Probabilistic Machine Learning: Advanced Topics

Adaptive Computation and Machine Learning
Francis Bach, editor

Bioinformatics: The Machine Learning Approach, Pierre Baldi and Søren Brunak, 1998
Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto, 1998
Graphical Models for Machine Learning and Digital Communication, Brendan J. Frey, 1998
Learning in Graphical Models, edit by Michael I. Jordan, 1999
Causation, Prediction, and Search, second edition, Peter Spirtes, Clark Glymour, and Richard Scheines, 2000
Principles of Data Mining, David Hand, Heikki Mannila, and Padhraic Smyth, 2000
Bioinformatics: The Machine Learning Approach, second edition, Pierre Baldi and Søren Brunak, 2001
Learning Kernel Classifiers: Theory and Algorithms, Ralf Herbrich, 2002
Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Bernhard Schölkopf and Alexander J. Smola, 2002
Introduction to Machine Learning, Ethem Alpaydin, 2004
Gaussian Processes for Machine Learning, Carl Edward Rasmussen and Christopher K.I. Williams, 2006
Semi-Supervised Learning, edited by Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, 2006
The Minimum Description Length Principle, Peter D. Grünwald, 2007
Introduction to Statistical Relational Learning, edited by Lise Getoor and Ben Taskar, 2007
Probabilistic Graphical Models: Principles and Techniques, Daphne Koller and Nir Friedman, 2009
Introduction to Machine Learning, second edition, Ethem Alpaydin, 2010
Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation, Masashi Sugiyama and Motoaki Kawanabe, 2012
Boosting: Foundations and Algorithms, Robert E. Schapire and Yoav Freund, 2012
Foundations of Machine Learning, Mehryar Mohri, Afshin Rostami, and Ameet Talwalker, 2012
Machine Learning: A Probabilistic Perspective, Kevin P. Murphy, 2012
Introduction to Machine Learning, third edition, Ethem Alpaydın, 2014
Deep Learning, Ian Goodfellow, Yoshua Bengio, and Aaron Courville, 2017
Elements of Causal Inference: Foundations and Learning Algorithms, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf, 2017
Machine Learning for Data Streams, with Practical Examples in MOA, Albert Bifet, Ricard Gavaldà, Geoffrey Holmes, Bernhard Pfahringer, 2018
Reinforcement Learning: An Introduction, second edition, Richard S. Sutton and Andrew G. Barto, 2018
Foundations of Machine Learning, second edition, Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalker, 2019
Introduction to Natural Language Processing, Jacob Eisenstein, 2019
Introduction to Machine Learning, fourth edition, Ethem Alpaydın, 2020
Knowledge Graphs: Fundamentals, Techniques, and Applications, Mayank Kejriwal, Craig A. Knoblock, and Pedro Szekely, 2021
Probabilistic Machine Learning: An Introduction, Kevin P. Murphy, 2022

Machine Learning from Weak Supervision: An Empirical Risk Minimization Approach, Masashi Sugiyama, Han Bao, Takashi Ishida, Nan Lu, Tomoya Sakai, and Gang Niu, 2022
Introduction to Online Convex Optimization, second edition, Elad Hazan, 2022
Distributional Reinforcement Learning, Marc G. Bellemare, Will Dabney, and Mark Rowland, 2023
Probabilistic Machine Learning: Advanced Topics, Kevin P. Murphy, 2023

Probabilistic Machine Learning:
Advanced Topics

Kevin P. Murphy

The MIT Press
Cambridge, Massachusetts
London, England

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in _____ by _____.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data is available.

ISBN:

10 9 8 7 6 5 4 3 2 1

This book is dedicated to my wife Margaret,
who has been the love of my life for 20+ years.

Brief Contents

Contents

VI Action 1089

Preface

> I am writing a longer [book] than usual because there is not enough time to write a short one. (Blaise Pascal, paraphrased.)

This book is a sequel to [Mur22]. That book mostly focused on techniques for learning functions $f : \mathcal{X} \to \mathcal{Y}$, where f is some nonlinear model, such as a deep neural network, $\mathcal{X}$ is the set of possible inputs (typically $\mathcal{X} = \mathbb{R}^D$), and $\mathcal{Y} = \{1, \ldots, C\}$ represents the set of labels for classification problems or $\mathcal{Y} = \mathbb{R}$ for regression problems. Judea Pearl, a well known AI researcher, has called this kind of ML a form of "glorified curve fitting" (quoted in [Har18]).

In this book, we expand the scope of ML to encompass more challenging problems. For example, we consider training and testing under different distributions; we consider generation of high dimensional outputs, such as images, text and graphs, so the output space is, say, $\mathcal{Y} = \mathbb{R}^{256 \times 256}$; we discuss methods for discovering "insights" about data, based on latent variable models; and we discuss how to use probabilistic models for causal inference and decision making under uncertainty.

We assume the reader has some prior exposure to ML and other relevant mathematical topics (e.g., probability, statistics, linear algebra, optimization). This background material is covered in the prequel to this book, [Mur22], amongst other sources (e.g., [Lin+21b; DFO20]).

Python code (mostly in JAX) to reproduce nearly all of the figures can be found online. In particular, if a figure caption says "Generated by `gauss_plot_2d.ipynb`", then you can find the corresponding Jupyter notebook at probml.github.io/notebooks#gauss_plot_2d.ipynb. Clicking on the figure link in the pdf version of the book will take you to this list of notebooks. Clicking on the notebook link will open it inside Google Colab, which will let you easily reproduce the figure for yourself, and modify the underlying source code to gain a deeper understanding of the methods. (Colab gives you access to a free GPU, which is useful for some of the more computationally heavy demos.)

In addition to the online code, at probml.github.io/supp there is some additional supplementary online content. This contains additional material which was excluded from the main book for space reasons. For exercises (and solutions) related to the topics in this book, see [Gut22].

Contributing authors

This book is the result of a lot of effort from a lot of people. I would especially like to thank the following people who wrote or co-wrote various sections or chapters:

- Alex Alemi (Google), who co-wrote Section 5.1 (KL divergence) (with Murphy).
- Jeff Bilmes (U. Washington), who wrote Section 6.11 (Submodular optimization).
- Marco Cuturi (Apple, work done at Google), who wrote Section 6.10 (Optimal Transport).
- Alexander D'Amour (Google), who co-wrote Chapter 36 (Causality) (with Veitch).
- Finale Doshi-Velez (Harvard), who co-wrote Chapter 33 (Interpretability) (with Kim).
- Roy Frostig (Google), who wrote Section 6.2 (Automatic differentiation).
- Justin Gilmer (Google), who wrote Section 19.8 (Adversarial examples).
- Giles Harper-Donnelly, who wrote Section 8.3.4 (Information form filtering and smoothing).
- Been Kim (Google), who co-wrote Chapter 33 (Interpretability) (with Doshi-Velez).
- Durk Kingma (Google), who co-wrote Chapter 24 (Energy-based models) (with Song).
- Simon Kornblith (Google), who co-wrote Chapter 32 (Representation learning) (with Poole).
- Balaji Lakshminarayanan (Google), who co-wrote Chapter 23 (Normalizing Flows) (with Papamakarios) and Chapter 26 (Generative adversarial networks) (with Mohamed and Rosca).
- Lihong Li (Amazon, work done at Google), who co-wrote Section 34.4 (Contextual bandits) and Chapter 35 (Reinforcement learning) (with Murphy).
- Xinglong Li (UBC), who wrote Section 15.2.8 (Multivariate linear regression), Section 29.4.4.1 (Blocked Gibbs sampling for HMMs), Section 29.8.4.1 (Blocked Gibbs sampling for LDS) and Section 31.3.3 (Fitting using variational Bayes).
- Shakir Mohamed (Deepmind), who co-wrote Chapter 26 (Generative adversarial networks) (with Lakshminarayanan and Rosca).
- George Papamakarios (Deepmind), who co-wrote Chapter 23 (Normalizing Flows) (with Lakshminarayanan).
- Zeel B Patel (IIT Gandhinagar), who co-wrote Section 34.7 (Active learning) (with Murphy).
- Ben Poole (Google), who co-wrote Chapter 32 (Representation learning) (with Kornblith).
- Mihaela Rosca (Deepmind / UCL), who co-wrote Chapter 26 (Generative adversarial networks).
- Vinayak Rao (Purdue), who wrote Chapter 31 (Non-parametric Bayesian models).
- Yang Song (Stanford), who co-wrote Chapter 24 (Energy-based models) (with Kingma).
- Victor Veitch (Google / U. Chicago), who co-wrote Chapter 36 (Causality) (with D'Amour).
- Andrew Wilson (NYU), who co-wrote Chapter 17 (Bayesian neural networks) and Chapter 18 (Gaussian processes) (with Murphy).

Other contributors

I would like to thank the following people who helped in various other ways:

- Many people who helped make or improve the figures, including: Vibhuti Bansal, Shobhit Belwal, Aadesh Desai, Vishal Ghoniya, Anand Hegde, Ankita Kumari Jain, Madhav Kanda, Aleyna Kara, Rohit Khoiwal, Taksh Panchal, Zeel B Patel, Karm Patel, Dhruv Patel, Prey Patel, Nitish Sharma, Hetvi Shastri, Mahmoud Soliman, Gautam Vashishtha.
- Participants in the Google Summer of Code (GSOC) for 2021, including Ming Liang Ang, Aleyna Kara, Gerardo Duran-Martin, Srikar Reddy Jilugu, Drishti Patel, and co-mentor Mahmoud Soliman.
- Participants in the Google Summer of Code (GSOC) for 2022, including Peter Chang, Giles Harper-Donnelly, Xinglong Li, Zeel B Patel, Karm Patel, Qingyao Sun, and co-mentors Nipun Batra and Scott Linderman.

- Many other people who contributed code (see auto-generated list at https://github.com/probml/pyprobml#acknowledgements).
- Many people who proofread parts of the book, including: Aalto Seminar students, Bill Behrman, Kay Brodersen, Peter Chang, Krzysztof Choromanski, Adrien Corenflos, Tom Dietterich, Gerardo Duran-Martin, John Fearns (who proof-read almost the entire book), Lehman Krunoslav, Amir Globerson, Giles Harper-Donnelly, Ravin Kumar, Junpeng Lao, Stephen Mandt, Simon Prince, Rif Saurous, Erik Sudderth, Hal Varian, Chris Williams, Raymond Yeh, and others listed at https://github.com/probml/pml2-book/issues?q=is:issue.

About the cover

The cover illustrates a variational autoencoder (Chapter 21) being used to map from a 2d Gaussian to image space.

Kevin Patrick Murphy
Palo Alto, California
July 2022.

1 Introduction

> "Intelligence is not just about pattern recognition and function approximation. It's about modeling the world". — Josh Tenenbaum, NeurIPS 2021.

Much of current machine learning focuses on the task of mapping inputs to outputs (i.e., approximating functions of the form $f : \mathcal{X} \to \mathcal{Y}$), often using "**deep learning**" (see e.g., [LBH15; Sch14; Sej20]). Judea Pearl, a well known AI researcher, has called this "glorified curve fitting" (quoted in [Har18]). This is a little unfair, since when $\mathcal{X}$ and/or $\mathcal{Y}$ are high-dimensional spaces — such as images, sentences, graphs, or sequences of decisions/actions — then the term "curve fitting" is rather misleading, since one-dimensional intuitions often do not work in higher dimensional settings (see e.g., [BPL21a]). Nevertheless, the quote gets at what many feel is lacking in current attempts to "solve AI" using machine learning techniques, namely that they are too focused on prediction of observable patterns, and not focused enough on "understanding" the underlying *latent structure* behind these patterns.

Gaining a "deep understanding" of the structure behind the observed data is necessary for advancing science, as well as for certain applications, such as healthcare (see e.g., [DD22]), where identifying the *root causes* or mechanisms behind various diseases is the key to developing cures. In addition, such "deep understanding" is necessary in order to develop *robust* and *efficient* systems. By "robust" we mean methods that work well even if there are unexpected changes to the data distribution to which the system is applied, which is an important concern in many areas, such as robotics (see e.g., [Roy+21]). By "efficient" we generally mean data or statistically efficient i.e., methods that can learn quickly from small amounts of data (c.f., [Lu+21b]). This is important since data can be limited in some domains, such as healthcare and robotics, even though it is abundant in other domains, such as language and vision, due to the ability to scrape the internet. We are also interested in computationally efficient methods, although this is a secondary concern as computing power continues to grow. (We also note that this trend has been instrumental to much of the recent progress in AI, as noted in [Sut19].)

To develop robust and efficient systems, this book adopts a model-based approach, in which we try to learn *parsimonious representations* of the underlying "**data generating process**" (**DGP**) given samples from one or more datasets (c.f., [Lak+17; Win+19; Sch20; Ben+21a; Cun22; MTS22]). This is in fact similar to the scientific method, where we try to explain (features of) the observations by developing theories or models. One way to formalize this process is in terms of **Bayesian inference** applied to probabilistic models, as argued in [Jay03; Box80; GS13]. We discuss inference algorithms in detail in Part II of the book.[1] But before we get there, in Part I we cover some relevant background

1. Note that, in the deep learning community, the term "inference" means applying a function to some inputs to

material that will be needed. (This part can be skipped by readers who are already familiar with these basics.)

Once we have a set of inference methods in our toolbox (some of which may be as simple as computing a maximum likelihood estimate using an optimization method, such as stochastic gradient descent) we can turn our focus to discussing different kinds of models. The choice of model depends on our task, the kind and amount of data we have, and our metric(s) of success. We will broadly consider four main kinds of task: prediction (e.g., classification and regression), generation (e.g., of images or text), discovery (of "meaningful structure" in data), and control (optimal decision making). We give more details below.

In Part III, we discuss models for prediction. These models are conditional distributions of the form $p(\boldsymbol{y}|\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$ is some input (often high dimensional), and $\boldsymbol{y} \in \mathcal{Y}$ is the desired output (often low dimensional). In this part of the book, we assume there is one right answer that we want to predict, although we may be uncertain about it.

In Part IV, we discuss models for generation. These models are distributions of the form $p(\boldsymbol{x})$ or $p(\boldsymbol{x}|\boldsymbol{c})$, where $\boldsymbol{c}$ are optional conditioning inputs, and where there may be multiple valid outputs. For example, given a text prompt $\boldsymbol{c}$, we may want to generate a diverse set of images $\boldsymbol{x}$ that "match" the caption. Evaluating such models is harder than in the prediction setting, since it is less clear what the desired output should be.

In Part V, we discuss latent variable models, which are joint models of the form $p(\boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$, where $\boldsymbol{z}$ is the hidden state and $\boldsymbol{x}$ are the observations that are assumed to be generated from $\boldsymbol{z}$. The goal is to compute $p(\boldsymbol{z}|\boldsymbol{x})$, in order to uncover some (hopefully meaningful / useful) underlying state or patterns in the observed data. We also consider methods for trying to discover patterns learned implicitly by predictive models of the form $p(\boldsymbol{y}|\boldsymbol{x})$, without relying on an explicit generative model of the data.

Finally, in Part VI, we discuss models and algorithms which can be used to make decisions under uncertainty. This naturally leads into the very important topic of causality, with which we close the book.

In view of the broad scope of the book, we cannot go into detail on every topic. However, we have attempted to cover all the basics. In some cases, we also provide a "deeper dive" into the research frontier (as of 2022). We hope that by bringing all these topics together, you will find it easier to make connections between all these seemingly disparate areas, and can thereby deepen your understanding of the field of machine learning.

compute the output. This is unrelated to Bayesian inference, which is concerned with the much harder task of inverting a function, and working backwards from observed outputs to possible hidden inputs (causes). The latter is more closely related to what the deep learning community calls "training".

Part I

Fundamentals

2 Probability

> The mathematical rules of probability theory are not merely rules for calculating frequencies of 'random variables'; they are also the unique rules for conducting inference (i.e. plausible reasoning) of any kind. — E .T. Jaynes [Jay03].

2.1 Introduction

We assume the reader is already familiar with basic probability theory. For example, see [Cha21], or chapter 2 of the prequel to this book, [Mur22]. In this chapter, we briefly review some of this material, to make the book self-contained.

2.2 Some common probability distributions

There are a wide variety of probability distributions that are used for various kinds of models. We summarize some of the more commonly used ones in the sections below. See Supplementary Chapter 2 for more information, and https://ben18785.shinyapps.io/distribution-zoo/ for an interactive visualization.

2.2.1 Discrete distributions

In this section, we discuss some discrete distributions defined on subsets of the (non-negative) integers.

2.2.1.1 Bernoulli and binomial distributions

Let $x \in \{0, 1, \ldots, N\}$. The **binomial distribution** is defined by

$$\text{Bin}(x|N, \mu) \triangleq \binom{N}{x} \mu^x (1-\mu)^{N-x} \tag{2.1}$$

where $\binom{N}{k} \triangleq \frac{N!}{(N-k)!k!}$ is the number of ways to choose k items from N (this is known as the **binomial coefficient**, and is pronounced "N choose k").

If $N = 1$, so $x \in \{0, 1\}$, the binomial distribution reduces to the **Bernoulli distribution**:

$$\text{Ber}(x|\mu) = \begin{cases} 1-\mu & \text{if } x = 0 \\ \mu & \text{if } x = 1 \end{cases} \tag{2.2}$$

where $\mu = \mathbb{E}[x] = p(x = 1)$ is the mean.

2.2.1.2 Categorical and multinomial distributions

If the variable is discrete-valued, $x \in \{1, \ldots, K\}$, we can use the **categorical** distribution:

$$\text{Cat}(x|\boldsymbol{\theta}) \triangleq \prod_{k=1}^{K} \theta_k^{\mathbb{I}(x=k)} \tag{2.3}$$

Alternatively, we can represent the K-valued variable x with the one-hot binary vector $\boldsymbol{x}$, which lets us write

$$\text{Cat}(\boldsymbol{x}|\boldsymbol{\theta}) \triangleq \prod_{k=1}^{K} \theta_k^{x_k} \tag{2.4}$$

If the k'th element of $\boldsymbol{x}$ counts the number of times the value k is seen in $N = \sum_{k=1}^{K} x_k$ trials, then we get the **multinomial distribution**:

$$\mathcal{M}(\boldsymbol{x}|N, \boldsymbol{\theta}) \triangleq \binom{N}{x_1 \ldots x_K} \prod_{k=1}^{K} \theta_k^{x_k} \tag{2.5}$$

where the **multinomial coefficient** is defined as

$$\binom{N}{k_1 \ldots k_m} \triangleq \frac{N!}{k_1! \ldots k_m!} \tag{2.6}$$

2.2.1.3 Poisson distribution

Suppose $X \in \{0, 1, 2, \ldots\}$. We say that a random variable has a **Poisson** distribution with parameter $\lambda > 0$, written $X \sim \text{Poi}(\lambda)$, if its pmf (probability mass function) is

$$\text{Poi}(x|\lambda) = e^{-\lambda} \frac{\lambda^x}{x!} \tag{2.7}$$

where λ is the mean (and variance) of x.

2.2.1.4 Negative binomial distribution

Suppose we have an "urn" with N balls, R of which are red and B of which are blue. Suppose we perform **sampling with replacement** until we get $n \geq 1$ balls. Let X be the number of these that are blue. It can be shown that $X \sim \text{Bin}(n, p)$, where $p = B/N$ is the fraction of blue balls; thus X follows the binomial distribution, discussed in Section 2.2.1.1.

Now suppose we consider drawing a red ball a "failure", and drawing a blue ball a "success". Suppose we keep drawing balls until we observe r failures. Let X be the resulting number of successes (blue balls); it can be shown that $X \sim \text{NegBinom}(r, p)$, which is the **negative binomial distribution** defined by

$$\text{NegBinom}(x|r, p) \triangleq \binom{x + r - 1}{x} (1 - p)^r p^x \tag{2.8}$$

for $x \in \{0, 1, 2, \ldots\}$. (If r is real-valued, we replace $\binom{x+r-1}{x}$ with $\frac{\Gamma(x+r)}{x!\Gamma(r)}$, exploiting the fact that $(x-1)! = \Gamma(x)$.)

This distribution has the following moments:

$$\mathbb{E}[x] = \frac{p\,r}{1-p}, \; \mathbb{V}[x] = \frac{p\,r}{(1-p)^2} \tag{2.9}$$

This two parameter family has more modeling more flexibility than the Poisson distribution, since it can represent the mean and variance separately. This is useful e.g., for modeling "contagious" events, which have positively correlated occurrences, causing a larger variance than if the occurrences were independent. In fact, The Poisson distribution is a special case of the negative binomial, since it can be shown that $\text{Poi}(\lambda) = \lim_{r\to\infty} \text{NegBinom}(r, \frac{\lambda}{1+\lambda})$. Another special case is when $r = 1$; this is called the **geometric distribution**.

2.2.2 Continuous distributions on $\mathbb{R}$

In this section, we discuss some univariate distributions defined on the reals, $p(x)$ for $x \in \mathbb{R}$.

2.2.2.1 Gaussian (Normal)

The most widely used univariate distribution is the **Gaussian distribution**, also called the **normal distribution**. (See [Mur22, Sec 2.6.4] for a discussion of these names.) The pdf (probability density function) of the Gaussian is given by

$$\mathcal{N}(x|\mu, \sigma^2) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{2.10}$$

where $\sqrt{2\pi\sigma^2}$ is the normalization constant needed to ensure the density integrates to 1. The parameter μ encodes the mean of the distribution, which is the same as the mode, since the distribution is unimodal. The parameter σ^2 encodes the variance. Sometimes we talk about the **precision** of a Gaussian, by which we mean the inverse variance: $\lambda = 1/\sigma^2$. A high precision means a narrow distribution (low variance) centered on μ.

The cumulative distribution function or cdf of the Gaussian is defined as

$$\Phi(x; \mu, \sigma^2) \triangleq \int_{-\infty}^{x} \mathcal{N}(z|\mu, \sigma^2) dz \tag{2.11}$$

If $\mu = 0$ and $\sigma = 1$ (known as the **standard Normal** distribution), we just write $\Phi(x)$.

2.2.2.2 Half-normal

For some problems, we want a distribution over non-negative reals. One way to create such a distribution is to define $Y = |X|$, where $X \sim \mathcal{N}(0, \sigma^2)$. The induced distribution for Y is called the **half-normal distribution**, which has the pdf

$$\mathcal{N}_+(y|\sigma) \triangleq 2\mathcal{N}(y|0, \sigma^2) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{y^2}{2\sigma^2}\right) \quad y \geq 0 \tag{2.12}$$

This can be thought of as the $\mathcal{N}(0, \sigma^2)$ distribution "folded over" onto itself.

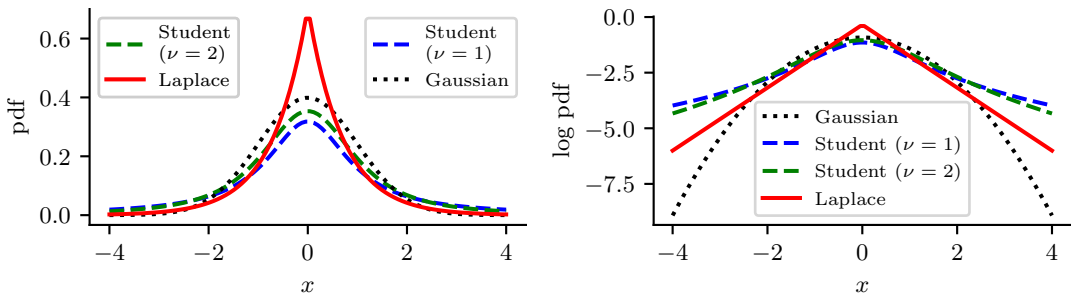


Figure 2.1: (a) The pdf's for a $\mathcal{N}(0,1)$, $\mathcal{T}_1(0,1)$ and $\text{Laplace}(0, 1/\sqrt{2})$. The mean is 0 and the variance is 1 for both the Gaussian and Laplace. The mean and variance of the Student distribution is undefined when $\nu = 1$. (b) Log of these pdf's. Note that the Student distribution is not log-concave for any parameter value, unlike the Laplace distribution. Nevertheless, both are unimodal. Generated by student_laplace_pdf_plot.ipynb.

2.2.2.3 Student t distribution

One problem with the Gaussian distribution is that it is sensitive to outliers, since the probability decays exponentially fast with the (squared) distance from the center. A more robust distribution is the **Student t-distribution**, which we shall call the **Student distribution** for short. Its pdf is as follows:

$$\mathcal{T}_\nu(x|\mu,\sigma^2) = \frac{1}{Z}\left[1 + \frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right]^{-(\frac{\nu+1}{2})} \tag{2.13}$$

$$Z = \frac{\sqrt{\nu\pi\sigma^2}\Gamma(\frac{\nu}{2})}{\Gamma(\frac{\nu+1}{2})} = \sqrt{\nu}\sigma B(\frac{1}{2},\frac{\nu}{2}) \tag{2.14}$$

where μ is the mean, $\sigma > 0$ is the scale parameter (not the standard deviation), and $\nu > 0$ is called the **degrees of freedom** (although a better term would be the **degree of normality** [Kru13], since large values of ν make the distribution act like a Gaussian). Here $\Gamma(a)$ is the **gamma function** defined by

$$\Gamma(a) \triangleq \int_0^\infty x^{a-1}e^{-x}dx \tag{2.15}$$

and $B(a,b)$ is the **beta function**, defined by

$$B(a,b) \triangleq \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{2.16}$$

2.2.2.4 Cauchy distribution

If $\nu = 1$, the Student distribution is known as the **Cauchy** or **Lorentz** distribution. Its pdf is defined by

$$\mathcal{C}(x|\mu,\gamma) = \frac{1}{Z}\left[1 + \left(\frac{x-\mu}{\gamma}\right)^2\right]^{-1} \tag{2.17}$$

where $Z = \gamma\beta(\frac{1}{2},\frac{1}{2}) = \gamma\pi$. This distribution is notable for having such heavy tails that the integral that defines the mean does not converge.

The **half Cauchy** distribution is a version of the Cauchy (with mean 0) that is "folded over" on itself, so all its probability density is on the positive reals. Thus it has the form

$$\mathcal{C}_+(x|\gamma) \triangleq \frac{2}{\pi\gamma}\left[1 + \left(\frac{x}{\gamma}\right)^2\right]^{-1} \tag{2.18}$$

2.2.2.5 Laplace distribution

Another distribution with heavy tails is the **Laplace distribution**, also known as the **double sided exponential** distribution. This has the following pdf:

$$\text{Laplace}(x|\mu,b) \triangleq \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right) \tag{2.19}$$

Here μ is a location parameter and $b > 0$ is a scale parameter. See Figure 2.1 for a plot.

2.2.2.6 Sub-Gaussian and super-Gaussian distributions

There are two main variants of the Gaussian distribution, known as **super-Gaussian** or **leptokurtic** ("Lepto" is Greek for "narrow") and **sub-Gaussian** or **platykurtic** ("Platy" is Greek for "broad"). These distributions differ in terms of their **kurtosis**, which is a measure of how heavy or light their tails are (i.e., how fast the density dies off to zero away from its mean). More precisely, the kurtosis is defined as

$$\text{kurt}(z) \triangleq \frac{\mu_4}{\sigma^4} = \frac{\mathbb{E}\left[(Z-\mu)^4\right]}{\left(\mathbb{E}\left[(Z-\mu)^2\right]\right)^2} \tag{2.20}$$

where σ is the standard deviation, and μ_4 is the 4'th **central moment**. (Thus $\mu_1 = \mu$ is the mean, and $\mu_2 = \sigma^2$ is the variance.) For a standard Gaussian, the kurtosis is 3, so some authors define the **excess kurtosis** as the kurtosis minus 3.

A super-Gaussian distribution (e.g., the Laplace) has positive excess kurtosis, and hence heavier tails than the Gaussian. A sub-Gaussian distribution, such as the uniform, has negative excess kurtosis, and hence lighter tails than the Gaussian. See Figure 2.2 for an illustration.

2.2.3 Continuous distributions on $\mathbb{R}^+$

In this section, we discuss some univariate distributions defined on the positive reals, $p(x)$ for $x \in \mathbb{R}^+$.

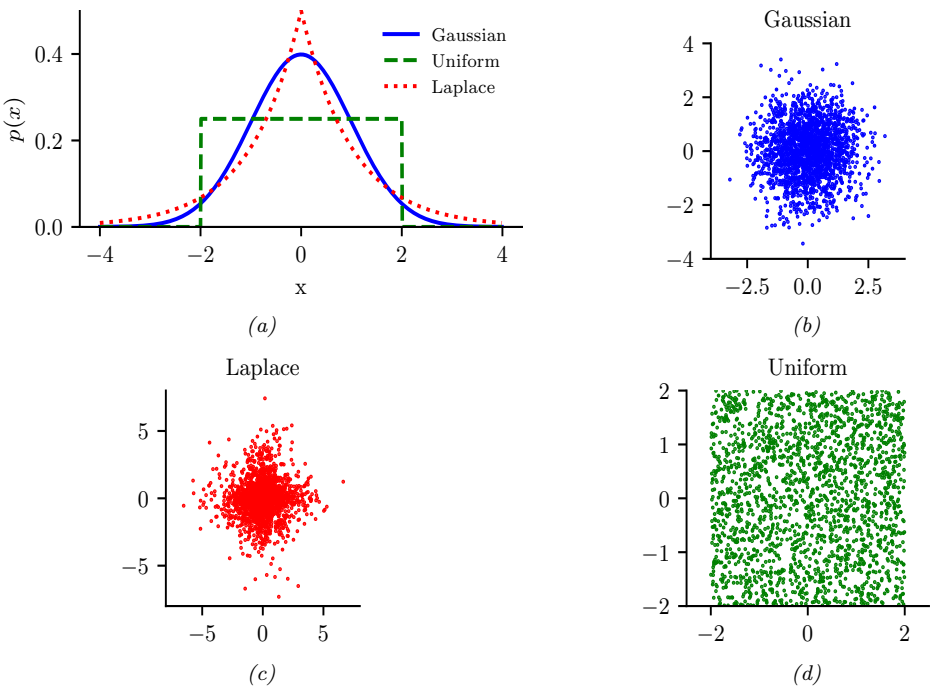


Figure 2.2: Illustration of Gaussian (blue), sub-Gaussian (uniform, green) and super-Gaussian (Laplace, red) distributions in 1d and 2d. Generated by sub_super_gauss_plot.ipynb.

2.2.3.1 Gamma distribution

The **gamma distribution** is a flexible distribution for positive real valued rv's, $x > 0$. It is defined in terms of two parameters, called the shape $a > 0$ and the rate $b > 0$:

$$\text{Ga}(x|\text{shape} = a, \text{rate} = b) \triangleq \frac{b^a}{\Gamma(a)} x^{a-1} e^{-xb} \tag{2.21}$$

Sometimes the distribution is parameterized in terms of the rate a and the **scale** $s = 1/b$:

$$\text{Ga}(x|\text{shape} = a, \text{scale} = s) \triangleq \frac{1}{s^a \Gamma(a)} x^{a-1} e^{-x/s} \tag{2.22}$$

2.2.3.2 Exponential distribution

The **exponential distribution** is a special case of the gamma distribution and is defined by

$$\text{Expon}(x|\lambda) \triangleq \text{Ga}(x|\text{shape} = 1, \text{rate} = \lambda) \tag{2.23}$$

This distribution describes the times between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate λ.

2.2.3.3 Chi-squared distribution

The **Chi-squared distribution** is a special case of the gamma distribution and is defined by

$$\chi^2_\nu(x) \triangleq \text{Ga}(x|\text{shape} = \frac{\nu}{2}, \text{rate} = \frac{1}{2}) \tag{2.24}$$

where ν is called the degrees of freedom. This is the distribution of the sum of squared Gaussian random variables. More precisely, if $Z_i \sim \mathcal{N}(0,1)$, and $S = \sum_{i=1}^{\nu} Z_i^2$, then $S \sim \chi^2_\nu$. Hence if $X \sim \mathcal{N}(0, \sigma^2)$ then $X^2 \sim \sigma^2 \chi^2_1$. Since $\mathbb{E}\left[\chi^2_1\right] = 1$ and $\mathbb{V}\left[\chi^2_1\right] = 2$, we have

$$\mathbb{E}\left[X^2\right] = \sigma^2, \mathbb{V}\left[X^2\right] = 2\sigma^4 \tag{2.25}$$

2.2.3.4 Inverse gamma

The **inverse Gamma distribution**, denoted $Y \sim \text{IG}(a, b)$, is the distribution of $Y = 1/X$ assuming $X \sim \text{Ga}(a, b)$. This pdf is defined by

$$\text{IG}(x|\text{shape} = a, \text{scale} = b) \triangleq \frac{b^a}{\Gamma(a)} x^{-(a+1)} e^{-b/x} \tag{2.26}$$

The mean only exists if $a > 1$. The variance only exists if $a > 2$.

The **scaled inverse chi-squared** distribution is a reparameterization of the inverse Gamma distribution:

$$\chi^{-2}(x|\nu, \sigma^2) = \text{IG}(x|\text{shape} = \frac{\nu}{2}, \text{scale} = \frac{\nu\sigma^2}{2}) \tag{2.27}$$

$$= \frac{1}{\Gamma(\nu/2)} \left(\frac{\nu\sigma^2}{2}\right)^{\nu/2} x^{-\frac{\nu}{2}-1} \exp\left(-\frac{\nu\sigma^2}{2x}\right) \tag{2.28}$$

The regular inverse chi-squared distribution, written $\chi^{-2}_\nu(x)$, is the special case where $\nu\sigma^2 = 1$ (i.e., $\sigma^2 = 1/\nu$). This corresponds to $\text{IG}(x|\text{shape} = \nu/2, \text{scale} = \frac{1}{2})$.

2.2.3.5 Pareto distribution

The **Pareto distribution** has the following pdf:

$$\text{Pareto}(x|m, \kappa) = \kappa m^\kappa \frac{1}{x^{(\kappa+1)}} \mathbb{I}(x \geq m) \tag{2.29}$$

See Figure 2.3(a) for some plots. We see that x must be greater than the minimum value m, but then rapidly decays after that. If we plot the distribution on a log-log scale, it forms the straight line $\log p(x) = -a \log x + \log(c)$, where $a = (\kappa + 1)$ and $c = \kappa m^\kappa$: see Figure 2.3(b) for an illustration.

When $m = 0$, the distribution has the form $p(x) = \kappa x^{-a}$. This is known as a **power law**. If $a = 1$, the distribution has the form $p(x) \propto 1/x$; if we interpret x as a frequency, this is called a $1/f$ function.

The Pareto distribution is useful for modeling the distribution of quantities that exhibit **heavy tails** or **long tails**, in which most values are small, but there are a few very large values. Many forms of data exhibit this property. ([ACL16] argue that this is because many datasets are generated by a variety of latent factors, which, when mixed together, naturally result in heavy tailed distributions.) We give some examples below.

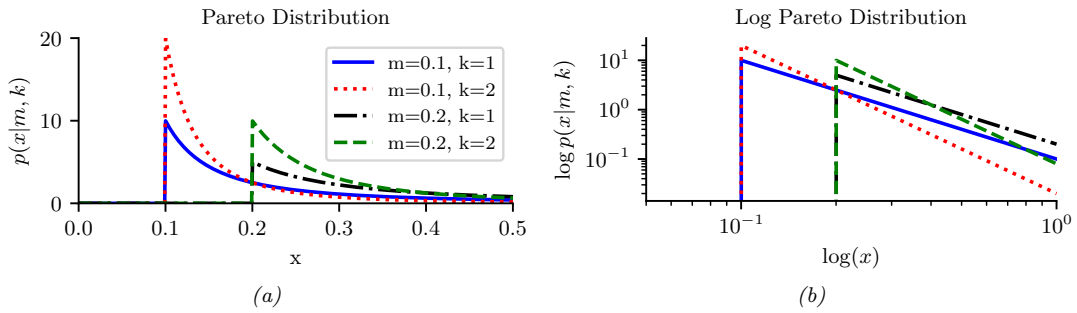


Figure 2.3: (a) The Pareto pdf Pareto$(x|k,m)$. (b) Same distribution on a log-log plot. Generated by pareto_dist_plot.ipynb.

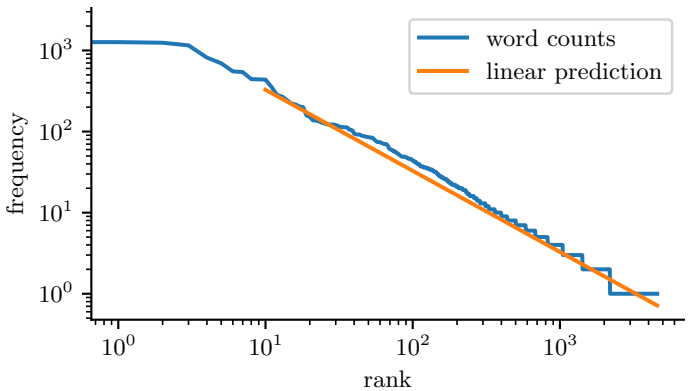


Figure 2.4: A log-log plot of the frequency vs the rank for the words in H. G. Wells' The Time Machine. *Generated by zipfs_law_plot.ipynb. Adapted from a figure from [Zha+20a, Sec 8.3].*

Modeling wealth distributions

The Pareto distribution is named after the Italian economist and sociologist Vilfredo Pareto. He created it in order to model the distribution of wealth across different countries. Indeed, in economics, the parameter κ is called the **pareto index**. If we set $\kappa = 1.16$, we recover the **80-20 rule**, which states that 80% of the wealth of a society is held by 20% of the population.[1]

Zipf's law

Zipf's law says that the most frequent word in a language (such as "the") occurs approximately twice as often as the second most frequent word ("of"), which occurs twice as often as the fourth

1. In fact, wealth distributions are even more skewed than this. For example, as of 2014, 80 billionaires now have as much wealth as 3.5 billion people! (Source: http://www.pbs.org/newshour/making-sense/wealthiest-getting-wealthier-lobbying-lot.) Such extreme income inequality exists in many plutocratic countries, including the USA (see e.g., [HP10]).

most frequent word, etc. This corresponds to a Pareto distribution of the form

$$p(x = r) \propto \kappa r^{-a} \tag{2.30}$$

where r is the rank of word x when sorted by frequency, and κ and a are constants. If we set $a = 1$, we recover Zipf's law.[2] Thus Zipf's law predicts that if we plot the log frequency of words vs their log rank, we will get a straight line with slope -1. This is in fact true, as illustrated in Figure 2.4.[3] See [Ada00] for further discussion of Zipf's law, and Section 2.6.2 for a discussion of language models.

2.2.4 Continuous distributions on $[0, 1]$

In this section, we discuss some univariate distributions defined on the $[0, 1]$ interval.

2.2.4.1 Beta distribution

The **beta distribution** has support over the interval $[0, 1]$ and is defined as follows:

$$\text{Beta}(x|a, b) = \frac{1}{B(a, b)} x^{a-1}(1 - x)^{b-1} \tag{2.31}$$

We require $a, b > 0$ to ensure the distribution is integrable (i.e., to ensure $B(a, b)$ exists). If $a = b = 1$, we get the uniform distribution. If a and b are both less than 1, we get a bimodal distribution with "spikes" at 0 and 1; if a and b are both greater than 1, the distribution is unimodal.

2.2.5 The multivariate Gaussian (normal) distribution

The most widely used joint probability distribution for continuous random variables is the **multivariate Gaussian** or **multivariate normal** (**MVN**). This is mostly because it is mathematically convenient, but also because the Gaussian assumption is fairly reasonable in many cases.

2.2.5.1 Definition

The MVN density is defined by the following:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right] \tag{2.32}$$

where $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}] \in \mathbb{R}^D$ is the mean vector, and $\boldsymbol{\Sigma} = \text{Cov}[\boldsymbol{x}]$ is the $D \times D$ covariance matrix. The normalization constant $Z = (2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}$ just ensures that the pdf integrates to 1. The expression inside the exponential (ignoring the factor of -0.5) is the squared **Mahalanobis distance** between the data vector $\boldsymbol{x}$ and the mean vector $\boldsymbol{\mu}$, given by

$$d_{\boldsymbol{\Sigma}}(\boldsymbol{x}, \boldsymbol{\mu})^2 = (\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) \tag{2.33}$$

2. For example, $p(x = 2) = \kappa 2^{-1} = 2\kappa 4^{-1} = 2p(x = 4)$.
3. We remove the first 10 words from the plot, since they don't fit the prediction as well.

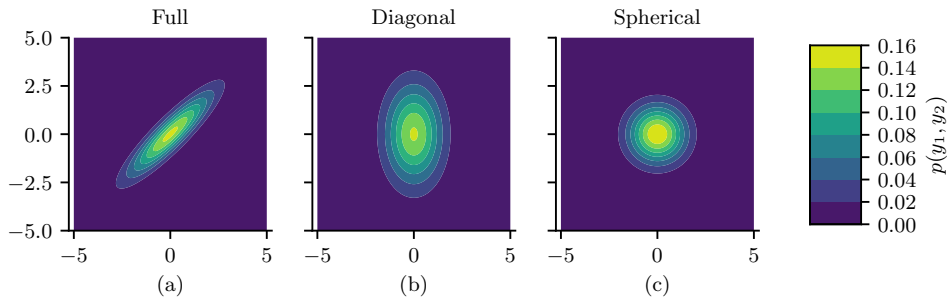


Figure 2.5: Visualization of a 2d Gaussian density in terms of level sets of constant probability density. (a) A full covariance matrix has elliptical contours. (b) A diagonal covariance matrix is an axis aligned ellipse. (c) A spherical covariance matrix has a circular shape. Generated by gauss_plot_2d.ipynb.

In 2d, the MVN is known as the **bivariate Gaussian** distribution. Its pdf can be represented as $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{x} \in \mathbb{R}^2$, $\boldsymbol{\mu} \in \mathbb{R}^2$ and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_2^2 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \tag{2.34}$$

where the correlation coefficient is given by $\rho \triangleq \frac{\sigma_{12}^2}{\sigma_1\sigma_2}$.

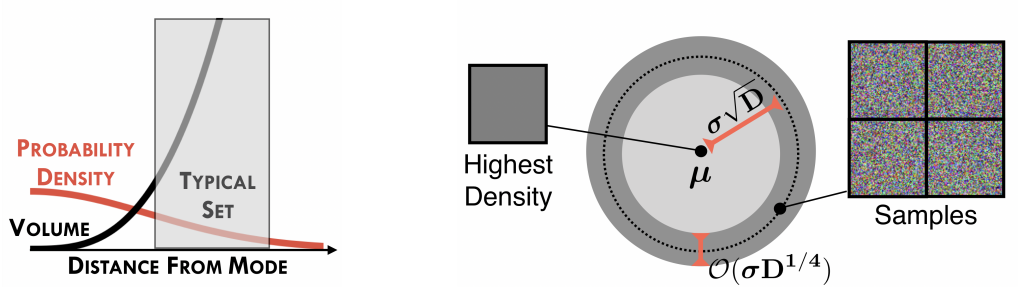
Figure 2.5 plots some MVN densities in 2d for three different kinds of covariance matrices. A **full covariance matrix** has $D(D+1)/2$ parameters, where we divide by 2 since $\boldsymbol{\Sigma}$ is symmetric. A **diagonal covariance matrix** has D parameters, and has 0s in the off-diagonal terms. A **spherical covariance matrix**, also called **isotropic covariance matrix**, has the form $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}_D$, so it only has one free parameter, namely σ^2.

2.2.5.2 Gaussian shells

Multivariate Gaussians can behave rather counterintuitively in high dimensions. In particular, we can ask: if we draw samples $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$, where D is the number of dimensions, where do we expect most of the $\boldsymbol{x}$ to lie? Since the peak (mode) of the pdf is at the origin, it is natural to expect most samples to be near the origin. However, in high dimensions, the typical set of a Gaussian is a thin shell or annulus with a distance from origin given by $r = \sigma\sqrt{D}$ and a thickness of $O(\sigma D^{\frac{1}{4}})$. The intuitive reason for this is as follows: although the density decays as $e^{-r^2/2}$, meaning density decreases from the origin, the volume of a sphere grows as r^D, meaning volume increases from the origin, and since mass is density times volume, the majority of points end up in this annulus where these two terms "balance out". This is called the "**Gaussian soap bubble**" phenomenon, and is illustrated in Figure 2.6.[4]

To see why the typical set for a Gaussian is concentrated in a thin annulus at radius $\sqrt{D}$, consider the squared distance of a point $\boldsymbol{x}$ from the origin, $d(\boldsymbol{x}) = \sqrt{\sum_{i=1}^{D} x_i^2}$, where $x_i \sim \mathcal{N}(0, 1)$. The expected squared distance is given by $\mathbb{E}\left[d^2\right] = \sum_{i=1}^{D} \mathbb{E}\left[x_i^2\right] = D$, and the variance of the squared

4. For a more detailed explanation, see this blog post by Ferenc Huszar: https://www.inference.vc/high-dimensional-gaussian-distributions-are-soap-bubble/.



(a) *(b)*

Figure 2.6: (a) Cartoon illustration of why the typical set of a Gaussian is not centered at the mode of the distribution. (b) Illustration of the typical set of a Gaussian, which is concentrated in a thin annulus of thickness $\sigma D^{1/4}$ and distance $\sigma D^{1/2}$ from the origin. We also show an image with the highest density (the all gray image on the left). as well as some high probability samples (the speckle noise images on the right). From Figure 1 of [Nal+19a]. Used with kind permission of Eric Nalisnick.

distance is given by $\mathbb{V}\left[d^2\right] = \sum_{i=1}^{D} \mathbb{V}\left[x_i^2\right] = D$. As D grows, the coefficient of variation (i.e., the SD relative to the mean) goes to zero:

$$\lim_{D\to\infty} \frac{\text{std}\left[d^2\right]}{\mathbb{E}\left[d^2\right]} = \lim_{D\to\infty} \frac{\sqrt{D}}{D} = 0 \tag{2.35}$$

Thus the expected square distance concentrates around D, so the expected distance concentrates around $\mathbb{E}\left[d(\boldsymbol{x})\right] = \sqrt{D}$. See [Ver18] for a more rigorous proof, and Section 5.2.3 for a discussion of typical sets.

To see what this means in the context of images, in Figure 2.6b, we show some grayscale images that are sampled from a Gaussian of the form $\mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$, where $\boldsymbol{\mu}$ corresponds to the all-gray image. However, it is extremely unlikely that randomly sampled images would be close to all-gray, as shown in the figure.

2.2.5.3 Marginals and conditionals of a MVN

Let us partition our vector of random variables $\boldsymbol{x}$ into two parts, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, so

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix} \tag{2.36}$$

The marginals of this distribution are given by the following (see Section 2.2.5.5 for the proof):

$$p(\boldsymbol{x}_1) = \int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{x}_2 \triangleq \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_1^m, \boldsymbol{\Sigma}_1^m) = \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \tag{2.37}$$

$$p(\boldsymbol{x}_2) = \int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{x}_1 \triangleq \mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_2^m, \boldsymbol{\Sigma}_2^m) = \mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{2.38}$$

The conditional distributions can be shown to have the following form (see Section 2.2.5.5 for the proof):

$$p(\boldsymbol{x}_1|\boldsymbol{x}_2) = \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_{1|2}^c, \boldsymbol{\Sigma}_{1|2}^c) = \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2),\ \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}) \tag{2.39}$$

$$p(\boldsymbol{x}_2|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_{2|1}^c, \boldsymbol{\Sigma}_{2|1}^c) = \mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1),\ \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}) \tag{2.40}$$

Note that the posterior mean of $p(\boldsymbol{x}_1|\boldsymbol{x}_2)$ is a linear function of $\boldsymbol{x}_2$, but the posterior covariance is independent of $\boldsymbol{x}_2$; this is a peculiar property of Gaussian distributions.

2.2.5.4 Information (canonical) form

It is common to parameterize the MVN in terms of the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. However, for reasons which are explained in Section 2.3.2.5, it is sometimes useful to represent the Gaussian distribution using **canonical parameters** or **natural parameters**, defined as

$$\boldsymbol{\Lambda} \triangleq \boldsymbol{\Sigma}^{-1}, \quad \boldsymbol{\eta} \triangleq \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \tag{2.41}$$

The matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is known as the **precision matrix**, and the vector $\boldsymbol{\eta}$ is known as the **precision-weighted mean**. We can convert back to the more familiar **moment parameters** using

$$\boldsymbol{\mu} = \boldsymbol{\Lambda}^{-1}\boldsymbol{\eta}, \quad \boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1} \tag{2.42}$$

Hence we can write the MVN in **canonical form** (also called **information form**) as follows:

$$\mathcal{N}_c(\boldsymbol{x}|\boldsymbol{\eta}, \boldsymbol{\Lambda}) \triangleq c \exp\left(\boldsymbol{x}^\mathsf{T}\boldsymbol{\eta} - \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{x}\right) \tag{2.43}$$

$$c \triangleq \frac{\exp(-\frac{1}{2}\boldsymbol{\eta}^\mathsf{T}\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta})}{(2\pi)^{D/2}\sqrt{\det(\boldsymbol{\Lambda}^{-1})}} \tag{2.44}$$

where we use the notation $\mathcal{N}_c()$ to distinguish it from the standard parameterization $\mathcal{N}()$. For more information on moment and natural parameters, see Section 2.3.2.5.

