may wonder if we could find a uniquely decodable code with a length profile violating Kraft’s inequality. After all, not all uniquely decodable codes are prefix-free. However, we will show here that this is simply impossible. Essentially, the length profile of any uniquely decodable code has to satisfy Kraft’s inequality.

Recall that  $l_{\max} = \max_{x \in \mathcal{X}} l(x)$  is the maximum length of a coded symbol. We will show that  $\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq (kl_{\max})^{1/k}$  if the code is uniquely decodable. Consequently,  $\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$  as we allow  $k$  to go to infinity, thus satisfying Kraft’s inequality. Now, let’s get into the details. Consider a code sequence from coding  $k$  symbols  $\mathbf{x} = x_1, x_2, \dots, x_k$ , we have

$$\begin{aligned}
\left(\sum_{x \in \mathcal{X}} 2^{-l(x)}\right)^k &= \left(\sum_{x_1 \in \mathcal{X}} 2^{-l(x_1)}\right) \left(\sum_{x_2 \in \mathcal{X}} 2^{-l(x_2)}\right) \cdots \left(\sum_{x_k \in \mathcal{X}} 2^{-l(x_k)}\right) \\
&= \sum_{x_1, x_2, \dots, x_k \in \mathcal{X}^k} 2^{-(l(x_1)+l(x_2)+\dots+l(x_k))} \\
&= \sum_{\mathbf{x} \in \mathcal{X}^k} 2^{-l(\mathbf{x})} = \sum_{m=1}^{kl_{\max}} a(m)2^{-m},
\end{aligned}$$

where  $a(m)$  is the number of codewords with length  $m$ . In the last equality, we tally the sum differently from before. Rather than summing over all  $k$ -symbol inputs, we sum over coded sequences of different lengths. Since there are  $2^m$  different binary sequences of length  $m$ ,  $a(m)$ , the number of length- $m$  codewords, has to be less than or equal to  $2^m$  if the code is uniquely decodable. Therefore, we have

$$\left(\sum_{x \in \mathcal{X}} 2^{-l(x)}\right)^k = \sum_{m=1}^{kl_{\max}} a(m)2^{-m} \leq \sum_{m=1}^{kl_{\max}} 2^m 2^{-m} = kl_{\max}.$$

Consequently,

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq (kl_{\max})^{1/k} \rightarrow 1 \text{ as } k \text{ goes to infinity.}$$

Thus, any uniquely decodable code satisfies Kraft's inequality as stated in (3.2).

### 3.4.2 A proof of Source Coding Theorem

Now, let's provide a proof of the source coding theorem by finding the minimum rate required to compress a source losslessly. Recall that the rate is simply  $E[l(X)] = \sum_{k=1}^K p(x_k)l(x_k) = \sum_{k=1}^K p_k l_k$ , where we define  $p_k \triangleq p(x_k)$  and  $l_k \triangleq l(x_k)$  for simplicity. For lossless recovery, the code must satisfy Kraft's inequality. Therefore, we can find the minimum rate by solving the following optimization problem:

$$\begin{aligned}
&\min_{l_1, l_2, \dots, l_K} \sum_{k=1}^K p_k l_k \\
&\text{subject to } \sum_{k=1}^K 2^{-l_k} \leq 1 \text{ and } l_1, \dots, l_K \geq 0 \\
&\equiv \max_{l_1, l_2, \dots, l_K} - \sum_{k=1}^K p_k l_k \\
&\text{subject to } \sum_{k=1}^K 2^{-l_k} - 1 \leq 0 \text{ and } -l_1, \dots, -l_K \leq 0.
\end{aligned}$$

Let's write down the KKT conditions (please see Appendix). We have:



$$-\nabla \left( \sum_{k=1}^K p_k l_k \right) - \mu_0 \nabla \left( \sum_{k=1}^K 2^{-l_k} - 1 \right) + \sum_{k=1}^K \mu_k \nabla l_k = 0 \quad (3.3)$$

$$\sum_{k=1}^K 2^{-l_k} - 1 \leq 0, \quad l_1, \dots, l_K \geq 0, \quad \mu_0, \mu_1, \dots, \mu_K \geq 0 \quad (3.4)$$

$$\mu_0 \left( \sum_{k=1}^K 2^{-l_k} - 1 \right) = 0, \quad \mu_k l_k = 0. \quad (3.5)$$

Without loss of generality, we will assume all  $p_k \neq 0$  and expect  $l_k > 0$ . Consequently,  $\mu_k = 0$  from (3.5). Consider the  $j$ -th element in (3.3), we have

$$p_j + \mu_0 2^{-l_j} \log 2 = 0 \Rightarrow 2^{-l_j} = \frac{p_j}{\mu_0 \log 2}. \quad (3.6)$$

Moreover, from Kraft's inequality,  $\sum_{k=1}^K 2^{-l_k} \leq 1$ , we have

$$\sum_{k=1}^K \frac{p_k}{\mu_0 \log 2} = \frac{1}{\mu_0 \log 2} \leq 1 \Rightarrow \mu_0 \geq \frac{1}{\log 2} \quad (3.7)$$

Note that as  $\mu_0$  decreases,  $\frac{p_j}{\mu_0 \log 2}$  increases and  $l_j$  decreases as from (3.6). Therefore, if we want to decrease the code rate, we should reduce  $\mu_0$  as much as possible. From (3.7), we should take  $\mu_0 = \frac{1}{\log 2}$ . Then

$$2^{-l_j} = p_j \Rightarrow l_j = -\log_2 p_j.$$

Thus, the minimum rate becomes

$$\sum_{k=1}^K p_k l_k = - \sum_{k=1}^K p_k \log_2 p_k \triangleq H(p_1, \dots, p_K) = H(X).$$

### Remark

The above proof suggests that we can achieve the theoretical compression limit  $H(X)$  if we assign the codeword length for message  $j$ ,  $l_j$ , to  $-\log_2 p_j$ . This reaffirms that a message occurring with probability  $p$  should have information content of  $-\log_2 p$ .

Moreover, since  $-\log_2 p_j$  is typically not an integer, attaining the theoretical limit necessitates the use of fractional codeword lengths. In contrast, our current setup of encoding individual symbols only results in integer codeword lengths. However, by leveraging the symbol grouping technique mentioned earlier, we can, in principle, achieve code lengths with arbitrary precision by allowing the grouping of a large number of symbols. This enables us to approach the compression limit as close as we want.

### 3.5 Quantify entropy using LLN

We will now try to show the Source Coding Theorem again but from a different perspective. This is my favorite proof of the theory. But we need to introduce a new idea known as a typical sequence to proceed.

For sufficiently long sequences sampled from a DMS, one can show that they all behave similarly statistics-wise. We call sequences that share the similar statistics **typical sequences**. The definition is almost a tautology. As almost all sequences sampled from a DMS will be typical. Before discussing typical sequences, we need to introduce the Law of Large Number (LLN), which essentially says that the empirical average will converge to the statistical average given enough sample.

#### 3.5.1 Law of Large Number (LLN)

Consider samples  $x_1, x_2, \dots, x_N$  drawn from a DMS. The LLN states that the empirical average of  $f(x_i)$  will approach the expected value as  $N \rightarrow \infty$ . That is,

**law of large number**

$$\frac{1}{N} \sum_{i=1}^N f(x_i) = E[f(X)] \quad \text{as } N \rightarrow \infty$$

The LLN should be nothing surprising as we use it in everyday life. This is precisely how polls are supposed to work. Pollsters randomly draw samples from a portion of the population and expect the prediction from the sample mean to converge to the population mean as the sample size becomes sufficiently large.

The LLN is a rather strong result. We will only show a weak version here. For any  $a > 0$ ,  $Pr\left(\left|\frac{1}{N} \sum_{i=1}^N f(X_i) - E[f(X)]\right| \geq a\right) \rightarrow 0$  as  $N \rightarrow \infty$ . (i.e., the empirical average converges to the expectation *in probability*.) More precisely, we will show

$$Pr\left(\left|\frac{1}{N} \sum_{i=1}^N f(X_i) - E[f(X)]\right| \geq a\right) \leq \frac{Var(f(X))}{Na^2} \propto \frac{1}{N}.$$

To show that, we will use the Chebyshev's Inequality, which says

**Chebyshev's inequality**

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Let's first prove the Chebyshev's Inequality.

*Proof of Chebyshev's Inequality.* First note that for any r.v.  $X \geq 0$ , we have

$$Pr(X \geq b) \leq \frac{E[X]}{b}, \quad (3.8)$$

which is known as the Markov's Inequality and can be shown readily as

$$\begin{aligned} X &= I(X \geq b) \cdot X + I(X < b) \cdot X \\ &\geq I(X \geq b) \cdot b \\ \Rightarrow E[X] &\geq Pr(X \geq b) \cdot b \end{aligned}$$

Now, take  $X = |Y - E[Y]|^2$  and  $b = a^2$ , by Markov's Inequality stated in (3.8),

$$\begin{aligned} Pr(|Y - E[Y]| \geq a) &= Pr(|Y - E[Y]|^2 \geq a^2) \\ &\leq \frac{E[|Y - E[Y]|^2]}{a^2} = \frac{Var(Y)}{a^2} \end{aligned}$$

□

*Proof of weak LLN.* Let  $Z_N = \frac{1}{N} \sum_{i=1}^N f(X_i)$ , apparently  $E[Z_N] = E[f(X)]$  and

$$Var(Z_N) = \frac{1}{N^2} \sum_{i=1}^N Var(f(X)) = \frac{Var(f(X))}{N}$$

Thus, by Chebyshev's Inequality,

$$\begin{aligned} &Pr\left(\left|\frac{1}{N} \sum_{i=1}^N f(X_i) - E[f(X)]\right| \geq a\right) \\ &= Pr(|Z_N - E[Z_N]| \geq a) \leq \frac{Var(Z_N)}{a^2} = \frac{Var(f(X))}{Na^2} \end{aligned}$$

□

Before continuing with typical sequences, let's consider an interesting application of LLN in the following as an example.

### Example 3.1: Optimal repeated bets: Kelly's Criterion

Let's assume I initially have 1 dollar and I bet an  $r$  fraction of my current net worth each time on an  $a$ -for-1 bet. This means that for each dollar bet, I will collect " $a$ " dollars if I win the bet and lose the entire dollar otherwise.

Given that the probability of winning the bet is  $p$ , the expected wealth after one bet is

$$1 - r + rpa.$$

Clearly, if  $pa < 1$ , I shouldn't bet any money at all. However, if  $pa > 1$ , the expected wealth after one bet is maximized when  $r = 1$ . Does this mean we should always go all-in?

Now, let's consider repeated bets. Let  $Y_i$  represent the fraction of wealth after the  $i$ th bet.

The net wealth  $W_N$  after  $N$  bets is given by:

$$W_N = \prod_{i=1}^N Y_i$$

where

$$Y_i = \begin{cases} (1-r) + ar & \text{with probability } p \\ 1-r & \text{with probability } (1-p) \end{cases}$$

By the Law of Large Numbers (LLN), we have

$$\frac{1}{N} \log W_N = \frac{1}{N} \sum_{i=1}^N \log Y_i \rightarrow E[\log Y].$$

Thus,

$$\log W_N \rightarrow N [p \log(1 + (a-1)r) + (1-p) \log(1-r)].$$

To maximize this gain, we need to maximize the function:

$$f(r) = p \log(1 + (a-1)r) + (1-p) \log(1-r)$$

with respect to  $r$ . Setting the derivative  $\frac{df}{dr}$  to zero, we get:

$$\frac{p(a-1)}{1+r(a-1)} - \frac{1-p}{1-r} = 0 \Rightarrow r = \frac{ap-1}{a-1}.$$

Note that we should never go all-in unless  $p = 1$ !

### 3.5.2 Asymptotic equipartition and typical sequences

Consider a sequence of symbols  $x_1, x_2, \dots, x_N$  sampled from a discrete memoryless source (DMS). We want to compute the sample average of the log-probabilities of each sampled symbol. By the Law of Large Numbers (LLN), we have

$$\frac{1}{N} \sum_{i=1}^N \log \frac{1}{p(x_i)} \rightarrow E \left[ \log \frac{1}{p(X)} \right] = H(X) \quad (3.9)$$

Now, for the left-hand side (LHS):

$$\frac{1}{N} \sum_{i=1}^N \log \frac{1}{p(x_i)} = \frac{1}{N} \log \frac{1}{\prod_{i=1}^N p(x_i)} = -\frac{1}{N} \log p(x^N), \quad (3.10)$$

where  $x^N = (x_1, x_2, \dots, x_N)$ .

Combining (3.9) and (3.10), we have the probability of the sampled sequence

$$p(x^N) \rightarrow 2^{-NH(X)} \quad (3.11)$$

for any sequence sampled from the source!

### Set of typical sequences

We will describe the sequence  $x^N$  with  $p(x^N) \sim 2^{-NH(X)}$  as typical. More precisely, let's define the set of typical sequences as follows

$$\mathcal{A}_\epsilon^N(X) = \{x^N \mid 2^{-N(H(X)+\epsilon)} \leq p(x^N) \leq 2^{-N(H(X)-\epsilon)}\}. \quad (3.12)$$

By the Law of Large Numbers (LLN), for any  $\epsilon > 0$ , we can find a sufficiently large  $N$  such that any sampled sequence from the source is typical. Consequently, all these sequences will have almost the same probability. This phenomenon is referred to as the Asymptotic equipartition (AEP), which implies that the probability space is asymptotically equally partitioned by typical sequences of equal probability.

### Size of typical set

Since all typical sequences have probability  $\sim 2^{-NH(X)}$  and they fill up the entire probability space (everything is typical), there should be approximately  $\frac{1}{2^{-NH(X)}} = 2^{NH(X)}$  typical sequences.

More precisely, the size of the typical set  $\mathcal{A}_\epsilon^N(X)$  is bounded by

$$(1 - \delta)2^{N(H(X)-\epsilon)} \leq |\mathcal{A}_\epsilon^N(X)| \leq 2^{N(H(X)+\epsilon)} \quad (3.13)$$

This can be shown rather easily as follows.

$$\begin{aligned} 1 &\geq Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \stackrel{(a)}{\geq} \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)+\epsilon)} \\ &= |\mathcal{A}_\epsilon^N(X)| 2^{-N(H(X)+\epsilon)}, \end{aligned}$$

where (a) is from the definition of  $\mathcal{A}_\epsilon^N(X)$ . And for any  $\delta > 0$ , given a sufficiently large  $N$ , we have

$$\begin{aligned} 1 - \delta &\leq \stackrel{(a)}{Pr}(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \stackrel{(b)}{\leq} \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)-\epsilon)} \\ &= |\mathcal{A}_\epsilon^N(X)| 2^{-N(H(X)-\epsilon)}, \end{aligned}$$

where (a) is because of the LLN and (b) is from the definition of  $\mathcal{A}_\epsilon^N(X)$ .

### Example 3.2: Coin flipping example

Consider flipping a bias coin with  $Pr(\text{Head}) = 0.3$  say  $N = 1000$  times

- All typical sequences will have approximately 300 heads and 700 tails. That means,

we should get approximately 300 heads out of the 1000 tosses.

- AEP (and LLN) tells us that it is almost impossible to get, say, a sequence of 100 heads and 900 tails

Now, with the knowledge of typical sequences, we can give a very simple proof of the Source Coding Theorem.

### A forward proof of Source Coding Theorem using AEP

Consider a DMS  $X$  and all length- $N$  sequences that can be generated from  $X$ . By AEP, all these sequences are typical for sufficiently large  $N$ . Moreover, there are  $2^{NH(X)}$  such sequences. Therefore, we can create a code that simply index all these sequences with  $\log 2^{NH(X)} = NH(X)$  bits. Thus, the required source coding rate, i.e., bits needed to represent each symbol on average, is  $\frac{NH(X)}{N} = H(X)$  bits.

## 3.6 Quantify entropy by construction

We argued how we can represent a DMS with  $H(X)$  bits per symbol through optimization in Section 3.4 and AEP in Section 3.5. However, in both cases, the codes described are rather abstract and not quite concrete. In this section, we will yet give another proof of the Source Coding Theorem with a more “constructive” approach. Hopefully, this gives further insight for this important theorem.

We will begin by introducing the SFE code. Although it may not appear to be highly efficient, it is straightforward to analyze and sufficient for our discussion.

### 3.6.1 Shannon-Fano-Elias code

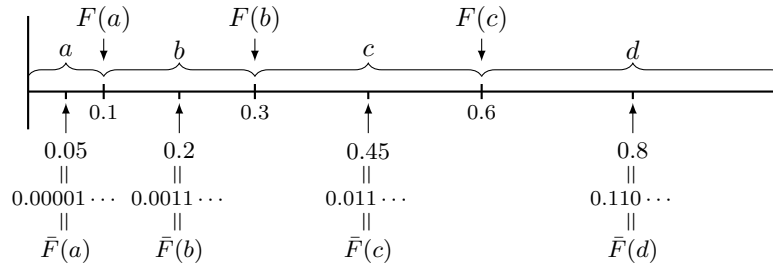


Figure 3.3: A SFE code example

The key idea of the SFE code is to create a code word from the binary representation of any **Shannon-Fano-Elias code** number within the interval  $[0, 1]$ .

To generate the SFE codebook for a DMS  $X$ , we first sort the alphabet of  $X$  and then create the cumulative mass function  $F(\cdot)$  of  $X$ , where  $F(x) = \sum_{x' \leq x} p(x')$ . Next, we define  $\bar{F}(x) = F(x) - 0.5 \cdot p(x)$ . The codeword for  $x$  is then obtained by taking the first  $l(x)$  bits of the fractional part of the binary representation of  $\bar{F}(x)$ , where  $l(x) = \lceil -\log p(x) \rceil + 1$ .

The process of constructing the codebook may seem mysterious at the moment. Before explaining the reasoning behind these choices, let's look at a concrete example.

**Example 3.3: Example: A SFE code**

Consider a DMS  $X$  as shown in Fig. 3.3 ( $p(\alpha) = 0.1, p(\beta) = 0.2, p(\gamma) = 0.3, p(\delta) = 0.4$ ). We have

$$\begin{aligned} F(\alpha) &= \bar{F}(\alpha) = 0.05 \approx 0.00001b \\ F(\beta) &= 0.3, \bar{F}(\beta) = 0.2 \approx 0.0011b \\ F(\gamma) &= 0.6, \bar{F}(\gamma) = 0.45 \approx 0.0111b \\ F(\delta) &= 1, \bar{F}(\delta) = 0.8 \approx 0.110b \end{aligned}$$

As  $l(\alpha) = \lceil -\log 0.1 \rceil + 1 = 5, l(\beta) = \lceil -\log 0.2 \rceil + 1 = 4, l(\gamma) = \lceil -\log 0.3 \rceil + 1 = 3$ , and  $l(\delta) = \lceil -\log 0.4 \rceil + 1 = 3$ . We have

$$\begin{aligned} c(\alpha) &= 00001 \\ c(\beta) &= 0011 \\ c(\gamma) &= 011 \\ c(\delta) &= 110 \end{aligned}$$

**SFE code is prefix-free**

One of the most important properties of the SFE code is that it is prefix-free given the described construction. First, note that the conversion between codewords and intervals can go both ways. We can map a fractional number within  $[0, 1]$  to a codeword, and we can also map any codeword to an interval within  $[0, 1]$ .

For example, the codeword 110 corresponds to

$$u(110) = [0.110b, 0.110\dot{1}b] = [0.11b, 0.111b] = [0.75, 0.875]$$

and the codeword 011 corresponds to

$$u(011) = [0.1b, 0.11\dot{1}b] = [0.1b, 1b] = [0.5, 1).$$

With a slight abuse of notation, let's denote  $u(x)$  as the **codeword interval** corresponding to the codeword of  $x$ . And we can make the following observations regarding  $u(x)$ :

**Observation 1** Given an SFE codeword  $c(x)$  with length  $l(x) = |c(x)|$ , the length of the corresponding interval  $|u(x)| = 2^{-l(x)}$ .

**Observation 2** If  $u(x_1)$  and  $u(x_2)$  do not overlap, then  $c(x_1)$  and  $c(x_2)$  cannot be prefixes of one another. Consequently, if all codeword intervals do not overlap, the code must be prefix-free.

**Observation 3** The respective intervals of all SFE codewords are disjoint.

**codeword  
interval**

The first observation should be rather obvious. It may not be immediately clear with the other two observations. Let's give the quick proofs below.

*Proof of Observation 2.* Given statements  $A$  and  $B$ , note that  $A \Rightarrow B \equiv \neg B \Rightarrow \neg A$ . Therefore, let's show instead that if  $c(x_1)$  and  $c(x_2)$  are prefixes of one another, then  $u(x_1)$  and  $u(x_2)$  overlap. For example, consider the codeword 101 and its prefix 10. The corresponding intervals  $[0.101, 0.11)$  and  $[0.10, 0.11)$  do overlap with one another.

Without loss of generality, assume that  $c(x_1)$  is a prefix of  $c(x_2)$ . The lower boundary of  $u(x_1)$  is below the lower boundary of  $u(x_2)$ , and the upper boundary of  $u(x_1)$  is above the upper boundary of  $u(x_2)$ . Thus,  $u(x_2) \subseteq u(x_1)$ , and hence  $u(x_1)$  and  $u(x_2)$  overlap with each other.  $\square$

*Proof of Observation 3.* From Observation 1, the length of the interval is

$$2^{-l(x)} = 2^{-([\lceil -\log p(x) \rceil + 1])} = 0.5 \cdot 2^{-\lceil -\log p(x) \rceil} \leq 0.5 \cdot 2^{-\log p(x)} = 0.5 \cdot p(x).$$

Since the interval must include  $\bar{F}(x)$  and  $\bar{F}(x)$  is  $0.5 \cdot p(x)$  away from the boundary of the **code intervals** of probability interval of  $x$ , the interval  $u(x)$  must fall completely within the probability interval of  $x$ . **SFE codes are disjoint** Since the probability intervals of all  $x \in \mathcal{X}$  are disjoint, the intervals  $u(x)$ ,  $x \in \mathcal{X}$ , are disjoint as **disjoint** well as illustrated in Figure 3.4.  $\square$

Combining Observations 2 and 3, we thus show that SFE is prefix-free and consequently uniquely-decodable. **SFE codes are prefix-free**

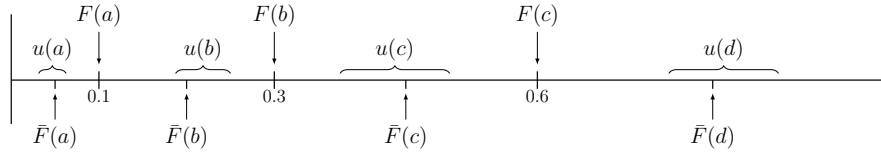


Figure 3.4:  $u(x)$  will not go beyond its probability interval and will not overlap with other intervals as described by Observation 3. Consequently, an SFE code is prefix-free by Observation 2.

### 3.6.2 A constructive proof of Source Coding Theorem

From our earlier discussion, we can always construct an SFE code for any DMS  $X$ . Moreover, we can easily verify that the average code rate of an SFE code is bounded by  $H(X) + 2$  as follows

$$\begin{aligned} \sum_{x \in \mathcal{X}} p(x)l(x) &= \sum_{x \in \mathcal{X}} p(x) \left( \left\lceil \log_2 \frac{1}{p(x)} \right\rceil + 1 \right) \\ &\leq \sum_{x \in \mathcal{X}} p(x) \left( \log_2 \frac{1}{p(x)} + 2 \right) \\ &= H(X) + 2 \end{aligned}$$



The SFE code is not highly optimized. However, we can increase its efficiency easily with the “symbol grouping” trick again.

Let’s first consider combining two adjacent symbols into a super symbol  $X_S$ . By our earlier discussion, the code rate is bounded by  $H(X_S) + 2$ , where

$$\begin{aligned}
 H(X_S) &= - \sum_{x_1, x_2 \in \mathcal{X}^2} p(x_1, x_2) \log_2 p(x_1, x_2) \\
 &= - \sum_{x_1, x_2 \in \mathcal{X}^2} p(x_1, x_2) \log_2 (p(x_1)p(x_2)) \\
 &= - \sum_{x_1, x_2 \in \mathcal{X}^2} p(x_1, x_2) \log_2 p(x_1) - \sum_{x_1, x_2 \in \mathcal{X}^2} p(x_1, x_2) \log_2 p(x_2) \\
 &= - \sum_{x_1 \in \mathcal{X}} p(x_1) \log_2 p(x_1) - \sum_{x_2 \in \mathcal{X}} p(x_2) \log_2 p(x_2) \\
 &= 2H(X)
 \end{aligned}$$

Therefore, the code rate per original symbol is then upper bounded by:

$$\frac{1}{2} (H(X_S) + 2) = H(X) + 1.$$

We see now that the code rate is closer to the compression limit  $H(X)$ . By leveraging the symbol grouping trick further with the SFE code, we can provide another forward proof of the Source Coding Theorem as follows.

### Forward Proof of Source Coding Theorem with SFE Code

In theory, we can group as many symbols as we want using the symbol grouping trick. Say we group  $N$  symbols at a time and compress them using the SFE code. The code rate per original symbol is then upper bounded by

$$\frac{1}{N} (H(X_S) + 2) = \frac{1}{N} (NH(X) + 2) = H(X) + \frac{2}{N}.$$

Therefore, as long as a given rate  $R > H(X)$ , we can always find a large enough  $N$  such that the code rate using the symbol grouping trick and SFE code is below  $R$ . This concludes the forward proof.

One might think that using the “grouping trick” with many symbols is not realistic in practice, since the encoder and decoder complexity would grow exponentially with  $N$ . However, it turns out that it is possible to use very large  $N$  (essentially infinitely large  $N$ ) in combination with the SFE code. The resulting code is known as arithmetic coding. However, this is beyond the scope of this book, and readers interested in the topic are referred to the original paper [1].

## 3.7 Exercise

1. Show that the code described in (3.1) is uniquely decodable.

2. Show that if two codewords share a descendant, then one must be a prefix of the other.
3. Show that if Kraft's inequality is violated, one codeword has to be a prefix of another.
4. If Kelly bet is applied every time as in Example 3.1, what is the approximate final wealth after  $N$  bets?

# Chapter 4

## Information measures

Besides the entropy that was introduced in Chapter ??, we will discuss in more detail other common information measures such as conditional entropy, differential entropy, joint entropy, mutual information, and KL-divergence in this chapter.

### 4.1 Entropy and differential entropy

#### 4.1.1 Revisiting the entropy

Recall from the last chapter, we defined the entropy of a discrete r.v. as

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log p(x) = E[-\log p(X)]. \quad (4.1)$$

The expression may look mysterious at first, but it suggests that the amount of information for an outcome  $x$  is  $-\log p(x) = \log \frac{1}{p(x)}$ , and the entropy is just the average information across all outcomes.

This seems quite natural: a less likely outcome (small  $p(x)$ ) should yield more information ( $\log \frac{1}{p(x)}$ ). The precise expression should also come as no surprise if we interpret it as follows. For example, consider a uniform random variable with 4 outcomes; each outcome will have a probability  $\frac{1}{4} = 0.25$  of occurring. To represent each outcome, we need  $\log 4 = \log \frac{1}{0.25}$  bits.

#### Key Takeaway

A less likely event has “more” information, but on average, the rare event often requires fewer bits to store as it rarely happens. And  $H(X)$  is just the average number of bits required to store all possible outcomes.

As we discussed in the last chapter, we define entropy as the minimum number of bits required to store a r.v. on average. Alternatively, we can also treat entropy as the average uncertainty of

the outcome of the r.v., or the amount of information gained after the outcome is revealed. Thus, if the r.v. is deterministic (not actually “random”), the entropy should be 0.

### A Remark on Notation

It is most common to represent the entropy of a r.v.  $X$  by  $H(X)$ . However, from the expression in (4.1),  $H(X)$  only really depends on the distribution of  $X$ . Strictly speaking, it is probably more appropriate to represent it by  $H(p(X))$  instead. However, we will stick with the notation  $H(X)$  as everyone else does. It is understood that  $H(X)$  means  $H(p(X))$ , where  $p(X)$  is the PMF of  $X$ .

In the case of a binary r.v.,  $p(X)$  can be completely characterized by a single parameter (say  $p(X = 1) = q$ ). One may write  $H(q)$  instead of  $H(p(X))$ . Thus, when  $q$  is scalar, we have

$$H(q) = -q \log(q) - (1 - q) \log(1 - q) \quad (4.2)$$



### Example 4.1: Entropy of a biased coin

Consider a biased coin with  $Pr(\text{Head}) = p$

$$\begin{aligned} H(X) &= -Pr(\text{Head}) \log Pr(\text{Head}) - Pr(\text{Tail}) \log Pr(\text{Tail}) \\ &= -p \log p - (1 - p) \log(1 - p) = H(p). \end{aligned}$$

Note that  $H(X)$  reaches its maximum value of 1 at  $p = 0.5$ , and it decreases to 0 when  $p = 0$  or  $p = 1$ . These values make sense, as the outcome is most uncertain when  $p = 0.5$  and becomes deterministic when  $p = 0$  or  $p = 1$ .

## 4.1.2 Differential entropy

The earlier definition makes little sense for a continuous  $X$ , since the probability of getting a particular outcome  $x$  in the continuous case is always 0. Thus, we have a different definition in for a continuous  $X$  as

$$h(X) = - \int_{x \in \mathcal{X}} p(x) \log p(x) dx = E[-\log p(x)], \quad (4.3)$$

**differential  
entropy**

where  $h(X)$  is known as the differential entropy and  $p(x)$  is now the PDF rather than the PMF. To gain more insights, let's compute the differential entropy for several common distributions in the following.

**Example 4.2: Differential entropy for uniform distribution**

Consider a r.v. with uniform distribution  $p(X) = \begin{cases} 1/a & 0 \leq x \leq a \\ 0 & \text{otherwise} \end{cases}$ , the differential entropy will then be

$$h(X) = - \int_{x=0}^a \frac{1}{a} \log \frac{1}{a} dx = \log a.$$

**Example 4.3: Differential entropy for exponential distribution**

For an exponentially distributed r.v.  $T \sim \text{Exp}(\lambda)$ ,

$$\begin{aligned} h(T) &= E[-\log p(T)] = E[-\log(\lambda \exp(-\lambda T))] \\ &= E[\lambda T - \log \lambda] \\ &= 1 - \log \lambda. \end{aligned}$$

**Example 4.4: Differential entropy for Gaussian distribution**

For a Gaussian distributed  $X \sim \mathcal{N}(\mu, \sigma^2)$ ,

$$\begin{aligned} h(X) &= E[-\log p(X)] = E \left[ -\log \left( \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(X-\mu)^2}{2\sigma^2} \right) \right] \\ &= E \left[ \log \sqrt{2\pi\sigma^2} + \frac{(X-\mu)^2}{2\sigma^2} \log e \right] \\ &= \log \sqrt{2\pi\sigma^2} + \frac{1}{2} \log e \\ &= \log \sqrt{2\pi e\sigma^2}. \end{aligned}$$

It is worth mentioning that  $h(X)$  only depends on  $\sigma^2$  and is independent of  $\mu$  as one would expect.

**Example 4.5: Differential entropy for multivariate Gaussian**

For  $N$ -dim multivariate Gaussian distributed  $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ ,

$$\begin{aligned}
h(\mathbf{X}) &= E[-\log p(\mathbf{X})] \\
&= -E \left[ \log \left( \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp \left( -\frac{1}{2} (\mathbf{X} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu}) \right) \right) \right] \\
&\stackrel{(a)}{=} \log \sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2} E \left[ \sum_{i,j} (X_i - \mu_i) [\Sigma^{-1}]_{i,j} (X_j - \mu_j) \right] \\
&\stackrel{(b)}{=} \log \sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2} \sum_{i,j} [\Sigma^{-1}]_{i,j} E[(X_j - \mu_j)(X_i - \mu_i)] \\
&\stackrel{(c)}{=} \log \sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2} \sum_{i,j} [\Sigma^{-1}]_{i,j} \Sigma_{j,i} \\
&\stackrel{(d)}{=} \log \sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2} \sum_{i,j} I_N \\
&= \log \sqrt{\det(2\pi\Sigma)} + \frac{N \log e}{2} = \log \sqrt{e^N \det(2\pi\Sigma)} = \log \sqrt{\det(2\pi e\Sigma)},
\end{aligned}$$

where (a) just results from expanding the matrix multiplications, (b) follows from applying expectation to each term of the sum and with the deterministic scalars  $[\Sigma^{-1}]_{i,j}$  pulling out from the expectations, (c) is due to covariance  $\Sigma_{i,j} = \Sigma_{j,i} \triangleq E[(X_j - \mu_j)(X_i - \mu_i)]$ , and  $I_N$  is the identity matrix of size  $N$  in (d).

**4.1.3 Connection between differential entropy and entropy**

As the amount of information contained in a continuous r.v. is always infinite, one may wonder how we should interpret differential entropy physically. Fortunately, we can connect the differential entropy of a continuous r.v. with the entropy of a quantized version of the r.v..

Let's consider a continuous random variable  $X$  and let  $X_\Delta$  be a "quantized" version of it with a quantization step size of  $\Delta$  as in Figure 4.1. Then,

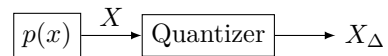


Figure 4.1: The entropy of the quantized  $X_\Delta$  and the differential entropy of  $X$  are related by  $H(X_\Delta) = h(X) - \log_2 \Delta$ .

$$\begin{aligned}
H(X_\Delta) &= \sum -p_{X_\Delta}(x_\Delta) \log p_{X_\Delta}(x_\Delta) \\
&\approx \sum -p_X(x_\Delta) \Delta \log(p_X(x_\Delta) \Delta) \\
&\approx \int -p_X(x) \log(p_X(x) \Delta) dx \\
&= \int -p_X(x) \log p_X(x) dx - \int p_X(x) \log \Delta dx \\
&= h(X) - \log \Delta
\end{aligned}$$

This shows how the entropy of a quantized version of a continuous r.v. relates to its differential entropy. It is an important result and so we will highlight it as below.

#### Connecting entropy to differential entropy

$$H(X_\Delta) = h(X) - \log \Delta \quad (4.4)$$

#### Example 4.6: Bits needed to store the processing time of a packet

**Question:** Consider that the processing time of a packet follows an exponential distribution with an average of 1 ms. If we want to store the time with a precision of 0.01 ms, about how many bits are needed to store the result?

**Answer:** The processing time  $T$  follows an exponential distribution with parameter  $\lambda = 1/1 = 1 \text{ ms}^{-1}$ . Thus, the corresponding differential entropy is  $h(T) = 1 - \log(\lambda) = 1$ . To store the time with a precision of 0.01 ms, we need  $h(T) - \log 0.01 \approx 1 - \log(0.01) \approx 1 - (-2) = 3$  bits.

However, the correct calculation should account for the base-2 logarithm:

$$h(T) - \log_2(0.01) \approx 1 - \log_2(0.01) \approx 1 - (-6.64) = 7.64 \text{ bits.}$$

So, approximately 7.64 bits are needed to store the result.

#### 4.1.4 Bounds of entropy and differential entropy

##### Lower bound of entropy

It should be obvious that  $H(X) \geq 0$ . After all,  $H(X)$  represents the required bits to compress the source  $X$  and hence cannot be negative.

The proof is almost trivial. Since  $p(X) \leq 1$ ,  $-\log p(X) \geq 0$ , therefore:

$$H(X) = E[-\log p(X)] \geq 0.$$

connection  
between  
differential  
entropy and  
entropy

entropy is  
non-negative

### Differential Entropy Can Be Negative

The above is not true for differential entropy. We can have

$$h(X) < 0.$$



For example, for a uniformly distributed  $X$  from 0 to 0.5,

$$h(X) = \log 0.5 = -1.$$

### Upper bound of entropy

$H(X)$  is upper bounded by  $\log |\mathcal{X}|$ , which can be readily shown by Jensen's Inequality as described **Jensen's inequality** below.

### Jensen's Inequality

A convex (bowl-shaped) function  $f$ , as illustrated in Fig. 4.2, satisfies

$$E[f(X)] \geq f(E[X]).$$

Moreover, when  $f(\cdot)$  is strictly convex, i.e.,  $pf(x_1) + (1-p)f(x_2) > f(px_1 + (1-p)x_2)$ , we have equality only if  $X$  is a constant. Then,  $E[f(X)] = f(E[X])$ .

To gain some intuition about the inequality, let us consider  $X$  with only two outcomes  $x_1$  and  $x_2$  with probabilities  $p$  and  $1-p$ . It is easy to see that

$$E[f(X)] = pf(x_1) + (1-p)f(x_2) \geq f(px_1 + (1-p)x_2) = f(E[X]).$$

We can extend the above argument to discrete variables with more than two outcomes using induction. And the inequality generally holds for continuous r.v.s as well.

### Example 4.7: A famous paradox

#### A person with an average weight and an average height is overweight.

At first glance, this assertion seems paradoxical. However, it's crucial to realize that normal weight should be proportional to volume, and thus proportional to  $H^3$ . So in an idealized world, we assume that if a person's height is  $H$ , their weight  $W$  can be represented as  $W = kH^3$  for some constant  $k$ .

For someone with an average height, the expected weight should be  $kE[H]^3$ . But, given that this person has an "average" weight, it is  $E[W] = E[kH^3] = kE[H^3]$ . Due to the strictly convex nature of the function  $f(x) = x^3$ , Jensen's inequality gives us  $kE[H^3] > kE[H]^3$ . This indicates that the individual's weight is above what should be considered the standard weight.



**entropy is upper bounded by log-cardinality**

Now, let's try to find an upper bound for  $H(X)$ .

$$\begin{aligned} H(X) &= E[-\log p(X)] = E\left[\log \frac{1}{p(X)}\right] \\ &\stackrel{(a)}{\leq} \log E\left[\frac{1}{p(X)}\right] \quad (\text{by Jensen's inequality}) \\ &= \log \sum_{x \in \mathcal{X}} p(x) \frac{1}{p(x)} = \log \sum_{x \in \mathcal{X}} 1 = \log |\mathcal{X}|, \end{aligned}$$

where  $|\mathcal{X}|$  denotes the cardinality or size of the outcome set of  $X$ .

Since  $\log(\cdot)$  is strictly convex, we can attain the upper bound  $\log |\mathcal{X}|$  if and only if the equality at (a) holds. Consequently, we need  $\frac{1}{p(\cdot)}$  to be a constant function, in other words,  $p(\cdot)$  must be uniform.

#### Example 4.8: Coins and dices

The upper bound we just described should be unsurprising. For instance, to store the outcome of an unbiased coin flip, we need  $\log 2 = 1$  bit. If the coin is biased, we will need fewer bits to store the outcome. In the extreme case, if we always get a head or a tail, we don't need any bits to store the result.

Similarly, for an 8-sided die, on average we will need exactly  $\log 8 = 3$  bits to store the result when the die is fair. Otherwise, fewer than 3 bits are needed.

#### Bounds of differential entropy

As mentioned earlier, differential entropy in general does not have an upper or lower bound. In particular, differential entropy can be negative as shown in an example in (4.1.4).

Note that since we can still write  $h(X) = E[-\log p(X)]$ , by Jensen's inequality, we have

$$h(X) \leq \log E\left[\frac{1}{p(X)}\right] = \log \int_{x \in \mathcal{X}} p(x) \frac{1}{p(x)} dx = \log |\mathcal{X}|.$$

However, this expression is not quite useful since the support  $\mathcal{X}$  may be unbounded (e.g.,  $\mathcal{X} = (-\infty, \infty)$  as for a normally distributed  $X$ ), and so  $\log |\mathcal{X}|$  can be infinite as well.

**shifting mean won't change entropy**

Thus, it makes much more sense to consider the upper bound of differential entropy constrained by the statistics of the variable. But what statistics should we consider? Let's start with the simplest one: will the entropy of a variable be bounded if we fix its mean, the first-order statistic?

The answer is no. Note that the differential entropy only depends on the shape of the distribu-

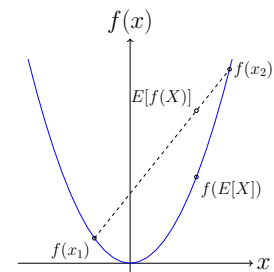


Figure 4.2: Jensen's inequality explained

tion, not its location. Shifting the distribution does not change the entropy because

$$\begin{aligned} h(X + C) &= E[-\log p(X + C)] \\ &= - \int p(x + C) \log p(x + C) dx \\ &= - \int p(x) \log p(x) dx = h(X). \end{aligned}$$

So, a constraint on the mean, which is equivalent to shifting the distribution's mean to the target mean, will not affect the value of differential entropy.

Let's consider the next least trivial statistic, the variance, which is a second-order statistic. Increasing the variance will make the distribution more spread out and thus increase its entropy. **increasing** Even if the variance is the same, different shapes of the distributions will result in different entropy **variance will** values. It turns out that for a fixed variance  $\sigma^2$ , the variable will have the largest differential **increase entropy** entropy if it is normally distributed (as will be shown later). Thus,

$$h(X) \leq \log \sqrt{2\pi e\sigma^2}. \quad (4.5)$$

### 4.1.5 Joint entropy

For multiple discrete random variables, say  $X$  and  $Y$ , we can extend the definition of entropy naturally as

**joint entropy**

$$H(X, Y) = E[-\log p(X, Y)].$$

For  $N$  discrete variables  $X_1, \dots, X_N$ ,

$$H(X_1, X_2, \dots, X_N) = E[-\log p(X_1, \dots, X_N)].$$

The definition is essentially the same for continuous variables  $X$  and  $Y$ ,

$$h(X, Y) = E[-\log p(X, Y)].$$

For  $N$  continuous variables  $X_1, \dots, X_N$ ,

$$h(X_1, X_2, \dots, X_N) = E[-\log p(X_1, \dots, X_N)].$$

Note, however, that the expectation  $E[\cdot]$  is defined differently for continuous and discrete variables, where the former involves an integral and the latter involves a summation.