It is also possible to derive the marginalization and conditioning formulas in information form (see Section 2.2.5.6 for the derivation). For the marginals we have

$$p(\boldsymbol{x}_1) = \mathcal{N}_c(\boldsymbol{x}_1|\boldsymbol{\eta}_1^m, \boldsymbol{\Lambda}_1^m) = \mathcal{N}_c(\boldsymbol{x}_1|\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\eta}_2, \boldsymbol{\Lambda}_{11} - \boldsymbol{\Lambda}_{12}\boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}) \tag{2.45}$$

$$p(\boldsymbol{x}_2) = \mathcal{N}_c(\boldsymbol{x}_2|\boldsymbol{\eta}_2^m, \boldsymbol{\Lambda}_2^m) = \mathcal{N}_c(\boldsymbol{x}_2|\boldsymbol{\eta}_2 - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\eta}_1, \boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}) \tag{2.46}$$

For the conditionals we have

$$p(\boldsymbol{x}_1|\boldsymbol{x}_2) = \mathcal{N}_c(\boldsymbol{x}_1|\boldsymbol{\eta}_{1|2}^c, \boldsymbol{\Lambda}_{1|2}^c) = \mathcal{N}_c(\boldsymbol{x}_1|\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2, \boldsymbol{\Lambda}_{11}) \tag{2.47}$$

$$p(\boldsymbol{x}_2|\boldsymbol{x}_1) = \mathcal{N}_c(\boldsymbol{x}_2|\boldsymbol{\eta}_{2|1}^c, \boldsymbol{\Lambda}_{2|1}^c) = \mathcal{N}_c(\boldsymbol{x}_2|\boldsymbol{\eta}_2 - \boldsymbol{\Lambda}_{21}\boldsymbol{x}_1, \boldsymbol{\Lambda}_{22}) \tag{2.48}$$

Thus we see that marginalization is easier in moment form, and conditioning is easier in information form.

2.2.5.5 Derivation: moment form

In this section, we derive Equation (2.37) and Equation (2.39) for marginalizing and conditioning an MVN in moment form.

Before we dive in, we need to introduce the following result, for the **inverse of a partitioned matrix** of the form

$$\mathbf{M} = \begin{pmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{pmatrix} \tag{2.49}$$

where we assume $\mathbf{E}$ and $\mathbf{H}$ are invertible. One can show (see e.g., [Mur22, Sec 7.3.2] for the proof) that

$$\mathbf{M}^{-1} = \begin{pmatrix} (\mathbf{M}/\mathbf{H})^{-1} & -(\mathbf{M}/\mathbf{H})^{-1}\mathbf{F}\mathbf{H}^{-1} \\ -\mathbf{H}^{-1}\mathbf{G}(\mathbf{M}/\mathbf{H})^{-1} & \mathbf{H}^{-1} + \mathbf{H}^{-1}\mathbf{G}(\mathbf{M}/\mathbf{H})^{-1}\mathbf{F}\mathbf{H}^{-1} \end{pmatrix} \tag{2.50}$$

$$= \begin{pmatrix} \mathbf{E}^{-1} + \mathbf{E}^{-1}\mathbf{F}(\mathbf{M}/\mathbf{E})^{-1}\mathbf{G}\mathbf{E}^{-1} & -\mathbf{E}^{-1}\mathbf{F}(\mathbf{M}/\mathbf{E})^{-1} \\ -(\mathbf{M}/\mathbf{E})^{-1}\mathbf{G}\mathbf{E}^{-1} & (\mathbf{M}/\mathbf{E})^{-1} \end{pmatrix} \tag{2.51}$$

where

$$\mathbf{M}/\mathbf{H} \triangleq \mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G} \tag{2.52}$$

$$\mathbf{M}/\mathbf{E} \triangleq \mathbf{H} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F} \tag{2.53}$$

We say that $\mathbf{M}/\mathbf{H}$ is the **Schur complement** of $\mathbf{M}$ wrt $\mathbf{H}$, and $\mathbf{M}/\mathbf{E}$ is the Schur complement of $\mathbf{M}$ wrt $\mathbf{E}$.

From the above, we also have the following important result, known as the **matrix inversion lemma** or the **Sherman-Morrison-Woodbury formula**:

$$(\mathbf{M}/\mathbf{H})^{-1} = (\mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1} = \mathbf{E}^{-1} + \mathbf{E}^{-1}\mathbf{F}(\mathbf{H} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1}\mathbf{G}\mathbf{E}^{-1} \tag{2.54}$$

Now we return to the derivation of the MVN conditioning equation. Let us factor the joint $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$ as $p(\boldsymbol{x}_2)p(\boldsymbol{x}_1|\boldsymbol{x}_2)$ as follows:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto \exp\left\{ -\frac{1}{2} \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}^\mathsf{T} \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix} \right\} \tag{2.55}$$

Using the equation for the inverse of a block structured matrix, the above exponent becomes

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto \exp\left\{ -\frac{1}{2} \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}^\mathsf{T} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} & \mathbf{I} \end{pmatrix} \begin{pmatrix} (\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22})^{-1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{22}^{-1} \end{pmatrix} \right. \tag{2.56}$$

$$\left. \times \begin{pmatrix} \mathbf{I} & -\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix} \right\} \tag{2.57}$$

$$= \exp\left\{ -\frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2))^\mathsf{T}(\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22})^{-1} \right. \tag{2.58}$$

$$\left. (\boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)) \right\} \times \exp\left\{ -\frac{1}{2}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^\mathsf{T}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \right\} \tag{2.59}$$

This is of the form

$$\exp(\text{quadratic form in } \boldsymbol{x}_1, \boldsymbol{x}_2) \times \exp(\text{quadratic form in } \boldsymbol{x}_2) \tag{2.60}$$

Hence we have successfully factorized the joint as

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = p(\boldsymbol{x}_1|\boldsymbol{x}_2)p(\boldsymbol{x}_2) \tag{2.61}$$

$$= \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})\mathcal{N}(\boldsymbol{x}_2|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{2.62}$$

where

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \tag{2.63}$$

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22} \triangleq \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \tag{2.64}$$

where $\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22}$ is as the Schur complement of $\boldsymbol{\Sigma}$ wrt $\boldsymbol{\Sigma}_{22}$.

2.2.5.6 Derivation: information form

In this section, we derive Equation (2.46) and Equation (2.47) for marginalizing and conditioning an MVN in information form.

First we derive the conditional formula.[5] Let us partition the information form parameters as follows:

$$\boldsymbol{\eta} = \begin{pmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{pmatrix} \tag{2.65}$$

We can now write the joint log probabilty of $\boldsymbol{x}_1, \boldsymbol{x}_2$ as

$$\ln p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}^\mathsf{T} \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} + \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}^\mathsf{T} \begin{pmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{pmatrix} + \text{const.} \tag{2.66}$$

$$\begin{aligned} &= -\frac{1}{2}\boldsymbol{x}_1^\mathsf{T}\boldsymbol{\Lambda}_{11}\boldsymbol{x}_1 - \frac{1}{2}\boldsymbol{x}_2^\mathsf{T}\boldsymbol{\Lambda}_{22}\boldsymbol{x}_2 - \frac{1}{2}\boldsymbol{x}_1^\mathsf{T}\boldsymbol{\Lambda}_{12}\boldsymbol{x}_2 - \frac{1}{2}\boldsymbol{x}_2^\mathsf{T}\boldsymbol{\Lambda}_{21}\boldsymbol{x}_1 \\ &\quad + \boldsymbol{x}_1^\mathsf{T}\boldsymbol{\eta}_1 + \boldsymbol{x}_2^\mathsf{T}\boldsymbol{\eta}_2 + \text{const.} \end{aligned} \tag{2.67}$$

where the constant term does not depend on $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$.

To calculate the parameters of the conditional distribution $p(\boldsymbol{x}_1|\boldsymbol{x}_2)$, we fix the value of $\boldsymbol{x}_2$ and collect the terms which are quadratic in $\boldsymbol{x}_1$ for the conditional precision and then linear in $\boldsymbol{x}_1$ for the conditional precision-weighted mean. The terms which are quadratic in $\boldsymbol{x}_1$ are just $-\frac{1}{2}\boldsymbol{x}_1^\mathsf{T}\boldsymbol{\Lambda}_{11}\boldsymbol{x}_1$, and hence

$$\boldsymbol{\Lambda}_{1|2}^c = \boldsymbol{\Lambda}_{11} \tag{2.68}$$

The terms which are linear in $\boldsymbol{x}_1$ are

$$-\frac{1}{2}\boldsymbol{x}_1^\mathsf{T}\boldsymbol{\Lambda}_{12}\boldsymbol{x}_2 - \frac{1}{2}\boldsymbol{x}_2^\mathsf{T}\boldsymbol{\Lambda}_{21}\boldsymbol{x}_1 + \boldsymbol{x}_1^\mathsf{T}\boldsymbol{\eta}_1 = \boldsymbol{x}_1^\mathsf{T}(\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2) \tag{2.69}$$

5. This derivation is due to Giles Harper-Donnelly.

since $\boldsymbol{\Lambda}_{21}^{\mathsf{T}} = \boldsymbol{\Lambda}_{12}$. Thus the conditional precision-weighted mean is

$$\boldsymbol{\eta}_{1|2}^{c} = \boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2. \tag{2.70}$$

We will now derive the results for marginalizing in information form. The marginal, $p(\boldsymbol{x}_2)$, can be calculated by integrating the joint, $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$, with respect to $\boldsymbol{x}_1$:

$$p(\boldsymbol{x}_2) = \int p(\boldsymbol{x}_1, \boldsymbol{x}_2) d\boldsymbol{x}_1 \tag{2.71}$$

$$\propto \int \exp\left\{-\frac{1}{2}\boldsymbol{x}_1^{\mathsf{T}}\boldsymbol{\Lambda}_{11}\boldsymbol{x}_1 - \frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{22}\boldsymbol{x}_2 - \frac{1}{2}\boldsymbol{x}_1^{\mathsf{T}}\boldsymbol{\Lambda}_{12}\boldsymbol{x}_2 - \frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{21}\boldsymbol{x}_1 + \boldsymbol{x}_1^{\mathsf{T}}\boldsymbol{\eta}_1 + \boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\eta}_2\right\} d\boldsymbol{x}_1, \tag{2.72}$$

where the terms in the exponent have been decomposed into the partitioned structure in Equation (2.65) as in Equation (2.67). Next, collecting all the terms involving $\boldsymbol{x}_1$,

$$p(\boldsymbol{x}_2) \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{22}\boldsymbol{x}_2 + \boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\eta}_2\right\} \int \exp\left\{-\frac{1}{2}\boldsymbol{x}_1^{\mathsf{T}}\boldsymbol{\Lambda}_{11}\boldsymbol{x}_1 + \boldsymbol{x}_1^{\mathsf{T}}(\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2)\right\} d\boldsymbol{x}_1, \tag{2.73}$$

we can recognise the integrand as an exponential quadratic form. Therefore the integral is equal to the normalising constant of a Gaussian with precision, $\boldsymbol{\Lambda}_{11}$, and precision weighted mean, $\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2$, which is given by the reciprocal of Equation (2.44). Substituting this in to our equation we have,

$$p(\boldsymbol{x}_2) \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{22}\boldsymbol{x}_2 + \boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\eta}_2\right\} \exp\left\{\frac{1}{2}(\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2)^{\mathsf{T}}\boldsymbol{\Lambda}_{11}^{-1}(\boldsymbol{\eta}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2)\right\} \tag{2.74}$$

$$\propto \exp\left\{-\frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{22}\boldsymbol{x}_2 + \boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\eta}_2 + \frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}\boldsymbol{x}_2 - \boldsymbol{x}_2^{\mathsf{T}}\boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\eta}_1\right\} \tag{2.75}$$

$$= \exp\left\{-\frac{1}{2}\boldsymbol{x}_2^{\mathsf{T}}(\boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12})\boldsymbol{x}_2 + \boldsymbol{x}_2^{\mathsf{T}}(\boldsymbol{\eta}_2 - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\eta}_1)\right\}, \tag{2.76}$$

which we now recognise as an exponential quadratic form in $\boldsymbol{x}_2$. Extract the quadratic terms to get the marginal precision,

$$\boldsymbol{\Lambda}_{22}^{m} = \boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}, \tag{2.77}$$

and the linear terms to get the marginal precision-weighted mean,

$$\boldsymbol{\eta}_2^{m} = \boldsymbol{\eta}_2 - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\eta}_1. \tag{2.78}$$

2.2.6 Linear Gaussian systems

Consider two random vectors $\boldsymbol{y} \in \mathbb{R}^D$ and $\boldsymbol{z} \in \mathbb{R}^L$, which are jointly Gaussian with the following joint distribution:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z} |\, \breve{\boldsymbol{\mu}}, \breve{\boldsymbol{\Sigma}}) \tag{2.79}$$

$$p(\boldsymbol{y}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{y}|\mathbf{W}\boldsymbol{z} + \boldsymbol{b}, \boldsymbol{\Omega}) \tag{2.80}$$

where $\mathbf{W}$ is a matrix of size $D \times L$. This is an example of a **linear Gaussian system**.

2.2.6.1 Joint distribution

The corresponding joint distribution, $p(\boldsymbol{z}, \boldsymbol{y}) = p(\boldsymbol{z})p(\boldsymbol{y}|\boldsymbol{z})$, is itself a $D + L$ dimensional Gaussian, with mean and covariance given by the following (this result can be obtained by moment matching):

$$p(\boldsymbol{z}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}, \boldsymbol{y}|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \tag{2.81a}$$

$$\tilde{\boldsymbol{\mu}} \triangleq \begin{pmatrix} \breve{\boldsymbol{\mu}} \\ \boldsymbol{m} \end{pmatrix} \triangleq \begin{pmatrix} \breve{\boldsymbol{\mu}} \\ \mathbf{W}\,\breve{\boldsymbol{\mu}} + \boldsymbol{b} \end{pmatrix} \tag{2.81b}$$

$$\tilde{\boldsymbol{\Sigma}} \triangleq \begin{pmatrix} \breve{\boldsymbol{\Sigma}} & \mathbf{C} \\ \mathbf{C} & \mathbf{S} \end{pmatrix} \triangleq \begin{pmatrix} \breve{\boldsymbol{\Sigma}} & \breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T} \\ \mathbf{W}\,\breve{\boldsymbol{\Sigma}} & \mathbf{W}\,\breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T} + \boldsymbol{\Omega} \end{pmatrix} \tag{2.81c}$$

See Algorithm 7 for some pseudocode to compute this joint distribution.

2.2.6.2 Posterior distribution (Bayes rule for Gaussians)

Now we consider computing the posterior $p(\boldsymbol{z}|\boldsymbol{y})$ from a linear Gaussian system. Using Equation (2.39) for conditioning a joint Gaussian, we find that the posterior is given by

$$p(\boldsymbol{z}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}|\,\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}) \tag{2.82a}$$

$$\widehat{\boldsymbol{\mu}} = \breve{\boldsymbol{\mu}} + \breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T}(\boldsymbol{\Omega} + \mathbf{W}\,\breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T})^{-1}(\boldsymbol{y} - (\mathbf{W}\,\breve{\boldsymbol{\mu}} + \boldsymbol{b})) \tag{2.82b}$$

$$\widehat{\boldsymbol{\Sigma}} = \breve{\boldsymbol{\Sigma}} - \breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T}(\boldsymbol{\Omega} + \mathbf{W}\,\breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T})^{-1}\mathbf{W}\,\breve{\boldsymbol{\Sigma}} \tag{2.82c}$$

This is known as **Bayes' rule for Gaussians**. We see that if the prior $p(\boldsymbol{z})$ is Gaussian, and the likelihood $p(\boldsymbol{y}|\boldsymbol{z})$ is Gaussian, then the posterior $p(\boldsymbol{z}|\boldsymbol{y})$ is also Gaussian. We therefore say that the Gaussian prior is a **conjugate prior** for the Gaussian likelihood, since the posterior distribution has the same type as the prior. (In other words, Gaussians are closed under Bayesian updating.)

We can simplify these equations by defining $\mathbf{S} = \mathbf{W}\,\breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T} + \boldsymbol{\Omega}$, $\mathbf{C} = \breve{\boldsymbol{\Sigma}}\,\mathbf{W}^\mathsf{T}$, and $\boldsymbol{m} = \mathbf{W}\,\breve{\boldsymbol{\mu}} + \boldsymbol{b}$, as in Equation (2.81). We also define the **Kalman gain matrix**:[6]

$$\mathbf{K} = \mathbf{C}\mathbf{S}^{-1} \tag{2.83}$$

From this, we get the posterior

$$\widehat{\boldsymbol{\mu}} = \breve{\boldsymbol{\mu}} + \mathbf{K}(\boldsymbol{y} - \boldsymbol{m}) \tag{2.84}$$

$$\widehat{\boldsymbol{\Sigma}} = \breve{\boldsymbol{\Sigma}} - \mathbf{K}\mathbf{C}^\mathsf{T} \tag{2.85}$$

Note that

$$\mathbf{K}\mathbf{S}\mathbf{K}^\mathsf{T} = \mathbf{C}\mathbf{S}^{-1}\mathbf{S}\mathbf{S}^{-\mathsf{T}}\mathbf{C}^\mathsf{T} = \mathbf{C}\mathbf{S}^{-1}\mathbf{C}^\mathsf{T} = \mathbf{K}\mathbf{C}^\mathsf{T} \tag{2.86}$$

and hence we can also write the posterior covariance as

$$\widehat{\boldsymbol{\Sigma}} = \breve{\boldsymbol{\Sigma}} - \mathbf{K}\mathbf{S}\mathbf{K}^\mathsf{T} \tag{2.87}$$

6. The name comes from the Kalman filter algorithm, which we discuss in Section 8.3.2.

Using the matrix inversion lemma from Equation (2.54), we can also rewrite the posterior in the following form [Bis06, p93], which takes $O(L^3)$ time instead of $O(D^3)$ time:

$$\widehat{\boldsymbol{\Sigma}} = (\breve{\boldsymbol{\Sigma}}^{-1} + \mathbf{W}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\mathbf{W})^{-1} \tag{2.88}$$

$$\widehat{\boldsymbol{\mu}} = \widehat{\boldsymbol{\Sigma}}\,[\mathbf{W}^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\,(\boldsymbol{y}-\boldsymbol{b}) + \breve{\boldsymbol{\Sigma}}^{-1}\breve{\boldsymbol{\mu}}] \tag{2.89}$$

Finally, note that the corresponding normalization constant for the posterior is just the marginal on $\boldsymbol{y}$ evaluated at the observed value:

$$\begin{aligned} p(\boldsymbol{y}) &= \int \mathcal{N}(\boldsymbol{z}|\,\breve{\boldsymbol{\mu}}, \breve{\boldsymbol{\Sigma}})\mathcal{N}(\boldsymbol{y}|\mathbf{W}\boldsymbol{z}+\boldsymbol{b}, \boldsymbol{\Omega})d\boldsymbol{z} \\ &= \mathcal{N}(\boldsymbol{y}|\mathbf{W}\,\breve{\boldsymbol{\mu}} + \boldsymbol{b}, \boldsymbol{\Omega} + \mathbf{W}\,\breve{\boldsymbol{\Sigma}}\,\mathbf{W}^{\mathsf{T}}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{m}, \mathbf{S}) \end{aligned} \tag{2.90}$$

From this, we can easily compute the log marginal likelihood. We summarize all these equations in Algorithm 7.

2.2.6.3 Example: Sensor fusion with known measurement noise

Suppose we have an unknown quantity of interest, $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$, from which we get two noisy measurements, $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{z}, \boldsymbol{\Sigma}_x)$ and $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{z}, \boldsymbol{\Sigma}_y)$. Pictorially, we can represent this example as $\boldsymbol{x} \leftarrow \boldsymbol{z} \rightarrow \boldsymbol{y}$. This is an example of a linear Gaussian system. Our goal is to combine the evidence together, to compute $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$. This is known as **sensor fusion**. (In this section, we assume $\boldsymbol{\theta} = (\boldsymbol{\Sigma}_x, \boldsymbol{\Sigma}_y)$ is known. See Supplementary Section 2.1.2 for the general case.)

We can combine $\boldsymbol{x}$ and $\boldsymbol{y}$ into a single vector $\boldsymbol{v}$, so the model can be represented as $\boldsymbol{z} \rightarrow \boldsymbol{v}$, where $p(\boldsymbol{v}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{v}|\mathbf{W}\boldsymbol{z}, \boldsymbol{\Sigma}_v)$, where $\mathbf{W} = [\mathbf{0}, \mathbf{I}; \mathbf{0}, \mathbf{I}]$ and $\boldsymbol{\Sigma}_v = [\boldsymbol{\Sigma}_x, \mathbf{0}; \mathbf{0}, \boldsymbol{\Sigma}_y]$ are block-structured matrices. We can then apply Bayes' rule for Gaussians (Section 2.2.6.2) to compute $p(\boldsymbol{z}|\boldsymbol{v})$.

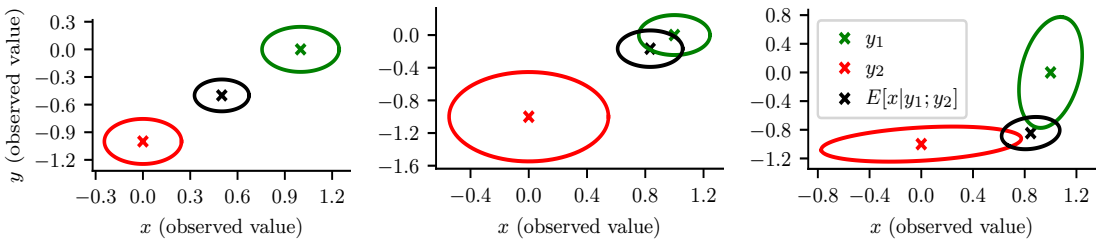
Figure 2.7(a) gives a 2d example, where we set $\boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}_y = 0.01\mathbf{I}_2$, so both sensors are equally reliable. In this case, the posterior mean is halfway between the two observations, $\boldsymbol{x}$ and $\boldsymbol{y}$. In Figure 2.7(b), we set $\boldsymbol{\Sigma}_x = 0.05\mathbf{I}_2$ and $\boldsymbol{\Sigma}_y = 0.01\mathbf{I}_2$, so sensor 2 is more reliable than sensor 1. In this case, the posterior mean is closer to $\boldsymbol{y}$. In Figure 2.7(c), we set

$$\boldsymbol{\Sigma}_x = 0.01\begin{pmatrix} 10 & 1 \\ 1 & 1 \end{pmatrix}, \quad \boldsymbol{\Sigma}_y = 0.01\begin{pmatrix} 1 & 1 \\ 1 & 10 \end{pmatrix} \tag{2.91}$$

so sensor 1 is more reliable in the second component (vertical direction), and sensor 2 is more reliable in the first component (horizontal direction). In this case, the posterior mean uses $\boldsymbol{x}$'s vertical component and $\boldsymbol{y}$'s horizontal component.

2.2.7 A general calculus for linear Gaussian systems

In this section, we discuss a general method for performing inference in linear Gaussian systems. The key is to define joint distributions over the relevant variables in terms of a **potential function**, represented in information form. We can then easily derive rules for marginalizing potentials, multiplying and dividing potentials, and conditioning them on observations. Once we have defined these operations, we can use them inside of the belief propagation algorithm (Section 9.2) or junction tree algorithm (Supplementary Section 9.2) to compute quantities of interest. We give the details on how to perform these operations below; our presentation is based on [Lau92; Mur02].



(a) (b) (c)

Figure 2.7: We observe $\boldsymbol{x} = (0, -1)$ (red cross) and $\boldsymbol{y} = (1, 0)$ (green cross) and estimate $\mathbb{E}[\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}]$ (black cross). (a) Equally reliable sensors, so the posterior mean estimate is in between the two circles. (b) Sensor 2 is more reliable, so the estimate shifts more towards the green circle. (c) Sensor 1 is more reliable in the vertical direction, Sensor 2 is more reliable in the horizontal direction. The estimate is an appropriate combination of the two measurements. Generated by sensor_fusion_2d.ipynb.

2.2.7.1 Moment and canonical parameterization

We can represent a Gaussian distribution in moment form or in canonical (information) form. In moment form we have

$$\phi(\boldsymbol{x}; p, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p \times \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right) \tag{2.92}$$

where $p = (2\pi)^{-n/2} |\boldsymbol{\Sigma}|^{-\frac{1}{2}}$ is the normalizing constant that ensures $\int_{\boldsymbol{x}} \phi(\boldsymbol{x}; p, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = 1$. ($n$ is the dimensionality of $\boldsymbol{x}$.) Expanding out the quadratic form and collecting terms we get the canonical form:

$$\phi(\boldsymbol{x}; g, \boldsymbol{h}, \mathbf{K}) = \exp\left(g + \boldsymbol{x}^\mathsf{T}\boldsymbol{h} - \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathbf{K}\boldsymbol{x}\right) = \exp\left(g + \sum_i h_i x_i - \frac{1}{2}\sum_i \sum_k K_{ij} x_i x_j\right) \tag{2.93}$$

where

$$\mathbf{K} = \boldsymbol{\Sigma}^{-1} \tag{2.94}$$

$$\boldsymbol{h} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \tag{2.95}$$

$$g = \log p - \frac{1}{2}\boldsymbol{\mu}^\mathsf{T}\mathbf{K}\boldsymbol{\mu} \tag{2.96}$$

$\mathbf{K}$ is often called the precision matrix.

Note that potentials need not be probability distributions, and need not be normalizable (integrate to 1). We keep track of the constant terms (p or g) so we can compute the likelihood of the evidence.

2.2.7.2 Multiplication and division

We can define multiplication and division in the Gaussian case by using canonical forms, as follows. To multiply $\phi_1(x_1, \ldots, x_k; g_1, \boldsymbol{h}_1, \mathbf{K}_1)$ by $\phi_2(x_{k+1}, \ldots, x_n; g_2, \boldsymbol{h}_2, \mathbf{K}_2)$, we extend them both to the

same domain $x_1, \ldots, x_n$ by adding zeros to the appropriate dimensions, and then computing

$$(g_1, \boldsymbol{h}_1, \mathbf{K}_1) * (g_2, \boldsymbol{h}_2, \mathbf{K}_2) = (g_1 + g_2, \boldsymbol{h}_1 + \boldsymbol{h}_2, \mathbf{K}_1 + \mathbf{K}_2) \tag{2.97}$$

Division is defined as follows:

$$(g_1, \boldsymbol{h}_1, \mathbf{K}_1)/(g_2, \boldsymbol{h}_2, \mathbf{K}_2) = (g_1 - g_2, \boldsymbol{h}_1 - \boldsymbol{h}_2, \mathbf{K}_1 - \mathbf{K}_2) \tag{2.98}$$

2.2.7.3 Marginalization

Let ϕ_W be a potential over a set W of variables. We can compute the potential over a subset $V \subset W$ of variables by marginalizing, denoted $\phi_V = \sum_{W \setminus V} \phi_W$. Let

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \end{pmatrix}, \qquad \mathbf{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix}, \tag{2.99}$$

with $\boldsymbol{x}_1$ having dimension n_1 and $\boldsymbol{x}_2$ having dimension n_2. It can be shown that

$$\int_{\boldsymbol{x}_1} \phi(\boldsymbol{x}_1, \boldsymbol{x}_2; g, \boldsymbol{h}, \mathbf{K}) = \phi(\boldsymbol{x}_2; \hat{g}, \hat{\boldsymbol{h}}, \hat{\mathbf{K}}) \tag{2.100}$$

where

$$\hat{g} = g + \frac{1}{2}\left(n_1 \log(2\pi) - \log|\mathbf{K}_{11}| + \boldsymbol{h}_1^\intercal \mathbf{K}_{11}^{-1} \boldsymbol{h}_1\right) \tag{2.101}$$

$$\hat{\boldsymbol{h}} = \boldsymbol{h}_2 - \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\boldsymbol{h}_1 \tag{2.102}$$

$$\hat{\mathbf{K}} = \mathbf{K}_{22} - \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{K}_{12} \tag{2.103}$$

2.2.7.4 Conditioning on evidence

Consider a potential defined on $(\boldsymbol{x}, \boldsymbol{y})$. Suppose we observe the value $\boldsymbol{y}$. The new potential is given by the following reduced dimensionality object:

$$\phi^*(\boldsymbol{x}) = \exp[g + \begin{pmatrix} \boldsymbol{x}^T & \boldsymbol{y}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_X \\ \boldsymbol{h}_Y \end{pmatrix} - \frac{1}{2} \begin{pmatrix} \boldsymbol{x}^T & \boldsymbol{y}^T \end{pmatrix} \begin{pmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \tag{2.104}$$

$$= \exp[\left(g + \boldsymbol{h}_Y^T \boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^T \mathbf{K}_{YY} \boldsymbol{y}\right) + \boldsymbol{x}^T(\boldsymbol{h}_X - \mathbf{K}_{XY}\boldsymbol{y}) - \frac{1}{2}\boldsymbol{x}^T \mathbf{K}_{XX} \boldsymbol{x}] \tag{2.105}$$

This generalizes the corresponding equation in [Lau92] to the vector-valued case.

2.2.7.5 Converting a linear-Gaussian CPD to a canonical potential

Finally we discuss how to create the initial potentials, assuming we start with a directed Gaussian graphical model. In particular, consider a node with a linear-Gaussian CPD:

$$p(\boldsymbol{x}|\boldsymbol{u}) = c\exp\left[-\frac{1}{2}\left((\boldsymbol{x}-\boldsymbol{\mu}-\mathbf{B}^T\boldsymbol{u})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}-\mathbf{B}^T\boldsymbol{u})\right)\right] \tag{2.106}$$

$$= \exp\left[-\frac{1}{2}\begin{pmatrix}\boldsymbol{x} & \boldsymbol{u}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}^{-1} & -\boldsymbol{\Sigma}^{-1}\mathbf{B}^T \\ -\mathbf{B}\boldsymbol{\Sigma}^{-1} & \mathbf{B}\boldsymbol{\Sigma}^{-1}\mathbf{B}^T\end{pmatrix}\begin{pmatrix}\boldsymbol{x} \\ \boldsymbol{u}\end{pmatrix}\right. \tag{2.107}$$

$$\left. + \begin{pmatrix}\boldsymbol{x} & \boldsymbol{u}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\mathbf{B}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\end{pmatrix} - \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \log c\right] \tag{2.108}$$

where $c = (2\pi)^{-n/2}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}$. Hence we set the canonical parameters to

$$g = -\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}| \tag{2.109}$$

$$\boldsymbol{h} = \begin{pmatrix}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\mathbf{B}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\end{pmatrix} \tag{2.110}$$

$$\mathbf{K} = \begin{pmatrix}\boldsymbol{\Sigma}^{-1} & -\boldsymbol{\Sigma}^{-1}\mathbf{B}^T \\ -\mathbf{B}\boldsymbol{\Sigma}^{-1} & \mathbf{B}\boldsymbol{\Sigma}^{-1}\mathbf{B}^T\end{pmatrix}. = \begin{pmatrix}\mathbf{I} \\ -\mathbf{B}\end{pmatrix}\boldsymbol{\Sigma}^{-1}\begin{pmatrix}\mathbf{I} & -\mathbf{B}\end{pmatrix} \tag{2.111}$$

In the special case that x is a scalar, the corresponding result can be found in [Lau92]. In particular we have $\Sigma^{-1} = 1/\sigma^2$, $B = b$ and $n = 1$, so the above becomes

$$g = \frac{-\mu^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2) \tag{2.112}$$

$$\boldsymbol{h} = \frac{\mu}{\sigma^2}\begin{pmatrix}1 \\ -\boldsymbol{b}\end{pmatrix} \tag{2.113}$$

$$\mathbf{K} = \frac{1}{\sigma}\begin{pmatrix}1 & -\boldsymbol{b}^T \\ -\boldsymbol{b} & \boldsymbol{b}\boldsymbol{b}^T\end{pmatrix}. \tag{2.114}$$

2.2.7.6 Example: Product of Gaussians

As an application of the above results, we can derive the (unnormalized) product of two Gaussians, as follows (see also [Kaa12, Sec 8.1.8]):

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \times \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \propto \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3) \tag{2.115}$$

where

$$\boldsymbol{\Sigma}_3 = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1} \tag{2.116}$$

$$\boldsymbol{\mu}_3 = \boldsymbol{\Sigma}_3(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) \tag{2.117}$$

We see that the posterior precision is a sum of the individual precisions, and the posterior mean is a precision-weighted combination of the individual means. We can also rewrite the result in the

following way, which only requires one matrix inversion:

$$\boldsymbol{\Sigma}_3 = \boldsymbol{\Sigma}_1(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}\boldsymbol{\Sigma}_2 \tag{2.118}$$

$$\boldsymbol{\mu}_3 = \boldsymbol{\Sigma}_2(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_1(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}\boldsymbol{\mu}_2 \tag{2.119}$$

In the scalar case, this becomes

$$\mathcal{N}(x|\mu_1, \sigma_1^2)\mathcal{N}(x|\mu_2, \sigma_2^2) \propto \mathcal{N}\left(x|\frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}, \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\right) \tag{2.120}$$

2.2.8 Some other multivariate continuous distributions

In this section, we summarize some other widely used multivariate continuous distributions.

2.2.8.1 Dirichlet distribution

A multivariate generalization of the beta distribution is the **Dirichlet**[7] distribution, which has support over the **probability simplex**, defined by

$$S_K = \{\boldsymbol{x} : 0 \leq x_k \leq 1, \sum_{k=1}^{K} x_k = 1\} \tag{2.121}$$

The pdf is defined as follows:

$$\text{Dir}(\boldsymbol{x}|\boldsymbol{\alpha}) \triangleq \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} x_k^{\alpha_k - 1} \mathbb{I}(\boldsymbol{x} \in S_K) \tag{2.122}$$

where $B(\boldsymbol{\alpha})$ is the multivariate beta function,

$$B(\boldsymbol{\alpha}) \triangleq \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)} \tag{2.123}$$

Figure 2.8 shows some plots of the Dirichlet when $K = 3$. We see that $\alpha_0 = \sum_k \alpha_k$ controls the strength of the distribution (how peaked it is), and the α_k control where the peak occurs. For example, $\text{Dir}(1, 1, 1)$ is a uniform distribution, $\text{Dir}(2, 2, 2)$ is a broad distribution centered at $(1/3, 1/3, 1/3)$, and $\text{Dir}(20, 20, 20)$ is a narrow distribution centered at $(1/3, 1/3, 1/3)$. $\text{Dir}(3, 3, 20)$ is an asymmetric distribution that puts more density in one of the corners. If $\alpha_k < 1$ for all k, we get "spikes" at the corners of the simplex. Samples from the distribution when $\alpha_k < 1$ will be sparse, as shown in Figure 2.9.

For future reference, here are some useful properties of the Dirichlet distribution:

$$\mathbb{E}[x_k] = \frac{\alpha_k}{\alpha_0}, \ \text{mode}[x_k] = \frac{\alpha_k - 1}{\alpha_0 - K}, \ \mathbb{V}[x_k] = \frac{\alpha_k(\alpha_0 - \alpha_k)}{\alpha_0^2(\alpha_0 + 1)} \tag{2.124}$$

where $\alpha_0 = \sum_k \alpha_k$.

Often we use a symmetric Dirichlet prior of the form $\alpha_k = \alpha/K$. In this case, we have $\mathbb{E}[x_k] = 1/K$, and $\mathbb{V}[x_k] = \frac{K-1}{K^2(\alpha+1)}$. So we see that increasing α increases the precision (decreases the variance) of the distribution.

7. Johann Dirichlet was a German mathematician, 1805–1859.

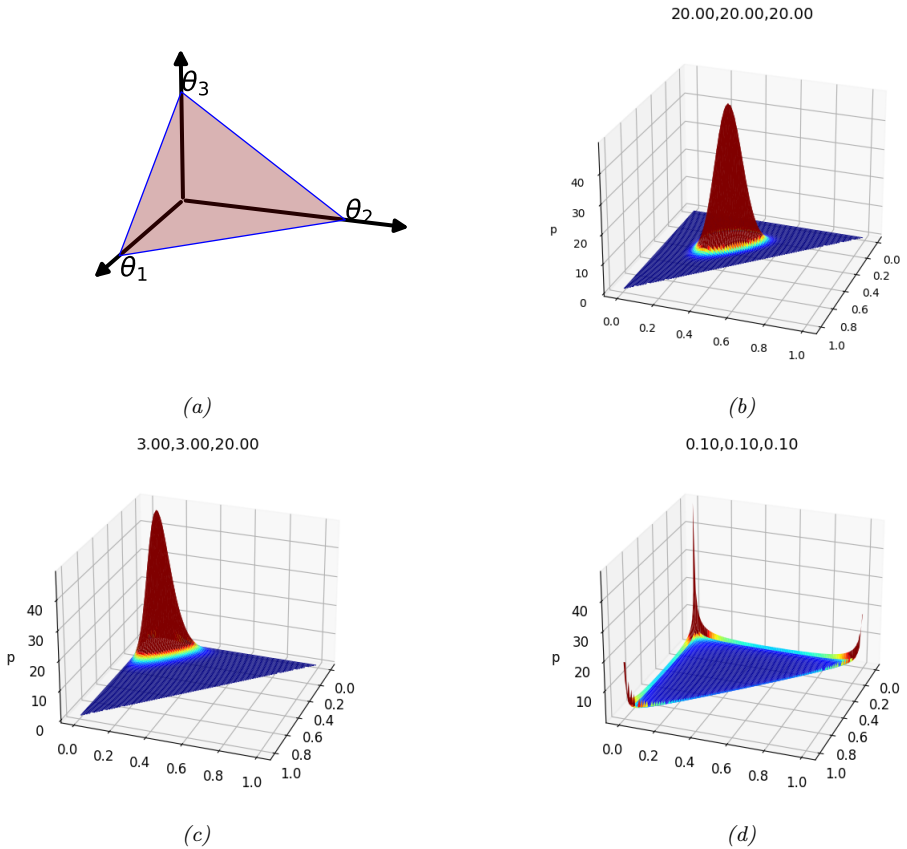


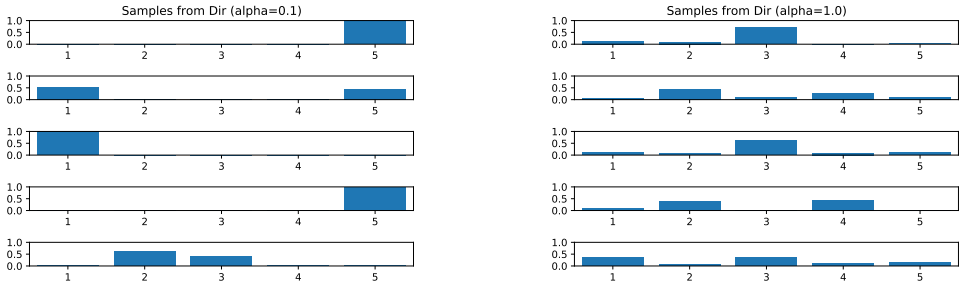
Figure 2.8: (a) The Dirichlet distribution when $K = 3$ defines a distribution over the simplex, which can be represented by the triangular surface. Points on this surface satisfy $0 \leq \theta_c \leq 1$ and $\sum_{c=1}^{3} \theta_c = 1$. Generated by dirichlet_3d_triangle_plot.ipynb. (b) Plot of the Dirichlet density for $\boldsymbol{\alpha} = (20, 20, 20)$. (c) Plot of the Dirichlet density for $\boldsymbol{\alpha} = (3, 3, 20)$. (d) Plot of the Dirichlet density for $\boldsymbol{\alpha} = (0.1, 0.1, 0.1)$. Generated by dirichlet_3d_spiky_plot.ipynb.

2.2.8.2 Multivariate Student distribution

One problem with Gaussians is that they are sensitive to outliers. Fortunately, we can easily extend the Student distribution, discussed in Section 2.2.2.3, to D dimensions. In particular, the pdf of the **multivariate Student distribution** is given by

$$\mathcal{T}_\nu(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{Z}\left[1 + \frac{1}{\nu}(\boldsymbol{x} - \boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right]^{-(\frac{\nu+D}{2})} \tag{2.125}$$

$$Z = \frac{\Gamma(\nu/2)}{\Gamma(\nu/2 + D/2)} \frac{\nu^{D/2}\pi^{D/2}}{|\boldsymbol{\Sigma}|^{-1/2}} \tag{2.126}$$



(a) *(b)*

Figure 2.9: Samples from a 5-dimensional symmetric Dirichlet distribution for different parameter values. (a) $\boldsymbol{\alpha} = (0.1, \ldots, 0.1)$*. This results in very sparse distributions, with many 0s. (b)* $\boldsymbol{\alpha} = (1, \ldots, 1)$*. This results in more uniform (and dense) distributions. Generated by dirichlet_samples_plot.ipynb.*

where $\boldsymbol{\Sigma}$ is called the scale matrix.

The Student has fatter tails than a Gaussian. The smaller ν is, the fatter the tails. As $\nu \to \infty$, the distribution tends towards a Gaussian. The distribution has these properties:

$$\text{mean} = \boldsymbol{\mu}, \ \text{mode} = \boldsymbol{\mu}, \text{cov} \ = \frac{\nu}{\nu - 2}\boldsymbol{\Sigma} \tag{2.127}$$

The mean is only well defined (finite) if $\nu > 1$. Similarly, the covariance is only well defined if $\nu > 2$.

2.2.8.3 Circular normal (von Mises Fisher) distribution

Sometimes data lives on the unit sphere, rather than being any point in Euclidean space. For example, any D dimensional vector that is ℓ_2-normalized lives on the unit $(D-1)$ sphere embedded in $\mathbb{R}^D$.

There is an extension of the Gaussian distribution that is suitable for such angular data, known as the **von Mises-Fisher** distribution, or the **circular normal** distribution. It has the following pdf:

$$\text{vMF}(\boldsymbol{x}|\boldsymbol{\mu}, \kappa) \triangleq \frac{1}{Z}\exp(\kappa \boldsymbol{\mu}^\mathsf{T}\boldsymbol{x}) \tag{2.128}$$

$$Z = \frac{(2\pi)^{D/2} I_{D/2-1}(\kappa)}{\kappa^{D/2-1}} \tag{2.129}$$

where $\boldsymbol{\mu}$ is the mean (with $||\boldsymbol{\mu}|| = 1$), $\kappa \geq 0$ is the concentration or precision parameter (analogous to $1/\sigma$ for a standard Gaussian), and Z is the normalization constant, with $I_r(\cdot)$ being the modified Bessel function of the first kind and order r. The vMF is like a spherical multivariate Gaussian, parameterized by **cosine distance** instead of Euclidean distance.

The vMF distribution can be used inside of a mixture model to cluster ℓ_2-normalized vectors, as an alternative to using a Gaussian mixture model [Ban+05]. If $\kappa \to 0$, this reduces to the spherical K-means algorithm. It can also be used inside of an admixture model (Section 28.4.2); this is called the spherical topic model [Rei+10].

If $D = 2$, an alternative is to use the **von Mises** distribution on the unit circle, which has the form

$$\text{vMF}(x|\mu, \kappa) = \frac{1}{Z}\exp(\kappa\cos(x-\mu)) \tag{2.130}$$

$$Z = 2\pi I_0(\kappa) \tag{2.131}$$

2.2.8.4 Matrix-normal distribution (MN)

The **matrix normal** distribution is defined by the following probability density function over matrices $\mathbf{X} \in \mathbb{R}^{n\times p}$:

$$\mathcal{MN}(\mathbf{X}|\mathbf{M}, \mathbf{U}, \mathbf{V}) \triangleq \frac{|\mathbf{V}|^{n/2}}{2\pi^{np/2}|\mathbf{U}|^{p/2}}\exp\left\{-\frac{1}{2}\text{tr}\left[(\mathbf{X}-\mathbf{M})^\mathsf{T}\mathbf{U}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{V}\right]\right\} \tag{2.132}$$

where $\mathbf{M} \in \mathbb{R}^{n\times p}$ is the mean value of $\mathbf{X}$, $\mathbf{U} \in \mathcal{S}_{++}^{n\times n}$ is the covariance among rows, and $\mathbf{V} \in \mathcal{S}_{++}^{p\times p}$ is the precision among columns. It can be seen that

$$\text{vec}(\mathbf{X}) \sim \mathcal{N}(\text{vec}(\mathbf{M}), \mathbf{V}^{-1} \otimes \mathbf{U}). \tag{2.133}$$

Note that there is another version of the definition of the matrix normal distribution using the column-covariance matrix $\tilde{\mathbf{V}} = \mathbf{V}^{-1}$ instead of $\mathbf{V}$, which leads to the density

$$\frac{1}{2\pi^{np/2}|\mathbf{U}|^{p/2}|\tilde{\mathbf{V}}|^{n/2}}\exp\left\{-\frac{1}{2}\text{tr}\left[(\mathbf{X}-\mathbf{M})^\mathsf{T}\mathbf{U}^{-1}(\mathbf{X}-\mathbf{M})\tilde{\mathbf{V}}^{-1}\right]\right\}. \tag{2.134}$$

These two versions of definition are obviously equivalent, but we will see that the definition we adopt in Equation (2.132) will leads to a neat update of the posterior distribution (just as the precision matrix is more convenient to use than the covariance matrix in analyzing the posterior of the multivariate normal distribution with a conjugate prior).

2.2.8.5 Wishart distribution

The **Wishart** distribution is the generalization of the Gamma distribution to positive definite matrices. Press [Pre05, p107] has said "The Wishart distribution ranks next to the Normal distribution in order of importance and usefulness in multivariate statistics". We will mostly use it to model our uncertainty when estimating covariance matrices (see Section 3.3).

The pdf of the Wishart is defined as follows:

$$\text{Wi}(\mathbf{\Sigma}|\mathbf{S}, \nu) \triangleq \frac{1}{Z}|\mathbf{\Sigma}|^{(\nu-D-1)/2}\exp\left(-\frac{1}{2}\text{tr}(\mathbf{\Sigma}\mathbf{S}^{-1})\right) \tag{2.135}$$

$$Z \triangleq |\mathbf{S}|^{-\nu/2}2^{\nu D/2}\Gamma_D(\nu/2) \tag{2.136}$$

Here ν is called the "degrees of freedom" and $\mathbf{S}$ is the "scale matrix". (We shall get more intuition for these parameters shortly.) The normalization constant only exists (and hence the pdf is only well defined) if $\nu > D-1$.

The distribution has these properties:

$$\text{mean} = \nu\mathbf{S},\ \text{mode} = (\nu - D - 1)\mathbf{S} \tag{2.137}$$

Note that the mode only exists if $\nu > D + 1$.

If $D = 1$, the Wishart reduces to the Gamma distribution:

$$\text{Wi}(\lambda|s^{-1}, \nu) = \text{Ga}(\lambda|\text{shape} = \frac{\nu}{2}, \text{rate} = \frac{1}{2s}) \tag{2.138}$$

If $s = 2$, this reduces to the chi-squared distribution.

There is an interesting connection between the Wishart distribution and the Gaussian. In particular, let $\boldsymbol{x}_n \sim \mathcal{N}(0, \boldsymbol{\Sigma})$. One can show that the scatter matrix, $\mathbf{S} = \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\intercal$, has a Wishart distribution: $\mathbf{S} \sim \text{Wi}(\boldsymbol{\Sigma}, N)$.

2.2.8.6 Inverse Wishart distribution

If $\lambda \sim \text{Ga}(a, b)$, then that $\frac{1}{\lambda} \sim \text{IG}(a, b)$. Similarly, if $\boldsymbol{\Sigma}^{-1} \sim \text{Wi}(\mathbf{S}^{-1}, \nu)$ then $\boldsymbol{\Sigma} \sim \text{IW}(\mathbf{S}, \nu + D + 1)$, where IW is the **inverse Wishart**, the multidimensional generalization of the inverse Gamma. It is defined as follows, for $\nu > D - 1$ and $\mathbf{S} \succ 0$:

$$\text{IW}(\boldsymbol{\Sigma}|\mathbf{S}, \nu) = \frac{1}{Z}|\boldsymbol{\Sigma}|^{-(\nu+D+1)/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{S}\boldsymbol{\Sigma}^{-1})\right) \tag{2.139}$$

$$Z_{\text{IW}} = |\mathbf{S}|^{-\nu/2} 2^{\nu D/2} \Gamma_D(\nu/2) \tag{2.140}$$

One can show that the distribution has these properties

$$\text{mean} = \frac{\mathbf{S}}{\nu - D - 1}, \ \text{mode} = \frac{\mathbf{S}}{\nu + D + 1} \tag{2.141}$$

If $D = 1$, this reduces to the inverse Gamma:

$$\text{IW}(\sigma^2|s^{-1}, \nu) = \text{IG}(\sigma^2|\nu/2, s/2) \tag{2.142}$$

If $s = 1$, this reduces to the inverse chi-squared distribution.

2.3 The exponential family

In this section, we define the **exponential family**, which includes many common probability distributions. The exponential family plays a crucial role in statistics and machine learning, for various reasons, including the following:

- The exponential family is the unique family of distributions that has maximum entropy (and hence makes the least set of assumptions) subject to some user-chosen constraints, as discussed in Section 2.3.7.
- The exponential family is at the core of GLMs, as discussed in Section 15.1.
- The exponential family is at the core of variational inference, as discussed in Chapter 10.
- Under certain regularity conditions, the exponential family is the only family of distributions with finite-sized sufficient statistics, as discussed in Section 2.3.5.
- All members of the exponential family have a **conjugate prior** [DY79], which simplifies Bayesian inference of the parameters, as discussed in Section 3.2.

2.3.1 Definition

Consider a family of probability distributions parameterized by $\boldsymbol{\eta} \in \mathbb{R}^K$ with fixed support over $\mathcal{X}^D \subseteq \mathbb{R}^D$. We say that the distribution $p(\boldsymbol{x}|\boldsymbol{\eta})$ is in the **exponential family** if its density can be written in the following way:

$$p(\boldsymbol{x}|\boldsymbol{\eta}) \triangleq \frac{1}{Z(\boldsymbol{\eta})} h(\boldsymbol{x}) \exp[\boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\boldsymbol{x})] = h(\boldsymbol{x}) \exp[\boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\boldsymbol{x}) - A(\boldsymbol{\eta})] \tag{2.143}$$

where $h(\boldsymbol{x})$ is a scaling constant (also known as the **base measure**, often 1), $\mathcal{T}(\boldsymbol{x}) \in \mathbb{R}^K$ are the **sufficient statistics**, $\boldsymbol{\eta}$ are the **natural parameters** or **canonical parameters**, $Z(\boldsymbol{\eta})$ is a normalization constant known as the **partition function**, and $A(\boldsymbol{\eta}) = \log Z(\boldsymbol{\eta})$ is the **log partition function**. In Section 2.3.3, we show that A is a convex function over the convex set $\Omega \triangleq \{\boldsymbol{\eta} \in \mathbb{R}^K : A(\boldsymbol{\eta}) < \infty\}$.

It is convenient if the natural parameters are independent of each other. Formally, we say that an exponential family is **minimal** if there is no $\boldsymbol{\eta} \in \mathbb{R}^K \setminus \{0\}$ such that $\boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\boldsymbol{x}) = 0$. This last condition can be violated in the case of multinomial distributions, because of the sum to one constraint on the parameters; however, it is easy to reparameterize the distribution using $K-1$ independent parameters, as we show below.

Equation (2.143) can be generalized by defining $\boldsymbol{\eta} = f(\boldsymbol{\phi})$, where $\boldsymbol{\phi}$ is some other, possibly smaller, set of parameters. In this case, the distribution has the form

$$p(\boldsymbol{x}|\boldsymbol{\phi}) = h(\boldsymbol{x}) \exp[f(\boldsymbol{\phi})^\mathsf{T} \mathcal{T}(\boldsymbol{x}) - A(f(\boldsymbol{\phi}))] \tag{2.144}$$

If the mapping from $\boldsymbol{\phi}$ to $\boldsymbol{\eta}$ is nonlinear, we call this a **curved exponential family**. If $\boldsymbol{\eta} = f(\boldsymbol{\phi}) = \boldsymbol{\phi}$, the model is said to be in **canonical form**. If, in addition, $\mathcal{T}(\boldsymbol{x}) = \boldsymbol{x}$, we say this is a **natural exponential family** or **NEF**. In this case, it can be written as

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = h(\boldsymbol{x}) \exp[\boldsymbol{\eta}^\mathsf{T} \boldsymbol{x} - A(\boldsymbol{\eta})] \tag{2.145}$$

We define the **moment parameters** as the mean of the sufficient statistics vector:

$$\boldsymbol{m} = \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})\right] \tag{2.146}$$

We will see some examples below.

2.3.2 Examples

In this section, we consider some common examples of distributions in the exponential family. Each corresponds to a different way of defining $h(\boldsymbol{x})$ and $\mathcal{T}(\boldsymbol{x})$ (since Z and hence A is derived from knowing h and $\mathcal{T}$).

2.3.2.1 Bernoulli distribution

The Bernoulli distribution can be written in exponential family form as follows:

$$\begin{aligned} \mathrm{Ber}(x|\mu) &= \mu^x (1-\mu)^{1-x} && (2.147)\\ &= \exp[x \log(\mu) + (1-x)\log(1-\mu)] && (2.148)\\ &= \exp[\mathcal{T}(x)^\mathsf{T} \boldsymbol{\eta}] && (2.149) \end{aligned}$$

where $\mathcal{T}(x) = [\mathbb{I}(x=1), \mathbb{I}(x=0)]$, $\boldsymbol{\eta} = [\log(\mu), \log(1-\mu)]$, and μ is the mean parameter. However, this is an **over-complete representation** since there is a linear dependendence between the features. We can see this as follows:

$$\mathbf{1}^\mathsf{T}\mathcal{T}(x) = \mathbb{I}(x=0) + \mathbb{I}(x=1) = 1 \tag{2.150}$$

If the representation is overcomplete, $\boldsymbol{\eta}$ is not uniquely identifiable. It is common to use a **minimal representation**, which means there is a unique $\boldsymbol{\eta}$ associated with the distribution. In this case, we can just define

$$\text{Ber}(x|\mu) = \exp\left[x \log\left(\frac{\mu}{1-\mu}\right) + \log(1-\mu)\right] \tag{2.151}$$

We can put this into exponential family form by defining

$$\eta = \log\left(\frac{\mu}{1-\mu}\right) \tag{2.152}$$

$$\mathcal{T}(x) = x \tag{2.153}$$

$$A(\eta) = -\log(1-\mu) = \log(1+e^\eta) \tag{2.154}$$

$$h(x) = 1 \tag{2.155}$$

We can recover the mean parameter μ from the canonical parameter η using

$$\mu = \sigma(\eta) = \frac{1}{1+e^{-\eta}} \tag{2.156}$$

which we recognize as the logistic (sigmoid) function.

2.3.2.2 Categorical distribution

We can represent the discrete distribution with K categories as follows (where $x_k = \mathbb{I}(x=k)$):

$$\text{Cat}(x|\boldsymbol{\mu}) = \prod_{k=1}^K \mu_k^{x_k} = \exp\left[\sum_{k=1}^K x_k \log \mu_k\right] \tag{2.157}$$

$$= \exp\left[\sum_{k=1}^{K-1} x_k \log \mu_k + \left(1 - \sum_{k=1}^{K-1} x_k\right) \log(1 - \sum_{k=1}^{K-1} \mu_k)\right] \tag{2.158}$$

$$= \exp\left[\sum_{k=1}^{K-1} x_k \log\left(\frac{\mu_k}{1-\sum_{j=1}^{K-1}\mu_j}\right) + \log(1 - \sum_{k=1}^{K-1} \mu_k)\right] \tag{2.159}$$

$$= \exp\left[\sum_{k=1}^{K-1} x_k \log\left(\frac{\mu_k}{\mu_K}\right) + \log \mu_K\right] \tag{2.160}$$

where $\mu_K = 1 - \sum_{k=1}^{K-1} \mu_k$. We can write this in exponential family form as follows:

$$\text{Cat}(x|\boldsymbol{\eta}) = \exp(\boldsymbol{\eta}^\mathsf{T}\mathcal{T}(\boldsymbol{x}) - A(\boldsymbol{\eta})) \tag{2.161}$$

$$\boldsymbol{\eta} = [\log \frac{\mu_1}{\mu_K}, \ldots, \log \frac{\mu_{K-1}}{\mu_K}] \tag{2.162}$$

$$A(\boldsymbol{\eta}) = -\log(\mu_K) \tag{2.163}$$

$$\mathcal{T}(x) = [\mathbb{I}(x = 1), \ldots, \mathbb{I}(x = K - 1)] \tag{2.164}$$

$$h(x) = 1 \tag{2.165}$$

We can recover the mean parameters from the canonical parameters using

$$\mu_k = \frac{e^{\eta_k}}{1 + \sum_{j=1}^{K-1} e^{\eta_j}} \tag{2.166}$$

If we define $\eta_K = 0$, we can rewrite this as follows:

$$\mu_k = \frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}} \tag{2.167}$$

for $k = 1 : K$. Hence $\boldsymbol{\mu} = \text{softmax}(\boldsymbol{\eta})$, where softmax is the softmax or multinomial logit function in Equation (15.131). From this, we find

$$\mu_K = 1 - \frac{\sum_{k=1}^{K-1} e^{\eta_k}}{1 + \sum_{k=1}^{K-1} e^{\eta_k}} = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\eta_k}} \tag{2.168}$$

and hence

$$A(\boldsymbol{\eta}) = -\log(\mu_K) = \log\left(\sum_{k=1}^{K} e^{\eta_k}\right) \tag{2.169}$$

2.3.2.3 Univariate Gaussian

The univariate Gaussian is usually written as follows:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp[-\frac{1}{2\sigma^2}(x - \mu)^2] \tag{2.170}$$

$$= \frac{1}{(2\pi)^{\frac{1}{2}}} \exp[\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 - \log\sigma] \tag{2.171}$$

We can put this in exponential family form by defining

$$\boldsymbol{\eta} = \begin{pmatrix} \mu/\sigma^2 \\ -\frac{1}{2\sigma^2} \end{pmatrix} \tag{2.172}$$

$$\mathcal{T}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \tag{2.173}$$

$$A(\boldsymbol{\eta}) = \frac{\mu^2}{2\sigma^2} + \log\sigma = \frac{-\eta_1^2}{4\eta_2} - \frac{1}{2}\log(-2\eta_2) \tag{2.174}$$

$$h(x) = \frac{1}{\sqrt{2\pi}} \tag{2.175}$$

The moment parameters are

$$\boldsymbol{m} = [\mu, \mu^2 + \sigma^2] \tag{2.176}$$

2.3.2.4 Univariate Gaussian with fixed variance

If we fix $\sigma^2 = 1$, we can write the Gaussian as a natural exponential family, by defining

$$\eta = \mu \tag{2.177}$$
$$\mathcal{T}(x) = x \tag{2.178}$$
$$A(\mu) = \frac{\mu^2}{2\sigma^2} + \log \sigma = \frac{\mu^2}{2} \tag{2.179}$$
$$h(x) = \frac{1}{\sqrt{2\pi}} \exp[-\frac{x^2}{2}] = \mathcal{N}(x|0,1) \tag{2.180}$$

Note that this in example, the base measure $h(x)$ is not constant.

2.3.2.5 Multivariate Gaussian

It is common to parameterize the multivariate normal (MVN) in terms of the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. The corresponding pdf is given by

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}\sqrt{\det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \tag{2.181}$$
$$= c \exp\left(\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right) \tag{2.182}$$
$$c \triangleq \frac{\exp(-\frac{1}{2}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})}{(2\pi)^{D/2}\sqrt{\det(\boldsymbol{\Sigma})}} \tag{2.183}$$

However, we can also represent the Gaussian using **canonical parameters** or **natural parameters**, also called the **information form**:

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} \tag{2.184}$$
$$\boldsymbol{\xi} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \tag{2.185}$$
$$\mathcal{N}_c(\boldsymbol{x}|\boldsymbol{\xi}, \boldsymbol{\Lambda}) \triangleq c' \exp\left(\boldsymbol{x}^\mathsf{T}\boldsymbol{\xi} - \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{x}\right) \tag{2.186}$$
$$c' = \frac{\exp(-\frac{1}{2}\boldsymbol{\xi}^\mathsf{T}\boldsymbol{\Lambda}^{-1}\boldsymbol{\xi})}{(2\pi)^{D/2}\sqrt{\det(\boldsymbol{\Lambda}^{-1})}} \tag{2.187}$$

where we use the notation $\mathcal{N}_c()$ to distinguish it from the standard parameterization $\mathcal{N}()$. Here $\boldsymbol{\Lambda}$ is called the **precision matrix** and $\boldsymbol{\xi}$ is the precision-weighted mean vector.

We can convert this to exponential family notation as follows:

$$\mathcal{N}_c(\boldsymbol{x}|\boldsymbol{\xi},\boldsymbol{\Lambda}) = \underbrace{(2\pi)^{-D/2}}_{h(\boldsymbol{x})} \underbrace{\exp\left[\frac{1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}\boldsymbol{\xi}^\mathsf{T}\boldsymbol{\Lambda}^{-1}\boldsymbol{\xi}\right]}_{g(\boldsymbol{\eta})} \exp\left[-\frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\boldsymbol{\xi}\right] \tag{2.188}$$

$$= h(\boldsymbol{x})g(\boldsymbol{\eta})\exp\left[-\frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\boldsymbol{\xi}\right] \tag{2.189}$$

$$= h(\boldsymbol{x})g(\boldsymbol{\eta})\exp\left[-\frac{1}{2}(\sum_{ij} x_i x_j \Lambda_{ij}) + \boldsymbol{x}^\mathsf{T}\boldsymbol{\xi}\right] \tag{2.190}$$

$$= h(\boldsymbol{x})g(\boldsymbol{\eta})\exp\left[-\frac{1}{2}\mathrm{vec}(\boldsymbol{\Lambda})^\mathsf{T}\mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\mathsf{T}) + \boldsymbol{x}^\mathsf{T}\boldsymbol{\xi}\right] \tag{2.191}$$

$$= h(\boldsymbol{x})\exp\left[\boldsymbol{\eta}^\mathsf{T}\mathcal{T}(\boldsymbol{x}) - A(\boldsymbol{\eta})\right] \tag{2.192}$$

where

$$h(\boldsymbol{x}) = (2\pi)^{-D/2} \tag{2.193}$$

$$\boldsymbol{\eta} = [\boldsymbol{\xi}; -\frac{1}{2}\mathrm{vec}(\boldsymbol{\Lambda})] = [\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}; -\frac{1}{2}\mathrm{vec}(\boldsymbol{\Sigma}^{-1})] \tag{2.194}$$

$$\mathcal{T}(\boldsymbol{x}) = [\boldsymbol{x}; \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\mathsf{T})] \tag{2.195}$$

$$A(\boldsymbol{\eta}) = -\log g(\boldsymbol{\eta}) = -\frac{1}{2}\log|\boldsymbol{\Lambda}| + \frac{1}{2}\boldsymbol{\xi}^\mathsf{T}\boldsymbol{\Lambda}^{-1}\boldsymbol{\xi} \tag{2.196}$$

From this, we see that the mean (moment) parameters are given by

$$\boldsymbol{m} = \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})\right] = [\boldsymbol{\mu}; \boldsymbol{\mu}\boldsymbol{\mu}^\mathsf{T} + \boldsymbol{\Sigma}] \tag{2.197}$$

(Note that the above is not a minimal representation, since $\boldsymbol{\Lambda}$ is a symmetric matrix. We can convert to minimal form by working with the upper or lower half of each matrix.)

2.3.2.6 Non-examples

Not all distributions of interest belong to the exponential family. For example, the Student distribution (Section 2.2.2.3) does not belong, since its pdf (Equation (2.13)) does not have the required form. (However, there is a generalization, known as the **ϕ-exponential family** [Nau04; Tsa88] which does include the Student distribution.)

As a more subtle example, consider the uniform distribution, $Y \sim \mathrm{Unif}(\theta_1, \theta_2)$. The pdf has the form

$$p(y|\boldsymbol{\theta}) = \frac{1}{\theta_2 - \theta_1}\mathbb{I}\left(\theta_1 \leq y \leq \theta_2\right) \tag{2.198}$$

It is tempting to think this is in the exponential family, with $h(y) = 1$, $\mathcal{T}(y) = \mathbf{0}$, and $Z(\boldsymbol{\theta}) = \theta_2 - \theta_1$. However, the support of this distribution (i.e., the set of values $\mathcal{Y} = \{y : p(y) > 0\}$ depends on the parameters $\boldsymbol{\theta}$, which violates an assumption of the exponential family.

2.3.3 Log partition function is cumulant generating function

The first and second **cumulants** of a distribution are its mean $\mathbb{E}[X]$ and variance $\mathbb{V}[X]$, whereas the first and second moments are $\mathbb{E}[X]$ and $\mathbb{E}\left[X^2\right]$. We can also compute higher order cumulants (and moments). An important property of the exponential family is that derivatives of the log partition function can be used to generate all the **cumulants** of the sufficient statistics In particular, the first and second cumulants are given by

$$\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \mathbb{E}[\mathcal{T}(\boldsymbol{x})] \tag{2.199}$$

$$\nabla^2_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \mathrm{Cov}[\mathcal{T}(\boldsymbol{x})] \tag{2.200}$$

We prove this result below.

2.3.3.1 Derivation of the mean

For simplicity, we focus on the 1d case. For the first derivative we have

$$\frac{dA}{d\eta} = \frac{d}{d\eta}\left(\log \int \exp(\eta \mathcal{T}(x)) h(x) dx\right) \tag{2.201}$$

$$= \frac{\frac{d}{d\eta}\int \exp(\eta \mathcal{T}(x)) h(x) dx}{\int \exp(\eta \mathcal{T}(x)) h(x) dx} \tag{2.202}$$

$$= \frac{\int \mathcal{T}(x) \exp(\eta \mathcal{T}(x)) h(x) dx}{\exp(A(\eta))} \tag{2.203}$$

$$= \int \mathcal{T}(x) \exp(\eta \mathcal{T}(x) - A(\eta)) h(x) dx \tag{2.204}$$

$$= \int \mathcal{T}(x) p(x) dx = \mathbb{E}[\mathcal{T}(x)] \tag{2.205}$$

For example, consider the Bernoulli distribution. We have $A(\eta) = \log(1 + e^{\eta})$, so the mean is given by

$$\frac{dA}{d\eta} = \frac{e^{\eta}}{1 + e^{\eta}} = \frac{1}{1 + e^{-\eta}} = \sigma(\eta) = \mu \tag{2.206}$$

2.3.3.2 Derivation of the variance

For simplicity, we focus on the 1d case. For the second derivative we have

$$\frac{d^2 A}{d\eta^2} = \frac{d}{d\eta} \int \mathcal{T}(x) \exp(\eta \mathcal{T}(x) - A(\eta)) h(x) dx \tag{2.207}$$

$$= \int \mathcal{T}(x) \exp\left(\eta \mathcal{T}(x) - A(\eta)\right) h(x) (\mathcal{T}(x) - A'(\eta)) dx \tag{2.208}$$

$$= \int \mathcal{T}(x) p(x) (\mathcal{T}(x) - A'(\eta)) dx \tag{2.209}$$

$$= \int \mathcal{T}^2(x) p(x) dx - A'(\eta) \int \mathcal{T}(x) p(x) dx \tag{2.210}$$

$$= \mathbb{E}\left[\mathcal{T}^2(X)\right] - \mathbb{E}[\mathcal{T}(x)]^2 = \mathbb{V}[\mathcal{T}(x)] \tag{2.211}$$

where we used the fact that $A'(\eta) = \frac{dA}{d\eta} = \mathbb{E}[\mathcal{T}(x)]$. For example, for the Bernoulli distribution we have

$$\frac{d^2A}{d\eta^2} = \frac{d}{d\eta}(1+e^{-\eta})^{-1} = (1+e^{-\eta})^{-2}e^{-\eta} \tag{2.212}$$

$$= \frac{e^{-\eta}}{1+e^{-\eta}}\frac{1}{1+e^{-\eta}} = \frac{1}{e^{\eta}+1}\frac{1}{1+e^{-\eta}} = (1-\mu)\mu \tag{2.213}$$

2.3.3.3 Connection with the Fisher information matrix

In Section 2.4, we show that, under some regularity conditions, the **Fisher information matrix** is given by

$$\mathbf{F}(\eta) \triangleq \mathbb{E}_{p(\boldsymbol{x}|\eta)}\left[\nabla \log p(\boldsymbol{x}|\eta) \nabla \log p(\boldsymbol{x}|\eta)^{\mathsf{T}}\right] = -\mathbb{E}_{p(\boldsymbol{x}|\eta)}\left[\nabla^2_{\boldsymbol{\eta}} \log p(\boldsymbol{x}|\eta)\right] \tag{2.214}$$

Hence for an exponential family model we have

$$\mathbf{F}(\eta) = -\mathbb{E}_{p(\boldsymbol{x}|\eta)}\left[\nabla^2_{\boldsymbol{\eta}}(\boldsymbol{\eta}^{\mathsf{T}}\mathcal{T}(\boldsymbol{x}) - A(\boldsymbol{\eta}))\right] = \nabla^2_{\boldsymbol{\eta}} A(\eta) = \mathrm{Cov}\left[\mathcal{T}(\boldsymbol{x})\right] \tag{2.215}$$

Thus the Hessian of the log partition function is the same as the FIM, which is the same as the covariance of the sufficient statistics. See Section 2.4.5 for details.

2.3.4 Canonical (natural) vs mean (moment) parameters

Let Ω be the set of normalizable natural parameters:

$$\Omega \triangleq \{\boldsymbol{\eta} \in \mathbb{R}^K : Z(\boldsymbol{\eta}) < \infty\} \tag{2.216}$$

We say that an exponential family is **regular** if Ω is an open set. It can be shown that Ω is a convex set, and $A(\boldsymbol{\eta})$ is a convex function defined over this set.

In Section 2.3.3, we prove that the derivative of the log partition function is equal to the mean of the sufficient statistics, i.e.,

$$\boldsymbol{m} = \mathbb{E}[\mathcal{T}(\boldsymbol{x})] = \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) \tag{2.217}$$

The set of valid moment parameters is given by

$$\mathcal{M} = \{\boldsymbol{m} \in \mathbb{R}^K : \mathbb{E}_p[\mathcal{T}(\boldsymbol{x})] = \boldsymbol{m}\} \tag{2.218}$$

for some distribution p.

We have seen that we can convert from the natural parameters to the moment parameters using

$$\boldsymbol{m} = \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) \tag{2.219}$$

If the family is minimal, one can show that

$$\boldsymbol{\eta} = \nabla_{\boldsymbol{m}} A^*(\boldsymbol{m}) \tag{2.220}$$

where $A^*(\boldsymbol{m})$ is the convex conjugate of A:

$$A^*(\boldsymbol{m}) \triangleq \sup_{\boldsymbol{\eta} \in \Omega} \boldsymbol{\mu}^\mathsf{T} \boldsymbol{\eta} - A(\boldsymbol{\eta}) \tag{2.221}$$

Thus the pair of operators $(\nabla A, \nabla A^*)$ lets us go back and forth between the natural parameters $\boldsymbol{\eta} \in \Omega$ and the mean parameters $\boldsymbol{m} \in \mathcal{M}$.

For future reference, note that the Bregman divergences (Section 5.1.8) associated with A and A^* are as follows:

$$B_A(\boldsymbol{\lambda}_1 || \boldsymbol{\lambda}_2) = A(\boldsymbol{\lambda}_1) - A(\boldsymbol{\lambda}_2) - (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)^\mathsf{T} \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda}_2) \tag{2.222}$$

$$B_{A^*}(\boldsymbol{\mu}_1 || \boldsymbol{\mu}_2) = A(\boldsymbol{\mu}_1) - A(\boldsymbol{\mu}_2) - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\mathsf{T} \nabla_{\boldsymbol{\mu}} A(\boldsymbol{\mu}_2) \tag{2.223}$$

$$\tag{2.224}$$

2.3.5 MLE for the exponential family

The likelihood of an exponential family model has the form

$$p(\mathcal{D}|\boldsymbol{\eta}) = \left[\prod_{n=1}^{N_\mathcal{D}} h(\boldsymbol{x}_n)\right] \exp\left(\boldsymbol{\eta}^\mathsf{T}[\sum_{n=1}^{N_\mathcal{D}} \mathcal{T}(\boldsymbol{x}_n)] - N A(\boldsymbol{\eta})\right) \propto \exp\left[\boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\mathcal{D}) - N A(\boldsymbol{\eta})\right] \tag{2.225}$$

where $\mathcal{T}(\mathcal{D})$ are the sufficient statistics:

$$\mathcal{T}(\mathcal{D}) = [\sum_{n=1}^{N_\mathcal{D}} \mathcal{T}_1(\boldsymbol{x}_n), \ldots, \sum_{n=1}^{N_\mathcal{D}} \mathcal{T}_K(\boldsymbol{x}_n)] \tag{2.226}$$

For example, for the Bernoulli model we have $\mathcal{T}(\mathcal{D}) = [\sum_n \mathbb{I}(x_n = 1)]$, and for the univariate Gaussian, we have $\mathcal{T}(\mathcal{D}) = [\sum_n x_n, \sum_n x_n^2]$.

The **Pitman-Koopman-Darmois theorem** states that, under certain regularity conditions, the exponential family is the only family of distributions with finite sufficient statistics. (Here, finite means a size independent of the size of the data set.) In other words, for an exponential family with natural parameters $\boldsymbol{\eta}$, we have

$$p(\mathcal{D}|\boldsymbol{\eta}) = p(\mathcal{T}(\mathcal{D})|\boldsymbol{\eta}) \tag{2.227}$$

We now show how to use this result to compute the MLE. The log likelihood is given by

$$\log p(\mathcal{D}|\boldsymbol{\eta}) = \boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\mathcal{D}) - N_\mathcal{D} A(\boldsymbol{\eta}) + \text{const} \tag{2.228}$$

Since $-A(\boldsymbol{\eta})$ is concave in $\boldsymbol{\eta}$, and $\boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\mathcal{D})$ is linear in $\boldsymbol{\eta}$, we see that the log likelihood is concave, and hence has a unique global maximum. To derive this maximum, we use the fact (shown in Section 2.3.3) that the derivative of the log partition function yields the expected value of the sufficient statistic vector:

$$\nabla_{\boldsymbol{\eta}} \log p(\mathcal{D}|\boldsymbol{\eta}) = \nabla_{\boldsymbol{\eta}} \boldsymbol{\eta}^\mathsf{T} \mathcal{T}(\mathcal{D}) - N_\mathcal{D} \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \mathcal{T}(\mathcal{D}) - N_\mathcal{D} \mathbb{E}[\mathcal{T}(\boldsymbol{x})] \tag{2.229}$$

For a single data case, this becomes

$$\nabla_{\boldsymbol{\eta}} \log p(\boldsymbol{x}|\boldsymbol{\eta}) = \mathcal{T}(\boldsymbol{x}) - \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})\right] \tag{2.230}$$

Setting the gradient in Equation (2.229) to zero, we see that at the MLE, the empirical average of the sufficient statistics must equal the model's theoretical expected sufficient statistics, i.e., $\hat{\boldsymbol{\eta}}$ must satisfy

$$\mathbb{E}\left[\mathcal{T}(\boldsymbol{x})\right] = \frac{1}{N_{\mathcal{D}}} \sum_{n=1}^{N_{\mathcal{D}}} \mathcal{T}(\boldsymbol{x}_n) \tag{2.231}$$

This is called **moment matching**. For example, in the Bernoulli distribution, we have $\mathcal{T}(x) = \mathbb{I}(X = 1)$, so the MLE satisfies

$$\mathbb{E}\left[\mathcal{T}(x)\right] = p(X = 1) = \mu = \frac{1}{N_{\mathcal{D}}} \sum_{n=1}^{N_{\mathcal{D}}} \mathbb{I}(x_n = 1) \tag{2.232}$$

2.3.6 Exponential dispersion family

In this section, we consider a slight extension of the natural exponential family known as the **exponential dispersion family**. This will be useful when we discuss GLMs in Section 15.1. For a scalar variable, this has the form

$$p(x|\eta, \sigma^2) = h(x, \sigma^2) \exp\left[\frac{\eta x - A(\eta)}{\sigma^2}\right] \tag{2.233}$$

Here σ^2 is called the **dispersion parameter**. For fixed σ^2, this is a natural exponential family.

2.3.7 Maximum entropy derivation of the exponential family

Suppose we want to find a distribution $p(\boldsymbol{x})$ to describe some data, where all we know are the expected values (F_k) of certain features or functions $f_k(\boldsymbol{x})$:

$$\int d\boldsymbol{x}\, p(\boldsymbol{x}) f_k(\boldsymbol{x}) = F_k \tag{2.234}$$

For example, f_1 might compute x, f_2 might compute x^2, making F_1 the empirical mean and F_2 the empirical second moment. Our prior belief in the distribution is $q(x)$.

To formalize what we mean by "least number of assumptions", we will search for the distribution that is as close as possible to our prior $q(\boldsymbol{x})$, in the sense of KL divergence (Section 5.1), while satisfying our constraints.

If we use a uniform prior, $q(\boldsymbol{x}) \propto 1$, minimizing the KL divergence is equivalent to maximizing the entropy (Section 5.2). The result is called a **maximum entropy model**.

To minimize KL subject to the constraints in Equation (2.234), and the constraint that $p(\boldsymbol{x}) \geq 0$ and $\sum_{\boldsymbol{x}} p(\boldsymbol{x}) = 1$, we need to use Lagrange multipliers. The Lagrangian is given by

$$J(p, \boldsymbol{\lambda}) = -\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} + \lambda_0 \left(1 - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\right) + \sum_k \lambda_k \left(F_k - \sum_{\boldsymbol{x}} p(\boldsymbol{x}) f_k(\boldsymbol{x})\right) \tag{2.235}$$

We can use the calculus of variations to take derivatives wrt the function p, but we will adopt a simpler approach and treat $\boldsymbol{p}$ as a fixed length vector (since we are assuming that $\boldsymbol{x}$ is discrete). Then we have

$$\frac{\partial J}{\partial p_c} = -1 - \log \frac{p(x=c)}{q(x=c)} - \lambda_0 - \sum_k \lambda_k f_k(x=c) \tag{2.236}$$

Setting $\frac{\partial J}{\partial p_c} = 0$ for each c yields

$$p(\boldsymbol{x}) = \frac{q(\boldsymbol{x})}{Z} \exp\left(-\sum_k \lambda_k f_k(\boldsymbol{x})\right) \tag{2.237}$$

where we have defined $Z \triangleq e^{1+\lambda_0}$. Using the sum-to-one constraint, we have

$$1 = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) = \frac{1}{Z} \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \exp\left(-\sum_k \lambda_k f_k(\boldsymbol{x})\right) \tag{2.238}$$

Hence the normalization constant is given by

$$Z = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \exp\left(-\sum_k \lambda_k f_k(\boldsymbol{x})\right) \tag{2.239}$$

This has exactly the form of the exponential family, where $\boldsymbol{f}(\boldsymbol{x})$ is the vector of sufficient statistics, $-\boldsymbol{\lambda}$ are the natural parameters, and $q(\boldsymbol{x})$ is our base measure.

For example, if the features are $f_1(x) = x$ and $f_2(x) = x^2$, and we want to match the first and second moments, we get the Gaussian disribution.

2.4 Fisher information matrix (FIM)

In this section, we discuss an important quantity called the **Fisher information matrix**, which is related to the curvature of the log likelihood function. This has many applications, such as characterizing the asymptotic sampling distribution of the MLE, deriving Jeffreys' uninformative priors (Section 3.6.2) and in natural gradient descent.

2.4.1 Definition

The **score function** is defined to be the gradient of the log likelihood:

$$\boldsymbol{s}(\boldsymbol{\theta}) \triangleq \nabla \log p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{2.240}$$

The **Fisher information matrix** (**FIM**) is defined to be the covariance of the score function:

$$\mathbf{F}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{\theta})}\left[\nabla \log p(\boldsymbol{x}|\boldsymbol{\theta}) \nabla \log p(\boldsymbol{x}|\boldsymbol{\theta})^{\mathsf{T}}\right] \tag{2.241}$$

so the (i,j)'th entry has the form

$$F_{ij} = \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{\theta}}\left[\left(\frac{\partial}{\partial \theta_i} \log p(\boldsymbol{x}|\boldsymbol{\theta})\right)\left(\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{x}|\boldsymbol{\theta})\right)\right] \tag{2.242}$$

We give an interpretation of this quantity below.

2.4.2 Equivalence between the FIM and the Hessian of the NLL

In this section, we prove that the Fisher information matrix equals the expected Hessian of the negative log likelihood (NLL)

$$\text{NLL}(\boldsymbol{\theta}) = -\log p(\mathcal{D}|\boldsymbol{\theta}) \tag{2.243}$$

Since the Hessian measures the curvature of the likelihood, we see that the FIM tells us how well the likelihood function can identify the best set of parameters. (If a likelihood function is flat, we cannot infer anything about the parameters, but if it is a delta function at a single point, the best parameter vector will be uniquely determined.) Thus the FIM is intimately related to the frequentist notion of uncertainty of the MLE, which is captured by the variance we expect to see in the MLE if we were to compute it on multiple different datasets drawn from our model.

More precisely, we have the following theorem.

Theorem 2.4.1. *If $\log p(\boldsymbol{x}|\boldsymbol{\theta})$ is twice differentiable, and under certain regularity conditions, the FIM is equal to the expected Hessian of the NLL, i.e.,*

$$\mathbf{F}(\boldsymbol{\theta})_{ij} \triangleq \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{\theta}}\left[\left(\frac{\partial}{\partial\theta_i}\log p(\boldsymbol{x}|\boldsymbol{\theta})\right)\left(\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{x}|\boldsymbol{\theta})\right)\right] = \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{\theta}}\left[\frac{\partial^2}{\partial\theta_i\theta_j}\log p(\boldsymbol{x}|\boldsymbol{\theta})\right] \tag{2.244}$$

Before we prove this result, we establish the following important lemma.

Lemma 1. *The expected value of the score function is zero, i.e.,*

$$\mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{\theta})}\left[\nabla \log p(\boldsymbol{x}|\boldsymbol{\theta})\right] = \mathbf{0} \tag{2.245}$$

We will prove this lemma in the scalar case. First, note that since $\int p(x|\theta)dx = 1$, we have

$$\frac{\partial}{\partial\theta}\int p(x|\theta)dx = 0 \tag{2.246}$$

Combining this with the identity

$$\frac{\partial}{\partial\theta}p(x|\theta) = \left[\frac{\partial}{\partial\theta}\log p(x|\theta)\right]p(x|\theta) \tag{2.247}$$

we have

$$0 = \int \frac{\partial}{\partial\theta}p(x|\theta)dx = \int\left[\frac{\partial}{\partial\theta}\log p(x|\theta)\right]p(x|\theta)dx = \mathbb{E}\left[s(\theta)\right] \tag{2.248}$$

Now we return to the proof of our main theorem. For simplicity, we will focus on the scalar case, following the presentation of [Ric95, p263].

Proof. Taking derivatives of Equation (2.248), we have

$$0 = \frac{\partial}{\partial\theta}\int\left[\frac{\partial}{\partial\theta}\log p(x|\theta)\right]p(x|\theta)dx \tag{2.249}$$

$$= \int\left[\frac{\partial^2}{\partial\theta^2}\log p(x|\theta)\right]p(x|\theta)dx + \int\left[\frac{\partial}{\partial\theta}\log p(x|\theta)\right]\frac{\partial}{\partial\theta}p(x|\theta)dx \tag{2.250}$$

$$= \int\left[\frac{\partial^2}{\partial\theta^2}\log p(x|\theta)\right]p(x|\theta)dx + \int\left[\frac{\partial}{\partial\theta}\log p(x|\theta)\right]^2 p(x|\theta)dx \tag{2.251}$$

and hence

$$-\mathbb{E}_{x\sim\theta}\left[\frac{\partial^2}{\partial\theta^2}\log p(x|\theta)\right] = \mathbb{E}_{x\sim\theta}\left[\left(\frac{\partial}{\partial\theta}\log p(x|\theta)\right)^2\right] \tag{2.252}$$

as claimed. □

Now consider the Hessian of the NLL given N iid samples $\mathcal{D} = \{\boldsymbol{x}_n : n = 1 : N\}$:

$$H_{ij} \triangleq -\frac{\partial^2}{\partial\theta_i\theta_j}\log p(\mathcal{D}|\boldsymbol{\theta}) = -\sum_{n=1}^{N}\frac{\partial^2}{\partial\theta_i\theta_j}\log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{2.253}$$

From the above theorem, we have

$$\mathbb{E}_{p(\mathcal{D}|\boldsymbol{\theta})}\left[\mathbf{H}(\mathcal{D})|_{\boldsymbol{\theta}}\right] = N\mathbf{F}(\boldsymbol{\theta}) \tag{2.254}$$

We will use this result later in this chapter.

2.4.3 Examples

In this section, we give some simple examples of how to compute the FIM.

2.4.3.1 FIM for the Binomial

Suppose $x \sim \text{Bin}(n, \theta)$. The log likelihood for a single sample is

$$l(\theta|x) = x\log\theta + (n-x)\log(1-\theta) \tag{2.255}$$

The score function is just the gradient of the log-likelihood:

$$s(\theta|x) \triangleq \frac{d}{d\theta}l(\theta|x) = \frac{x}{\theta} - \frac{n-x}{1-\theta} \tag{2.256}$$

The gradient of the score function is

$$s'(\theta|x) = -\frac{x}{\theta^2} - \frac{n-x}{(1-\theta)^2} \tag{2.257}$$

Hence the Fisher information is given by

$$F(\theta) = \mathbb{E}_{x\sim\theta}\left[-s'(\theta|x)\right] = \frac{n\theta}{\theta^2} + \frac{n-n\theta}{(1-\theta)^2} = \frac{n}{\theta} + \frac{n}{1-\theta} = \frac{n}{\theta(1-\theta)} \tag{2.258}$$

2.4.3.2 FIM for the Gaussian

Consider a univariate Gaussian $p(x|\boldsymbol{\theta}) = \mathcal{N}(x|\mu, v)$. We have

$$\ell(\boldsymbol{\theta}) = \log p(x|\boldsymbol{\theta}) = -\frac{1}{2v}(x-\mu)^2 - \frac{1}{2}\log(v) - \frac{1}{2}\log(2\pi) \tag{2.259}$$

The partial derivatives are given by

$$\frac{\partial \ell}{\partial \mu} = (x-\mu)v^{-1}, \; \frac{\partial^2 \ell}{\partial \mu^2} = -v^{-1} \tag{2.260}$$

$$\frac{\partial \ell}{\partial v} = \frac{1}{2}v^{-2}(x-\mu)^2 - \frac{1}{2}v^{-1}, \; \frac{\partial \ell}{\partial v^2} = -v^{-3}(x-\mu)^2 + \frac{1}{2}v^{-2} \tag{2.261}$$

$$\frac{\partial \ell}{\partial \mu \partial v} = -v^{-2}(x-\mu) \tag{2.262}$$

and hence

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{pmatrix} \mathbb{E}\left[v^{-1}\right] & \mathbb{E}\left[v^{-2}(x-\mu)\right] \\ \mathbb{E}\left[v^{-2}(x-\mu)\right] & \mathbb{E}\left[v^{-3}(x-\mu)^2 - \frac{1}{2}v^{-2}\right] \end{pmatrix} = \begin{pmatrix} \frac{1}{v} & 0 \\ 0 & \frac{1}{2v^2} \end{pmatrix} \tag{2.263}$$

2.4.3.3 FIM for logistic regression

Consider ℓ_2-regularized binary logistic regression. The negative log joint has the following form:

$$\mathcal{E}(\boldsymbol{w}) = -\log[p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})p(\boldsymbol{w}|\lambda)] = -\boldsymbol{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\boldsymbol{y} + \sum_{n=1}^{N} \log(1 + e^{\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n}) + \frac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w} \tag{2.264}$$

The derivative has the form

$$\nabla_{\boldsymbol{w}} \mathcal{E}(\boldsymbol{w}) = -\mathbf{X}^\mathsf{T}\boldsymbol{y} + \mathbf{X}^\mathsf{T}\boldsymbol{s} + \lambda \boldsymbol{w} \tag{2.265}$$

where $s_n = \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n)$. The FIM is given by

$$\mathbf{F}(\boldsymbol{w}) = \mathbb{E}_{p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w},\lambda)}\left[\nabla^2 \mathcal{E}(\boldsymbol{w})\right] = \mathbf{X}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{X} + \lambda \mathbf{I} \tag{2.266}$$

where $\boldsymbol{\Lambda}$ is the $N \times N$ diagonal matrix with entries

$$\Lambda_{nn} = \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n)(1 - \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n)) \tag{2.267}$$

2.4.4 Approximating KL divergence using FIM

Mahalanobis distance based on the Fisher information can be viewed as an approximation to the KL divergence between two distributions, as we now show.

Let $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $p_{\boldsymbol{\theta}'}(\boldsymbol{x})$ be two distributions, where $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{\delta}$. We can measure how close the second distribution is to the first in terms their predictive distribution (as opposed to comparing $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ in parameter space) as follows:

$$D_{\mathbb{KL}}\left(p_{\boldsymbol{\theta}} \parallel p_{\boldsymbol{\theta}'}\right) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log p_{\boldsymbol{\theta}'}(\boldsymbol{x})\right] \tag{2.268}$$

Let us approximate this with a second order Taylor series expansion:

$$D_{\mathbb{KL}}\left(p_{\boldsymbol{\theta}} \parallel p_{\boldsymbol{\theta}'}\right) \approx -\boldsymbol{\delta}^\mathsf{T}\mathbb{E}\left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right] - \frac{1}{2}\boldsymbol{\delta}^\mathsf{T}\mathbb{E}\left[\nabla^2 \log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right]\boldsymbol{\delta} \tag{2.269}$$

Since the expected score function is zero (from Equation (2.245)), the first term vanishes, so we have

$$D_{\mathbb{KL}}\left(p_{\boldsymbol{\theta}} \parallel p_{\boldsymbol{\theta}'}\right) \approx \frac{1}{2}\boldsymbol{\delta}^{\mathsf{T}}\mathbf{F}(\boldsymbol{\theta})\boldsymbol{\delta} \tag{2.270}$$

where $\mathbf{F}$ is the FIM

$$\mathbf{F} = -\mathbb{E}\left[\nabla^2 \log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right] = \mathbb{E}\left[(\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x}))(\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x}))^{\mathsf{T}}\right] \tag{2.271}$$

This result is the basis of the **natural gradient** method discussed in Section 6.4.

2.4.5 Fisher information matrix for exponential family

In this section, we discuss how to derive the FIM for an exponential family distribution with natural parameters $\boldsymbol{\eta}$. Recall from Equation (2.199) that the gradient of the log partition function is the expected sufficient statistics

$$\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})\right] = \boldsymbol{m} \tag{2.272}$$

and from Equation (2.230) that the gradient of the log likelihood is the statistics minus their expected value:

$$\nabla_{\boldsymbol{\eta}} \log p(\boldsymbol{x}|\boldsymbol{\eta}) = \mathcal{T}(\boldsymbol{x}) - \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})\right] \tag{2.273}$$

Hence the FIM wrt the natural parameters $\mathbf{F}_{\boldsymbol{\eta}}$ is given by

$$(\mathbf{F}_{\boldsymbol{\eta}})_{ij} = \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{\eta})}\left[\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{\eta})}{\partial \eta_i}\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{\eta})}{\partial \eta_j}\right] \tag{2.274}$$

$$= \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{\eta})}\left[(\mathcal{T}(\boldsymbol{x})_i - m_i)(\mathcal{T}(\boldsymbol{x})_j - m_j)\right] \tag{2.275}$$

$$= \mathrm{Cov}\left[\mathcal{T}(\boldsymbol{x})_i, \mathcal{T}(\boldsymbol{x})_j\right] \tag{2.276}$$

or, in short,

$$\mathbf{F}_{\boldsymbol{\eta}} = \mathrm{Cov}\left[\mathcal{T}(\boldsymbol{x})\right] \tag{2.277}$$

Sometimes we need to compute the Fisher wrt the moment parameters $\boldsymbol{m}$:

$$(\mathbf{F}_{\boldsymbol{m}})_{ij} = \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{m})}\left[\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{\eta})}{\partial m_i}\frac{\partial \log p(\boldsymbol{x}|\boldsymbol{\eta})}{\partial m_j}\right] \tag{2.278}$$

From the chain rule we have

$$\frac{\partial \log p(x)}{\partial \alpha} = \frac{\partial \log p(x)}{\partial \beta}\frac{\partial \beta}{\partial \alpha} \tag{2.279}$$

and hence

$$\mathbf{F}_{\alpha} = \frac{\partial \boldsymbol{\beta}^{\mathsf{T}}}{\partial \boldsymbol{\alpha}}\mathbf{F}_{\boldsymbol{\beta}}\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\alpha}} \tag{2.280}$$

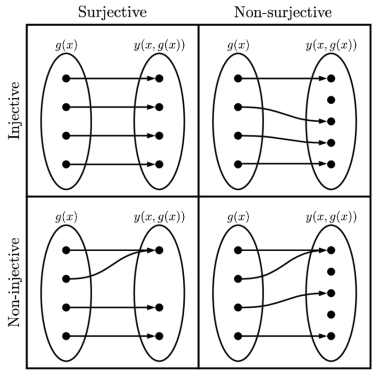


Figure 2.10: Illustration of injective and surjective functions.

Using the log trick

$$\nabla \mathbb{E}_{p(\boldsymbol{x})}\left[f(\boldsymbol{x})\right] = \mathbb{E}_{p(\boldsymbol{x})}\left[f(\boldsymbol{x}) \nabla \log p(\boldsymbol{x})\right] \tag{2.281}$$

and Equation (2.273) we have

$$\frac{\partial m_i}{\partial \eta_j} = \frac{\partial \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})_i\right]}{\partial \eta_j} = \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})_i \frac{\partial \log p(\boldsymbol{x}|\boldsymbol{\eta})}{\partial \eta_j}\right] = \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})_i (\mathcal{T}(\boldsymbol{x})_j - m_j)\right] \tag{2.282}$$

$$= \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})_i \mathcal{T}(\boldsymbol{x})_j\right] - \mathbb{E}\left[\mathcal{T}(\boldsymbol{x})_i\right] m_j = \mathrm{Cov}\left[\mathcal{T}(\boldsymbol{x})_i \mathcal{T}(\boldsymbol{x})_j\right] = (\mathbf{F}_{\boldsymbol{\eta}})_{ij} \tag{2.283}$$

and hence

$$\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{m}} = \mathbf{F}_{\boldsymbol{\eta}}^{-1} \tag{2.284}$$

so

$$\mathbf{F}_{\boldsymbol{m}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{m}}^{\mathsf{T}} \mathbf{F}_{\boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{m}} = \mathbf{F}_{\boldsymbol{\eta}}^{-1} \mathbf{F}_{\boldsymbol{\eta}} \mathbf{F}_{\boldsymbol{\eta}}^{-1} = \mathbf{F}_{\boldsymbol{\eta}}^{-1} = \mathrm{Cov}\left[\mathcal{T}(\boldsymbol{x})\right]^{-1} \tag{2.285}$$

2.5 Transformations of random variables

Suppose $\boldsymbol{x} \sim p_x(\boldsymbol{x})$ is some random variable, and $\boldsymbol{y} = f(\boldsymbol{x})$ is some deterministic transformation of it. In this section, we discuss how to compute $p_y(\boldsymbol{y})$.

2.5.1 Invertible transformations (bijections)

Let f be a **bijection** that maps $\mathbb{R}^n$ to $\mathbb{R}^n$. (A bijection is a function that is **injective**, **surjective**, and **one-to-one**, as illustrated in Figure 2.10; this means that the function has a well-defined inverse.) Suppose we want to compute the pdf of $\boldsymbol{y} = f(\boldsymbol{x})$. The **change of variables** formula tells us that

$$p_y(\boldsymbol{y}) = p_x\left(f^{-1}(\boldsymbol{y})\right) \left| \det\left[\mathbf{J}_{f^{-1}}(\boldsymbol{y})\right] \right| \tag{2.286}$$

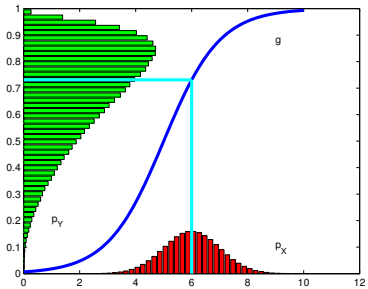


Figure 2.11: Example of the transformation of a density under a nonlinear transform. Note how the mode of the transformed distribution is not the transform of the original mode. Adapted from Exercise 1.4 of [Bis06]. Generated by bayes_change_of_var.ipynb.

where $\mathbf{J}_{f^{-1}}(\boldsymbol{y})$ is the Jacobian of the inverse mapping f^{-1} evaluated at $\boldsymbol{y}$, and $|\det \mathbf{J}|$ is the absolute value of the determinant of $\mathbf{J}$. In other words,

$$\mathbf{J}_{f^{-1}}(\boldsymbol{y}) = \begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \cdots & \frac{\partial x_1}{\partial y_n} \\ & \vdots & \\ \frac{\partial x_n}{\partial y_1} & \cdots & \frac{\partial x_n}{\partial y_n} \end{pmatrix} \tag{2.287}$$

If the Jacobian matrix is triangular, the determinant reduces to the product ot the diagonals:

$$\det(\mathbf{J}) = \prod_{i=1}^{n} \frac{\partial x_i}{\partial y_i} \tag{2.288}$$

2.5.2 Monte Carlo approximation

Sometime it is difficult to compute the Jacobian. In this case, we can make a Monte Carlo approximation, by drawing S samples $\boldsymbol{x}^s \sim p(\boldsymbol{x})$, computing $\boldsymbol{y}^s = f(\boldsymbol{x}^s)$, and then constructing the empirical pdf

$$p_{\mathcal{D}}(\boldsymbol{y}) = \frac{1}{S} \sum_{s=1}^{S} \delta(\boldsymbol{y} - \boldsymbol{y}^s) \tag{2.289}$$

For example, let $x \sim \mathcal{N}(6, 1)$ and $y = f(x)$, where $f(x) = \frac{1}{1+\exp(-x+5)}$. We can approximate $p(y)$ using Monte Carlo, as shown in Figure 2.11.