## 4.2 Conditional entropy

We can expand the joint entropy  $H(X, Y)$  of  $X$  and  $Y$  as follows:

$$\begin{aligned} H(X, Y) &= E[-\log p(X, Y)] = E[-\log p(X) - \log p(Y|X)] \\ &= E[-\log p(X)] + E[-\log p(Y|X)] \\ &= H(X) + \underbrace{E[-\log p(Y|X)]}_{H(Y|X)}. \end{aligned}$$

The joint entropy ends up equal to the sum of the entropy  $H(X)$  and  $E[-\log p(Y|X)]$ , which we define as the conditional entropy of  $Y$  given  $X$ . In other words, conditional entropy is defined as

$$H(Y|X) \triangleq H(X, Y) - H(X).$$

The definition is self-explanatory. If  $H(X, Y)$  is the total uncertainty of both  $X$  and  $Y$ , it should equal the uncertainty of  $X$  plus the uncertainty of  $Y$  knowing  $X$ .

Similarly, we can define the conditional differential entropy as

$$h(Y|X) \triangleq h(X, Y) - h(X).$$

### Conditioning reduces entropy

Generally speaking, we expect that additional knowledge should reduce uncertainty. So we should have

$$H(Y) \geq H(Y|X), \tag{4.6}$$

where this principle is sometimes referred to as *conditioning reduces entropy*. A formal proof is given later in Section 4.4.1 after we introduce KL-divergence.

### 4.2.1 Conditional entropy as an average of entropy over the conditioned r.v.

Let's expand  $H(Y|X)$  in another way. One can easily show that

$$H(Y|X) = \sum_x p(x) H(Y|x), \tag{4.7}$$

conditional  
entropy

conditioning  
reduces entropy

conditional  
entropy is an  
average over the  
conditioned  
variables

since

$$\begin{aligned}
 H(Y|X) &= E[-\log p(Y|X)] \\
 &= \sum_{x,y} p(x,y)(-\log p(y|x)) \\
 &= \sum_x p(x) \sum_y p(y|x)(-\log p(y|x)) \\
 &= \sum_x p(x)H(Y|x).
 \end{aligned}$$

As shown in (4.7), the conditional entropy  $H(Y|X)$  is essentially the average of  $H(Y|x)$  over all possible values of  $x$ .

## 4.2.2 Conditional entropy and compression with side information

Just like entropy can be considered as the minimum number of bits required to represent a DMS losslessly, we can consider conditional entropy as the minimum number of bits required to represent a DMS with *side information* losslessly.

Let's consider a joint DMS  $(X, Y)$ , and say the side information (a.k.a. *helper information*)  $Y$  is available freely at both the encoder and decoder and we only need to compress  $X$  as shown in Fig. 4.3. We can argue that the average bits required is  $H(X|Y)$  using the LLN as in the original Source Coding Theorem.

For any particular  $y$ , let's group all  $x$  values that come with this  $y$ . Then, by the LLN, we can encode all these  $x$  values at the rate  $E[-\log p(X|y)] \triangleq H(X|y)$  bits per sample.

As for the entire sequence, a fraction  $p(y)$  of them will have the same  $y$ . So the overall required rate is the weighted sum  $\sum_{y \in \mathcal{Y}} p(y)H(X|y)$ , which is just equal to  $H(X|Y)$ .

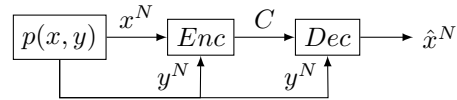


Figure 4.3: Compressing  $X$  with side information  $Y$  given at both encoder and decoder requires on average  $H(X|Y)$  bits

## 4.2.3 Chain rule

Recall that

$$H(X, Y) = H(X) + H(Y|X), \quad (4.8)$$

we can extend this to the case with more variables as follows,

$$\begin{aligned}
 H(X_1, X_2, \dots, X_N) &= H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \dots \\
 &\quad + H(X_N|X_1, X_2, \dots, X_{N-1}).
 \end{aligned}$$

The interpretation of the above equation is very natural. The total uncertainty of  $X_1, X_2, \dots, X_N$  is equal to the uncertainty of  $X_1$  plus the uncertainty of  $X_2$  given  $X_1$ , plus the uncertainty of  $X_3$  given  $X_1$  and  $X_2$ , and so on.

**source coding  
with side  
information**

**chain rule of  
entropy**

The expression is not difficult to show; we just need to repeatedly apply (4.8) and a conditional version of (4.8) as below

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z), \quad (4.9)$$

where we will leave the proof of (4.9) to the readers as an exercise. Using (4.8) and (4.9),

$$H(X_1, X_2, \dots, X_N) = H(X_1) + H(X_2, \dots, X_N|X_1) \quad (4.10)$$

$$= H(X_1) + H(X_2|X_1) + H(X_3, \dots, X_N|X_1) \quad (4.11)$$

$$\dots \quad (4.12)$$

$$= H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \dots \\ + H(X_N|X_1, X_2, \dots, X_{N-1}). \quad (4.13)$$

It is easy to show that differential entropy satisfies a similar formula as follows,

$$h(X_1, X_2, \dots, X_N) = h(X_1) + h(X_2|X_1) + h(X_3|X_1, X_2) + \dots \\ + h(X_N|X_1, X_2, \dots, X_{N-1}).$$

#### Example 4.9: Weather and umbrella

Say the joint probability distribution of a day's weather and someone bringing an umbrella is given below:

$$\begin{aligned} \Pr(\text{Rain, With umbrella}) &= 0.2 & \Pr(\text{Rain, No umbrella}) &= 0.1 \\ \Pr(\text{Sunny, With umbrella}) &= 0.2 & \Pr(\text{Sunny, No umbrella}) &= 0.5 \end{aligned}$$

Let's use  $W$  to denote the weather condition and  $U$  to denote if someone brought an umbrella or not. So,

$$W \in \{\text{Rain, Sunny}\} \quad U \in \{\text{With umbrella, No umbrella}\}$$

Let's compute all combinations of (joint, marginal, and conditional) entropies of  $U$  and  $W$ :

- $H(W, U) = -0.2 \log 0.2 - 0.1 \log 0.1 - 0.2 \log 0.2 - 0.5 \log 0.5 = 1.76$  bits.
- $H(W) = -0.3 \log 0.3 - 0.7 \log 0.7 = 0.88$  bits, since  $p_W(\text{Rain}) = 0.2 + 0.1 = 0.3$  and  $p_W(\text{Sunny}) = 0.2 + 0.5 = 0.7$ .
- $H(U) = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.97$  bits, since  $p_U(\text{With umbrella}) = 0.2 + 0.2 = 0.4$  and  $p_U(\text{No umbrella}) = 0.1 + 0.5 = 0.6$ .
- $H(W|U) = H(W, U) - H(U) = 0.79$  bits.
- $H(U|W) = H(W, U) - H(W) = 0.88$  bits.

#### 4.2.4 Converse proofs of source coding theorems

##### Converse proof of source coding theorem

We have shown in the last chapter that a compression scheme exists for coding rates above  $H(X)$  bits per sample. Let's show the converse theorem that if the coding rate is less than  $H(X)$  bits per sample, the recovered source has to be lossy.

We will use a tool known as Fano's inequality. There are different variations of Fano's inequality, **Fano's inequality** often used to show the converse proof of a theorem. We will come across other forms of Fano's inequality later this section and in 6.5.

Denote  $C$  as the compressed input and  $\hat{X}^N$  as the recovered sequence. If there is no error (i.e.,  $\Pr(X^N \neq \hat{X}^N) \rightarrow 0$ ), then  $\frac{1}{N}H(X^N|C) < \epsilon$  for any  $\epsilon > 0$  given a sufficiently large  $N$ . This essentially means that  $\frac{1}{N}H(X^N|C)$  can be made arbitrarily small if there is no decoding error. This makes sense intuitively, as if there is no error, the original input  $X^N$  should be completely determined given  $C$ , thus the conditional entropy should go to zero.

Let's delay the proof of Fano's inequality and complete the converse proof of the source coding theorem first. Note that we will write the code rate slightly differently from before as  $\frac{H(C)}{N}$  rather **converse proof of Source Coding Theorem** than  $\frac{\log \# \text{ messages}}{N}$  since we need to connect the rate directly to  $H(C)$ . This definition is reasonable because it measures the actual information content of the compressed codeword per message symbol. For any arbitrarily small  $\epsilon$ , we have

$$\begin{aligned} R + \epsilon &= \frac{H(C)}{N} + \epsilon \stackrel{(a)}{\geq} \frac{1}{N}[H(C) + H(X^N|C)] \\ &\stackrel{(b)}{=} \frac{1}{N}H(C, X^N) = \frac{1}{N}[H(X^N) + \underbrace{H(C|X^N)}_0] \\ &= H(X), \end{aligned}$$

where (a) is due to Fano's inequality and in (b),  $H(C|X^N)$  equals 0 since  $C$  is deterministic given  $X^N$ .

Since  $\epsilon$  can be made arbitrarily small by choosing a sufficiently large  $N$ , we can conclude that  $R \geq H(X)$  as the reconstruction error rate approaches zero. This is precisely the result we aimed to prove for the converse theorem.

Now, let's show the Fano's inequality.

##### Proof of Fano's inequality

Let's show the statement that  $\frac{1}{N}H(X^N|C) < \epsilon$  for any  $\epsilon > 0$  given a sufficiently large  $N$  if  $\Pr(X^N \neq \hat{X}^N) \rightarrow 0$ . Let's denote  $E$  as the error event so that  $E = 1$  if  $X^N \neq \hat{X}^N$  and 0 otherwise.

Then

$$\begin{aligned}
 H(X^N|C) &\stackrel{(a)}{=} H(E, X^N|C) - \cancel{H(E|C, X^N)} \xrightarrow{0} \\
 &\stackrel{(b)}{=} H(E|C) + H(X^N|E, C) \\
 &\stackrel{(c)}{\leq} 1 + \Pr(E=0) \cancel{H(X^N|C, E=0)} \xrightarrow{0} + \Pr(E=1)H(X^N|C, E=1) \\
 &\stackrel{(d)}{\leq} 1 + \Pr(E=1)H(X^N),
 \end{aligned}$$

where (a) follows from the chain rule and the fact that  $E$  is completely determined given  $C$  and  $X^N$ , (b) is due to the chain rule, (c) follows from the fact that  $H(E|C)$  is bounded by 1 as  $E$  is binary and that we can expand  $H(X^N|E, C)$  over different outcomes of  $E$  as in (4.7), and for (d),  $X^N$  is deterministic given  $C$  when there is no error, and  $H(X^N|C, E=1) \leq H(X^N)$  since conditioning reduces entropy. Thus, as  $\Pr(E=1) \rightarrow 0$ ,  $\frac{1}{N}H(X^N|C) \leq \frac{1}{N} + \Pr(E=1)H(X) < \epsilon$  for a sufficiently large  $N$ .

### Converse proof of source coding with side information

In motivating the conditional entropy, we argued in Section 4.2.2 that we can compress a source  $X$  losslessly with side information  $Y$  (given at both the encoder and decoder) at a rate  $H(X|Y)$ . However, the argument just upper-bounded the uncertainty of  $X$  given  $Y$  by  $H(X|Y)$ . We didn't show that no other scheme can exist to compress  $X$  at a rate below  $H(X|Y)$ . Let's show that here using another version of Fano's inequality as before.

#### Fano for coding with side information

In a nutshell, Fano's inequality states that  $\frac{1}{N}H(X^N|C, Y^N) \rightarrow 0$  as the error rate goes to zero. More precisely, for any  $\epsilon > 0$ , we have  $\frac{1}{N}H(X^N|C, Y^N) \leq \epsilon$  given a sufficiently large  $N$ . This is reasonable since when there is no error, the original  $X^N$  can be recovered perfectly and thus is deterministic given the codeword  $C$  and the side information  $Y^N$ .

The proof of this version of Fano's inequality is very similar to the one discussed earlier; we will leave that to the readers as an exercise. Now, let's continue the converse proof of the source coding

theorem with side information. For any  $\epsilon > 0$ , we have

$$\begin{aligned}
 \frac{1}{N}(H(C) + \epsilon) &\stackrel{(a)}{\geq} \frac{1}{N}(H(C|Y^N) + \epsilon) \\
 &\stackrel{(b)}{\geq} \frac{1}{N}[H(C|Y^N) + H(X^N|C, Y^N)] \\
 &\stackrel{(c)}{=} \frac{1}{N}H(X^N, C|Y^N) \\
 &\stackrel{(d)}{=} \frac{1}{N}[H(X^N|Y^N) + \cancel{H(C|X^N, Y^N)}] \rightarrow 0 \\
 &\stackrel{(e)}{=} \frac{1}{N} \sum_{n=1}^N H(X_n|Y^N, X^{n-1}) \\
 &\stackrel{(f)}{=} \frac{1}{N} \sum_{n=1}^N H(X_n|Y_n) \\
 &\stackrel{(g)}{=} H(X|Y),
 \end{aligned}$$

**converse proof  
of source coding  
with side  
information**

where (a) is because conditioning reduces entropy, (b) comes from Fano's inequality, (c) and (e) are due to the chain rule, for (d), we use the chain rule again and  $H(C|X^N, Y^N) = 0$  since  $C$  is deterministic given input  $X^N$  and  $Y^N$ , (f) comes from the Markov property of  $X_n \leftrightarrow Y_n \leftrightarrow Y^{n-1}, Y_{n+1}^N, X^{n-1}$ , and (g) is because the joint source  $(X, Y)$  is stationary.

### 4.3 KL-Divergence

In many applications, we would like to measure the difference between two probability distributions. One great tool for this purpose is the KL-divergence, which is also known by various other names, **KL-divergence** such as information divergence, discrimination information, and relative entropy in some literature.

#### KL-divergence

For two distributions of a discrete r.v.  $X$ ,  $p(x)$  and  $q(x)$ , we define the KL-divergence as

$$KL(p(x)||q(x)) \triangleq \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$

If  $X$  is a continuous r.v., the summation becomes an integral instead. That is,

$$KL(p(x)||q(x)) \triangleq \int_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx.$$

Roughly speaking, KL-divergence measures the “distance” between two distributions. Note that if  $p(x) = q(x)$  for all  $x$ , then  $KL(p(x)||q(x)) = 0$  as desired.



**KL-divergence is neither symmetric nor a metric**

It is important to note that  $KL(p(x)||q(x)) \neq KL(q(x)||p(x))$  in general, which is clear from the definition. So, KL-divergence is not a metric or distance measure in the strict sense. That's why we call it KL-divergence rather than "KL-distance."



**KL-divergence is not symmetric**

**KL-divergence is non-negative**

As we use KL-divergence to measure the difference between two distributions, it is important that the divergence itself be non-negative. Indeed, we can show this easily as below.

$$\begin{aligned}
 KL(p(x)||q(x)) &= \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \\
 &= - \sum_{x \in \mathcal{X}} p(x) \log \frac{q(x)}{p(x)} \\
 &= - \sum_{x \in \mathcal{X}} \frac{p(x)}{\ln 2} \ln \frac{q(x)}{p(x)} \\
 &\stackrel{(a)}{\geq} - \sum_{x \in \mathcal{X}} \frac{p(x)}{\ln 2} \left( \frac{q(x)}{p(x)} - 1 \right) \\
 &= \frac{1}{\ln 2} \left( \sum_{x \in \mathcal{X}} p(x) - \sum_{x \in \mathcal{X}} q(x) \right) = 0, \quad (4.14)
 \end{aligned}$$

where (a) is due to  $\ln(x) \leq x - 1$ , which can be easily verified as illustrated in Fig. 4.4.

Moreover, the equality only holds when  $x = 1$ .

One can easily show that  $KL(p(x)||q(x)) \geq 0$  even for distributions with continuous r.v.. The proof is almost identical to the above, just replacing summations with integrals. I will leave the proof as an exercise.

**4.3.1 Some applications of KL-divergence**

KL-divergence has many applications. We will just present some in this section.

**Gaussian distribution has highest entropy**

We have derived the expression of entropy for a normal distribution earlier. We also mentioned without proof in Section 4.1.4 that a normal distribution has the highest entropy for a fixed variance. We can provide a proof here now using KL-divergence.

Let's show the more general multivariate case. For all variables with a fixed covariance matrix  $\Sigma$ , without loss of generality, let's restrict the r.v. to zero mean. For convenience, denote  $\mathcal{N}(\mathbf{x}; \mathbf{0}, \Sigma) =$

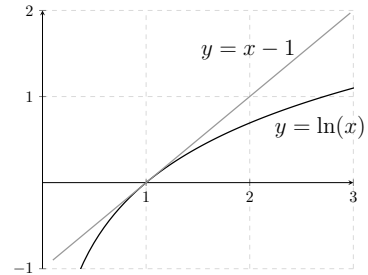


Figure 4.4:  $\ln(x) \leq x - 1$ , and the equality holds only when  $x = 1$

**proof of KL-divergence being non-negative**

$\frac{1}{\sqrt{\det(2\pi\Sigma)}}e^{-\frac{1}{2}\mathbf{x}^\top\Sigma^{-1}\mathbf{x}}$  as  $\phi(\mathbf{x})$ . For any other distribution  $f(\mathbf{x})$  with the same covariance matrix  $\Sigma$ , first note that

$$\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x},$$

because

**Gaussian has  
highest entropy**

$$\begin{aligned} \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} &= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[ -\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2}\mathbf{x}^\top\Sigma^{-1}\mathbf{x} \right] d\mathbf{x} \\ &= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[ -\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} x_i [\Sigma^{-1}]_{i,j} x_j \right] d\mathbf{x} \\ &= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[ -\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} [\Sigma^{-1}]_{i,j} x_i x_j \right] d\mathbf{x} \\ &\stackrel{(a)}{=} \int_{\mathbf{x}} f(\mathbf{x}) \left[ -\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} [\Sigma^{-1}]_{i,j} x_i x_j \right] d\mathbf{x} \\ &= \int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}, \end{aligned}$$

where (a) is due to the fact that  $f(\mathbf{x})$  and  $\phi(\mathbf{x})$  have the same covariance  $\Sigma$ .

Then,

$$\begin{aligned} 0 \leq KL(f\|\phi) &= \int_{\mathbf{x}} f(\mathbf{x}) \log \frac{f(\mathbf{x})}{\phi(\mathbf{x})} d\mathbf{x} = -h(f) - \int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} \\ &= -h(f) - \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} \\ &= -h(f) + h(\phi). \end{aligned}$$

Therefore,  $h(f) \leq h(\phi)$  for any distribution  $f$  with covariance  $\Sigma$ .

#### Example 4.10: Connection with Second law of thermodynamics and central limit theorem

The second law of thermodynamics states that the total entropy of an isolated system tends to increase over time, approaching a maximum value. Entropy, a measure of disorder or randomness, is intimately connected with the statistical behavior of large numbers of particles. In this context, the Central Limit Theorem (CLT) provides a mathematical framework for understanding the evolution of such systems. The CLT asserts that the sum of a large number of independent, identically distributed (i.i.d.) random variables with finite mean and variance converges to a Gaussian distribution, regardless of the original distribution of the variables. This convergence to a Gaussian distribution can be interpreted as the system reaching a state of maximum entropy.

Consider a closed system containing a large number of particles initially localized at the

origin. As time progresses, each particle undergoes Brownian motion, moving randomly from its previous position. The position of each particle at a given time is the sum of many small, independent displacements. According to the CLT, the distribution of the particles' positions will tend towards a Gaussian distribution as the number of steps increases. This spreading of particle positions corresponds to an increase in disorder, or entropy, within the system. Thus, the CLT provides insight into the second law of thermodynamics, which states that the system evolves towards a state of higher entropy, represented by the Gaussian distribution with maximum entropy for a given mean and variance.

### Thiel index

#### Thiel index

The Thiel index is a metric used to measure economic inequality among different groups or within a group of individuals. The idea is to measure the closeness of the distribution of wealth in a population to the distribution of the population itself using KL-divergence. The more similar the distributions, the higher the equality of the population.

For a population split into groups, let  $p_i$  be the economic wealth proportion of group  $i$ , and  $q_i$  be the population size proportion of group  $i$ . Then the Thiel index is simply defined as  $KL(p||q)$ .

Let's look at a more concrete example of a group of  $N$  individuals. If they all have the same wealth, both  $p$  and  $q$  are uniform ( $p_i = q_i = 1/N$ ), thus the Thiel index  $KL(p||q) = 0$ .

If one individual in the group owns everything,  $q$  is uniform but  $p$  is a  $\delta$ -function. Thus the Thiel index  $KL(p||q) = \sum_i p_i \log \frac{p_i}{q_i} = \log \frac{1}{1/N} = \log N$ .

### Cross-entropy and cross-entropy loss

In machine learning, it is often necessary to assess the quality of a trained system. Consider the scores of two models classifying three images into cats, dogs, and ships, as shown below.

Model A						Model B							
computed			targets			correct?	computed			targets			correct?
0.3	0.3	0.4	0	0	1 (cat)	yes	0.1	0.2	0.7	0	0	1 (cat)	yes
0.3	0.4	0.3	0	1	0 (dog)	yes	0.1	0.7	0.2	0	1	0 (dog)	yes
0.1	0.2	0.7	1	0	0 (ship)	no	0.3	0.4	0.3	1	0	0 (ship)	no

Under the *computed* column, the scores for the three classes for each sample are shown. The ground truth is presented in the second column. For example, both models A and B correctly classify the first sample as a cat.

At first glance, both models appear to perform equally well (or poorly), with each making one classification error by misclassifying the last animal into a cat or a dog. However, a closer look suggests that the predictions of the right model are better than those of the left one because its scores are more skewed (indicating higher confidence) when it makes a correct classification.

Rather than evaluating the models solely based on classification error, which simply counts the number of misclassifications, we can achieve a better assessment by treating both the computed results and the target results as distributions and comparing them using KL-divergence. Specifically,

$$\begin{aligned}
 KL(p_{target}||p_{computed}) &= \sum_{group} p_{target}(group) \log \frac{p_{target}(group)}{p_{computed}(group)} \\
 &= -H(p_{target}) - \underbrace{\sum_{group} p_{target}(group) \log p_{computed}(group)}_{\text{cross entropy}}
 \end{aligned}$$

Note that the first term  $H(p_{target})$  does not depend on the model and is therefore irrelevant in evaluating the models. The second term is known as the cross-entropy, which is essentially a *simplified* version of KL-divergence and is used widely in machine learning.

**cross-entropy**

### Cross-entropy

$$\begin{aligned}
 \text{Cross entropy}(p||q) &\triangleq \sum_x p(x) \log \frac{1}{q(x)} = E_p[-\log q(X)] \\
 &= H(p) + KL(p||q)
 \end{aligned}$$

### Text processing and TF-IDF

In text processing, it is common to measure the similarity between two documents  $D_1$  and  $D_2$ .

How should we represent documents? One common approach is the *bag of words* method. This converts a document into a vector of numbers, where each number represents the count of a corresponding word. One can then compare two documents using cross-entropy:

$$\text{Cross-entropy}(p_1||p_2) = \sum_w p_1(w) \log \frac{1}{p_2(w)},$$

where  $p_1$  and  $p_2$  are the word distributions of documents  $D_1$  and  $D_2$ , respectively.

It may also be interesting to compare the word distribution of a document to the word distribution across all documents. Let  $q$  be the word distribution across all documents,

**TF-IDF**

$$\begin{aligned}
 \text{Cross-entropy}(p_1||q) &= \sum_w p_1(w) \log \frac{1}{q(w)} \\
 &= \sum_w \underbrace{\frac{\# \text{ of occurrences of } w \text{ in } D_1}{\text{total } \# \text{ of words in } D_1}}_{\text{TF-IDF}(w)} \log \frac{\text{total } \# \text{ of documents}}{\# \text{ of documents with } w},
 \end{aligned}$$

where  $\text{TF-IDF}(w)$ , short for term frequency-inverse document frequency, reflects how important the word  $w$  is to the target document and is used in search engines.

### Maximum likelihood estimation

Consider fitting  $\{x^1, x^2, \dots, x^m\}$  drawn from  $p_{\text{data}}(x)$  to a model  $p_\theta$  using maximum likelihood estimation. Consequently, we have

**ML minimizes cross-entropy and maximizes KL-divergence**

$$\begin{aligned} \theta^* &= \arg \max_{\theta} \prod_{i=1}^m p_\theta(x^i) = \arg \max_{\theta} \log \prod_{i=1}^m p_\theta(x^i) \\ &= \arg \max_{\theta} \sum_{i=1}^m \log p_\theta(x^i) \\ &\stackrel{(a)}{=} \arg \max_{\theta} \frac{1}{m} \sum_{i=1}^m \log p_\theta(x^i) \stackrel{(b)}{\approx} \arg \max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} [\log p_\theta(x)] \\ &= \arg \max_{\theta} \int_x p_{\text{data}}(x) \log p_\theta(x) dx \\ &\stackrel{(c)}{=} \arg \min_{\theta} \left[ - \int_x p_{\text{data}}(x) \log p_\theta(x) dx \right] \end{aligned} \tag{4.15}$$

$$= \arg \min_{\theta} \text{Cross entropy}(p_{\text{data}} \| p_\theta), \tag{4.16}$$

where (a) is due to scaling not affecting the maximization process, (b) comes from the law of large numbers (LLN), and (c) is due to maximizing  $f(x)$  being equivalent to minimizing  $-f(x)$ .

We can also append a dummy term to (4.15) to have:

$$\begin{aligned} \theta^* &= \arg \min_{\theta} \left[ - \int_x p_{\text{data}}(x) \log p_\theta(x) dx \right] \\ &= \arg \min_{\theta} \left[ - \int_x p_{\text{data}}(x) \log p_\theta(x) dx \right] + \int_x p_{\text{data}}(x) \log p_{\text{data}}(x) dx \\ &= \arg \min_{\theta} \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_\theta(x)} dx = \arg \min_{\theta} KL(p_{\text{data}} \| p_\theta) \end{aligned} \tag{4.17}$$

From (4.16) and (4.17), we see that fitting samples drawn from a  $p_{\text{data}}$  distribution using maximum likelihood estimation is the same as minimizing the cross-entropy or the KL-divergence between the data distribution and the model distribution.

### Model evidence with latent variables and Evidence lower bound (ELBO)

Given observations  $x$  and a model parameterizable by parameter  $\theta$ , we often want to optimize the model by fitting  $\theta$  with the observation  $x$ . That is, we aim to maximize  $p(x; \theta)$ . However, often there are hidden (latent) variables  $z$  involved, and we have  $p(x, z; \theta) = p(x|z)p(z; \theta)$ , where  $p(x|z)$  is often easily accessible in the problem (for example, consider the mixture model below). So instead, to fit  $\theta$  with the observation, we may solve:

$$\max \log p(x; \theta) = \max \log \sum_z p(x|z)p(z; \theta),$$

where maximizing  $\log p$  is the same as maximizing  $p$  since the log function is monotonic. Often, we want to fit  $\theta$  with more than one observation; say we have  $x_1, x_2, \dots, x_N$ , which are independent given  $\theta$ . The above becomes:

$$\max \log p(x_1, \dots, x_N; \theta) = \max \log \sum_{z_1, \dots, z_N} \prod_{i=1}^N p(x_i | z_i; \theta) p(z_i; \theta), \quad (4.18)$$

which quickly becomes intractable as  $N$  increases, since we are summing over a space of  $|\mathcal{Z}|^N$ .

To reduce the computation, we introduce a dummy distribution  $q(z^N)$  and write

**evidence lower bound**

$$\begin{aligned} \log p(x_1, \dots, x_N; \theta) &= \sum_{z^N} q(z^N) \log p(x_1, \dots, x_N; \theta) \\ &= \sum_{z^N} q(z^N) \log \frac{p(x^N, z^N; \theta)}{p(z^N | x^N; \theta)} \\ &= \sum_{z^N} q(z^N) \log \frac{p(x^N, z^N; \theta)}{q(z^N)} \frac{q(z^N)}{p(z^N | x^N; \theta)} \\ &= \underbrace{\sum_{z^N} q(z^N) \log p(x^N, z^N; \theta)}_{\text{ELBO}} - H(q(z^N)) + KL(q(z^N) || p(z^N | x^N; \theta)), \quad (4.19) \end{aligned}$$

where ELBO stands for the Evidence Lower Bound. ELBO is a lower bound since the KL-divergence in the last equation is always non-negative.

Introducing  $q(z^N)$  and leveraging ELBO allows us to solve for  $\theta$  in an iterative manner, leading to the powerful Expectation-Maximization (EM) algorithm as follows. On one hand, we want to **EM algorithm** minimize the KL-divergence term, which leads to:

$$q(z^N) \leftarrow \prod q_i(z_i), \quad (4.20)$$

where  $q_i(z_i) = p(z_i | x_i; \theta)$  since each pair of  $x_i, z_i$  are independent given  $\theta$ . This step is often known as the Expectation step (E-step)<sup>1</sup>.

On the other hand, the step maximizing ELBO is often known as the Maximization step (M-

---

<sup>1</sup>The reason for the name E-step is not clear under this derivation, but it is so-called because the step can also be derived as computing the expectation of the complete data likelihood. An imprecise but helpful mnemonic is that this step computes  $q$ , which will be used as expected weights needed to compute  $\theta$ .

step) for obvious reasons, which is the same as maximizing<sup>2</sup>:

$$\begin{aligned}
\sum_{z^N} q(z^N) \log p(x^N, z^N; \theta) &\simeq \sum_{z^N} q(z^N) \log \prod_{i=1}^N p(x_i, z_i; \theta) \\
&= \sum_{z^N} \prod_{j=1}^N q_j(z_j) \sum_{i=1}^N \log p(x_i, z_i; \theta) \\
&= \sum_{i=1}^N \sum_{z_i} q_i(z_i) \log p(x_i, z_i; \theta) \sum_{z^N \setminus z_i} \prod_{j \neq i} q_j(z_j) \\
&= \sum_{i=1}^N \sum_{z_i} q_i(z_i) \log p(x_i, z_i; \theta) \underbrace{\prod_{j \neq i} \sum_{z_j} q_j(z_j)}_1 \\
&= \sum_{i=1}^N \sum_z q_i(z) \log p(x_i, z; \theta). \tag{4.21}
\end{aligned}$$

A concrete example is given below and it should makes things clearer.

#### Example 4.11: EM algorithm for mixture of Gaussian model

Consider a careless teacher who measures the height of a class of students without labeling the data with names. After measuring the data, the teacher wants to estimate the average heights of the male and female students separately. How can he do that without knowing the gender of each height data point?

Let  $x_1, x_2, \dots, x_N$  be the observed heights, and let  $z_i \in \{M, F\}$  be the latent gender variable for  $x_i$ . The parameter  $\theta = (\mu_M, \sigma_M, \mu_F, \sigma_F, w_M, w_F)$  contains the means and standard deviations of the heights of male and female students.  $w_M$  and  $w_F$  are the fractions of male and female students in the class, thus  $w_M + w_F = 1$ . We assume both male and female populations are Gaussian distributed. So,

$$p(x, z; \theta) = w_z \mathcal{N}(x; \mu_z, \sigma_z)$$

The goal is to estimate  $\theta = (\mu_M, \sigma_M, \mu_F, \sigma_F, w_M, w_F)$  with the observed data using the Expectation-Maximization (EM) algorithm.

E-step:

For the E-step, the gender distribution of  $x_i$  can be estimated as:

$$q_i(z) \leftarrow p(z|x_i; \theta) = \frac{w_z \mathcal{N}(x_i; \mu_z, \sigma_z)}{w_M \mathcal{N}(x_i; \mu_M, \sigma_M) + w_F \mathcal{N}(x_i; \mu_F, \sigma_F)}$$

for  $z \in \{M, F\}$ .

M-step:

For the M-step, we maximize the Evidence Lower Bound (ELBO), which is equivalent to

<sup>2</sup> $\simeq$  indicates that the terms may not be equal but are equivalent terms to maximize

maximizing:

$$\sum_{i=1}^N \sum_z q_i(z) \log p(x_i, z; \theta)$$

This expression can be expanded as:

$$\sum_{i=1}^N \sum_z q_i(z) \left( \log w_z - \log \sqrt{2\pi\sigma_z^2} - \frac{(x_i - \mu_z)^2}{2\sigma_z^2} \right). \quad (4.22)$$

To maximize this with respect to  $\mu_M, \mu_F, \sigma_M, \sigma_F$ , and  $w_M, w_F$  (keeping in mind the constraint  $w_M + w_F = 1$ ), we introduce a Lagrange multiplier to handle the constraint<sup>a</sup>. After solving the maximization problem, the updates for the parameters are given by

$$w_z \leftarrow \frac{1}{N} \sum_{i=1}^N q_i(z) \quad (4.23)$$

$$\mu_z \leftarrow \frac{\sum_{i=1}^N q_i(z) x_i}{\sum_{i=1}^N q_i(z)} \quad (4.24)$$

$$\sigma_z^2 \leftarrow \frac{\sum_{i=1}^N q_i(z) (x_i - \mu_z)^2}{\sum_{i=1}^N q_i(z)} \quad (4.25)$$

These equations provide the necessary updates for the parameters during each iteration of the EM algorithm, eventually converging to estimates of the average heights of male and female students, along with their respective variances and proportions in the class.

<sup>a</sup>Be careful that we can't take the derivative of (4.22) and set it to zero directly since we have the additional constraint  $w_M + w_F = 1$ . We have to introduce a Lagrange multiplier  $\lambda$  and take the derivative of (4.22) +  $\lambda(w_F + w_M)$  and set it to 0 instead.

## 4.4 Mutual Information

As  $H(X)$  is equivalent to the uncertainty of  $X$  and  $H(X|Y)$  is the remaining uncertainty of  $X$  knowing  $Y$ , we expect that  $H(X) \geq H(X|Y)$  and the difference  $H(X) - H(X|Y)$  should quantify the amount of the uncertainty of  $X$  that can be resolved by  $Y$ . If we interpret  $H(X)$  to be the amount of information revealed by knowing the outcome of  $X$  and  $H(X|Y)$  to be the amount of information revealed by knowing the outcome of  $X$  given that we already know  $Y$ , then  $H(X) - H(X|Y)$  can be interpreted as the information shared by  $X$  and  $Y$ . Therefore, we also call this difference “mutual information,” where we usually denote it by  $I(X; Y)$ <sup>3</sup>.

Similarly, we can define the “conditional mutual information” as the difference of  $H(X|Z)$  and  $H(X|Y, Z)$  and interpret that as the information shared between  $X$  and  $Y$  given  $Z$ .

<sup>3</sup>Some literature may use notations such as  $I(X : Y)$  and even  $I(X, Y)$ . But the latter is a confusing one that should be avoided. Since if we have more than two variables, say  $I(X, Y, Z)$ , it becomes unclear if we mean  $I(X; Y, Z)$  or  $I(X, Y; Z)$ .



### Mutual information and conditional mutual information

Mutual information between  $X$  and  $Y$ :  $I(X; Y) \triangleq H(X) - H(X|Y)$

Conditional mutual information between  $X$  and  $Y$  given  $Z$ :  $I(X; Y|Z) \triangleq H(X|Z) - H(X|Y, Z)$

#### 4.4.1 Properties of Mutual Information

Let's look into some properties of mutual information in this section.

##### Mutual Information is Symmetric and Non-Negative

For any  $X$  and  $Y$ , we can easily show that

$$I(X; Y) = KL(p(x, y) \| p(x)p(y)) \quad (4.26)$$

because

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) = E[-\log p(X)] - E[-\log p(X|Y)] \\ &= -\sum_x p(x) \log p(x) + \sum_{x,y} p(x, y) \log p(x|y) \\ &\stackrel{(a)}{=} -\sum_{x,y} p(x, y) \log p(x) + \sum_{x,y} p(x, y) \log p(x|y) \\ &= \sum_{x,y} p(x, y) \log \frac{p(x|y)}{p(x)} \\ &= \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= KL(p(x, y) \| p(x)p(y)), \end{aligned}$$

where (a) is due to reverse marginalization. Consequently, we have  $I(X; Y) = I(Y; X)$ . We can also see that  $I(X; Y)$  is symmetric because

$$I(X; Y) = H(X) - H(X|Y) \stackrel{(a)}{=} H(X) + H(Y) - H(X, Y),$$

where (a) is from the chain rule of joint entropy.

Moreover, since KL-divergence is non-negative as shown in Section 4.3,  $I(X; Y) \geq 0$ .

##### Conditional mutual information is symmetric and non-negative

Similarly, we can show that

$$I(X; Y|Z) = \sum_z p(z) KL(p(x, y|z) \| p(x|z)p(y|z)). \quad (4.27)$$

mutual  
information and  
conditional  
information

mutual  
information is  
symmetric and  
non-negative

The proof is very similar to the unconditional case and we will leave it to the readers as an exercise.

Consequently, we also have  $I(X; Y|Z) = I(Y; X|Z) \geq 0$  from (4.27). Just as in the unconditional case, we can see that  $I(X; Y|Z)$  is symmetric as we can write

$$I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = H(X|Z) + H(Y|Z) - H(X, Y|Z).$$

**conditional  
mutual  
information is  
symmetric and  
non-negative**

### Conditioning reduces entropy

Given more information, the residual information (uncertainty) should decrease. More precisely,

$$H(X) \geq H(X|Y) \quad \text{and} \quad H(X|Y) \geq H(X|Y, Z)$$

**proof of  
conditioning  
reduces entropy**

This is obvious from our previous discussion since  $H(X) - H(X|Y) = I(X; Y) \geq 0$  and  $H(X|Y) - H(X|Y, Z) = I(X; Z|Y) \geq 0$ . This precisely is a proof that conditioning reduces entropy, as described in (4.6).

### (Conditional) independence and (conditional) mutual information

From (4.26), we can readily see that  $I(X; Y) = 0$  if and only if  $X \perp Y$ . Moreover, if  $X \perp Y|Z$ , we have  $p(x, y|z) = p(x|z)p(y|z)$  for all  $z$ , and so from (4.27), we have  $I(X; Y|Z) = 0$ . Conversely, if

$$I(X; Y|Z) = \sum_z p(z) KL(p(x, y|z) \| p(x|z)p(y|z)) = 0,$$

since KL-divergence and probabilities are non-negative, this implies  $p(x, y|z) = p(x|z)p(y|z)$  for all  $z$  such that  $p(z) > 0$ . Therefore, we have  $X \perp Y|Z$ .

**(conditional)  
independence if  
and only if  
(conditional)  
mutual  
information is  
zero**

Actually, the results should be hardly surprising. If there is no shared information between  $X$  and  $Y$ , they should be independent! Similarly, if there is no shared information between  $X$  and  $Y$  given  $Z$ , they should be conditionally independent given  $Z$ .

### Chain rule for mutual information

Using the chain rule for joint entropy, we can derive a similar chain rule for mutual information

$$\begin{aligned} I(X_1, X_2, \dots, X_N; Y) &= H(X_1, X_2, \dots, X_N) - H(X_1, X_2, \dots, X_N|Y) \\ &= \sum_{i=1}^N H(X_i|X^{i-1}) - H(X_i|X^{i-1}, Y) \\ &= \sum_{i=1}^N I(X_i; Y|X^{i-1}) \end{aligned}$$

**chain rule for  
mutual  
information**

This chain rule for mutual information allows us to decompose the mutual information between multiple random variables and a single random variable into a sum of conditional mutual information.

**Data processing inequality****data processing inequality**

The data processing inequality is a fundamental result in information theory that describes the limits of information processing. It states that processing or transforming data cannot increase the amount of information available for estimating a random variable.

Formally, if random variables  $X, Y, Z$  form a Markov chain ( $X \leftrightarrow Y \leftrightarrow Z$ ), then the data processing inequality can be expressed as:

$$I(X; Y) \geq I(X; Z).$$

*Proof.*

$$\begin{aligned} I(X; Y) &= I(X; Y, Z) - I(X; Z|Y) \\ &= I(X; Y, Z) \quad (\text{since } X \leftrightarrow Y \leftrightarrow Z) \\ &= I(X; Z) + I(X; Y|Z) \\ &\geq I(X; Z). \end{aligned}$$

□

In essence, the data processing inequality asserts that processing or transforming data cannot increase the amount of information available for estimating a random variable. This means that the original observation  $Y$  contains the most information about  $X$ , and any processing or transformation of  $Y$  into  $Z$  cannot increase this information. While this result assumes the existence of an optimal estimator, in practice, data processing may be necessary to facilitate estimation when dealing with noisy or complex data.

**4.4.2 Mutual information for continuous variables**

For continuous random variables  $X, Y, Z$ , we can define the mutual information between  $X$  and  $Y$  as

$$I(X; Y) = h(X) - h(X|Y)$$

and the conditional mutual information between  $X$  and  $Y$  given  $Z$  as

$$I(X; Y|Z) = h(X|Z) - h(X|Y, Z).$$

Interestingly, all the properties of mutual information described in Section 4.4.1 still hold true for continuous random variables. This includes the non-negativity of mutual information, i.e.,  $I(X; Y) \geq 0$ . This result may be especially surprising, since unlike the discrete case, differential entropies do not need to be non-negative. However, the non-negativity of mutual information follows from the fact that conditioning reduces entropy, i.e.,  $h(X|Y) \leq h(X)$ , and similarly for conditional entropies.

This result has important implications for continuous random variables, as it provides a measure of the dependence between variables that is always non-negative, even when the individual entropies may be negative.

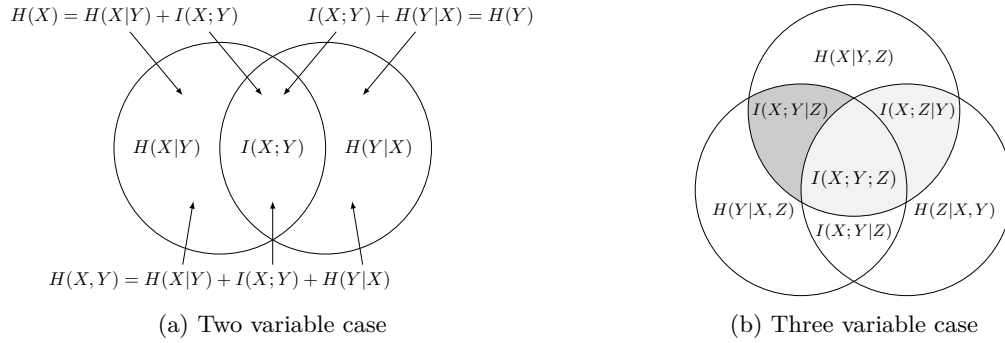


Figure 4.5: A Venn diagram is a powerful tool for visualizing and memorizing entropy formulas. The intersection of regions corresponds to mutual information, while conditioning on a variable excludes the area of the respective conditioned variable. For example, in the right figure, the dark grey area represents  $I(X;Y|Z)$  and the light grey area represents  $I(X;Z)$ . From the figure, we can deduce that  $H(X) = H(X|Y,Z) + I(X;Y|Z) + I(X;Z)$ . It is important to note that the intersection of all three circles, sometimes denoted as  $I(X;Y;Z)$ , is not defined in many texts because it lacks an intuitive physical meaning and does not need to be positive. Therefore, we must be cautious in drawing conclusions when  $I(X;Y;Z)$  is involved. For example, the inequality  $I(X;Y) \geq I(X;Y|Z)$  is generally not true, even though the figure may suggest otherwise.