2.5.3 Probability integral transform

Suppose that X is a random variable with cdf P_X. Let $Y(X) = P_X(X)$ be a transformation of X. We now show that Y has a uniform distribution, a result known as the **probability integral transform** (PIT):

$$P_Y(y) = \Pr(Y \leq y) = \Pr(P_X(X) \leq y) \tag{2.290}$$

$$= \Pr(X \leq P_X^{-1}(y)) = P_X(P_X^{-1}(y)) = y \tag{2.291}$$

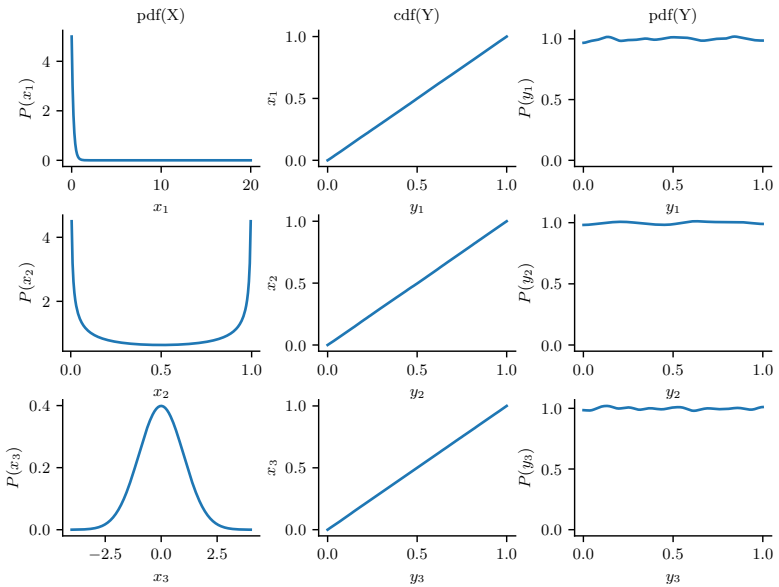


Figure 2.12: Illustration of the probability integral transform. Left column: 3 different pdf's for $p(X)$ from which we sample $x_n \sim p(x)$. Middle column: empirical cdf of $y_n = P_X(x_n)$. Right column: empirical pdf of $p(y_n)$ using a kernel density estimate. Adapted from Figure 11.17 of [MKL11]. Generated by ecdf_sample.ipynb.

For example, in Figure 2.12, we show various distributions with pdf's p_X on the left column. We sample from these, to get $x_n \sim p_x$. Next we compute the empirical cdf of $Y = P_X(X)$, by computing $y_n = P_X(x_n)$ and then sorting the values; the results, shown in the middle column, show that this distribution is uniform. We can also approximate the pdf of Y by using kernel density estimation; this is shown in the right column, and we see that it is (approximately) flat.

We can use the PIT to test if a set of samples come from a given distribution using the **Kolmogorov–Smirnov test**. To do this, we plot the empirical cdf of the samples and the theoretical cdf of the distribution, and compute the maximum distance between these two curves, as illustrated in Figure 2.13. Formally, the KS statistic is defined as

$$D_n = \max_x |P_n(x) - P(x)| \tag{2.292}$$

where n is the sample size, P_n is the empirical cdf, and P is the theoretical cdf. The value D_n should approach 0 (as $n \to \infty$) if the samples are drawn from P.

Another application of the PIT is to generate samples from a distribution: if we have a way to sample from a uniform distribution, $u_n \sim \text{Unif}(0, 1)$, we can convert this to samples from any other distribution with cdf P_X by setting $x_n = P_X^{-1}(u_n)$.

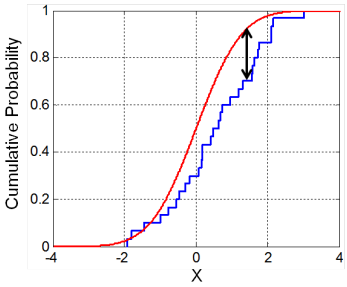


Figure 2.13: Illustration of the Kolmogorov–Smirnov statistic. The red line is a model CDF, the blue line is an empirical CDF, and the black arrow is the K–S statistic. From https://en.wikipedia.org/wiki/Kolmogorov_Smirnov_test. Used with kind permission of Wikipedia author Bscan.

2.6 Markov chains

Suppose that $\boldsymbol{x}_t$ captures all the relevant information about the state of the system. This means it is a **sufficient statistic** for predicting the future given the past, i.e.,

$$p(\boldsymbol{x}_{t+\tau}|\boldsymbol{x}_t, \boldsymbol{x}_{1:t-1}) = p(\boldsymbol{x}_{t+\tau}|\boldsymbol{x}_t) \tag{2.293}$$

for any $\tau \geq 0$. This is called the **Markov assumption**. In this case, we can write the joint distribution for any finite length sequence as follows:

$$p(\boldsymbol{x}_{1:T}) = p(\boldsymbol{x}_1)p(\boldsymbol{x}_2|\boldsymbol{x}_1)p(\boldsymbol{x}_3|\boldsymbol{x}_2)p(\boldsymbol{x}_4|\boldsymbol{x}_3)\ldots = p(\boldsymbol{x}_1)\prod_{t=2}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{2.294}$$

This is called a **Markov chain** or **Markov model**. Below we cover some of the basics of this topic; more details on the theory can be found in [Kun20].

2.6.1 Parameterization

In this section, we discuss how to represent a Markov model parametrically.

2.6.1.1 Markov transition kernels

The conditional distribution $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is called the **transition function**, **transition kernel** or **Markov kernel**. This is just a conditional distribution over the states at time t given the state at time $t-1$, and hence it satisfies the conditions $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \geq 0$ and $\int_{\boldsymbol{x}\in\mathcal{X}} dx\, p(\boldsymbol{x}_t = \boldsymbol{x}|\boldsymbol{x}_{t-1}) = 1$.

If we assume the transition function $p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})$ is independent of time, then the model is said to be **homogeneous**, **stationary**, or **time-invariant**. This is an example of **parameter tying**, since the same parameter is shared by multiple variables. This assumption allows us to model an arbitrary number of variables using a fixed number of parameters. We will make the time-invariant assumption throughout the rest of this section.

(a) (b)

Figure 2.14: State transition diagrams for some simple Markov chains. Left: a 2-state chain. Right: a 3-state left-to-right chain.

2.6.1.2 Markov transition matrices

In this section, we assume that the variables are discrete, so $X_t \in \{1, \ldots, K\}$. This is called a **finite-state Markov chain**. In this case, the conditional distribution $p(X_t|X_{t-1})$ can be written as a $K \times K$ matrix $\mathbf{A}$, known as the **transition matrix**, where $A_{ij} = p(X_t = j|X_{t-1} = i)$ is the probability of going from state i to state j. Each row of the matrix sums to one, $\sum_j A_{ij} = 1$, so this is called a **stochastic matrix**.

A stationary, finite-state Markov chain is equivalent to a **stochastic automaton**. It is common to visualize such automata by drawing a directed graph, where nodes represent states and arrows represent legal transitions, i.e., non-zero elements of $\mathbf{A}$. This is known as a **state transition diagram**. The weights associated with the arcs are the probabilities. For example, the following 2-state chain

$$\mathbf{A} = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \tag{2.295}$$

is illustrated in Figure 2.14(a). The following 3-state chain

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & 0 \\ 0 & A_{22} & A_{23} \\ 0 & 0 & 1 \end{pmatrix} \tag{2.296}$$

is illustrated in Figure 2.14(b). This is called a **left-to-right transition matrix**.

The A_{ij} element of the transition matrix specifies the probability of getting from i to j in one step. The n-step transition matrix $\mathbf{A}(n)$ is defined as

$$A_{ij}(n) \triangleq p(X_{t+n} = j|X_t = i) \tag{2.297}$$

which is the probability of getting from i to j in exactly n steps. Obviously $\mathbf{A}(1) = \mathbf{A}$. The **Chapman-Kolmogorov** equations state that

$$A_{ij}(m+n) = \sum_{k=1}^{K} A_{ik}(m) A_{kj}(n) \tag{2.298}$$

In words, the probability of getting from i to j in $m+n$ steps is just the probability of getting from i to k in m steps, and then from k to j in n steps, summed up over all k. We can write the above as

```
christians first inhabit wherein thou hast forgive if a man childless and of laying of core these
are the heavens shall reel to and fro to seek god they set their horses and children of israel
```

Figure 2.15: Example output from an 10-gram character-level Markov model trained on the King James Bible. The prefix "christians" is given to the model. Generated by ngram_character_demo.ipynb.

a matrix multiplication

$$\mathbf{A}(m+n) = \mathbf{A}(m)\mathbf{A}(n) \tag{2.299}$$

Hence

$$\mathbf{A}(n) = \mathbf{A}\ \mathbf{A}(n-1) = \mathbf{A}\ \mathbf{A}\ \mathbf{A}(n-2) = \cdots = \mathbf{A}^n \tag{2.300}$$

Thus we can simulate multiple steps of a Markov chain by "powering up" the transition matrix.

2.6.1.3 Higher-order Markov models

The first-order Markov assumption is rather strong. Fortunately, we can easily generalize first-order models to depend on the last n observations, thus creating a model of order (memory length) n:

$$p(\boldsymbol{x}_{1:T}) = p(\boldsymbol{x}_{1:n}) \prod_{t=n+1}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-n:t-1}) \tag{2.301}$$

This is called a **Markov model of order n**. If $n = 1$, this is called a **bigram model**, since we need to represent pairs of characters, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. If $n = 2$, this is called a **trigram model**, since we need to represent triples of characters, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2})$. In general, this is called an **n-gram model**.

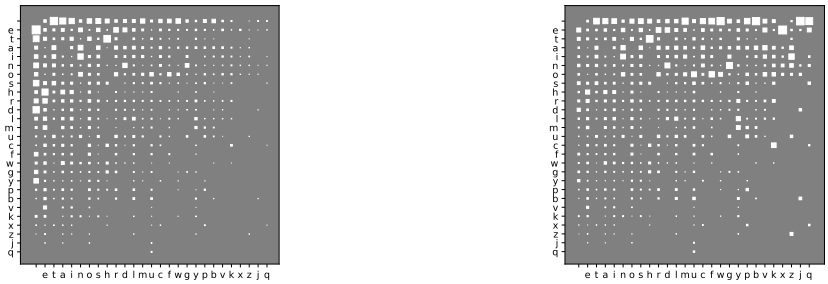
Note, however, we can always convert a higher order Markov model to a first order one by defining an augmented state space that contains the past n observations. For example, if $n = 2$, we define $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$ and use

$$p(\tilde{\boldsymbol{x}}_{1:T}) = p(\tilde{\boldsymbol{x}}_2) \prod_{t=3}^{T} p(\tilde{\boldsymbol{x}}_t|\tilde{\boldsymbol{x}}_{t-1}) = p(\boldsymbol{x}_1, \boldsymbol{x}_2) \prod_{t=3}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}) \tag{2.302}$$

Therefore we will just focus on first-order models throughout the rest of this section.

2.6.2 Application: Language modeling

One important application of Markov models is to create **language models** (**LM**), which are models which can generate (or score) a sequence of words. When we use a finite-state Markov model with a memory of length $m = n - 1$, it is called an **n-gram model**. For example, if $m = 1$, we get a **unigram model** (no dependence on previous words); if $m = 2$, we get a **bigram model** (depends on previous word); if $m = 3$, we get a **trigram model** (depends on previous two words); etc. See Figure 2.15 for some generated text.



(a) *(b)*

Figure 2.16: (a) **Hinton diagram** *showing character bigram counts as estimated from H. G. Wells' book* The Time Machine. *Characters are sorted in decreasing unigram frequency; the first one is a space character. The most frequent bigram is 'e-', where - represents space. (b) Same as (a) but each row is normalized across the columns. Generated by bigram_hinton_diagram.ipynb.*

These days, most LMs are built using recurrent neural nets (see Section 16.3.4), which have unbounded memory. However, simple n-gram models can still do quite well when trained with enough data [Che17].

Language models have various applications, such as priors for spelling correction (see Section 29.3.3) or automatic speech recognition. In addition, conditional language models can be used to generate sequences given inputs, such as mapping one language to another, or an image to a sequence, etc.

2.6.3 Parameter estimation

In this section, we discuss how to estimate the parameters of a Markov model.

2.6.3.1 Maximum likelihood estimation

The probability of any particular sequence of length T is given by

$$p(x_{1:T}|\boldsymbol{\theta}) = \pi(x_1)A(x_1, x_2)\dots A(x_{T-1}, x_T) \tag{2.303}$$

$$= \prod_{j=1}^{K}(\pi_j)^{\mathbb{I}(x_1=j)} \prod_{t=2}^{T}\prod_{j=1}^{K}\prod_{k=1}^{K}(A_{jk})^{\mathbb{I}(x_t=k, x_{t-1}=j)} \tag{2.304}$$

Hence the log-likelihood of a set of sequences $\mathcal{D} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$, where $\boldsymbol{x}_i = (x_{i1}, \dots, x_{i,T_i})$ is a sequence of length T_i, is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N}\log p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_j N_j^1 \log \pi_j + \sum_j\sum_k N_{jk}\log A_{jk} \tag{2.305}$$

where we define the following counts:

$$N_j^1 \triangleq \sum_{i=1}^{N}\mathbb{I}\left(x_{i1}=j\right), \quad N_{jk} \triangleq \sum_{i=1}^{N}\sum_{t=1}^{T_i-1}\mathbb{I}\left(x_{i,t}=j, x_{i,t+1}=k\right), \quad N_j = \sum_k N_{jk} \tag{2.306}$$

By adding Lagrange multipliers to enforce the sum to one constaints, one can show (see e.g., [Mur22, Sec 4.2.4]) that the MLE is given by the normalized counts:

$$\hat{\pi}_j = \frac{N_j^1}{\sum_{j'} N_{j'}^1}, \quad \hat{A}_{jk} = \frac{N_{jk}}{N_j} \tag{2.307}$$

We often replace N_j^1, which is how often symbol j is seen at the start of a sequence, by N_j, which is how often symbol j is seen anywhere in a sequence. This lets us estimate parameters from a single sequence.

The counts N_j are known as **unigram statistics**, and N_{jk} are known as **bigram statistics**. For example, Figure 2.16 shows some 2-gram counts for the characters $\{a, \ldots, z, -\}$ (where - represents space) as estimated from H. G. Wells' book *The Time Machine*.

2.6.3.2 Sparse data problem

When we try to fit n-gram models for large n, we quickly encounter problems with overfitting due to data sparsity. To see that, note that many of the estimated counts N_{jk} will be 0, since now j indexes over discrete contexts of size K^{n-1}, which will become increasingly rare. Even for bigram models ($n = 2$), problems can arise if K is large. For example, if we have $K \sim 50,000$ words in our vocabulary, then a bi-gram model will have about 2.5 billion free parameters, corresponding to all possible word pairs. It is very unlikely we will see all of these in our training data. However, we do not want to predict that a particular word string is totally impossible just because we happen not to have seen it in our training text — that would be a severe form of overfitting.[8]

A "brute force" solution to this problem is to gather lots and lots of data. For example, Google has fit n-gram models (for $n = 1:5$) based on one trillion words extracted from the web. Their data, which is over 100GB when uncompressed, is publically available.[9] Although such an approach can be surprisingly successful (as discussed in [HNP09]), it is rather unsatisfying, since humans are able to learn language from much less data (see e.g., [TX00]).

2.6.3.3 MAP estimation

A simple solution to the sparse data problem is to use MAP estimation with a uniform Dirichlet prior, $\mathbf{A}_{j:} \sim \text{Dir}(\alpha \mathbf{1})$. In this case, the MAP estimate becomes

$$\hat{A}_{jk} = \frac{N_{jk} + \alpha}{N_j + K\alpha} \tag{2.308}$$

If $\alpha = 1$, this is called **add-one smoothing**.

The main problem with add-one smoothing is that it assumes that all n-grams are equally likely, which is not very realistic. We discuss a more sophisticated approach, based on hierarchical Bayes, in Section 3.8.3.

8. A famous example of an improbable, but syntactically valid, English word string, due to Noam Chomsky [Cho57], is "colourless green ideas sleep furiously". We would not want our model to predict that this string is impossible. Even ungrammatical constructs should be allowed by our model with a certain probability, since people frequently violate grammatical rules, especially in spoken language.

9. See http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html for details.

(a) *(b)*

Figure 2.17: Some Markov chains. (a) A 3-state aperiodic chain. (b) A reducible 4-state chain.

2.6.4 Stationary distribution of a Markov chain

Suppose we continually draw consecutive samples from a Markov chain. In the case of a finite state space, we can think of this as "hopping" from one state to another. We will tend to spend more time in some states than others, depending on the transition graph. The long term distribution over states is known as the **stationary distribution** of the chain. In this section, we discuss some of the relevant theory. In Chapter 12, we discuss an important application, known as MCMC, which is a way to generate samples from hard-to-normalize probability distributions. In Supplementary Section 2.2 we consider Google's PageRank algorithm for ranking web pages, which also leverages the concept of stationary distributions.

2.6.4.1 What is a stationary distribution?

Let $A_{ij} = p(X_t = j|X_{t-1} = i)$ be the one-step transition matrix, and let $\pi_t(j) = p(X_t = j)$ be the probability of being in state j at time t.

If we have an initial distribution over states of $\boldsymbol{\pi}_0$, then at time 1 we have

$$\pi_1(j) = \sum_i \pi_0(i) A_{ij} \tag{2.309}$$

or, in matrix notation, $\boldsymbol{\pi}_1 = \boldsymbol{\pi}_0 \mathbf{A}$, where we have followed the standard convention of assuming $\boldsymbol{\pi}$ is a *row* vector, so we post-multiply by the transition matrix.

Now imagine iterating these equations. If we ever reach a stage where $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{A}$, then we say we have reached the **stationary distribution** (also called the **invariant distribution** or **equilibrium distribution**). Once we enter the stationary distribution, we will never leave.

For example, consider the chain in Figure 2.17(a). To find its stationary distribution, we write

$$\begin{pmatrix} \pi_1 & \pi_2 & \pi_3 \end{pmatrix} = \begin{pmatrix} \pi_1 & \pi_2 & \pi_3 \end{pmatrix} \begin{pmatrix} 1 - A_{12} - A_{13} & A_{12} & A_{13} \\ A_{21} & 1 - A_{21} - A_{23} & A_{23} \\ A_{31} & A_{32} & 1 - A_{31} - A_{32} \end{pmatrix} \tag{2.310}$$

Hence $\pi_1(A_{12} + A_{13}) = \pi_2 A_{21} + \pi_3 A_{31}$. In general, we have

$$\pi_i \sum_{j \neq i} A_{ij} = \sum_{j \neq i} \pi_j A_{ji} \tag{2.311}$$

In other words, the probability of being in state i times the net flow out of state i must equal the probability of being in each other state j times the net flow from that state into i. These are called the **global balance equations**. We can then solve these equations, subject to the constraint that $\sum_j \pi_j = 1$, to find the stationary distribution, as we discuss below.

2.6.4.2 Computing the stationary distribution

To find the stationary distribution, we can just solve the eigenvector equation $\mathbf{A}^\mathsf{T}\boldsymbol{v} = \boldsymbol{v}$, and then to set $\boldsymbol{\pi} = \boldsymbol{v}^\mathsf{T}$, where $\boldsymbol{v}$ is an eigenvector with eigenvalue 1. (We can be sure such an eigenvector exists, since $\mathbf{A}$ is a row-stochastic matrix, so $\mathbf{A1} = \mathbf{1}$; also recall that the eigenvalues of $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$ are the same.) Of course, since eigenvectors are unique only up to constants of proportionality, we must normalize $\boldsymbol{v}$ at the end to ensure it sums to one.

Note, however, that the eigenvectors are only guaranteed to be real-valued if all entries in the matrix are strictly positive, $A_{ij} > 0$ (and hence $A_{ij} < 1$, due to the sum-to-one constraint). A more general approach, which can handle chains where some transition probabilities are 0 or 1 (such as Figure 2.17(a)), is as follows. We have K constraints from $\boldsymbol{\pi}(\mathbf{I} - \mathbf{A}) = \mathbf{0}_{K\times 1}$ and 1 constraint from $\boldsymbol{\pi}\mathbf{1}_{K\times 1} = 1$. Hence we have to solve $\boldsymbol{\pi}\mathbf{M} = \boldsymbol{r}$, where $\mathbf{M} = [\mathbf{I} - \mathbf{A}, \mathbf{1}]$ is a $K \times (K+1)$ matrix, and $\boldsymbol{r} = [0, 0, \ldots, 0, 1]$ is a $1 \times (K+1)$ vector. However, this is overconstrained, so we will drop the last column of $\mathbf{I} - \mathbf{A}$ in our definition of $\mathbf{M}$, and drop the last 0 from $\boldsymbol{r}$. For example, for a 3 state chain we have to solve this linear system:

$$\begin{pmatrix}\pi_1 & \pi_2 & \pi_3\end{pmatrix}\begin{pmatrix}1-A_{11} & -A_{12} & 1\\ -A_{21} & 1-A_{22} & 1\\ -A_{31} & -A_{32} & 1\end{pmatrix} = \begin{pmatrix}0 & 0 & 1\end{pmatrix} \tag{2.312}$$

For the chain in Figure 2.17(a) we find $\boldsymbol{\pi} = [0.4, 0.4, 0.2]$. We can easily verify this is correct, since $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{A}$.

Unfortunately, not all chains have a stationary distribution, as we explain below.

2.6.4.3 When does a stationary distribution exist?

Consider the 4-state chain in Figure 2.17(b). If we start in state 4, we will stay there forever, since 4 is an **absorbing state**. Thus $\boldsymbol{\pi} = (0, 0, 0, 1)$ is one possible stationary distribution. However, if we start in 1 or 2, we will oscillate between those two states for ever. So $\boldsymbol{\pi} = (0.5, 0.5, 0, 0)$ is another possible stationary distribution. If we start in state 3, we could end up in either of the above stationary distributions with equal probability. The corresponding transition graph has two disjoint connected components.

We see from this example that a necessary condition to have a unique stationary distribution is that the state transition diagram be a singly connected component, i.e., we can get from any state to any other state. Such chains are called **irreducible**.

Now consider the 2-state chain in Figure 2.14(a). This is irreducible provided $\alpha, \beta > 0$. Suppose $\alpha = \beta = 0.9$. It is clear by symmetry that this chain will spend 50% of its time in each state. Thus $\boldsymbol{\pi} = (0.5, 0.5)$. But now suppose $\alpha = \beta = 1$. In this case, the chain will oscillate between the two states, but the long-term distribution on states depends on where you start from. If we start in state 1, then on every odd time step (1,3,5,...) we will be in state 1; but if we start in state 2, then on every odd time step we will be in state 2.

This example motivates the following definition. Let us say that a chain has a **limiting distribution** if $\pi_j = \lim_{n\to\infty} A_{ij}^n$ exists and is independent of the starting state i, for all j. If this holds, then the long-run distribution over states will be independent of the starting state:

$$p(X_t = j) = \sum_i p(X_0 = i) A_{ij}(t) \to \pi_j \text{ as } t \to \infty \tag{2.313}$$

Let us now characterize when a limiting distribution exists. Define the **period** of state i to be $d(i) \triangleq \text{gcd}\{t : A_{ii}(t) > 0\}$, where gcd stands for **greatest common divisor**, i.e., the largest integer that divides all the members of the set. For example, in Figure 2.17(a), we have $d(1) = d(2) = gcd(2, 3, 4, 6, ...) = 1$ and $d(3) = gcd(3, 5, 6, ...) = 1$. We say a state i is **aperiodic** if $d(i) = 1$. (A sufficient condition to ensure this is if state i has a self-loop, but this is not a necessary condition.) We say a chain is aperiodic if all its states are aperiodic. One can show the following important result:

Theorem 2.6.1. *Every irreducible (singly connected), aperiodic finite state Markov chain has a limiting distribution, which is equal to $\boldsymbol{\pi}$, its unique stationary distribution.*

A special case of this result says that every regular finite state chain has a unique stationary distribution, where a **regular** chain is one whose transition matrix satisfies $A_{ij}^n > 0$ for some integer n and all i, j, i.e., it is possible to get from any state to any other state in n steps. Consequently, after n steps, the chain could be in any state, no matter where it started. One can show that sufficient conditions to ensure regularity are that the chain be irreducible (singly connected) and that every state have a self-transition.

To handle the case of Markov chains whose state space is not finite (e.g, the countable set of all integers, or all the uncountable set of all reals), we need to generalize some of the earlier definitions. Since the details are rather technical, we just briefly state the main results without proof. See e.g., [GS92] for details.

For a stationary distribution to exist, we require irreducibility (singly connected) and aperiodicity, as before. But we also require that each state is **recurrent**, which means that you will return to that state with probability 1. As a simple example of a non-recurrent state (i.e., a **transient** state), consider Figure 2.17(b): states 3 is transient because one immediately leaves it and either spins around state 4 forever, or oscillates between states 1 and 2 forever. There is no way to return to state 3.

It is clear that any finite-state irreducible chain is recurrent, since you can always get back to where you started from. But now consider an example with an infinite state space. Suppose we perform a random walk on the integers, $\mathcal{X} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. Let $A_{i,i+1} = p$ be the probability of moving right, and $A_{i,i-1} = 1 - p$ be the probability of moving left. Suppose we start at $X_1 = 0$. If $p > 0.5$, we will shoot off to $+\infty$; we are not guaranteed to return. Similarly, if $p < 0.5$, we will shoot off to $-\infty$. So in both cases, the chain is not recurrent, even though it is irreducible. If $p = 0.5$, we can return to the initial state with probability 1, so the chain is recurrent. However, the distribution keeps spreading out over a larger and larger set of the integers, so the expected time to return is infinite. This prevents the chain from having a stationary distribution.

More formally, we define a state to be **non-null recurrent** if the expected time to return to this state is finite. We say that a state is **ergodic** if it is aperiodic, recurrent and non-null,. We say that a chain is ergodic if all its states are ergodic. With these definitions, we can now state our main theorem:

Theorem 2.6.2. *Every irreducible, ergodic Markov chain has a limiting distribution, which is equal to $\boldsymbol{\pi}$, its unique stationary distribution.*

This generalizes Theorem 2.6.1, since for irreducible finite-state chains, all states are recurrent and non-null.

2.6.4.4 Detailed balance

Establishing ergodicity can be difficult. We now give an alternative condition that is easier to verify.

We say that a Markov chain $\mathbf{A}$ is **time reversible** if there exists a distribution $\boldsymbol{\pi}$ such that

$$\pi_i A_{ij} = \pi_j A_{ji} \tag{2.314}$$

These are called the **detailed balance equations**. This says that the flow from i to j must equal the flow from j to i, weighted by the appropriate source probabilities.

We have the following important result.

Theorem 2.6.3. *If a Markov chain with transition matrix $\mathbf{A}$ is regular and satisfies the detailed balance equations wrt distribution $\boldsymbol{\pi}$, then $\boldsymbol{\pi}$ is a stationary distribution of the chain.*

Proof. To see this, note that

$$\sum_i \pi_i A_{ij} = \sum_i \pi_j A_{ji} = \pi_j \sum_i A_{ji} = \pi_j \tag{2.315}$$

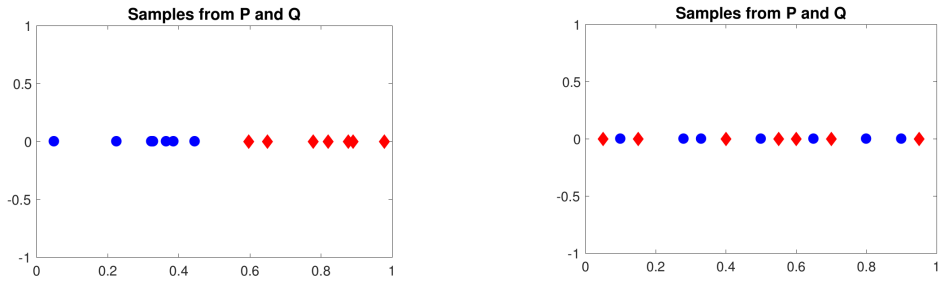
and hence $\boldsymbol{\pi} = \mathbf{A}\boldsymbol{\pi}$. □

Note that this condition is sufficient but not necessary (see Figure 2.17(a) for an example of a chain with a stationary distribution which does not satisfy detailed balance).

2.7 Divergence measures between probability distributions

In this section, we discuss various ways to compare two probability distributions, P and Q, defined on the same space. For example, suppose the distributions are defined in terms of samples, $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \sim P$ and $\mathcal{X}' = \{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_M\} \sim Q$. Determining if the samples come from the same distribution is known as a **two-sample test** (see Figure 2.18 for an illustration). This can be computed by defining some suitable **divergence metric** $D(P, Q)$ and comparing it to a threshold. (We use the term "divergence" rather than distance since we will not require D to be symmetric.) Alternatively, suppose P is an empirical distribution of data, and Q is the distribution induced by a model. We can check how well the model approximates the data by comparing $D(P, Q)$ to a threshold; this is called a **goodness-of-fit** test.

There are two main ways to compute the divergence between a pair of distributions: in terms of their difference, $P - Q$ (see e.g., [Sug+13]) or in terms of their ratio, P/Q (see e.g., [SSK12]). We briefly discuss both of these below. (Our presentation is based, in part, on [GSJ19].)



(a) *(b)*

Figure 2.18: Samples from two distributions which are (a) different and (b) similar. From a figure from [GSJ19]. Used with kind permission of Arthur Getton.

2.7.1 f-divergence

In this section, we compare distributions in terms of their density ratio $r(\boldsymbol{x}) = p(\boldsymbol{x})/q(\boldsymbol{x})$. In particular, consider the **f-divergence** [Mor63; AS66; Csi67; LV06; CS04], which is defined as follows:

$$D_f(p||q) = \int q(\boldsymbol{x}) f\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) d\boldsymbol{x} \tag{2.316}$$

where $f : \mathbb{R}_+ \to \mathbb{R}$ is a convex function satisfying $f(1) = 0$. From Jensen's inequality (Section 5.1.2.2), it follows that $D_f(p||q) \geq 0$, and obviously $D_f(p||p) = 0$, so D_f is a valid divergence. Below we discuss some important special cases of f-divergences. (Note that f-divergences are also called ϕ-divergences.)

2.7.1.1 KL divergence

Suppose we compute the f-divergence using $f(r) = r\log(r)$. In this case, we get a quantity called the **Kullback Leibler divergence**, defined as follows:

$$D_{\mathbb{KL}}(p \parallel q) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} \tag{2.317}$$

See Section 5.1 for more details.

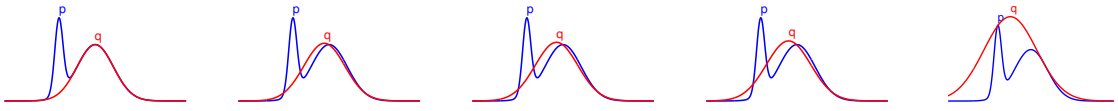
2.7.1.2 Alpha divergence

If $f(x) = \frac{4}{1-\alpha^2}(1 - x^{\frac{1+\alpha}{2}})$, the f-divergence becomes the the **alpha divergence** [Ama09], which is as follows:

$$D_\alpha^A(p||q) \triangleq \frac{4}{1-\alpha^2}\left(1 - \int p(\boldsymbol{x})^{(1+\alpha)/2} q(\boldsymbol{x})^{(1-\alpha)/2} d\boldsymbol{x}\right) \tag{2.318}$$

where we assume $\alpha \neq \pm 1$. Another common parameterization , and the one used by Minka in [Min05], is as follows:

$$D_\alpha^M(p||q) = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(\boldsymbol{x})^\alpha q(\boldsymbol{x})^{1-\alpha} d\boldsymbol{x}\right) \tag{2.319}$$



(a) $\alpha = -\infty$ (b) $\alpha = 0$ (c) $\alpha = 0.5$ (d) $\alpha = 1$ (e) $\alpha = \infty$

Figure 2.19: The Gaussian q which minimizes α-divergence to p (a mixture of two Gaussians), for varying α. From Figure 1 of [Min05]. Used with kind permission of Tom Minka.

This can be converted to Amari's notation using $D^A_{\alpha'} = D^M_\alpha$ where $\alpha' = 2\alpha - 1$. (We will use the Minka convention.)

We see from Figure 2.19 that as $\alpha \to -\infty$, q prefers to match one mode of p, whereas when $\alpha \to \infty$, q prefers to cover all of p. More precisely, one can show that as $\alpha \to 0$, the alpha-divergence tends towards $D_{\mathbb{KL}}(q \parallel p)$, and as $\alpha \to 1$, the alpha-divergence tends towards $D_{\mathbb{KL}}(p \parallel q)$. Also, when $\alpha = 0.5$, the alpha-divergence equals the Hellinger distance (Section 2.7.1.3).

2.7.1.3 Hellinger distance

The (squared) **Hellinger distance** is defined as follows:

$$D^2_H(p||q) \triangleq \frac{1}{2}\int \left(p(\boldsymbol{x})^{\frac{1}{2}} - q(\boldsymbol{x})^{\frac{1}{2}}\right)^2 d\boldsymbol{x} = 1 - \int \sqrt{p(\boldsymbol{x})q(\boldsymbol{x})}d\boldsymbol{x} \tag{2.320}$$

This is a valid distance metric, since it is symmetric, non-negative and satisfies the triangle inequality.

We see that this is equal (up to constant factors) to the f-divergence with $f(r) = (\sqrt{r} - 1)^2$, since

$$\int d\boldsymbol{x}\, q(\boldsymbol{x}) \left(\frac{p^{\frac{1}{2}}(\boldsymbol{x})}{q^{\frac{1}{2}}(\boldsymbol{x})} - 1\right)^2 = \int d\boldsymbol{x}\, q(\boldsymbol{x}) \left(\frac{p^{\frac{1}{2}}(\boldsymbol{x}) - q^{\frac{1}{2}}(\boldsymbol{x})}{q^{\frac{1}{2}}(\boldsymbol{x})}\right)^2 = \int d\boldsymbol{x}\, \left(p^{\frac{1}{2}}(\boldsymbol{x}) - q^{\frac{1}{2}}(\boldsymbol{x})\right)^2 \tag{2.321}$$

2.7.1.4 Chi-squared distance

The **chi-squared distance** χ^2 is defined by

$$\chi^2(p, q) \triangleq \frac{1}{2}\int \frac{(q(\boldsymbol{x}) - p(\boldsymbol{x}))^2}{q(\boldsymbol{x})} d\boldsymbol{x} \tag{2.322}$$

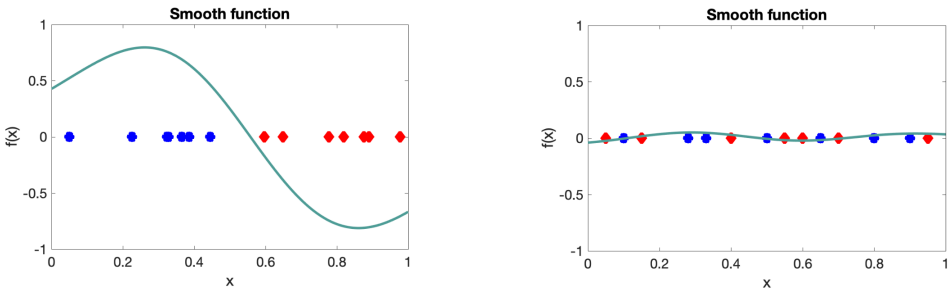
This is equal (up to constant factors) to an f-divergence where $f(r) = (r-1)^2$, since

$$\int d\boldsymbol{x}\, q(\boldsymbol{x}) \left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} - 1\right)^2 = \int d\boldsymbol{x}\, q(\boldsymbol{x}) \left(\frac{p(\boldsymbol{x}) - q(\boldsymbol{x})}{q(\boldsymbol{x})}\right)^2 = \int d\boldsymbol{x}\, \frac{1}{q(\boldsymbol{x})} (p(\boldsymbol{x}) - q(\boldsymbol{x}))^2 \tag{2.323}$$

2.7.2 Integral probability metrics

In this section, we compute the divergence between two distributions in terms of $P - Q$ using an **integral probability metric** or **IPM** [Sri+09]. This is defined as follows:

$$D_{\mathcal{F}}(P, Q) \triangleq \sup_{f \in \mathcal{F}} |\mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})] - \mathbb{E}_{q(\boldsymbol{x}')}[f(\boldsymbol{x}')]| \tag{2.324}$$



(a) (b)

Figure 2.20: A smooth witness function for comparing two distributions which are (a) different and (b) similar. From a figure from [GSJ19]. Used with kind permission of Arthur Getton.

where $\mathcal{F}$ is some class of "smooth" functions. The function f that maximizes the difference between these two expectations is called the **witness function**. See Figure 2.20 for an illustration.

There are several ways to define the function class $\mathcal{F}$. One approach is to use an RKHS, defined in terms of a positive definite kernel function; this gives rise to the method known as maximum mean discrepancy or MMD. See Section 2.7.3 for details.

Another approach is to define $\mathcal{F}$ to be the set of functions that have bounded Lipschitz constant, i.e., $\mathcal{F} = \{||f||_L \leq 1\}$, where

$$||f||_L = \sup_{\boldsymbol{x} \neq \boldsymbol{x}'} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{x}')|}{||\boldsymbol{x} - \boldsymbol{x}'||} \tag{2.325}$$

The IPM in this case is equal to the **Wasserstein-1 distance**

$$W_1(P, Q) \triangleq \sup_{||f||_L \leq 1} |\mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})] - \mathbb{E}_{q(\boldsymbol{x}')}[f(\boldsymbol{x}')]| \tag{2.326}$$

See Section 6.10.2.4 for details.

2.7.3 Maximum mean discrepancy (MMD)

In this section, we describe the **maximum mean discrepancy** or **MMD** method of [Gre+12], which defines a discrepancy measure $D(P, Q)$ using samples from the two distributions. The samples are compared using positive definite kernels (Section 18.2), which can handle high-dimensional inputs. This approach can be used to define two-sample tests, and to train implicit generative models (Section 26.2.4).

2.7.3.1 MMD as an IPM

The MMD is an integral probability metric (Section 2.7.2) of the form

$$\text{MMD}(P, Q; \mathcal{F}) = \sup_{f \in \mathcal{F}: ||f|| \leq 1} \left[\mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})] - \mathbb{E}_{q(\boldsymbol{x}')}[f(\boldsymbol{x}')]\right] \tag{2.327}$$

where $\mathcal{F}$ is an RKHS (Section 18.3.7.1) defined by a positive definite kernel function $\mathcal{K}$. We can represent functions in this set as an infinite sum of basis functions

$$f(\boldsymbol{x}) = \langle f, \phi(\boldsymbol{x})\rangle_{\mathcal{F}} = \sum_{l=1}^{\infty} f_l \phi_l(\boldsymbol{x}) \tag{2.328}$$

We restrict the set of witness functions f to be those that are in the unit ball of this RKHS, so $||f||_{\mathcal{F}}^2 = \sum_{l=1}^{\infty} f_l^2 \leq 1$.

By the linearity of expectation, we have

$$\mathbb{E}_{p(\boldsymbol{x})}\left[f(\boldsymbol{x})\right] = \langle f, \mathbb{E}_{p(\boldsymbol{x})}\left[\phi(\boldsymbol{x})\right]\rangle_{\mathcal{F}} = \langle f, \boldsymbol{\mu}_P\rangle_{\mathcal{F}} \tag{2.329}$$

where $\boldsymbol{\mu}_P$ is called the **kernel mean embedding** of distribution P [Mua+17]. Hence

$$\mathrm{MMD}(P, Q; \mathcal{F}) = \sup_{||f||\leq 1} \langle f, \boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\rangle_{\mathcal{F}} = \frac{\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q}{||\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q||} \tag{2.330}$$

since the unit vector $\boldsymbol{f}$ that maximizes the inner product is parallel to the difference in feature means.

To get some intuition, suppose $\boldsymbol{\phi}(x) = [x, x^2]$. In this case, the MMD computes the difference in the first two moments of the two distributions. This may not be enough to distinguish all possible distributions. However, using a Gaussian kernel is equivalent to comparing two infinitely large feature vectors, as we show in Section 18.2.3, and hence we are effectively comparing all the moments of the two distributions. Indeed, one can show that MMD=0 iff $P = Q$, provided we use a non-degenerate kernel.

2.7.3.2 Computing the MMD using the kernel trick

In this section, we describe how to compute Equation (2.330) in practice, given two sets of samples, $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ and $\mathcal{X}' = \{\boldsymbol{x}'_m\}_{m=1}^M$, where $\boldsymbol{x}_n \sim P$ and $\boldsymbol{x}'_m \sim Q$. Let $\boldsymbol{\mu}_P = \frac{1}{N}\sum_{n=1}^N \boldsymbol{\phi}(\boldsymbol{x}_n)$ and $\boldsymbol{\mu}_Q = \frac{1}{M}\sum_{m=1}^M \boldsymbol{\phi}(\boldsymbol{x}'_m)$ be empirical estimates of the kernel mean embeddings of the two distributions. Then the squared MMD is given by

$$\mathrm{MMD}^2(\mathcal{X}, \mathcal{X}') \triangleq ||\frac{1}{N}\sum_{n=1}^N \boldsymbol{\phi}(\boldsymbol{x}_n) - \frac{1}{M}\sum_{m=1}^M \boldsymbol{\phi}(\boldsymbol{x}'_m)||^2 \tag{2.331}$$

$$= \frac{1}{N^2}\sum_{n=1}^N\sum_{n'=1}^N \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}_{n'}) - \frac{2}{NM}\sum_{n=1}^N\sum_{m=1}^M \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}'_m)$$

$$+ \frac{1}{M^2}\sum_{m=1}^M\sum_{m'=1}^M \boldsymbol{\phi}(\boldsymbol{x}'_{m'})^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}'_m) \tag{2.332}$$

Since Equation (2.332) only involves inner products of the feature vectors, we can use the kernel trick (Section 18.2.2) to rewrite the above as follows:

$$\mathrm{MMD}^2(\mathcal{X}, \mathcal{X}') = \frac{1}{N^2}\sum_{n=1}^N\sum_{n'=1}^N \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) - \frac{2}{NM}\sum_{n=1}^N\sum_{m=1}^M \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}'_m) + \frac{1}{M^2}\sum_{m=1}^M\sum_{m'=1}^M \mathcal{K}(\boldsymbol{x}'_m, \boldsymbol{x}'_{m'}) \tag{2.333}$$

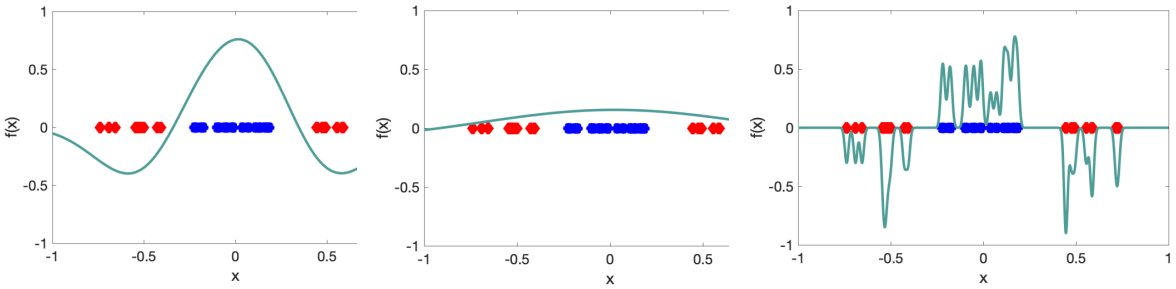


Figure 2.21: Effect of bandwidth parameter σ on the witness function defined by a Gaussian kernel. From a figure from [GSJ19]. Used with kind permission of Dougal Sutherland.

2.7.3.3 Linear time computation

The MMD takes $O(N^2)$ time to compute, where N is the number of samples from each distribution. In [Chw+15], they present a different test statistic called the **unnormalized mean embedding** or **UME**, that can be computed in $O(N)$ time.

The key idea is to notice that evaluating

$$\text{witness}^2(\boldsymbol{v}) = (\boldsymbol{\mu}_Q(\boldsymbol{v}) - \boldsymbol{\mu}_P(\boldsymbol{v}))^2 \tag{2.334}$$

at a set of test locations $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_J$ is enough to detect a difference between P and Q. Hence we define the (squared) UME as follows:

$$\text{UME}^2(P, Q) = \frac{1}{J} \sum_{j=1}^{J} \left[\boldsymbol{\mu}_P(\boldsymbol{v}_j) - \boldsymbol{\mu}_Q(\boldsymbol{v}_j)\right]^2 \tag{2.335}$$

where $\boldsymbol{\mu}_P(\boldsymbol{v}) = \mathbb{E}_{p(\boldsymbol{x})}\left[\mathcal{K}(\boldsymbol{x}, \boldsymbol{v})\right]$ can be estimated empirically in $O(N)$ time, and similarly for $\boldsymbol{\mu}_Q(\boldsymbol{v})$.

A normalized version of UME, known as NME, is presented in [Jit+16]. By maximizing NME wrt the locations $\boldsymbol{v}_j$, we can maximize the statistical power of the test, and find locations where P and Q differ the most. This provides an interpretable two-sample test for high dimensional data.

2.7.3.4 Choosing the right kernel

The effectiveness of MMD (and UME) obviously crucially depends on the right choice of kernel. Even for distiguishing 1d samples, the choice of kernel can be very important. For example, consider a Gaussian kernel, $\mathcal{K}_\sigma(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{2\sigma^2}||\boldsymbol{x} - \boldsymbol{x}'||^2)$. The effect of changing σ in terms of the ability to distinguish two different sets of 1d samples is shown in Figure 2.21. Fortunately, the MMD is differentiable wrt the kernel parameters, so we can choose the optimal σ^2 so as to maximize the power of the test [Sut+17]. (See also [Fla+16] for a Bayesian approach, which maximizes the marginal likelihood of a GP representation of the kernel mean embedding.)

For high-dimensional data such as images, it can be useful to use a pre-trained CNN model as a way to compute low-dimensional features. For example, we can define $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}_\sigma(\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}'))$,

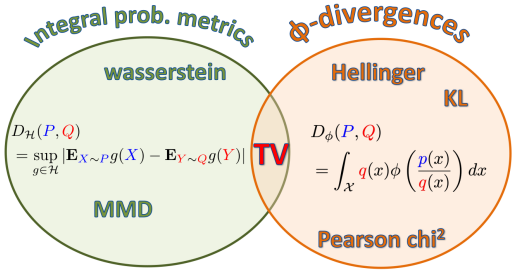


Figure 2.22: Summary of the two main kinds of divergence measures between two probability distributions P and Q. From a figure from [GSJ19]. Used with kind permission of Arthur Getton.

where $\boldsymbol{h}$ is some hidden layer of a CNN. such as the "Inception" model of [Sze+15]. The resulting MMD metric is known as the **kernel inception distance** [Bińkowski et al. 2018]. This is similar to the Frechet inception distance [Heu+17a], but has nicer statistical properties, and is better correlated with human perceptual judgement [Zho+19a].

2.7.4 Total variation distance

The **total variation distance** between two probability distributions is defined as follows:

$$D_{\mathrm{TV}}(p,q) \triangleq \frac{1}{2}||\boldsymbol{p}-\boldsymbol{q}||_1 = \frac{1}{2}\int |p(\boldsymbol{x})-q(\boldsymbol{x})|d\boldsymbol{x} \tag{2.336}$$

This is equal to an f-divergence where $f(r) = |r-1|/2$, since

$$\frac{1}{2}\int q(\boldsymbol{x})|\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}-1|d\boldsymbol{x} = \frac{1}{2}\int q(\boldsymbol{x})|\frac{p(\boldsymbol{x})-q(\boldsymbol{x})}{q(\boldsymbol{x})}|d\boldsymbol{x} = \frac{1}{2}\int |p(\boldsymbol{x})-q(\boldsymbol{x})|d\boldsymbol{x} \tag{2.337}$$

One can also show that the TV distance is an integral probability measure. In fact, it is the only divergence that is both an IPM and an f-divergence [Sri+09]. See Figure 2.22 for a visual summary.

2.7.5 Density ratio estimation using binary classifiers

In this section, we discuss a simple approach for comparing two distributions that turns out to be equivalent to IPMs and f-divergences.

Consider a binary classification problem in which points from P have label $y=1$ and points from Q have label $y=0$, i.e., $P(\boldsymbol{x}) = p(\boldsymbol{x}|y=1)$ and $Q(\boldsymbol{x}) = p(\boldsymbol{x}|y=0)$. Let $p(y=1)=\pi$ be the class prior. By Bayes' rule, the density ratio $r(\boldsymbol{x}) = P(\boldsymbol{x})/Q(\boldsymbol{x})$ is given by

$$\frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|y=1)}{p(\boldsymbol{x}|y=0)} = \frac{p(y=1|\boldsymbol{x})p(\boldsymbol{x})}{p(y=1)} / \frac{p(y=0|\boldsymbol{x})p(\boldsymbol{x})}{p(y=0)} \tag{2.338}$$

$$= \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})}\frac{1-\pi}{\pi} \tag{2.339}$$

If we assume $\pi = 0.5$, then we can estimate the ratio $r(\boldsymbol{x})$ by fitting a binary classifier or discriminator $h(\boldsymbol{x}) = p(y = 1|\boldsymbol{x})$ and then computing $r = h/(1 - h)$. This is called the **density ratio estimation** or **DRE** trick.

We can optimize the classifer h by minimizing the risk (expected loss). For example, if we use log-loss, we have

$$R(h) = \mathbb{E}_{p(\boldsymbol{x}|y)p(y)}\left[-y \log h(\boldsymbol{x}) - (1-y)\log(1-h(\boldsymbol{x}))\right] \tag{2.340}$$

$$= \pi \mathbb{E}_{P(\boldsymbol{x})}\left[-\log h(\boldsymbol{x})\right] + (1-\pi)\mathbb{E}_{Q(\boldsymbol{x})}\left[-\log(1-h(\boldsymbol{x}))\right] \tag{2.341}$$

We can also use other loss functions $\ell(y, h(\boldsymbol{x}))$ (see Section 26.2.2).

Let $R^{\ell}_{h^*} = \inf_{h \in \mathcal{F}} R(h)$ be the minimum risk achievable for loss function ℓ, where we minimize over some function class $\mathcal{F}$.[10] In [NWJ09], they show that for every f-divergence, there is a loss function ℓ such that $-D_f(P, Q) = R^{\ell}_{h^*}$. For example (using the notation $\tilde{y} \in \{-1, 1\}$ instead of $y \in \{0, 1\}$), total-variation distance corresponds to hinge loss, $\ell(\tilde{y}, h) = \max(0, 1 - \tilde{y}h)$; Hellinger distance corresponds to exponential loss, $\ell(\tilde{y}, h) = \exp(-\tilde{y}h)$; and χ^2 divergence corresponds to logistic loss, $\ell(\tilde{y}, h) = \log(1 + \exp(-\tilde{y}h))$.