## 4.5 Venn diagram for information measures

There are quite a few formulas involved for information measures in this chapter. One great memorizing tool is to leverage a Venn diagram, as shown in Figure 4.5.

In Figure 4.5a, we show a two-variable case with variables  $X$  and  $Y$ . The two circles correspond to  $H(X)$  and  $H(Y)$ . The intersection corresponds to the mutual information  $I(X;Y)$ . The left crescent corresponds to  $H(X|Y)$ , noting that when we condition on a variable, we exclude the respective region of the variable. Similarly, the right crescent corresponds to  $H(Y|X)$ . With the respective areas in place, we can easily write out the formulas as shown in Figure 4.5a.


Now, let's consider adding an additional variable  $Z$ , as shown in Figure 4.5b. The same rules described above still hold. For example, the dark gray region corresponds to  $I(X;Y|Z)$  since it is the intersection of regions corresponding to  $X$  and  $Y$ , excluding the region corresponding to  $Z$ . The light gray area represents  $I(X;Z)$  since it is the intersection of regions corresponding to  $X$  and  $Z$ . Consequently, we have:

$$H(X) = H(X|Y,Z) + I(X;Y|Z) + I(X;Z), \quad (4.28)$$

$$\Rightarrow H(X) - H(X|Y,Z) = I(X;Y|Z) + I(X;Z), \quad (4.29)$$

$$\Rightarrow I(X;Y,Z) = I(X;Y|Z) + I(X;Z) \quad (4.30)$$

which is indeed true as the last line is just chain rule of mutual information.

Unlike the two-variable case, we have an additional area in the middle that we have not yet discussed. We denote it as  $I(X;Y;Z)$ , even though this quantity is not unanimously defined. It is important to note that  $I(X;Y;Z)$  does not need to be non-negative, unlike the areas of other regions. Therefore, we need to be very careful in drawing conclusions when  $I(X;Y;Z)$  is involved.   $I(X;Y;Z)$  can be negative

For example, in general, we do not have  $I(X; Y) \geq I(X; Y|Z)$ , even though the figure may suggest so.

In fact,  $I(X; Y)$  can be less than  $I(X; Y|Z)$  when  $X$  and  $Y$  are independent but not conditionally independent given  $Z$  (see Section 2.4.3). Even though we cannot determine whether  $I(X; Y)$  or  $I(X; Y|Z)$  is larger from the Venn diagram alone, since  $I(X; Y) - I(X; Y|Z) = I(X; Y; Z) = I(X; Z) - I(X; Z|Y)$ , we have  $I(X; Y) \geq I(X; Y|Z)$  if and only if  $I(X; Z) \geq I(X; Z|Y)$ . And this is not immediately apparent.

## 4.6 Summary

- Conditioning reduces entropy
- Chain rules:
  - $H(X, Y, Z) = H(Z) + H(Y|X) + H(Z|X, Y)$
  - $H(X, Y, U|V) = H(X|V) + H(Y|X, V) + H(U|Y, X, V)$
  - $I(X, Y, Z; U) = I(X; U) + I(Y; U|X) + I(Z; U|X, Y)$
  - $I(X, Y, Z; U|V) = I(X; U|V) + I(Y; U|V, X) + I(Z; U|V, X, Y)$
- Data processing inequality: if  $X \perp Y|Z$ ,  $I(X; Y) \geq I(X; Z)$
- Independence and mutual information:
  - $X \perp Y \Leftrightarrow I(X; Y) = 0$
  - $X \perp Y|Z \Leftrightarrow I(X; Y|Z) = 0$
- KL-divergence:  $KL(p||q) \triangleq \sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0$

## 4.7 Exercise

1. Show  $H(X, Y|Z) = H(X|Z) + H(Y|X, Z)$  as described in (4.9).
2. Prove Fano's inequality used to show the converse proof of the source coding theorem with side information, where side information  $Y$  is given to both the encoder and the decoder as shown in Figure 4.3. More precisely, please show that for any  $\epsilon > 0$ , we have  $\frac{1}{N}H(X^N|C, Y^N) \leq \epsilon$  given a sufficiently large  $N$ .
3. Show that given any two PDFs,  $p(x)$  and  $q(x)$  of a continuous variable  $X$ , we have  $KL(p||q) \geq 0$ . Hint: the proof is very similar to the one for the discrete case as in (4.14).
4. Finish the derivation of the EM-algorithm by showing the update rules described in (4.23)-(4.25).
5. For multivariate Gaussian mixture models with vector observations  $\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N$ , each cluster  $z$  will behave like  $\sim \mathcal{N}(\boldsymbol{\mu}_z, \Sigma_z)$ . Show that the update for mean  $\boldsymbol{\mu}_z$  and covariance matrix  $\Sigma_z$  are respectively

$$\boldsymbol{\mu}_z \leftarrow \frac{\sum_{i=1}^N q_i(z) \mathbf{x}_i}{\sum_{i=1}^N q_i(z)} \quad (4.31)$$

and

$$\Sigma_z \leftarrow \frac{\sum_{i=1}^N q_i(z)(\mathbf{x}_i - \boldsymbol{\mu}_i)(\mathbf{x}_i - \boldsymbol{\mu}_i)^\top}{\sum_{i=1}^N q_i(z)} \quad (4.32)$$

6. Show that  $I(X; Y|Z) = \sum_z p(z)KL(p(x, y|z)||p(x|z)p(y|z))$  as in (4.27).

# Chapter 5

## Interlude: Some IT application examples

In this chapter, we will apply the information measures learned in previous chapters to two interesting applications: Shannon's perfect secrecy in cryptography and decision trees in classification.

### 5.1 Shannon's Perfect Secrecy

Imagine yourself as a spy, tasked with transmitting a sensitive message to a fellow agent while maintaining utmost secrecy from adversaries and the public. This classic conundrum in cryptography is often illustrated through a simple model. In this scenario, the original message, known as the **plaintext**  $M$ , is encrypted into a **ciphertext**  $C$  using a specific **key**  $K$ . This key is also shared with the intended recipient, enabling them to decrypt the message and access its contents.

Shannon's seminal result in this domain states that to ensure perfect secrecy, the following fundamental inequality must be satisfied

$$H(M) \leq H(K). \quad (5.1)$$

This inequality signifies that the entropy of the key should be at least the entropy of the message.

Without loss of generality, we assume a **non-probabilistic** encryption scheme. In such a scheme, each plaintext message is uniquely mapped to a ciphertext given a fixed key, eliminating any ambiguity during the decoding process. Consequently, the conditional entropy

$$H(M|C, K) = 0. \quad (5.2)$$

A crucial requirement for perfect secrecy is the independence of the ciphertext and the plaintext message. Ideally, one should not be able to infer any information about the message from the ciphertext. Hence,  $C$  and  $M$  should be independent variables. This leads to the conclusion that  $I(C; M) = 0$ . Therefore, we have

$$H(M) = H(M|C) \quad (5.3)$$

since  $H(M) \stackrel{(a)}{=} H(M|C) + I(C; M) \stackrel{(b)}{=} H(M|C)$ , where (a) is from the definition of mutual information and (b) is due to the independence of  $M$  and  $C$ .

**plaintext,  
ciphertext, and  
key  
Shannon's  
perfect secrecy**

**ciphertext and  
plaintext should  
be independent**

Next, note that we also have

$$H(M|C) \leq H(K|C). \quad (5.4)$$

since

$$\begin{aligned} H(M|C) &\stackrel{(a)}{\leq} H(M|C) + H(K|M, C) \\ &\stackrel{(b)}{=} H(M, K|C) \\ &\stackrel{(c)}{=} H(K|C) + H(M|K, C) \\ &\stackrel{(d)}{=} H(K|C), \end{aligned}$$

**proof of  
Shannon's  
perfect secrecy**

where (a) is due to the non-negativity property of entropy, (b) and (c) follow from the chain rule of entropy, and (d) is due to  $H(M|K, C) = 0$  for a non-probabilistic encryption scheme.

Combining (5.3) and (5.4), we immediately have

$$H(M) = H(M|C) \leq H(K|C) \stackrel{(a)}{\leq} H(K), \quad (5.5)$$

where (a) is due to the property that conditioning reduces entropy.

Shannon's concept of perfect secrecy presents a rather pessimistic outlook because to send a message with perfect secrecy, a key of at least the same size as the message must be securely distributed. The classic approach for achieving this is the one-time pad, which modulates the message with a key of the same size. In practice, Shannon's perfect secrecy is too expensive in terms of key consumption.

Rather than striving for perfect secrecy, it is often sufficient for the message to be computationally secure. Computational security means that while it is theoretically possible to break the encryption, it would require an infeasible amount of computational resources, such as time or processing power, to do so. In other words, with current technology and knowledge, it would take an impractically long time for an attacker to decrypt the message without the key. This form of security relies on the assumption that certain mathematical problems (such as factoring large prime numbers or computing discrete logarithms) are difficult to solve.

Modern cryptographic techniques, such as RSA, AES, and ECC, aim to provide computational security. These methods do not require the key to be as large as the message, making them more practical for everyday use. While they do not guarantee absolute security, they provide a level of protection that is considered sufficient to keep information safe against all known attacks within a reasonable time frame.

## 5.2 Identifying Vampires

Consider a world where real vampires are lurking among us. It is crucial to effectively identify vampires so we can avoid them.<sup>1</sup> Assume we have collected a small dataset of vampires and non-vampires based on various **attributes**, as shown in Table 5.1. These attributes include: Does **attribute** the individual have a shadow (yes, no, or unsure)? Does the individual like garlic? What is the individual's complexion like? Does the individual have an accent?

<sup>1</sup>This interesting example is borrowed from the late Patrick Winston at MIT.



Identifying vampires given these attributes presents several challenges. Firstly, we often deal with non-numerical data, which limits the direct use of some tools such as regression. Next, the relevance of certain information can vary; some attributes may be critical in certain contexts but irrelevant in others. Additionally, obtaining these attributes can be costly (and potentially life threatening), making it essential to minimize the number of attributes that need to be examined.

Vampire?	Shadow?	Garlic?	Complexion?	Accent?
No	?	Yes	Pale	None
No	Yes	Yes	Ruddy	None
Yes	?	No	Ruddy	None
Yes	No	No	Average	Heavy
Yes	?	No	Average	Odd
No	Yes	No	Pale	Heavy
No	No	No	Average	Heavy
No	?	Yes	Ruddy	Odd

Table 5.1: Romanian Data Base. The data entry with unsure shadow attribute is highlighted.

### 5.2.1 Picking good attributes based on counting

To facilitate a clearer understanding of the dataset presented in Table 5.1, we illustrate the distribution of vampires and non-vampires based on each attribute's outcomes using decision trees, as shown in Figure 5.1. In these trees, the symbols have the following meanings:

- '+' denotes the presence of a vampire
- '-' denotes the presence of a regular folk (i.e., a non-vampire)

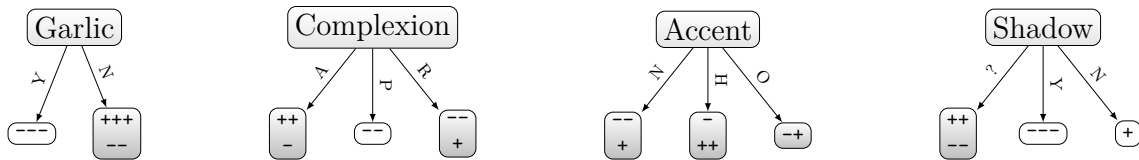


Figure 5.1: Test trees for different attributes are shown, where each '+' sign indicates a vampire count, and each '-' sign indicates a non-vampire count.

Each attribute produces a set of outcomes that are either homogeneous (indicated in white, signifying a high probability of being or not being a vampire) or mixed. The total size of these homogeneous sets can intuitively indicate the efficacy of each test, with larger sizes suggesting better performance. For example, the total homogeneous set sizes are 4 for Shadow, 3 for Garlic, 2 for Complexion, and 0 for Accent, suggesting that Shadow is likely a more effective attribute than Accent for identifying vampires.

However, this simple approach has two limitations:

1. As the dataset size increases, the probability of obtaining homogeneous sets decreases.

2. This approach neglects almost homogeneous sets, even if only one sample differs from the rest.

### 5.2.2 Information theoretic approach

To determine the best attribute using information theory, we can treat the dataset as a representative sample of the real-world distribution. We consider the variable  $V$  to indicate whether a sampled individual is a vampire, and the attributes  $G, C, A$ , and  $S$  (Garlic, Complexion, Accent, and Shadow) as variables as well. We evaluate the importance of each attribute based on their estimated empirical distributions.

Let's denote the entropy of  $V$  as  $H(V)$ , which represents the uncertainty about an individual being a vampire. From the dataset with 3 vampires and 5 non-vampires, we calculate:

$$H(V) = -\frac{3}{8} \log_2 \frac{3}{8} - \frac{5}{8} \log_2 \frac{5}{8} \approx 0.95.$$

To compare attributes, we use conditional entropies. For instance, if  $H(V|\text{attribute}) = 0$ , **conditional** knowing the attribute makes us certain about the individual being a vampire or not. In general, **entropy should** smaller conditional entropies indicate greater certainty in our classification based on the attribute. **be minimized**

Let's assess the conditional entropies individually using our example data. We'll begin with the Garlic attribute.

For Garlic, we calculate the conditional entropy  $H(V|G)$ , which represents the uncertainty about an individual being a vampire given their preference for garlic. Using the dataset, we can estimate this value.

$$H(V|G) \stackrel{(a)}{=} Pr(G = Y)H(V|G = Y) + Pr(G = N)H(V|G = N) \quad (5.6)$$

$$\stackrel{(b)}{=} \frac{3}{8}H(V|G = Y) + \frac{5}{8}H(V|G = N) \quad (5.7)$$

$$\stackrel{(c)}{=} \frac{3}{8} \cdot 0 + \frac{5}{8}H(V|G = N) \quad (5.8)$$

$$\stackrel{(d)}{=} \frac{5}{8}H\left(\frac{3}{5}\right) = \frac{5}{8} \cdot 0.97 = 0.61, \quad (5.9)$$

where (a) follows from (4.7), (b) is derived from the fact that 3 out of 8 individuals like garlic, (c) results from the observation that all individuals who like garlic are not vampires, and (d) follows from the fact that 3 out of 5 non-garlic lovers are vampires, and recall that  $H(p)$  is a shorthand notation for  $-p \log_2 p - (1-p) \log_2 (1-p)$ .

As for complexion, we have

$$\begin{aligned} H(V|C) &\stackrel{(a)}{=} p_C(A)H(V|C = A) + p_C(P)H(V|C = P) + p_C(R)H(V|C = R) \\ &= \frac{3}{8}H\left(\frac{1}{3}\right) + \frac{2}{8}H(0) + \frac{3}{8}H\left(\frac{1}{3}\right) = 0.69, \end{aligned} \quad (5.10)$$

where  $p_C(x) = Pr(C = x)$  in (a).

And for accent,

$$\begin{aligned}
 H(V|A) &= p_A(N)H(V|A = N) + p_A(H)H(V|A = H) + p_A(O)H(V|A = O) \\
 &= \frac{3}{8}H\left(\frac{1}{3}\right) + \frac{3}{8}H\left(\frac{1}{3}\right) + \frac{2}{8}H\left(\frac{1}{2}\right) = 0.94.
 \end{aligned}
 \tag{5.11}$$

Finally, for shadow,

$$\begin{aligned}
 H(V|S) &= p_S(?)H(V|S = ?) + p_S(Y)H(V|S = Y) + p_S(N)H(V|S = N) \\
 &= \frac{4}{8}H\left(\frac{1}{2}\right) + \frac{3}{8}H(0) + \frac{1}{8}H(0) = 0.5.
 \end{aligned}
 \tag{5.12}$$

We observe that  $S$  is the optimal attribute for identifying vampires, as  $H(V|S)$  has the smallest value. Furthermore, this result indicates the remaining uncertainty when knowing an individual's shadow.

Note that the conditional entropy  $1 - H(V|\text{attribute} = x)$  measures the homogeneity of individuals with the specified attribute value. A highly homogeneous set has high certainty, resulting in a small conditional entropy, and thus  $1 - H(V|\text{attribute} = x) \approx 1$ . Minimizing conditional entropy is equivalent to maximizing homogeneity, similar to the initial approach. However, unlike the earlier method, which makes a hard decision and only considers completely homogeneous cases, the conditional entropy approach provides a soft decision that estimates a score for the degree of homogeneity even when it's not fully homogeneous.

Since  $I(V; \text{attribute}) = H(V) - H(V|\text{attribute})$ , minimizing  $H(V|\text{attribute})$  is equivalent to maximizing  $I(V; \text{attribute})$ . This means we aim to select attributes that share the most relevant information with the vampire classification. This further justifies the approach as a reasonable choice.

**Picking a Second Test**

**minimizing conditional entropy is making soft-decision**

**minimizing conditional entropy is same as maximizing mutual information**

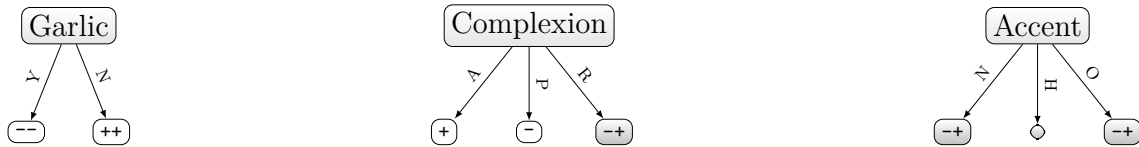


Figure 5.2: Test trees for different attributes given Shadow is unknown. Each '+' sign indicates a vampire count, whereas each '-' sign indicates a non-vampire count.

Now that Shadow is selected as the first test, we need to determine the optimal second test. One straightforward approach is to compare the conditional entropies  $H(V|S, G)$ ,  $H(V|S, A)$ , and  $H(V|S, C)$  and choose the attribute with the smallest value.

However, if we can select the second test based on the outcome of the first test, we should analyze each case separately. For instance, since  $H(V|S = Y) = H(V|S = N) = 0$ , no additional testing is necessary when  $S = Y$  or  $S = N$ . For the

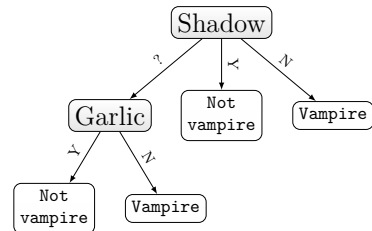


Figure 5.3: Overall decision tree

case where  $S = ?$ , the test trees conditioned on  $S = ?$  are shown in Figure 5.2. The remaining uncertainty after the second tests are respectively

$$\begin{aligned} H(V|S = ?, G) &= \frac{2}{4}H(V|S = ?, G = Y) + \frac{2}{4}H(V|S = ?, G = N) \\ &= \frac{2}{4}H\left(\frac{0}{2}\right) + \frac{2}{4}H\left(\frac{0}{2}\right) = 0, \end{aligned}$$

$$\begin{aligned} H(V|S = ?, C) &= \frac{1}{4}H(V|S = ?, C = A) + \frac{1}{4}H(V|S = ?, C = P) + \frac{2}{4}H(V|S = ?, C = R) \\ &= \frac{1}{4}H\left(\frac{0}{1}\right) + \frac{1}{4}H\left(\frac{0}{1}\right) + \frac{2}{4}H\left(\frac{1}{2}\right) = 0.5, \end{aligned}$$

and

$$\begin{aligned} H(V|S = ?, A) &= \frac{2}{4}H(V|S = ?, A = N) + \frac{2}{4}H(V|S = ?, A = O) \\ &= \frac{2}{4}H\left(\frac{1}{2}\right) + \frac{2}{4}H\left(\frac{1}{2}\right) = 1. \end{aligned}$$

Therefore, we conclude that the Garlic test is the optimal choice. Furthermore, after administering the Garlic test, we can determine with certainty whether the individual is a vampire or not, rendering additional testing unnecessary. The overall decision tree for this scenario is illustrated in Figure 5.3.

### 5.2.3 Potential Extensions in Decision Tree Analysis

In extending the decision tree methodology, it's important to note that the test outcomes don't need to be discrete. Let  $X_i$  represent the test outcome of attribute  $i$ , where some  $X_i$  are continuous. We simply need to select the attribute  $i$  that minimizes  $H(V|X_i)$ , just as before.

Another extension involves constructing multiple trees and leveraging their outcomes to make the final decision, which can help reduce overfitting. The high-level procedure is as follows: **random forest**

1. Select a random subset of training samples and a random subset of attributes.
2. Train a tree using the selected subset of samples and attributes.
3. Classify a test sample using each of the trees, and make the final decision based on a majority vote.

This resulting algorithm is commonly known as the random forest algorithm.

## 5.3 Exercise

1. For the vampire identification problem, let's determine the second test if we use complexion as the attribute in the first test. Please build the entire decision tree so that a vampire can

always be correctly identified for the training data.



# Chapter 6

## Channel coding

Channel coding stands as a cornerstone in communication theory. This chapter will explore the essential concepts and theorems in channel coding.

### 6.1 What is channel coding?

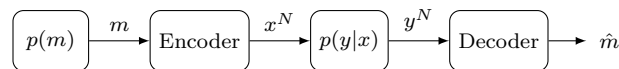


Figure 6.1: Channel coding setup

#### channel coding

**Channel coding** is a crucial component of communication systems, ensuring reliable data transmission over noisy channels by introducing redundancy to the original data. This enables the detection and correction of errors that occur during transmission, preventing significant information loss or misinterpretation. Without channel coding, errors induced by noise, interference, or other impairments in the communication channel could corrupt the transmitted data. By employing channel coding techniques such as error detection and correction codes, the system can enhance the integrity and accuracy of the received information, improving the overall performance and reliability of the communication system. This is vital for applications ranging from simple data transfers to complex and critical systems like satellite communications, mobile networks, and data storage solutions.

#### 6.1.1 Channel Coding Components

#### channel coding setup

As shown in Figure 6.1, the channel coding setup typically consists of:

1. **Message Generation:** Messages to be transmitted through the channel. We will assume that they follow a probability distribution  $p(m)$ .
2. **Encoding Process:** The encoder converts message  $m$  into a sequence  $x^N$  for transmission.

3. **Channel Behavior:** Characterized by  $p(y|x)$ , it impacts the transmitted signal, resulting in  $y^N$ .
4. **Decoding Process:** The decoder reconstructs the message, estimating  $\hat{m}$  from  $y^N$ .

The channel's memoryless nature implies that the output depends solely on the current input. The probability of receiving  $y^N$  for a given  $x^N$  is  $p(y^N|x^N) = \prod_{i=1}^N p(y_i|x_i)$ .

### 6.1.2 Channel Coding Rate and Channel Capacity

An important question in communication theory is how fast we can transmit signals reliably across a channel. To quantify the speed of transmission, we use the **channel coding rate**, often denoted **channel coding** by  $R$ . The channel coding rate is defined as the information content of the transmitted message **rate** per channel use. Therefore, given  $N$  channel uses to transmit a message  $M$ , we have:

$$R = \frac{H(M)}{N},$$

where  $H(M)$  is the entropy of the transmitted message.

We naturally desire a large channel coding rate  $R$  to achieve a higher transmission rate. However, we cannot indefinitely increase  $R$  without compromising reliability. We call the maximum possible channel coding rate that still maintains reliable transmission the **channel capacity**. Interestingly, **capacity** it's not immediately apparent that such a capacity even exists, as reliable transmission may not be achievable regardless of how small the rate is.

A pivotal result by Shannon is the demonstration that channel capacity indeed exists. This fundamental limit indicates that reliable communication is possible if the channel coding rate  $R$  is less than the channel capacity  $C$ . If  $R > C$ , error-free transmission is impossible. This is known as the Channel Coding Theorem, which we will state formally below.

#### Channel Coding Theorem

There is a capacity  $C$  for any channel such that if the channel coding rate  $R$  is larger than  $C$ , then there will be a non-zero error probability for the transmission. Conversely, if  $R < C$ , one can find a channel coding scheme such that the error probability can be made arbitrarily small. Moreover, the capacity is the maximum possible mutual information between the channel input  $X$  and output  $Y$ , optimized over all possible input distributions  $p(x)$ . That is,

$$C = \max_{p(x)} I(X; Y) \quad (6.1)$$

**channel coding theorem**

The major highlight of this chapter will be the proof of the channel coding theorem. We will defer this proof until later in the chapter. Before that, we will explore various examples to develop a more nuanced appreciation for the theorem's significance and how it applies to different scenarios.

### 6.1.3 Capacities of continuous channels

The original channel coding theorem is typically presented for discrete channels with discrete input and output. However, it can be shown that the result also applies to continuous cases.



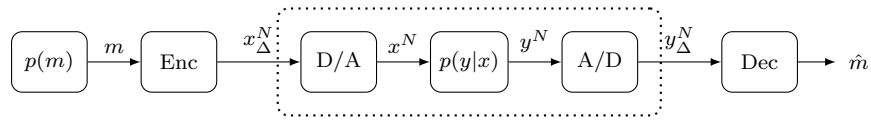


Figure 6.2: A continuous channel is converted to a discrete channel (inside the dotted box) using A/D (Analog-to-Digital) and D/A (Digital-to-Analog) converters.

Consider a continuous channel with both continuous input and output. To apply the channel coding theorem, one approach is convert a discrete input to a continuous signal with Digital-to-Analog (D/A) before sending through the channel and then convert the channel output back to a discrete signal with Analog-to-Digital (A/D) converter. The channel, accompanied by these converters, can then be treated as a discrete channel, as illustrated in Figure 6.2.

Let the quantization step size of the A/D and D/A converters be  $\Delta$  and denote  $C_\Delta$  as the capacity of the resulting discrete channel, and  $X_\Delta$  and  $Y_\Delta$  be the discrete signals before converting  $X$  and after converting  $Y$  with the D/A and A/D converters, respectively. Therefore, we have

**channel coding theorem holds for continuous channels as well**

$$\begin{aligned} C_\Delta &= \max_{p(x)} I(X_\Delta; Y_\Delta) = \max_{p(x)} H(Y_\Delta) - H(Y_\Delta|X_\Delta) \\ &\stackrel{(a)}{\approx} \max_{p(x)} h(Y) - \log \Delta - h(Y|X_\Delta) + \log \Delta \\ &= \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} I(X; Y), \end{aligned}$$

where we use (4.4) in (a). Moreover, as the quantization step size  $\Delta \rightarrow 0$ , (a) becomes an equality rather than an approximation, and  $C_\Delta$  approach the capacity of the original channel without quantization. Thus we have

$$C = C_\Delta = \max_{p(x)} I(X; Y),$$

as  $\Delta \rightarrow 0$ .

Note that the above argument can also be extended to the case with a mixture of discrete and continuous signals. For example, we may have a discrete input  $X$  and continuous output  $Y$ , and the capacity remains the same expression in (6.1).

## 6.2 Communication Channel Examples

We will examine several channel examples and evaluate their capacities using (6.1). First, let's consider the binary symmetric channel, where both the input and output are binary, and the error probabilities given input 0 and 1 are symmetric as follows.

### 6.2.1 Binary symmetric channel

In the case of a binary symmetric channel, both the input and output are binary, taking values 0 or 1. The channel is symmetric, characterized by the probabilities

$$\begin{aligned} p_{Y|X}(1|0) &= p_{Y|X}(0|1) = p, \\ p_{Y|X}(0|0) &= p_{Y|X}(1|1) = 1 - p, \end{aligned}$$

where  $p$  represents the crossover probability. The capacity of such a channel is given by

$$\begin{aligned} C &= \max_{p(x)} I(X; Y) = \max_{p(x)} [H(Y) - H(Y|X)] \\ &\stackrel{(a)}{=} \max_{p(x)} [H(Y) - H(p)] \\ &\stackrel{(b)}{=} 1 - H(p), \end{aligned}$$

**capacity of  
binary  
symmetric  
channel**

where (a) follows from the fact that  $H(Y|X) = \Pr(X = 0)H(Y|X = 0) + \Pr(X = 1)H(Y|X = 1) = \Pr(X = 0)H(p) + \Pr(X = 1)H(p) = H(p)$ <sup>1</sup>, and (b) holds because  $H(Y)$  is maximized when  $X$  is equally probable for 0 and 1, and thus  $Y$  is also equally probable for 0 and 1.

### 6.2.2 Simple Gaussian Channel

Let's consider a simple continuous Gaussian channel next, where the output  $Y$  is expressed as  $Y = X + Z$ , with  $Z$  being zero-mean Gaussian noise independent of the input  $X$ . The channel capacity can then be evaluated as

$$\begin{aligned} C &= \max_{p(x)} I(X; Y) = \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} h(Y) - h(X + Z|X) \\ &\stackrel{(a)}{=} \max_{p(x)} h(Y) - h(Z|X) \\ &\stackrel{(b)}{=} \max_{p(x)} h(Y) - h(Z) = \max_{p(x)} h(Y) - \frac{1}{2} \log 2\pi e \sigma_Z^2 \\ &\stackrel{(c)}{=} \frac{1}{2} \log 2\pi e \sigma_Y^2 - \frac{1}{2} \log 2\pi e \sigma_Z^2 \\ &\stackrel{(d)}{=} \frac{1}{2} \log \frac{\sigma_X^2 + \sigma_Z^2}{\sigma_Z^2} = \frac{1}{2} \log \left( 1 + \frac{\sigma_X^2}{\sigma_Z^2} \right) = \frac{1}{2} \log(1 + \text{SNR}), \end{aligned}$$

where in (a), given  $X$  makes  $X$  a constant and any constant shift does not change the entropy, (b) is due to  $X \perp Z$ , (c) holds with equality if  $X$  and consequently  $Y$  are Gaussian, (d) follows from  $\text{Var}(Y) = \text{Var}(X) + \text{Var}(Z)$  given  $X \perp Z$ , and SNR denotes the signal-to-noise ratio in the last equation. Note that the capacity is a function of the input power and can be made arbitrarily large with unlimited power. This is reasonable because increasing the power makes the noise less significant by comparison.

<sup>1</sup>For a probability  $p$  such that  $0 \leq p \leq 1$ , recall that  $H(p) \triangleq -p \log_2 p - (1 - p) \log_2 (1 - p)$ .

### 6.2.3 Additive White Gaussian Noise Channel

The simple Gaussian channel considered earlier only has one channel. In realistic communication systems, we often have Gaussian channels in parallel from different frequency bands. The most well-known and common one is the **additive white Gaussian noise** (AWGN) channel, where the noise across all bands has the same noise power (hence called white<sup>2</sup>). Let's try to find the capacity of the AWGN channel.

Denote  $W$  as the bandwidth of the channel and  $N_0$  as the noise spectral density. It follows from Nyquist and Shannon's sampling results that a signal with bandwidth  $W$  requires at least  $2W$  samples per second for full reconstruction. This suggests  $2W$  degrees of freedom per second, equivalent to  $2W$  parallel Gaussian channels for every second. Given the noise power spectral density  $N_0$ , the signal-to-noise ratio (SNR) for each channel is given by

$$\text{SNR} \stackrel{(a)}{=} \frac{\sigma_X^2}{2W(N_0/2)} = \frac{P}{WN_0},$$

where we need to divide  $N_0$  by 2 in (a) since the noise power is spread across both positive and negative frequencies.

Therefore, the channel capacity can be expressed as:

$$C = 2W \cdot \frac{1}{2} \log_2(1 + \text{SNR}) = W \log_2 \left( 1 + \frac{P}{WN_0} \right).$$

### 6.2.4 Gaussian Colored Noise Channels

What if the noise power is not white across the spectrum? In this case, we have a Gaussian colored noise channel. Similar to the AWGN case, we essentially have parallel Gaussian channels, but each channel has different noise power. The resulting channel is referred to as a **Gaussian colored noise channel**. Finding its capacity becomes a power allocation problem because, unlike AWGN, we want to assign more power resources to the more effective channels. To simplify the discussion, let's consider a discrete approximation of parallel Gaussian channels, as shown in Figure 6.3, where the colored channel is approximated as a number of parallel Gaussian channels with different noise powers.



Figure 6.3: Discretize a colored noise channel: original channel (left), discretized channel (right).

Let  $K$  be the number of parallel Gaussian channels, each with noise powers  $\sigma_1^2, \sigma_2^2, \dots, \sigma_K^2$ . Suppose we can allocate a total power of  $P$  across all channels, with the powers assigned to individual channels being  $P_1, P_2, \dots, P_K$  such that  $\sum_{i=1}^K P_i \leq P$ . For each  $k$ -th channel, it is possible to transmit  $\frac{1}{2} \log \left( 1 + \frac{P_k}{\sigma_k^2} \right)$  bits per channel use. Thus, we can summarize the optimization problem

<sup>2</sup>Consider white light as originating from a mixture of lights of about the same power.

additive white  
Gaussian noise  
channel

capacity of  
AWGN channel

Gaussian colored  
noise channel

as

$$\max \sum_{k=1}^K \frac{1}{2} \log \left( 1 + \frac{P_k}{\sigma_k^2} \right) \quad \text{such that} \quad P_1, \dots, P_K \geq 0, \quad \sum_{k=1}^K P_k \leq P.$$

**finding capacity  
of colored  
channel**

### KKT Conditions

The Karush-Kuhn-Tucker (KKT) conditions<sup>3</sup> for this optimization problem can be listed as follows. **KKT condition for finding colored channel capacity**

$$\frac{\partial}{\partial P_i} \left[ \sum_{k=1}^K \frac{1}{2} \log \left( 1 + \frac{P_k}{\sigma_k^2} \right) + \sum_{k=1}^K \lambda_k P_k - \mu \left( \sum_{k=1}^K P_k - P \right) \right] = 0, \quad (6.2)$$

$$\mu, \lambda_1, \dots, \lambda_K \geq 0, \quad (6.3)$$

$$P_1, \dots, P_K \geq 0, \quad (6.4)$$

$$\sum_{k=1}^K P_k \leq P, \quad (6.5)$$

$$\mu \left( \sum_{k=1}^K P_k - P \right) = 0, \quad \lambda_k P_k = 0, \quad \forall k. \quad (6.6)$$

### Capacity of Parallel Channels

From (6.2), we can show that

$$P_i + \sigma_i^2 = \frac{1}{2(\mu - \lambda_i)}. \quad (6.7)$$

Using the complementary condition  $\lambda_i P_i = 0$  from (6.6), we know that for  $P_i > 0$ ,  $\lambda_i = 0$  and for  $\lambda_i > 0$ ,  $P_i = 0$ . Therefore, we have

$$P_i + \sigma_i^2 = \frac{1}{2\mu} = \text{constant, when } P_i > 0 \quad (6.8)$$

and

$$\sigma_i^2 = \frac{1}{2(\mu - \lambda_i)}, \text{ when } P_i = 0. \quad (6.9)$$

where (6.9) is irrelevant physically as we are not too interested in the exact value of  $\lambda_i$  when we already know that  $P_i = 0$ . On the other hand, (6.8) captures the essence for efficient allocation and we will interpret it in more detail next.

Note that (6.8) also implies that  $\mu$  cannot be zero, and consequently,  $\sum_{k=1}^K P_k = P$  from (6.6), which makes sense, as we should always utilize all available power to maximize capacity.

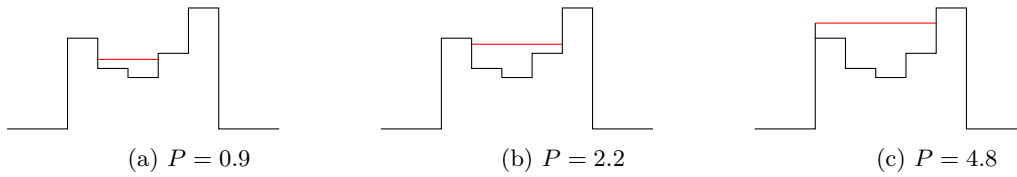


Figure 6.4: Waterfilling interpretation of power allocation in parallel channels. Five available channels have noise powers:  $\sigma_1^2 = 3, \sigma_2^2 = 2, \sigma_3^2 = 1.7, \sigma_4^2 = 2.5, \sigma_5^2 = 4$ .

### Waterfilling Interpretation

There is a nice interpretation of the solution described in (6.8). Let's illustrate this with a simple example using five parallel channels, as shown in Figure 6.4. In Figure 6.4a, the total power is 0.9, and we want to allocate it such that  $P_1 + P_2 + P_3 + P_4 + P_5 = 0.9$  and  $P_i + \sigma_i^2 = \text{constant}$  when  $P_i \neq 0$ . We start by allocating power to the least noisy channel, which is the third channel with  $\sigma_3^2 = 1.7$ . We allocate 0.3 to it, making  $P_3 + \sigma_3^2 = 2 = P_2$ , the power of the second least noisy channel. Next, we allocate the remaining power ( $0.9 - 0.3 = 0.6$ ) equally to the second and third channels, so that  $P_2 + \sigma_2^2 = P_3 + \sigma_3^2$ . This results in  $P_2 = 0.3, P_3 = 0.6$ , and  $P_2 + \sigma_2^2 = P_3 + \sigma_3^2 = 2.3$ , which is less than  $\sigma_1^2, \sigma_4^2$ , and  $\sigma_5^2$ . We can compute  $\lambda_1, \lambda_4$ , and  $\lambda_5$  from (6.9), but it's unnecessary since we know that  $P_1 = P_4 = P_5 = 0$ . This is because we cannot have, for example,  $P_1 + \sigma_1^2 = P_2 + \sigma_2^2$  unless  $P_1 < 0$ , which is not physically possible.

This process is analogous to pouring water into a hilly landscape. The water flows to the lowest levels, and the water level should be the same everywhere. Areas above the water level will be dry, suggesting that no power is allocated there. This way of allocating power is often known as the **waterfilling algorithm**.

We may continue our example as shown in Figure 6.4b, where  $P = 2.2$ . Instead of allocating 0.3 to both the second and third channels after the initial allocation of 0.3 to the third channel, let's allocate 0.5 in the next round. We now have  $P_2 = 0.5, P_3 = 0.5 + 0.3 = 0.8$ , and  $P_2 + \sigma_2^2 = P_3 + \sigma_3^2 = 2.5 = \sigma_4^2$ . So, for the remaining power, we have to allocate to not just the second and third channels but also the fourth one to ensure an equal water level everywhere. The remaining power now is  $2.2 - 0.5 - 0.8 = 0.9$ . So, as we split the remaining power across the three channels, we finally have  $P_2 = 0.5 + 0.3 = 0.8, P_3 = 0.8 + 0.3 = 1.1$ , and  $P_4 = 0.3$ . Moreover, we have  $P_2 + \sigma_2^2 = P_3 + \sigma_3^2 = P_4 + \sigma_4^2 = 2.8 < \sigma_1^2$  and  $\sigma_5^2$ . So, again, we should have  $P_1 = P_5 = 0$ .

Using the same argument, we can try to allocate the power when  $P = 4.8$  as shown in Figure 6.4c, but we will leave it as an exercise.

## 6.3 Jointly typical sequences

Before we present the forward proof of the channel coding theorem, we will first develop some essential tools to facilitate our discussion. We will begin by extending the concept of typical sequences, introduced in Section 3.5.2, to jointly typical sequences. Then, we will utilize this new concept to establish two crucial lemmas: the packing lemma and the covering lemma.

<sup>3</sup>Please see appendix.

### 6.3.1 What are jointly typical sequences

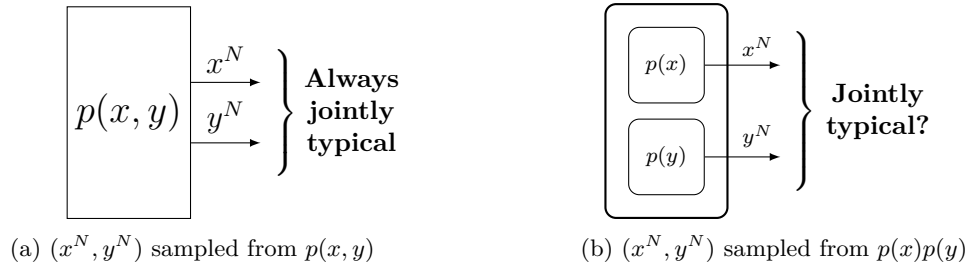


Figure 6.5: Any pair of sequences sampled from a joint source will be jointly typical, as illustrated in the left figure. However, if the two sequences are sampled from two independent sources, there is no guarantee that the sequences will be jointly typical as illustrated in the right figure.

For a pair of sequences  $x^N$  and  $y^N$ , we define them as **jointly typical** if both  $x^N$  and  $y^N$  are **jointly typical** and the following condition holds

$$2^{-N(H(X,Y)+\epsilon)} \leq p(x^N, y^N) \leq 2^{-N(H(X,Y)-\epsilon)}. \quad (6.10)$$

#### Sampling from a joint source is always jointly typical

Consider sampling a sequence pair  $(x^N, y^N)$  from a joint source, as illustrated in Figure 6.5a. Note that we can treat each pair  $(x_i, y_i)$  as a single source. Using the same argument as for typical sequences and the law of large numbers, the condition depicted in (6.10) will always be satisfied as  $N$  becomes sufficiently large. Moreover, when we consider only one of the sources, say  $X$  only, **sequences** from the joint source, the sampled sequence  $x^N$  will apparently be typical. The same holds for  $y^N$ . **sampled from** Therefore, all the conditions for jointly typical sequences will be satisfied. In summary, whenever we **joint source are** sample a sequence pair from a joint source, the pair will be jointly typical, given that the sampling **jointly typical** length is sufficiently large.

#### Sampling from two separate sources

The more interesting case is when the sequence pair  $(x^N, y^N)$  is from two separate sources, as shown in Figure 6.5b. In this case, each sequence, when considered independently, is typical. However, they are not jointly typical since (6.10) is generally not satisfied.

Let's denote  $\mathcal{A}_\epsilon^{(N)}(X, Y)$  as the set of jointly typical sequences. Given a sequence pair randomly sampled as in Figure 6.5b, how likely is it that the sample pair will be jointly typical? First, we know that there are approximately  $2^{NH(X)}$  typical sequences of  $X$  with length  $N$ . If we consider the pair  $(X, Y)$  as a single variable, there should be approximately  $2^{NH(X,Y)}$  jointly typical sequences by the same argument. Therefore,  $|\mathcal{A}_\epsilon^{(N)}(X, Y)| \approx 2^{NH(X,Y)}$ .

More precisely, we have,

$$(1 - \delta)2^{N(H(X,Y)-\epsilon)} \leq |\mathcal{A}_\epsilon^{(N)}| \leq 2^{N(H(X,Y)+\epsilon)}$$

**probability of  
independent  
sequences to be  
jointly typical**

(6.11)

similar to (3.13). Note that the  $\delta$  above can be made arbitrarily small by sufficiently increasing  $N$ . Therefore, it can essentially be ignored in practice.

Thus we have,

$$\begin{aligned}
& Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)) \\
&= \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)\}} p(x^N, y^N) \\
&\stackrel{(a)}{=} \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)\}} p(x^N)p(y^N) \\
&\stackrel{(b)}{\leq} \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)\}} 2^{-N(H(X)-\epsilon)} 2^{-N(H(Y)-\epsilon)} \\
&\leq 2^{-N(I(X;Y)-3\epsilon)}, \tag{6.12}
\end{aligned}$$

where (a) is due to the independence of  $x^N$  and  $y^N$ , and (b) is directly from the definition of typical sequences as in (3.12). Similarly, we can also evaluate the probability lower bound for these sequences being jointly typical as

$$\begin{aligned}
& Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)) \\
&= \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)\}} p(x^N, y^N) \\
&= \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)\}} p(x^N)p(y^N) \\
&\stackrel{(a)}{\geq} \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}(X, Y)\}} 2^{-N(H(X)+\epsilon)} 2^{-N(H(Y)+\epsilon)} \\
&\stackrel{(b)}{\geq} (1 - \delta) 2^{-N(I(X;Y)+3\epsilon)}, \tag{6.13}
\end{aligned}$$

where (a) is from (3.12) and (b) is from (6.11)

In summary, we see that the probability of independently sampled sequences being jointly typical is approximately  $2^{-NI(X;Y)}$ , which decays asymptotically unless  $I(X;Y) = 0$ , indicating  $X$  and  $Y$  are independent. In the case when  $X^N$  and  $Y^N$  are already expected to be independent in the setup, i.e.,  $I(X;Y) = 0$ , the bounds become useless as the sequences will be considered as jointly typical even if they were sampled independently.

### 6.3.2 Packing and Covering Lemmas

As discussed in the previous section, we don't expect two sequences independently sampled from separate sources to be jointly typical for sufficiently long sequence lengths. However, what if we have multiple such sequences? Specifically, if we want to guarantee finding at least one jointly typical sequence among these sequences or ensure that no such jointly typical sequence exists, what is the minimum number of sequences required and what is the maximum number of sequences we can have? These questions are addressed by the covering and packing lemmas, which are presented

in this section.

**Packing Lemma**

Let's consider the second case first. Given a sequence  $Y^N$ , we know that an independent  $X^N$  is unlikely to be jointly typical with  $Y^N$ . However, if we have many such  $X^N$  sequences, eventually, one of them should be jointly typical with  $Y^N$ . The question is, how many  $X^N$  sequences can we **packing lemma** pack with  $Y^N$  such that still, no  $X^N$  is jointly typical with  $Y^N$ ? The answer of this question is addressed by the so-called **packing lemma**. Since the probability of an arbitrarily sampled  $X^N$  being jointly typical with  $Y^N$  is approximately  $2^{-NI(X;Y)}$ , a reasonable guess for the number of  $X^N$  sequences we can pack is simply  $\frac{1}{2^{-NI(X;Y)}} = 2^{NI(X;Y)}$ . This guess turns out to be a correct one, and we will now show it formally.

Let  $M = 2^{NR}$  be the number of  $X^N$  sequences to be drawn. The probability that any one of these  $M$  sequences is jointly typical with  $Y^N$  is upper bounded by

$$\begin{aligned} Pr(\text{Any one of } M \text{ } X^N \text{ jointly typical with } Y^N) &\stackrel{(a)}{\leq} MPr((X^N, Y^N) \in \mathcal{A}_\epsilon^N(X, Y)) \\ &\stackrel{(b)}{\leq} M2^{-N(I(X;Y)-3\epsilon)} \\ &\stackrel{(c)}{\leq} 2^{-N(I(X;Y)-R-3\epsilon)} \\ &\rightarrow 0 \text{ as } N \rightarrow \infty \text{ and } I(X;Y) - 3\epsilon > R, \end{aligned}$$

where (a) is due to the union bound, (b) is from (6.12), and (c) is due to  $2^{NR} = M$ .

As  $\epsilon$  can be made arbitrarily small with increasing  $N$ , it follows that as long as  $I(X;Y) > R$ , we can find a sufficiently large  $N$  such that we can pack  $2^{NR}$   $X^N$  sequences with  $Y^N$  without any of the  $X^N$  being jointly typical with  $Y^N$ .

**Covering Lemma**

The packing lemma states that as long as the number of sequences  $X^N$  packed with  $Y^N$  is less than  $2^{NI(X;Y)}$ , none of the  $X^N$  sequences will be jointly typical with  $Y^N$ . But what happens if we have more than  $2^{NI(X;Y)}$  sequences? Does this imply that some  $X^N$  must be jointly typical with  $Y^N$ ? The answer is affirmative. This result is known as the **covering lemma**, which will be formally proved in the following.

Let's assume again that we draw  $M(= 2^{NR})$  such  $X^N$  sequences. The probability that none of the  $X^N$  sequences is jointly

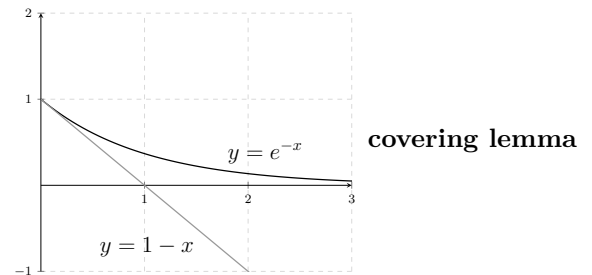


Figure 6.6:  $e^{-x} \geq 1 - x$



typical with  $Y^N$  is given by

$$\begin{aligned}
& Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X, Y) \text{ for all } m) \\
&= \prod_{m=1}^M Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(Y, X)) \\
&= \prod_{m=1}^M \left[ 1 - Pr((X^N(m), Y^N) \in \mathcal{A}_\epsilon^{(N)}(Y, X)) \right] \\
&\stackrel{(a)}{\leq} (1 - (1 - \delta)2^{-N(I(Y;X)+3\epsilon)})^M \\
&\stackrel{(b)}{\leq} \exp(-M(1 - \delta)2^{-N(I(Y;X)+3\epsilon)}) \\
&\stackrel{(c)}{\leq} \exp(-(1 - \delta)2^{N(R-I(Y;X)-3\epsilon)}) \\
&\rightarrow 0 \text{ as } N \rightarrow \infty \text{ and } R > I(X;Y) + 3\epsilon,
\end{aligned}$$

where (a) is due to (6.13), (b) is due to the inequality  $e^{-x} \geq 1 - x$ , as illustrated in Figure 6.6, and (c) is due to the fact that  $M = 2^{NR}$ .

As  $\epsilon$  can be made arbitrarily small, the probability that none of the  $X^N$  sequences are jointly typical with  $Y^N$  goes to zero when  $R > I(X;Y)$  or the number of  $X^N$  sequences is larger than  $2^{NI(X;Y)}$ . Consequently, we must have at least one  $X^N$  sequence that is jointly typical with  $Y^N$  in that case.

### Summary of Packing and Covering Lemmas

From the packing and covering lemmas, we see that  $2^{NI(X;Y)}$  acts as a critical point. If the number of sequences  $X^N$  is less than  $2^{NI(X;Y)}$ , we are in one phase where the packing lemma tells us that none of the  $X^N$  sequences are jointly typical with  $Y^N$ . Conversely, if the number exceeds  $2^{NI(X;Y)}$ , the covering lemma indicates that we must be able to find some  $X^N$  sequences that are jointly typical with  $Y^N$ .

As we will see shortly, the packing lemma is instrumental in the forward proof of the channel coding theorem, while the covering lemma is crucial in the proof of the rate-distortion theorem. However, the rate-distortion theorem is beyond the scope of this book. Let's continue with the forward proof of the channel coding theorem.

## 6.4 Forward proof of channel coding theorem

Now, we have enough tools to give a forward proof of the channel coding theorem. That is, given that the code rate is below the capacity, we should be able to find a reliable coding scheme to transmit messages with the target code rate. One key idea of the proof is that we should use the channel multiple times for every message sent. For each message to be sent, the encoder will map the message to a length- $N$  codeword to send through the channel. So each message will need  $N$  channel uses. And if the code rate is  $R$ , then the number of different messages (and consequently the size of the codebook) should be  $2^{NR}$ . The overall proof strategy is to construct a codebook randomly, derive a pair of encoding and decoding scheme with the codebook, and finally analyze

the error rate and show that the error rate can be made arbitrarily small and hence demonstrate a reliable transmission.

### 6.4.1 Setup coding scheme

#### Codebook construction

Let's start with codebook construction. Recall that from the channel coding theorem, the capacity is given by  $C = \max_{p(x)} I(X; Y)$ . Assume that  $p^*(x)$  is the input distribution that actually achieve the capacity, i.e.,  $p^*(x) = \arg \max_{p(x)} I(X; Y)$ . Recall that there should be  $2^{NR}$  messages and consequently codewords given code rate  $R$  and length- $N$  codewords. Here, we use a **random coding** strategy. We will generate a random codebook with each codeword sampled from the **random coding** Discrete Memoryless Source (DMS)  $p^*(x)$   $N$  times as

$$\begin{aligned} \mathbf{c}(1) &= (x_1(1), x_2(1), \dots, x_N(1)), \\ \mathbf{c}(2) &= (x_1(2), x_2(2), \dots, x_N(2)), \\ &\vdots \\ \mathbf{c}(2^{NR}) &= (x_1(2^{NR}), x_2(2^{NR}), \dots, x_N(2^{NR})). \end{aligned}$$

#### Encoding and decoding procedures

The encoding procedure is straightforward: we simply map each message to its corresponding codeword. Thus, given an input message  $m$ , the encoder output is  $\mathbf{c}(m) = (x_1(m), x_2(m), \dots, x_N(m))$ .

For decoding, upon receiving the sequence  $\mathbf{y} = (y_1, y_2, \dots, y_N)$ , the decoder selects a sequence  $\mathbf{c}(m)$  from the codebook  $\{\mathbf{c}(1), \dots, \mathbf{c}(2^{NR})\}$  such that  $(\mathbf{c}(m), \mathbf{y})$  are jointly typical, i.e.,  $p_{X^N, Y^N}(\mathbf{c}(m), \mathbf{y}) \approx 2^{-NH(X, Y)}$ . If no such  $\mathbf{c}(m)$  exists or if multiple such sequences exist, an error is declared. Otherwise, the decoded message is output as  $m$ .

### 6.4.2 Performance analysis

Instead of focusing on the performance of a particular code instance, the trick of the random coding argument is to consider the average performance over all codes randomly generated from  $p^*(x)$ . Assume that the original message  $M = m$ . A decoding error occurs when either of the following **error events** happens:

- $E_1$ : The codeword of the original message  $\mathbf{C}(m)$  is not jointly typical with the received code vector  $\mathbf{Y}$ . That is,  $(\mathbf{C}(m), \mathbf{Y}) \notin A_\epsilon^N(X, Y)$ ;
- $E_2$ : For some  $M' \neq m$ , we have  $(\mathbf{C}(M'), \mathbf{Y}) \in A_\epsilon^N(X, Y)$ .

**decoding error events**

We can quickly show that the probabilities of both events go to zero as  $N$  goes to infinity.

First,  $P(E_1)$  goes to zero since  $(\mathbf{C}(m), \mathbf{Y})$  is essentially drawn from the joint source  $p^*(x)p(y|x)$ . By the definition of jointly typical sequences and the law of large numbers, any sequence pair drawn from the respective joint source will be jointly typical, so  $P(E_1)$  goes to zero.

As for  $E_2$ , recall that  $X$  is sampled from  $p^*(x)$ , the distribution that achieves the capacity. To emphasize this, let's denote  $X$  here as  $X^*$  and the channel output  $Y$  as  $Y^* \sim \sum_x p(y|x)p^*(x)$ . Note that we have  $C = I(X^*; Y^*)$ . Moreover, since  $R < C$ , the number of codewords,  $2^{NR}$ , is less than

$2^{NC} = 2^{NI(X^*;Y^*)}$ . Consequently, by the packing lemma, there is no other codeword besides  $\mathbf{C}(m)$  that can be jointly typical with  $Y^N$  as  $N$  goes to infinity. Therefore,  $P(E_2) \rightarrow 0$ .

### Additional considerations

Our goal is to show that there exists a code  $\mathbf{c}^*(\cdot)$  such that the probability of error  $\Pr(\text{error}|\mathbf{c}^*, m)$  approaches zero for any message  $m$ . However, we actually demonstrated that the average error probability over all random codes can be made arbitrarily small for a given message  $m$ . More importantly, our earlier discussion does not guarantee that different messages can share the same codebook, deviating from our original aim of finding a single codebook that works for all messages, yielding zero error probability for the corresponding codewords.

Note that even if we specify a particular message  $m$  over our previous discussion, the early argument still holds if the message is randomly drawn as well. This implies that the average error over all codes and messages,

$$\sum_{\mathbf{m}} p(m) \sum_{\mathbf{c}} p(\mathbf{c}) \Pr(\text{error}|\mathbf{c}, m)$$

also tends to zero as the length of the codeword  $N$  increases. Therefore, the best code  $\mathbf{c}^*$  in terms of the lowest average error must have

$$\sum_{\mathbf{m}} p(\mathbf{m}) \Pr(\text{error}|\mathbf{c}^*, m) \equiv \delta \rightarrow 0.$$

However, we are not done yet. Even for this good codebook, the average error over all messages goes to zero, but there is no guarantee that we can't have a bad codeword and a respective message with high error probability. To fix that, without loss of generality, we can first assume that all messages are equally likely, leading to

$$\frac{1}{2^{NR}} \sum_m \Pr(\text{error}|\mathbf{c}^*, m) = \delta.$$

Now, by discarding the worse half of the codewords, for any remaining messages  $m$ , we must have<sup>4</sup>

$$\Pr(\text{error}|\mathbf{c}^*, m) \leq 2\delta \rightarrow 0 \text{ as } N \rightarrow \infty.$$

Note that discarding the bad half of the messages reduces the rate from  $R$  to  $\frac{\log_2(2^{NR}/2)}{N} = \frac{\log_2(2^{N(R-1/N)})}{N} = R - \frac{1}{N}$ . However, the final rate can still be made arbitrarily close to the capacity as  $N \rightarrow \infty$ .

## 6.5 Converse proof of channel coding theorem

In the forward proof, we show that there exists a reliable coding scheme as long as the code rate is less than the capacity. For the converse, we want to show that as long as the rate is larger than

---

<sup>4</sup>Why? Consider a simple example of four ascending values  $x_1 \leq x_2 \leq x_3 \leq x_4$  with an empirical mean  $\mu$ . If we focus on the smaller half,  $x_1$  and  $x_2$ , none of these values can be larger than  $2\mu$ . Otherwise, we have  $2\mu < x_3 \leq x_4$ . These two values alone would lead to a total sum exceeding  $4\mu$  and consequently an empirical mean larger than  $\mu$ .

the capacity, the error rate will be non-zero, making reliable transmission impossible.

Equivalently, as long as the probability of error is zero, the rate of the code  $R$  cannot exceed the capacity. We will prove this statement instead.

To proceed with the converse proof, we need to introduce a result known as Fano's inequality.

### 6.5.1 Fano's inequality

#### Fano's inequality

Denote  $Pr(\text{error}) = P_e = Pr(M \neq \hat{M})$ . Then,

$$H(M|Y^N) \leq 1 + P_e H(M)$$

Intuitively, if  $P_e \rightarrow 0$ , on average we will know  $M$  for certain given  $\mathbf{Y}$ , and thus

$$\frac{1}{N} H(M|Y^N) \rightarrow 0$$

Fano's inequality  
for channel  
coding theorem

*Proof.* Let  $E = I(M \neq \hat{M})$ , then

$$\begin{aligned} H(M|Y^N) &= H(M, E|Y^N) - H(E|Y^N, M) \\ &\stackrel{(a)}{=} H(M, E|Y^N) = H(E|Y^N) + H(M|Y^N, E) \\ &\stackrel{(b)}{\leq} H(E) + H(M|Y^N, E) \\ &\stackrel{(c)}{\leq} 1 + P(E=0)H(M|Y^N, E=0) + P(E=1)H(M|Y^N, E=1) \\ &\stackrel{(d)}{\leq} 1 + 0 + P_e H(M|Y^N, E=1) \stackrel{(e)}{\leq} 1 + P_e H(M), \end{aligned}$$

where (a) is due to the fact that  $E$  is deterministic given  $Y^N$  and  $M$ , (b) and (e) are due to the principle that conditioning reduces entropy, (c) follows from the definition of conditional entropy and the fact that  $E$  is binary, and (d) is due to  $M$  being deterministic given  $Y^N$  when there is no error. □

### 6.5.2 Converse proof

We are now ready to proceed with the converse proof. Recall that our objective is to show that if the error rate approaches zero, the code rate must be less than or equal to the capacity. In our earlier discussion, we defined the code rate as the logarithm of the number of messages per channel use. While this definition was reasonable, we need to establish a direct connection between the code rate and the message variable  $M$  here. Therefore, we define the code rate more precisely as  $R = \frac{H(M)}{N}$ , which represents the actual amount of message information transmitted through the

**converse proof  
of channel  
coding theorem**

channel per channel use. Thus,

$$\begin{aligned}
R &= \frac{H(M)}{N} = \frac{1}{N} [I(M; Y^N) + H(M|Y^N)] \\
&\stackrel{(a)}{\leq} \frac{1}{N} [I(X^N; Y^N) + H(M|Y^N)] = \frac{1}{N} [H(Y^N) - H(Y^N|X^N) + H(M|Y^N)] \\
&\stackrel{(b)}{=} \frac{1}{N} \left[ H(Y^N) - \sum_i H(Y_i|X^N, Y^{i-1}) + H(M|Y^N) \right] \\
&\stackrel{(c)}{=} \frac{1}{N} \left[ H(Y^N) - \sum_i H(Y_i|X_i) + H(M|Y^N) \right] \\
&\stackrel{(d)}{=} \frac{1}{N} \left[ \sum_i H(Y_i|Y^{i-1}) - \sum_i H(Y_i|X_i) + H(M|Y^N) \right] \\
&\stackrel{(e)}{\leq} \frac{1}{N} \left[ \sum_i H(Y_i) - \sum_i H(Y_i|X_i) + H(M|Y^N) \right] \\
&= \frac{1}{N} \left[ \sum_i I(X_i; Y_i) + H(M|Y^N) \right] = I(X; Y) + \frac{H(M|Y^N)}{N} \stackrel{(f)}{\rightarrow} I(X; Y)
\end{aligned}$$

where (a) follows from the data processing inequality, (b) and (d) follow from the chain rule of entropies and conditional entropies, (c) follows from the Markov chain  $Y_i \leftrightarrow X_i \leftrightarrow (X^{i-1}, Y^{i-1})$ , (e) follows from the fact that conditioning reduces entropy, and (f) follows from Fano's inequality as  $N \rightarrow \infty$ .

## 6.6 Summary

Channel coding is crucial for efficient and reliable communication. Understanding its setup, rate, and capacity is fundamental to modern communication technologies.

## 6.7 Exercise

1. Try to verify (6.7) in the derivation of the capacity of the Gaussian colored noise channel.
2. Try to find the optimal power allocation for the Gaussian colored noise channel as shown in Figure 6.4c. How should we reallocate the power when  $P = 8$  instead?



# Chapter 7

## Graphical models

graphical models provide a powerful framework for representing and reasoning about complex dependencies among a set of random variables. While traditional Bayesian inference focuses on updating beliefs about unknown parameters given observed data, graphical models extend this approach by offering a structured way to visualize and analyze the relationships between multiple variables. This added structure is particularly valuable in dealing with high-dimensional data and intricate dependency structures that are common in real-world applications such as genetics, image analysis, natural language processing, and social network analysis. By encapsulating conditional dependencies and independencies, graphical models simplify the computation of joint and marginal distributions, facilitate efficient algorithms for exact and approximate inference, and provide clear insights into the underlying causal mechanisms. Thus, studying graphical models not only enriches our understanding of probabilistic inference but also equips us with practical tools for tackling a wide range of complex problems in data science and artificial intelligence.

In this chapter, we will provide a brief introduction to common graphical models. We will then examine an efficient method for applying inference on graphical models: belief propagation (BP) and its application in channel decoding. Additionally, we will demonstrate how BP can be derived from the Bethe approximation in statistical physics. Finally, we will explore how BP can be extended from discrete to continuous random variables through Gaussian approximation, resulting in the Gaussian belief propagation (GaBP) algorithm. We will also demonstrate how the renowned Kalman filter can be easily explained within the framework of GaBP.

### 7.1 Bayesian networks

A Bayesian network (BN) represents the relationships between variables using a directed graph without loops, known as a directed acyclic graph (DAG). The network structure encodes different dependencies among the variables, allowing for the extraction of various conditional independence relationships. This structure enables the representation of complex distributions in a more tractable form, reducing model complexity and facilitating easier inference.

Consider a simple example with three variables  $X$ ,  $Y$ , and  $Z$ . If the joint distribution can be



Figure 7.1: BNs for joint distribution  $p(x, y, z)$  equal to  $p(x)p(y|x)p(z|y)$  and  $p(z)p(y|z)p(x|y)$ . Despite the difference in arrow direction, the two BNs represent the same underlying probability structure.

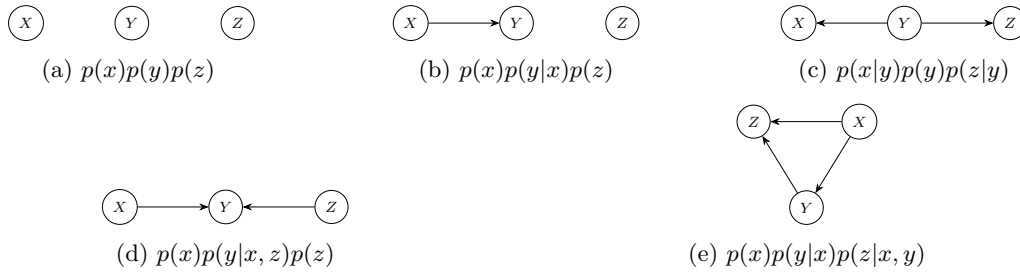


Figure 7.2: More BNs with three variables  $X, Y,$  and  $Z$ . The subcaption specifies the joint distribution  $p(x, y, z)$  of the respective graph.

factorized as

$$p(x, y, z) = p(x)p(y|x)p(z|y)$$

then the corresponding BN will be a three-node graph with  $X, Y,$  and  $Z$  as vertices, as shown in Fig. 7.1a. The graph contains two directed edges, one from  $X$  to  $Y$  and another from  $Y$  to  $Z$ , representing the conditional dependencies between the variables.

By Bayes' rule, we have:

$$p(x)p(y|x)p(z|y) = \cancel{p(x)} \frac{p(x, y)}{\cancel{p(x)}} \frac{p(z, y)}{p(y)} = \cancel{p(y)} p(x|y) \frac{p(z)p(y|z)}{\cancel{p(y)}} = p(z)p(y|z)p(x|y).$$

**conditional independence but not independence**

Therefore, the BN with directed edges from  $Z$  to  $Y$  and from  $Y$  to  $X$ , as shown in Fig. 7.1b, represents the same underlying structure as in Fig. 7.1a. These networks are referred to as **Markov equivalent** in this case, highlighting the fact that they encode the same conditional independence relationships between the variables.

**Markov equivalent**

Given  $X, Y,$  and  $Z$ , we may also have an empty graph with no edge at all, corresponding to the case where

$$p(x, y, z) = p(x)p(y)p(z)$$

and all three variables are independent, as shown in Fig. 7.2a.

We may also have just one edge from  $x$  to  $y$ , corresponding to the case where

$$p(x, y, z) = p(x)p(y|x)p(z)$$

which describes the situation where  $Z$  is independent of  $X$  and  $Y$ , as shown in Fig. 7.2b.

Now, back to the cases with two edges, let's consider the case where both directed edges are coming out of the same variable. For example, we have two directed edges from  $Y$  to  $X$  and from



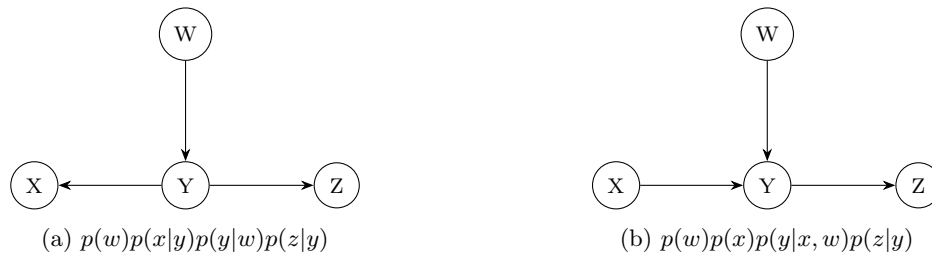


Figure 7.3: Two different BNs for four variables  $W, X, Y$ , and  $Z$ . If  $W$  does not exist, the two BNs actually would be Markov equivalent.

$Y$  to  $Z$ , as shown in Fig. 7.2c. This corresponds to the joint distribution

$$p(x, y, z) = p(y)p(x|y)p(z|y) \stackrel{(a)}{=} p(x)p(y|x)p(z|y),$$

where (a) is simply from Bayes' rule. From (a), we can see that this case is actually the same as the case we have earlier in Fig. 7.1. However, this Markov equivalence is not generalizable when we have more variables. For example, if we also have another variable  $W$  and an edge from  $W$  to  $Y$ , as shown in Fig. 7.3a, and we flip the direction of the edge  $Y$  to  $X$ , as shown in Fig. 7.3b, then  $W$  and  $X$  become independent, but the earlier representation in Fig. 7.3a does not imply that.

Let's revisit the scenario with only three variables and two edges, as described earlier. For the model studied in Fig. 7.2c (Markov equivalent to networks in Fig. 7.1), variables  $X$  and  $Z$  are not independent. However, they become conditionally independent given the variable  $Y$  in the middle.

The situation changes completely when we flip the edge directions, as shown in Fig. 7.2d. The corresponding joint probability for this graph is

$$p(x, y, z) = p(x)p(z)p(y|x, z).$$

Here,  $X$  and  $Z$  are independent, since

$$p(x, z) = \sum_y p(x, y, z) = \sum_y p(x)p(z)p(y|x, z) = p(x)p(z).$$

However, we don't have  $p(x, z|y) = p(x|y)p(z|y)$ , implying that  $X$  and  $Z$  are not conditionally independent given  $Y$ . This result may seem counterintuitive, especially for students encountering this concept for the first time. The fact that two variables that are initially independent can become dependent after observing additional variables may be difficult to grasp. The following concrete example aims to illustrate this concept more clearly:

#### Example 7.1: Two independent binary variables

Let  $X$  and  $Z$  are two independent binary outcomes taking values  $\{0, 1\}$  and  $Y = X \oplus Z$ , where  $\oplus$  is the XOR operation. In this case,  $X$  is completely determined by  $Z$  when  $Y$  is given, making them maximally correlated given  $Y$ .

Now, let's consider the case with three edges. Note that we cannot have edges forming a loop,

**independence  
but not  
conditional  
independence**

as BNs must be acyclic graphs. One possibility is having three directed edges: one from  $X$  to  $Y$ , one from  $Y$  to  $Z$ , and the last one from  $X$  to  $Z$ , as shown in Fig. 7.2e.

In this case, the joint probability will be

$$p(x, y, z) = p(x)p(y|x)p(z|x, y).$$

Note that the joint probability can be directly obtained using Bayes' rule, and the graph itself does not inherently assume any dependence or independence structure. Any other three-variable BNs with three edges will be identical and will not imply any independence among the variables either.

### 7.1.1 D-separation in Bayesian networks

As we have seen, BNs can be used to derive the dependencies among variables. In fact, a key function of graphical models like BNs is to represent such dependencies. The concept of **d-separation** highlights the conditional independence of variables given certain other variables, as can be observed from the network structure. The “d” in d-separation refers to “directional”, emphasizing the direction of the dependencies.

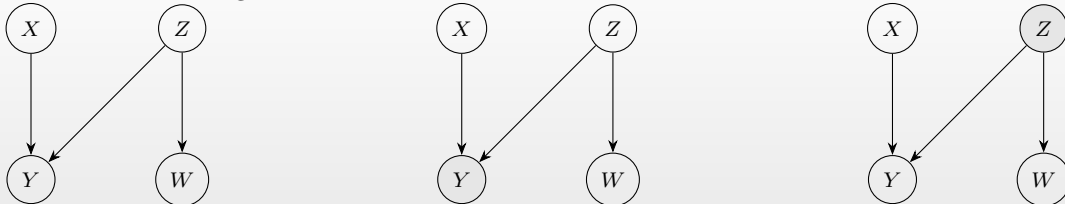
Rather than memorizing the rules of d-separation, it is more intuitive to understand the simple examples with three variables. For instance,  $X$  and  $Z$  are conditionally independent given  $Y$  in Figs. 7.1a, 7.1b, and 7.2c, but not in Fig. 7.2d.

Observing the conditioned variable can break the dependency path between variables. For example, in the three-variable three-edge case (Fig. 7.2e), observing  $Y$  breaks the dependency path  $X \rightarrow Y \rightarrow Z$  for  $X$  and  $Z$ . However,  $X$  and  $Z$  are not yet conditionally independent due to the direct path  $X \rightarrow Z$ .

Let's consider one more example with four variables as below.

#### Example 7.2: Make observations

Consider four variables  $X, Y, Z$ , and  $W$ , and three edges ( $X \rightarrow Y$ ,  $Z \rightarrow Y$ , and  $Z \rightarrow W$ ) as shown in leftmost figure below.



Note that, as root nodes,  $X$  and  $Z$  are independent. Consequently,  $X$  and  $W$  are independent since  $W$  only depends on  $Z$ . However, if  $Y$  is observed as in the middle figure,  $X$  and  $Z$  are no longer independent, and neither are  $X$  and  $W$ . Therefore,  $X$  and  $W$  are not conditionally independent given  $Y$ , or not d-separable given  $Y$ .

Finally, let's observe  $Z$  instead as in the rightmost figure. Since  $Y$  and  $W$  depend on each other through  $Z$ , they are no longer dependent when  $Z$  is observed. Consequently,  $Y$  and  $W$  are d-separable given  $Z$ . Since  $X$  depends on  $W$  through its connection with  $Y$ ,  $X$  and  $W$  are also d-separable given  $Z$ .

### 7.1.2 Burglar and raccoon

Consider a scenario involving a burglar visit ( $B$ ), a raccoon appearing ( $R$ ), a dog barking ( $D$ ), cops being called ( $C$ ), and a trash can falling ( $T$ ). The relationships between these variables are captured by the BN shown in Fig. 7.4. This network represents the conditional dependencies and independencies among the variables, providing a compact and intuitive representation of the relationships between the events.

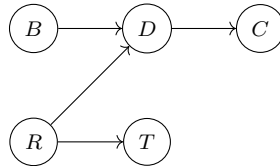


Figure 7.4: BN of the burglar and raccoon problem

According to the network, the joint probability distribution for this scenario can be factorized as:

$$p(B, D, C, T, R) = p(B)p(R)p(D|B, R)p(C|D)p(T|R)$$

#### Fewer parameters in a Bayesian network representation

Comparing the number of parameters in the complete model versus the BN, we find that the BN representation is more compact due to the introduced structural constraints.

**graphical models  
have fewer  
parameters**

The complete model requires  $2^5 - 1 = 31$  parameters. In contrast, the BN, due to its structure, reduces the number of parameters to 10 as counted below.

Probability Distribution	Number of Parameters
$P(C D)$	2
$P(D B, R)$	4
$P(B)$	1
$P(T R)$	2
$P(R)$	1
Total	10

Thus, the BN model significantly reduces the number of parameters to less than one-third of the complete model.

#### Pre-sum-product algorithm for inference

Let's estimate the probability of a burglar visiting given that the cop came but the trash can didn't fall. That is, we want to find  $p(b|c, \neg t)$ .

To compute that, we need to know the prior probabilities and conditional probabilities, which are given below:

$$p(r) = 0.2, \quad p(b) = 0.01.$$

$B$	$R$	$p(d B, R)$	$R$	$p(t R)$	$D$	$p(c D)$
$\neg b$	$\neg r$	0.1	$\neg r$	0.05	$\neg d$	0.01
$\neg b$	$r$	0.5	$r$	0.7	$d$	0.4
$b$	$\neg r$	0.85				
$b$	$r$	0.99				

Let's first find the joint probability  $p(b, c, \neg t)$ .

$$p(b, c, \neg t) = p(b)p(c|d)p(d|b, r)p(\neg t|r)p(r) + p(b)p(c|d)p(d|b, \neg r)p(\neg t|\neg r)p(\neg r) \\ + p(b)p(c|\neg d)p(\neg d|b, r)p(\neg t|r)p(r) + p(b)p(c|\neg d)p(\neg d|b, \neg r)p(\neg t|\neg r)p(\neg r) \quad (7.1)$$

$$= p(b)[p(c|d)(p(d|b, r)p(\neg t|r)p(r) + p(d|b, \neg r)p(\neg t|\neg r)p(\neg r)) \\ + p(c|\neg d)(p(\neg d|b, r)p(\neg t|r)p(r) + p(\neg d|b, \neg r)p(\neg t|\neg r)p(\neg r))] \quad (7.2) \\ = p(b) \cdot (p(c|d) \cdot (0.99 \cdot 0.3 \cdot 0.2 + 0.85 \cdot 0.95 \cdot 0.8) \\ + p(c|\neg d) \cdot (0.01 \cdot 0.3 \cdot 0.2 + 0.15 \cdot 0.95 \cdot 0.8)) \\ = p(b) \cdot (0.4 \cdot 0.7054 + 0.01 \cdot 0.1146) \\ = 0.01 \cdot 0.2833 = 0.002833$$

Similarly, we can find  $p(\neg b, c, \neg t) = 0.049045$ .

Consequently,

$$p(b|c, \neg t) = \frac{p(b, c, \neg t)}{p(c, \neg t)} \\ = \frac{p(b, c, \neg t)}{p(b, c, \neg t) + p(\neg b, c, \neg t)} \\ = \frac{0.002833}{0.002833 + 0.049045} = 0.055$$

Now, let's revisit the earlier computation. Probability inference often involves computing marginal probabilities, as shown earlier. Marginal probabilities can be computed by summing joint probabilities naively, as in (7.1). However, as shown in (7.2), we can pull out common factors from the joint probabilities to defer the product later and save computation. For example, there are a total of  $4 \cdot 4 = 16$  multiplications and 3 additions in (7.1). In contrast, there are only  $2 \cdot 4 + 2 + 1 = 11$  multiplications and 3 additions in (7.2). While the computational saving may seem small in the example above, the gain is significant as the number of variables increases and when the graphical model is sparse. **pull out common factors can save computation**

We will see in a later section that the BP algorithm is essentially a message-passing procedure to compute marginal probabilities systematically, as in (7.2). Before delving into the BP algorithm, we will first introduce two other important graphical models: undirected graphs and factor graphs.

## 7.2 Undirected graphs and factor graphs

In the context of Bayesian inference, an undirected graph (also known as a Markov network or **undirected Markov random field**) encodes the conditional independence relationships between random variables. In such a graph, nodes represent random variables, and edges represent direct probabilistic **graph**

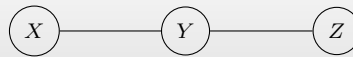
dependencies between them. Unlike BNs, which imply causal relationships, undirected graphs represent mutual relationships without implying direction or causality. These graphs are particularly useful for modeling situations where the direction of influence is either symmetric, not well-defined, or unknown.

Compared to BNs, undirected graphs provide a more intuitive and straightforward approach to reasoning about dependencies between variables. In an undirected graph, the independence of two variables can be determined by simply checking if they are connected by any path: if they are not connected, they are independent. Furthermore, two variables are conditionally independent given some variables if they become disconnected when the nodes corresponding to the given variables are removed from the graph. This simplicity makes undirected graphs an attractive choice for modeling dependencies.

**independence conditions in undirected graphs**

### Example 7.3: Three-variable undirected graph

Consider a simple undirected graph with three variables  $X$ ,  $Y$ , and  $Z$  with two edges  $X - Y$  and  $Y - Z$  as shown below.

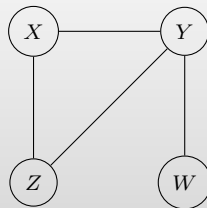


Variables  $X$  and  $Z$  are then conditionally independent given  $Y$ . The joint distribution can be written as  $p(y)p(x|y)p(z|y)$ .

An important concept in undirected graph is clique, which is defined in the following.

### Clique

A clique in a graph is a subset of nodes such that every two distinct nodes are adjacent. As in the example below,  $\{X, Y, Z\}$  and  $\{Y, W\}$  are two different cliques. But  $\{X, Y, Z, W\}$  and  $\{Y, W, Z\}$  are not.



**clique**

## 7.2.1 Factor graph representation

By a surprising result known as the Hammersley-Clifford theorem, the joint probability of any undirected graph model is proportional to the product of factor functions of the form  $\prod_i \phi_i(\mathbf{x}_i)$ , where  $\mathbf{x}_i$  contains variables that form a clique. For example, the joint probability of the graph in Example 7.3 can be rewritten as  $f_1(x, y)f_2(y, z)$  with  $f_1(x, y) = p(x|y)$  and  $f_2(y, z) = p(y)p(z|y)$ . We can use a bipartite graph to represent the factor product above in the following way:

1. **Represent each factor:** Represent each factor in the product with a vertex (typically known

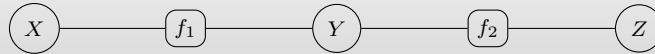
as a factor node and displayed with a square by convention). A factor corresponds to a clique of variables that are fully connected to each other.

2. **Represent each random variable:** Represent each random variable with a vertex (typically known as a variable node and displayed with a circle by convention).
3. **Connect factor nodes to arguments:** Connect each factor node to the corresponding variable nodes of its arguments with undirected edges.

The graph constructed above is known as a **factor graph**. While BP can be applied directly to undirected graphs, the factor graph is the most suitable representation for implementing BP. To apply BP for Bayesian inference, we typically first convert a BN to an undirected graph and then to a factor graph. The conversion of a BN to an undirected graph is done through a procedure called **moralization**, which will be discussed in the next subsection.

#### Example 7.4: Three-variable factor graph

Consider the undirected graph described in Example 7.3, there are only two cliques  $\{X, Y\}$  and  $\{Y, Z\}$ . Creating two factor nodes for each of the cliques and connecting respective variable nodes to the factor nodes, we have the respective factor graph as shown below:



### 7.2.2 The moralization of Bayesian networks

Consider again the simple BN with variables  $X$ ,  $Y$ , and  $Z$ , and edges  $X \rightarrow Y$  and  $Z \rightarrow Y$  (Figure 7.5b). What is the corresponding undirected graph that represents this structure?

A naive approach would be to simply replace the directed edges with undirected edges (Figure 7.5b). However, this fails to capture the dependency between  $X$  and  $Z$ . In the original model, when  $Y$  is observed or given,  $X$  and  $Z$  are no longer independent. But the undirected graph in Figure 7.5b implies that  $X$  and  $Z$  are conditionally independent given  $Y$ , which is not true in the original model. To address this, we need to add an additional edge  $X - Z$  (Figure 7.5).

This process is known as *moralization*, analogous to “moralizing” a child born out of wedlock by having the parents get married. **moralization**

Note that even after moralization, the resulting undirected graph (Figure 7.5c) still does not perfectly capture the properties of the original BN (Figure 7.5a). For instance,  $X$  and  $Z$  are independent in the original model, but there is no such assumption in the moralized model. This imperfect conversion is an inherent limitation of undirected graphs, which cannot perfectly capture properties of some BNs. Similarly, there are undirected graphs that cannot be perfectly represented by BNs, as we will discuss in the next subsection.

### 7.2.3 Limitations of different graphical models

One might expect that BNs have greater representation power than undirected graphs, since the edge direction information is lost in the latter case. However, it turns out that there are also undirected graphs that cannot be represented by BNs. In fact, both models have their own strengths

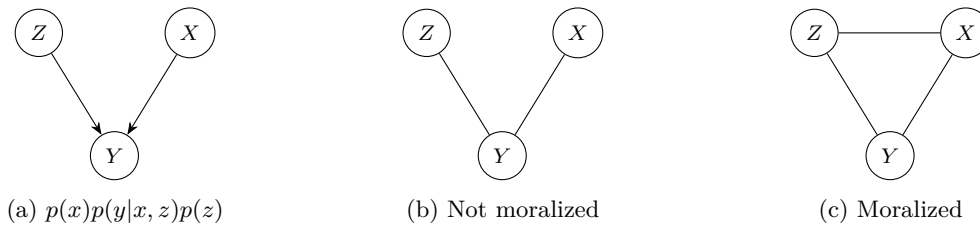


Figure 7.5: A BN (left) is converted to an undirected graph. Simply converting directed edges to undirected edges (middle) is insufficient. The middle graph implies  $X \perp Z|Y$ , a property that does not exist in the original model. To ensure that the resulting undirected graph does not introduce properties that do not exist in the original graph, parents of any node must be connected (right). This process is known as moralization.

and limitations, and neither has strictly greater representation power than the other. This means that there are certain probability distributions that can be represented by a BN but not by an undirected graph, and vice versa. This highlights the importance of choosing the appropriate model for the specific problem at hand.