We can also establish a connection between binary classifiers and IPMs [Sri+09]. In particular, let $\ell(\tilde{y}, h) = -2\tilde{y}h$, and $p(\tilde{y} = 1) = p(\tilde{y} = -1) = 0.5$. Then we have

$$R_{h^*} = \inf_h \int \ell(\tilde{y}, h(\boldsymbol{x}))p(\boldsymbol{x}|\tilde{y})p(\tilde{y})d\boldsymbol{x}d\tilde{y} \tag{2.342}$$

$$= \inf_h 0.5 \int \ell(1, h(\boldsymbol{x}))p(\boldsymbol{x}|\tilde{y} = 1)d\boldsymbol{x} + 0.5 \int \ell(-1, h(\boldsymbol{x}))p(\boldsymbol{x}|\tilde{y} = -1)d\boldsymbol{x} \tag{2.343}$$

$$= \inf_h \int h(\boldsymbol{x})Q(\boldsymbol{x})d\boldsymbol{x} - \int h(\boldsymbol{x})P(\boldsymbol{x})d\boldsymbol{x} \tag{2.344}$$

$$= \sup_h -\int h(\boldsymbol{x})Q(\boldsymbol{x})d\boldsymbol{x} + \int h(\boldsymbol{x})P(\boldsymbol{x})d\boldsymbol{x} \tag{2.345}$$

which matches Equation (2.324). Thus the classifier plays the same role as the witness function.

10. If P is a fixed distribution, and we minimize the above objective wrt h, while also maximizing it wrt a model $Q(\boldsymbol{x})$, we recover a technique known as a generative adversarial network for fitting an implicit model to a distribution of samples P (see Chapter 26 for details). However, in this section, we assume Q is known.

3 Bayesian statistics

3.1 Introduction

Probability theory (which we discussed in Chapter 2) is concerned with the forwards mapping from parameters $\boldsymbol{\theta}$ to data $\boldsymbol{x}$. (In the conditional setting, the forwards mapping is from $(\boldsymbol{\theta}, \boldsymbol{x})$ to $\boldsymbol{y}$, but we mostly focus on the unconditional setting for notational simplicity.) **Statistics** is concerned with the inverse problem, in which we want to infer the unknown parameters $\boldsymbol{\theta}$ given observations $\boldsymbol{x}$. (If we have multiple independent samples drawn from $\boldsymbol{\theta}$, we denote the dataset by $\mathcal{D} = \{\boldsymbol{x}_n : n = 1 : N\}$.) Indeed, statistics was originally called **inverse probability theory**. Nowadays, there are two main approaches to statistics, **frequentist statistics** and **Bayesian statistics**, as we discuss below.

3.1.1 Frequentist statistics

The frequentist approach to statistics (also called **classical statistics** or **orthodox statistics**) is based on the concept of **repeated trials**. More precisely, suppose we have an **estimator**, such as the maximum likelihood estimator, $\hat{\boldsymbol{\theta}}(\mathcal{D}) = \text{argmax}_{\boldsymbol{\theta}} \, p(\mathcal{D}|\boldsymbol{\theta})$. Now imagine what would happen if the estimator were applied to multiple random datasets of the same size, drawn from the same but unknown "true distribution", $p^*(\mathcal{D})$. The resulting distribution of estimated values, $\{\hat{\boldsymbol{\theta}}(\mathcal{D}') : \mathcal{D}' \sim p^*\}$, is called the **sampling distribution** of the estimator. The variability of this distribution can be regarded as a measure of uncertainty about the value that was estimated from the dataset that was actually observed, $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}(\mathcal{D}_{\text{obs}})$. (For more details, see e.g., Section 4.7 of the prequel to this book, [Mur22].)

In the machine learning literature, it is common to describe any method that computes a **point estimate** of the parameters (e.g., the MLE or the MAP estimate) as a "frequentist" method, but this is incorrect. It is the use of a sampling distribution to represent uncertainty that makes a method frequentist. If we ignore uncertainty, we are just performing an optimization problem, which is neither frequentist nor Bayesian.

3.1.2 Bayesian statistics

In the Bayesian approach to statistics, we treat the unknown parameters $\boldsymbol{\theta}$ just like any other unknown random variable. Hence we can model it with a probability distribution. The observed data is considered fixed, and we do not need to worry about hypothetical repeated random trials with different data. We represent knowledge about the possible values of $\boldsymbol{\theta}$ given the data using a (conditional) probability distribution, $p(\boldsymbol{\theta}|\mathcal{D})$.

To compute this distribution, we start by specifying our initial knowledge or beliefs using a **prior distribution**, $p(\boldsymbol{\theta})$. Our assumptions about how the data depends on the parameters are captured in the **likelihood function** $p(\mathcal{D}|\boldsymbol{\theta})$. We can then combine these using **Bayes' rule** to compute the **posterior distribution**, which represents our knowledge about the parameters after seeing the data:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})} = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{\int p(\mathcal{D}, \boldsymbol{\theta}')p(\boldsymbol{\theta}')d\boldsymbol{\theta}'} \tag{3.1}$$

where the quantity $p(\mathcal{D})$ is called the **marginal likelihood** or **evidence**. The task of computing this posterior is called **Bayesian inference**, **posterior inference** or just **inference**. We discuss different algorithms for solving this problem in Part II. For more details, see e.g., [Gel+14a; MKL21; MFR20].

3.1.3 Arguments for the Bayesian approach

The Bayesian approach is more general than the frequentist approach, since it can be applied to problems that don't repeat (e.g., we can ask what is the probability that the ice caps will melt by 2030, which is not a meaningful question in the frequentist world view.) In addition, it avoids certain conceptual pathologies that plague frequentist statistics (see discussions in e.g., [Efr86; Jay03; Cla21]). Furthermore, the Bayesian approach is widely used in engineering and business, and there is also considerable evidence that it is used by humans and other animals [MKG21][1]. There are also some more technical reasons for using the Bayesian approach, which we discuss below.

3.1.3.1 De Finetti's theorem

The Bayesian approach is based on the idea of putting a prior on our parameters, and then using this extended model to represent our data. In this section, we justify why this is a reasonable thing to do.

First, a definition. We say that a sequence of random variables $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots)$ is **infinitely exchangeable** if, for any n, the joint probability $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is invariant to permutation of the indices. That is, for any permutation π, we have

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = p(\boldsymbol{x}_{\pi_1}, \ldots, \boldsymbol{x}_{\pi_n}) \tag{3.2}$$

Note that iid implies exchangeability but not vice versa. For example, suppose $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is a set of MNIST images, and let $\boldsymbol{x}_0$ be a background image. The sequence $(\boldsymbol{x}_0 + \boldsymbol{x}_1, \ldots, \boldsymbol{x}_0 + \boldsymbol{x}_n)$ is infinitely exchangeable but not iid, since all the variables share a hidden common factor, namely the background $\boldsymbol{x}_0$. Thus the more examples we see, the better we will be able to estimate the shared $\boldsymbol{x}_0$, and thus the better we can predict future elements.

Thus the key advantage of working with the exchangeability assumption is that it allows us to learn from similarities between data points. In fact, one can show the following result:

Theorem 3.1.1 (**de Finetti's theorem**). *A sequence of random variables* $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots)$ *is infinitely*

1. We will see that Bayesian inference can be computationally expensive. Brains have evolved to use various shortcuts to make the Bayesian computations efficient, see e.g., [Lak+17; Gri20].

exchangeable iff, for all n, we have

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \int \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{3.3}$$

where $\boldsymbol{\theta}$ is some hidden common random variable (possibly infinite dimensional). That is, $\boldsymbol{x}_i$ are iid conditional on $\boldsymbol{\theta}$.

We often interpret $\boldsymbol{\theta}$ as a parameter. The theorem tells us that, if our data is exchangeable, then there must exist a parameter $\boldsymbol{\theta}$, and a likelihood $p(\boldsymbol{x}_i|\boldsymbol{\theta})$, and a prior $p(\boldsymbol{\theta})$. Thus the Bayesian approach follows automatically from exchangeability [O'N09].

3.1.3.2 The Dutch book theorem

Bayesian inference forms the foundation of Bayesian decision theory, which we discuss in Section 34.1. It can be shown that any other method for representing uncertainty is guaranteed to cause a decision maker to lose money if deployed in the context of a gambling game. This is called the **Dutch book** theorem. For details, see e.g., [Háj08]. Of course, this argument extends beyond gambling and applies to any form of decision making under uncertainty.

3.1.3.3 Online learning

The posterior captures our (un)certainty about the parameter given the data and whatever prior knowledge we had. Once we have computed it, we can "throw away" the data. This makes Bayesian inference particularly well-suited to online learning, where the dataset grows without bound, since we can recursively update our belief state:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_t|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_{1:t-1}) \tag{3.4}$$

Being able to rapidly update our beliefs about the world (e.g., model parameters) in response to possibly small amounts of new data is essential for creating intelligent agents that can adapt to changing distributions (see Chapter 19) and perform sequential decision making under uncertainty (see Part VI).

3.1.4 Arguments against the Bayesian approach

There are several arguments against the Bayesian approach. Some are philosophical, and focus on the sensitivity to the choice of prior (which we discuss in Section 3.5). However, in practice the main obstacle is computational. To see why, note that evaluating the posterior in Equation (3.1) can be computationally expensive, because of the need to compute the normalizing constant $p(\mathcal{D})$ in the denominator, which often involves a high dimensional integral. In Part II, we will discuss many different algorithms for exact and approximate Bayesian computation.

3.1.5 Why not just use MAP estimation?

Bayesian inference uses a prior, which can help prevent overfitting. A simpler approach to this problem is to just compute the **MAP estimate** or maximum a posterior estimate, since this avoids

the need to compute $p(\mathcal{D})$:

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \left[\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right] \tag{3.5}$$

Although this is often considered a Bayesian approach, since it is derived from a prior and a likelihood, it is not "fully Bayesian", since it is just a point estimate. We discuss the downsides of MAP estimation below.

3.1.5.1 The MAP estimate gives no measure of uncertainty

In many statistical applications (especially in science) it is important to know how much one can trust a given parameter estimate. For example, consider tossing a coin. If we get N_1 heads and N_0 tails, we know that the maximum likelihood estimate is

$$\hat{\theta}_{\text{mle}} = \frac{N_1}{N_1 + N_0} = \frac{N_1}{N} \tag{3.6}$$

where $N = N_1 + N_0$. But this is just a point estimate, with no associated uncertainty. For example, if we toss a coin 5 times, and see 4 heads, we estimate $\hat{\theta}_{\text{mle}} = 4/5$, but do we really believe the coin is that biased? We cannot be sure, because the sample size is so small.

Now suppose we put a prior on θ. As we explain in Section 3.2.1, it is convenient to use a beta distribution as a prior, $p(\theta) = \text{Beta}(\theta|\, \breve{\alpha}, \breve{\beta})$, where $\breve{\alpha}$ and $\breve{\beta}$ are known as **pseudo counts**. We will show that the posterior has the form $p(\theta) = \text{Beta}(\theta|\, \widehat{\alpha}, \widehat{\beta})$, where $\widehat{\alpha}{=}\breve{\alpha} + N_1$ and $\widehat{\beta}{=}\breve{\beta} + N_0$. The mode of this distribution (i.e., the MAP estimate) is

$$\hat{\theta}_{\text{map}} = \frac{\widehat{\alpha} - 1}{\widehat{\alpha} - 1 + \widehat{\beta} - 1} \tag{3.7}$$

Even though we used a prior, this is still just another point estimate, with no notion of uncertainty. However, we can derive measures of confidence or uncertainty from posterior distribution. For example, we can compute a 95% **credible interval** $I = (\ell, u)$, which satisfies $p(\theta \in I|\mathcal{D}) = 0.95$, by setting $\ell = F^{-1}(\alpha/2)$ and $u = F^{-1}(1 - \alpha/2)$, where F is the cdf of the posterior, and F^{-1} is the inverse cdf. Alternatively, we can compute the posterior standard deviation (sometimes called the **standard error**), given by $\sigma = \sqrt{\mathbb{V}\left[\theta|\mathcal{D}\right]}$. See [Mur22, Sec 4.6.2.8] for details.

3.1.5.2 The plugin approximation does not capture predictive uncertainty

In machine learning applications, the parameters of our model are usually of little interest, since they are usually **unidentifiable** and hence uninterpretable. Instead, we are interested in **predictive uncertainty**. This can be useful in applications which involve decision making, such as reinforcement learning, active learning, or safety-critical applications. We can derive uncertainty in the predictions induced by uncertainty in the parameters by computing the **posterior predictive distribution**:

$$p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{3.8}$$

By **integrating out**, or **marginalizing out**, the unknown parameters, we reduce the chance of overfitting, since we are effectively computing the weighted average of predictions from an infinite

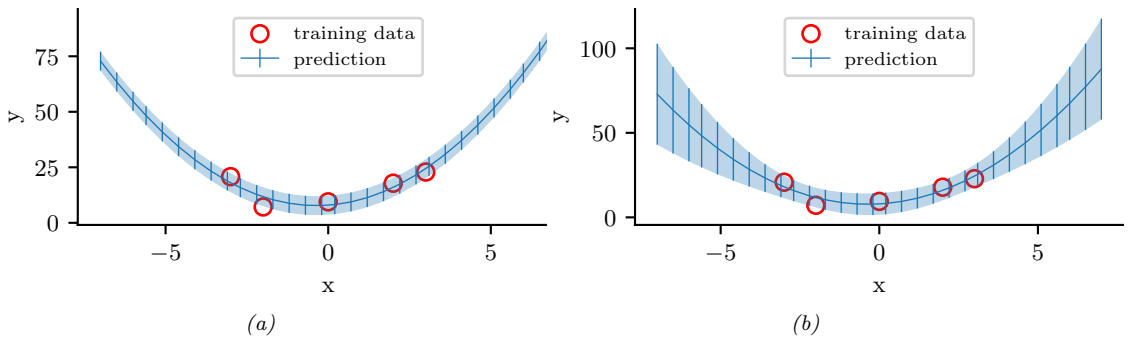


Figure 3.1: Predictions made by a polynomial regression model fit to a small dataset. (a) Plugin approximation to predictive density using the MLE. The curves shows the posterior mean, $\mathbb{E}[y|\boldsymbol{x}]$, and the error bars show the posterior standard deviation, $\text{std}[y|\boldsymbol{x}]$, around this mean. (b) Bayesian posterior predictive density, obtained by integrating out the parameters. Generated by linreg_post_pred_plot.ipynb.

number of models. This act of integrating over uncertainty is at the heart of the Bayesian approach to machine learning. (Of course, the Bayesian approach requires a prior, but so too do methods that rely on regularization, so the prior is not so much the distinguishing aspect.)

Suppose we approximate the posterior by a point estimate. We can represent this using a **delta function**, $p(\boldsymbol{\theta}|\mathcal{D}) \approx \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$; the value $\hat{\boldsymbol{\theta}}$ could the MLE or a MAP estimate. If we substitute this into Equation (3.8), and use the sifting property of delta functions, we get the following approximation to the posterior predictive:

$$p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}) \approx \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})\delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})d\boldsymbol{\theta} = p(\boldsymbol{y}|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) \tag{3.9}$$

This is called a **plugin approximation**, and is very widely used, due to its simplicity.

However, the plugin approximation ignores uncertainty in the parameter estimates, which can result in an underestimate of the uncertainty. For example, Figure 3.1a plots the plugin approximation $p(y|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$ for a linear regression model $p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\boldsymbol{w}^\mathsf{T}\boldsymbol{x}, \sigma^2)$, where we plug in the MLEs for $\boldsymbol{w}$ and σ^2 (the plot looks similar if we plug in the MAP estimates). We see that the size of the predicted variance is a constant (namely $\hat{\sigma}^2$).

The uncertainty captured by σ is called **aleatoric uncertainty** or **intrinsic uncertainty**, and would persist even if we knew the true model and true parameters. In practice we don't know the parameters. This induces an additional, and orthogonal, source of uncertainty, called **epistemic uncertainty** (since it arises due to a lack of knowledge about the truth). In the Bayesian approach, we take this into account. The result is shown in Figure 3.1b. We see that now the error bars get wider as we move away from the training data; this is due to the Bayesian estimate of the parameters being adaptive to the test data, i.e., in the Bayesian approach, we predict using $p(y|\boldsymbol{x}, \mathcal{D}) = \mathcal{N}(y|\hat{\boldsymbol{w}}_{\text{bayes}}(\boldsymbol{x})^\mathsf{T}\boldsymbol{x}, \hat{\sigma}^2_{\text{bayes}})$ whereas in the plugin approach, we predict using $p(y|\boldsymbol{x}, \mathcal{D}) \approx p(y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) = \mathcal{N}(y|\hat{\boldsymbol{w}}^\mathsf{T}_{\text{mle}}\boldsymbol{x}, \hat{\sigma}^2_{\text{mle}})$.

For more details on Bayesian linear regression, see Section 15.2. For details on how to derive Bayesian predictions for nonlinear models such as neural nets, see Section 17.1.

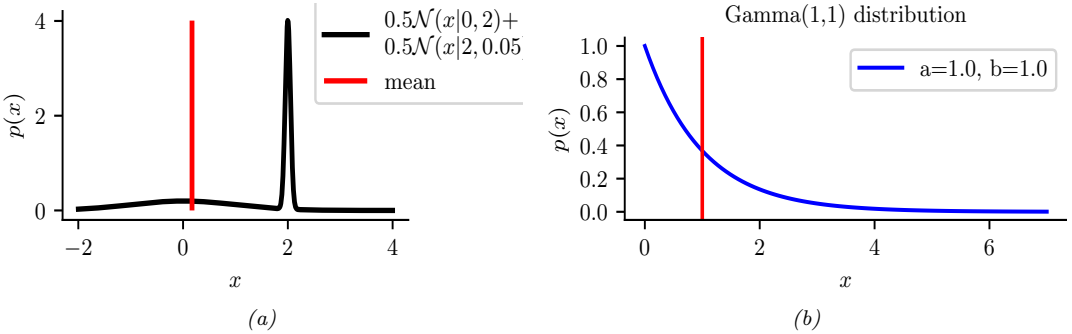


Figure 3.2: Two distributions in which the mode (highest point) is untypical of the distribution; the mean (vertical red line) is a better summary. (a) A bimodal distribution. Generated by bimodal_dist_plot.ipynb. (b) A skewed Ga(1, 1) *distribution. Generated by gamma_dist_plot.ipynb.*

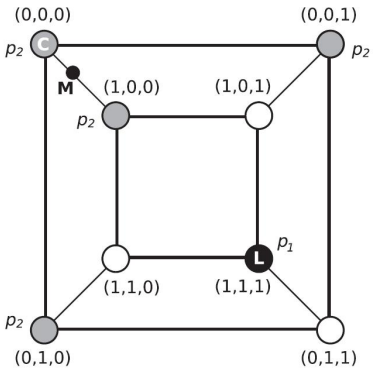


Figure 3.3: A distribution on a discrete space in which the mode (black point L, with probability p_1) is untypical of most of the probability mass (gray circles, with probability $p_2 < p_1$). The small black circle labeled M (near the top left) is the posterior mean, which is not well defined in a discrete state space. C (the top left vertex) is the centroid estimator, made up of the maximizer of the posterior marginals. See text for details. From Figure 1 of [CL07]. Used with kind permission of Luis Carvalho.

3.1.5.3 The MAP estimate is often untypical of the posterior

The MAP estimate is often easy to compute. However, the mode of a posterior distribution is often a very poor choice as a summary statistic, since the mode is usually quite untypical of the distribution, unlike the mean or median. This is illustrated in Figure 3.2(a) for a 1D continuous space, where we see that the mode is an isolated peak (black line), far from most of the probability mass. By contrast, the mean (red line) is near the middle of the distribution.

Another example is shown in Figure 3.2(b): here the mode is 0, but the mean is non-zero. Such skewed distributions often arise when inferring variance parameters, especially in hierarchical models. In such cases the MAP estimate (and hence the MLE) is obviously a very bad estimate.

Similar problems with MAP estimates can arise in discrete spaces, such as when estimating graph

structures, or long sequences of symbols. Figure 3.3 shows a distribution on $\{0,1\}^3$, where points are arranged such that they are connected to their nearest neighbors, as measured by Hamming distance. The black state (circle) labeled L (configuration (1,1,1)) has probability p_1; the 4 gray states have probability $p_2 < p_1$; and the 3 white states have probability 0. Although the black state is the most probable, it is untypical of the posterior: all its nearest neighbors have probability zero, meaning it is very isolated. By contrast, the gray states, although slightly less probable, are all connected to other gray states, and together they constitute much more of the total probability mass.

3.1.5.4 The MAP estimate is only optimal for 0-1 loss

The MAP estimate is the optimal estimate when the loss function is 0-1 loss, $\ell(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathbb{I}\left(\boldsymbol{\theta} \neq \hat{\boldsymbol{\theta}}\right)$, as we show in Section 34.1.2. However, this does not give any "partial credit" for estimating some of the components of $\boldsymbol{\theta}$ correctly. An alternative is to use the **Hamming loss**: $\ell(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{d=1}^{D} \mathbb{I}\left(\theta_d \neq \hat{\theta}_d\right)$. In this case, one can show that the optimal estimator is the vector of **max marginals**

$$\hat{\boldsymbol{\theta}} = \left[\operatorname*{argmax}_{\theta_d} \int_{\boldsymbol{\theta}_{-d}} p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}_{-d}\right]_{d=1}^{D} \tag{3.10}$$

This is also called the **maximizer of posterior marginals** or **MPM** estimate. Note that computing the max marginals involves marginalization and maximization, and thus depends on the whole distribution; this tends to be more robust than the MAP estimate [MMP87].

3.1.5.5 The MAP estimate is not invariant to reparameterization

A more subtle problem with MAP estimation is that the result we get depends on how we parameterize the probability distribution, which is not very desirable. For example, when representing a Bernoulli distribution, we should be able to parameterize it in terms of probability of success, or in terms of the log-odds (logit), without that affecting our beliefs.

For example, let $\hat{x} = \operatorname{argmax}_x p_x(x)$ be the MAP estimate for x. Now let $y = f(x)$ be a transformation of x. In general it is not the case that $\hat{y} = \operatorname{argmax}_y p_y(y)$ is given by $f(\hat{x})$. For example, let $x \sim \mathcal{N}(6,1)$ and $y = f(x)$, where $f(x) = \frac{1}{1+\exp(-x+5)}$. We can use the change of variables (Section 2.5.1) to conclude $p_y(y) = p_x(f^{-1}(y))|\frac{df^{-1}(y)}{dy}|$. Alternatively we can use a Monte Carlo approximation. The result is shown in Figure 2.11. We see that the original Gaussian for $p(x)$ has become "squashed" by the sigmoid nonlinearity. In particular, we see that the mode of the transformed distribution is not equal to the transform of the original mode.

We have seen that the MAP estimate depends on the parameterization. The MLE does not suffer from this since the likelihood is a function, not a probability density. Bayesian inference does not suffer from this problem either, since the change of measure is taken into account when integrating over the parameter space.

3.1.5.6 MAP estimation cannot handle the cold-start problem

As the amount of data increases, the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ sometimes shrinks to a point, since the likelihood term $p(\mathcal{D}|\boldsymbol{\theta})$ dominates the fixed prior $p(\boldsymbol{\theta})$. That is, $p(\boldsymbol{\theta}|\mathcal{D}) \to \delta_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\theta})$, where $\hat{\boldsymbol{\theta}}$ is the MLE.

(We will illustrate this phenomenon in more detail later in this chapter.) Thus one may think that in the era of **big data**, we do not need to model posterior uncertainty. However, this assumes that the data is informative about all of the unknown variables. In many problems there is a **long tail** of data, in which a small number of items occur frequently, but most items occur rarely. Consequently there may be a lot of uncertainty about these rare items (see e.g., [Jor11]).

For example, in a recommender system, we will always be faced with new users, about whom we know very little (the so-called **cold start problem**). Bayesian methods can help create personalized recommendations in such cases, by borrowing statistical strength from other similar users for whom we have more data (see Section 3.7). Similarly, when performing online learning and sequential decision making, an agent will often encounter new data that may not have been seen before (indeed, it may actively seek such novelty), so it may often be in a small data regime. For example, see the discussion of Bayesian optimization in Section 6.8 and bandits in Section 34.4.

3.2 Conjugate priors for simple models

In this section, we consider the problem of inferring the posterior for parameters of simple probability models. These will form the foundations for many more complex models. We will use priors that are **conjugate** to the likelihood. This is defined as follows: a prior $p(\boldsymbol{\theta}) \in \mathcal{F}$ is a conjugate prior for a likelihood function $p(\mathcal{D}|\boldsymbol{\theta})$ if the posterior is in the same parameterized family as the prior, i.e., $p(\boldsymbol{\theta}|\mathcal{D}) \in \mathcal{F}$. In other words, $\mathcal{F}$ is closed under Bayesian updating. If the family $\mathcal{F}$ corresponds to the exponential family (defined in Section 2.3), then the computations can be performed in closed form. In the sections below, we give some common examples of this framework, which we will use later in the book.

3.2.1 The binomial model

One of the simplest examples of conjugate Bayesian analysis is the beta-binomial model. This is covered in detail in Section 4.6.2 of the prequel to this book, [Mur22]. For completeness, we just state the results here without further discussion.

For N trials, the binomial likelihood is

$$p(\mathcal{D}|\theta) = \text{Bin}(y|N,\theta) = \binom{N}{y}\theta^y(1-\theta)^{N-y} \tag{3.11}$$

If $N = 1$, this reduces to the Bernoulli likelihood.

The conjugate prior is a beta distribution:

$$p(\theta) = \text{Beta}(\theta|\, \breve{\alpha}, \breve{\beta}) \propto \theta^{\breve{\alpha}-1}(1-\theta)^{\breve{\beta}-1} \tag{3.12}$$

The values $\breve{\alpha}$ and $\breve{\beta}$ are known as **pseudo counts**, since they play a role analogous to the empirical counts N_1 and N_0.

If we multiply the likelihood with the beta prior we get a beta posterior:

$$p(\theta|\mathcal{D}) \propto \theta^{N_1}(1-\theta)^{N_0}\ \theta^{\breve{\alpha}-1}(1-\theta)^{\breve{\beta}-1} \tag{3.13}$$

$$\propto \text{Beta}(\theta|\, \breve{\alpha}+N_1, \breve{\beta}+N_0) \tag{3.14}$$

$$= \text{Beta}(\theta|\, \widehat{\alpha}, \widehat{\beta}) \tag{3.15}$$

where $\widehat{\alpha} \triangleq \breve{\alpha} + N_1$ and $\widehat{\beta} \triangleq \breve{\beta} + N_0$ are the parameters of the posterior.

The posterior mean is a convex combination of the prior mean, $m = \breve{\alpha} / \breve{N}$ (where $\breve{N} \triangleq \breve{\alpha} + \breve{\beta}$ is the prior strength), and the MLE: $\hat{\theta}_{\text{mle}} = \frac{N_{\mathcal{D}1}}{N_{\mathcal{D}}}$:

$$\mathbb{E}[\theta|\mathcal{D}] = \frac{\breve{\alpha} + N_1}{\breve{\alpha} + N_1 + \breve{\beta} + N_0} = \frac{\breve{N}\, m + N_{\mathcal{D}1}}{N_{\mathcal{D}} + \breve{N}} = \frac{\breve{N}}{N_{\mathcal{D}} + \breve{N}} m + \frac{N_{\mathcal{D}}}{N_{\mathcal{D}} + \breve{N}} \frac{N_{\mathcal{D}1}}{N_{\mathcal{D}}} = \lambda m + (1-\lambda)\hat{\theta}_{\text{mle}} \tag{3.16}$$

where $\lambda = \frac{\breve{N}}{\widehat{N}}$ is the ratio of the prior to posterior equivalent sample size. So the weaker the prior, the smaller is λ, and hence the closer the posterior mean is to the MLE. In particular, if we set the prior pseudocounts to 0, then $\lambda = 0$, so the posterior mean becomes the MLE.

The posterior mode is given by

$$\hat{\theta} = \operatorname*{argmax}_{\theta} p(\theta|\mathcal{D}) = \frac{\breve{\alpha} + N_1 - 1}{\breve{\alpha} + N_1 - 1 + \breve{\beta} + N_0 - 1} = \frac{\breve{\alpha} + N_1 - 1}{\breve{\alpha} + \breve{\beta} + N - 2} \tag{3.17}$$

If we set the prior pseudocounts to 1, the MAP estimate becomes the MLE, which can overfit. To avoid this, it is common to set the prior pseudo counts to 2, which encodes a weak prior that $\theta = 0.5$; the corresponding MAP estimate is then just like the MLE, except we add 1 to add the empirical counts before normalizing. This is called **add one smoothing**.

The variance of the beta posterior is given by

$$\mathbb{V}[\theta|\mathcal{D}] = \frac{\widehat{\alpha}\widehat{\beta}}{(\widehat{\alpha} + \widehat{\beta})^2(\widehat{\alpha} + \widehat{\beta} + 1)} = \mathbb{E}[\theta|\mathcal{D}]^2 \frac{\widehat{\beta}}{\widehat{\alpha}\,(1 + \widehat{\alpha} + \widehat{\beta})} \tag{3.18}$$

where $\widehat{\alpha} = \breve{\alpha} + N_1$ and $\widehat{\beta} = \breve{\beta} + N_0$. If $N_{\mathcal{D}} \gg \breve{\alpha} + \breve{\beta}$, this simplifies to

$$\mathbb{V}[\theta|\mathcal{D}] \approx \frac{N_{\mathcal{D}1} N_{\mathcal{D}0}}{{N_{\mathcal{D}}}^3} = \frac{\hat{\theta}(1-\hat{\theta})}{N_{\mathcal{D}}} \tag{3.19}$$

where $\hat{\theta}$ is the MLE. Hence the standard error is given by

$$\sigma = \sqrt{\mathbb{V}[\theta|\mathcal{D}]} \approx \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{N_{\mathcal{D}}}} \tag{3.20}$$

We see that the uncertainty goes down at a rate of $1/\sqrt{N}$.

Finally, we can show that the marginal likelihood for the beta-binomial model is given by

$$p(\mathcal{D}) = \binom{N_{\mathcal{D}}}{N_{\mathcal{D}1}} \frac{B(\breve{\alpha} + N_{\mathcal{D}1}, \breve{\beta} + N_{\mathcal{D}0})}{B(\breve{\alpha}, \breve{\beta})} = \binom{N_{\mathcal{D}}}{N_{\mathcal{D}1}} \frac{B(\widehat{\alpha}, \widehat{\beta})}{B(\breve{\alpha}, \breve{\beta})} \tag{3.21}$$

In the Bernoulli case, where $N = 1$, the first term can be omitted.

3.2.2 The multinomial model

In this section, we generalize the results from Section 3.2.1 from binary variables (e.g., coins) to K-ary variables (e.g., dice).

Let $y \sim \text{Cat}(\boldsymbol{\theta})$ be a discrete random variable drawn from a **categorical distribution**. The likelihood has the form

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \text{Cat}(y_n|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{c=1}^{C} \theta_c^{\mathbb{I}(y_n=c)} = \prod_{c=1}^{C} \theta_c^{N_c} \tag{3.22}$$

where $N_c = \sum_n \mathbb{I}(y_n = c)$. We can generalize this to the **multinomial distribution** by defining $\boldsymbol{y} \sim \mathcal{M}(N, \boldsymbol{\theta})$, where N is the number of trials, and $y_c = N_c$ is the number of times value c is observed. The likelihood becomes

$$p(\boldsymbol{y}|N, \boldsymbol{\theta}) = \binom{N}{N_1 \ldots N_C} \prod_{c=1}^{C} \theta_c^{N_c} \tag{3.23}$$

This is the same as the categorical likelihood modulo a scaling factor. Going forward, we will work with the categorical model, for notational simplicity.

The conjugate prior for a categorical distribution is the Dirichlet distribution, which we discussed in Section 2.2.8.1. We denote this by $p(\boldsymbol{\theta}) = \text{Dir}(\boldsymbol{\theta}|\, \breve{\boldsymbol{\alpha}})$, where $\breve{\boldsymbol{\alpha}}$ is the vector of prior pseudo-counts. Often we use a symmetric Dirichlet prior of the form $\breve{\alpha}_k = \breve{\alpha}/K$. In this case, we have $\mathbb{E}[\theta_k] = 1/K$, and $\mathbb{V}[\theta_k] = \frac{K-1}{K^2(\breve{\alpha}+1)}$. Thus we see that increasing the prior sample size $\breve{\alpha}$ decreases the variance of the prior, which is equivalent to using a stronger prior.

We can combine the multinomial likelihood and Dirichlet prior to compute the Dirichlet posterior, as follows:

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})\text{Dir}(\boldsymbol{\theta}|\, \breve{\boldsymbol{\alpha}}) \propto \left[\prod_k \theta_k^{N_k}\right]\left[\prod_k \theta_k^{\breve{\alpha}_k - 1}\right] \tag{3.24}$$

$$\propto \text{Dir}(\boldsymbol{\theta}|\, \breve{\alpha}_1 + N_1, \ldots, \breve{\alpha}_K + N_K) = \text{Dir}(\boldsymbol{\theta}|\, \widehat{\boldsymbol{\alpha}}) \tag{3.25}$$

where $\widehat{\alpha}_k = \breve{\alpha}_k + N_k$ are the parameters of the posterior. So we see that the posterior can be computed by adding the empirical counts to the prior counts. In particular, the posterior mode is given by

$$\hat{\theta}_k = \frac{\widehat{\alpha}_k - 1}{\sum_{k'=1}^{K} \widehat{\alpha}_k - 1} = \frac{N_k + \breve{\alpha}_k - 1}{\sum_{k'=1}^{K} N_k + \breve{\alpha}_k - 1} \tag{3.26}$$

If we set $\alpha_k = 1$ we recover the MLE; if we set $\alpha_k = 2$, we recover the add-one smoothing estimate.

The marginal likelihood for the Dirichlet-categorical model is given by the following:

$$p(\mathcal{D}) = \frac{B(\mathbf{N} + \boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \tag{3.27}$$

where

$$B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)} \tag{3.28}$$

Hence we can rewrite the above result in the following form, which is what is usually presented in the literature:

$$p(\mathcal{D}) = \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(N_{\mathcal{D}} + \sum_k \alpha_k)} \prod_k \frac{\Gamma(N_{\mathcal{D}k} + \alpha_k)}{\Gamma(\alpha_k)} \tag{3.29}$$

For more details on this model, see [Mur22, Sec 4.6.3].

3.2.3 The univariate Gaussian model

In this section, we derive the posterior $p(\mu, \sigma^2|\mathcal{D})$ for a univariate Gaussian. For simplicity, we consider this in three steps: inferring just μ, inferring just σ^2, and then inferring both. See Section 3.3 for the multivariate case.

3.2.3.1 Posterior of μ given σ^2

If σ^2 is a known constant, the likelihood for μ has the form

$$p(\mathcal{D}|\mu) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N_{\mathcal{D}}}(y_n-\mu)^2\right) \tag{3.30}$$

One can show that the conjugate prior is another Gaussian, $\mathcal{N}(\mu|\breve{m}, \breve{\tau}^2)$. Applying Bayes' rule for Gaussians (Equation (2.82)), we find that the corresponding posterior is given by

$$p(\mu|\mathcal{D}, \sigma^2) = \mathcal{N}(\mu|\,\widehat{m}, \widehat{\tau}^2) \tag{3.31}$$

$$\widehat{\tau}^2 = \frac{1}{\frac{N}{\sigma^2}+\frac{1}{\breve{\tau}^2}} = \frac{\sigma^2\,\breve{\tau}^2}{N\,\breve{\tau}^2+\sigma^2} \tag{3.32}$$

$$\widehat{m} = \widehat{\tau}^2\left(\frac{\breve{m}}{\breve{\tau}^2}+\frac{N\overline{y}}{\sigma^2}\right) = \frac{\sigma^2}{N\,\breve{\tau}^2+\sigma^2}\,\breve{m} + \frac{N\,\breve{\tau}^2}{N\,\breve{\tau}^2+\sigma^2}\overline{y} \tag{3.33}$$

where $\overline{y} \triangleq \frac{1}{N}\sum_{n=1}^{N} y_n$ is the empirical mean.

This result is easier to understand if we work in terms of the precision parameters, which are just inverse variances. Specifically, let $\lambda = 1/\sigma^2$ be the observation precision, and $\breve{\lambda} = 1/\,\breve{\tau}^2$ be the precision of the prior. We can then rewrite the posterior as follows:

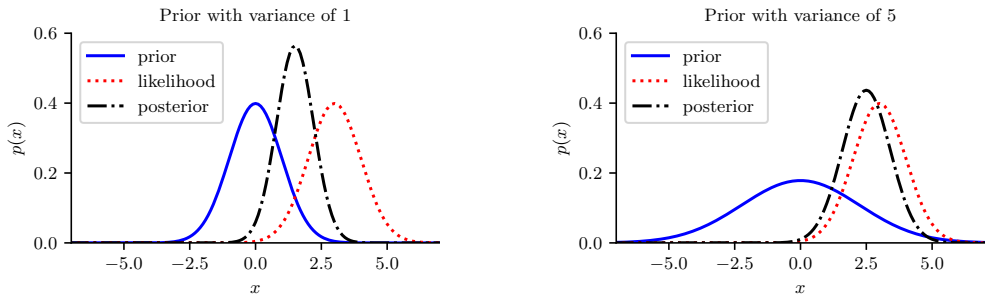
$$p(\mu|\mathcal{D}, \lambda) = \mathcal{N}(\mu|\,\widehat{m}, \widehat{\lambda}^{-1}) \tag{3.34}$$

$$\widehat{\lambda} = \breve{\lambda} + N\lambda \tag{3.35}$$

$$\widehat{m} = \frac{N\lambda\overline{y} + \breve{\lambda}\breve{m}}{\widehat{\lambda}} = \frac{N\lambda}{N\lambda+\breve{\lambda}}\overline{y} + \frac{\breve{\lambda}}{N\lambda+\breve{\lambda}}\,\breve{m} \tag{3.36}$$

These equations are quite intuitive: the posterior precision $\widehat{\lambda}$ is the prior precision $\breve{\lambda}$ plus N units of measurement precision λ. Also, the posterior mean $\widehat{m}$ is a convex combination of the empirical mean $\overline{y}$ and the prior mean $\breve{m}$. This makes it clear that the posterior mean is a compromise between the empirical mean and the prior. If the prior is weak relative to the signal strength ($\breve{\lambda}$ is small relative to λ), we put more weight on the empirical mean. If the prior is strong relative to the signal strength ($\breve{\lambda}$ is large relative to λ), we put more weight on the prior. This is illustrated in Figure 3.4. Note also that the posterior mean is written in terms of $N\lambda\overline{x}$, so having N measurements each of precision λ is like having one measurement with value $\overline{x}$ and precision $N\lambda$.

To gain further insight into these equations, consider the posterior after seeing a single data point



(a) *(b)*

Figure 3.4: Inferring the mean of a univariate Gaussian with known σ^2. (a) Using strong prior, $p(\mu) = \mathcal{N}(\mu|0,1)$. (b) Using weak prior, $p(\mu) = \mathcal{N}(\mu|0,5)$. Generated by gauss_infer_1d.ipynb.

y (so $N = 1$). Then the posterior mean can be written in the following equivalent ways:

$$\widehat{m} = \frac{\breve{\lambda}}{\widehat{\lambda}}\breve{m} + \frac{\lambda}{\widehat{\lambda}}y \tag{3.37}$$

$$= \breve{m} + \frac{\lambda}{\widehat{\lambda}}(y - \breve{m}) \tag{3.38}$$

$$= y - \frac{\breve{\lambda}}{\widehat{\lambda}}(y - \breve{m}) \tag{3.39}$$

The first equation is a convex combination of the prior mean and the data. The second equation is the prior mean adjusted towards the data y. The third equation is the data adjusted towards the prior mean; this is called a **shrinkage** estimate. This is easier to see if we define the weight $w = \breve{\lambda}/\widehat{\lambda}$. Then we have

$$\widehat{m} = y - w(y - \breve{m}) = (1 - w)y + w\,\breve{m} \tag{3.40}$$

Note that, for a Gaussian, the posterior mean and posterior mode are the same. Thus we can use the above equations to perform MAP estimation.

3.2.3.2 Posterior of σ^2 given μ

If μ is a known constant, the likelihood for σ^2 has the form

$$p(\mathcal{D}|\sigma^2) \propto (\sigma^2)^{-N_{\mathcal{D}}/2} \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N_{\mathcal{D}}}(y_n - \mu)^2\right) \tag{3.41}$$

where we can no longer ignore the $1/(\sigma^2)$ term in front. The standard conjugate prior is the inverse Gamma distribution (Section 2.2.3.4), given by

$$\mathrm{IG}(\sigma^2|\,\breve{a}, \breve{b}) = \frac{\breve{b}^{\breve{a}}}{\Gamma(\breve{a})}(\sigma^2)^{-(\breve{a}+1)} \exp(-\frac{\breve{b}}{\sigma^2}) \tag{3.42}$$

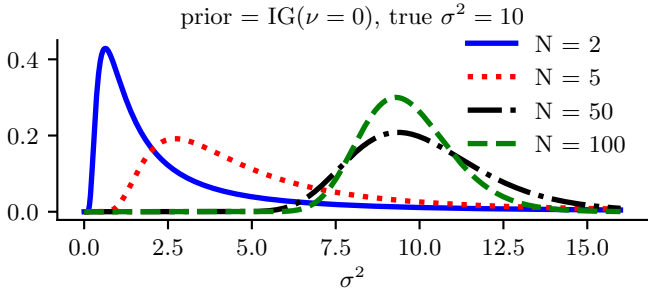


Figure 3.5: Sequential updating of the posterior for σ^2 starting from an uninformative prior. The data was generated from a Gaussian with known mean $\mu = 5$ and unknown variance $\sigma^2 = 10$. Generated by gauss_seq_update_sigma_1d.ipynb

Multiplying the likelihood and the prior, we see that the posterior is also IG:

$$p(\sigma^2|\mu, \mathcal{D}) = \text{IG}(\sigma^2|\, \widehat{a}, \widehat{b}) \tag{3.43}$$

$$\widehat{a} = \breve{a} + N/2 \tag{3.44}$$

$$\widehat{b} = \breve{b} + \frac{1}{2}\sum_{n=1}^{N_{\mathcal{D}}}(y_n - \mu)^2 \tag{3.45}$$

See Figure 3.5 for an illustration.

One small annoyance with using the IG($\breve{a}, \breve{b}$) distribution is that the strength of the prior is encoded in both $\breve{a}$ and $\breve{b}$. Therefore, in the Bayesian statistics literature it is common to use an alternative parameterization of the IG distribution, known as the (scaled) **inverse chi-squared distribution**:

$$\chi^{-2}(\sigma^2|\, \breve{\nu}, \breve{\tau}^2) = \text{IG}(\sigma^2|\frac{\breve{\nu}}{2}, \frac{\breve{\nu}\,\breve{\tau}^2}{2}) \propto (\sigma^2)^{-\breve{\nu}/2-1}\exp(-\frac{\breve{\nu}\,\breve{\tau}^2}{2\sigma^2}) \tag{3.46}$$

Here $\breve{\nu}$ (called the degrees of freedom or dof parameter) controls the strength of the prior, and $\breve{\tau}^2$ encodes the prior mean. With this prior, the posterior becomes

$$p(\sigma^2|\mathcal{D}, \mu) = \chi^{-2}(\sigma^2|\, \widehat{\nu}, \widehat{\tau}^2) \tag{3.47}$$

$$\widehat{\nu} = \breve{\nu} + N_{\mathcal{D}} \tag{3.48}$$

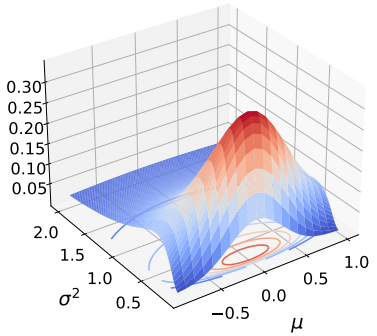
$$\widehat{\tau}^2 = \frac{\breve{\nu}\,\breve{\tau}^2 + \sum_{n=1}^{N_{\mathcal{D}}}(y_n - \mu)^2}{\widehat{\nu}} \tag{3.49}$$

We see that the posterior dof $\widehat{\nu}$ is the prior dof $\breve{\nu}$ plus $N_{\mathcal{D}}$, and the posterior sum of squares $\widehat{\nu}\widehat{\tau}^2$ is the prior sum of squares $\breve{\nu}\breve{\tau}^2$ plus the data sum of squares.

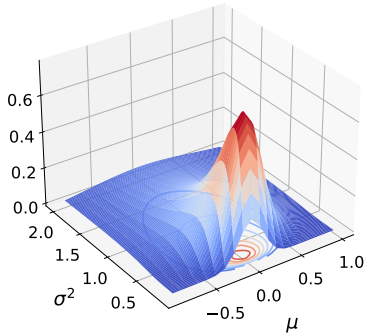
3.2.3.3 Posterior of μ and σ^2: conjugate prior

Now suppose we want to infer both the mean and variance. The corresponding conjugate prior is the **normal inverse Gamma**:

$$\text{NIG}(\mu, \sigma^2|\, \breve{m}, \breve{\kappa}, \breve{a}, \breve{b}) \triangleq \mathcal{N}(\mu|\, \breve{m}, \sigma^2/\, \breve{\kappa})\ \text{IG}(\sigma^2|\, \breve{a}, \breve{b}) \tag{3.50}$$



(a) $NI\chi^2(\mu_0 = 0, \kappa_0 = 1, \nu_0 = 1, \sigma_0^2 = 1)$



(b) $NI\chi^2(\mu_0 = 0, \kappa_0 = 5, \nu_0 = 1, \sigma_0^2 = 1)$

Figure 3.6: The $NI\chi^2(\mu, \sigma^2 | m, \kappa, \nu, \sigma^2)$ distribution. m is the prior mean and κ is how strongly we believe this; σ^2 is the prior variance and ν is how strongly we believe this. (a) $m = 0, \kappa = 1, \nu = 1, \sigma^2 = 1$. Notice that the contour plot (underneath the surface) is shaped like a "squashed egg". (b) We increase the strength of our belief in the mean by setting $\kappa = 5$, so the distribution for μ around $m = 0$ becomes narrower. Generated by nix_plots.ipynb.

However, it is common to use a reparameterization of this known as the **normal inverse chi-squared** or **NIX** distribution [Gel+14a, p67], which is defined by

$$NI\chi^2(\mu, \sigma^2 | \breve{m}, \breve{\kappa}, \breve{\nu}, \breve{\tau}^2) \triangleq \mathcal{N}(\mu | \breve{m}, \sigma^2 / \breve{\kappa}) \; \chi^{-2}(\sigma^2 | \breve{\nu}, \breve{\tau}^2) \tag{3.51}$$

$$\propto (\frac{1}{\sigma^2})^{(\breve{\nu}+3)/2} \exp\left(-\frac{\breve{\nu}\breve{\tau}^2 + \breve{\kappa}\,(\mu - \breve{m})^2}{2\sigma^2}\right) \tag{3.52}$$

See Figure 3.6 for some plots. Along the μ axis, the distribution is shaped like a Gaussian, and along the σ^2 axis, the distribution is shaped like a χ^{-2}; the contours of the joint density have a "squashed egg" appearance. Interestingly, we see that the contours for μ are more peaked for small values of σ^2, which makes sense, since if the data is low variance, we will be able to estimate its mean more reliably.

One can show (based on Section 3.3.3.3) that the posterior is given by

$$p(\mu, \sigma^2 | \mathcal{D}) = NI\chi^2(\mu, \sigma^2 | \widehat{m}, \widehat{\kappa}, \widehat{\nu}, \widehat{\tau}^2) \tag{3.53}$$

$$\widehat{m} = \frac{\breve{\kappa}\breve{m} + N\overline{x}}{\widehat{\kappa}} \tag{3.54}$$

$$\widehat{\kappa} = \breve{\kappa} + N \tag{3.55}$$

$$\widehat{\nu} = \breve{\nu} + N \tag{3.56}$$

$$\widehat{\nu}\widehat{\tau}^2 = \breve{\nu}\breve{\tau}^2 + \sum_{n=1}^{N}(y_n - \overline{y})^2 + \frac{N\,\breve{\kappa}}{\breve{\kappa} + N}(\breve{m} - \overline{y})^2 \tag{3.57}$$

The interpretation of this is as follows. For μ, the posterior mean $\widehat{m}$ is a convex combination of the prior mean $\breve{m}$ and the MLE $\overline{x}$; the strength of this posterior, $\widehat{\kappa}$, is the prior strength $\breve{\kappa}$ plus the

number of data points N. For σ^2, we work instead with the sum of squares: the posterior sum of squares, $\widehat{\nu}\widehat{\tau}^2$, is the prior sum of squares $\breve{\nu}\breve{\tau}^2$ plus the data sum of squares, $\sum_{n=1}^{N}(y_n - \overline{y})^2$, plus a term due to the discrepancy between the prior mean $\breve{m}$ and the MLE $\overline{y}$. The strength of this posterior, $\widehat{\nu}$, is the prior strength $\breve{\nu}$ plus the number of data points N;

The posterior marginal for σ^2 is just

$$p(\sigma^2|\mathcal{D}) = \int p(\mu, \sigma^2|\mathcal{D})d\mu = \chi^{-2}(\sigma^2|\ \widehat{\nu}, \widehat{\tau}^2) \tag{3.58}$$

with the posterior mean given by $\mathbb{E}\left[\sigma^2|\mathcal{D}\right] = \frac{\widehat{\nu}}{\widehat{\nu}-2}\ \widehat{\tau}^2$.