### Undirected graphs cannot represent all Bayesian networks

We have seen that the BN in Figure 7.5a cannot be perfectly represented by any undirected graph. Even after moralization, which adds an edge between  $X$  and  $Z$ , the resulting fully connected graph with three vertices  $X$ ,  $Y$ , and  $Z$  still fails to capture the independence between  $X$  and  $Z$  present in the original BN.

So why bother with moralization at all? The reason is that adding edges reduces the assumptions built into the model. Solving a relaxed problem with fewer assumptions may make the problem more challenging, but it won't lead to incorrect solutions. On the other hand, adding incorrect assumptions that don't exist in the original problem can yield severely wrong results. Therefore, the moralization step is crucial when converting directed to undirected graphs. By keeping all parents "in wedlock," we ensure that the converted model does not introduce unwarranted assumptions, even if it cannot perfectly capture the original model's properties.

### Bayesian networks cannot represent all undirected graphs

Does this mean that BNs have higher representation power than undirected graphs? Not necessarily, as there are undirected graphs that cannot be captured by BNs either.

Consider the undirected graph with four variables  $X, Y, Z$ , and  $W$  and four edges  $X - Y$ ,  $Y - W$ ,  $Z - W$ , and  $Z - X$  as shown in Figure 7.6a. This graph captures the conditional independence of  $X$  and  $W$  given  $Y$  and  $Z$ . However, this conditional independence is only captured by one of the three possible BNs as in Figures 7.6b, 7.6c, and 7.6d.

When we convert these BNs to undirected graphs through moralization, we are required to add an additional edge between parents  $Y$  and  $Z$ , resulting in the undirected graph shown in Figure 7.6e. However, this graph fails to capture the conditional independence between  $Y$  and  $Z$  given  $W$  and  $X$ . In conclusion, no BN can exactly capture the structure of the original undirected graph.

**model only  
representable by  
BNs**

**model only  
representable by  
undirected  
graphs**

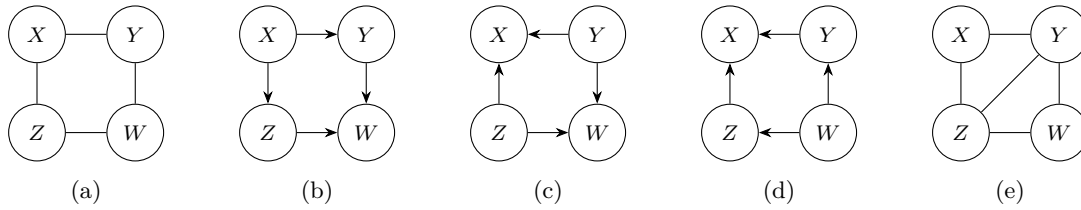


Figure 7.6: An undirected graph (a) cannot be represented by any BN. The undirected graph embeds the property  $X \perp W | \{Y, Z\}$ . The only BNs that satisfy this conditional independence property are shown in the middle ((b), (c), and (d)). However, by the requirement of moralization, these BNs all require an additional connection  $Y - Z$  when converted back to an undirected graph (e). This additional connection is not present in the original undirected graph (a), proving that it cannot be represented by any BN.

### 7.3 BP algorithm

The BP algorithm, also known as the sum-product algorithm, is a widely used method in probabilistic graphical models for performing inference on variables within a graph. It is exact for tree-structured graphs, meaning that it can potentially compute the exact marginal probabilities and most likely states in a single pass. For graphs with cycles, BP can be applied iteratively, but may not always converge to the exact solution.

The BP algorithm is best explained using a factor graph. Consider a factor graph with variables  $x_1, x_2, \dots, x_i, \dots$ , where numerical numbers denote variable nodes, and alphabets  $a, b, \dots$  denote factor nodes. Let's use  $\mathcal{N}(i)$  to denote the set of neighboring factor nodes of a variable node  $i$  and  $\mathcal{N}(a)$  to denote the set of neighboring variable nodes of the factor node  $a$ . Moreover, we write  $\mathbf{x}_a = \{x_j\}_{j \in \mathcal{N}(a)}$  as the set of variable connecting to factor node  $a$ . The BP algorithm can be summarized as follows:

- **Initialization:** For any factor node  $a$  connecting to only one variable node  $i$ , set  $m_{a \rightarrow i}(x_i) = f_a(\mathbf{x}_a) = f_a(x_i)$ .

**belief  
propagation**

- **Message passing:**

- **Variable node update:** For variable node  $i$  and factor node  $a \in \mathcal{N}(i)$ , update the message from variable node  $i$  to factor node  $a$  as

**BP variable  
update**

$$m_{i \rightarrow a}(x_i) = \prod_{b \in \mathcal{N}(i) \setminus a} m_{b \rightarrow i}(x_i). \quad (7.3)$$

- **Factor node update:** For factor node  $a$  and variable node  $i \in \mathcal{N}(a)$ , update the message from factor node  $a$  to variable node  $i$  as

**BP factor  
update**

$$m_{a \rightarrow i}(x_i) = \sum_{\mathbf{x}_a \setminus x_i} f_a(\mathbf{x}_a) \prod_{j \in \mathcal{N}(a) \setminus i} m_{j \rightarrow a}(x_j), \quad (7.4)$$

where  $\mathbf{x}_a = \{x_j\}_{j \in \mathcal{N}(a)}$  is the set of variable connecting to factor node  $a$ .



**BP belief update** • **Belief update:** For variable node  $i$ , update the belief as<sup>1</sup>

$$\beta_i(x_i) = \prod_{a \in \mathcal{N}(i)} m_{a \rightarrow i}(x_i). \quad (7.5)$$

- **Stopping criteria:** Repeat message update and/or belief update until the algorithm stops when the maximum number of iterations is reached or some other conditions are satisfied.

One aspect that was not specified in the description is the schedule of message passing. Typically, an alternating schedule is used, where variables pass messages to factor nodes, followed by factor nodes passing messages to variable nodes. Although more efficient scheduling is possible, it can be challenging to design. For instance, in the case of trees (graphs without loops), messages can be scheduled from leaves to root, allowing for exact marginal probabilities to be calculated in a single pass. However, when applying BP to graphs with loops<sup>2</sup>, the solution will be an approximation in any case, and optimal scheduling is difficult to achieve. As a result, a simple alternating schedule is commonly used.

**loopy BP**

### Intuitive interpretation of the BP algorithm

The message passing and update rules may seem mysterious at first, but they can be interpreted in a more intuitive way. Consider a factor-to-variable message as the belief that a factor node has about a variable node, and a variable-to-factor message as a variable node's self-belief.

**BP message as belief**

More precisely, we have

$$m_{a \rightarrow i}(x_i) \sim p(x_i)$$

from the perspective of factor node  $a$  and

$$m_{i \rightarrow a}(x_i) \sim p(x_i)$$

from the perspective of the variable node  $i$  thinking about itself. Then the variable node update in (7.3) simply states variable  $i$  pass knowledge about itself to factor  $a$  from all message other than  $a$  and assume the messages to be independent.

The factor node update in (7.4) can be interpreted in a similar manner if we consider

$$f_a(\mathbf{x}_a) \sim p(x_i | \mathbf{x}_{\mathcal{N}(a) \setminus i})$$

and

$$m_{j \rightarrow a}(x_j) \sim p(x_j)$$

<sup>1</sup>We may want to normalize the belief as it may not be inherently normalized. Specifically, we update  $\beta_i(x_i) \leftarrow \frac{\beta_i(x_i)}{\sum_{x_j} \beta_i(x_j)}$ . Although it is not absolutely necessary, normalizing the messages after each update can help ensure numerical stability.

<sup>2</sup>In the context of graphs with loops, BP is also referred to as loopy BP.

Then

$$m_{a \rightarrow i}(x_i) = \sum_{\mathbf{x}_a} f_a(\mathbf{x}_a) \prod_{j \in \mathcal{N}(a) \setminus i} m_{j \rightarrow a}(x_j) \quad (7.6)$$

$$\sim \sum_{\mathbf{x}_a \setminus x_i} p(x_i | \mathbf{x}_{\mathcal{N}(a) \setminus i}) \prod_{j \in \mathcal{N}(a) \setminus i} p(x_j) \quad (7.7)$$

$$= \sum_{\mathbf{x}_a \setminus x_i} p(\mathbf{x}_a) = p(x_i). \quad (7.8)$$

### Burglar and raccoon revisit

Let's revisit the burglar and raccoon problem using BP. We will begin by converting the BNs in Fig. 7.4 to an undirected graph, as shown in Fig. 7.7a, and then to a factor graph, as shown in Fig. 7.7b. Since the factor graph is a tree, we can use the BP algorithm to find the marginal probabilities of all variables in a single pass (from leaves to root and then back)<sup>3</sup>.

**BP in action**

#### Example 7.5: Burglar and raccoon problem

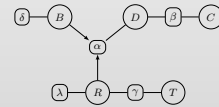
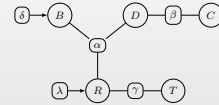
Let's do a step-by-step update procedure to illustrate how the messages will be computed. To avoid clutter, I exclude some messages such as  $m_{R \rightarrow \gamma}$  and  $m_{\gamma \rightarrow T}$ , but they can be computed by the same manner as the listed messages.

Initialization:

$$\begin{aligned} m_{\delta \rightarrow B}(b) &= f_{\delta}(b) = p(b) = 0.01, \\ m_{\lambda \rightarrow R}(r) &= f_{\lambda}(r) = p(r) = 0.2 \end{aligned}$$

Variable node update:

$$\begin{aligned} m_{B \rightarrow \alpha}(b) &= m_{\delta \rightarrow B}(b) = 0.01, \\ m_{R \rightarrow \alpha}(r) &= m_{\lambda \rightarrow R}(r) = 0.2 \end{aligned}$$



<sup>3</sup>In this example, although the final belief should be computed as the product of all neighboring messages, most messages are actually non-informative and can be ignored. For instance,  $\beta_D(d)$  would equal the product of messages from both  $\alpha$  and  $\beta$ , but it turns out that we can ignore the latter message,  $m_{\beta \rightarrow D}$ . Therefore,  $\beta_D(d) = m_{\alpha \rightarrow D}(d) = p(d)$  can be computed in one downward pass from leaves to root. In general, we need one downward pass and one upward pass to get the exact marginal probabilities for a tree graph. If the graph has loops, we will need more passes, and the solution may not even converge. However, for sparse loopy graphs, BP generally works quite well.

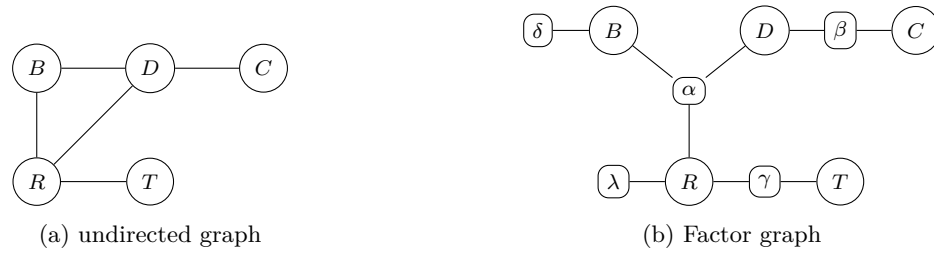
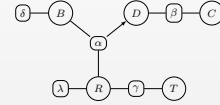


Figure 7.7: The undirected graph and factor graph representations of the burglar and raccoon problem are shown below. In the factor graph, a factor is formed for each clique, which includes  $\{B, D, R\}$ ,  $\{C, D\}$ , and  $\{R, T\}$ . Additionally, prior factors are included for  $B$  and  $R$ . The joint probability is equal to the product of the factors:  $f_\alpha(B, D, R)f_\beta(C, D)f_\gamma(R, T)f_\delta(B)f_\lambda(R)$  where  $f_\alpha(B, D, R) = p(D|B, R)$ ,  $f_\beta(C, D) = p(C|D)$ ,  $f_\gamma(R, T) = p(T|R)$ ,  $f_\delta(B) = p(B)$ , and  $f_\lambda(R) = p(R)$ .

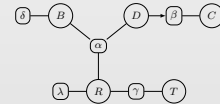
Factor node update:

$$\begin{aligned}
 m_{\alpha \rightarrow D}(d) &= \sum_{B,R} f_\alpha(B, R, d) m_{B \rightarrow \alpha}(B) m_{R \rightarrow \alpha}(R) \\
 &= 0.1 \cdot m_{B \rightarrow \alpha}(-b) m_{R \rightarrow \alpha}(-r) \\
 &\quad + 0.5 \cdot m_{B \rightarrow \alpha}(-b) m_{R \rightarrow \alpha}(r) \\
 &\quad + 0.85 \cdot m_{B \rightarrow \alpha}(b) m_{R \rightarrow \alpha}(-r) \\
 &\quad + 0.99 \cdot m_{B \rightarrow \alpha}(b) m_{R \rightarrow \alpha}(r) \\
 &= 0.1 \cdot 0.99 \cdot 0.8 + 0.5 \cdot 0.99 \cdot 0.2 \\
 &\quad + 0.85 \cdot 0.01 \cdot 0.8 + 0.99 \cdot 0.01 \cdot 0.2 \\
 &= 0.187
 \end{aligned}$$



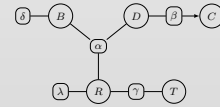
Variable node update:

$$m_{D \rightarrow \beta}(d) = m_{\alpha \rightarrow D}(d) = 0.187,$$



Factor node update:

$$\begin{aligned}
 m_{\beta \rightarrow C}(c) &= \sum_D f_\beta(c, D) m_{D \rightarrow \beta}(D) \\
 &= 0.01 \cdot m_{D \rightarrow \beta}(-d) + 0.4 \cdot m_{D \rightarrow \beta}(d) \\
 &= 0.01 \cdot (1 - 0.187) + 0.4 \cdot 0.187 \\
 &= 0.083
 \end{aligned}$$



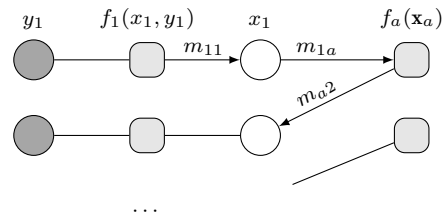
### 7.3.1 Low-density parity-check code

Low-Density Parity-Check (LDPC) codes are a class of error-correcting codes that play a crucial **LDPC codes** role in ensuring reliable data transmission over noisy channels in digital communication systems.

In LDPC codes, the term “parity-check” refers to the constraints imposed by the parity-check matrix on the codewords. Each row of the matrix represents a parity-check equation, a linear constraint that the bits of a valid codeword must satisfy. Specifically, a codeword is valid if the modulo-2 sum (XOR) of certain bits equals zero. These parity-check equations ensure the necessary structure for detecting and correcting errors during data transmission or storage.

The “low-density” aspect refers to the sparsity of the parity-check matrix used to define the code. This matrix is sparse, containing relatively few non-zero entries (ones) compared to zeros. The sparse structure is crucial because it enables efficient encoding and decoding using iterative algorithms like BP, making the computations practical and fast.

The major challenge was decoding LDPC codes efficiently. Due to the lack of structure in random codes, the tricks that enable fast decoding for structured algebraic codes were no longer applicable. Fortunately, despite the factor graph of a LDPC code having loops (not a tree), the sparse structure ensures that BP works well for LDPC codes, making BP a great decoding choice.



#### Decoding of LDPC code

Let’s represent the variable structure of an LDPC code using a factor graph, as shown in Fig. 7.8. The transmitted bits are represented by  $x_1, \dots, x_N$  (white), and the received bits are represented by  $y_1, \dots, y_N$  (dark grey). Each square represents a check bit with a value equal to the sum of the code bits connecting to it. The vector  $x_1, x_2, \dots, x_N$  is a codeword only if all checks are zero.

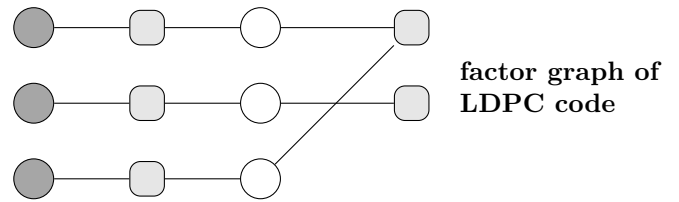


Figure 7.8: The factor graph of a LDPC code

The channel is modeled by factor functions  $f_i(x_i, y_i)$ , which are typically stationary. For a binary symmetric channel (BSC) with crossover probability  $q$ ,

$$f_i(x_i, y_i) = p(y_i|x_i) = \begin{cases} q, & \text{if } y_i \neq x_i, \\ 1 - q, & \text{otherwise.} \end{cases}$$

Upon receiving  $y_i$ , the message from observed variable node  $y_i$  to factor node  $f_i(x_i, y_i)$  is simply  $\delta(y, y_i)$ . Consequently, the message from factor node  $f_i(x_i, y_i)$  to variable node  $x_i$  is

$$m_{i \rightarrow i}(x_i) = \sum_y f_i(x_i, y) \delta(y, y_i) = f_i(x_i, y_i).$$

Note that this message does not change over time during decoding since the observation  $y_i$  does not change. Therefore, we only need to update messages for variable nodes  $x_i$  and factor nodes  $f_a(\cdot)$ .

For the factor node  $f_a(\mathbf{x})$  for the parity check  $a$ ,

$$f_a(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \text{ contains an even number of 1s,} \\ 1, & \text{if } \mathbf{x} \text{ contains an odd number of 1s.} \end{cases}$$

Now that we have defined the messages, we are ready to derive the BP update steps. Since the unknown variables are binary, it is more convenient to represent the messages using likelihood or log-likelihood ratios. We define the likelihood ratio as

**likelihood and  
log-likelihood  
ratios**

$$l_{a \rightarrow i} \triangleq \frac{m_{a \rightarrow i}(0)}{m_{a \rightarrow i}(1)}, \quad L_{a \rightarrow i} \triangleq \log l_{a \rightarrow i}$$

and

$$l_{i \rightarrow a} \triangleq \frac{m_{i \rightarrow a}(0)}{m_{i \rightarrow a}(1)}, \quad L_{i \rightarrow a} \triangleq \log l_{i \rightarrow a}$$

**LDPC variable  
update**

for any variable node  $i$  and factor node  $a$ . Then, the update for the variable node  $x_i$  is

$$L_{i \rightarrow a} = \sum_{b \in \mathcal{N}(i) \setminus a} L_{b \rightarrow i}.$$

Next, let's consider the factor node update for check node  $a$ . Without loss of generality, assume three variable nodes 1, 2, and 3 are connected to the check node  $a$ . The updates for the messages from the check node to the variable nodes are

$$m_{a \rightarrow 1}(1) = m_{2 \rightarrow a}(1)m_{3 \rightarrow a}(0) + m_{2 \rightarrow a}(0)m_{3 \rightarrow a}(1)$$

and

$$m_{a \rightarrow 1}(0) = m_{2 \rightarrow a}(0)m_{3 \rightarrow a}(0) + m_{2 \rightarrow a}(1)m_{3 \rightarrow a}(1).$$

By substituting in the likelihood ratios and log-likelihood ratios, we obtain

$$l_{a \rightarrow 1} \triangleq \frac{m_{a \rightarrow 1}(0)}{m_{a \rightarrow 1}(1)} = \frac{1 + l_{2 \rightarrow a}l_{3 \rightarrow a}}{l_{2 \rightarrow a} + l_{3 \rightarrow a}}$$

and

$$e^{L_{a \rightarrow 1}} = l_{a \rightarrow 1} = \frac{1 + e^{L_{2 \rightarrow a}}e^{L_{3 \rightarrow a}}}{e^{L_{2 \rightarrow a}} + e^{L_{3 \rightarrow a}}}.$$

Note that we have a neat factor node update rule as follows

$$\tanh\left(\frac{L_{a \rightarrow 1}}{2}\right) = \frac{e^{\frac{L_{a \rightarrow 1}}{2}} - e^{-\frac{L_{a \rightarrow 1}}{2}}}{e^{\frac{L_{a \rightarrow 1}}{2}} + e^{-\frac{L_{a \rightarrow 1}}{2}}} = \frac{e^{L_{a \rightarrow 1}} - 1}{e^{L_{a \rightarrow 1}} + 1} \quad (7.9)$$

$$= \frac{1 + e^{L_{2 \rightarrow a}} e^{L_{3 \rightarrow a}} - e^{L_{2 \rightarrow a}} - e^{L_{3 \rightarrow a}}}{1 + e^{L_{2 \rightarrow a}} e^{L_{3 \rightarrow a}} + e^{L_{2 \rightarrow a}} + e^{L_{3 \rightarrow a}}} \quad (7.10)$$

$$= \frac{(e^{L_{2 \rightarrow a}} - 1)(e^{L_{3 \rightarrow a}} - 1)}{(e^{L_{2 \rightarrow a}} + 1)(e^{L_{3 \rightarrow a}} + 1)} \quad (7.11)$$

$$= \tanh\left(\frac{L_{2 \rightarrow a}}{2}\right) \tanh\left(\frac{L_{3 \rightarrow a}}{2}\right). \quad (7.12)$$

When we have more than three variable nodes connecting to the check node  $a$ , it is easy to show using induction that

$$\tanh\left(\frac{L_{a \rightarrow i}}{2}\right) = \prod_{j \in \mathcal{N}(a) \setminus i} \tanh\left(\frac{L_{j \rightarrow a}}{2}\right).$$

**LDPC factor  
update**

### 7.3.2 BP and statistical physics

Previously, we have considered BP as a method for computing inference in graph models. Specifically, given a probability distribution  $p(\mathbf{x})$  and a particular graph model, we aimed to find the marginal probability  $p(x_i)$  for some node  $i$ .

Now, we will explore BP from a different perspective—that of statistical physics. We will demonstrate how BP can be derived using the Bethe approximation, a method commonly employed in statistical physics.

In statistical physics, for a system with a number of states with energy  $E(\mathbf{x})$ , the probability of state  $\mathbf{x}$  under Boltzmann statistics is given by

$$p(\mathbf{x}) = \frac{e^{-\frac{E(\mathbf{x})}{kT}}}{\sum_{\mathbf{x}} e^{-\frac{E(\mathbf{x})}{kT}}} \triangleq \frac{e^{-\frac{E(\mathbf{x})}{kT}}}{Z},$$

**Boltzmann  
distribution**

where  $T$  is the temperature of the system,  $k$  is the Boltzmann constant, and  $Z$  is the partition function, essentially serving as a normalization factor. Without loss of generality, we can “absorb”  $kT$  into  $E(\mathbf{x})$  by redefining the energy, assuming  $kT = 1$ . Therefore, we have

$$p(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z}. \quad (7.13)$$

Given the energy of any particular state  $\mathbf{x}$ , we can specify the average internal energy of any ensemble of systems with any distribution  $b(\mathbf{x})$  as

**Boltzmann  
internal energy**

$$U(b) = \sum_{\mathbf{x}} b(\mathbf{x}) E(\mathbf{x}).$$

**Boltzmann entropy**

Moreover, the average entropy of the ensemble of systems will be

$$H(b) = - \sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x}).$$

**variational free energy**

The quantity  $F(b) = U(b) - H(b)$  is important and is sometimes known as the Gibbs free energy in statistical physics. In the context of variational inference<sup>4</sup>, this is also referred to as the variational free energy. Note that

$$\begin{aligned} F(b) &= U(b) - H(b) \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) E(\mathbf{x}) + \sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x}) \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) (E(\mathbf{x}) + \ln b(\mathbf{x})) \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) \left( \ln b(\mathbf{x}) + E(\mathbf{x}) - \ln e^{-E(\mathbf{x})} \right) \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) \ln \frac{b(\mathbf{x})}{e^{-E(\mathbf{x})}} \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) \ln \frac{b(\mathbf{x})}{Zp(\mathbf{x})} \\ &= -\ln Z + KL(b \parallel p), \end{aligned}$$

where  $KL(b \parallel p)$  is the KL-divergence, which is always non-negative and is equal to zero only when  $b = p$ . Therefore,  $F(b)$  takes its minimum value  $-\ln Z$  when  $b = p$ .

We then have an alternative way to find  $p$  given  $E(\mathbf{x})$ , namely,

$$p = \arg \min_b F(b).$$

**why minimize free energy?**

At first glance, this may seem redundant since we can obtain  $p(\mathbf{x})$  directly from the Boltzmann distribution equation. However, there are practical reasons for using this approach

- We may not have an explicit form of the normalization factor (partition function  $Z$ ).
- We might not know the precise form of  $E(\mathbf{x})$ . For example, consider the energy states of atoms within a large molecule, where each variable represents the location of an atom. The energy depends on the complex interactions between all atoms, making it difficult to determine exactly.
- Instead of the joint distribution  $p(\mathbf{x})$ , we might be interested in marginal distributions. For example, we may care more about the distribution of a particular atom's location rather than the joint distribution of all atoms. This is analogous to our discussion on BP, where the primary purpose is finding marginal distributions.

---

<sup>4</sup>The variational inference refers to the approach that involves approximating a complex probability distribution by a simpler, tractable one. This is done by optimizing some functional, often an energy or a free energy, with respect to the parameters of the simpler distribution.

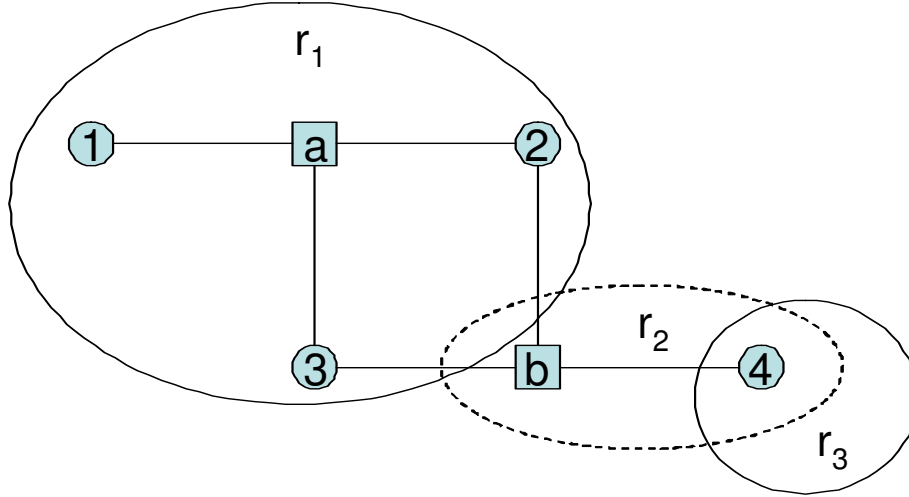


Figure 7.9: A region is defined as a set of nodes that includes all neighboring variable nodes if a factor node is included. For instance,  $r_1$ , which comprises nodes  $a, 1, 2$ , and  $3$ , is a valid region. Similarly,  $r_3$ , which only consists of node  $4$ , is also a valid region. On the other hand,  $r_2$  is not a valid region because it includes factor node  $b$  but excludes nodes  $2$  and  $3$ , which are neighboring variable nodes.

Now, let us incorporate our familiar factor notation. Suppose we have a system with a known distribution described by its factor graph, where the probability of a state  $\mathbf{x}$  is written as

$$p(\mathbf{x}) = \frac{\prod_a^M f_a(\mathbf{x}_a)}{Z}, \quad (7.14)$$

**Boltzmann  
factor  
representation**

where  $f_a(\cdot)$  is a factor function and  $\mathbf{x}_a$  is a subset of variables connecting to factor node  $a$ . We can establish the energy of each state from  $p(\mathbf{x})$ . Note that we denote  $\mathbf{x}_a$  as a subset of all state variables  $x_1, x_2, \dots, x_N$ . We assume there are  $M$  such factors and we denote  $\prod_a^M$  as the product over all factor nodes.

Comparing (7.13) and (7.14), we see that

$$E(\mathbf{x}_a) = -\ln(f_a(\mathbf{x}_a)). \quad (7.15)$$

### Region-Based Approximation

If we can optimize  $F(b)$  and find its marginal  $b_i(x_i)$  simultaneously, we can obtain  $p(x_i)$ , the marginal probability for a particular node  $i$ . This is equivalent to the inference problem addressed by BP. However, optimizing  $F(b)$  directly is challenging due to the dependencies introduced by factor node connections. Instead, we seek an approximation of  $F(b)$  by decomposing it into a sum of contributions from isolated regions.

In a factor graph, a region is defined as any combination of factor nodes and their neighboring **region** variable nodes. Figure 7.9 illustrates examples and a counterexample of regions. For any region  $r$ , we can define the average energy and entropy as



$$U_r(b) = \sum_{a \in r} \sum_{\mathbf{x}_a} b(\mathbf{x}_a) \ln f_a(\mathbf{x}_a)$$

and

$$H_r(b) = - \sum_{a \in r} \sum_{\mathbf{x}_a} b(\mathbf{x}_a) \ln b(\mathbf{x}_a).$$

Suppose we partition all nodes into a set of regions  $\mathcal{R} = \{r_1, r_2, \dots, r_K\}$ . We can approximate  $F(b)$  as

$$\tilde{F}(b) = \sum_{r \in \mathcal{R}} U_r(b) - H_r(b)$$

However, this approximation often leads to significant overcounting of energy by including some nodes multiple times. We can modify  $\tilde{F}(b)$  to:

$$\hat{F}(b) = \sum_{r \in \mathcal{R}} c_r U_r(b) - c_r H_r(b)$$

where  $c_r$  are as counting numbers that serve as scaling factors, ensuring that

$$\sum_{r \in \mathcal{R}} c_r I(a) = \sum_{r \in \mathcal{R}} c_r I(i) = 1 \quad (7.16)$$

for any factor node  $a$  and variable node  $i$ .

### Bethe Approximation

#### Bethe approximation

Bethe approximation is just a very simple case of region-based approximation with the set of all regions  $\mathcal{R} = \mathcal{R}_S \cup \mathcal{R}_L$ , where

- $\mathcal{R}_S$ : the set of small regions where each small region is composed of a variable node
- $\mathcal{R}_L$ : the set of large regions where each large region is composed of a factor node and its corresponding neighboring variable nodes
- $c_r = 1$  if  $r \in \mathcal{R}_L$
- $c_r = 1 - d_r$  if  $r \in \mathcal{R}_S$ , where  $d_r$  is the degree of the single variable node in  $r$ .

We can easily verify that the counting number conditions in (7.16) are satisfied. Further, we have

$$U_{Bethe}(b) = - \sum_a^M \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \ln f_a(\mathbf{x}_a),$$

$$H_{Bethe}(b) = - \sum_a^M \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) \ln b_a(\mathbf{x}_a) + \sum_{i=1}^N (d_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i),$$

and

$$F_{Bethe}(b) = U_{Bethe} - H_{Bethe}.$$

Now, we can start optimizing  $F_{Bethe}$  over  $b$ . Note that since  $b$  is a probability distribution, its marginals  $b_i(x_i)$  and  $b_a(\mathbf{x}_a)$  have to satisfy

$$\sum_{x_i} b_i(x_i) = 1, \forall i$$

and

$$\sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) = 1, \forall a.$$

Moreover, since  $\mathbf{x}_a$  really represent  $\{x_i | i \in \mathcal{N}(a)\}$ , where  $\mathcal{N}(a)$  is the set of all neighbors of  $a$ . Therefore, we need to have

$$\sum_{\mathbf{x}_a \setminus x_i} b_a(\mathbf{x}_a) = b_i(x_i), \forall a, \forall i \in \mathcal{N}(a), \text{ and } \forall x_i.$$

Incorporate the above constraints into the Lagrangian formulation, let

$$\begin{aligned} L(b) = F_{Bethe}(b) &+ \sum_a \gamma_a \left( \sum_{\mathbf{x}_a} b_a(\mathbf{x}_a) - 1 \right) + \sum_i \gamma_i \left( \sum_{x_i} b_i(x_i) - 1 \right) \\ &+ \sum_a \sum_{i \in \mathcal{N}(a)} \sum_{x_i} \lambda_{ai}(x_i) \left( \sum_{\mathbf{x}_a \setminus x_i} b_a(\mathbf{x}_a) - b_i(x_i) \right). \end{aligned}$$

Take derivative of  $L(b)$  over  $b_c(\tilde{\mathbf{x}}_c)$  and  $b_j(\tilde{x}_j)$  and set them to 0. After some computation, we have

$$-\ln f_c(\tilde{\mathbf{x}}_c) + \ln \hat{b}_c(\tilde{\mathbf{x}}_c) + 1 + \gamma_c - \sum_{i \in \mathcal{N}(c)} \lambda_{ci}(\tilde{x}_i) = 0 \quad (7.17)$$

and

$$-(d_j - 1) \ln \hat{b}_j(\tilde{x}_j) - d_j + 1 + \gamma_j + \sum_{a \in \mathcal{N}(j)} \lambda_{aj}(\tilde{x}_j) = 0. \quad (7.18)$$

This gives us

$$\hat{b}_c(\tilde{\mathbf{x}}_c) = f_c(\tilde{\mathbf{x}}_c) \exp \left[ -1 - \gamma_c + \sum_{i \in \mathcal{N}(c)} \lambda_{ci}(\tilde{x}_i) \right] \quad (7.19)$$

and

$$\hat{b}_j(\tilde{x}_j) = \exp \left[ \frac{1}{(d_j - 1)} \left( -d_j + 1 + \gamma_j + \sum_{a \in \mathcal{N}(j)} \lambda_{aj}(\tilde{x}_j) \right) \right] \quad (7.20)$$

With some profound inspiration, one may identify  $\lambda_{ci}(\tilde{x}_i)$  as the log-variable-to-factor node message **Lagrange multiplier as log-message** in BP, namely,

$$\lambda_{ci}(\tilde{x}_i) = \ln m_{i \rightarrow c}(\tilde{x}_i) = \ln \prod_{d \in \mathcal{N}(i) \setminus c} m_{d \rightarrow i}(\tilde{x}_i). \quad (7.21)$$

Substituting (7.21) into (7.19) and (7.20), we have

$$\hat{b}_c(\tilde{\mathbf{x}}_c) \propto f_c(\tilde{\mathbf{x}}_c) \prod_{i \in \mathcal{N}(c)} m_{i \rightarrow c}(\tilde{x}_i) \quad (7.22)$$

and

$$\begin{aligned} \hat{b}_j(\tilde{x}_j) &\propto \exp \left[ \frac{1}{(d_j - 1)} \left( \sum_{a \in \mathcal{N}(j)} \sum_{d \in \mathcal{N}(j) \setminus a} \ln m_{d \rightarrow j}(\tilde{x}_j) \right) \right] \\ &\propto \exp \left[ \frac{1}{(d_j - 1)} \left( (d_j - 1) \sum_{d \in \mathcal{N}(j)} \ln m_{d \rightarrow j}(\tilde{x}_j) \right) \right] \\ &\propto \prod_{d \in \mathcal{N}(j)} m_{d \rightarrow j}(\tilde{x}_j). \end{aligned} \quad (7.23)$$

Finally, for a variable node  $i$  adjacent to a factor node  $a$ , since  $\hat{b}_i(x_i)$  is marginal distribution of  $\hat{b}_a(\mathbf{x}_a)$ , we have

variable update  
implies factor  
update under  
Bethe  
approximation

$$\begin{aligned} &\sum_{\mathbf{x}_a \setminus x_i} \hat{b}_a(\mathbf{x}_a) &&= \hat{b}_i(x_i) \\ \Rightarrow &\sum_{\mathbf{x}_a \setminus x_i} f_a(\mathbf{x}_a) \prod_{j \in \mathcal{N}(a)} m_{j \rightarrow a}(x_j) &&= \prod_{c \in \mathcal{N}(i)} m_{c \rightarrow i}(x_i) \\ \Rightarrow &m_{i \rightarrow a}(x_i) \sum_{\mathbf{x}_a \setminus x_i} f_a(\mathbf{x}_a) \prod_{j \in \mathcal{N}(a) \setminus i} m_{j \rightarrow a}(x_j) &&= \prod_{c \in \mathcal{N}(i)} m_{c \rightarrow i}(x_i) \\ \stackrel{(a)}{\Rightarrow} &\prod_{c \in \mathcal{N}(i) \setminus a} m_{c \rightarrow i}(x_i) \sum_{\mathbf{x}_a \setminus x_i} f_a(\mathbf{x}_a) \prod_{j \in \mathcal{N}(a) \setminus i} m_{j \rightarrow a}(x_j) &&= \prod_{c \in \mathcal{N}(i)} m_{c \rightarrow i}(x_i) \\ \Rightarrow &\sum_{\mathbf{x}_a \setminus x_i} f_a(\mathbf{x}_a) \prod_{j \in \mathcal{N}(a) \setminus i} m_{j \rightarrow a}(x_j) &&= m_{a \rightarrow i}(x_i), \end{aligned} \quad (7.24)$$

where (a) is derived from the belief propagation (BP) variable node update rule (7.3) and we observe that we can recover the BP factor node update rule (7.4) from (7.24). Therefore, if BP converges, it will converge to the solution of (7.22) and (7.23), i.e., the solution of the Bethe approximation.

## 7.4 Gaussian BP

In this section, we generalize the discrete Belief Propagation (BP) algorithm to the continuous case by introducing Gaussian BP. This variant parameterizes the model using Gaussian distributions, which have desirable properties that simplify the update steps.

Recall that the BP update steps in (7.3), (7.4), and (7.5) involve the product of distributions and marginalization. Fortunately, Gaussian distributions remain Gaussian under these operations, ensuring that the updated distributions stay Gaussian if we assume Gaussian initial distributions.

We will first review the properties of multivariate Gaussian distributions, including how their

parameters change under marginalization, conditioning, and multiplication. Then, we will derive the Gaussian BP update rules. Finally, we will demonstrate how the Kalman filter, a well-known algorithm, can be derived using the Gaussian BP framework.

### 7.4.1 Manipulating multivariate Gaussian

A multivariate Gaussian distribution is characterized by its mean vector  $\boldsymbol{\mu} = \mathbb{E}[\mathbf{X}]$  and covariance matrix  $\Sigma = \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\top]$ . Its probability density function (PDF) can be specified in two forms:

The moment form of the Gaussian PDF is given by

**moment form**

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad (7.25)$$

where the normalization factor  $\sqrt{\det(2\pi\Sigma)}$  depends only on  $\Sigma$  and not on  $\mathbf{x}$ . We can extract the parameters  $\boldsymbol{\mu}$  and  $\Sigma$  from the exponent alone, and the normalization factor can be ignored in most cases.

Alternatively, the Gaussian PDF can be written in the canonical form as

**canonical form**

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\eta}, \Lambda) = \frac{1}{Z(\boldsymbol{\eta}, \Lambda)} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Lambda \mathbf{x} + \boldsymbol{\eta}^\top \mathbf{x}\right), \quad (7.26)$$

where  $Z(\boldsymbol{\eta}, \Lambda)$  is a normalization factor independent of  $\mathbf{x}$ . The canonical form is often more convenient for operations such as product and conditioning. We will switch between the two forms as needed, as some operations are easier to handle in one form than the other.

From the moment form, we can derive the canonical form as follows:

$$\begin{aligned} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) &\propto \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \\ &= \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x} + \frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu} + \frac{1}{2}\boldsymbol{\mu}^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}^\top \Sigma^{-1}\boldsymbol{\mu}\right) \\ &\stackrel{(a)}{\propto} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x} + \frac{1}{2}(\Sigma^{-1}\boldsymbol{\mu})^\top \mathbf{x} + \frac{1}{2}(\Sigma^{-1}\boldsymbol{\mu})^\top \mathbf{x}\right) \\ &\stackrel{(b)}{=} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x} + (\Sigma^{-1}\boldsymbol{\mu})^\top \mathbf{x}\right) \\ &\stackrel{(c)}{=} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x} + \mathbf{x}^\top (\Sigma^{-1}\boldsymbol{\mu})\right), \end{aligned}$$

where we drop the last term in (a) as it does not depend on  $\mathbf{x}$ , and we use the fact that  $a^\top b = b^\top a$  when  $a$  and  $b$  are vectors in (a) and (c), and we use the fact that  $\Sigma$  and consequently  $\Sigma^{-1}$  are symmetric in (b).

Comparing to the canonical form in (7.26), we see that

**form conversion**

$$\Lambda = \Sigma^{-1}, \quad (7.27)$$

$$\boldsymbol{\eta} = \Sigma^{-1}\boldsymbol{\mu}. \quad (7.28)$$

The matrix  $\Lambda$  is also known as the precision matrix, and its eigenvalues determine the variation of the variable. The larger the eigenvalues of  $\Lambda$ , the smaller the variation, hence the name “precision”.

### Marginalizing Gaussian

Consider a joint Gaussian distribution for the random vector  $\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix}$  with mean  $\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}$  and covariance matrix  $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$ .

It is easy to show that marginalizing  $\mathbf{X}_2$  from  $\mathbf{X}$  will still result in a Gaussian distribution. Moreover, we have  $\mathbf{X}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_{11})$ . The proof is straightforward and is left as an exercise.

While it may not be immediately obvious that the resulting distribution after marginalization is Gaussian, it is intuitive that the mean and covariance should not change after marginalization. Therefore,

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_1, \quad (7.29)$$

$$\Sigma \leftarrow \Sigma_{11}. \quad (7.30)$$

Now, let's consider what happens if we parametrize with  $\boldsymbol{\eta}$  and  $\Lambda$  instead. We will use the following fact to help answer this question.

#### Schur complement

**Fact 7.1.** Assume  $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{pmatrix}$ , then  $A^{-1} = \tilde{A} - \tilde{B}\tilde{D}^{-1}\tilde{C}$ , where the right hand side is sometimes known as a Schur complement denoted as  $\begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{pmatrix} \Big| \tilde{D}$

*Proof.* Note that  $\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$ . Thus  $A\tilde{A} + B\tilde{C} = I$  and  $A\tilde{B} + B\tilde{D} = 0$ . So  $A(\tilde{A} - \tilde{B}\tilde{D}^{-1}\tilde{C}) = A\tilde{A} - (A\tilde{B})\tilde{D}^{-1}\tilde{C} = A\tilde{A} + B\tilde{D}\tilde{D}^{-1}\tilde{C} = A\tilde{A} + B\tilde{C} = I$   $\square$

Since  $\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}^{-1} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix}$ , we have

$$\Sigma_{11}^{-1} = \Lambda_{11} - \Lambda_{12}\Lambda_{22}^{-1}\Lambda_{21}$$

from Fact 7.1. Therefore, from (7.30), we have

$$\Lambda \leftarrow \Sigma_{11}^{-1} = \Lambda_{11} - \Lambda_{12}\Lambda_{22}^{-1}\Lambda_{21} \quad (7.31)$$

and since

$$\begin{pmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{pmatrix} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix} \Rightarrow \begin{cases} \boldsymbol{\eta}_1 = \Lambda_{11}\boldsymbol{\mu}_1 + \Lambda_{12}\boldsymbol{\mu}_2, \\ \boldsymbol{\eta}_2 = \Lambda_{21}\boldsymbol{\mu}_1 + \Lambda_{22}\boldsymbol{\mu}_2, \end{cases} \quad (7.32)$$

marginalize  
moment form

marginalize  
canonical form

$$\begin{aligned}
\boldsymbol{\eta} \leftarrow \Sigma_{11}^{-1} \boldsymbol{\mu}_1 &= (\Lambda_{11} - \Lambda_{12} \Lambda_{22}^{-1} \Lambda_{21}) \boldsymbol{\mu}_1 \\
&= \Lambda_{11} \boldsymbol{\mu}_1 + \Lambda_{12} \boldsymbol{\mu}_2 - \Lambda_{12} \boldsymbol{\mu}_2 - \Lambda_{12} \Lambda_{22}^{-1} \Lambda_{21} \boldsymbol{\mu}_1 \\
&= (\Lambda_{11} \boldsymbol{\mu}_1 + \Lambda_{12} \boldsymbol{\mu}_2) - \Lambda_{12} \Lambda_{22}^{-1} (\Lambda_{22} \boldsymbol{\mu}_2 + \Lambda_{21} \boldsymbol{\mu}_1) \\
&\stackrel{(a)}{=} \boldsymbol{\eta}_1 - \Lambda_{12} \Lambda_{22}^{-1} \boldsymbol{\eta}_2,
\end{aligned} \tag{7.33}$$

where (a) is from (7.32).

### Product of Gaussians

It is easier to manipulate product using the canonical form. So consider two Gaussian distributions  $\mathcal{N}(\mathbf{x}; \boldsymbol{\eta}_1, \Lambda_1)$  and  $\mathcal{N}(\mathbf{x}; \boldsymbol{\eta}_2, \Lambda_2)$ . The product is simply

$$\begin{aligned}
\mathcal{N}(\mathbf{x}; \boldsymbol{\eta}_1, \Lambda_1) \mathcal{N}(\mathbf{x}; \boldsymbol{\eta}_2, \Lambda_2) &\propto \exp \left( -\frac{1}{2} \mathbf{x}^\top \Lambda_1 \mathbf{x} + \boldsymbol{\eta}_1^\top \mathbf{x} - \frac{1}{2} \mathbf{x}^\top \Lambda_2 \mathbf{x} + \boldsymbol{\eta}_2^\top \mathbf{x} \right) \\
&= \exp \left( -\frac{1}{2} \mathbf{x}^\top (\Lambda_1 + \Lambda_2) \mathbf{x} + (\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2)^\top \mathbf{x} \right)
\end{aligned}$$

So we simply have

$$\boldsymbol{\eta} \leftarrow \boldsymbol{\eta}_1 + \boldsymbol{\eta}_2, \tag{7.34}$$

$$\Lambda \leftarrow \Lambda_1 + \Lambda_2. \tag{7.35}$$

**product of  
canonical forms**

If we parametrize with the moment form instead, we have

$$\boldsymbol{\mu} \leftarrow \Sigma \boldsymbol{\eta} = \Lambda^{-1} \boldsymbol{\eta} = (\Lambda_1 + \Lambda_2)^{-1} (\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2) = (\Lambda_1 + \Lambda_2)^{-1} (\Lambda_1 \boldsymbol{\mu}_1 + \Lambda_2 \boldsymbol{\mu}_2) \tag{7.36}$$

$$\Sigma \leftarrow \Lambda^{-1} = (\Lambda_1 + \Lambda_2)^{-1} \tag{7.37}$$

**product of  
moment forms**

Note that the product of the two distributions can be interpreted as the distribution of the predicted variable after incorporating two independent observations. Consequently, it is intuitive to view the resulting precision matrix as the sum of individual precision matrices, which “adds up” the information from each observation. Moreover, the resulting mean is a weighted sum of the means from the two observations, where the weights are proportional to the precision matrices of individual observations.

### Conditioning Gaussian

While conditioning was not used in the BP algorithm earlier, we can also incorporate it to address making direct observations of a variable. This is used, for example, in the Kalman filter example elaborated later in this chapter.

Consider  $\mathbf{Z} \sim \mathcal{N}(\boldsymbol{\mu}_Z, \Sigma_Z)$  and  $\mathbf{Z} = \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix}$ . What will  $\mathbf{X}$  be like if  $\mathbf{Y}$  is observed to be  $\mathbf{y}$ ?

Basically, we want to find  $p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}$ . From previous results, we have  $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_Y, \Sigma_{YY})$ .

Therefore,

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &\propto \exp\left(-\frac{1}{2}\left[\begin{pmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{pmatrix}^T \Sigma^{-1} \begin{pmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{pmatrix} - \tilde{\mathbf{y}}^T \Sigma_{\mathbf{Y}\mathbf{Y}}^{-1} \tilde{\mathbf{y}}\right]\right) \\ &\propto \exp\left(-\frac{1}{2}\left[\tilde{\mathbf{x}}^T \Lambda_{\mathbf{X}\mathbf{X}} \tilde{\mathbf{x}} + \tilde{\mathbf{x}}^T \Lambda_{\mathbf{X}\mathbf{Y}} \tilde{\mathbf{y}} + \tilde{\mathbf{y}}^T \Lambda_{\mathbf{Y}\mathbf{X}} \tilde{\mathbf{x}}\right]\right) \end{aligned}$$

where we use  $\tilde{\mathbf{x}}$  and  $\tilde{\mathbf{y}}$  as shorthands for  $\mathbf{x} - \boldsymbol{\mu}_{\mathbf{X}}$  and  $\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}}$ , respectively.

Completing the square for  $\tilde{\mathbf{x}}$ , we have

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &\propto \exp\left(-\frac{1}{2}\left(\tilde{\mathbf{x}} + \Lambda_{\mathbf{X}\mathbf{X}}^{-1} \Lambda_{\mathbf{X}\mathbf{Y}} \tilde{\mathbf{y}}\right)^T \Lambda_{\mathbf{X}\mathbf{X}} \left(\tilde{\mathbf{x}} + \Lambda_{\mathbf{X}\mathbf{X}}^{-1} \Lambda_{\mathbf{X}\mathbf{Y}} \tilde{\mathbf{y}}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{X}} + \Lambda_{\mathbf{X}\mathbf{X}}^{-1} \Lambda_{\mathbf{X}\mathbf{Y}} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}})\right)^T \Lambda_{\mathbf{X}\mathbf{X}} \left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{X}} + \Lambda_{\mathbf{X}\mathbf{X}}^{-1} \Lambda_{\mathbf{X}\mathbf{Y}} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}})\right)\right) \end{aligned}$$

Therefore,  $\mathbf{X}|\mathbf{y}$  is Gaussian distributed with mean  $\boldsymbol{\mu}_{\mathbf{X}} - \Lambda_{\mathbf{X}\mathbf{X}}^{-1} \Lambda_{\mathbf{X}\mathbf{Y}} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}})$  and covariance  $\Lambda_{\mathbf{X}\mathbf{X}}^{-1}$ .

Note that since  $\Lambda_{\mathbf{X}\mathbf{X}} \Sigma_{\mathbf{X}\mathbf{Y}} + \Lambda_{\mathbf{X}\mathbf{Y}} \Sigma_{\mathbf{Y}\mathbf{Y}} = 0 \Rightarrow \Lambda_{\mathbf{X}\mathbf{X}}^{-1} \Lambda_{\mathbf{X}\mathbf{Y}} = -\Sigma_{\mathbf{X}\mathbf{Y}} \Sigma_{\mathbf{Y}\mathbf{Y}}^{-1}$ , and from Fact 7.1, we have

**condition  
moment form**

$$\mathbf{X}|\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}} + \Sigma_{\mathbf{X}\mathbf{Y}} \Sigma_{\mathbf{Y}\mathbf{Y}}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}}), \Sigma_{\mathbf{X}\mathbf{X}} - \Sigma_{\mathbf{X}\mathbf{Y}} \Sigma_{\mathbf{Y}\mathbf{Y}}^{-1} \Sigma_{\mathbf{Y}\mathbf{X}}) \quad (7.38)$$

where  $\Sigma_{\mathbf{X}\mathbf{X}} - \Sigma_{\mathbf{X}\mathbf{Y}} \Sigma_{\mathbf{Y}\mathbf{Y}}^{-1} \Sigma_{\mathbf{Y}\mathbf{X}} \triangleq \Sigma|\Sigma_{\mathbf{Y}\mathbf{Y}}$  is a Schur complement.

When the observation of  $\mathbf{Y}$  is exactly the mean, the conditioned mean remains unchanged. Otherwise, it requires modification, and the size of the adjustment decreases as  $\Sigma_{\mathbf{Y}\mathbf{Y}}$ , the variance of  $\mathbf{Y}$  in the 1-D case, increases. A larger  $\Sigma_{\mathbf{Y}\mathbf{Y}}$  implies a less reliable observation. The adjustment is then scaled by  $\Sigma_{\mathbf{X}\mathbf{Y}}$ , which translates the variation in  $\mathbf{Y}$  to the variation in  $\mathbf{X}$ . Notably, if  $\mathbf{X}$  and  $\mathbf{Y}$  are negatively correlated, the sign of the adjustment will be reversed.

Regarding the variance of the conditioned variable, it always decreases. The decrease is more pronounced when  $\Sigma_{\mathbf{Y}\mathbf{Y}}$  is “smaller” and  $\Sigma_{\mathbf{X}\mathbf{Y}}$  is “larger”<sup>5</sup>. This makes sense, as a smaller  $\Sigma_{\mathbf{Y}\mathbf{Y}}$  indicates a more reliable observation, and a larger  $\Sigma_{\mathbf{X}\mathbf{Y}}$  suggests a stronger correlation between  $\mathbf{X}$  and  $\mathbf{Y}$ .

### 7.4.2 Gaussian BP Update

Let’s consider the update rules for GaBP one by one. We will use the canonical form for this analysis.

#### Variable node update

Let’s begin by examining the variable node update. Assume we have two incoming factor node messages,  $m_{a \rightarrow 1} = (\boldsymbol{\eta}_{a \rightarrow 1}, \Lambda_{a \rightarrow 1})$  and  $m_{b \rightarrow 1} = (\boldsymbol{\eta}_{b \rightarrow 1}, \Lambda_{b \rightarrow 1})$ , as depicted in Figure 7.10a. According to (7.3), the outgoing message  $m_{1 \rightarrow c} = (\boldsymbol{\eta}_{1 \rightarrow c}, \Lambda_{1 \rightarrow c})$  is simply the product of the incoming messages. Therefore, using (7.34) and (7.35), we obtain

**GaBP variable  
update**

<sup>5</sup>When we say a positive definite matrix is larger or smaller, we mean that its eigenvalues are overall larger or closer to zero.

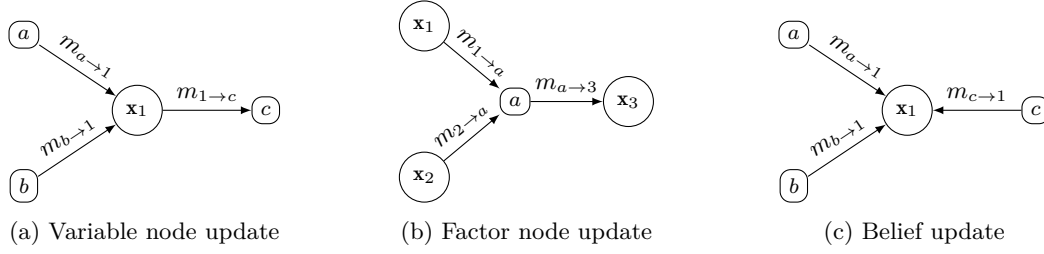


Figure 7.10: Node updates for GaBP. (a) Variable node update:  $m_{1 \rightarrow c} = (\boldsymbol{\eta}_{1 \rightarrow c}, \Lambda_{1 \rightarrow c}) = (\boldsymbol{\eta}_{a \rightarrow 1} + \boldsymbol{\eta}_{b \rightarrow 1}, \Lambda_{a \rightarrow 1} + \Lambda_{b \rightarrow 1})$  (b) Factor node update:  $m_{a \rightarrow 3} = (\boldsymbol{\eta}_{a \rightarrow 3}, \Lambda_{a \rightarrow 3}) = \left( \boldsymbol{\eta}_3 - (\Lambda_{31} \ \Lambda_{32}) \begin{pmatrix} \Lambda_{11} + \Lambda_{1 \rightarrow a} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} + \Lambda_{2 \rightarrow a} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{\eta}_1 + \boldsymbol{\eta}_{1 \rightarrow a} \\ \boldsymbol{\eta}_2 + \boldsymbol{\eta}_{2 \rightarrow a} \end{pmatrix}, \Lambda_{33} - (\Lambda_{31} \ \Lambda_{32}) \begin{pmatrix} \Lambda_{11} + \Lambda_{1 \rightarrow a} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} + \Lambda_{2 \rightarrow a} \end{pmatrix}^{-1} \begin{pmatrix} \Lambda_{13} \\ \Lambda_{23} \end{pmatrix} \right)$ . (c) Belief update:  $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\eta}_1, \Lambda_1) = \mathcal{N}(\boldsymbol{\eta}_{a \rightarrow 1} + \boldsymbol{\eta}_{b \rightarrow 1}, \Lambda_{a \rightarrow 1} + \Lambda_{b \rightarrow 1})$ .

$$\boldsymbol{\eta}_{1 \rightarrow c} = \boldsymbol{\eta}_{a \rightarrow 1} + \boldsymbol{\eta}_{b \rightarrow 1} \quad (7.39)$$

$$\Lambda_{1 \rightarrow c} = \Lambda_{a \rightarrow 1} + \Lambda_{b \rightarrow 1} \quad (7.40)$$

### Factor node update

Assume we have two incoming messages,  $m_{1 \rightarrow a} = (\boldsymbol{\eta}_{1 \rightarrow a}, \Lambda_{1 \rightarrow a})$  and  $m_{2 \rightarrow a} = (\boldsymbol{\eta}_{2 \rightarrow a}, \Lambda_{2 \rightarrow a})$ , as shown in Figure 7.10b. Let's model the factor function  $f_a(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$  with a multivariate Gaussian PDF  $\mathcal{N}(\boldsymbol{\eta}, \Lambda) = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \\ \boldsymbol{\eta}_3 \end{pmatrix}, \begin{pmatrix} \Lambda_{11} & \Lambda_{12} & \Lambda_{13} \\ \Lambda_{21} & \Lambda_{22} & \Lambda_{23} \\ \Lambda_{31} & \Lambda_{32} & \Lambda_{33} \end{pmatrix}\right)$ .

The factor node update, as described in (7.4), can be interpreted in two steps: the product of messages and the factor function, followed by marginalization. First, the product of messages and the factor function results in

$$\boldsymbol{\eta} = \begin{pmatrix} \boldsymbol{\eta}_1 + \boldsymbol{\eta}_{1 \rightarrow a} \\ \boldsymbol{\eta}_2 + \boldsymbol{\eta}_{2 \rightarrow a} \\ \boldsymbol{\eta}_3 \end{pmatrix}, \quad (7.41)$$

$$\Lambda = \begin{pmatrix} \Lambda_{11} + \Lambda_{1 \rightarrow a} & \Lambda_{12} & \Lambda_{13} \\ \Lambda_{21} & \Lambda_{22} + \Lambda_{2 \rightarrow a} & \Lambda_{23} \\ \Lambda_{31} & \Lambda_{32} & \Lambda_{33} \end{pmatrix}. \quad (7.42)$$

Next, marginalizing  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , from (7.33) and (7.31), we have

$$\boldsymbol{\eta}_{a \rightarrow 3} = \boldsymbol{\eta}_3 - (\Lambda_{31} \ \Lambda_{32}) \begin{pmatrix} \Lambda_{11} + \Lambda_{1 \rightarrow a} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} + \Lambda_{2 \rightarrow a} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{\eta}_1 + \boldsymbol{\eta}_{1 \rightarrow a} \\ \boldsymbol{\eta}_2 + \boldsymbol{\eta}_{2 \rightarrow a} \end{pmatrix} \quad (7.43)$$

**GaBP factor update**



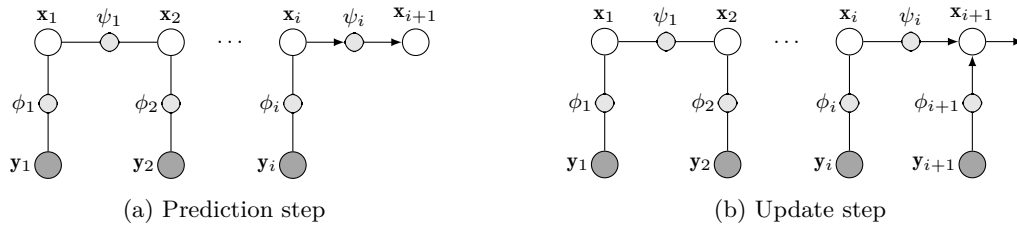


Figure 7.11: Kalman filter as interpreted by BP algorithm.

and

$$\Lambda_{a \rightarrow 3} = \Lambda_{33} - \begin{pmatrix} \Lambda_{31} & \Lambda_{32} \end{pmatrix} \begin{pmatrix} \Lambda_{11} + \Lambda_{1 \rightarrow a} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} + \Lambda_{2 \rightarrow a} \end{pmatrix}^{-1} \begin{pmatrix} \Lambda_{13} \\ \Lambda_{23} \end{pmatrix}. \quad (7.44)$$

The above update rule can be easily generalized in the same manner when the number of incoming messages is greater than two.

### Belief update

Finally, the belief update at a node is simply the product of all incoming messages, as shown in Figure 7.10c. The update rule, as given in (7.5), becomes

$$\boldsymbol{\eta}_1 = \boldsymbol{\eta}_{a \rightarrow 1} + \boldsymbol{\eta}_{b \rightarrow 1} + \boldsymbol{\eta}_{c \rightarrow 1}, \quad (7.45)$$

$$\Lambda_1 = \Lambda_{a \rightarrow 1} + \Lambda_{b \rightarrow 1} + \Lambda_{c \rightarrow 1}. \quad (7.46)$$

### 7.4.3 Kalman filter and BP

Let's consider a simple Kalman filter setup with an unobserved state  $\mathbf{x}_i$  and an observed  $\mathbf{y}_i$  at time  $i$ . For simplicity, we assume no control vector is applied to the system. The system is governed by the following set of equations

$$\mathbf{x}_{i+1} = A\mathbf{x}_i + \mathbf{z}_i, \quad (7.47)$$

$$\mathbf{y}_{i+1} = C\mathbf{x}_{i+1} + \mathbf{w}_i, \quad (7.48)$$

where  $\mathbf{z}_i \sim \mathcal{N}(0, Q)$  and  $\mathbf{w}_i \sim \mathcal{N}(0, R)$ .

A factor graph can be constructed as shown in Figure 7.11. The joint probability of the variables is proportional to the product of the factor functions, as given by

$$p(\mathbf{x}^N, \mathbf{y}^N) = \frac{1}{Z} \prod_{i=1}^{N-1} \psi(\mathbf{x}_i, \mathbf{x}_{i+1}) \prod_{i=1}^N \phi(\mathbf{x}_i, \mathbf{y}_i). \quad (7.49)$$

There is freedom in choosing the factor functions as long as (7.49) is satisfied. A reasonable and

convenient choice would be  $Z = 1$ ,

**Kalman factor  
functions**

$$\psi(\mathbf{x}_i, \mathbf{x}_{i+1}) = \begin{cases} p(\mathbf{x}_1, \mathbf{x}_2), & \text{if } i = 1, \\ p(\mathbf{x}_{i+1}|\mathbf{x}_i), & \text{otherwise.} \end{cases} \quad (7.50)$$

and

$$\phi(\mathbf{x}_i, \mathbf{y}_i) = p(\mathbf{y}_i|\mathbf{x}_i). \quad (7.51)$$

The objective of the Kalman filter is to estimate the state  $\mathbf{x}_i$  at each time step  $i$ , given the observations. The estimation process can be divided into two steps: the prediction step, which estimates  $\mathbf{x}_{i+1}$  based on the previous state  $\mathbf{x}_i$  and observations, and the update step, which adjusts the estimate of  $\mathbf{x}_{i+1}$  after observing  $\mathbf{y}_{i+1}$ . Following the common convention in Kalman filtering, we will use  $\boldsymbol{\mu}_{i+1|i}$  and  $\Sigma_{i+1|i}$  to denote the mean and covariance estimates, respectively, after the prediction step. Similarly, we will use  $\boldsymbol{\mu}_{i+1|i+1}$  and  $\Sigma_{i+1|i+1}$  to denote the respective estimates after the update step.

### Prediction step

As shown in Figure 7.11a, the prediction step of  $\mathbf{x}_{i+1}$  from  $\mathbf{x}_i$  can be interpreted as the factor node update step of  $\psi_i$ . The message from  $\mathbf{x}_i$  to  $\psi_i$ ,  $m_{i \rightarrow \psi_i}$ , essentially captures the probability of  $\mathbf{x}_i$  given earlier observations  $\mathbf{y}^i$ . With a slight change of notation from the earlier GaBP discussion, we will take  $m_{i \rightarrow \psi_i}$  as  $p(\mathbf{x}_i|\mathbf{y}^i)$  rather than the parameters. By the original factor node update rule in (7.4), we have<sup>6</sup>

$$m_{\psi_i \rightarrow i+1}(\mathbf{x}_{i+1}) = \int_{\mathbf{x}_i} m_{i \rightarrow \psi_i}(\mathbf{x}_i) \psi_i(\mathbf{x}_i, \mathbf{x}_{i+1}) d\mathbf{x}_i \quad (7.52)$$

$$= \int_{\mathbf{x}_i} m_{i \rightarrow \psi_i}(\mathbf{x}_i) p(\mathbf{x}_{i+1}|\mathbf{x}_i) d\mathbf{x}_i \quad (7.53)$$

$$= \int_{\mathbf{x}_i} p(\mathbf{x}_{i+1}, \mathbf{x}_i|\mathbf{y}^i) d\mathbf{x}_i \quad (7.54)$$

We can model  $p(\mathbf{x}_{i+1}, \mathbf{x}_i|\mathbf{y}^i)$  as a joint Gaussian distribution, since all linear manipulations involved in the setup keep the distribution Gaussian. Here, we will use the moment form rather than the canonical form as parameters to match the traditional Kalman filter discussion. To find the precise model, we just need to compute the mean and covariance of  $(\mathbf{X}_i, \mathbf{X}_{i+1})$ . And without affecting our discussion, we will drop the observation  $\mathbf{y}^i$  in the following.

Assume that  $E[\mathbf{X}_i] = \boldsymbol{\mu}_{i|i}$  and  $cov[\mathbf{X}_i] = \Sigma_{i|i}$ , then

$$E[\mathbf{X}_{i+1}] = AE[\mathbf{X}_i] + E[Z_i] = A\boldsymbol{\mu}_{i|i} \quad (7.55)$$

---

<sup>6</sup>We use integration rather than summation here because we are dealing with continuous variables.

and

$$E[(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})(\mathbf{X}_{i+1} - \boldsymbol{\mu}_{i+1|i})^\top] = E[(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})(A\mathbf{X}_i + Z_i - A\boldsymbol{\mu}_i)^\top] \quad (7.56)$$

$$= E[(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})(A\mathbf{X}_i - A\boldsymbol{\mu}_i)^\top] + E[(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})Z_i^\top] \quad (7.57)$$

$$\stackrel{(a)}{=} E[(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})(A\mathbf{X}_i - A\boldsymbol{\mu}_i)^\top] + E[\mathbf{X}_i - \boldsymbol{\mu}_{i|i}]E[Z_i^\top] \quad (7.58)$$

$$\stackrel{(b)}{=} E[(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})^\top A^\top] = \Sigma_{i|i}A^\top, \quad (7.59)$$

where (a) is due to  $Z_i$  independent of  $\mathbf{X}_i$  and (b) is because  $Z_i$  is zero-mean. Moreover, we have

$$E[(\mathbf{X}_{i+1} - \boldsymbol{\mu}_{i+1|i})(\mathbf{X}_{i+1} - \boldsymbol{\mu}_{i+1|i})^\top] \quad (7.60)$$

$$= E[(A\mathbf{X}_i + Z_i - A\boldsymbol{\mu}_{i|i})(A\mathbf{X}_i + Z_i - A\boldsymbol{\mu}_i)^\top] \quad (7.61)$$

$$= E[(A\mathbf{X}_i - A\boldsymbol{\mu}_{i|i})(A\mathbf{X}_i - A\boldsymbol{\mu}_i)^\top] + E[Z_i(A\mathbf{X}_i - A\boldsymbol{\mu}_{i|i})^\top] \\ + E[(A\mathbf{X}_i - A\boldsymbol{\mu}_{i|i})Z_i^\top] + E[Z_iZ_i^\top] \quad (7.62)$$

$$\stackrel{(a)}{=} E[A(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})(\mathbf{X}_i - \boldsymbol{\mu}_{i|i})^\top A^\top] + E[Z_iZ_i^\top] = A\Sigma_{i|i}A^\top + Q, \quad (7.63)$$

where (a) is due to  $Z_i$  independent of  $\mathbf{X}_i$  and zero-mean.

Therefore we have

$$p(\mathbf{x}_i, \mathbf{x}_{i+1} | \mathbf{y}^i) = \mathcal{N} \left( \begin{pmatrix} \mathbf{x}_i \\ \mathbf{x}_{i+1} \end{pmatrix}; \begin{pmatrix} \boldsymbol{\mu}_i \\ A\boldsymbol{\mu}_i \end{pmatrix}, \begin{pmatrix} \Sigma_{i|i} & \Sigma_{i|i}A^\top \\ A\Sigma_{i|i} & A\Sigma_{i|i}A^\top + Q \end{pmatrix} \right) \quad (7.64)$$

### Kalman prediction step

Marginalizing  $\mathbf{x}_i$ , we get

$$\Sigma_{i+1|i} = A\Sigma_{i|i}A^\top + Q, \quad (7.65)$$

$$\boldsymbol{\mu}_{i+1|i} = A\boldsymbol{\mu}_{i|i}. \quad (7.66)$$

### Update step

The update step is essentially the variable node update of  $\mathbf{x}_{i+1}$ . However, instead of directly using the variable node update rule, we will derive the update from first principles. Note that

$$m_{\psi_i \rightarrow i+1}(\mathbf{x}_{i+1}) \cdot \phi_{i+1}(\mathbf{x}_{i+1}, \mathbf{y}_{i+1}) = p(\mathbf{x}_{i+1} | \mathbf{y}^i) p(\mathbf{y}_{i+1} | \mathbf{x}_{i+1}) \quad (7.67)$$

$$= p(\mathbf{x}_{i+1}, \mathbf{y}_{i+1} | \mathbf{y}^i), \quad (7.68)$$

which can be modeled by a joint Gaussian as before. Assume that  $\mathbf{X}_{i+1}$  has a mean  $\boldsymbol{\mu}_{i+1|i}$  and covariance  $\Sigma_{i+1|i}$ , from (7.48) and using the same argument as in the prediction step, we can

immediately model  $\begin{pmatrix} \mathbf{X}_{i+1} \\ \mathbf{Y}_{i+1} \end{pmatrix}$  by

$$\mathcal{N} \left( \begin{pmatrix} \boldsymbol{\mu}_{i+1|i} \\ C\boldsymbol{\mu}_{i+1|i} \end{pmatrix}, \begin{pmatrix} \Sigma_{i+1|i} & \Sigma_{i+1|i}C^\top \\ C\Sigma_{i+1|i} & C\Sigma_{i+1|i}C^\top + R \end{pmatrix} \right) \quad (7.69)$$

Conditioning on  $\mathbf{Y}_{i+1} = \mathbf{y}_{i+1}$ , we have

$$\mathbf{X}_{i+1} | \mathbf{y}_{i+1} \sim \mathcal{N}(\boldsymbol{\mu}_{i+1|i+1}, \Sigma_{i+1|i+1}),$$

where

**Kalman update  
step**

$$\boldsymbol{\mu}_{i+1|i+1} = \boldsymbol{\mu}_{i+1|i} + \underbrace{\Sigma_{i+1|i} C^\top (C \Sigma_{i+1|i} C^\top + R)^{-1}}_G (\mathbf{y} - C \boldsymbol{\mu}_{i+1|i}) \quad (7.70)$$

$$= \boldsymbol{\mu}_{i+1|i} + G(\mathbf{y} - C \boldsymbol{\mu}_{i+1|i}) \quad (7.71)$$

$$\Sigma_{i+1|i+1} = \Sigma_{i+1|i} - \underbrace{\Sigma_{i+1|i} C^\top (C \Sigma_{i+1|i} C^\top + R)^{-1}}_G C \Sigma_{i+1|i} \quad (7.72)$$

$$= \Sigma_{i+1|i} - G C \Sigma_{i+1|i} \quad (7.73)$$

and  $G$  is often referred to as the Kalman gain.

**Kalman gain**

## Exercise

1. Show that marginalizing  $\mathbf{X}_2$  from joint Gaussian distributed  $\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix}$  will still result in a Gaussian distributed  $\mathbf{X}_1$ .
2. Try to verify (7.17) and (7.18) in the derivation of BP using Bethe approximation.
3. Let  $\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix}$  has its PDF in the canonical form  $\mathcal{N}\left(\begin{pmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{pmatrix}, \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix}\right)$ , what will be  $\boldsymbol{\eta}$  and  $\Lambda$  if  $\mathbf{X}_1 | \mathbf{x}_2 \sim \mathcal{N}(\boldsymbol{\eta}, \Lambda)$ ?
4. Write down the factor node update message  $m_{a \rightarrow 1}$  for GaBP for a factor node  $a$  connecting to 4 variable nodes  $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ . Write down the message again assuming that  $a$  is only connecting to two nodes  $\mathbf{x}_1$  and  $\mathbf{x}_2$ .

# Chapter 8

## Score, Fisher information, Cramér-Rao lower bound, and score matching

In this chapter, we will explore the concepts of the score, Fisher information, and the renowned Cramér-Rao lower bound. Roughly speaking, Fisher information quantifies the amount of information that an observable random variable carries about an unknown parameter. Thus, the higher the Fisher information, the more accurately we can estimate the parameter from the variable. The theoretical limit of this estimation accuracy is concretely described by the Cramér-Rao lower bound. To conclude the first section, we will also discuss a very useful estimation technique known as score matching. Using a slightly different definition of the score from classical statistics, score matching avoids the effect of the normalization constant (or partition function) present in maximum likelihood estimation (MLE). Moreover, a surprising result is that we do not need the actual score in the objective function; rather, the objective function can be represented by the parameterized model score only. These advantages make score matching an attractive alternative to MLE, and it has gained significant popularity in the last couple of decades.

### 8.1 Overview of Fisher information and Cramér-Rao lower bound

Given an observation  $X$  drawn from a distribution defined by the parameter  $\theta$ , a fundamental problem in statistical inference is to estimate  $\theta$  based on  $X$ . As discussed in Section 2.6, this can be addressed by the method of MLE, which seeks to optimize the parameter  $\theta$  such that the likelihood  $p(X = x; \theta)$ , or equivalently the log-likelihood  $\log p(X = x; \theta)$ , is maximized for the observed data  $X = x$ .

Upon obtaining an estimate of  $\theta$ , a natural follow-up question arises: how reliable, or accurate, is this estimate on average? This is crucial for understanding the quality and robustness of the estimation process. The Cramér-Rao lower bound offers insight into this question. It establishes a lower limit on the variance of any unbiased estimator of the parameter  $\theta$ , which is directly related to the so-called Fisher Information. This lower bound provides a benchmark to evaluate

the performance of an estimator obtained through MLE or any other estimation procedure.

## 8.2 The score function

To facilitate the understanding of Fisher information, we first introduce the score function. The **score function** reflects the signed sensitivity of the log-likelihood to changes in the parameter  $\theta$ . Consider a family of probability density functions  $p(x; \theta)$ , parameterized by the scalar  $\theta$ . The score function is defined as

$$V(X; \theta) \triangleq \frac{\partial}{\partial \theta} \ln p(X; \theta) = \frac{1}{p(X; \theta)} \frac{\partial p(X; \theta)}{\partial \theta}. \quad (8.1)$$

Note that the score function itself is a random variable since  $X$  is random.

An important property of the score function is that its expected value is zero regardless of the value of  $\theta$ . This fact is so important that we will highlight it again in the following.

### Expected score is zero

For any random observation  $X$  and respective parameter  $\theta$ ,  $E[V(X; \theta)] = 0$  regardless of the value of  $\theta$ .

expected score is zero

The fact can be easily shown with a little bit calculus as follows. We can

$$\begin{aligned} E[V(X; \theta)] &= \int \left( \frac{\partial p(x; \theta)}{\partial \theta} \cdot \frac{1}{p(x; \theta)} \right) p(x; \theta) dx \\ &\stackrel{(a)}{=} \frac{\partial}{\partial \theta} \int p(x; \theta) dx \\ &= \frac{\partial}{\partial \theta} 1 = 0, \end{aligned} \quad (8.2)$$

where we can pull the partial derivative out of the integral in (a) since the integral is with respect to  $x$  rather than  $\theta$ .

## 8.3 Fisher Information

**Fisher information** provides a measure of the amount of information that an observable random variable carries about an unknown parameter upon which the probability of the observation depends.

It is intuitive that  $\theta$  is more sensitive with  $X$  when the respective score  $V(X; \theta)$  has a larger magnitude. This suggests defining the average power of  $V(X; \theta)$  as a measure to quantify the potential of estimating  $\theta$  from the observation  $X$ . We call this quantity the Fisher information, usually denoted by  $J_X(\theta)$  or simply  $J(\theta)$ . We use the subscript  $X$  to emphasize that the Fisher information is computed with respect to  $X$ . Since  $V(X; \theta)$  is zero-mean, we have

$$J_X(\theta) \triangleq E[V(X; \theta)^2] = \text{Var}(V(X; \theta)), \quad (8.3)$$

where both expectation and variance are applied with respect to  $X$ .

**score of multiple observations** In the context of multiple independent and identically distributed (i.i.d.) observations  $X_1, \dots, X_n$ , note that the score with respect to (w.r.t.) the  $n$  observations is equal to the sum of the scores w.r.t. individual observations since

$$V(X_1, \dots, X_n; \theta) = \frac{\partial}{\partial \theta} \ln p(X_1, \dots, X_n; \theta) \quad (8.4)$$

$$= \frac{\partial}{\partial \theta} \sum_{i=1}^n \ln p(X_i; \theta) \quad (8.5)$$

$$= \sum_{i=1}^n V(X_i; \theta). \quad (8.6)$$

**Fisher information of multiple observations**

Moreover,

$$J_{X_1, \dots, X_n}(\theta) = E [V(X_1, \dots, X_n)^2] \quad (8.7)$$

$$= E \left[ \left( \sum_{i=1}^n V(X_i) \right)^2 \right] \quad (8.8)$$

$$= E \left[ \sum_{i=1}^n V(X_i)^2 \right] + 2E \left[ \sum_{i \neq j} V(X_i) V(X_j) \right] \quad (8.9)$$

$$= E \left[ \sum_{i=1}^n V(X_i)^2 \right] + 2 \sum_{i \neq j} E[V(X_i)] E[V(X_j)] \quad (8.10)$$

$$= \sum_{i=1}^n J_{X_i}(\theta). \quad (8.11)$$

Thus, the total Fisher information from multiple observations can be expressed as the sum of the Fisher information from individual observations. This additive property is natural and desirable, as it allows us to interpret the Fisher information as a measure of information, similar to entropy.

## 8.4 Cramér-Rao Lower Bound

If we restrict ourselves to unbiased estimators  $T(\cdot)$  such that  $E[T(X)] = \theta$ , the performance of an estimator can be completely characterized by the variance of the estimate. Namely, the larger the variance, the poorer the estimator.

The Fisher information  $J_X(\theta)$ , which indicates the average sensitivity of the parameter  $\theta$  to the observation  $X$ , implies that an increase in  $J_X(\theta)$  should lead to a decrease in the variance or uncertainty of the estimate. This relationship is succinctly articulated by the **Cramér-Rao lower bound**, which elegantly captures this phenomenon.

**Cramér-Rao lower bound**

**Cramér-Rao Lower Bound**

If  $E[T(X)] = \theta$ , then the variance of  $T(X)$  satisfies:

$$\text{Var}(T(X)) \geq \frac{1}{J_X(\theta)}. \quad (8.12)$$

*Proof of Cramér-Rao Lower Bound.* The proof is straightforward using the Cauchy-Schwarz inequality (see Exercise 1) for random variables, i.e.  $E[XY]^2 \leq E[X^2]E[Y^2]$  for all random variables  $X$  and  $Y$ . To avoid notation clutter, I will not write  $X$  explicitly as an argument of  $T$  and  $V$ . It should be clear by now that  $T$  and  $V$  depend on  $X$  and thus are themselves random variables. By the Cauchy-Schwarz inequality, we have

$$E^2[(T - E[T])(V - E[V])] \leq E[(T - E[T])^2]E[(V - E[V])^2] \quad (8.13)$$

$$= \text{Var}(T)\text{Var}(V) = \text{Var}(T)J(\theta). \quad (8.14)$$

Therefore,

$$\text{Var}(T) \geq \frac{E[(T - E[T])(V - E[V])]}{J(\theta)} \stackrel{(a)}{=} \frac{E[(T - E[T])V]}{J(\theta)} \quad (8.15)$$

$$= \frac{E[TV] - E[T]E[V]}{J(\theta)} \stackrel{(b)}{=} \frac{E[TV]}{J(\theta)} \quad (8.16)$$

$$= \frac{1}{J(\theta)} \int T(x) \frac{\partial p(x; \theta) / \partial \theta}{p(x; \theta)} p(x; \theta) dx \quad (8.17)$$

$$= \frac{1}{J(\theta)} \frac{\partial}{\partial \theta} \int T(x) p(x; \theta) dx \quad (8.18)$$

$$= \frac{1}{J(\theta)} \frac{\partial}{\partial \theta} E[T] \stackrel{(c)}{=} \frac{1}{J(\theta)} \frac{\partial}{\partial \theta} \theta = \frac{1}{J(\theta)}, \quad (8.19)$$

where (a) and (b) follow from  $E[V] = 0$ , and (c) is due to the fact that  $T$  is an unbiased estimator.  $\square$

**Example 8.1: Estimating the Mean of a Normal Distribution**

To illustrate the Cramér-Rao bound, consider the problem of estimating the mean  $\mu$  of a normally distributed population with known variance  $\sigma^2$ . Given  $M$  samples  $X_1, X_2, \dots, X_M$ , a natural estimator for  $\mu$  is the sample mean:

$$\hat{\mu}(X_1, \dots, X_M) = \frac{1}{M} \sum_{i=1}^M X_i. \quad (8.20)$$

This estimator is unbiased, as  $E[\hat{\mu}(X_1, \dots, X_M)] = \frac{1}{M} \sum_{i=1}^M E[X_i] = \mu$ , and its variance is given by  $\text{Var}(\hat{\mu}) = \frac{1}{M} \sum_{i=1}^M \text{Var}(X_i) = \frac{\sigma^2}{M}$ .

To demonstrate that this estimator is theoretically optimal, we will show that the variance



of the estimate is precisely the Cramér-Rao bound.

First, note that (see Exercise 2) the Fisher information for a single observation from the normal distribution is  $J_X(\mu) = \frac{1}{\sigma^2}$ . Hence, for  $M$  observations, the Fisher information is  $J_{X_1, \dots, X_M}(\mu) = \frac{M}{\sigma^2}$ . According to the Cramér-Rao lower bound, no unbiased estimator of  $\mu$  can have a variance smaller than  $J_{X_1, \dots, X_M}(\mu)^{-1} = \frac{\sigma^2}{M}$ . As the variance of the estimator is precisely  $\frac{\sigma^2}{M}$ , the sample mean estimator is indeed optimal.

## 8.5 Score matching

Before ending this chapter, it is worth mentioning score matching, a technique gaining popularity in recent years as an alternative to MLE for parameter estimation in probability density models, particularly when dealing with complex or high-dimensional data. Unlike MLE, which requires the computation of a normalizing constant that can be intractable for many models, score matching bypasses this issue by focusing on the score function. By matching the model's score function to that of the true data distribution, it becomes especially useful for models with unnormalized densities. As a result, score matching has gained popularity in fields such as machine learning and statistics, where it facilitates the training of sophisticated models like energy-based models and offers a robust solution in scenarios where traditional likelihood-based methods falter.

### 8.5.1 When MLE fails: Energy-Based Models (EBMs)

**energy-based  
model**

An EBM is a type of probabilistic model where the probability density function is defined in terms of an energy function  $E(\mathbf{x}; \theta)$ . The density is given by:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}; \theta))}{Z(\theta)}$$

where  $Z(\theta) = \int \exp(-E(\mathbf{x}; \theta)) d\mathbf{x}$  is the normalizing constant, also known as the partition function.

In MLE, we maximize the likelihood or equivalently the log-likelihood of the observed data, which for an EBM involves maximizing:

$$\mathcal{L}(\theta) = \sum_{i=1}^M \log p_\theta(\mathbf{x}_i) = \sum_{i=1}^M (-E(\mathbf{x}_i; \theta) - \log Z(\theta))$$

The computation of the partition function  $Z(\theta)$  is often intractable because it requires integrating over the entire observation space, which is computationally prohibitive for high-dimensional or complex data.

### 8.5.2 Score Matching for EBM

Score matching, on the other hand, does not require the computation of the normalizing constant. Instead, it focuses on matching the scores, defined as the gradients of the log-density. Mathemati-

cally, the score function for the EBM is defined as

$$\psi(\mathbf{x}; \theta) = \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x}) = -\nabla_{\mathbf{x}} E(\mathbf{x}; \theta)$$

**! score  
function for  
EBM**

One thing a cautious reader may have noticed is that the score definition here is slightly different from that used in traditional statistics, where the partial derivative is taken with respect to the parameter  $\theta$ . Instead, the score here is computed as the gradient with respect to the data  $\mathbf{x}$  directly.