The posterior marginal for μ has a Student distribution, which follows from the fact that the Student distribution is a (scaled) mixture of Gaussians:

$$p(\mu|\mathcal{D}) = \int p(\mu, \sigma^2|D)d\sigma^2 = \mathcal{T}(\mu|\ \widehat{m}, \widehat{\tau}^2\ /\ \widehat{\kappa}, \widehat{\nu}) \tag{3.59}$$

with the posterior mean given by $\mathbb{E}\left[\mu|\mathcal{D}\right] = \widehat{m}$.

3.2.3.4 Posterior of μ and σ^2: uninformative prior

If we "know nothing" about the parameters a priori, we can use an uniformative prior. We discuss how to create such priors in Section 3.6. A common approach is to use a Jeffreys prior. In Section 3.6.2.3, we show that the Jeffreys prior for a location and scale parameter has the form

$$p(\mu, \sigma^2) \propto p(\mu)p(\sigma^2) \propto \sigma^{-2} \tag{3.60}$$

We can simulate this with a conjugate prior by using

$$p(\mu, \sigma^2) = NI\chi^2(\mu, \sigma^2|\ \breve{m}{=}\ 0, \breve{\kappa}{=}\ 0, \breve{\nu}{=}\ -1, \breve{\tau}^2{=}\ 0) \tag{3.61}$$

With this prior, the posterior has the form

$$p(\mu, \sigma^2|\mathcal{D}) = NI\chi^2(\mu, \sigma^2|\ \widehat{m}{=}\ \overline{y}, \widehat{\kappa}{=}\ N, \widehat{\nu}{=}\ N-1, \widehat{\tau}^2{=}\ s^2) \tag{3.62}$$

where

$$s^2 \triangleq \frac{1}{N-1}\sum_{n=1}^{N}(y_n - \overline{y})^2 = \frac{N}{N-1}\hat{\sigma}^2_{\text{mle}} \tag{3.63}$$

s is known as the **sample standard deviation**. Hence the marginal posterior for the mean is given by

$$p(\mu|\mathcal{D}) = \mathcal{T}(\mu|\overline{y}, \frac{s^2}{N}, N-1) = \mathcal{T}(\mu|\overline{y}, \frac{\sum_{n=1}^{N}(y_n - \overline{y})^2}{N(N-1)}, N-1) \tag{3.64}$$

Thus the posterior variance of μ is

$$\mathbb{V}\left[\mu|\mathcal{D}\right] = \frac{\widehat{\nu}}{\widehat{\nu}\ -2}\ \widehat{\tau}^2{=}\ \frac{N-1}{N-3}\frac{s^2}{N} \rightarrow \frac{s^2}{N} \tag{3.65}$$

The square root of this is called the **standard error of the mean**:

$$\text{se}(\mu) \triangleq \sqrt{\mathbb{V}\left[\mu|\mathcal{D}\right]} \approx \frac{s}{\sqrt{N}} \tag{3.66}$$

Thus we can approximate the 95% **credible interval** for μ using

$$I_{.95}(\mu|\mathcal{D}) = \overline{y} \pm 2\frac{s}{\sqrt{N}} \tag{3.67}$$

3.3 Conjugate priors for the multivariate Gaussian

In this section, we derive the posterior $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D})$ for a multivariate Gaussian. For simplicity, we consider this in three steps: inferring just $\boldsymbol{\mu}$, inferring just $\boldsymbol{\Sigma}$, and then inferring both.

3.3.1 Posterior of $\boldsymbol{\mu}$ given $\boldsymbol{\Sigma}$

The likelihood has the form

$$p(\mathcal{D}|\boldsymbol{\mu}) = \mathcal{N}(\overline{\boldsymbol{y}}|\boldsymbol{\mu}, \frac{1}{N}\boldsymbol{\Sigma}) \tag{3.68}$$

For simplicity, we will use a conjugate prior, which in this case is a Gaussian. In particular, if $p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\ \breve{\boldsymbol{m}}, \breve{\mathbf{V}})$ then we can derive a Gaussian posterior for $\boldsymbol{\mu}$ based on the results in Section 2.2.6.2 We get

$$p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu}|\ \widehat{\boldsymbol{m}}, \widehat{\mathbf{V}}) \tag{3.69}$$

$$\widehat{\mathbf{V}}^{-1} = \breve{\mathbf{V}}^{-1} + N_{\mathcal{D}}\boldsymbol{\Sigma}^{-1} \tag{3.70}$$

$$\widehat{\boldsymbol{m}} = \widehat{\mathbf{V}}\ (\boldsymbol{\Sigma}^{-1}(N_{\mathcal{D}}\overline{\boldsymbol{y}}) + \breve{\mathbf{V}}^{-1}\breve{\boldsymbol{m}}) \tag{3.71}$$

Figure 3.7 gives a 2d example of these results.

3.3.2 Posterior of $\boldsymbol{\Sigma}$ given $\boldsymbol{\mu}$

We now discuss how to compute $p(\boldsymbol{\Sigma}|\mathcal{D}, \boldsymbol{\mu})$.

3.3.2.1 Likelihood

We can rewrite the likelihood as follows:

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-\frac{N_{\mathcal{D}}}{2}} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{S}_{\mu}\boldsymbol{\Sigma}^{-1})\right) \tag{3.72}$$

where

$$\mathbf{S}_{\mu} \triangleq \sum_{n=1}^{N}(\boldsymbol{y}_n - \boldsymbol{\mu})(\boldsymbol{y}_n - \boldsymbol{\mu})^{\mathsf{T}} \tag{3.73}$$

is the scatter matrix around $\boldsymbol{\mu}$.

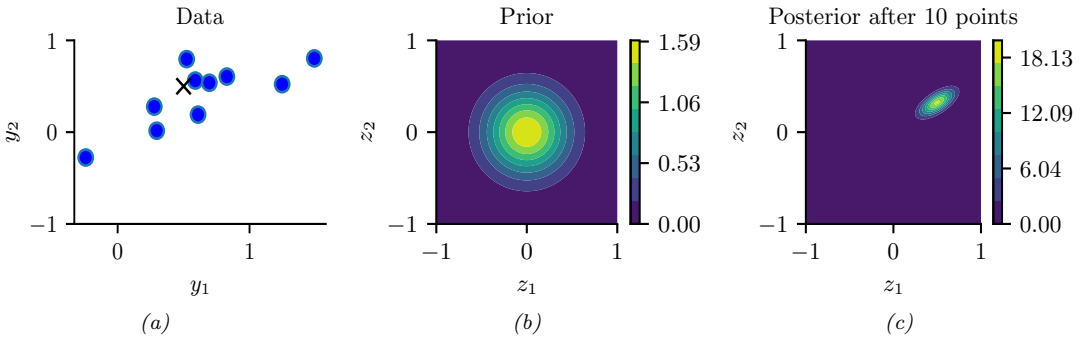


Figure 3.7: Illustration of Bayesian inference for a 2d Gaussian random vector $\boldsymbol{z}$. (a) The data is generated from $\boldsymbol{y}_n \sim \mathcal{N}(\boldsymbol{z}, \boldsymbol{\Sigma}_y)$, where $\boldsymbol{z} = [0.5, 0.5]^\mathsf{T}$ and $\boldsymbol{\Sigma}_y = 0.1([2, 1; 1, 1])$. We assume the sensor noise covariance $\boldsymbol{\Sigma}_y$ is known but $\boldsymbol{z}$ is unknown. The black cross represents $\boldsymbol{z}$. (b) The prior is $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, 0.1\mathbf{I}_2)$. (c) We show the posterior after 10 data points have been observed. Generated by gauss_infer_2d.ipynb.

3.3.2.2 Prior

The conjugate prior is known as the **inverse Wishart** distribution, which is a distribution over positive definite matrices, as we explained in Section 2.2.8.5. This has the following pdf:

$$\mathrm{IW}(\boldsymbol{\Sigma}|\, \breve{\boldsymbol{\Psi}}, \breve{\nu}) \propto |\boldsymbol{\Sigma}|^{-(\breve{\nu}+D+1)/2} \exp\left(-\frac{1}{2}\mathrm{tr}(\breve{\boldsymbol{\Psi}}\, \boldsymbol{\Sigma}^{-1})\right) \tag{3.74}$$

Here $\breve{\nu} > D-1$ is the degrees of freedom (dof), and $\breve{\boldsymbol{\Psi}}$ is a symmetric pd matrix. We see that $\breve{\boldsymbol{\Psi}}$ plays the role of the prior scatter matrix, and $N_0 \triangleq \breve{\nu} + D + 1$ controls the strength of the prior, and hence plays a role analogous to the sample size $N_{\mathcal{D}}$.

3.3.2.3 Posterior

Multiplying the likelihood and prior we find that the posterior is also inverse Wishart:

$$\begin{aligned} p(\boldsymbol{\Sigma}|\mathcal{D}, \boldsymbol{\mu}) &\propto |\boldsymbol{\Sigma}|^{-\frac{N_{\mathcal{D}}}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_\mu)\right) |\boldsymbol{\Sigma}|^{-(\breve{\nu}+D+1)/2} \\ &\quad \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\, \breve{\boldsymbol{\Psi}})\right) && (3.75) \\ &= |\boldsymbol{\Sigma}|^{-\frac{N_{\mathcal{D}}+(\breve{\nu}+D+1)}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left[\boldsymbol{\Sigma}^{-1}(\mathbf{S}_\mu + \breve{\boldsymbol{\Psi}})\right]\right) && (3.76) \\ &= \mathrm{IW}(\boldsymbol{\Sigma}|\, \widehat{\boldsymbol{\Psi}}, \widehat{\nu}) && (3.77) \\ \widehat{\nu} &= \breve{\nu} + N_{\mathcal{D}} && (3.78) \\ \widehat{\boldsymbol{\Psi}} &= \breve{\boldsymbol{\Psi}} + \mathbf{S}_\mu && (3.79) \end{aligned}$$

In words, this says that the posterior strength $\widehat{\nu}$ is the prior strength $\breve{\nu}$ plus the number of observations $N_{\mathcal{D}}$, and the posterior scatter matrix $\widehat{\boldsymbol{\Psi}}$ is the prior scatter matrix $\breve{\boldsymbol{\Psi}}$ plus the data scatter matrix $\mathbf{S}_\mu$.

3.3.3 Posterior of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$

In this section, we compute $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D})$ using a conjugate prior.

3.3.3.1 Likelihood

The likelihood is given by

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-\frac{N_{\mathcal{D}}}{2}} \exp\left(-\frac{1}{2}\sum_{n=1}^{N_{\mathcal{D}}}(\boldsymbol{y}_n - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}_n - \boldsymbol{\mu})\right) \tag{3.80}$$

One can show that

$$\sum_{n=1}^{N_{\mathcal{D}}}(\boldsymbol{y}_n - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}_n - \boldsymbol{\mu}) = \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}) + N(\overline{\boldsymbol{y}} - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\overline{\boldsymbol{y}} - \boldsymbol{\mu}) \tag{3.81}$$

where

$$\mathbf{S} \triangleq \mathbf{S}_{\overline{\boldsymbol{y}}} = \sum_{n=1}^{N}(\boldsymbol{y}_n - \overline{\boldsymbol{y}})(\boldsymbol{y}_n - \overline{\boldsymbol{y}})^{\mathsf{T}} = \mathbf{Y}^{\mathsf{T}}\mathbf{C}_N\mathbf{Y} \tag{3.82}$$

is empirical **scatter matrix**, and $\mathbf{C}_N$ is the **centering matrix**

$$\mathbf{C}_N \triangleq \mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^{\mathsf{T}} \tag{3.83}$$

Hence we can rewrite the likelihood as follows:

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-\frac{N_{\mathcal{D}}}{2}} \exp\left(-\frac{N_{\mathcal{D}}}{2}(\boldsymbol{\mu} - \overline{\boldsymbol{y}})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \overline{\boldsymbol{y}})\right) \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S})\right) \tag{3.84}$$

We will use this form below.

3.3.3.2 Prior

The obvious prior to use is the following

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu}|\, \breve{\boldsymbol{m}}, \breve{\mathbf{V}})\mathrm{IW}(\boldsymbol{\Sigma}|\, \breve{\boldsymbol{\Psi}}, \breve{\nu}) \tag{3.85}$$

where IW is the inverse Wishart distribution. Unfortunately, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ appear together in a non-factorized way in the likelihood in Equation (3.84) (see the first exponent term), so the factored prior in Equation (3.85) is not conjugate to the likelihood.[2]

The above prior is sometimes called **conditionally conjugate**, since both conditionals, $p(\boldsymbol{\mu}|\boldsymbol{\Sigma})$ and $p(\boldsymbol{\Sigma}|\boldsymbol{\mu})$, are individually conjugate. To create a fully conjugate prior, we need to use a prior where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are dependent on each other. We will use a joint distribution of the form $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\boldsymbol{\mu}|\boldsymbol{\Sigma})p(\boldsymbol{\Sigma})$.

2. Using the language of directed graphical models, we see that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ become dependent when conditioned on $\mathcal{D}$ due to explaining away. See Figure 3.8(a).

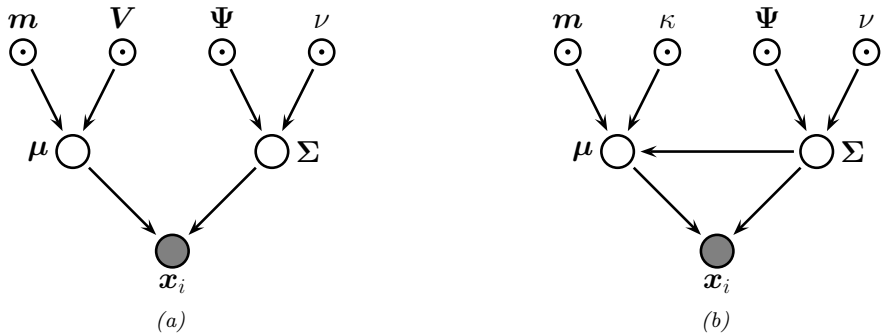


Figure 3.8: Graphical models representing different kinds of assumptions about the parameter priors. (a) A semi-conjugate prior for a Gaussian. (b) A conjugate prior for a Gaussian.

Looking at the form of the likelihood equation, Equation (3.84), we see that a natural conjugate prior has the form of a **Normal-inverse-Wishart** or **NIW** distribution, defined as follows:

$$\text{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \breve{\boldsymbol{m}}, \breve{\kappa}, \breve{\nu}, \breve{\boldsymbol{\Psi}}) \triangleq \mathcal{N}(\boldsymbol{\mu} | \breve{\boldsymbol{m}}, \frac{1}{\breve{\kappa}}\boldsymbol{\Sigma}) \times \text{IW}(\boldsymbol{\Sigma} | \breve{\boldsymbol{\Psi}}, \breve{\nu}) \tag{3.86}$$

$$= \frac{1}{Z_{\text{NIW}}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{\breve{\kappa}}{2}(\boldsymbol{\mu} - \breve{\boldsymbol{m}})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \breve{\boldsymbol{m}})\right)$$

$$\times |\boldsymbol{\Sigma}|^{-\frac{\breve{\nu}+D+1}{2}} \exp\left(-\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}^{-1} \breve{\boldsymbol{\Psi}})\right) \tag{3.87}$$

where the normalization constant is given by

$$Z_{\text{NIW}} \triangleq 2^{\breve{\nu} D/2} \Gamma_D(\breve{\nu}/2)(2\pi/\breve{\kappa})^{D/2} |\breve{\boldsymbol{\Psi}}|^{-\breve{\nu}/2} \tag{3.88}$$

The parameters of the NIW can be interpreted as follows: $\breve{\boldsymbol{m}}$ is our prior mean for $\boldsymbol{\mu}$, and $\breve{\kappa}$ is how strongly we believe this prior; $\breve{\boldsymbol{\Psi}}$ is (proportional to) our prior mean for $\boldsymbol{\Sigma}$, and $\breve{\nu}$ is how strongly we believe this prior.[3]

3.3.3.3 Posterior

To derive the posterior, let us first rewrite the scatter matrix as follows:

$$\mathbf{S} = \mathbf{Y}^{\mathsf{T}}\mathbf{Y} - \frac{1}{N}(\sum_{n=1}^{N} \boldsymbol{y}_n)(\sum_{n=1}^{N} \boldsymbol{y}_n)^{\mathsf{T}} = \mathbf{Y}^{\mathsf{T}}\mathbf{Y} - N\overline{\boldsymbol{y}}\overline{\boldsymbol{y}}^{\mathsf{T}} \tag{3.89}$$

where $\mathbf{Y}^{\mathsf{T}}\mathbf{Y} = \sum_{n=1}^{N} \boldsymbol{y}_n \boldsymbol{y}_n^{\mathsf{T}}$ is the **sum of squares** matrix.

3. Note that our uncertainty in the mean is proportional to the covariance. In particular, if we believe that the variance is large, then our uncertainty in μ must be large too. This makes sense intuitively, since if the data has large spread, it will be hard to pin down its mean.

Now we can multiply the likelihood and the prior to give

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) \propto |\boldsymbol{\Sigma}|^{-\frac{N_{\mathcal{D}}}{2}} \exp\left(-\frac{N_{\mathcal{D}}}{2}(\boldsymbol{\mu} - \overline{\boldsymbol{y}})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \overline{\boldsymbol{y}})\right) \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S})\right) \tag{3.90}$$

$$\times |\boldsymbol{\Sigma}|^{-\frac{\breve{\nu}+D+2}{2}} \exp\left(-\frac{\breve{\kappa}}{2}(\boldsymbol{\mu} - \breve{\boldsymbol{m}})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \breve{\boldsymbol{m}})\right) \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\breve{\boldsymbol{\Psi}})\right) \tag{3.91}$$

$$= |\boldsymbol{\Sigma}|^{-(N_{\mathcal{D}}+\breve{\nu}+D+2)/2} \exp(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{M})) \tag{3.92}$$

where

$$\mathbf{M} \triangleq N(\boldsymbol{\mu} - \overline{\boldsymbol{y}})(\boldsymbol{\mu} - \overline{\boldsymbol{y}})^{\mathsf{T}} + \breve{\kappa}\,(\boldsymbol{\mu} - \breve{\boldsymbol{m}})(\boldsymbol{\mu} - \breve{\boldsymbol{m}})^{\mathsf{T}} + \mathbf{S} + \breve{\boldsymbol{\Psi}} \tag{3.93}$$

$$= (\breve{\kappa} + N)\boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}} - \boldsymbol{\mu}(\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}})^{\mathsf{T}} - (\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}})\boldsymbol{\mu}^{\mathsf{T}} + \breve{\kappa}\breve{\boldsymbol{m}}\breve{\boldsymbol{m}}^{\mathsf{T}} + \mathbf{Y}^{\mathsf{T}}\mathbf{Y} + \breve{\boldsymbol{\Psi}} \tag{3.94}$$

We can simplify the $\mathbf{M}$ matrix using a trick called **completing the square** Applying this to the above, we have

$$(\breve{\kappa} + N)\boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}} - \boldsymbol{\mu}(\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}})^{\mathsf{T}} - (\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}})\boldsymbol{\mu}^{\mathsf{T}} \tag{3.95}$$

$$= (\breve{\kappa} + N)\left(\boldsymbol{\mu} - \frac{\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}}}{\breve{\kappa} + N}\right)\left(\boldsymbol{\mu} - \frac{\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}}}{\breve{\kappa} + N}\right)^{\mathsf{T}} \tag{3.96}$$

$$- \frac{(\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}})(\breve{\kappa}\breve{\boldsymbol{m}} + N\overline{\boldsymbol{y}})^{\mathsf{T}}}{\breve{\kappa} + N} \tag{3.97}$$

$$= \widehat{\kappa}\,(\boldsymbol{\mu} - \widehat{\boldsymbol{m}})(\boldsymbol{\mu} - \widehat{\boldsymbol{m}})^{\mathsf{T}} - \widehat{\kappa}\widehat{\boldsymbol{m}}\widehat{\boldsymbol{m}}^{\mathsf{T}} \tag{3.98}$$

Hence we can rewrite the posterior as follows:

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) \propto |\boldsymbol{\Sigma}|^{(\widehat{\nu}+D+2)/2} \exp\left(-\frac{1}{2}\mathrm{tr}\left[\boldsymbol{\Sigma}^{-1}\left(\widehat{\kappa}\,(\boldsymbol{\mu} - \widehat{\boldsymbol{m}})(\boldsymbol{\mu} - \widehat{\boldsymbol{m}})^{\mathsf{T}} + \widehat{\boldsymbol{\Psi}}\right)\right]\right) \tag{3.99}$$

$$= \mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\, \widehat{\boldsymbol{m}}, \widehat{\kappa}, \widehat{\nu}, \widehat{\boldsymbol{\Psi}}) \tag{3.100}$$

where

$$\widehat{\boldsymbol{m}} = \frac{\breve{\kappa}\breve{\boldsymbol{m}} + N_{\mathcal{D}}\overline{\boldsymbol{y}}}{\widehat{\kappa}} = \frac{\breve{\kappa}}{\breve{\kappa} + N_{\mathcal{D}}}\,\breve{\boldsymbol{m}} + \frac{N_{\mathcal{D}}}{\breve{\kappa} + N_{\mathcal{D}}}\overline{\boldsymbol{y}} \tag{3.101}$$

$$\widehat{\kappa} = \breve{\kappa} + N_{\mathcal{D}} \tag{3.102}$$

$$\widehat{\nu} = \breve{\nu} + N_{\mathcal{D}} \tag{3.103}$$

$$\widehat{\boldsymbol{\Psi}} = \breve{\boldsymbol{\Psi}} + \mathbf{S} + \frac{\breve{\kappa}\, N_{\mathcal{D}}}{\breve{\kappa} + N_{\mathcal{D}}}(\overline{\boldsymbol{y}} - \breve{\boldsymbol{m}})(\overline{\boldsymbol{y}} - \breve{\boldsymbol{m}})^{\mathsf{T}} \tag{3.104}$$

$$= \breve{\boldsymbol{\Psi}} + \mathbf{Y}^{\mathsf{T}}\mathbf{Y} + \breve{\kappa}\breve{\boldsymbol{m}}\breve{\boldsymbol{m}}^{\mathsf{T}} - \widehat{\kappa}\widehat{\boldsymbol{m}}\widehat{\boldsymbol{m}}^{\mathsf{T}} \tag{3.105}$$

This result is actually quite intuitive: the posterior mean $\widehat{\boldsymbol{m}}$ is a convex combination of the prior mean and the MLE; the posterior scatter matrix $\widehat{\boldsymbol{\Psi}}$ is the prior scatter matrix $\breve{\boldsymbol{\Psi}}$ plus the empirical scatter matrix $\mathbf{S}$ plus an extra term due to the uncertainty in the mean (which creates its own virtual scatter matrix); and the posterior confidence factors $\widehat{\kappa}$ and $\widehat{\nu}$ are both incremented by the size of the data we condition on.

3.3.3.4 Posterior marginals

We have computed the joint posterior

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\Sigma}, \mathcal{D})p(\boldsymbol{\Sigma}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}|\ \widehat{\boldsymbol{m}}, \frac{1}{\widehat{\kappa}}\boldsymbol{\Sigma})\text{IW}(\boldsymbol{\Sigma}|\ \widehat{\boldsymbol{\Psi}}, \widehat{\nu}) \tag{3.106}$$

We now discuss how to compute the posterior marginals, $p(\boldsymbol{\Sigma}|\mathcal{D})$ and $p(\boldsymbol{\mu}|\mathcal{D})$.

It is easy to see that the posterior marginal for $\boldsymbol{\Sigma}$ is

$$p(\boldsymbol{\Sigma}|\mathcal{D}) = \int p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D})d\boldsymbol{\mu} = \text{IW}(\boldsymbol{\Sigma}|\ \widehat{\boldsymbol{\Psi}}, \widehat{\nu}) \tag{3.107}$$

For the mean, one can show that

$$p(\boldsymbol{\mu}|\mathcal{D}) = \int p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D})d\boldsymbol{\Sigma} = \mathcal{T}(\boldsymbol{\mu}|\ \widehat{\boldsymbol{\mu}}, \frac{\widehat{\boldsymbol{\Psi}}}{\widehat{\kappa}\widehat{\nu}'}, \widehat{\nu}') \tag{3.108}$$

where $\widehat{\nu}' \triangleq \widehat{\nu} - D + 1$. Intuitively this result follows because $p(\boldsymbol{\mu}|\mathcal{D})$ is an infinite mixture of Gaussians, where each mixture component has a value of $\boldsymbol{\Sigma}$ drawn from the IW distribution; by mixing these altogether, we induce a Student distribution, which has heavier tails than a single Gaussian.

3.3.3.5 Posterior mode

The maximum a posteriori (MAP) estimate of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is the mode of the posterior NIW distribution with density

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{Y}) = \mathcal{N}(\boldsymbol{\mu}|\ \widehat{\boldsymbol{\mu}}, \widehat{\kappa}^{-1}\ \boldsymbol{\Sigma})\text{IW}(\boldsymbol{\Sigma}|\ \widehat{\nu}, \widehat{\boldsymbol{\Psi}}) \tag{3.109}$$

To find the mode, we firstly notice that $\boldsymbol{\mu}$ only appears in the conditional distribution $\mathcal{N}(\boldsymbol{\mu}|\ \widehat{\boldsymbol{\mu}}, \widehat{\kappa}^{-1}\ \boldsymbol{\Sigma})$, and the mode of this normal distribution equals its mean, i.e., $\boldsymbol{\mu} = \widehat{\boldsymbol{\mu}}$. Also notice that this holds for any choice of $\boldsymbol{\Sigma}$. So we can plug $\boldsymbol{\mu} = \widehat{\boldsymbol{\mu}}$ in Equation (3.109) and derive the mode of $\boldsymbol{\Sigma}$. Notice that

$$-2 * \log p(\boldsymbol{\mu} = \widehat{\boldsymbol{\mu}}, \boldsymbol{\Sigma}|\mathbf{Y}) = (\widehat{\nu} + D + 2)\log(|\boldsymbol{\Sigma}|) + \text{trace}(\widehat{\boldsymbol{\Psi}}\ \boldsymbol{\Sigma}^{-1}) + c \tag{3.110}$$

where c is a constant irrelevant to $\boldsymbol{\Sigma}$. We then take the derivative over $\boldsymbol{\Sigma}$:

$$\frac{\partial \log p(\boldsymbol{\mu} = \widehat{\boldsymbol{\mu}}\ \boldsymbol{\Sigma}|\mathbf{Y})}{\partial \boldsymbol{\Sigma}} = (\widehat{\nu} + D + 2)\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\ \widehat{\boldsymbol{\Psi}}\ \boldsymbol{\Sigma}^{-1} \tag{3.111}$$

By setting the derivative to 0 and solve for $\boldsymbol{\Sigma}$, we see that $(\widehat{\nu} + D + 2)^{-1}\ \widehat{\boldsymbol{\Psi}}$ is the matrix that maximizes Equation (3.110). By checking that $\widehat{\boldsymbol{\Psi}}$ is a positive definite matrix, we conclude that $\widehat{\boldsymbol{\Psi}}$ is the MAP estimate of the covariance matrix $\boldsymbol{\Sigma}$.

In conclusion, the MAP estimate of $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ are

$$\hat{\boldsymbol{\mu}} = \frac{\breve{\kappa}\breve{\boldsymbol{\mu}} + N\bar{\boldsymbol{y}}}{\breve{\kappa} + N} \tag{3.112}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{\widehat{\nu} + D + 2}\ \widehat{\boldsymbol{\Psi}} \tag{3.113}$$

3.3.3.6 Posterior predictive

We now discuss how to predict future data by integrating out the parameters. If $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) \sim \text{NIW}(\widehat{\boldsymbol{m}}, \widehat{\kappa}, \widehat{\nu}, \widehat{\boldsymbol{\Psi}})$, then one can show that the posterior predictive distribution, for a single observation vector, is as follows:

$$p(\boldsymbol{y}|\mathcal{D}) = \int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\text{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\, \widehat{\boldsymbol{m}}, \widehat{\kappa}, \widehat{\nu}, \widehat{\boldsymbol{\Psi}})d\boldsymbol{\mu}d\boldsymbol{\Sigma} \tag{3.114}$$

$$= \mathcal{T}(\boldsymbol{y}|\, \widehat{\boldsymbol{m}}, \frac{\widehat{\boldsymbol{\Psi}}\ (\widehat{\kappa}+1)}{\widehat{\kappa}\widehat{\nu}'}, \widehat{\nu}') \tag{3.115}$$

where $\widehat{\nu}'=\widehat{\nu}-D+1$.

3.4 Conjugate priors for the exponential family

We have seen that exact Bayesian analysis is considerably simplified if the prior is conjugate to the likelihood. Since the posterior must have the same form as the prior, and hence the same number of parameters, the likelihood function must have fixed-sized sufficient statistics, so that we can write $p(\mathcal{D}|\boldsymbol{\theta}) = p(\boldsymbol{s}(\mathcal{D})|\boldsymbol{\theta})$. This suggests that the only family of distributions for which conjugate priors exist is the exponential family, a result proved in [DY79].[4] In the sections below, we show how to perform conjugate analysis for a generic exponential family model.

3.4.0.1 Likelihood

Recall that the likelihood of the exponential family is given by

$$p(\mathcal{D}|\boldsymbol{\eta}) = h(\mathcal{D}) \exp(\boldsymbol{\eta}^\mathsf{T} \boldsymbol{s}(\mathcal{D}) - N_\mathcal{D} A(\boldsymbol{\eta})) \tag{3.116}$$

where $\boldsymbol{s}(\mathcal{D}) = \sum_{n=1}^N \boldsymbol{s}(\boldsymbol{x}_n)$ and $h(\mathcal{D}) \triangleq \prod_{n=1}^N h(\boldsymbol{x}_n)$.

3.4.0.2 Prior

We will write the prior in a form that mirrors the likelihood:

$$p(\boldsymbol{\eta}|\, \breve{\boldsymbol{\tau}}, \breve{\nu}) = \frac{1}{Z(\breve{\boldsymbol{\tau}}, \breve{\nu})} \exp(\breve{\boldsymbol{\tau}}^\mathsf{T}\, \boldsymbol{\eta} - \breve{\nu}\, A(\boldsymbol{\eta})) \tag{3.117}$$

where $\breve{\nu}$ is the strength of the prior, and $\breve{\boldsymbol{\tau}}$ / $\breve{\nu}$ is the prior mean, and $Z(\breve{\boldsymbol{\tau}}, \breve{\nu})$ is a normalizing factor. The parameters $\breve{\boldsymbol{\tau}}$ can be derived from **virtual samples** representing our prior beliefs.

4. There are some exceptions. For example, the uniform distribution $\text{Unif}(x|0, \theta)$ has finite sufficient statistics $(N, m = \max_i x_i)$, as discussed in Section 2.3.2.6; hence this distribution has a conjugate prior, namely the Pareto distribution (Section 2.2.3.5), $p(\theta) = \text{Pareto}(\theta|\theta_0, \kappa)$, yielding the posterior $p(\theta|\boldsymbol{x}) = \text{Pareto}(\max(\theta_0, m), \kappa + N)$.

3.4.0.3 Posterior

The posterior is given by

$$p(\boldsymbol{\eta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\eta})p(\boldsymbol{\eta})}{p(\mathcal{D})} \tag{3.118}$$

$$= \frac{h(\mathcal{D})}{Z(\breve{\boldsymbol{\tau}}, \breve{\nu})p(\mathcal{D})} \exp\left((\breve{\boldsymbol{\tau}} + \boldsymbol{s}(\mathcal{D}))^{\mathsf{T}}\boldsymbol{\eta} - (\breve{\nu} + N_{\mathcal{D}})A(\boldsymbol{\eta})\right) \tag{3.119}$$

$$= \frac{1}{Z(\widehat{\boldsymbol{\tau}}, \widehat{\nu})} \exp\left(\widehat{\boldsymbol{\tau}}^{\mathsf{T}}\,\boldsymbol{\eta} - \widehat{\nu}\,A(\boldsymbol{\eta})\right) \tag{3.120}$$

where

$$\widehat{\boldsymbol{\tau}} = \breve{\boldsymbol{\tau}} + \boldsymbol{s}(\mathcal{D}) \tag{3.121}$$

$$\widehat{\nu} = \breve{\nu} + N_{\mathcal{D}} \tag{3.122}$$

$$Z(\widehat{\boldsymbol{\tau}}, \widehat{\nu}) = \frac{Z(\breve{\boldsymbol{\tau}}, \breve{\nu})}{h(\mathcal{D})} p(\mathcal{D}) \tag{3.123}$$

We see that this has the same form as the prior, but where we update the sufficient statistics and the sample size. We also see that the marginal likelihood is given by

$$p(\mathcal{D}) = \frac{Z(\widehat{\boldsymbol{\tau}}, \widehat{\nu})h(\mathcal{D})}{Z(\breve{\boldsymbol{\tau}}, \breve{\nu})} \tag{3.124}$$

The posterior mean is given by a convex combination of the prior mean and the empirical mean (which is the MLE):

$$\mathbb{E}\left[\boldsymbol{\eta}|\mathcal{D}\right] = \frac{\widehat{\boldsymbol{\tau}}}{\widehat{\nu}} = \frac{\breve{\boldsymbol{\tau}} + \boldsymbol{s}(\mathcal{D})}{\breve{\nu} + N_{\mathcal{D}}} = \frac{\breve{\nu}}{\breve{\nu} + N_{\mathcal{D}}} \frac{\breve{\boldsymbol{\tau}}}{\breve{\nu}} + \frac{N}{\breve{\nu} + N_{\mathcal{D}}} \frac{\boldsymbol{s}(\mathcal{D})}{N} \tag{3.125}$$

$$= \lambda \mathbb{E}\left[\boldsymbol{\eta}\right] + (1 - \lambda)\hat{\boldsymbol{\eta}}_{\text{mle}} \tag{3.126}$$

where $\lambda = \frac{\breve{\nu}}{\breve{\nu} + N_{\mathcal{D}}}$.

3.4.0.4 Posterior predictive density

We will now derive the predictive density for future observables $\mathcal{D}' = (\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_{N_{\mathcal{D}'}})$ given past data $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\mathcal{D}}})$:

$$p(\mathcal{D}'|\mathcal{D}) = \int p(\mathcal{D}'|\boldsymbol{\eta})p(\boldsymbol{\eta}|\mathcal{D})d\boldsymbol{\eta} \tag{3.127}$$

$$= \int h(\mathcal{D}') \exp(\boldsymbol{\eta}^{\mathsf{T}}\boldsymbol{s}(\mathcal{D}') - N'A(\boldsymbol{\eta})) \frac{1}{Z(\widehat{\boldsymbol{\tau}}, \widehat{\nu})} \exp(\boldsymbol{\eta}^{\mathsf{T}}\,\widehat{\boldsymbol{\tau}} - \widehat{\nu}\,A(\boldsymbol{\eta}))d\boldsymbol{\eta} \tag{3.128}$$

$$= h(\mathcal{D}') \frac{Z(\breve{\boldsymbol{\tau}} + \boldsymbol{s}(\mathcal{D}) + \boldsymbol{s}(\mathcal{D}'), \breve{\nu} + N + N')}{Z(\breve{\boldsymbol{\tau}} + \boldsymbol{s}(\mathcal{D}), \breve{\nu} + N)} \tag{3.129}$$

3.4.0.5 Example: Bernoulli distribution

As a simple example, let us revisit the Beta-Bernoulli model in our new notation.

The likelihood is given by

$$p(\mathcal{D}|\theta) = (1-\theta)^{N_{\mathcal{D}}} \exp\left(\log(\frac{\theta}{1-\theta}) \sum_i x_n\right) \tag{3.130}$$

Hence the conjugate prior is given by

$$p(\theta|\nu_0, \tau_0) \propto (1-\theta)^{\nu_0} \exp\left(\log(\frac{\theta}{1-\theta})\tau_0\right) \tag{3.131}$$

$$= \theta^{\tau_0}(1-\theta)^{\nu_0-\tau_0} \tag{3.132}$$

If we define $\alpha = \tau_0 + 1$ and $\beta = \nu_0 - \tau_0 + 1$, we see that this is a beta distribution.

We can derive the posterior as follows, where $s = \sum_i \mathbb{I}(x_i = 1)$ is the sufficient statistic:

$$p(\theta|\mathcal{D}) \propto \theta^{\tau_0+s}(1-\theta)^{\nu_0-\tau_0+n-s} \tag{3.133}$$

$$= \theta^{\tau_n}(1-\theta)^{\nu_n-\tau_n} \tag{3.134}$$

We can derive the posterior predictive distribution as follows. Assume $p(\theta) = \text{Beta}(\theta|\alpha, \beta)$, and let $s = s(\mathcal{D})$ be the number of heads in the past data. We can predict the probability of a given sequence of future heads, $\mathcal{D}' = (\tilde{x}_1, \ldots, \tilde{x}_m)$, with sufficient statistic $s' = \sum_{n=1}^m \mathbb{I}(\tilde{x}_i = 1)$, as follows:

$$p(\mathcal{D}'|\mathcal{D}) = \int_0^1 p(\mathcal{D}'|\theta|\text{Beta}(\theta|\alpha_n, \beta_n)d\theta \tag{3.135}$$

$$= \frac{\Gamma(\alpha_n+\beta_n)}{\Gamma(\alpha_n)\Gamma(\beta_n)} \int_0^1 \theta^{\alpha_n+t'-1}(1-\theta)^{\beta_n+m-t'-1}d\theta \tag{3.136}$$

$$= \frac{\Gamma(\alpha_n+\beta_n)}{\Gamma(\alpha_n)\Gamma(\beta_n)} \frac{\Gamma(\alpha_{n+m})\Gamma(\beta_{n+m})}{\Gamma(\alpha_{n+m}+\beta_{n+m})} \tag{3.137}$$

where

$$\alpha_{n+m} = \alpha_n + s' = \alpha + s + s' \tag{3.138}$$

$$\beta_{n+m} = \beta_n + (m - s') = \beta + (n - s) + (m - s') \tag{3.139}$$

3.5 Beyond conjugate priors

In Section 3.2, we saw various examples of conjugate priors, all of which have come from the exponential family (see Section 2.3). These priors have the advantage of being easy to interpret (in terms of sufficient statistics from a virtual prior dataset), and easy to compute with. However, for most models, there is no prior in the exponential family that is conjugate to the likelihood. Furthermore, even where there is a conjugate prior, the assumption of conjugacy may be too limiting. Therefore in the sections below, we briefly discuss various other kinds of priors. (We defer the question of posterior inference with these priors until Section 7.1, where we discuss algorithmic issues, since we can no longer use closed-form solutions when the prior is not conjugate.)

3.5.1 Robust (heavy-tailed) priors

The assessment of the influence of the prior on the posterior is called **sensitivity analysis**, or **robustness analysis**. There are many ways to create **robust priors**. (see e.g., [IR00]). Here we consider a simple approach, namely the use of a heavy-tailed distribution.

To motivate this, let us consider an example from [Ber85, p7]. Suppose $x \sim \mathcal{N}(\theta, 1)$. We observe that $x = 5$ and we want to estimate θ. The MLE is of course $\hat{\theta} = 5$, which seems reasonable. The posterior mean under a uniform prior is also $\overline{\theta} = 5$. But now suppose we know that the prior median is 0, and that there is 25% probability that θ lies in any of the intervals $(-\infty, -1)$, $(-1, 0)$, $(0, 1)$, $(1, \infty)$. Let us also assume the prior is smooth and unimodal.

One can show that that a Gaussian prior of the form $\mathcal{N}(\theta|0, 2.19^2)$ satisfies these prior constraints. But in this case the posterior mean is given by 3.43, which doesn't seem very satisfactory. An alternative distribution that captures the same prior information is the Cauchy prior $\mathcal{T}_1(\theta|0, 1)$. With this prior, we find (using numerical method integration: see robust_prior_demo.ipynb for the code) that the posterior mean is about 4.6, which seems much more reasonable. In general, priors with heavy tails tend to give results which are more sensitive to the data, which is usually what we desire.

Heavy-tailed priors are usually not conjugate. However, we can often approximate a heavy-tailed prior by using a (possibly infinite) mixture of conjugate priors. For example, in Section 28.2.3, we show that the Student distribution (of which the Cauchy is a special case) can be written as an infinite mixture of Gaussians, where the mixing weights come from a Gamma distribution. This is an example of a hierarchical prior; see Section 3.7 for details.

3.5.2 Priors for variance parameters

In this section, we discuss some commonly used priors for variance parameters. Such priors play an important role in determining how much regularization a model exhibits. For example, consider a linear regression model, $p(y|\boldsymbol{x}, \boldsymbol{w}, \sigma^2) = \mathcal{N}(y|\boldsymbol{w}^\mathsf{T}\boldsymbol{x}, \sigma^2)$. Suppose we use a Gaussian prior on the weights, $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, \tau^2\mathbf{I})$. The value of τ^2 (relative to σ^2) plays a role similar to the strength of an ℓ_2-regularization term in ridge regression. In the Bayesian setting, we need to ensure we use sensible priors for the variance parameters, τ^2 and σ^2. This becomes even more important when we discuss hierarchical models, in Section 3.7.

3.5.2.1 Priors for scalar variances

Consider trying to infer a variance parameter σ^2 from a Gaussian likelihood with known mean, as in Section 3.2.3.2. The uninformative prior is $p(\sigma^2) = \text{IG}(\sigma^2|0, 0)$, which is improper, meaning it does not integrate to 1. This is fine as long as the posterior is proper. This will be the case if the prior is on the variance of the noise of $N \geq 2$ observable variables. Unfortunately the posterior is not proper, even if $N \to \infty$, if we use this prior for the variance of the (non observable) weights in a regression model [Gel06; PS12], as we discuss in Section 3.7.

One solution to this is to use a **weakly informative** proper prior such as $\text{IG}(\epsilon, \epsilon)$ for small ϵ. However, this turns out to not work very well, for reasons that are explained in [Gel06; PS12]. Instead, it is recommended to use other priors, such as uniform, exponential, half-normal, half-Student-t, or half-Cauchy; all of these are bounded below by 0, and just require 1 or 2 hyperparameters. (The term "half" refers to the fact that the distribution is "folded over" onto itself on the positive side of the real axis.)

3.5.2.2 Priors for covariance matrices

The conjugate prior for a covariance matrix is the inverse Wishart (Section 2.2.8.6). However, it can be hard to set the parameters for this in an uninformative way. One approach, discussed in [HW13], is to use a scale mixture of inverse Wisharts, where the scaling parameters have inverse gamma distributions. It is possible to choose shape and scale parameters to ensure that all the correlation parameters have uniform $(-1, 1)$ marginals, and all the standard deviations have half-Student distributions.

Unfortunately, the Wishart distribution has heavy tails, which can lead to poor performance when used in a sampling algorithm.[5] A more common approach, following Equation (3.140), is to represent the $D \times D$ covariance matrix $\mathbf{\Sigma}$ in terms of a product of the marginal standard deviations, $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_D)$, and the $D \times D$ correlation matrix $\mathbf{R}$, as follows:

$$\mathbf{\Sigma} = \text{diag}(\boldsymbol{\sigma})\, \mathbf{R}\, \text{diag}(\boldsymbol{\sigma}) \tag{3.140}$$

For example, if $D = 2$,we have

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \tag{3.141}$$

We can put a factored prior on the standard deviations, following the recommendations of Section 3.5.2. For example,

$$p(\boldsymbol{\sigma}) = \prod_{d=1}^{D} \text{Expon}(\sigma_d | 1) \tag{3.142}$$

For the correlation matrix, it is common to use as a prior the **LKJ distribution**, named after the authors of [LKJ09]. This has the form

$$\text{LKJ}(\mathbf{R}|\eta) \propto |\mathbf{R}|^{\eta - 1} \tag{3.143}$$

so it only has one free parameter. When $\eta = 1$, it is a uniform prior; when $\eta = 2$, it is a "weakly regularizing" prior, that encourages small correlations (close to 0). See Figure 3.9 for a plot.

In practice, it is more common to define $\mathbf{R}$ in terms of its Cholesky decomposition, $\mathbf{R} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$, where $\mathbf{L}$ is an unconstrained lower triangular matrix. We then represent the prior using

$$\text{LKJchol}(\mathbf{L}|\eta) \propto |\mathbf{L}|^{-\eta - 1} \tag{3.144}$$

3.6 Noninformative priors

When we have little or no domain specific knowledge, it is desirable to use an **uninformative**, **noninformative** or **objective** priors, to "let the data speak for itself". Unfortunately, there is no unique way to define such priors, and they all encode some kind of knowledge. It is therefore better to use the term **diffuse prior**, **minimally informative prior** or **default prior**.

In the sections below, we briefly mention some common approaches for creating default priors. For further details, see e.g., [KW96] and the Stan website.[6]

5. See comments from Michael Betancourt at https://github.com/pymc-devs/pymc/issues/538.
6. https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations.

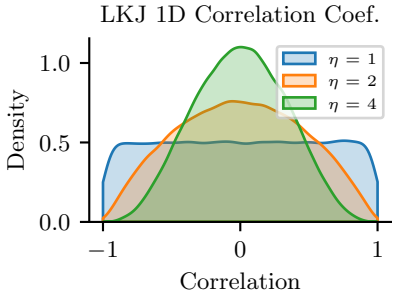


Figure 3.9: Distribution on the correlation coefficient ρ induced by a 2d LKJ distribution with varying parameter. Adapted from Figure 14.3 of [McE20]. Generated by lkj_1d.ipynb.

3.6.1 Maximum entropy priors

A natural way to define an uninformative prior is to use one that has **maximum entropy**, since it makes the least commitments to any particular value in the state space (see Section 5.2 for a discussion of entropy). This is a formalization of Laplace's **principle of insufficient reason**, in which he argued that if there is no reason to prefer one prior over another, we should pick a "flat" one.

For example, in the case of a Bernoulli distribution with rate $\theta \in [0, 1]$, the maximum entropy prior is the uniform distribution, $p(\theta) = \text{Beta}(\theta|1, 1)$, which makes intuitive sense.

However, in some cases we know something about our random variable $\boldsymbol{\theta}$, and we would like our prior to match these constraints, but otherwise be maximally entropic. More precisely, suppose we want to find a distribution $p(\boldsymbol{\theta})$ with maximum entropy, subject to the constraints that the expected values of certain features or functions $f_k(\boldsymbol{\theta})$ match some known quantities F_k. This is called a **maxent prior**. In Section 2.3.7, we show that such distributions must belong to the exponential family (Section 2.3).

For example, suppose $\theta \in \{1, 2, \ldots, 10\}$, and let $p_c = p(\theta = c)$ be the corresponding prior. Suppose we know that the prior mean is 1.5. We can encode this using the following constraint

$$\mathbb{E}\left[f_1(\theta)\right] = \mathbb{E}\left[\theta\right] = \sum_c c\, p_c = 1.5 \tag{3.145}$$

In addition, we have the constraint $\sum_c p_c = 1$. Thus we need to solve the following optimization problem:

$$\min_{\boldsymbol{p}} \mathbb{H}(\boldsymbol{p}) \quad \text{s.t.} \quad \sum_c c\, p_c = 1.5, \;\; \sum_c p_c = 1.0 \tag{3.146}$$

This gives the decaying exponential curve in Figure 3.10. Now suppose we know that θ is either 3 or 4 with probability 0.8. We can encode this using

$$\mathbb{E}\left[f_1(\theta)\right] = \mathbb{E}\left[\mathbb{I}\left(\theta \in \{3, 4\}\right)\right] = \Pr(\theta \in \{3, 4\}) = 0.8 \tag{3.147}$$

This gives the inverted U-curve in Figure 3.10. We note that this distribution is flat in as many places as possible.

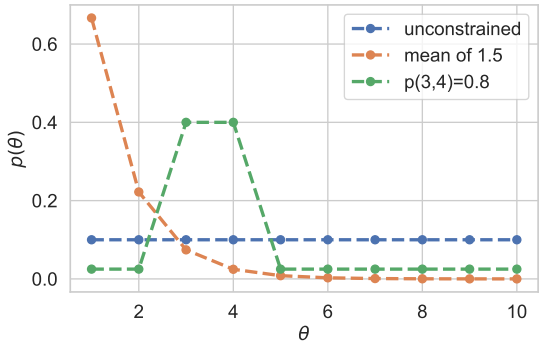


Figure 3.10: Illustration of 3 different maximum entropy priors. Adapted from Figure 1.10 of [MKL11]. Generated by maxent_priors.ipynb.

3.6.2 Jeffreys priors

Let θ be a random variable with prior $p_\theta(\theta)$, and let $\phi = f(\theta)$ be some invertible transformation of θ. We want to choose a prior that is **invariant** to this function f, so that the posterior does not depend on how we parameterize the model.

For example, consider a Bernoulli distribution with rate parameter θ. Suppose Alice uses a binomial likelihood with data $\mathcal{D}$, and computes $p(\theta|\mathcal{D})$. Now suppose Bob uses the same likelihood and data, but parameterizes the model in terms of the odds parameter, $\phi = \frac{\theta}{1-\theta}$. He converts Alice's prior to $p(\phi)$ using the change of variables formula, and them computes $p(\phi|\mathcal{D})$. If he then converts back to the θ parameterization, he should get the same result as Alice.

We can achieve this goal that provided we use a **Jeffreys prior**, named after Harold Jeffreys.[7] In 1d, the Jeffreys prior is given by $p(\theta) \propto \sqrt{F(\theta)}$, where F is the Fisher information (Section 2.4). In multiple dimensions, the Jeffreys prior has the form $p(\boldsymbol{\theta}) \propto \sqrt{\det \mathbf{F}(\boldsymbol{\theta})}$, where $\mathbf{F}$ is the Fisher information matrix (Section 2.4).