The naive objective function attempts to minimize the expected square difference between the observed score and the true score. That is,

$$J(\theta) = \frac{1}{2} \int \|\psi(\mathbf{x}; \theta) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} \quad (8.21)$$

$$\approx \frac{1}{M} \sum_{m=1}^M \|\psi(\mathbf{x}^{(m)}; \theta) - \nabla_{\mathbf{x}} \log p(\mathbf{x}^{(m)})\|^2, \quad (8.22)$$

where  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(M)}$  are  $M$  observed samples. The above naive objective is not practical as it is impossible to know the true score function  $\nabla_{\mathbf{x}} \log p(\cdot)$  to begin with.

Surprisingly, it can be shown that the earlier objective is equivalent to minimizing instead the following objective

$$J(\theta) = \frac{1}{2} \int \left[ \|\psi(\mathbf{x}; \theta)\|^2 + 2\nabla_{\mathbf{x}} \cdot \psi(\mathbf{x}; \theta) \right] p(\mathbf{x}) d\mathbf{x} \quad (8.23)$$

$$\approx \frac{1}{2M} \sum_{m=1}^M \left[ \|\psi(\mathbf{x}_m; \theta)\|^2 + 2\nabla_{\mathbf{x}} \cdot \psi(\mathbf{x}_m; \theta) \right], \quad (8.24)$$

where  $\nabla_{\mathbf{x}} \cdot \psi(\mathbf{x}; \theta)$  denotes the divergence of  $\psi$  w.r.t.  $\mathbf{x}$ , i.e.,  $\nabla_{\mathbf{x}} \cdot \psi(\mathbf{x}; \theta) \triangleq \sum_{n=1}^N \frac{\partial \psi_n(\mathbf{x})}{\partial x_n}$ . The above objective only involves the score function and its derivatives, which can be computed directly from the energy function without knowing the actual score or the need for  $Z(\theta)$ .

### Derivation of the Score Matching Objective Function

We will show here that the objective function defined in (8.23) is equivalent to the original objective function in (8.21). Let's start with the definition of the score matching objective function:

$$J(\theta) = \frac{1}{2} \int \|\psi(\mathbf{x}; \theta) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x},$$

where recall that  $\psi(\mathbf{x}; \theta) = \nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})$  is the score function of the model.

Expanding the squared term inside the integral:

$$\begin{aligned} J(\theta) &= \frac{1}{2} \int \|\psi(\mathbf{x}; \theta)\|^2 p(\mathbf{x}) d\mathbf{x} - \int (\psi(\mathbf{x}; \theta) \cdot \nabla_{\mathbf{x}} \log p(\mathbf{x})) p(\mathbf{x}) d\mathbf{x} \\ &\quad + \frac{1}{2} \int \|\nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x}. \end{aligned} \quad (8.25)$$

The last term does not depend on  $\theta$  and can be ignored for optimization purposes.

For the cross-term,

$$\int (\psi(\mathbf{x}; \theta) \cdot \nabla_{\mathbf{x}} \log p(\mathbf{x})) p(\mathbf{x}) d\mathbf{x} = \int \left( \psi(\mathbf{x}; \theta) \cdot \frac{\nabla_{\mathbf{x}} p(\mathbf{x})}{p(\mathbf{x})} \right) p(\mathbf{x}) d\mathbf{x} \quad (8.26)$$

$$= \int \psi(\mathbf{x}; \theta) \cdot \nabla_{\mathbf{x}} p(\mathbf{x}) d\mathbf{x} \quad (8.27)$$

$$= \int \left( \sum_{n=1}^N \psi_n(\mathbf{x}; \theta) \frac{\partial p(\mathbf{x})}{\partial x_n} \right) d\mathbf{x}, \quad (8.28)$$

where  $N$  is the dimension of the data  $\mathbf{x}$ , and  $\psi_n(\mathbf{x}; \theta)$  and  $x_n$  are the  $n$ -th components of  $\psi(\mathbf{x}; \theta)$  and  $\mathbf{x}$ , respectively.

We can pull out the dimension for  $x_n$  from the multidimensional integral by splitting  $d\mathbf{x}$  into  $dx_n d\mathbf{x}^{\setminus n}$ , where  $d\mathbf{x}^{\setminus n}$  denotes  $dx_1 dx_2 \cdots dx_{n-1} dx_{n+1} \cdots$ . Swapping the order of the split integral with summation, we have:

$$\int \left( \sum_{n=1}^N \psi_n(\mathbf{x}; \theta) \frac{\partial p(\mathbf{x})}{\partial x_n} \right) d\mathbf{x} \quad (8.29)$$

$$= \int d\mathbf{x}^{\setminus n} \sum_{n=1}^N \int \left( \psi_n(\mathbf{x}; \theta) \frac{\partial p(\mathbf{x})}{\partial x_n} \right) dx_n \quad (8.30)$$

$$\stackrel{(a)}{=} \int d\mathbf{x}^{\setminus n} \sum_{n=1}^N \int \left( \frac{\partial (\psi_n(\mathbf{x}; \theta) p(\mathbf{x}))}{\partial x_n} - \frac{\partial \psi_n(\mathbf{x}; \theta)}{\partial x_n} p(\mathbf{x}) \right) dx_n \quad (8.31)$$

$$\stackrel{(b)}{=} \int d\mathbf{x}^{\setminus n} \sum_{n=1}^N \left[ \psi_n(\mathbf{x}; \theta) p(\mathbf{x}) \Big|_{x_n=-\infty}^{\infty} - \int \left( \frac{\partial \psi_n(\mathbf{x}; \theta)}{\partial x_n} p(\mathbf{x}) \right) dx_n \right] \quad (8.32)$$

$$= - \int p(\mathbf{x}) d\mathbf{x} \sum_{n=1}^N \left( \frac{\partial \psi_n(\mathbf{x}; \theta)}{\partial x_n} \right) \quad (8.33)$$

$$= - \int \nabla \cdot \psi(\mathbf{x}; \theta) p(\mathbf{x}) d\mathbf{x}, \quad (8.34)$$

where (a) uses integration by parts and the first term in (b) goes to zero because  $p(\mathbf{x})$  goes to zero as  $x_n$  goes to infinity.

Combining the result in (8.34) with the first term of (8.25), we get the revised objective function in (8.23).

### 8.5.3 Example: Score Matching for Multivariate Gaussian Distribution

Let's try to apply score matching to a multivariate Gaussian distribution with mean vector  $\boldsymbol{\mu}$  and precision matrix  $\Lambda$  (the inverse of the covariance matrix  $\boldsymbol{\Sigma}$ ). The probability density function is given by:

$$p(\mathbf{x}; \boldsymbol{\mu}, \Lambda) = \frac{1}{Z(\boldsymbol{\mu}, \Lambda)} \exp \left( -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Lambda (\mathbf{x} - \boldsymbol{\mu}) \right)$$

where  $\mathbf{x} \in \mathbb{R}^d$ ,  $\boldsymbol{\mu} \in \mathbb{R}^d$ , and  $\Lambda \in \mathbb{R}^{d \times d}$  is a symmetric positive-definite matrix. Here,  $Z(\boldsymbol{\mu}, \Lambda)$  is the normalizing constant.

First, we need to compute the score function as

$$\psi(\mathbf{x}; \boldsymbol{\mu}, \Lambda) = \nabla_{\mathbf{x}} \log p(\mathbf{x}; \boldsymbol{\mu}, \Lambda) = -\Lambda(\mathbf{x} - \boldsymbol{\mu}) \quad (8.35)$$

Now, let's substitute  $\psi(\mathbf{x}; \boldsymbol{\mu}, \Lambda)$  into the alternative score matching objective as specified in (8.23). Assuming that we have made  $M$  observations  $\mathbf{x}_1, \dots, \mathbf{x}_M$ , we can then approximate the objective as

$$J(\boldsymbol{\mu}, \Lambda) = \frac{1}{2M} \sum_{m=1}^M \left( \|\psi(\mathbf{x}_m; \boldsymbol{\mu}, \Lambda)\|^2 + 2\nabla_{\mathbf{x}} \cdot \psi(\mathbf{x}_m; \boldsymbol{\mu}, \Lambda) \right) \quad (8.36)$$

$$= \frac{1}{2M} \sum_{m=1}^M \left( \|\Lambda(\mathbf{x}_m - \boldsymbol{\mu})\|^2 - 2\nabla_{\mathbf{x}} \cdot \Lambda(\mathbf{x}_m - \boldsymbol{\mu}) \right) \quad (8.37)$$

$$\stackrel{(a)}{=} \frac{1}{2M} \sum_{m=1}^M \left( \|\Lambda(\mathbf{x}_m - \boldsymbol{\mu})\|^2 - 2 \operatorname{tr}(\Lambda) \right) \quad (8.38)$$

$$= \frac{1}{2M} \sum_{m=1}^M \left( (\mathbf{x}_m - \boldsymbol{\mu})^\top \Lambda^\top \Lambda (\mathbf{x}_m - \boldsymbol{\mu}) - 2 \operatorname{tr}(\Lambda) \right), \quad (8.39)$$

where (a) is due to

$$\nabla \cdot \Lambda \mathbf{x} = \sum_{n=1}^N \frac{\partial (\Lambda \mathbf{x})_n}{\partial x_n} = \sum_{n=1}^N \frac{\partial \left( \sum_{n'=1}^N \Lambda_{nn'} x_{n'} \right)}{\partial x_n} = \sum_{n=1}^N \sum_{n'=1}^N \Lambda_{n,n'} \delta_{n,n'} = \sum_{n=1}^N \Lambda_{n,n}.$$

Taking the gradient of the objective function in (8.39) with respect to  $\boldsymbol{\mu}$  to minimize it, we have

$$\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\mu}, \Lambda) = \frac{1}{2M} \sum_{m=1}^M \nabla_{\boldsymbol{\mu}} \left( (\mathbf{x}_m - \boldsymbol{\mu})^\top \Lambda^\top \Lambda (\mathbf{x}_m - \boldsymbol{\mu}) - 2 \operatorname{tr}(\Lambda) \right) \quad (8.40)$$

$$= \frac{1}{M} \sum_{m=1}^M \Lambda^\top \Lambda (\mathbf{x}_m - \boldsymbol{\mu}) \quad (8.41)$$

$$= \Lambda^\top \Lambda \left( \frac{1}{M} \sum_{m=1}^M \mathbf{x}_m - \boldsymbol{\mu} \right), \quad (8.42)$$

which is equal to zero only when  $\boldsymbol{\mu}$  equals the empirical mean  $\frac{1}{M} \sum_{m=1}^M \mathbf{x}_m$ .

Similarly, taking the gradient of  $J(\boldsymbol{\mu}, \Lambda)$  with respect to  $\Lambda$ , we have

$$\nabla_{\Lambda} J(\boldsymbol{\mu}, \Lambda) = \frac{1}{2M} \sum_{m=1}^M \nabla_{\Lambda} ((\mathbf{x}_m - \boldsymbol{\mu})^{\top} \Lambda^{\top} \Lambda (\mathbf{x}_m - \boldsymbol{\mu}) - 2 \operatorname{tr}(\Lambda)) \quad (8.43)$$

$$\stackrel{(a)}{=} \frac{1}{2M} \sum_{m=1}^M (2\Lambda (\mathbf{x}_m - \boldsymbol{\mu})(\mathbf{x}_m - \boldsymbol{\mu})^{\top} - 2I) \quad (8.44)$$

$$= \Lambda \left( \frac{1}{M} \sum_{m=1}^M (\mathbf{x}_m - \boldsymbol{\mu})(\mathbf{x}_m - \boldsymbol{\mu})^{\top} \right) - I, \quad (8.45)$$

where (a) uses  $\nabla_{\Lambda} (\mathbf{y}^{\top} \Lambda^{\top} \Lambda \mathbf{y}) = 2\Lambda \mathbf{y} \mathbf{y}^{\top}$  and  $\nabla_{\Lambda} \operatorname{tr}(\Lambda) = I$ . From (8.45),  $\nabla_{\Lambda} J(\boldsymbol{\mu}, \Lambda) = 0$  only when  $\Lambda$  equals the empirical approximation of the inverse covariance matrix  $\left( \frac{1}{M} \sum_{m=1}^M (\mathbf{x}_m - \boldsymbol{\mu})(\mathbf{x}_m - \boldsymbol{\mu})^{\top} \right)^{-1}$ .

## 8.6 Exercise

1. Prove the Cauchy-Schwarz inequality  $E[X^2]E[Y^2] \geq E[XY]^2$  by considering  $E[(X - \lambda Y)^2] \geq 0$  and finding  $\lambda$  that minimizing  $E[(X - \lambda Y)^2]$ .
2. Show that for a fixed variance and an unknown mean, the Fisher information given a single observation of a Gaussian distributed variable  $\mathcal{N}(\mu, \sigma^2)$  is equal to  $\frac{1}{\sigma^2}$ . That is, please show that  $J(\boldsymbol{\mu}, \Lambda) = E \left[ \left( \frac{\partial}{\partial \mu} \ln \mathcal{N}(x; \mu, \sigma^2) \right)^2 \right] = \frac{1}{\sigma^2}$ .
3. Given  $N$  observations of  $X \sim \mathcal{N}(0, \sigma^2)$ ,  $X_1, \dots, X_N$ , show that the estimate  $\frac{1}{N} \sum_{i=1}^N X_i^2$  is the optimal estimate of  $\sigma_W^2$  by showing that the estimate reaches the Cramar-Rao Lower Bound.
4. From the derivation of (8.45), please verify that  $\nabla_{\Lambda} (\mathbf{y}^{\top} \Lambda^{\top} \Lambda \mathbf{y}) = 2\Lambda \mathbf{y} \mathbf{y}^{\top}$  and  $\nabla_{\Lambda} \operatorname{tr}(\Lambda) = I$ .



# Appendix A

## Using Lea to solve IT problems

Lea is simple probabilistic programming Python package. There are quite a few probabilistic programming environments such as ... But we select Lea here for its simplicity.

### A.1 Installation

To install Lea, you will need to first install Python. The simplest option is to install Anaconda. For Python beginners, one best environment to work with is the “jupyter-notebook”, which allows users to put in both notes and codes in the same document. For those who have worked with Mathematica before should find it at home, any Python code can be run by pressing shift-enter. Inside jupyter-notebook, one can “pip” install Lea by<sup>1</sup>

```
1 !pip install lea
```

Let’s test our installation by computing  $H(0.38)$ , the entropy of a Bernoulli variable with probability  $p = 0.38$ .

```
1 import lea
2 lea.bernoulli(0.38).entropy #H(0.38)
```

The code itself should be self-explanatory. The first line just load the Lea package. To access Lea’s methods or variables, we should prefix any methods and variables with “lea.”

### A.2 Examples

Python is well-regarded for its shallow learning curve and readability. Here we will assume that the readers have some experience on Python/Numpy. For users with no exposure to Python/Numpy at all, I would like to refer them to many great online tutorials online. For example, I highly recommend the online tutorial<sup>2</sup> by Justin Johnson.

<sup>1</sup>Note that the “!” before pip tells jupyter to run the command as system command rather than Python code. So if you are in the terminal with pip available, you can simply type *pip install lea* to install Lea.

<sup>2</sup><https://cs231n.github.io/python-numpy-tutorial/>

Lea itself has some great documentation online<sup>3</sup> as well. Rather than repeating those, I will just provide several simple examples to let readers get started quickly.

### A.2.1 Burglar alarm

Let's consider a variant of the classic burglar alarm example. Let say there is a chance of 0.001 of a burglar broke in, and a probability of 0.002 of earthquake. And depending on the combination, a burglar alarm may be activated. For example, if both burglary and earthquake happen at the same time, 0.95 of chance the alarm will ring. But if both do not happen, there is still 0.001 chance of false positive and alarm rings.

Whenever the alarm rings, your neighbors John and Mary may call with probabilities 0.9 and 0.7, respectively. They may still call you even when alarm didn't ring, but with smaller probabilities of 0.05 and 0.01. The events are depicted by the undirected graph as shown in Figure A.1, where *B*, *E*, *A*, *J* and *M* corresponds to the events of burglar broke in, earthquake happened, alarm rang, John called, and Mary called.

The entire problem is described in the Lea code below. Note that `lea.event` is very similar to `lea.bernoulli` except the outcomes of the former are `True` and `False` rather than 1 and 0. In line 4, we use `lea.joint` to describe the alarm variable as a conditional event. Depending on different outcomes of burglary and earthquake, alarm will be defined by a different distribution. For example, line 5 indicates that when both burglary and earthquake are `True`, alarm will be defined as a Bernoulli event with probability 0.95 to be `True`. The rest of the variables `john_calls` and `mary_calls` are defined similarly.

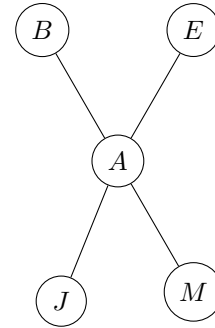


Figure A.1: An undirected graph for the burglar alarm problem

```

1 import lea
2 burglary = lea.event(0.001)
3 earthquake = lea.event(0.002)
4 alarm = lea.joint(burglary,earthquake)\
5     .switch({ (True ,True ) : lea.event(0.950),
6             (True ,False) : lea.event(0.940),
7             (False,True ) : lea.event(0.290),
8             (False,False) : lea.event(0.001) })
9 john_calls = alarm.switch({ True : lea.event(0.90),
10                          False : lea.event(0.05) })
11 mary_calls = alarm.switch({ True : lea.event(0.70),
12                          False : lea.event(0.01) })

```

After the problem was set up, we can find the probabilities for different event readily. For example, we may find the probability of alarm rang by

```

1 lea.P(alarm)

```

And you should get **0.0025164420000000002** if everything was typed correctly.

We can also find the conditional probability of Mary called given the alarm rang by

<sup>3</sup>[https://bitbucket.org/piedenis/lea/wiki/Lea3\\_Tutorial\\_1](https://bitbucket.org/piedenis/lea/wiki/Lea3_Tutorial_1)



```
1 mary_calls.given(alarm)
```

It should return

```
1 False : 0.30000000000000004
2 True  : 0.7
```

We can also compute some information measure directly. For example, we can compute the mutual information between Mary called and John called by

```
1 lea.mutual_information(john_calls,mary_calls)
```

and we should get **0.001740282763951495**.

### A.2.2 A fair dice and a loaded dice

Considered a loaded dice is put into a jar along with three more fair dices. And assume that the loaded dice is two times more likely to get a six then the rest of the outcomes. Let's draw a dice from the jar and toss the drawn dice twice. What is the chance that the dice is loaded if we get both six?

Let's model the whole setup as below.

```
1 import lea
2 import numpy as np
3
4 fair = lea.leaf.dice(1)
5 loaded = lea.pmf(dict(zip(range(1,7),np.array([1,1,1,1,1,2])))
6
7 D = lea.vals(*'fffl') # dice: (f)air/(l)oaded
8 tosses = D.switch({'f':lea.joint(fair,fair.new()),
9                  'l':lea.joint(loaded,loaded.new())})
```

In line 2, we import numpy as np, which is a common convention. Note that Lea has a built-in method, `lea.leaf.dice()`, as defined in line 4. In line 5, we define a loaded dice by depicting its p.m.f. from scratch using `lea.pmf`, which expects a Python dictionary<sup>4</sup> as input, where the keys are the outcomes and the values are the corresponding probabilities. Note that the values `[1,1,1,1,1,2]` of the dictionary do not need to sum up to one as Lea will automatically normalize them. But the values should be non-negative as they represent probabilities. The last 2 corresponds to the probability of six, which is twice the rest as desired.

In line 7, we define the dice variable  $D$  with the population directly. We specified that there are three (f)air dices and one (l)oaded dice<sup>5</sup>. In line 8, we construct `tosses` as a conditional random variable that the next two outcomes are drawn from a fair (loaded) dice if  $D$  is fair (loaded). Note that `new()` is necessary in lines 8 and 9. Otherwise, the two toss outcomes will be identical.

Now, we can ask the original question of *what is the chance of the dice to be loaded if we got both six* with the code below

<sup>4</sup>Note that we use `zip` and `dict` to create a dictionary input for `lea.pmf` in line 5. `zip` converts two equal-length lists into a list of pairs, and `dict` converts the latter into a dictionary with the first element in each pair as key and the second element as value.

<sup>5</sup>Note that `*` in line 7 is a dereferencing operator. In Python, any string is actually a list of characters, so `'fff'` is the same as `['f','f','f']` and `*` will remove the bracket of the list. Thus, the line is the same as `D = lea.vals('f','f','f','l')`.

```
1 D.given(tosses==(6,6))
```

The code is self-explanatory and return the conditional distribution<sup>6</sup> of  $D$  given the both outcomes are six. Lea will output something like

```
1 f : 0.5051546391752577
2 l : 0.4948453608247423
```

Despite we got two six in a row, we probably should not accuse the opponent to be cheating yet even seeing two six as the probability is still less than half. Btw, what is the probability of getting a six again? It should be

$$\begin{aligned} & Pr(D = f)Pr(6|D = f) + Pr(D = l)Pr(6|D = l) \\ &= 0.505 \cdot 1/6 + 0.495 \cdot 2/7 \\ &= 0.226 \end{aligned}$$

Let's modify our code slightly and compute it directly. Define *tosses* as outcome of three tosses rather than two

```
1 tosses = D.switch({'f':lea.joint(fair,fair.new(),fair.new()),
2                  'l':lea.joint(loaded,loaded.new(),loaded.new())})
```

Then, we can output the distribution of the last toss given the first two being six as

```
1 tosses[2].given(tosses[0:2]==(6,6))
```

And this returns

```
1 1 : 0.15488463426607757
2 2 : 0.15488463426607757
3 3 : 0.15488463426607757
4 4 : 0.15488463426607757
5 5 : 0.15488463426607757
6 6 : 0.22557682866961215
```

What if we want to find the probability of getting a six after getting 7 sixes to start with? We can surely redefine *tosses* and repeat *fair.new()* and *loaded.new()* seven times. However, we can also redefine it more flexibly as<sup>7</sup>

```
1 tosses=D.switch({'f':lea.joint(*[fair.new() for i in range(8)]),
2                  'l':lea.joint(*[loaded.new() for i in range(8)])})
```

We can then compute distribution of the last toss given seven sixes by

```
1 tosses[7].given(tosses[:7]==tuple([6]*7))
```

<sup>6</sup>We may also return just the probability of the dice to be loaded by a slightly more wordy expression *lea.P(D.given(tosses==(6,6))=='l')*.

<sup>7</sup>It may be tempting to replace *lea.joint(\*[fair.new() for i in range(8)])* by *lea.joint(\*([fair.new()]\*8))*. However, the latter will not work as *fair.new()* will only be called once and the eight variables will be identical.

And this should returns something like

```

1 1 : 0.1443929328644471
2 2 : 0.1443929328644471
3 3 : 0.1443929328644471
4 4 : 0.1443929328644471
5 5 : 0.1443929328644471
6 6 : 0.27803533567776456

```

Note that the code will run a little while as the number of outcomes ( $6^8 = 1,679,616$ ) is not actually small and Lea is not optimized in computing conditional distribution.

Now, let's try to compute the mutual information between two adjacent toss. To reduce the amount of compute, let us only model four consecutive tosses and use `lea.mutual_information` to compute the value.

```

1 tosses=D.switch({'f':lea.joint(*[fair.new() for i in range(4)]),
2                   'l':lea.joint(*[loaded.new() for i in range(4)]})
3 lea.mutual_information(tosses[0],tosses[1])

```

You should get something like 0.000202. It may be surprising to some that the value is non-zero. As we all learned from our first course in probability that consecutive tosses of a dice should be independent event. However, this is only true when we have the *complete* statistical knowledge of our dice<sup>8</sup>. Moreover, the mutual information for other tosses, say between `toss[0]` and `toss[2]` as computed below,

```

1 lea.mutual_information(tosses[0],tosses[2])

```

should be the same as well.

Finally, let's try to compute the conditional mutual information between two tosses given the dice  $D$ . Note that Lea does not have a function for conditional mutual information, but we can readily compute that as

$$H(\text{tosses}[1]|D) - H(\text{tosses}[1]|D, \text{tosses}[0]).$$

In Lea, this will be computed by

```

1 tosses[1].cond_entropy(D)-tosses[1].cond_entropy(lea.joint(D,tosses[0]))

```

and the value should be very close to zero. The result precisely coincides what we learned in an elementary probability course when consecutive tosses should be independent when complete statistical knowledge of the dice is given.

### A.2.3 Weather on a tropical island

In a tropical island where temperature is rather stable throughout the year. Say the temperature (in degree Celsius) is random with different distributions<sup>9</sup> for sunny and rainy days as tabulated in Table A.1.

<sup>8</sup>Here we mean complete statistical knowledge is that the probability distribution of the dice outcome is precisely known and no hidden variable is involved. The statistical knowledge is not complete here as depends on the outcome of the dice, which we do not know, the outcome distribution can vary.

<sup>9</sup>For simplicity we assume the temperature is discrete.

Table A.1: Temperature distributions of sunny and rainy days

Temp	22	23	24	25	26	27	28	29	30	31	32	33	34
Sunny	0	0	0	0	0.05	0.05	0.1	0.1	0.1	0.2	0.25	0.1	0.05
Rainy	0.1	0.2	0.3	0.2	0.1	0.1	0	0	0	0	0	0	0

Moreover, let's assume that the weather of the island is first-order Markov. Namely, the weather today (sunny or rainy) will only depend on the weather yesterday. More precisely, if today is sunny, the probability of tomorrow to be sunny is 0.95. If today is rainy, the probability of tomorrow remaining rainy is only 0.2. Let's first build the model as follows.

```

1 import numpy as np
2 from lea import markov
3
4 p_sunny=[0.05,0.05,0.1,0.1,0.1,0.2,0.25,0.1,0.05]
5 p_sunny=np.array(p_sunny)
6 p_rainy=[0.1,0.2,0.3,0.2,0.1,0.1]
7 p_rainy=np.array(p_rainy)
8
9 dist_sunny=lea.pmf(dict(zip(range(26,35),p_sunny)))
10 dist_rainy=lea.pmf(dict(zip(range(22,28),p_rainy)))
11
12 weather = markov.chain_from_matrix(('sunny','rainy'),
13                                   ('sunny',(0.95, 0.05)),
14                                   ('rainy',(0.8, 0.2)))

```

The code should be rather self-explanatory. We see most of the methods in the earlier examples already. We have introduced two separate pmfs, `dist_sunny` and `dist_rainy` for the sunny and rainy days, respectively. The only new method is `chain_from_matrix` from the `lea.markov` module. That allows us to create a Markov model variable easily with the transition probabilities.

### If it is rainy today, what is the expected temperature tomorrow?

We can use `weather.next_state('rainy')` to get the weather variable given rainy today. Using `switch` to generate a conditional random variable as in previous example, we can find the expected temperature as follows.

```

1 weather.next_state('rainy').switch({'sunny':dist_sunny.new(),
2                                     'rainy':dist_rainy.new()}).mean
3 >> 29.299999999999997

```

Therefore, the answer is 29.3 C.

### What is the expected temperature tomorrow (without knowing the weather today)?

Without knowing the weather today, we should just assume that the weather distribution reaches a steady state. We can assume it by iterate it by many times (say 1000) from any starting state (say *sunny*). Thus, the expected temperature can be estimated as follows.

```

1 weather.next_state('sunny',1000).switch({'sunny':dist_sunny.new(),
2     'rainy':dist_rainy.new()}).mean
3 >> 30.182352941176468

```

Therefore, the expected temperature is 30.18 C.

### What is the probability that today is sunny if the temperature is higher than 24.5 C?

Since the question does not provide any information of the weather, we will assume we are at the steady state and approximate the weather state by iterating from a sunny state 1000 times (see line 1 below) as in the previous question. We can define the temperature variable (*temp*) conditioned on today's weather

by using the *switch* command as before (see line 2). Finally, we can find the conditional probability of today's weather given temperature larger than 24.5 C by using the *given* statement (see line 3).

```

1 today=weather.next_state('sunny',1000)
2 temp=today.switch({'sunny':dist_sunny.new(),'rainy':dist_rainy.new()})
3 today.given(temp>24.5) # 1c
4
5 >> rainy : 0.024390243902439025
6 >> sunny : 0.975609756097561

```

Therefore, the probability of *sunny* is approximately 0.98.

### If today is rainy, what is the probability that the temperature tomorrow is above 25.5 C?

The question is similar to before but we know today is rainy already. So tomorrow weather will be *weather.next\_state('rainy')* and we can construct the temperature variable as in lines 1 and 2 below. Simply input *temp > 25.5* as in line 3 below will find the probability of temperature below 25.5 C.

```

1 temp=weather.next_state('rainy')
2     .switch({'sunny':dist_sunny,'rainy':dist_rainy})
3 temp> 25.5
4
5 False : 0.16000000000000003
6 True  : 0.8400000000000001

```

Thus, the probability is 0.84.



# Appendix B

## Common distributions

We introduce several common distributions used in this book. The material here is self-contained, requiring only a background in calculus. However, the list of distributions considered here is by no means exhaustive. More information on other distributions can be easily found online from sources such as Wikipedia.

### B.1 Normal distribution

Among all continuous distributions, the normal distribution (also known as the Gaussian distribution) is probably the most well-known and important. As shown in Fig. B.1, the probability density function (PDF) is symmetric around the mean and has a bell shape that exponentially decays away from the mean. Thus, the peak (mode) is exactly the mean as well.

By the Central Limit Theorem (CLT), if we add multiple independent random variables together, the sum will tend to become more and more like a normal distribution, regardless of the original distributions of the variables. One interpretation of the Central Limit Theorem is related to the second law of thermodynamics. Given all other things fixed, entropy tends to increase as a system evolves. Since the normal distribution has the highest entropy among all distributions with a fixed variance, any system of random variables will tend to become more Gaussian as time goes on. See Section 4.3.1 for a detailed discussion on interpreting a special case of the Second Law of Thermodynamics as a consequence of the CLT.

#### PDF

The normal distribution is a continuous probability distribution. Given a mean  $\mu$  and a variance  $\sigma^2$ , its probability density function (pdf) is given by

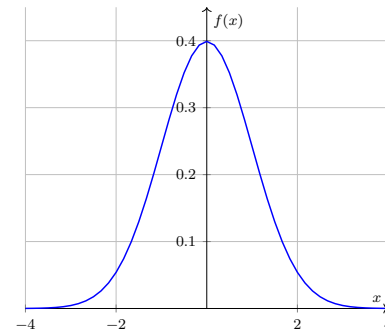


Figure B.1: Gaussian PDF

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Although the expression may appear complicated, it is reasonably easy to remember. The exponent decreases quadratically as the value of  $x$  moves away from the mean  $\mu$ , scaled by the variance  $\sigma^2$ . The normalization factor  $\frac{1}{\sqrt{2\pi\sigma^2}}$  ensures that the pdf integrates to one, since  $\int_{x \in \mathbb{R}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \sqrt{2\pi\sigma^2}$ .

### Summary statistics

The mean and variance of the normal distribution  $\mathcal{N}(x; \mu, \sigma^2)$  are, of course, just  $\mu$  and  $\sigma^2$ . Interestingly, the probability density function (pdf) is parameterized and completely characterized by these two parameters alone.

### B.1.1 Multivariate normal distribution

The multivariate normal (or Gaussian) distribution is a generalization of the one-dimensional Gaussian distribution to higher dimensions. It is commonly used to model vectors of correlated random variables.

#### PDF

The PDF of the multivariate Gaussian distribution for a random vector  $\mathbf{x} \in \mathbb{R}^N$  is given by

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

where  $\boldsymbol{\mu} \in \mathbb{R}^N$  is the mean vector and  $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$  is the covariance matrix.

#### Summary statistics

The mean vector  $\boldsymbol{\mu}$  is an  $N$ -dimensional vector that represents the expected value or average of the random vector  $\mathbf{x}$ . Each element  $\mu_i$  of  $\boldsymbol{\mu}$  is the mean of the corresponding element  $x_i$  of  $\mathbf{x}$ .

The covariance matrix  $\boldsymbol{\Sigma}$  is an  $N \times N$  symmetric positive-definite matrix that describes the covariance between each pair of elements in the random vector  $\mathbf{x}$ . The diagonal elements  $\sigma_{ii}$  of  $\boldsymbol{\Sigma}$  are the variances of the individual elements of  $\mathbf{x}$ , while the off-diagonal elements  $\sigma_{ij}$  (for  $i \neq j$ ) represent the covariances between different elements  $x_i$  and  $x_j$ , given by  $E[(X_i - \mu_i)(X_j - \mu_j)]$ .

The covariance matrix  $\boldsymbol{\Sigma}$  provides important information about the shape and orientation of the distribution. When  $\boldsymbol{\Sigma}$  is diagonal, the elements of  $\mathbf{x}$  are uncorrelated and independent<sup>1</sup>, and the distribution is spherical. When  $\boldsymbol{\Sigma}$  has off-diagonal elements, it indicates correlations between the elements of  $\mathbf{x}$ , resulting in an elliptical distribution shape. See Fig. B.2 for a illustration.

<sup>1</sup>In general, independence is a stronger condition than uncorrelatedness. However, for the multivariate Gaussian distribution, the two conditions imply one another.



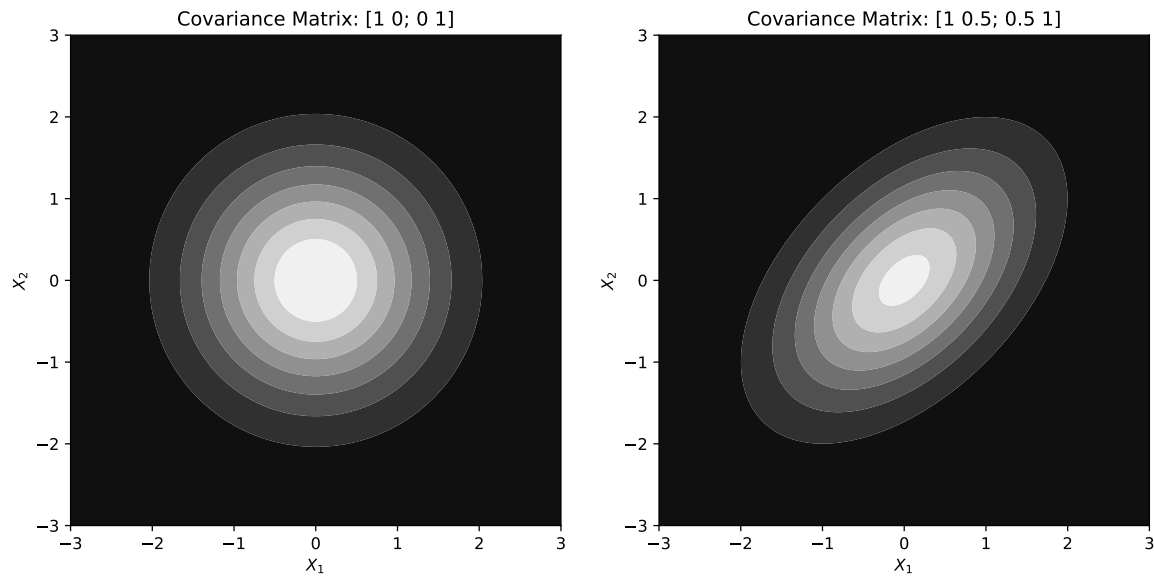


Figure B.2: The PDFs of 2-D multivariate normal distributions with covariance matrices  $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (left) and  $\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$  (right).

## B.2 Bernoulli distribution

The Bernoulli distribution is most well-known for describing the random behavior of tossing a coin once.

### PMF

Denote  $X = 1$  for a head and  $X = 0$  for a tail. Let  $Pr(X = 1) = p$ . Then the Bernoulli distribution is simply

$$\text{Bern}(x; p) = \begin{cases} p, & x = 1, \\ 1 - p, & x = 0. \end{cases}$$

More concisely, we can write it as

$$\text{Bern}(x; p) = p^x(1 - p)^{1-x},$$

### Summary Statistics

The mean and variance can be easily computed as

$$E[X] = p \cdot 1 + (1 - p) \cdot 0 = p, \quad (\text{B.1})$$

$$\text{Var}[X] = p \cdot (1 - p)^2 + (1 - p) \cdot p^2 = p(1 - p). \quad (\text{B.2})$$

## B.3 Binomial distribution

The Binomial distribution, an extension of the Bernoulli distribution, describes the experiment of  $N$  repeating Bernoulli trials.

### PMF

Consider the coin tossing example again, where we will toss a coin  $N$  times. Let  $x$  be the number of obtained heads. The probability mass function (PMF) is given by:

$$\text{Bin}(x; p, N) = \binom{N}{x} p^x (1 - p)^{N-x},$$

where  $\binom{N}{x}$  is the binomial coefficient.

As shown in Fig. B.3, the binomial distribution can be approximated well with a normal distribution for large  $N$ .

### Summary Statistics

Using the fact that  $\sum_{x=0}^N \text{Bin}(x; p, N) = 1$  for any  $p$  and  $N$ , we can easily compute the mean of the Binomial distribution as

$$\begin{aligned} E[X] &= \sum_{x=0}^N \text{Bin}(x; p, N) \cdot x \\ &= \sum_{x=1}^N \frac{N!}{(x-1)!(N-x)!} p^x (1-p)^{N-x} \\ &= Np \sum_{x=1}^N \frac{(N-1)!}{(x-1)!(N-x)!} p^{x-1} (1-p)^{N-x} \\ &= Np \sum_{x=0}^{N-1} \text{Bin}(x; p, N-1) \\ &= Np. \end{aligned}$$

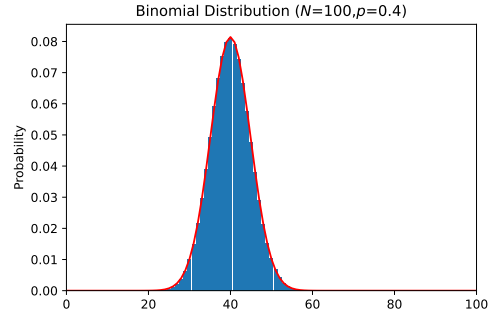


Figure B.3: The binomial distribution is represented by solid bars, while the normal distribution approximation (with mean  $Np = 40$  and variance  $Np(1 - p) = 24$ ) is depicted by the curve above them.

Similarly, we have

$$\begin{aligned} E[X(X-1)] &= \sum_{x=2}^N \frac{N!}{(x-2)!(N-x)!} p^x (1-p)^{N-x} \\ &= N(N-1)p^2 \sum_{x=0}^{N-2} \text{Bin}(x; p, N-2) \\ &= N(N-1)p^2. \end{aligned}$$

Therefore, the variance is given by

$$\begin{aligned} \text{Var}[X] &= E[X^2] - E[X]^2 \\ &= E[X(X-1)] + E[X] - E[X]^2 \\ &= N(N-1)p^2 + Np - (Np)^2 \\ &= Np(1-p). \end{aligned}$$

## B.4 Beta Distribution

For a fixed  $N$  and a valid  $p$  ( $0 \leq p \leq 1$ ), we can specify a binomial distribution  $\text{Bin}(x; p, N)$ . By defining a distribution over all valid  $p$ , we can create a distribution of binomial distributions.

This type of distribution is often used as a prior in Bayesian estimation. For example, the posterior distribution will be proportional to  $\text{Bin}(x; p, N)q(p)$ .

Note that there are infinite ways to define such a distribution of  $p$  as long as  $0 \leq p \leq 1$  and the distribution is normalized such that  $\int_{p=0}^1 q(p) dp = 1$ , where  $q(p)$  is the PDF of  $p$ . However, for mathematical convenience, it is useful to choose  $q(p)$  to have the same form as  $\text{Bin}(x; p, N)$ . Consequently, we have  $q(p) \propto p^a(1-p)^b$ , resulting in the Beta distribution, which is known to be the conjugate prior of Binomial distribution (see Section 2.6.2).

### PDF

Given parameters  $a$  and  $b$ , the PDF of the Beta distribution is given by<sup>2</sup>

$$\text{Beta}(x; a, b) = \frac{x^{a-1}(1-x)^{b-1}}{B(a, b)},$$

where  $x \in [0, 1]$ ,  $B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ , and  $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t} dt$  is the Gamma function. The normalization factor  $B(a, b)$  ensures that  $\int_{x=0}^1 \text{Beta}(x; a, b) dx = 1$ . The PDFs  $\text{Beta}(x; a, b)$  for various values of  $a$  and  $b$  are shown in Figure B.4. Note that when  $a = b = 1$ ,  $\text{Beta}(x; 1, 1) = 1$ , resulting in a uniform prior.

<sup>2</sup>By convention,  $\text{Beta}(x; a, b) \propto x^{a-1}(1-x)^{b-1}$  rather than  $\propto x^a(1-x)^b$ .

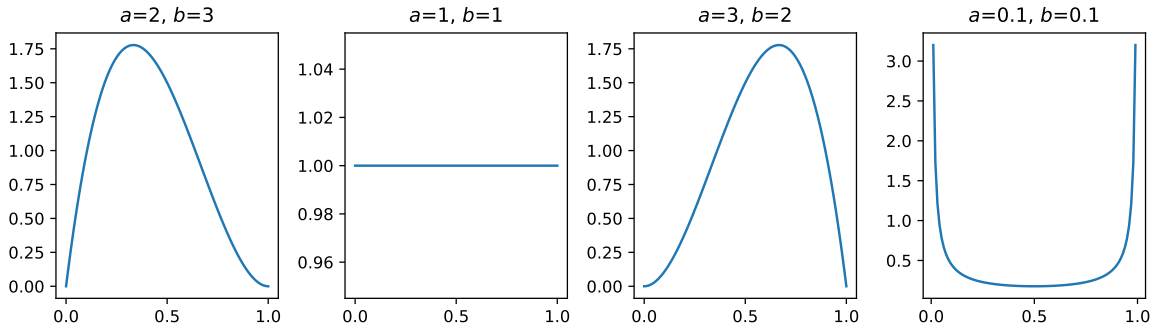


Figure B.4: PDF of different Beta distributions

### Summary statistics

The  $\Gamma(z)$  in  $B(a, b)$  may look intimidating to some students. However, for most purposes, we only need to remember one simple property to effectively manipulate it.