To see why the Jeffreys prior is invariant to parameterization, consider the 1d case. Suppose

7. Harold Jeffreys, 1891 – 1989, was an English mathematician, statistician, geophysicist, and astronomer. He is not to be confused with Richard Jeffrey, a philosopher who advocated the subjective interpretation of probability [Jef04].

$p_\theta(\theta) \propto \sqrt{F(\theta)}$. Using the change of variables, we can derive the corresponding prior for ϕ as follows:

$$p_\phi(\phi) = p_\theta(\theta) \left| \frac{d\theta}{d\phi} \right| \tag{3.148}$$

$$\propto \sqrt{F(\theta) \left(\frac{d\theta}{d\phi} \right)^2} = \sqrt{\mathbb{E}\left[\left(\frac{d \log p(x|\theta)}{d\theta} \right)^2 \right] \left(\frac{d\theta}{d\phi} \right)^2} \tag{3.149}$$

$$= \sqrt{\mathbb{E}\left[\left(\frac{d \log p(x|\theta)}{d\theta} \frac{d\theta}{d\phi} \right)^2 \right]} = \sqrt{\mathbb{E}\left[\left(\frac{d \log p(x|\phi)}{d\phi} \right)^2 \right]} \tag{3.150}$$

$$= \sqrt{F(\phi)} \tag{3.151}$$

Thus the prior distribution is the same whether we use the θ parameterization or the ϕ parameterization.

We give some examples of Jeffreys priors below.

3.6.2.1 Jeffreys prior for binomial distribution

Let us derive the Jeffreys prior for the binomial distribution using the rate parameterization θ. From Equation (2.258), we have

$$p(\theta) \propto \theta^{-\frac{1}{2}} (1-\theta)^{-\frac{1}{2}} = \frac{1}{\sqrt{\theta(1-\theta)}} \propto \text{Beta}(\theta|\frac{1}{2}, \frac{1}{2}) \tag{3.152}$$

Now consider the odds parameterization, $\phi = \theta/(1-\theta)$, so $\theta = \frac{\phi}{\phi+1}$. The likelihood becomes

$$p(x|\phi) \propto \left(\frac{\phi}{\phi+1} \right)^x \left(1 - \frac{\phi}{\phi+1} \right)^{n-x} = \phi^x (\phi+1)^{-x} (\phi+1)^{-n+x} = \phi^x (\phi+1)^{-n} \tag{3.153}$$

Thus the log likelihood is

$$\ell = x \log \phi - n \log \phi + 1 \tag{3.154}$$

The first and second derivatives are

$$\frac{d\ell}{d\phi} = \frac{x}{\phi} - \frac{n}{\phi+1} \tag{3.155}$$

$$\frac{d^2\ell}{d\phi^2} = -\frac{x}{\phi^2} + \frac{n}{(\phi+1)^2} \tag{3.156}$$

Since $\mathbb{E}[x] = n\theta = n\frac{\phi}{\phi+1}$, the Fisher information matrix is given by

$$F(\phi) = -\mathbb{E}\left[\frac{d^2\ell}{d\phi^2} \right] \frac{n}{\phi(\phi+1)} - \frac{n}{(\phi+1)^2} \tag{3.157}$$

$$= \frac{n(\phi+1) - n\phi}{\phi(\phi+1)^2} = \frac{n}{\phi(\phi+1)^2} \tag{3.158}$$

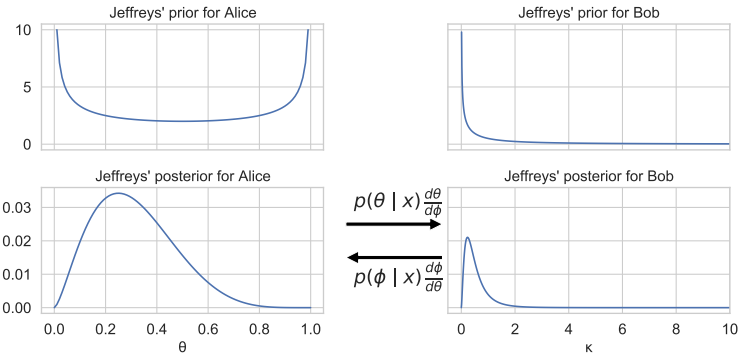


Figure 3.11: Illustration of Jeffrey's prior for Alice (who uses the rate θ) and Bob (who uses the odds $\phi = \theta/(1-\theta)$). Adapted from Figure 1.9 of [MKL11]. Generated by jeffreys_prior_binomial.ipynb.

Hence

$$p_\phi(\phi) \propto \phi^{-0.5}(1+\phi)^{-1} \tag{3.159}$$

See Figure 3.11 for an illustration.

3.6.2.2 Jeffreys prior for multinomial distribution

For a categorical random variable with K states, one can show that the Jeffreys prior is given by

$$p(\boldsymbol{\theta}) \propto \text{Dir}(\boldsymbol{\theta}|\frac{1}{2},\ldots,\frac{1}{2}) \tag{3.160}$$

Note that this is different from the more obvious choices of $\text{Dir}(\frac{1}{K},\ldots,\frac{1}{K})$ or $\text{Dir}(1,\ldots,1)$.

3.6.2.3 Jeffreys prior for the mean and variance of a univariate Gaussian

Consider a 1d Gaussian $x \sim \mathcal{N}(\mu,\sigma^2)$ with both parameters unknown, so $\boldsymbol{\theta} = (\mu,\sigma)$. From Equation (2.263), the Fisher information matrix is

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{pmatrix} 1/\sigma^2 & 0 \\ 0 & 2/\sigma^2 \end{pmatrix} \tag{3.161}$$

so $\sqrt{\det(\mathbf{F}(\boldsymbol{\theta}))} = \sqrt{\frac{2}{\sigma^4}}$. However, the standard Jeffreys uninformative prior for the Gaussian is defined as the product of independent uninformative priors (see [KW96]), i.e.,

$$p(\mu,\sigma^2) \propto p(\mu)p(\sigma^2) \propto 1/\sigma^2 \tag{3.162}$$

It turns out that we can emulate this prior with a conjugate NIX prior:

$$p(\mu,\sigma^2) = NI\chi^2(\mu,\sigma^2|\mu_0=0,\breve{\kappa}=0,\breve{\nu}=-1,\breve{\sigma}^2=0) \tag{3.163}$$

This lets us easily reuse the results for conjugate analysis of the Gaussian in Section 3.2.3.3, as we showed in Section 3.2.3.4.

3.6.3 Invariant priors

If we have "objective" prior knowledge about a problem in the form of invariances, we may be able to encode this into a prior, as we show below.

3.6.3.1 Translation-invariant priors

A **location-scale family** is a family of probability distributions parameterized by a location μ and scale σ. If x is an rv in this family, then $y = a + bx$ is also an rv in the same family.

When inferring the location parameter μ, it is intuitively reasonable to want to use a **translation-invariant prior**, which satisfies the property that the probability mass assigned to any interval, $[A, B]$ is the same as that assigned to any other shifted interval of the same width, such as $[A-c, B-c]$. That is,

$$\int_{A-c}^{B-c} p(\mu)d\mu = \int_{A}^{B} p(\mu)d\mu \tag{3.164}$$

This can be achieved using

$$p(\mu) \propto 1 \tag{3.165}$$

since

$$\int_{A-c}^{B-c} 1d\mu = (B-c) - (A-c) = (B-A) = \int_{A}^{B} 1d\mu \tag{3.166}$$

This is the same as the Jeffreys prior for a Gaussian with unknown mean μ and fixed variance. This follows since $F(\mu) = 1/\sigma^2 \propto 1$, from Equation (2.263), and hence $p(\mu) \propto 1$.

3.6.3.2 Scale-invariant prior

When inferring the scale parameter σ, we may want to use a **scale-invariant prior**, which satisfies the property that the probability mass assigned to any interval $[A, B]$ is the same as that assigned to any other interval $[A/c, B/c]$, where $c > 0$. That is,

$$\int_{A/c}^{B/c} p(\sigma)d\sigma = \int_{A}^{B} p(\sigma)d\sigma \tag{3.167}$$

This can be achieved by using

$$p(\sigma) \propto 1/\sigma \tag{3.168}$$

since then

$$\int_{A/c}^{B/c} \frac{1}{\sigma}d\sigma = [\log\sigma]_{A/c}^{B/c} = \log(B/c) - \log(A/c) = \log(B) - \log(A) = \int_{A}^{B} \frac{1}{\sigma}d\sigma \tag{3.169}$$

This is the same as the Jeffreys prior for a Gaussian with fixed mean μ and unknown scale σ. This follows since $F(\sigma) = 2/\sigma^2$, from Equation (2.263), and hence $p(\sigma) \propto 1/\sigma$.

3.6.3.3 Learning invariant priors

Whenever we have knowledge of some kind of invariance we want our model to satisfy, we can use this to encode a corresponding prior. Sometimes this is done analytically (see e.g., [Rob07, Ch.9]). When this is intractable, it may be be possible to learn invariant priors by solving a variational optimization problem (see e.g., [NS18]).

3.6.4 Reference priors

One way to define a noninformative prior is as a distribution which is maximally far from all possible posteriors, when averaged over datasets. This is the basic idea behind a **reference prior** [Ber05; BBS09]. More precisely, we say that $p(\boldsymbol{\theta})$ is a reference prior if it maximizes the expected KL divergence between posterior and prior:

$$p^*(\boldsymbol{\theta}) = \underset{p(\boldsymbol{\theta})}{\operatorname{argmax}} \int_{\mathcal{D}} p(\mathcal{D}) D_{\mathbb{KL}}\left(p(\boldsymbol{\theta}|\mathcal{D}) \parallel p(\boldsymbol{\theta})\right) d\mathcal{D} \tag{3.170}$$

where $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$. This is the same as maximizing the mutual information $\mathbb{I}(\boldsymbol{\theta}, \mathcal{D})$.

We can eliminate the integral over datasets by noting that

$$\int p(\mathcal{D}) \int p(\boldsymbol{\theta}|\mathcal{D}) \log \frac{p(\boldsymbol{\theta}|\mathcal{D})}{p(\boldsymbol{\theta})} = \int p(\boldsymbol{\theta}) \int p(\mathcal{D}|\boldsymbol{\theta}) \log \frac{p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})} = \mathbb{E}_{\boldsymbol{\theta}}\left[D_{\mathbb{KL}}\left(p(\mathcal{D}|\boldsymbol{\theta}) \parallel p(\mathcal{D})\right)\right] \tag{3.171}$$

where we used the fact that $\frac{p(\boldsymbol{\theta}|\mathcal{D})}{p(\boldsymbol{\theta})} = \frac{p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})}$.

One can show that, in 1d, the corresponding prior is equivalent to the Jeffreys prior. In higher dimensions, we can compute the reference prior for one parameter at a time, using the chain rule. However, this can become computationally intractable. See [NS17] for a tractable approximation based on variational inference (Section 10.1).

3.7 Hierarchical priors

Bayesian models require specifying a prior $p(\boldsymbol{\theta})$ for the parameters. The parameters of the prior are called **hyperparameters**, and will be denoted by $\boldsymbol{\phi}$. If these are unknown, we can put a prior on them; this defines a **hierarchical Bayesian model**, or **multi-level model**, which can visualize like this: $\boldsymbol{\phi} \rightarrow \boldsymbol{\theta} \rightarrow \mathcal{D}$. We assume the prior on the hyper-parameters is fixed (e.g., we may use some kind of minimally informative prior), so the joint distribution has the form

$$p(\boldsymbol{\phi}, \boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\phi}) p(\boldsymbol{\theta}|\boldsymbol{\phi}) p(\mathcal{D}|\boldsymbol{\theta}) \tag{3.172}$$

The hope is that we can learn the hyperparameters by treating the parameters themselves as datapoints.

A common setting in which such an approach makes sense is when we have $J > 1$ related datasets, $\mathcal{D}_j$, each with their own parameters $\boldsymbol{\theta}_j$. Inferring $p(\boldsymbol{\theta}_j|\mathcal{D}_j)$ independently for each group j can give poor results if $\mathcal{D}_j$ is a small dataset (e.g., if condition j corresponds to a rare combination of features, or a sparsely population region). We could of course pool all the data to compute a single model, $p(\boldsymbol{\theta}|\mathcal{D})$, but that would not let us model the subpopulations. A hierarchical Bayesian model lets us

borrow statistical strength from groups with lots of data (and hence well-informed posteriors $p(\boldsymbol{\theta}_j|\mathcal{D})$) in order to help groups with little data (and hence highly uncertain posteriors $p(\boldsymbol{\theta}_j|\mathcal{D})$). The idea is that well-informed groups j will have a good estimate of $\boldsymbol{\theta}_j$, from which we can infer $\boldsymbol{\phi}$, which can be used to help estimate $\boldsymbol{\theta}_k$ for groups k with less data. (Information is shared via the hidden common parent node $\boldsymbol{\phi}$ in the graphical model, as shown in Figure 3.12.) We give some examples of this below.

After fitting such models, we can compute two kinds of posterior predictive distributions. If we want to predict observations for an existing group j, we need to use

$$p(y_j|\mathcal{D}) = \int p(y_j|\theta_j)p(\theta_j|\mathcal{D})d\theta_j \tag{3.173}$$

However, if we want to predict observations for a new group $*$ that has not yet been measured, but which is comparable to (or **exchangeable with**) the existing groups $1:J$, we need to use

$$p(y_*|\mathcal{D}) = \int p(y_*|\theta_*)p(\theta_*|\boldsymbol{\phi})p(\boldsymbol{\phi}|\mathcal{D})d\theta_* d\boldsymbol{\phi} \tag{3.174}$$

We give some examples below. (More information can be found in e.g., [GH07; Gel+14a].)

3.7.1 A hierarchical binomial model

Suppose we want to estimate the prevalence of some disease amongst different group of individuals, either people or animals. Let N_j be the size of the j'th group, and let y_j be the number of positive cases for group $j = 1:J$. We assume $y_j \sim \text{Bin}(N_j, \theta_j)$, and we want to estimate the rates θ_j. Since some groups may have small population sizes, we may get unreliable results if we estimate each θ_j separately; for example we may observe $y_j = 0$ resulting in $\hat{\theta}_j = 0$, even though the true infection rate is higher.

One solution is to assume all the θ_j are the same; this is called **parameter tying**. The resulting pooled MLE is just $\hat{\theta}_{\text{pooled}} = \frac{\sum_j y_j}{\sum_j N_j}$. But the assumption that all the groups have the same rate is a rather strong one. A compromise approach is to assume that the θ_j are similar, but that there may be group-specific variations. This can be modeled by assuming the θ_j are drawn from some common distribution, say $\theta_j \sim \text{Beta}(a, b)$. The full joint distribution can be written as

$$p(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\phi}) = p(\boldsymbol{\phi})p(\boldsymbol{\theta}|\boldsymbol{\phi})p(\mathcal{D}|\boldsymbol{\theta}) = p(\boldsymbol{\phi})\left[\prod_{j=1}^{J} \text{Beta}(\theta_j|\boldsymbol{\phi})\right]\left[\prod_{j=1}^{J} \text{Bin}(y_j|N_j, \theta_j)\right] \tag{3.175}$$

where $\boldsymbol{\phi} = (a, b)$. In Figure 3.12 we represent these assumptions using a directed graphical model (see Section 4.2.8 for an explanation of such diagrams).

It remains to specify the prior $p(\boldsymbol{\phi})$. Following [Gel+14a, p110], we use

$$p(a, b) \propto (a + b)^{-5/2} \tag{3.176}$$

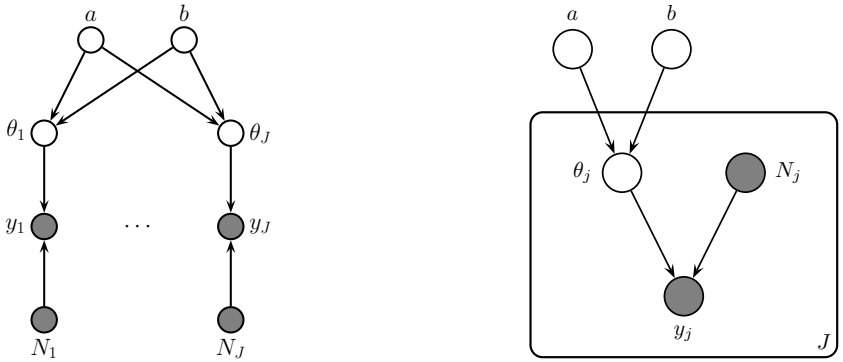


Figure 3.12: PGM for a hierarchical binomial model. (a) "Unrolled" model. (b) Same model, using plate notation.

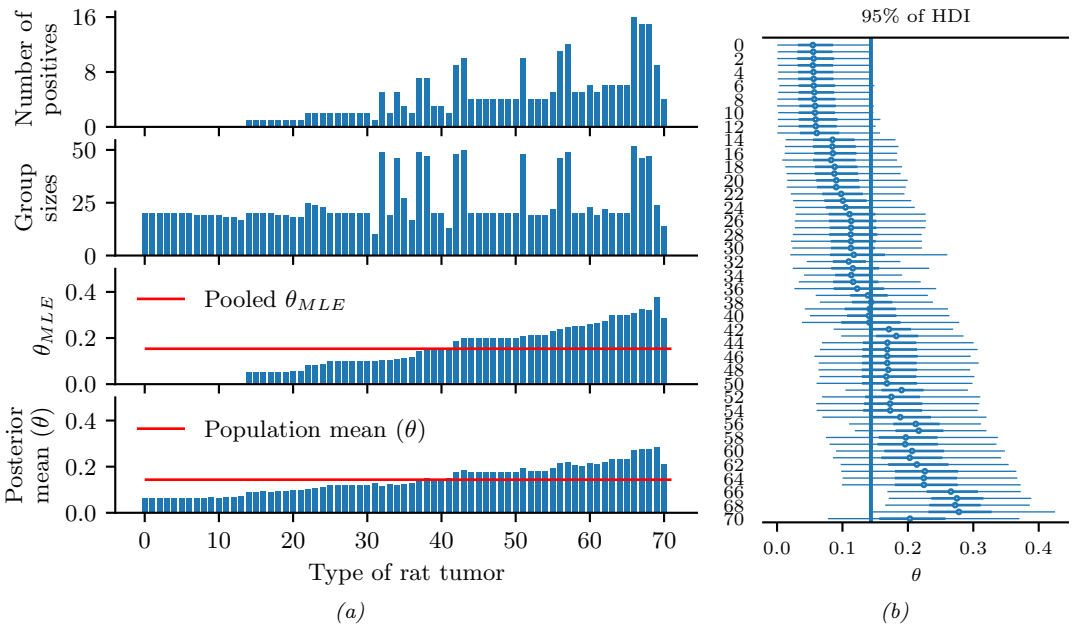


Figure 3.13: Data and inferences for the hierarchical binomial model fit using HMC. Generated by hierarchical_binom_rats.ipynb.

3.7.1.1 Posterior inference

We can perform approximate posterior inference in this model using a variety of methods. In Section 3.8.1 we discuss an optimization based approach, but here we discuss one of the most popular methods in Bayesian statistics, known as HMC or Hamiltonian Monte Calro. This is described in Section 12.5, but in short it is a form of MCMC (Markov Chain Monte Carlo) that exploits information from the gradient of the log joint to guide the sampling process. This algorithm generates samples in an unconstrained parameter space, so we need to define the log joint over all the parameters $\boldsymbol{\omega} = (\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}) \in \mathbb{R}^D$ as follows:

$$\log p(\mathcal{D}, \boldsymbol{\omega}) = \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|\boldsymbol{\phi}) + \log p(\boldsymbol{\phi}) \tag{3.177}$$

$$+ \sum_{j=1}^{J} \log |\mathrm{Jac}(\sigma)(\tilde{\theta}_j)| + \sum_{i=1}^{2} \log |\mathrm{Jac}(\sigma_+)(\tilde{\phi}_i)| \tag{3.178}$$

where $\theta_j = \sigma(\tilde{\theta}_j)$ is the sigmoid transform, and $\phi_i = \sigma_+(\tilde{\phi}_i)$ is the softplus transform. (We need to add the Jacobian terms to account for these deterministic transformations.) We can then use automatic differentation to compute $\nabla_{\boldsymbol{\omega}} \log p(\mathcal{D}, \boldsymbol{\omega})$, which we pass to the HMC algorithm. This algorithm returns a set of (correlated) samples from the posterior, $(\tilde{\boldsymbol{\phi}}^s, \tilde{\boldsymbol{\theta}}^s) \sim p(\boldsymbol{\omega}|\mathcal{D})$, which we can back transform to $(\boldsymbol{\phi}^s, \boldsymbol{\theta}^s)$. We can then estimate the posterior overquantities of interest by using a Monte Carlo approximation to $p(f(\boldsymbol{\theta})|\mathcal{D})$ for suitable f (e.g., to compute the posterior mean rate for group j, we set $f(\boldsymbol{\theta}) = \theta_j$).

3.7.1.2 Example: the rats dataset

In this section, we apply this model to analyse the number of rats that develop a certain kind of tumor during a particular clinical trial (see [Gel+14a, p102] for details). We show the raw data in rows 1–2 of Figure 3.13a(a),. In row 3 of Figure 3.13a(a) we show the MLE $\hat{\theta}_j$ for each group. We see that some groups have $\hat{\theta}_j = 0$, which is much less than the pooled MLE $\hat{\theta}_{\text{pooled}}$ (red line). In row 4 of Figure 3.13a(a) we show the posterior mean $\mathbb{E}[\theta_j|\mathcal{D}]$ estimated from all the data, as well as the population mean $\mathbb{E}[\theta|\mathcal{D}] = \mathbb{E}[a/(a+b)|\mathcal{D}]$ shown in the red line, We see that groups that had low counts have their estimates increased towards the population mean, and groups that have large counts have their estimates decreased towards the population mean. In other words, the groups regularize each other; this phenomenon is called **shrinkage**. The amount of shrinkage is controlled by the prior on (a, b), which is inferred from the data.

In Figure 3.13a(b), we show the 95% credible intervals for each parameter, as well as the overall population mean. (This is known as a **forest plot**.) We can use this to decide if any group is significantly different than any specified target value (e.g., the overall average).

3.7.2 A hierarchical Gaussian model

In this section, we consider a variation of the model in Section 3.7.1, where this time we have real-valued data instead of binary count data. More specificially we assume $y_{ij} \sim \mathcal{N}(\theta_j, \sigma^2)$, where θ_j is the unknown mean for group j, and σ^2 is the observation variance (assumed to be shared across groups and fixed, for simplicity). Note that having N_j observations y_{ij} each with variance σ^2 is like having one measurement $y_j \triangleq \frac{1}{N_j} \sum_{i=1}^{N_j} y_{ij}$ with variance $\sigma_j^2 \triangleq \sigma^2/N_j$. This lets us simplify notation

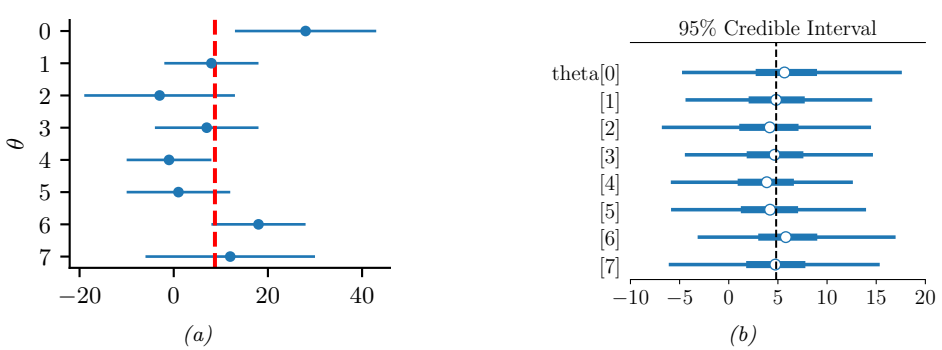


Figure 3.14: 8-schools dataset. (a) Raw data. Each row plots $y_j \pm \sigma_j$. Vertical line is the pooled estimate. (b) Posterior 95% credible intervals for θ_j. Vertical line is posterior mean $\mathbb{E}[\mu|\mathcal{D}]$. Generated by schools8.ipynb.

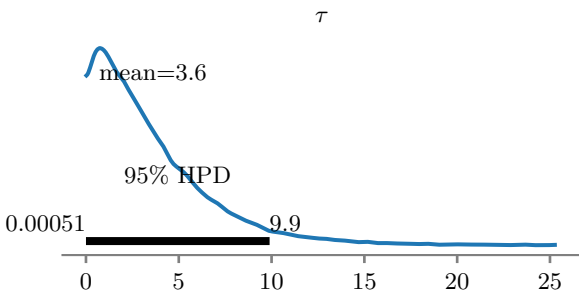


Figure 3.15: Marginal posterior density $p(\tau|\mathcal{D})$ for the 8-schools dataset. Generated by schools8.ipynb.

and use one observation per group, with likelihood $y_j \sim \mathcal{N}(\theta, \sigma_j^2)$, where we assume the σ_j's are known.

We will use a hierarchical model by assuming each group's parameters come from a common distribution, $\theta_j \sim \mathcal{N}(\mu, \tau^2)$. The model becomes

$$p(\mu, \tau^2, \boldsymbol{\theta}_{1:J}|\mathcal{D}) \propto p(\mu)p(\tau^2) \prod_{j=1}^{J} \mathcal{N}(\theta_j|\mu, \tau^2)\mathcal{N}(y_j|\theta_j, \sigma_j^2) \tag{3.179}$$

where $p(\mu)p(\tau^2)$ is some kind of prior over the hyper-parameters. See Figure 3.17a for the graphical model.

3.7.2.1 Example: the 8-schools dataset

Let us now apply this model to some data. We will consider the **eight schools** dataset from [Gel+14a, Sec 5.5]. The goal is to estimate the effects on a new coaching program on SAT scores. Let y_{nj} be the observed improvement in score for student n in school j compared to a baseline. Since each school has multiple students, we summarize its data using the empirical mean $\overline{y}_{\cdot j} = \frac{1}{N_j}\sum_{n=1}^{N_j} y_{nj}$ and standard deviation σ_j. See Figure 3.14a for an illustration of the data. We also show the pooled

MLE for θ, which is a precision weighted average of the data:

$$\overline{y}_{..} = \frac{\sum_{j=1}^{J} \frac{1}{\sigma_j^2} \overline{y}_{.j}}{\sum_{j=1}^{J} \frac{1}{\sigma_j^2}} \tag{3.180}$$

We see that school 0 has an unusually large improvement (28 points) compared to the overall mean, suggesting that the estimating θ_0 just based on $\mathcal{D}_0$ might be unreliable. However, we can easily apply our hierarchical model. We will use HMC to do approximate inference. (See Section 3.8.2 for a faster approximate method.)

After computing the (approximate) posterior, we can compute the marginal posteriors $p(\theta_j|\mathcal{D})$ for each school. These distributions are shown in Figure 3.14b. Once again, we see shrinkage towards the global mean $\overline{\mu} = \mathbb{E}[\mu|\mathcal{D}]$, which is close to the pooled estimate $\overline{y}_{..}$. In fact, if we fix the hyper-parameters to their posterior mean values, and use the approximation

$$p(\mu, \tau^2|\mathcal{D}) = \delta(\mu - \overline{\mu})\delta(\tau^2 - \overline{\tau}^2) \tag{3.181}$$

then we can use the results from Section 3.2.3.1 to compute the marginal posteriors

$$p(\theta_j|\mathcal{D}) \approx p(\theta_j|\mathcal{D}_j, \overline{\mu}, \overline{\tau}^2) \tag{3.182}$$

In particular, we can show that the posterior mean $\mathbb{E}[\theta_j|\mathcal{D}]$ is in between the MLE $\hat{\theta}_j = y_j$ and the global mean $\overline{\mu} = \mathbb{E}[\mu|\mathcal{D}]$:

$$\mathbb{E}\left[\theta_j|\mathcal{D}, \overline{\mu}, \overline{\tau}^2\right] = w_j\overline{\mu} + (1 - w_j)\hat{\theta}_j \tag{3.183}$$

where the amount of shrinkage towards the global mean is given by

$$w_j = \frac{\sigma_j^2}{\sigma_j^2 + \tau^2} \tag{3.184}$$

Thus we see that there is more shrinkage for groups with smaller measurement precision (e.g., due to smaller sample size), which makes intuitive sense. There is also more shrinkage if τ^2 is smaller; of course τ^2 is unknown, but we can compute a posterior for it, as shown in Figure 3.15.

3.7.2.2 Non-centered parameterization

It turns out that posterior inference in this model is difficult for many algorithms because of the tight dependence between the variance hyper parameter τ^2 and the group means θ_j, as illustrated by the **funnel shape** in Figure 3.16. In particular, consider making local move through parameter space. The algorithm can only "visit" the place where τ^2 is small (corresponding to strong shrinkage to the prior) if all the θ_j are close to the prior mean μ. It may be hard to move into the area where τ^2 is small unless all groups *simultaneously* move their θ_j estimates closer to μ.

A standard solution to this problem is to rewrite the model using the following **non-centered parameterization**:

$$\theta_j = \mu + \tau\eta_j \tag{3.185}$$

$$\eta_j \sim \mathcal{N}(0, 1) \tag{3.186}$$

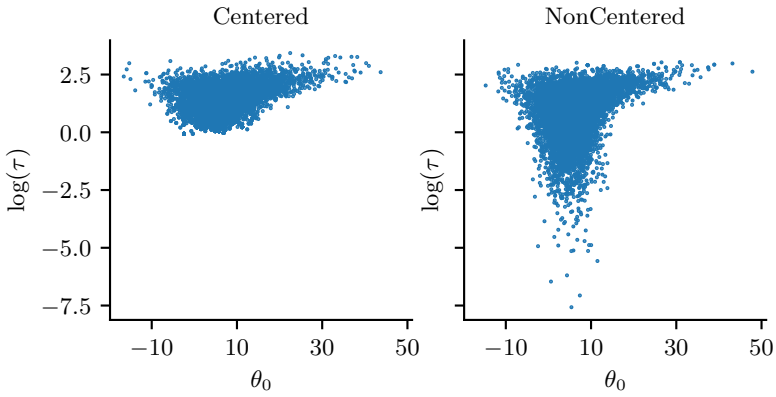


Figure 3.16: Posterior $p(\theta_0, \log(\tau)|\mathcal{D})$ for the 8 schools model using (a) centered parameterization and (b) non-centered parameterization. Generated by schools8.ipynb.

Figure 3.17: A hierarchical Gaussian Bayesian model. (a) Centered parameterization. (b) Non-centered parameterization.

See Figure 3.17b for the corresponding graphical model. By writing θ_j as a deterministic function of its parents plus a local noise term, we have reduced the dependence between θ_j and τ and hence the other θ_k variables, which can improve the computational efficiency of inference algorithms, as we discuss in Section 12.6.5. This kind of reparameterization is widely used in hierarchical Bayesian models.

3.7.3 Hierarchical conditional models

In Section 15.5, we discuss hierarchical Bayesian GLM models, which learn conditional distributions $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}_j)$ for each group j, using a prior of the form $p(\boldsymbol{\theta}_j|\boldsymbol{\phi})$. In Section 17.6, we discuss hierarchical Bayesian neural networks, which generalize this idea to nonlinear predictors.

3.8 Empirical Bayes

In Section 3.7, we discussed hierarchical Bayes as a way to infer parameters from data. Unfortunately, posterior inference in such models can be computationally challenging. In this section, we discuss a computationally convenient approximation, in which we first compute a point estimate of the hyperparameters, $\hat{\boldsymbol{\phi}}$, and then compute the conditional posterior, $p(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}, \mathcal{D})$, rather than the joint posterior, $p(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathcal{D})$.

To estimate the hyper-parameters, we can maximize the marginal likelihood:

$$\hat{\boldsymbol{\phi}}_{\text{mml}}(\mathcal{D}) = \operatorname*{argmax}_{\boldsymbol{\phi}} p(\mathcal{D}|\boldsymbol{\phi}) = \operatorname*{argmax}_{\boldsymbol{\phi}} \int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\phi}) d\boldsymbol{\theta} \tag{3.187}$$

This technique is known as **type II maximum likelihood**, since we are optimizing the hyper-parameters, rather than the parameters. Once we have estimated $\hat{\boldsymbol{\phi}}$, we compute the posterior $p(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}, \mathcal{D})$ in the usual way. This is easy to do, if the model is conjugate conditional on the hyper-parameters.

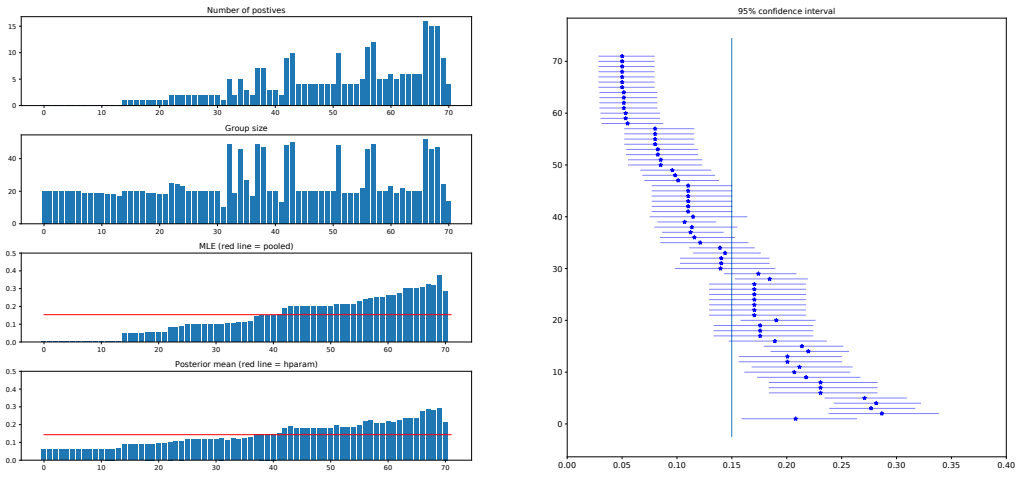
Since we are estimating the prior parameters from data, this approach is **empirical Bayes** (**EB**) [CL96]. This violates the principle that the prior should be chosen independently of the data. However, we can view it as a computationally cheap approximation to inference in the full hierarchical Bayesian model, just as we viewed MAP estimation as an approximation to inference in the one level model $\boldsymbol{\theta} \rightarrow \mathcal{D}$. In fact, we can construct a hierarchy in which the more integrals one performs, the "more Bayesian" one becomes, as shown below.

Method	Definition
Maximum likelihood	$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$
MAP estimation	$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\phi})$
ML-II (Empirical Bayes)	$\hat{\boldsymbol{\phi}} = \operatorname{argmax}_{\boldsymbol{\phi}} \int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\phi}) d\boldsymbol{\theta}$
MAP-II	$\hat{\boldsymbol{\phi}} = \operatorname{argmax}_{\boldsymbol{\phi}} \int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\phi}) p(\boldsymbol{\phi}) d\boldsymbol{\theta}$
Full Bayes	$p(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\phi}) p(\boldsymbol{\phi})$

Note that ML-II is less likely to overfit than "regular" maximum likelihood, because there are typically fewer hyper-parameters $\boldsymbol{\phi}$ than there are parameters $\boldsymbol{\theta}$. We give some simple examples below, and will see some ML applications later in the book.

3.8.1 EB for the hierarchical binomial model

In this section, we revisit the hierarchical binomial model from Section 3.7.1, but we use empirical Bayes instead of full Bayesian inference. We can analytically integrate out the θ_j's, and write down



(a) *(b)*

Figure 3.18: Data and inferences for the hierarchical binomial model fit using empirical Bayes. Generated by eb_binom.ipynb.

the marginal likelihood directly: The resulting expression is

$$p(\mathcal{D}|\boldsymbol{\phi}) = \prod_j \int \text{Bin}(y_j|N_j,\theta_j)\text{Beta}(\theta_j|a,b)d\theta_j \tag{3.188}$$

$$\propto \prod_j \frac{B(a+y_j, b+N_j-y_j)}{B(a,b)} \tag{3.189}$$

$$= \prod_j \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \frac{\Gamma(a+y_j)\Gamma(b+N_j-y_j)}{\Gamma(a+b+N_j)} \tag{3.190}$$

Various ways of maximizing this marginal likelihood wrt a and b are discussed in [Min00c].

Having estimated the hyper-parameters a and b, we can plug them in to compute the posterior $p(\theta_j|\hat{a},\hat{b},\mathcal{D})$ for each group, using conjugate analysis in the usual way. We show the results in Figure 3.18; they are very similar to the full Bayesian analysis shown in Figure 3.13, but the EB method is much faster.

3.8.2 EB for the hierarchical Gaussian model

In this section, we revisit the hierarchical Gaussian model from Section 3.7.2.1. However, we fit the model using empirical Bayes.

For simplicity, we will assume that $\sigma_j^2 = \sigma^2$ is the same for all groups. When the variances are equal, we can derive the EB estimate in closed form, as we now show. We have

$$p(y_j|\mu,\tau^2,\sigma^2) = \int \mathcal{N}(y_j|\theta_j,\sigma^2)\mathcal{N}(\theta_j|\mu,\tau^2)d\theta_j = \mathcal{N}(y_j|\mu,\tau^2+\sigma^2) \tag{3.191}$$

Hence the marginal likelihood is

$$p(\mathcal{D}|\mu,\tau^2,\sigma^2) = \prod_{j=1}^{J} \mathcal{N}(y_j|\mu,\tau^2+\sigma^2) \tag{3.192}$$

Thus we can estimate the hyper-parameters using the usual MLEs for a Gaussian. For μ, we have

$$\hat{\mu} = \frac{1}{J}\sum_{j=1}^{J} y_j = \overline{y} \tag{3.193}$$

which is the overall mean. For τ^2, we can use moment matching, which is equivalent to the MLE for a Gaussian. This means we equate the model variance to the empirical variance:

$$\hat{\tau}^2 + \sigma^2 = \frac{1}{J}\sum_{j=1}^{J}(y_j - \overline{y})^2 \triangleq v \tag{3.194}$$

so $\hat{\tau}^2 = v - \sigma^2$. Since we know τ^2 must be positive, it is common to use the following revised estimate:

$$\hat{\tau}^2 = \max\{0, v-\sigma^2\} = (v-\sigma^2)_+ \tag{3.195}$$

Given this, the posterior mean becomes

$$\hat{\theta}_j = \lambda\mu + (1-\lambda)y_j = \mu + (1-\lambda)(y_j - \mu) \tag{3.196}$$

where $\lambda_j = \lambda = \sigma^2/(\sigma^2+\tau^2)$.

Unfortunately, we cannot use the above method on the 8-schools dataset in Section 3.7.2.1, since it uses unequal σ_j. However, we can still use the EM algorithm or other optimization based methods.

3.8.3 EB for Markov models (n-gram smoothing)

The main problem with add-one smoothing, discussed in Section 2.6.3.3, is that it assumes that all n-grams are equally likely, which is not very realistic. A more sophisticated approach, called **deleted interpolation** [CG96], defines the transition matrix as a convex combination of the bigram frequencies $f_{jk} = N_{jk}/N_j$ and the unigram frequencies $f_k = N_k/N$:

$$A_{jk} = (1-\lambda)f_{jk} + \lambda f_k = (1-\lambda)\frac{N_{jk}}{N_j} + \lambda\frac{N_k}{N} \tag{3.197}$$

The term λ is usually set by cross validation. There is also a closely related technique called **backoff smoothing**; the idea is that if f_{jk} is too small, we "back off" to a more reliable estimate, namely f_k.

We now show that this heuristic can be interpreted as an empirical Bayes approximation to a hierarchical Bayesian model for the parameter vectors corresponding to each row of the transition matrix $\mathbf{A}$. Our presentation follows [MP95].

First, let us use an independent Dirichlet prior on each row of the transition matrix:

$$\mathbf{A}_j \sim \text{Dir}(\alpha_0 m_1, \ldots, \alpha_0 m_K) = \text{Dir}(\alpha_0 \boldsymbol{m}) = \text{Dir}(\boldsymbol{\alpha}) \tag{3.198}$$

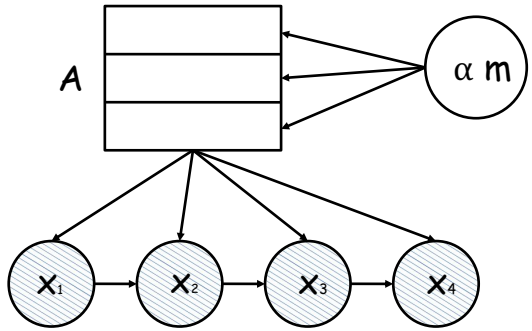


Figure 3.19: A Markov chain in which we put a different Dirichlet prior on every row of the transition matrix $\mathbf{A}$*, but the hyperparameters of the Dirichlet are shared.*

where $\mathbf{A}_j$ is row j of the transition matrix, $\boldsymbol{m}$ is the prior mean (satisfying $\sum_k m_k = 1$) and α_0 is the prior strength (see Figure 3.19). In terms of the earlier notation, we have $\boldsymbol{\theta}_j = \mathbf{A}_j$ and $\boldsymbol{\phi} = (\alpha, \boldsymbol{m})$.

The posterior is given by $\mathbf{A}_j \sim \text{Dir}(\boldsymbol{\alpha} + \mathbf{N}_j)$, where $\mathbf{N}_j = (N_{j1}, \ldots, N_{jK})$ is the vector that records the number of times we have transitioned out of state j to each of the other states. The posterior predictive density is

$$p(X_{t+1} = k | X_t = j, \mathcal{D}) = \frac{N_{jk} + \alpha_j m_k}{N_j + \alpha_0} = \frac{f_{jk} N_j + \alpha_j m_k}{N_j + \alpha_0} \tag{3.199}$$

$$= (1 - \lambda_j) f_{jk} + \lambda_j m_k \tag{3.200}$$

where

$$\lambda_j = \frac{\alpha_j}{N_j + \alpha_0} \tag{3.201}$$

This is very similar to Equation (3.197) but not identical. The main difference is that the Bayesian model uses a context-dependent weight λ_j to combine m_k with the empirical frequency f_{jk}, rather than a fixed weight λ. This is like *adaptive* deleted interpolation. Furthermore, rather than backing off to the empirical marginal frequencies f_k, we back off to the model parameter m_k.

The only remaining question is: what values should we use for α and $\boldsymbol{m}$? Let's use empirical Bayes. Since we assume each row of the transition matrix is a priori independent given $\boldsymbol{\alpha}$, the marginal likelihood for our Markov model is given by

$$p(\mathcal{D}|\boldsymbol{\alpha}) = \prod_j \frac{B(\mathbf{N}_j + \boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \tag{3.202}$$

where $\mathbf{N}_j = (N_{j1}, \ldots, N_{jK})$ are the counts for leaving state j and $B(\boldsymbol{\alpha})$ is the generalized beta function.

We can fit this using the methods discussed in [Min00c]. However, we can also use the following approximation [MP95, p12]:

$$m_k \propto |\{j : N_{jk} > 0\}| \tag{3.203}$$

This says that the prior probability of word k is given by the number of different contexts in which it occurs, rather than the number of times it occurs. To justify the reasonableness of this result, MacKay and Peto [MP95] give the following example.

```
Imagine, you see, that the language, you see, has, you see, a
frequently occuring couplet 'you see', you see, in which the second
word of the couplet, see, follows the first word, you, with very high
probability, you see. Then the marginal statistics, you see, are going
to become hugely dominated, you see, by the words you and see, with
equal frequency, you see.
```

If we use the standard smoothing formula, Equation (3.197), then P(you|novel) and P(see|novel), for some novel context word not seen before, would turn out to be the same, since the marginal frequencies of 'you' and 'see' are the same (11 times each). However, this seems unreasonable. 'You' appears in many contexts, so P(you|novel) should be high, but 'see' only follows 'you', so P(see|novel) should be low. If we use the Bayesian formula Equation (3.200), we will get this effect for free, since we back off to m_k not f_k, and m_k will be large for 'you' and small for 'see' by Equation (3.203).

Although elegant, this Bayesian model does not beat the state-of-the-art language model, known as **interpolated Kneser-Ney** [KN95; CG98]. By using ideas from nonparametric Bayes, one can create a language model that outperforms such heuristics, as discussed in [Teh06; Woo+09]. However, one can get even better results using recurrent neural nets (Section 16.3.4); the key to their success is that they don't treat each symbol "atomically", but instead learn a distributed embedding representation, which encodes the assumption that some symbols are more similar to each other than others.

3.9 Model selection and evaluation

All models are wrong, but some are useful. — George Box [BD87, p424].[8]

In this section, we assume we have a set of different models $\mathcal{M}$, each of which may fit the data to different degrees, and each of which may make different assumptions. We discuss how to pick the best model from this set, or to identify that none of them may be adequate.

3.9.1 Bayesian model selection

The natural way to pick the best model is to pick the most probable model according to Bayes rule:

$$\hat{m} = \operatorname*{argmax}_{m \in \mathcal{M}} p(m|\mathcal{D}) \tag{3.204}$$

where

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)p(m)}{\sum_{m \in \mathcal{M}} p(\mathcal{D}|m)p(m)} \tag{3.205}$$

is the posterior over models. This is called **Bayesian model selection**. If the prior over models is uniform, $p(m) = 1/|\mathcal{M}|$, then the MAP model is given by

$$\hat{m} = \operatorname*{argmax}_{m \in \mathcal{M}} p(\mathcal{D}|m) \tag{3.206}$$

8. George Box is a retired statistics professor at the University of Wisconsin.

The quantity $p(\mathcal{D}|m)$ is given by

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)p(\boldsymbol{\theta}|m)d\boldsymbol{\theta} \tag{3.207}$$

This is known as the **marginal likelihood**, or the **evidence** for model m. (See Section 3.9.2 for details on how to compute this quantity.) If the model assigns high prior predictive density to the observed data, then we deem it a good model. If, however, the model has too much flexibility, then some prior settings will not match the data; this probability mass will be "wasted", lowering the expected likelihood. This implicit regularization effect is called the **Bayesian Occam's razor**.

Note that **Bayesian hypothesis testing** can be considered as a special case of Bayesian model selection when we just have two models, commonly called the **null hypothesis**, M_0, and the **alternative hypothesis**, M_1. Let us define the **Bayes factor** as the ratio of marginal likelihoods:

$$B_{1,0} \triangleq \frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_0)} = \frac{p(M_1|\mathcal{D})}{p(M_0|\mathcal{D})} / \frac{p(M_1)}{p(M_0)} \tag{3.208}$$

(This is like a **likelihood ratio**, except we integrate out the parameters, which allows us to compare models of different complexity.) If $B_{1,0} > 1$ then we prefer model 1, otherwise we prefer model 0. By choosing the appropriate threshold on the Bayes factor, we can achieve any desired false positive vs false negative rate.

3.9.2 Estimating the marginal likelihood

If we use a conjugate prior, we can compute the marginal likelihood analytically, as we discussed in Section 3.2. However, in general, we must use numerical methods to approximate the integral in Equation (3.207).

A particularly simple estimator, is known as the **harmonic mean estimator**, and was proposed in [NR94]. It is defined as follows:

$$p(\mathcal{D}) \approx \left(\frac{1}{S} \sum_{s=1}^{S} \frac{1}{p(\mathcal{D}|\boldsymbol{\theta}_s)} \right)^{-1} \tag{3.209}$$

This follows from the following identity:

$$\mathbb{E}\left[\frac{1}{p(\mathcal{D}|\boldsymbol{\theta})}\right] = \int \frac{1}{p(\mathcal{D}|\boldsymbol{\theta})} p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} \tag{3.210}$$

$$= \int \frac{1}{p(\mathcal{D}|\boldsymbol{\theta})} \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} d\boldsymbol{\theta} \tag{3.211}$$

$$= \frac{1}{p(\mathcal{D})} \int p(\boldsymbol{\theta})d\boldsymbol{\theta} = \frac{1}{p(\mathcal{D})} \tag{3.212}$$

(We have assumed the prior is proper, so it integrates to 1.)

Unfortunately, the number of samples needed to get a good estimate is generally very large, making this approach useless in practice (see Radford Neal's blog post "The Harmonic Mean of the Likelihood:

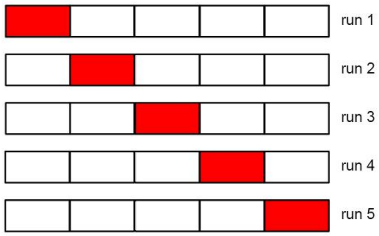


Figure 3.20: Schematic of 5-fold cross validation.

Worst Monte Carlo Method Ever"[9]). Various better estimators are available (see e.g., the review in [GM98; FW12]). In particular, in Chapter 13, we will see how sequential Monte Carlo can be used to approximate the evidence.

However, even though there are ways to approximate the marginal likelihood, it has a more fundamental (non-computational) problem, which is that it is very sensitive to the choice of prior. Since priors over parameters are often rather vague and arbitrary, this is rather undesirable. In addition, Bayesian model selection, as we have presented it so far, focuses on trying to pick the best model given the training set (albeit using the prior as a regularizer). We often get better results by selecting models based on predictive accuracy on a validation set. We discuss this topic, and its relationship to Bayes, in the sections below.

3.9.3 Connection between cross validation and marginal likelihood

A standard approach to model evaluation is to estimate its predictive performance (in terms of log likelihood) on a **validation set**, which is distinct from the training set which is used to fit the model If we don't have such a separate validation set, we can make one by partitioning the training set into K subsets or "**folds**", and then training on $K-1$ and testing on the K'th; we repeat this K times, as shown in Figure 3.20. This is known as **cross validation**.