**Fact** For  $z > 1$ , we have

$$\Gamma(z) = (z - 1)\Gamma(z - 1) \quad (\text{B.3})$$

*Proof.*

$$\begin{aligned} \Gamma(z) &= \int_0^{\infty} x^{z-1} e^{-x} dx \\ &= - \int_0^{\infty} x^{z-1} de^{-x} \\ &= -x^{z-1} e^{-x} \Big|_0^{\infty} + (z-1) \int_0^{\infty} x^{z-2} e^{-x} dx \\ &= (z-1) \int_0^{\infty} x^{z-2} e^{-x} dx \\ &= (z-1)\Gamma(z-1). \end{aligned}$$

□

By using the above fact repeatedly, we immediately have  $\Gamma(z) = (z-1)!$  for integer  $z > 1$ . Thus, we can interpret  $\Gamma(z)$  as a natural extension of  $(z-1)!$  to the entire real domain.

Since  $\int_{x=0}^1 p(x|a, b) = 1$ , we have

$$\int_{x=0}^1 x^{a-1} (1-x)^{b-1} dx = B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}. \quad (\text{B.4})$$

Along with (B.3), we have enough tools to find the mean and variance of a Beta-distributed  $X$  easily.

$$\begin{aligned}
E[X] &= \int_{x=0}^1 x \text{Beta}(x|a, b) dx = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_{x=0}^1 x^a (1-x)^{b-1} dx \\
&\stackrel{(a)}{=} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+1)\Gamma(b)}{\Gamma(a+b+1)} \stackrel{(b)}{=} \frac{a}{a+b},
\end{aligned}$$

where we take advantage of (B.4) and (B.3), respectively. Similarly,

$$E[X^2] = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_{x=0}^1 x^{a+1} (1-x)^{b-1} dx \quad (\text{B.5})$$

$$\stackrel{(a)}{=} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+2)\Gamma(b)}{\Gamma(a+b+2)} \quad (\text{B.6})$$

$$\stackrel{(b)}{=} \frac{a(a+1)}{(a+b)(a+b+1)}, \quad (\text{B.7})$$

where we again leverage (B.4) and (B.3) for (a) and (b) above. Thus,

$$\begin{aligned}
\text{Var}[X] &= E[X^2] - E[X]^2 \\
&= \frac{a(a+1)}{(a+b)(a+b+1)} - \frac{a^2}{(a+b)^2} \\
&= \frac{a(a+1)(a+b) - a^2(a+b+1)}{(a+b)^2(a+b+1)} \\
&= \frac{ab}{(a+b)^2(a+b+1)}.
\end{aligned}$$

Besides the mean and variance, knowing the mode of the Beta distribution can be handy, for example, in the case of maximum likelihood estimation. The mode of a distribution is the peak of the distribution. Recall that  $\text{Beta}(x|a, b) = \frac{x^{a-1}(1-x)^{b-1}}{B(a, b)}$ . Setting

$$\frac{\partial \text{Beta}(x|a, b)}{\partial x} = \frac{(a-1)x^{a-2}(1-x)^{b-1} - (b-1)x^{a-1}(1-x)^{b-2}}{B(a, b)} = 0,$$

we have  $(a-1)(1-x) = (b-1)x \Rightarrow x = \frac{a-1}{a+b-2}$  when  $a, b > 1$ . Note that when  $a$  or  $b$  is less than or equal to 1, the peak appears at either  $x = 0$  or  $x = 1$ .

## B.5 Multinomial distribution

Binomial distribution models the probability of a binary outcome. For a random event with discrete but non-binary (more than two) outcomes, we can model the event with a multinomial distribution.

### PMF

Let's say the probability of each possible outcome  $i$  is  $p_i$ , where  $p_1 + p_2 + \dots + p_n = 1$ . If we have conducted  $N$  experiments, let  $x_i$  be the number of times we obtain outcome  $i$  such that

$x_1 + x_2 + \cdots + x_n = N$ . The probability of this specific set of outcomes is given by

$$\text{Mult}(x_1, \dots, x_n | p_1, \dots, p_n) = \frac{N!}{x_1! x_2! \cdots x_n!} p_1^{x_1} p_2^{x_2} \cdots p_n^{x_n},$$

where the multinomial coefficient  $\frac{N!}{x_1! x_2! \cdots x_n!}$  counts the number of distinct ways to partition  $N$  items into  $n$  groups of specified sizes  $x_1, x_2, \dots, x_n$ .

## B.6 Dirichlet distribution

Just as in the case of the Beta distribution for the Binomial distribution, it is convenient to select a prior with the same form as the Multinomial distribution, i.e.,  $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}$ . The resulting conjugate prior of the Multinomial distribution is known as the Dirichlet distribution.

### PDF

The PDF of the Dirichlet distribution is given by

$$\text{Dir}(x_1, \dots, x_n | \alpha_1, \dots, \alpha_n) = \frac{\Gamma(\alpha_1 + \cdots + \alpha_n)}{\Gamma(\alpha_1) \Gamma(\alpha_2) \cdots \Gamma(\alpha_n)} x_1^{\alpha_1 - 1} x_2^{\alpha_2 - 1} \cdots x_n^{\alpha_n - 1}$$

Since the PDF should be normalized to 1, we have

$$\int_{\substack{x_1, x_2, \dots, x_n \geq 0 \\ x_1 + x_2 + \cdots + x_n = 1}} x_1^{\alpha_1 - 1} \cdots x_n^{\alpha_n - 1} dx_1 \cdots dx_n = \frac{\Gamma(\alpha_1) \Gamma(\alpha_2) \cdots \Gamma(\alpha_n)}{\Gamma(\alpha_1 + \cdots + \alpha_n)}. \quad (\text{B.8})$$

Note that  $x_1, x_2, \dots, x_n$  represent the probabilities for the outcomes of a discrete random variable. Therefore, the given constraints under the integral ( $x_1, x_2, \dots, x_n \geq 0$  and  $x_1 + x_2 + \cdots + x_n = 1$ ) are necessary.

### Summary statistics

Similar to the case of the Beta distribution, we can compute the mean using (B.3). Let's compute the mean of the first element  $X_1$ . The means of the other elements can be computed in a similar manner.

$$\begin{aligned} E[X_1] &= \frac{\Gamma(\alpha_1 + \cdots + \alpha_N)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_N)} \int x_1^{\alpha_1} x_2^{\alpha_2 - 1} \cdots x_N^{\alpha_N - 1} dx_1 dx_2 \cdots dx_N \\ &\stackrel{(a)}{=} \frac{\Gamma(\alpha_1 + \cdots + \alpha_N)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_N)} \frac{\Gamma(\alpha_1 + 1) \Gamma(\alpha_2) \cdots \Gamma(\alpha_N)}{\Gamma(\alpha_1 + \cdots + \alpha_N + 1)} \\ &\stackrel{(b)}{=} \frac{\alpha_1}{\alpha_1 + \cdots + \alpha_N}, \end{aligned}$$

where (a) and (b) are due to (B.8) and (B.3), respectively.

Similarly,

$$\begin{aligned}
 E[X_1^2] &= \frac{\Gamma(\alpha_1 + \cdots + \alpha_N)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_N)} \int x_1^{\alpha_1+1} x_2^{\alpha_2-1} \cdots x_N^{\alpha_N-1} dx_1 dx_2 \cdots dx_N \\
 &= \frac{\Gamma(\alpha_1 + \cdots + \alpha_N)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_N)} \frac{\Gamma(\alpha_1 + 2)\Gamma(\alpha_2) \cdots \Gamma(\alpha_N)}{\Gamma(\alpha_1 + \cdots + \alpha_N + 2)} \\
 &= \frac{(\alpha_1 + 1)\alpha_1}{(\alpha_1 + \cdots + \alpha_N + 1)(\alpha_1 + \cdots + \alpha_N)}.
 \end{aligned}$$

Thus,

$$\begin{aligned}
 \text{Var}(X_1) &= E[X_1^2] - E[X_1]^2 \\
 &= \frac{(\alpha_1 + 1)\alpha_1}{(\alpha_1 + \cdots + \alpha_N + 1)(\alpha_1 + \cdots + \alpha_N)} - \frac{\alpha_1^2}{(\alpha_1 + \cdots + \alpha_N)^2} \\
 &= \frac{\alpha_1(\alpha_1 + \alpha_0 - \alpha_1) - \alpha_1^2}{(\alpha_1 + \cdots + \alpha_N)^2(\alpha_1 + \cdots + \alpha_N + 1)} \\
 &= \frac{\alpha_1(\alpha_0 - \alpha_1)}{\alpha_0^2(\alpha_0 + 1)},
 \end{aligned}$$

where  $\alpha_0 = \alpha_1 + \cdots + \alpha_N$ .

One can show that the mode of  $\text{Dir}(\alpha_1, \dots, \alpha_N)$  with respect to  $\alpha_i$  when  $\alpha_1, \dots, \alpha_N > 1$  is

$$\frac{\alpha_i - 1}{\alpha_1 + \cdots + \alpha_N - N}.$$

The derivation is similar to the case of the Beta distribution and is left as an exercise.

## B.7 Exercise

1. Show that  $\text{Beta}(x; 1, 1) = 1$ .
2. Find the mode of  $\text{Dir}(x_1, \dots, x_N; \alpha_1, \dots, \alpha_N)$  with respect to  $x_i$ .





# Appendix C

## Lagrange multiplier and Karush-Kuhn-Tucker (KKT) conditions

### C.1 Constrained optimization

The goal of an optimization problem is finding the maximum or minimum of some function. For unconstrained optimization, there is no restriction on the possible input of the function. In contrast, a constrained optimization problem comes with a set of constraints on the input. In a canonical form, we can summarize all constrained optimization problems into something like the following:

$$\begin{aligned} & \min f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, \\ & h_i(\mathbf{x}) = 0, \end{aligned} \tag{C.1}$$

where  $g_i(\cdot)$  are the inequality constraints and  $h_i(\cdot)$  are the equality constraints.

We know from calculus that the extremum (maximum or minimum) of a smooth function occur when the point is flat or of zero gradient. However, for a constrained optimization problem, this no longer needs to be true. For example, the minimum of  $x^2$  for  $x \geq 1$  is equal to 1 (at  $x = 1$ ) as shown in Figure C.1a, where the gradient (or slope) of  $x^2$  is  $2x = 2 \neq 0$  at  $x = 1$ .

### C.2 Lagrange multiplier

The idea of the technique of Lagrange multiplier is to absorb the constraints into the objective function so as to reformulate the constrained optimization into an unconstrained optimization problem. As for the earlier problem as shown in (C.1), we could reformulate it as

$$\min f(\mathbf{x}) + \sum_i \lambda_i g_i(\mathbf{x}) + \sum_i \mu_i h_i(\mathbf{x}), \tag{C.2}$$

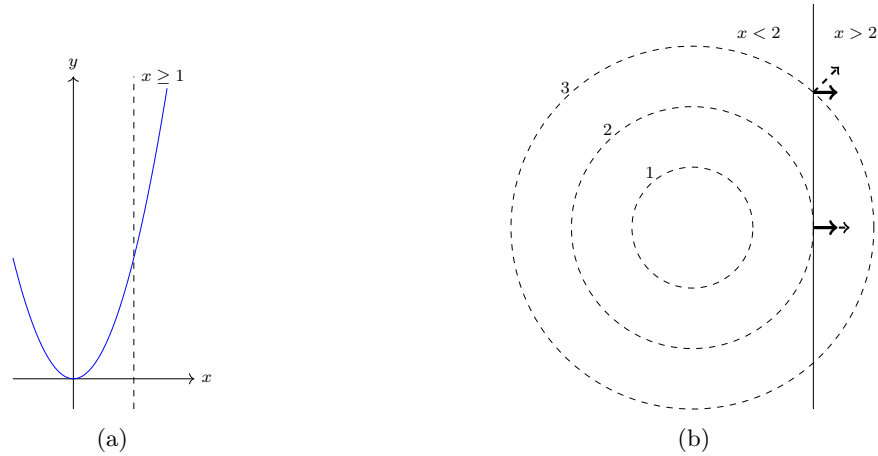


Figure C.1: Understanding Lagrange multiplier. In the left figure, the minimum of  $y = x^2$  for  $x \geq 1$  is at  $x = 1$ , where the slope there is not zero as one would expect for unconstrained optimization. In the right figure, we consider  $\min f(x, y)$  s.t.  $g(x, y) = 0$ , with  $f(x, y) = \sqrt{x^2 + y^2}$  and  $h(x, y) = x - 2$ .  $\nabla f(x, y)$  and  $\nabla h(x, y)$  are in parallel ( $\nabla f = \lambda \nabla h$ ) at the minimum  $(2, 0)$ .

where  $\lambda_i$  and  $\mu_i$  are known as the Lagrange multipliers.

### C.2.1 Geometric intuition

To understand why we can absorb the constraints as in (C.2). Note that it does not hurt to ignore the inequality constraints conceptually. If the optimum  $\mathbf{x}^*$  satisfies an inequality constraint as an equality (e.g.,  $g_i(\mathbf{x}^*) = 0$ ), then we can simply treat the constraint as another equality constraint. On the other hand, if we have  $g_i(\mathbf{x}^*) < 0$ , the constraint is inactive and has no direct effect on the optimal value<sup>1</sup>. Consequently, let us restrict ourselves to the equality constraints only in the discussion in this sub-section.

Let us consider a trivial constrained optimization problem

$$\begin{aligned} \min \quad & \sqrt{x^2 + y^2} \\ \text{s.t.} \quad & x = 2 \end{aligned} \tag{C.3}$$

as shown in Fig. C.1. Without the constraint, the solution is simply 0 with the optimum at  $(0, 0)$ . With the constraint, the minimum will be 2 as we are forced to pick  $x = 2$  while we can still choose 0 for  $y$ .

From Fig. C.1b, we see at the optimum  $(2, 0)$  that the gradients of the objective function  $f(x, y) = \sqrt{x^2 + y^2}$  and the constraint  $h(x, y) = x - 2$  must point to the same direction. This is because the hyperplane (a line in this case) that satisfies the constraint is always perpendicular to the gradient of the constrained function. So moving along this plane normal to the gradient will always preserve the constraint. Apparently if the gradient of the objective function is also not perpendicular to this plane, we can always make the value smaller by perturbing the point along the

<sup>1</sup>We will revisit shortly when we talk about KKT conditions.

plane. So both gradients must be normal to the constrained plane and thus they must be aligned. Mathematically, we have  $\nabla f = \lambda \nabla h$ , where  $\lambda$  is precisely the Lagrange multiplier. And thus

$$\nabla(f - \lambda h) = 0 \quad (\text{C.4})$$

and this is equivalent to solving the unconstrained optimization problem

$$\min f(\mathbf{x}) - \lambda h(\mathbf{x}). \quad (\text{C.5})$$

## C.2.2 Algebraic proof

### Equality constraint

After gaining some intuition of the origin of the Lagrange multiplier, let's try to argue the method algebraically. Consider a simple optimization problem with a single equality constraint,

$$\begin{aligned} & \max_{\mathbf{x}} f(\mathbf{x}) && (\text{C.6}) \\ \text{s.t.} & \quad h(\mathbf{x}) = 0 \end{aligned}$$

Define  $L(\mathbf{x}, \lambda) \triangleq f(\mathbf{x}) - \lambda h(\mathbf{x})$  and let  $\tilde{f}(\mathbf{x}) = \min_{\lambda} L(\mathbf{x}, \lambda)$ . Note that

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } h(\mathbf{x}) = 0 \\ -\infty & \text{otherwise} \end{cases} \quad (\text{C.7})$$

because we can choose  $\lambda = \infty$  when  $h(\mathbf{x})$  is non-zero.

Consequently, we can never get any solution that does not satisfy  $h(\mathbf{x}) = 0$  when we try to maximize  $\tilde{f}(\mathbf{x})$ . Therefore, the original constrained problem is identical to  $\max_{\mathbf{x}} \tilde{f}(\mathbf{x})$  or

$$\max_{\mathbf{x}} \min_{\lambda} (f(\mathbf{x}) - \lambda h(\mathbf{x})), \quad (\text{C.8})$$

where  $\lambda$  again is the Lagrange multiplier.

If the optimum is a saddle point<sup>2</sup> that  $L(\mathbf{x}, \lambda)$  is convex for w.r.t.  $\lambda$  and concave w.r.t.  $\mathbf{x}$ , we have

$$\underbrace{\max_{\mathbf{x}} \min_{\lambda} (f(\mathbf{x}) - \lambda h(\mathbf{x}))}_{\text{Primal problem}} = \underbrace{\min_{\lambda} \max_{\mathbf{x}} (f(\mathbf{x}) - \lambda h(\mathbf{x}))}_{\text{Dual problem}},$$

where the original problem in (??) and the newly formulated problem on the R.H.S. are known as the primal and dual problems, respectively. Furthermore, the dual problem contains  $\max_{\mathbf{x}} (f(\mathbf{x}) - \lambda h(\mathbf{x}))$  as a subproblem as discussed earlier.

---

<sup>2</sup>If the optimum is not a saddle point, the primal solution will be smaller than the dual solution, resulting a duality gap. The solution of the dual problem will provide an upper bound of the solution of the original problem instead.

**Inequality constraint**

Now, let's consider the following optimization problem with a single inequality constraint:

$$\begin{aligned} \max_{\mathbf{x}} \quad & f(\mathbf{x}) \\ & g(\mathbf{x}) \leq 0 \end{aligned}$$

This time let's define  $\tilde{f}(\mathbf{x}) = \min_{\lambda \geq 0} (f(\mathbf{x}) - \lambda g(\mathbf{x}))$ . By the same argument as before, note that

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } g(\mathbf{x}) \leq 0 \\ -\infty & \text{otherwise} \end{cases} \quad (\text{C.9})$$

Therefore, we can rewrite the problem as

$$\max_{\mathbf{x}} \min_{\lambda \geq 0} (f(\mathbf{x}) - \lambda g(\mathbf{x})), \quad (\text{C.10})$$

where (C.12) is almost identical to (C.8) but  $\lambda$  has to be non-negative this time.

Assume that the optimum is at a saddle point that one can swap the max and min as before, we have

$$\underbrace{\max_{\mathbf{x}} \min_{\lambda \geq 0} (f(\mathbf{x}) - \lambda g(\mathbf{x}))}_{\text{Primal problem}} = \underbrace{\min_{\lambda \geq 0} \max_{\mathbf{x}} (f(\mathbf{x}) - \lambda g(\mathbf{x}))}_{\text{Dual problem}}$$

and the dual problem once again leads to the Lagrange problem  $\max_{\mathbf{x}} (f(\mathbf{x}) - \lambda g(\mathbf{x}))$ .

**Multiple constraints**

We can easily generalize the above argument to problem multiple constraints. For example, consider the following problem with  $N$  inequality constraints  $g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_N(\mathbf{x})$  and  $M$  equality constraints  $h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_M(\mathbf{x})$

$$\begin{aligned} \max_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, h_j(\mathbf{x}) = 0 \end{aligned} \quad (\text{C.11})$$

By the same argument as before, note that we can rewrite the problem as

$$\max_{\mathbf{x}} \min_{\lambda_i \geq 0, \mu_j} (f(\mathbf{x}) - \sum_{i=1}^N \lambda_i g_i(\mathbf{x}) - \sum_{j=1}^M \mu_j h_j(\mathbf{x})), \quad (\text{C.12})$$

where the dual problem once again leads to the Lagrange problem

$$\max_{\mathbf{x}} \left( f(\mathbf{x}) - \sum_{i=1}^N \lambda_i g_i(\mathbf{x}) - \sum_{j=1}^M \mu_j h_j(\mathbf{x}) \right). \quad (\text{C.13})$$

### C.2.3 Physical interpretation of Lagrange multiplier

A Lagrange multiplier can be physically interpreted as the rate of change of the optimum relative to the respective active constraint. For example, consider the problem of a factory maximizing its production with a cost constraint and energy constraint as follows,

$$\begin{aligned} & \max_{\mathbf{x}} P(\mathbf{x}) \\ \text{s.t.} \quad & C(\mathbf{x}) \leq C_0, \end{aligned} \tag{C.14}$$

where  $C(\cdot)$  and  $C_0$  are the cost function and the cost budget, respectively.

As we apply Lagrange multipliers to the problem, we have<sup>3</sup>

$$\nabla P(\mathbf{x}^*) = \lambda^* \nabla C(\mathbf{x}^*) \tag{C.15}$$

for the optimal  $\mathbf{x}^*$ ,  $\lambda^*$ , and  $\mu^*$ . Let's assume that the cost is an active constraint, namely  $C(\mathbf{x}) = C_0$  reaches its maximum value. As we perturb the constraint, say to allow  $C(\mathbf{x}) \leq C_0 + \Delta C$ . We will have some  $\Delta \mathbf{x}$  such that  $C(\mathbf{x}^* + \Delta \mathbf{x}) = C_0 + \Delta C$  and the newly optimal production will then be

$$P(\mathbf{x}^*) + \Delta P = P(\mathbf{x}^* + \Delta \mathbf{x}) \tag{C.16}$$

$$= P(\mathbf{x}^*) + \nabla P(\mathbf{x}^*) \cdot \Delta \mathbf{x} \tag{C.17}$$

$$= P(\mathbf{x}^*) + \lambda^* \nabla C(\mathbf{x}^*) \cdot \Delta \mathbf{x} \tag{C.18}$$

$$= P(\mathbf{x}^*) + \lambda \Delta C. \tag{C.19}$$

Therefore, we have the Lagrange multiplier  $\lambda$  equal to the the rate of change of the optimum to the change of the constraint  $\lambda = \frac{\Delta P}{\Delta C}$ .

## C.3 Karush-Kuhn-Tucker (KKT) conditions

The Karush-Kuhn-Tucker (KKT) is a set of conditions that the optimum of a constrained optimization problem must satisfy. Most of the conditions are apparent using the Lagrange multiplier technique.

For the general optimization problem as defined in (C.11), the KKT conditions are listed below

$$\nabla f(\mathbf{x}^*) - \sum_{i=1}^N \lambda_i^* \nabla g_i(\mathbf{x}^*) - \sum_{j=1}^M \mu_j^* \nabla h_j(\mathbf{x}^*) = 0, \tag{C.20}$$

$$g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, N, \tag{C.21}$$

$$h_j(\mathbf{x}^*) = 0, j = 1, \dots, M, \tag{C.22}$$

$$\lambda_i^* \geq 0, i = 1, \dots, N, \tag{C.23}$$

$$\lambda_i^* g_i(\mathbf{x}^*) = 0, i = 1, \dots, N, \tag{C.24}$$

where (C.20) just came from (C.13), (C.21) and (C.22) are the original constraints that need to be satisfied, and (C.23) is needed to ensure the unconstrained objective function will reach minus infinity when any inequality constraint is not satisfied (see (C.12) and the respective discussion).

<sup>3</sup>To avoid notation clutter, we write  $P(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}$  as  $P(\mathbf{x}^*)$  here. Similar notation is also applied to  $C(\cdot)$ .

The last condition in (C.24) is commonly known as the complementary slackness condition, which will be elaborated in the following.

### Complementary slackness condition

The complementary slackness condition probably is not too surprising as at the optimum point  $(\mathbf{x}^*, \lambda_i^*, \mu_j^*)$ , we should have

$$f(\mathbf{x}^*) = \max_{\substack{\mathbf{x} \\ g(\mathbf{x}) \leq 0}} f(\mathbf{x}) \equiv \max_{\mathbf{x}} \min_{\lambda \geq 0} \left( f(\mathbf{x}) - \sum_i \lambda_i g_i(\mathbf{x}) - \sum_j \mu_j h_j(\mathbf{x}) \right) = f(\mathbf{x}^*) - \sum_i \lambda_i^* g_i(\mathbf{x}^*)$$

This means that  $\sum_i \lambda_i^* g_i(\mathbf{x}^*) = 0$ , which is not a proof but does suggest that individual  $\lambda_i^* g_i(\mathbf{x}^*)$  may equal to 0.

We can obtain a rigorous proof from the discussion in Section C.2.3. Note that as we perturb the constraint for  $g_i(\cdot)$  with  $\Delta g_i$ ,  $\lambda_i^* = \frac{\Delta f}{\Delta g_i}$  with  $\Delta f$  as the change of the optimum value. When the constraint is not active, i.e.,  $g_i(\mathbf{x}^*) < 0$ , perturbing the constraint ( $\Delta g_i \neq 0$ ) should introduce no change with the optimum value, and so  $\lambda_i^* = \frac{\Delta f}{\Delta g_i} = \frac{0}{\Delta g_i} = 0$ . Thus, we have  $\lambda_i^* g_i(\mathbf{x}^*) = 0$ .

On the other hand, if the constraint is active and  $g_i(\mathbf{x}^*) = 0$ , we have  $\lambda_i^* g_i(\mathbf{x}^*) = 0$  regardless the value of  $\lambda_i^*$ . So in either case, we have the complementary slackness condition satisfied for the optimum values.

# Appendix D

## Exponential family distributions and their conjugate priors

### D.1 Motivation

Consider a discrete random variable  $X$  with some known expectation  $E[T_i(X)] = \mu_i$ , where  $T_i(\cdot)$  is some observation. One may ask, what is the most probable distribution for this random variable?

To answer this, we can maximize the entropy  $H(X)$  with given the constraints  $E[T_i(X)] = \mu_i$ . Along with the constraints  $p(x) \geq 0$  and  $\sum_x p(x) = 1$ , we can formulate the optimization problem into the following,

$$\max_{p(x)} \min_{\substack{\eta_1, \dots, \eta_m, \lambda \\ \tilde{g}(x) \geq 0}} \underbrace{\left[ H(p) + \sum_{i=1}^m \eta_i (E[T_i(X)] - \mu_i) + \lambda \left( \sum_x p(x) - 1 \right) + \sum_x \tilde{g}(x) p(x) \right]}_{L(p)}, \quad (\text{D.1})$$

where  $\lambda, \eta_1, \eta_2, \dots, \eta_m, \tilde{g}(x)$  are the Lagrange multipliers.

For the optimum  $p(x)$ , we want<sup>1</sup>

$$0 = \frac{\partial L(p)}{\partial p(x')} = -\log p(x') - 1 + \sum_{i=1}^m \eta_i T_i(x') + \lambda + \tilde{g}(x') \quad (\text{D.2})$$

$$\Rightarrow p(x') = \underbrace{\exp \tilde{g}(x')}_{g(x')} \exp(\eta^\top T(x') - 1 + \lambda) \quad (\text{D.3})$$

<sup>1</sup>In most texts, the notation  $h(x)$  is used instead of  $g(x)$  for the formulas below. However, we employ  $g(x)$  in this context to prevent any confusion with differential entropy.

where  $T(x') = \begin{pmatrix} T_1(x') \\ T_2(x') \\ \vdots \\ T_m(x') \end{pmatrix}$  and  $\eta = \begin{pmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{pmatrix}$ . Apply the constraint  $\sum_x p(x) = 1$  and rename the dummy variable  $x'$  back to  $x$ , we have

$$p(x) = \frac{g(x) \exp(\eta^\top T(x))}{\sum_{x''} g(x'') \exp(\eta^\top T(x''))}, \quad (\text{D.4})$$

where the denominator  $Z(\eta) \triangleq \sum_{x''} g(x'') \exp(\eta^\top T(x''))$  is known as the partition function and note that it is only a function of  $\eta$  alone (but not  $x$ ). Denote  $A(\eta) \triangleq \log(Z(\eta))$  as the log-partition function. We can rewrite (D.4) as

$$p(x) = g(x) \exp(\eta^\top T(x) - A(\eta)), \quad (\text{D.5})$$

which is the canonical form of the exponential family distribution.

**Remark 1.** Although we assume  $X$  to be discrete in the above derivation, the derivation for continuous variable is essentially the same, with summations being substituted by integrals. The resulting canonical expression for the exponential family distributions remains the same, as presented in (D.5).

**Remark 2.** Given the expression in (D.5), we observe that  $p(x)$  is derived by maximizing the entropy subject to the constraints  $E[T_i(X)] = \mu_i$  for  $i = 1, \dots, m$ . Consequently, the set  $\{E[T_i(X)]\}_{i=1}^m$  encompasses all the sufficient statistics required to determine the distribution for a member of the exponential family.

### D.1.1 Gaussian distribution as an exponential family distribution

Consider a continuous random variable  $X$  with mean and variance constrained to  $\mu$  and  $\sigma^2$ , respectively. From Chapter 4, we show that the distribution that maximizes the differential entropy is Gaussian with the tool of KL-divergence. Let's take a more direct approach to show this here through optimization. Given the constraints  $T_1(X) = E[X] = \mu$  and  $T_2(X) = E[(X - \mu)^2] = \sigma^2$ , we can rewrite the optimization problem in (D.1) to

$$\max_{p(x)} \min_{\substack{\eta_1, \dots, \eta_m, \lambda \\ \tilde{g}(x) \geq 0}} \underbrace{\left[ h(p) + \eta_1 (E[T_1(X)] - \mu) + \eta_2 (E[T_2(X)] - \sigma^2) + \lambda \left( \int_x p(x) dx - 1 \right) + \int_x \tilde{g}(x) p(x) dx \right]}_{L(p)}, \quad (\text{D.6})$$



Minimizing  $L(p)$  by taking derivative of  $L$  with respect to  $p(x)$  and setting it to 0, we get<sup>2</sup>

$$\frac{\delta L}{\delta p(x)} = -\ln p(x) - 1 + \eta_1 T_1(x) + \eta_2 T_2(x) + \lambda + \tilde{g}(x) = 0 \quad (\text{D.7})$$

$$\Rightarrow p(x) = \exp \tilde{g}(x) \exp(\eta_1 T_1(x) + \eta_2 T_2(x) - 1 + \lambda) \quad (\text{D.8})$$

$$= \exp \tilde{g}(x) \exp(\eta_1 x + \eta_2 (x - \mu)^2 - 1 + \lambda) \quad (\text{D.9})$$

Apparently,  $p(x) > 0$  from the equation above. Because of the complementary slackness condition  $\tilde{g}(x) p(x) = 0$ ,  $\tilde{g}(x) = 0$  and so we simply have

$$p(x) = \exp(\eta_1 x + \eta_2 (x - \mu)^2 - 1 + \lambda) \quad (\text{D.10})$$

Finally,  $\eta_1, \eta_2$  and  $\lambda$  can be determined with the constraints  $E[X] = \mu$ ,  $E[(X - \mu)^2] = \sigma^2$ , and  $\int_x p(x) = 1$ . The only parameters to satisfy that will simply those of the Gaussian pdf. Thus,

$$p(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (\text{D.11})$$

Note that the choice of  $T$  is not unique, for example, we may pick  $T_1(x) = x$  and  $T_2(x) = x^2$ . Since

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{\underbrace{\sqrt{2\pi}}_{g(x)}} \exp\left(\underbrace{\frac{\mu}{\sigma^2}}_{\eta_1} x + \underbrace{\frac{-1}{2\sigma^2}}_{\eta_2} x^2 - \underbrace{\left(\frac{\mu^2}{2\sigma^2} + \log \sigma\right)}_A\right), \quad (\text{D.12})$$

we have  $\eta_1 = \frac{\mu}{\sigma^2}$ ,  $\eta_2 = -\frac{1}{2\sigma^2}$ ,  $g(x) = \frac{1}{\sqrt{2\pi}}$ , and  $A(\eta) = \frac{\mu^2}{2\sigma^2} + \log \sigma = -\frac{\eta_1^2}{4\eta_2} - \frac{1}{2} \log(-2\eta_2)$  in this case.

## D.2 Cumulant generating function

Recall the canonical form of the exponential family distribution is given by:

$$p(x) = g(x) \exp(\eta^\top T(x) - A(\eta)), \quad (\text{D.13})$$

where  $\eta$  is known as the natural parameter. An "unnatural" parametrization, denoted by  $\theta$ , can be represented as:

$$p(x) = g(x) \exp(\eta(\theta)^\top T(x) - A(\theta)). \quad (\text{D.14})$$

For the natural parametrization, the log-partition function coincides with the cumulant gener-

---

<sup>2</sup>We use  $\frac{\delta L}{\delta p}$  rather than  $\frac{\partial L}{\partial p}$  as  $L$  is really a functional (function of a function) when  $p(x)$  is a continuous function.

ating function. Specifically,

$$\frac{\partial A(\eta)}{\partial \eta_i} = \ln \sum_x g(x) \exp(\eta^\top T(x)) = \frac{\sum_x T_i(x) g(x) \exp(\eta^\top T(x))}{\sum_x g(x) \exp(\eta^\top T(x))} \quad (\text{D.15})$$

$$= \sum_x T_i(x) p(x) = E[T_i(X)]. \quad (\text{D.16})$$

Additionally, the following relationship can be easily verified:

$$\frac{\partial^2 A(\eta)}{\partial \eta_j \partial \eta_i} = E[T_i(X)T_j(X)] - E[T_i(X)]E[T_j(X)] = \text{Cov}[T_i(X), T_j(X)]. \quad (\text{D.17})$$

### D.3 Conjugate priors of exponential family distribution

Consider observations  $x_i \sim p(\cdot|\eta)$  for  $i = 1, \dots, n$  and the prior  $\eta \sim p(\cdot|\lambda)$ . The posterior distribution of  $\eta$  is given by:

$$p(\eta|x^n, \lambda) \propto p(\eta|\lambda) \prod_{i=1}^n p(x_i|\eta). \quad (\text{D.18})$$

Recall from Section 2.6.2 that if the prior  $p(\eta|\lambda)$  has the same form as the posterior, then we call  $p(\eta|\lambda)$  a conjugate prior of likelihood  $p(x|\eta)$ .

Any distribution from the exponential family has a conjugate prior, which also belongs to the exponential family. Specifically, consider the form:

$$p(x|\eta) = g(x) \exp(\eta^\top T(x) - A(\eta)). \quad (\text{D.19})$$

Its conjugate prior can be expressed as

$$p(\eta|\lambda) = \tilde{g}(\eta) \exp(\lambda_1^\top \eta - \lambda_2 A(\eta) - \tilde{A}(\lambda)) \quad (\text{D.20})$$

$$= \tilde{g}(\eta) \exp\left([\lambda_1^\top, \lambda_2] \begin{pmatrix} \eta \\ -A(\eta) \end{pmatrix} - \tilde{A}(\lambda)\right) \quad (\text{D.21})$$

$$= \tilde{g}(\eta) \exp\left(\lambda^\top \begin{pmatrix} \eta \\ -A(\eta) \end{pmatrix} - \tilde{A}(\lambda)\right) \quad (\text{D.22})$$

with  $\lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}$ . Equation (D.22) shows that the conjugate prior belongs to the exponential family as well. Moreover, the resulting posterior can be given by

$$p(\eta|x^n, \lambda) \propto p(\eta|\lambda) \prod_{i=1}^n p(x_i|\eta) \quad (\text{D.23})$$

$$= \left(\tilde{g}(\eta) \prod_{i=1}^n g(x_i)\right) \exp\left(\left(\lambda_1 + \sum_{i=1}^n T(x_i)\right)^\top \eta - (\lambda_2 + n)A(\eta) - \tilde{A}(\lambda)\right). \quad (\text{D.24})$$

This means

$$\begin{cases} \lambda_1 \leftarrow \lambda_1 + \sum_{i=1}^n T(x_i) \\ \lambda_2 \leftarrow \lambda_2 + n \end{cases} \quad (\text{D.25})$$

after observing  $x_1, \dots, x_n$ .

### D.3.1 Binomial distribution as an exponential family distribution

For a fixed  $n$ , the binomial distribution can be expressed in the exponential family form as:

$$f(x|\eta) = f(x|p) = \binom{n}{x} p^x (1-p)^{n-x} \quad (\text{D.26})$$

$$= \underbrace{\binom{n}{x}}_{g(x)} \exp \left( \underbrace{x \log \left( \frac{p}{1-p} \right)}_{\eta} - \underbrace{n \log \left( \frac{1}{1-p} \right)}_{A(\eta)} \right) \quad (\text{D.27})$$

Recall that the Beta distribution serves as a conjugate prior for the binomial distribution, and we can write:

$$f(p|\alpha, \beta) = \frac{p^{\alpha-1} (1-p)^{\beta-1}}{B(\alpha, \beta)} \quad (\text{D.28})$$

$$= \frac{1}{B(\alpha, \beta)} \exp((\alpha-1) \log(p) + (\beta-1) \log(1-p)) \quad (\text{D.29})$$

$$= \frac{1}{B(\alpha, \beta)} \exp \left( \underbrace{\alpha-1}_{\lambda_1} \log \left( \frac{p}{1-p} \right) - \underbrace{(\alpha+\beta-2)}_{\lambda_2} \underbrace{n \log \left( \frac{1}{1-p} \right)}_{A(\eta)} \right) \quad (\text{D.30})$$

From the earlier discussion in (D.25), we update the parameters as follows:  $\lambda_1 \leftarrow \lambda_1 + T(x) \Rightarrow \alpha \leftarrow \alpha + x$  and  $\lambda_2 \leftarrow \lambda_2 + 1 \Rightarrow \alpha + \beta \leftarrow \alpha + \beta + n$ . Combining the two, we have  $\alpha \rightarrow \alpha + x$  and  $\beta \rightarrow \beta + n - x$ , as discussed in Section 2.6.2.

**Remark 3.** Although there are  $n$  physical observations in the setup, we can consider that we only need to report once after the observations with the sufficient statistics  $k$ . Therefore  $\lambda_2$  is added by one above.

### D.3.2 Conjugate prior of unit variance Gaussian distribution

Considering the unit variance Gaussian with mean  $\eta$  and PDF

$$p(x|\eta) = \frac{1}{\sqrt{2\pi}} \exp(-(x - \eta)^2/2) \quad (\text{D.31})$$

$$= \underbrace{\frac{\exp(-x^2/2)}{\sqrt{2\pi}}}_{g(x)} \exp(\underbrace{\eta x}_{T(x)} - \underbrace{\eta^2/2}_{A(\eta)}). \quad (\text{D.32})$$

Thus, the conjugate prior should have the form

$$p(\eta|\lambda) = p(\eta|\lambda) = \tilde{g}(\eta) \exp(\lambda_1 \eta - \lambda_2 A(\eta) - \tilde{A}(\lambda_1, \lambda_2)) \quad (\text{D.33})$$

$$= \tilde{g}(\eta) \exp(\lambda_1 \eta - \lambda_2 \eta^2/2 - \tilde{A}(\lambda_1, \lambda_2)), \quad (\text{D.34})$$

which is just Gaussian distribution if we pick  $\tilde{g}(\eta) = 1$ . By inspection, we can write

$$p(\eta|\lambda) = \exp(\lambda_1 \eta - \lambda_2 \eta^2/2 - \tilde{A}(\lambda_1, \lambda_2)) \quad (\text{D.35})$$

$$= \exp\left(-\frac{\lambda_2}{2} \left(\eta - \frac{\lambda_1}{\lambda_2}\right)^2 + \frac{\lambda_1^2}{2\lambda_2} - \tilde{A}(\lambda_1, \lambda_2)\right) \quad (\text{D.36})$$

$$= \sqrt{\frac{\lambda_2}{2\pi}} \exp\left(-\frac{\lambda_2}{2} \left(\eta - \frac{\lambda_1}{\lambda_2}\right)^2\right) \quad (\text{D.37})$$

$$= \frac{1}{\sqrt{2\pi\sigma_\eta^2}} \exp\left(-\frac{(\eta - \mu_\eta)^2}{2\sigma_\eta^2}\right) \quad (\text{D.38})$$

with  $\mu_\eta = \frac{\lambda_1}{\lambda_2}$  and  $\sigma_\eta^2 = \frac{1}{\lambda_2}$ . This gives us  $\lambda_1 = \frac{\mu_\eta}{\sigma_\eta^2}$  and  $\lambda_2 = \frac{1}{\sigma_\eta^2}$ .

Now, given  $n$  observations  $x_1, x_2, \dots, x_n$ ,  $\lambda_1$  and  $\lambda_2$  will update to  $\lambda_1 + \sum_{i=1}^n T(x_i)$  and  $\lambda_2 + n$ . This gives us

$$\begin{cases} \mu_\eta \leftarrow \frac{\lambda_1 + \sum_{i=1}^n T(x_i)}{\lambda_2 + n} = \frac{\frac{\mu_\eta}{\sigma_\eta^2} + \sum_{i=1}^n x_i}{\frac{1}{\sigma_\eta^2} + n} \\ \sigma_\eta^2 \leftarrow \frac{1}{\lambda_2 + n} = \frac{1}{\frac{1}{\sigma_\eta^2} + n} \end{cases} \quad (\text{D.39})$$

**Remark 4.** *It's crucial not to confuse  $\sigma_\eta^2$ , the variance of the prior on the mean, with the variance of the observation, which is set to 1. Although  $\mu$  of the observation is a random variable with mean  $\mu_\eta = \lambda_1/\lambda_2$ , the variance  $\sigma_\eta^2$  decreases as more observations are made, as expected.*

**Remark 5.** *Considering precision (inverse variance) rather than variance, the update equations presented in (D.39) become more intuitive. Specifically, the updated precision is the sum of the prior precision  $1/\sigma_\eta^2$  and the total precision of the observations,  $n$ . The updated mean, then, is a weighted average of the prior mean and the sample mean, with weights proportional to their respective precisions.*

**D.4 Exercise**

1. Verify that  $\frac{\partial^2 A(\eta)}{\partial \eta_j \partial \eta_i} = Cov[T_i(X), T_j(X)]$  as shown in (??).



## Bibliography

- [1] I.H. Witten, R.M. Neal, and J.G. Cleary. Arithmetic coding for data compression. *Comm ACM*, 30(6):520–540, June 1987.





# Index

**BP** belief propagation 103, 108, 110, 112–114, 116–120, 122, 126, 129, 132

**Gaussian BP** Gaussian belief propagation 103, 127, 128, 130, 132

**covariance** 124, 125, 127, 130, 131

**Gaussian distribution** 123–126, 130

**multivariate Gaussian** 5, 124, 128

**canonical form** 124–127, 130, 132

**moment form** 124–127, 130

**graphical model** 5, 103, 106–108, 110, 112

**BN** Bayesian network 103–107, 109–112, 114

**moralization** 110–112

**factor graph** 108, 110, 112, 114–116, 120, 129

**Markov equivalence** 104, 105

**undirected graph** 108–112, 114, 115

**Kalman filter** 103, 124, 126, 129, 130

**KL-divergence** Kullback-Leiber divergence 4, 64, 65, 67, 68, 73, 74, 119