If we set $K = N$, the method is known as **leave-one-out cross validation** or **LOO-CV**, since we train on $N-1$ points and test on the remaining one, and we do this N times. More precisely, we have

$$J_{\text{LOO}}(m) \triangleq \sum_{n=1}^{N} \log p(\boldsymbol{y}_n | \boldsymbol{x}_n, \hat{\boldsymbol{\theta}}(\mathcal{D}_{-n}), m) \tag{3.213}$$

where $\hat{\boldsymbol{\theta}}_{-n}$ is the parameter estimate computing when we omit $(\boldsymbol{x}_n, \boldsymbol{y}_n)$ from the training set. (We discuss fast approxmations to this in Section 3.9.4.)

Interestingly, the LOO-CV version of log likelihood is closely related to the log marginal likelihood.

9. https://bit.ly/3t7id0k.

To see this, let us write the log marginal likelihood in sequential form as follows:

$$\log p(\mathcal{D}|m) = \log \prod_{n=1}^{N} p(\mathcal{D}_n|\mathcal{D}_{1:n-1}, m) = \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \mathcal{D}_{1:n-1}, m) \tag{3.214}$$

where

$$p(\boldsymbol{y}_n|\boldsymbol{x}_n, \mathcal{D}_{1:n-1}, m) = \int p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_{1:n-1}, m) d\boldsymbol{\theta} \tag{3.215}$$

Note that we evaluate the posterior on the first $n-1$ data points and use this to predict the n'th; this is called **prequential analysis** [DV99].

Suppose we use a point estimate for the parameters at time n, rather than the full posterior. We can then use a plugin approximation to the n'th predictive distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}_n, \mathcal{D}_{1:n-1}, m) \approx \int p(\boldsymbol{y}|\boldsymbol{x}_n, \boldsymbol{\theta}) \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_m(\mathcal{D}_{1:n-1})) d\boldsymbol{\theta} = p(\boldsymbol{y}|\boldsymbol{x}_n, \hat{\boldsymbol{\theta}}_m(\mathcal{D}_{1:n-1})) \tag{3.216}$$

Then Equation (3.214) simplifies to

$$\log p(\mathcal{D}|m) \approx \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \hat{\boldsymbol{\theta}}(\mathcal{D}_{1:n-1}), m) \tag{3.217}$$

This is very similar to Equation (3.213), except it is evaluated sequentially. A complex model will overfit the "early" examples and will then predict the remaining ones poorly, and thus will get low marginal likelihood as well as low cross-validation score. See [FH20] for further discussion.

3.9.4 Bayesian leave-one-out (LOO) estimate

In this section we discuss a computationally efficient method, based on importance sampling, to approximate the leave-one-out (LOO) estimate without having to fit the model N times.

Suppose we have computed the posterior given the full dataset for model m. We can use this to evaluate the resulting predictive distribution $p(\boldsymbol{y}|\mathcal{D}, m)$ on each datapoint in the dataset. This gives the **log-pointwise predictive-density** or **LPPD** score:

$$\text{LPPD} \triangleq \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \mathcal{D}, m) = \sum_{n=1}^{N} \log \int p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}, m) p(\boldsymbol{\theta}|\mathcal{D}, m) d\boldsymbol{\theta} \tag{3.218}$$

We can approximate LPPD with Monte Carlo:

$$\text{LPPD}(m) \approx \sum_{n=1}^{N} \log \left(\frac{1}{S} \sum_{s=1}^{S} p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}_s, m) \right) \tag{3.219}$$

where $\boldsymbol{\theta}_s \sim p(\boldsymbol{\theta}|\mathcal{D}, m)$ is a posterior sample.

The trouble with LPPD is that it predicts the n'th data point $\boldsymbol{y}_n$ using all the data, including $\boldsymbol{y}_n$. What we would like to compute is the **expected LPPD** (**ELPD**) on *future data*, $\boldsymbol{y}_*$:

$$\text{ELPD} \triangleq \mathbb{E}_{\boldsymbol{y}_*} \log p(\boldsymbol{y}_*|\boldsymbol{x}_*, \mathcal{D}, m) = \int p(\boldsymbol{y}_*|\boldsymbol{x}_*, \boldsymbol{\theta}, m) \log \int p(\boldsymbol{y}_*|\boldsymbol{x}_*, \boldsymbol{\theta}, m) p(\boldsymbol{\theta}|\mathcal{D}, m) d\boldsymbol{\theta} \tag{3.220}$$

Of course, the future data is unknown, but we can use a LOO approximation:

$$\text{ELPD}_{\text{LOO}} \triangleq \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \mathcal{D}_{-n}, m) = \sum_{n=1}^{N} \log \int p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}, m) p(\boldsymbol{\theta}|\mathcal{D}_{-n}, m) d\boldsymbol{\theta} \tag{3.221}$$

This is a Bayesian version of Equation (3.213). We can approximate this integral using Monte Carlo:

$$\text{ELPD}_{\text{LOO}} \approx \sum_{n=1}^{N} \log \left(\frac{1}{S} \sum_{s=1}^{S} p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}_{s,-n}, m) \right) \tag{3.222}$$

where $\boldsymbol{\theta}_{s,-n} \sim p(\boldsymbol{\theta}|\mathcal{D}_{-n}, m)$.

The above procedure requires computing N different posteriors, leaving one data point out at a time, which is slow. A faster alternative is to compute $p(\boldsymbol{\theta}|\mathcal{D}, m)$ once, and then use importance sampling (Section 11.5) to approximate the above integral. More precisely, let $f(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D}_{-n}, m)$ be the target distribution of interest, and let $g(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D}, m)$ be the proposal. Define the importance weight for each sample s when leaving out example n to be

$$w_{s,-n} = \frac{f(\boldsymbol{\theta}_s)}{g(\boldsymbol{\theta}_s)} = \frac{p(\boldsymbol{\theta}_s|\mathcal{D}_{-n})}{p(\boldsymbol{\theta}_s|\mathcal{D})} = \frac{p(\mathcal{D}_{-n}|\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)}{p(\mathcal{D}_{-n})} \frac{p(\mathcal{D})}{p(\mathcal{D}|\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)} \tag{3.223}$$

$$\propto \frac{p(\mathcal{D}_{-n}|\boldsymbol{\theta}_s)}{p(\mathcal{D}|\boldsymbol{\theta}_s)} = \frac{p(\mathcal{D}_{-n}|\boldsymbol{\theta}_s)}{p(\mathcal{D}_{-n}|\boldsymbol{\theta})p(\mathcal{D}_n|\boldsymbol{\theta}_s)} = \frac{1}{p(\mathcal{D}_n|\boldsymbol{\theta}_s)} \tag{3.224}$$

We then normalize the weights to get

$$\hat{w}_{s,-n} = \frac{w_{s,-n}}{\sum_{s'=1}^{S} w_{s',-n}} \tag{3.225}$$

and use them to get the estimate

$$\text{ELPD}_{\text{IS-LOO}} = \sum_{n=1}^{N} \log \left(\sum_{s=1}^{S} \hat{w}_{s,-n} p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}_s, m) \right) \tag{3.226}$$

Unfortunately, the importance weights may have high variance, where some weights are much larger than others. To reduce this effect, we fit a Pareto distribution (Section 2.2.3.5) to each set of weights for each sample, and use this to smooth the weights. This technique is called **Pareto smoothed importance sampling** or **PSIS** [VGG17]. The Pareto distribution has the form

$$p(r|u, \sigma, k) = \sigma^{-1}(1 + k(r-u)\sigma^{-1})^{-1/k-1} \tag{3.227}$$

where u is the location, σ is the scale, and k is the shape. The parameter values k_n (for each data point n) can be used to assess how well this approximation works. If we find $k_n > 0.5$ for any given point, it is likely an outlier, and the resulting LOO estimate is likely to be quite poor.

3.9.5 Information criteria

An alternative approach to cross validation is to score models using the negative log likelihood (or LPPD) on the training set plus a **complexity penalty** term:

$$\mathcal{L}(m) = -\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) + C(m) \tag{3.228}$$

This is called an **information criterion**. Different methods use different complexity terms $C(m)$, as we discuss below. See e.g., [GHV14] for further details.

A note on notation: it is conventional, when working with information criteria, to scale the NLL by -2 to get the **deviance**:

$$\text{deviance}(m) = -2 \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) \tag{3.229}$$

This makes the math "prettier" for certain Gaussian models.

3.9.5.1 Minimum description length (MDL)

We can think about the problem of scoring different models in terms of information theory (Chapter 5). The goal is for the sender to communicate the data to the receiver. First the sender needs to specify which model m to use; this takes $C(m) = -\log p(m)$ bits (see Section 5.2). Then the receiver can fit the model, by computing $\hat{\boldsymbol{\theta}}_m$, and can thus approximately reconstruct the data. To perfectly reconstruct the data, the sender needs to send the residual errors that cannot be explained by the model; this takes

$$-L(m) = -\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) = -\sum_n \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \hat{\boldsymbol{\theta}}, m) \tag{3.230}$$

bits. (We are ignoring the cost of sending the input features $\boldsymbol{x}_n$, if present.) The total cost is

$$\mathcal{L}_{\text{MDL}}(m) = -\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) + C(m) \tag{3.231}$$

Choosing the model which minimizes this cost is known as the **minimum description length** or **MDL** principle. See e.g., [HY01] for details.

3.9.5.2 The Bayesian information criterion (BIC)

The **Bayesian information criterion** or **BIC** [Sch78] is similar to the MDL, and has the form

$$\mathcal{L}_{\text{BIC}}(m) = -2 \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) + D_m \log N \tag{3.232}$$

where D_m is the **degrees of freedom** of model m.

We can derive the BIC score as a simple approximation to the log marginal likelihood. In particular, suppose we make a Gaussian approximation to the posterior, as discussed in Section 7.4.3. Then we get (from Equation (7.22)) the following:

$$\log p(\mathcal{D}|m) \approx \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{\text{map}}) + \log p(\hat{\boldsymbol{\theta}}_{\text{map}}) - \frac{1}{2} \log |\mathbf{H}| \tag{3.233}$$

where $\mathbf{H}$ is the Hessian of the negative log joint $\log p(\mathcal{D}, \boldsymbol{\theta})$ evaluated at the MAP estimate $\hat{\boldsymbol{\theta}}_{\text{map}}$. We see that Equation (3.233) is the log likelihood plus some penalty terms. If we have a uniform prior, $p(\boldsymbol{\theta}) \propto 1$, we can drop the prior term, and replace the MAP estimate with the MLE, $\hat{\boldsymbol{\theta}}$, yielding

$$\log p(\mathcal{D}|m) \approx \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}) - \frac{1}{2} \log |\mathbf{H}| \tag{3.234}$$

We now focus on approximating the $\log|\mathbf{H}|$ term, which is sometimes called the **Occam factor**, since it is a measure of model complexity (volume of the posterior distribution). We have $\mathbf{H} = \sum_{i=1}^{N} \mathbf{H}_i$, where $\mathbf{H}_i = \nabla\nabla \log p(\mathcal{D}_i|\boldsymbol{\theta})$. Let us approximate each $\mathbf{H}_i$ by a fixed matrix $\hat{\mathbf{H}}$. Then we have

$$\log|\mathbf{H}| = \log|N\hat{\mathbf{H}}| = \log(N^D|\hat{\mathbf{H}}|) = D\log N + \log|\hat{\mathbf{H}}| \tag{3.235}$$

where $D = \dim(\boldsymbol{\theta})$ and we have assumed $\mathbf{H}$ is full rank. We can drop the $\log|\hat{\mathbf{H}}|$ term, since it is independent of N, and thus will get overwhelmed by the likelihood. Putting all the pieces together, we get the **BIC score** that we want to maximize:

$$J_{\text{BIC}}(m) = \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) - \frac{D_m}{2}\log N \tag{3.236}$$

We can also define the **BIC loss**, that we want to minimize, by multiplying by -2:

$$\mathcal{L}_{\text{BIC}}(m) = -2\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) + D_m \log N \tag{3.237}$$

3.9.5.3 Akaike information criterion

The **Akaike information criterion** [Aka74] is closely related to BIC. It has the form

$$\mathcal{L}_{\text{AIC}}(m) = -2\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}, m) + 2D_m \tag{3.238}$$

This penalizes complex models less heavily than BIC, since the regularization term is independent of N. This estimator can be derived from a frequentist perspective.

3.9.5.4 Widely applicable information criterion (WAIC)

The main problem with MDL, BIC, and AIC is that it can be hard to compute the degrees of a freedom of a model, needed to define the complexity term, since most parameters are highly correlated and not uniquely identifiable from the likelihood. In particular, if the mapping from parameters to the likelihood is not one-to-one, then the model known as a **singular statistical model**, since the corresponding Fisher information matrix (Section 2.4), and hence the Hessian $\mathbf{H}$ above, may be singular (have determinant 0). An alternative criterion that works even in the singular case is known as the **widely applicable information criterion** (WAIC), also known as the **Watanabe–Akaike information criterion** [Wat10; Wat13].

WAIC is like other information criteria, except it is more Bayesian. First it replaces the log likelihood $L(m)$, which uses a point estimate of the parameters, with the LPPD, which marginalizes them out. (see Equation (3.219)). For the complexity term, WAIC uses the variance of the predictive distribution:

$$C(m) = \sum_{n=1}^{N} \mathbb{V}_{\boldsymbol{\theta}|\mathcal{D},m}[\log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}, m)] \approx \sum_{n=1}^{N} \mathbb{V}\{\log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}_s, m) : s = 1:S\} \tag{3.239}$$

The intuition for this is as follows: if, for a given datapoint n, the different posterior samples $\boldsymbol{\theta}_s$ make very different predictions, then the model is uncertain, and likely too flexible. The complexity term essentially counts how often this occurs. The final WAIC loss is

$$\mathcal{L}_{\text{WAIC}}(m) = -2\text{LPPD}(m) + 2C(m) \tag{3.240}$$

Interestingly, it can be shown that the PSIS LOO estimate in Section 3.9.4 is asymptotically equivalent to WAIC [VGG17].

3.9.6 Posterior predictive checks

Bayesian inference and decision making is optimal, but only if the modeling assumptions are correct. In this section, we discuss some ways to assess if a model is reasonable.

From a Bayesian perspective, this can seem a bit odd, since if we knew there was a better model, why don't we just use that? Here we assume that we do not have a specific alternative model in mind (so we are not performing model selection, unlike Section 3.9.1) Instead we are just trying to see if the data we observe is "typical" of what we might expect if our model were correct. This is called **model checking**.

In particular, suppose we knew the true parameters θ, which we use to generate S synthetic datasets, $\tilde{\mathcal{D}}^s = \{\boldsymbol{y}_n^s \sim p(\cdot|\boldsymbol{\theta}) : n = 1 : N\}$; these represent "plausible hallucinations" of the model. To assess the quality of our model, we can compute how "typical" our observed data $\mathcal{D}$ is compared to the model's hallucinations. To perform this comparison, we create one or more scalar **test statistics**, $\text{test}(\tilde{\mathcal{D}}^s)$, and compare them to the test statistics on the actual data, $\text{test}(\mathcal{D})$. These statistics should measure features of interest (since it will not, in general, be possible to capture every aspect of the data with a given model). If there is a large difference between the distribution of $\text{test}(\tilde{\mathcal{D}}^s)$ across different s and the value of $\text{test}(\mathcal{D})$, it suggests the model is not a good one. This approach called a **posterior predictive check** [Rub84].

3.9.6.1 Example: 1d Gaussian

To make things clearer, let us consider an example from [Gel+04]. In 1882, Newcomb measured the speed of light using a certain method and obtained $N = 66$ measurements, shown in Figure 3.21(a). There are clearly two outliers in the left tails, suggesting that the distribution is not Gaussian. Let us nonetheless fit a Gaussian to it. For simplicity, we will just compute the MLE, and use a plug-in approximation to the posterior predictive density:

$$p(\tilde{y}|\mathcal{D}) \approx \mathcal{N}(\tilde{y}|\hat{\mu}, \hat{\sigma}^2), \quad \hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} y_n, \quad \hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{\mu})^2 \tag{3.241}$$

Let $\tilde{\mathcal{D}}^s$ be the s'th dataset of size $N = 66$ sampled from this distribution, for $s = 1 : 1000$. The histogram of $\tilde{\mathcal{D}}^s$ for some of these samples is shown in Figure 3.21(b). It is clear that none of the samples contain the large negative examples that were seen in the real data. This suggests the model cannot capture the long tails present in the data. (We are assuming that these extreme values are scientifically interesting, and something we want the model to capture.)

A more formal way to test fit is to define a test statistic. Since we are interested in small values, let us use

$$\text{test}(\mathcal{D}) = \min\{y : y \in \mathcal{D}\} \tag{3.242}$$

The empirical distribution of $\text{test}(\tilde{\mathcal{D}}^s)$ for $s = 1 : 1000$ is shown in Figure 3.21(c). For the real data, $\text{test}(\mathcal{D}) = -44$, but the test statistics of the generated data, $\text{test}(\tilde{\mathcal{D}})$, are much larger. Indeed, we see that -44 is in the left tail of the predictive distribution, $p(\text{test}(\tilde{\mathcal{D}})|\mathcal{D})$.

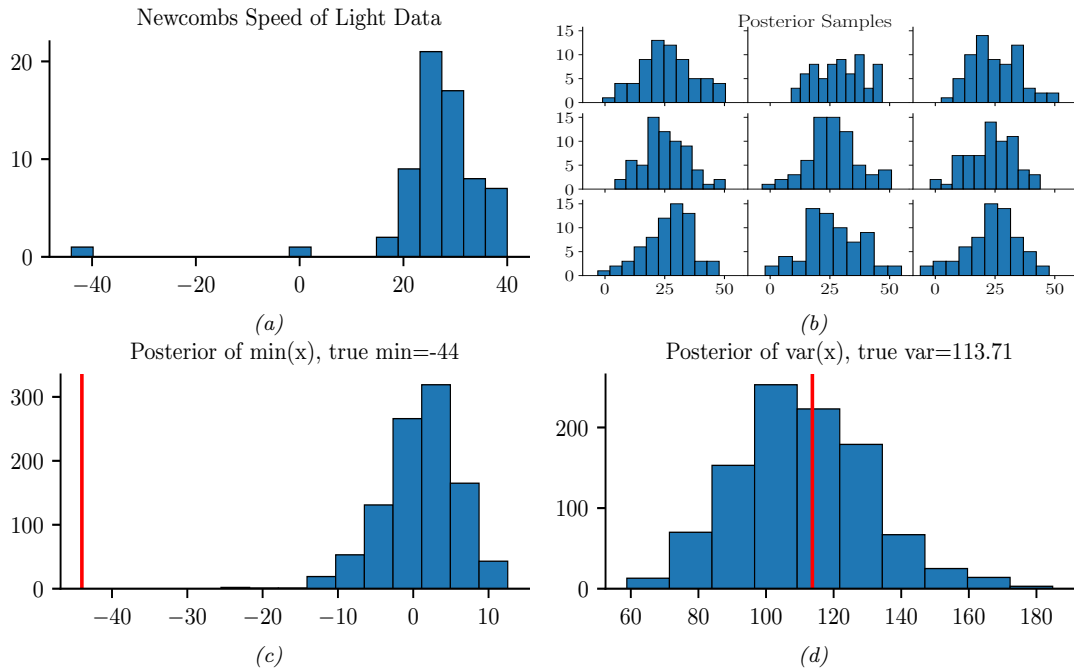


Figure 3.21: (a) Histogram of Newcomb's data. (b) Histograms of data sampled from Gaussian model. (c) Histogram of test statistic on data sampled from the model, which represents $p(test(\tilde{\mathcal{D}}^s)|\mathcal{D})$, where $test(\mathcal{D}) = \min\{y \in \mathcal{D}\}$. The vertical line is the test statistic on the true data, $test(\mathcal{D})$. (d) Same as (c) except $test(\mathcal{D}) = \mathbb{V}\{y \in \mathcal{D}\}$. Generated by newcomb_plugin_demo.ipynb.

3.9.7 Bayesian p-values

If some test statistic of the oberved data, $\text{test}(\mathcal{D})$, occurs in the left or right tail of the predictive distribution, then it is very unlikely under the model. We can quantify this using a **Bayesian p-value**, also called a **posterior-predictive p-value**:

$$p_B = P(\text{test}(\tilde{\mathcal{D}}) \geq \text{test}(\mathcal{D})|\mathcal{D}) \tag{3.243}$$

In contrast, a frequentist p-value is defined as

$$p_C = P(\text{test}(\tilde{\mathcal{D}}) \geq \text{test}(\mathcal{D})|\theta^*) \tag{3.244}$$

where θ^* is the true but unknown parameter. The key difference between the Bayesian and classical approach is that the Bayesian always conditions on what is known (namely the data $\mathcal{D}$), and never conditions on what is unknown (namely θ^*).

We can approximate the Bayesian p-value using Monte Carlo integration, as follows:

$$p_B = \int \mathbb{I}\left(\text{test}(\tilde{\mathcal{D}}) > \text{test}(\mathcal{D})\right) p(\tilde{\mathcal{D}}|\theta)p(\theta|\mathcal{D})d\theta \approx \frac{1}{S}\sum_{s=1}^{S} \mathbb{I}\left(\text{test}(\tilde{\mathcal{D}}^s) > \text{test}(\mathcal{D})\right) \tag{3.245}$$

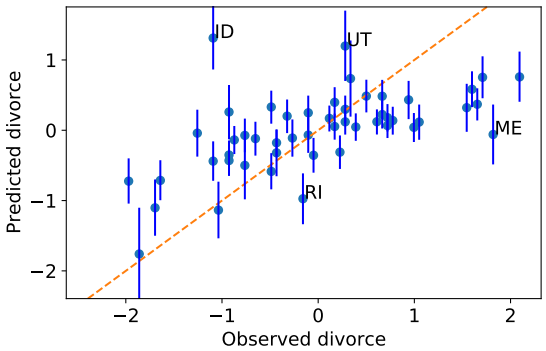


Figure 3.22: Posterior predictive distribution for divorce rate vs actual divorce rate for 50 US states. Both axes are standardized (i.e., z-scores). A few outliers are annotated. Adapted from Figure 5.5 of [McE20]. Generated by linreg_divorce_ppc.ipynb.

Any extreme value for p_B (i.e., a value near 0 or 1) means that the observed data is unlikely under the model, as assessed via test statistic test. However, if test($\mathcal{D}$) is a sufficient statistic of the model, it is likely to be well estimated, and the p-value will be near 0.5. For example, in the speed of light example, if we define our test statistic to be the variance of the data, test($\mathcal{D}$) = $\mathbb{V}\{y : y \in \mathcal{D}\}$, we get a p-value of 0.48. (See Figure 3.21(d).) This shows that the Gaussian model is capable of representing the variance in the data, even though it is not capable of representing the support (range) of the data.

The above example illustrates the very important point that we should not try to assess whether the data comes from a given model (for which the answer is nearly always that it does not), but rather, we should just try to assess whether the model captures the features we care about. See [Gel+04, ch.6] for a more extensive discussion of this topic.

3.9.7.1 Example: linear regression

When fitting conditional models, $p(\boldsymbol{y}|\boldsymbol{x})$, we will have a different prediction for each input $\boldsymbol{x}$. We can compare the predictive distribution $p(\boldsymbol{y}|\boldsymbol{x}_n)$ to the observed $\boldsymbol{y}_n$ to detect places where the model does poorly.

As an example of this, we consider the "waffle divorce" dataset from [McE20, Sec 5.1]. This contains the divorce rate D_n, marriage rate M_n and age A_n at first marriage for 50 different US states. We use a linear regression model to predict the divorce rate, $p(y = d|\boldsymbol{x} = (a, m)) = \mathcal{N}(d|\alpha + \beta_a a + \beta_m m, \sigma^2)$, using vague priors for the parameters. (In this example, we use a Laplace approximation to the posterior, discussed in Section 7.4.3.) We then compute the posterior predictive distribution $p(y|\boldsymbol{x}_n, \mathcal{D})$, which is a 1d Gaussian, and plot this vs each observed outcome y_n.

The result is shown in Figure 3.22. We see several outliers, some of which have been annotated. In particular, we see that both Idaho (ID) and Utah (UT) have a much lower divorce rate than predicted. This is because both of these states have an unusually large proportion of Mormons.

Of course, we expect errors in our predictive models. However, ideally the predictive error bars

for the inputs where the model is wrong would be larger, rather than the model confidently making errors. In this case, the overconfidence arises from our incorrect use of a linear model.

4 Probabilistic graphical models

4.1 Introduction

> I basically know of two principles for treating complicated systems in simple ways: the first is the principle of modularity and the second is the principle of abstraction. I am an apologist for computational probability in machine learning because I believe that probability theory implements these two principles in deep and intriguing ways — namely through factorization and through averaging. Exploiting these two mechanisms as fully as possible seems to me to be the way forward in machine learning. — Michael Jordan, 1997 (quoted in [Fre98]).

Probabilistic graphical models (**PGMs**) provide a convenient formalism for defining joint distributions on sets of random variables. In such graphs, the nodes represent random variables, and the (lack of) edges represent **conditional independence** (**CI**) assumptions between these variables. A better name for these models would be "independence diagrams", but the term "graphical models" is now entrenched.

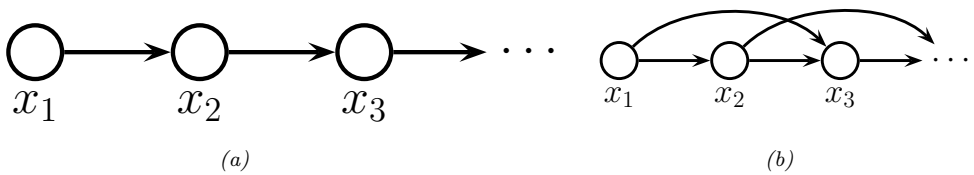
There are several kinds of graphical model, depending on whether the graph is directed, undirected, or some combination of directed and undirected, as we discuss in the sections below. More details on graphical models can be found in e.g., [KF09a].

4.2 Directed graphical models (Bayes nets)

In this section, we discuss directed probabilistic graphical models, or **DPGM**, which are based on **directed acyclic graphs** or **DAGs** (graphs that do not have any directed cycles). PGMs based on a DAG are often called **Bayesian networks** or **Bayes nets** for short; however, there is nothing inherently "Bayesian" about Bayesian networks: they are just a way of defining probability distributions. They are are also sometimes called **belief networks**. The term "belief" here refers to subjective probability. However, the probabilities used in these models are no more (and no less) subjective than in any other kind of probabilistic model.

4.2.1 Representing the joint distribution

The key property of a DAG is that the nodes can be ordered such that parents come before children. This is called a **topological ordering**. Given such an order, we define the **ordered Markov property** to be the assumption that a node is conditionally independent of all its predecessors in



(a) *(b)*

Figure 4.1: Illustration of first and second order Markov models.

the ordering given its parents, i.e.,

$$x_i \perp \boldsymbol{x}_{\text{pred}(i)\setminus\text{pa}(i)} | \boldsymbol{x}_{\text{pa}(i)} \tag{4.1}$$

where $\text{pa}(i)$ are the parents of node i, and $\text{pred}(i)$ are the predecessors of node i in the ordering. Consequently, we can represent the joint distribution as follows (assuming we use node ordering $1 : N_G$):

$$p(\boldsymbol{x}_{1:N_G}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\dots p(x_{N_G}|x_1, \dots, x_{N_G-1}) = \prod_{i=1}^{N_G} p(x_i|\boldsymbol{x}_{\text{pa}(i)}) \tag{4.2}$$

where $p(x_i|\boldsymbol{x}_{\text{pa}(i)})$ is the **conditional probability distribution** or **CPD** for node i. (The parameters of this distribution are omitted from the notation for brevity.)

The key advantage of the representation used in Equation (4.2) is that the number of parameters used to specify the joint distribution is substantially less, by virtue of the conditional independence assumptions that we have encoded in the graph, than an unstructured joint distribution. To see this, suppose all the variables are discrete and have K states each. Then an unstructured joint distribution needs $O(K^{N_G})$ parameters to specify the probability of every configuration. By contrast, with a DAG in which each node has at most N_P parents, we only need $O(N_G K^{N_P+1})$ parameters, which can be exponentially fewer if the DAG is sparse.

We give some examples of DPGM's in Section 4.2.2, and in Section 4.2.4, we discuss how to read off other conditional independence properties from the graph.

4.2.2 Examples

In this section, we give several examples of models that can be usefully represented as DPGM's.

4.2.2.1 Markov chains

We can represent the conditional independence assumptions of a first-order Markov model using the chain-structured DPGM shown in Figure 4.1(a). Consider a variable at a single time step t, which we call the "present". From the diagram, we see that information cannot flow from the past, $\boldsymbol{x}_{1:t-1}$, to the the future, $\boldsymbol{x}_{t+1:T}$, except via the present, x_t. (We formalize this in Section 4.2.4.) This means that the $\boldsymbol{x}_t$ is a sufficient statistic for the past, so the model is first-order Markov. This implies that the corresponding joint distribution can be written as follows:

$$p(\boldsymbol{x}_{1:T}) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_T|x_{T-1}) = p(x_1)\prod_{t=2}^{T} p(x_t|\boldsymbol{x}_{1:t-1}) \tag{4.3}$$

For discrete random variables, we can represent corresponding CPDs, $p(x_t = k|x_{t-1} = j)$, as a 2d table, known as a **conditional probability table** or **CPT**, $p(x_t = k|x_{t-1} = j) = \theta_{jk}$, where $0 \leq \theta_{jk} \leq 1$ and $\sum_{k=1}^{K} \theta_{jk} = 1$ (i.e., each row sums to 1).

The first-order Markov assumption is quite restrictive. If we want to allow for dependencies two steps into the past, we can create a Markov model of order 2. This is shown in Figure 4.1(b). The corresponding joint distribution has the form

$$p(\boldsymbol{x}_{1:T}) = p(x_1, x_2)p(x_3|x_1, x_2)p(x_4|x_2, x_3) \cdots p(x_T|x_{T-2}, x_{T-1}) = p(x_1, x_2) \prod_{t=3}^{T} p(x_t|\boldsymbol{x}_{t-2:t-1}) \quad (4.4)$$

As we increase the order of the Markov model, we need to add more edges. In the limit, the DAG becomes fully connected (subject to being acyclic), as shown in Figure 22.1. However, in this case, there are no useful conditional independencies, so the graphical model has no value.

4.2.2.2 The "Student" network

Figure 4.2 shows a model for capturing the inter-dependencies between 5 discrete random variables related to a hypothetical student taking a class: D = difficulty of class (easy, hard), I = intelligence (low, high), G = grade (A, B, C), S = SAT score (bad, good), L = letter of recommendation (bad, good). (This is a simplification of the "**Student network**" from [KF09a, p.281].) The chain rule tells us that we can represent the joint as follows:

$$p(D, I, G, L, S) = p(L|S, G, D, I) \times p(S|G, D, I) \times p(G|D, I) \times p(D|I) \times p(I) \quad (4.5)$$

where we have ordered the nodes topologically as I, D, G, S, L. Note that L is conditionally independent of all the other nodes earlier in this ordering given its parent G, so we can replace $p(L|S, G, D, I)$ by $p(L|G)$. We can simplify the other terms in a similar way to get

$$p(D, I, G, L, S) = p(L|G) \times p(S|I) \times p(G|D, I) \times p(D) \times p(I) \quad (4.6)$$

The ability to simplify a joint distribution in a product of small local pieces is the key idea behind graphical models.

In addition to the graph structure, we need to specify the conditional probability distributions (CPDs) at each node. For discrete random variables, we can represent the CPD as a table, which means we have a separate row (i.e., a separate categorical distribution) for each **conditioning case**, i.e., for each combination of parent values. We can represent the i'th CPT as follows:

$$\theta_{ijk} \triangleq p(x_i = k|\boldsymbol{x}_{\text{pa}(i)} = j) \quad (4.7)$$

The matrix $\boldsymbol{\theta}_{i,:,:}$ is a **row stochastic matrix**, that satisfies the properties $0 \leq \theta_{ijk} \leq 1$ and $\sum_{k=1}^{K_i} \theta_{ijk} = 1$ for each row j. Here i indexes nodes, $i \in [N_G]$; k indexes node states, $k \in [K_i]$, where K_i is the number of states for node i; and j indexes joint parent states, $j \in [J_i]$, where $J_i = \prod_{p \in \text{pa}(i)} K_p$.

The CPTs for the student network are shown next to each node in Figure 4.2. For example, we see that if the class is hard ($D = 1$) and the student has low intelligence ($I = 0$), the distribution over grades A, B and C we expect is $p(G|D = 1, I = 0) = [0.05, 0.25, 0.7]$; but if the student is intelligent, we get $p(G|D = 1, I = 1) = [0.5, 0.3, 0.2]$.

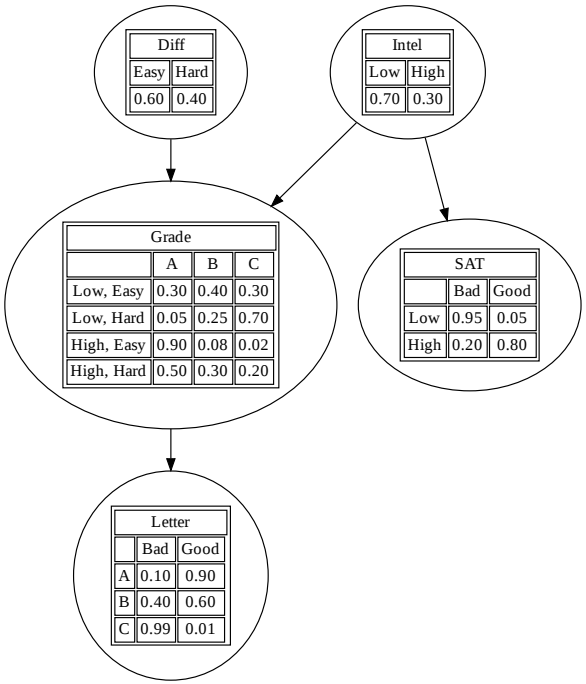
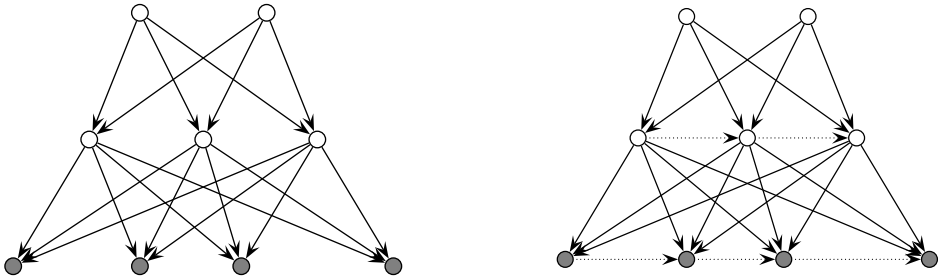


Figure 4.2: The (simplified) student network. "Diff" is the difficulty of the class. "Intel" is the intelligence of the student. "Grade" is the grade of the student in this class. "SAT" is the score of the student on the SAT exam. "Letter" is whether the teacher writes a good or bad letter of recommendation. The circles (nodes) represent random variables, the edges represent direct probabilistic dependencies. The tables inside each node represent the conditional probability distribution of the node given its parents. Generated by student_pgm.ipynb.

The number of parameters in a CPT is $O(K^{p+1})$, where K is the number of states per node, and p is the number of parents. Later we will consider more parsimonious representations, with fewer learnable parameters. (We discuss parameter learning in Section 4.2.7.)

Once we have specified the model, we can use it to answer probabilistic queries, as we discuss in Section 4.2.6. As an example, suppose we observe that the student gets a grade of C. The posterior probability that the student is intelligent is just $p(I = \text{Intelligent}|G = C) = 0.08$, since it is more likely that the low grade is explained by the class being hard (indeed, $p(D = H|G = C) = 0.63$). However, now suppose we also observe that the student gets a good SAT score. Now the posterior probability that the student is intelligent has jumped to $p(I = \text{Intelligent}|G = C, \text{SAT} = \text{Good}) = 0.58$, and probability that the class is hard has changed to $p(D = \text{Hard}|G = C, \text{SAT} = \text{Good}) = 0.76$, as shown in Figure 4.8. This negative mutual interaction between multiple causes of some observations is called the **explaining away** effect, also known as **Berkson's paradox** (see Section 4.2.4.2 for details).



(a) *(b)*

Figure 4.3: (a) Hierarchical latent variable model with 2 layers. (b) Same as (a) but with autoregressive connections within each layer. The observed $\boldsymbol{x}$ variables are the shaded leaf nodes at the bottom. The unshaded nodes are the hidden $\boldsymbol{z}$ variables.

4.2.2.3 Sigmoid belief nets

In this section, we consider a **deep generative model** of the form shown in Figure 4.3a. This corresponds to the following joint distribution:

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z}_2)p(\boldsymbol{z}_1|\boldsymbol{z}_2)p(\boldsymbol{x}|\boldsymbol{z}_1) = \prod_{k=1}^{K_2} p(z_{2,k}) \prod_{k=1}^{K_1} p(z_{1,k}|\boldsymbol{z}_2) \prod_{d=1}^{D} p(x_d|\boldsymbol{z}_1) \tag{4.8}$$

where the $\boldsymbol{x}$ nodes are the leaves the the $\boldsymbol{z}_\ell$ nodes are the internal hidden nodes. (We assume there are K_ℓ hidden nodes at level ℓ, and D visible leaf nodes.)

Now consider the special case where all the latent variables are binary, and all the latent CPDs are logistic regression models. That is,

$$p(\boldsymbol{z}_\ell|\boldsymbol{z}_{\ell+1}, \boldsymbol{\theta}) = \prod_{k=1}^{K_\ell} \text{Ber}(z_{\ell,k}|\sigma(\boldsymbol{w}_{\ell,k}^\mathsf{T}\boldsymbol{z}_{\ell+1})) \tag{4.9}$$

where $\sigma(u) = 1/(1+e^{-u})$ is the sigmoid (logistic) function. The result is called a **sigmoid belief net** [Nea92].

At the bottom layer, $p(\boldsymbol{x}|\boldsymbol{z}_1, \boldsymbol{\theta})$, we use whatever observation model is appropriate for the type of data we are dealing with. For example, for real valued data, we might use

$$p(\boldsymbol{x}|\boldsymbol{z}_1, \boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(x_d|\boldsymbol{w}_{1,d,\mu}^\mathsf{T}\boldsymbol{z}_1, \exp(\boldsymbol{w}_{1,d,\sigma}^\mathsf{T}\boldsymbol{z}_1)) \tag{4.10}$$

where $\boldsymbol{w}_{1,d,\mu}$ are the weights that control the mean of the d'th output, and $\boldsymbol{w}_{1,d,\sigma}$ are the weights that control the variance of the d'th output.

We can also add directed connections between the hidden variables within a layer, as shown in Figure 4.3b. This is called a **deep autoregressive network** or **DARN** model [Gre+14], which combines ideas from latent variable modeling and autoregressive modeling.

We discuss other forms of hierarchical generative models in Chapter 21.

4.2.3 Gaussian Bayes nets

Consider a DPGM where all the variables are real-valued, and all the CPDs have the following form, known as a **linear Gaussian CPD**:

$$p(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) = \mathcal{N}(x_i|\mu_i + \boldsymbol{w}_i^\mathsf{T}\boldsymbol{x}_{\mathrm{pa}(i)}, \sigma_i^2) \tag{4.11}$$

As we show below, multiplying all these CPDs together results in a large joint Gaussian distribution of the form $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{x} \in \mathbb{R}^{N_G}$. This is called a **directed Gaussian graphical model** or a **Gaussian Bayes net**.

We now explain how to derive $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, following [SK89, App. B]. For convenience, we will rewrite the CPDs in the following form:

$$x_i = \mu_i + \sum_{j \in \mathrm{pa}(i)} w_{i,j}(x_j - \mu_j) + \sigma_i z_i \tag{4.12}$$

where $z_i \sim \mathcal{N}(0,1)$, σ_i is the conditional standard deviation of x_i given its parents, $w_{i,j}$ is the strength of the $j \to i$ edge, and μ_i is the local mean.[1]

It is easy to see that the global mean is just the concatenation of the local means, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{N_G})$. We now derive the global covariance, $\boldsymbol{\Sigma}$. Let $\mathbf{S} \triangleq \mathrm{diag}(\boldsymbol{\sigma})$ be a diagonal matrix containing the standard deviations. We can rewrite Equation (4.12) in matrix-vector form as follows:

$$(\boldsymbol{x} - \boldsymbol{\mu}) = \mathbf{W}(\boldsymbol{x} - \boldsymbol{\mu}) + \mathbf{S}\boldsymbol{z} \tag{4.13}$$

where $\mathbf{W}$ is the matrix of regression weights. Now let $\boldsymbol{e}$ be a vector of noise terms: $\boldsymbol{e} \triangleq \mathbf{S}\boldsymbol{z}$. We can rearrange this to get $\boldsymbol{e} = (\mathbf{I} - \mathbf{W})(\boldsymbol{x} - \boldsymbol{\mu})$. Since $\mathbf{W}$ is lower triangular (because $w_{j,i} = 0$ if $j < i$ in the topological ordering), we have that $\mathbf{I} - \mathbf{W}$ is lower triangular with 1s on the diagonal. Hence

$$\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N_G} \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ -w_{2,1} & 1 & & & \\ -w_{3,2} & -w_{3,1} & 1 & & \\ \vdots & & & \ddots & \\ -w_{N_G,1} & -w_{N_G,2} & \ldots & -w_{N_G,N_G-1} & 1 \end{pmatrix} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \\ \vdots \\ x_{N_G} - \mu_{N_G} \end{pmatrix} \tag{4.14}$$

Since $\mathbf{I} - \mathbf{W}$ is always invertible, we can write

$$\boldsymbol{x} - \boldsymbol{\mu} = (\mathbf{I} - \mathbf{W})^{-1}\boldsymbol{e} \triangleq \mathbf{U}\boldsymbol{e} = \mathbf{U}\mathbf{S}\boldsymbol{z} \tag{4.15}$$

where we defined $\mathbf{U} = (\mathbf{I} - \mathbf{W})^{-1}$. Hence the covariance is given by

$$\boldsymbol{\Sigma} = \mathrm{Cov}\left[\boldsymbol{x}\right] = \mathrm{Cov}\left[\boldsymbol{x} - \boldsymbol{\mu}\right] = \mathrm{Cov}\left[\mathbf{U}\mathbf{S}\boldsymbol{z}\right] = \mathbf{U}\mathbf{S}\ \mathrm{Cov}\left[\boldsymbol{z}\right]\ \mathbf{S}\mathbf{U}^\mathsf{T} = \mathbf{U}\mathbf{S}^2\mathbf{U}^\mathsf{T} \tag{4.16}$$

since $\mathrm{Cov}\left[\boldsymbol{z}\right] = \mathbf{I}$.

1. If we do not subtract off the parent's mean (i.e., if we use $x_i = \mu_i + \sum_{j \in \mathrm{pa}(i)} w_{i,j}x_j + \sigma_i z_i$), the derivation of $\boldsymbol{\Sigma}$ is much messier, as can be seen by looking at [Bis06, p370].

4.2.4 Conditional independence properties

We will write $\boldsymbol{x}_A \perp_G \boldsymbol{x}_B | \boldsymbol{x}_C$ if A is conditionally independent of B given C in the graph G. (We discuss how to determine whether such a CI property is implied by a given graph in the sections below.) Let $I(G)$ be the set of all such CI statements encoded by the graph, and $I(p)$ be the set of all such CI statements that hold true in some distribution p. We say that G is an **I-map** (independence map) for p, or that p is **Markov** wrt G, iff $I(G) \subseteq I(p)$. In other words, the graph is an I-map if it does not make any assertions of CI that are not true of the distribution. This allows us to use the graph as a safe proxy for p when reasoning about p's CI properties. This is helpful for designing algorithms that work for large classes of distributions, regardless of their specific numerical parameters. Note that the fully connected graph is an I-map of all distributions, since it makes no CI assertions at all, as we show below. We therefore say G is a **minimal I-map** of p if G is an I-map of p, and if there is no $G' \subseteq G$ which is an I-map of p.

We now turn to the question of how to derive $I(G)$, i.e., which CI properties are entailed by a DAG.

4.2.4.1 Global Markov properties (d-separation)

We say an *undirected path* P is **d-separated** by a set of nodes C (containing the evidence) iff at least one of the following conditions hold:

1. P contains a chain or **pipe**, $s \to m \to t$ or $s \leftarrow m \leftarrow t$, where $m \in C$
2. P contains a tent or **fork**, $s \swarrow^{m} \searrow t$, where $m \in E$
3. P contains a **collider** or **v-structure**, $s \searrow_{m} \swarrow t$, where m is not in C and neither is any descendant of m.

Next, we say that a *set of nodes* A is d-separated from a different set of nodes B given a third observed set C iff each undirected path from every node $a \in A$ to every node $b \in B$ is d-separated by C. Finally, we define the CI properties of a DAG as follows:

$$\mathbf{X}_A \perp_G \mathbf{X}_B | \mathbf{X}_C \iff \text{A is d-separated from B given C} \tag{4.17}$$

This is called the (directed) **global Markov property**.

The **Bayes ball algorithm** [Sha98] is a simple way to see if A is d-separated from B given C, based on the above definition. The idea is this. We "shade" all nodes in C, indicating that they are observed. We then place "balls" at each node in A, let them "bounce around" according to some rules, and then ask if any of the balls reach any of the nodes in B. The three main rules are shown in Figure 4.4. Notice that balls can travel opposite to edge directions. We see that a ball can pass through a chain, but not if it is shaded in the middle. Similarly, a ball can pass through a fork, but not if it is shaded in the middle. However, a ball cannot pass through a v-structure, unless it is shaded in the middle.

We can justify the 3 rules of Bayes ball as follows. First consider a chain structure $X \to Y \to Z$, which encodes

$$p(x, y, z) = p(x)p(y|x)p(z|y) \tag{4.18}$$

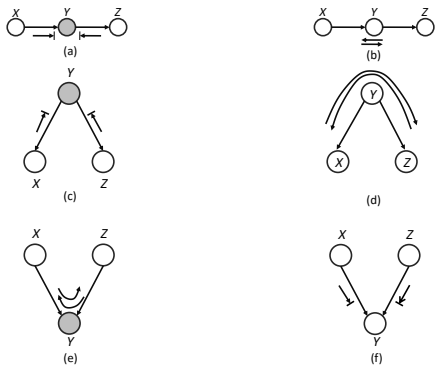


Figure 4.4: Bayes ball rules. A shaded node is one we condition on. If there is an arrow hitting a bar, it means the ball cannot pass through; otherwise the ball can pass through.

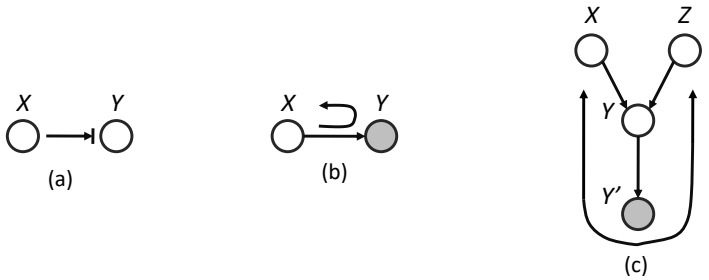


Figure 4.5: (a-b) Bayes ball boundary conditions. (c) Example of why we need boundary conditions. Y' is an observed child of Y, rendering Y "effectively observed", so the ball bounces back up on its way from X to Z.

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x, y,)}{p(y)} = \frac{p(x)p(y|x)p(z|y)}{p(y)} = \frac{p(x, y)p(z|y)}{p(y)} = p(x|y)p(z|y) \tag{4.19}$$

and therefore $X \perp Z \mid Y$. So observing the middle node of chain breaks it in two (as in a Markov chain).

Now consider the tent structure $X \leftarrow Y \rightarrow Z$. The joint is

$$p(x, y, z) = p(y)p(x|y)p(z|y) \tag{4.20}$$

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x, y, z)}{p(y)} = \frac{p(y)p(x|y)p(z|y)}{p(y)} = p(x|y)p(z|y) \tag{4.21}$$

and therefore $X \perp Z \mid Y$. So observing a root node separates its children (as in a naive Bayes classifier: see Section 4.2.8.2).

X	Y	Z
D	I	
D	I	S
D	S	
D	S	I
D	S	L, I
D	S	G, I
D	S	G, L, I
D	L	G
D	L	G, S
D	L	G, I
D	L	I, G, S

Table 4.1: Conditional independence relationships implied by the student DAG (Figure 4.2). Each line has the form $X \perp Y|Z$. Generated by student_ pgm.ipynb.

Finally consider a v-structure $X \to Y \leftarrow Z$. The joint is

$$p(x, y, z) = p(x)p(z)p(y|x, z) \tag{4.22}$$

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x)p(z)p(y|x, z)}{p(y)} \tag{4.23}$$

so $X \not\perp Z|Y$. However, in the unconditional distribution, we have

$$p(x, z) = p(x)p(z) \tag{4.24}$$

so we see that X and Z are marginally independent. So we see that conditioning on a common child at the bottom of a v-structure makes its parents become dependent. This important effect is called **explaining away**, **inter-causal reasoning**, or **Berkson's paradox** (see Section 4.2.4.2 for a discussion).

Finally, Bayes Ball also needs the "boundary conditions" shown in Figure 4.5(a-b). These rules say that a ball hitting a hidden leaf stops, but a ball hitting an observed leaf "bounces back". To understand where this rule comes from, consider Figure 4.5(c). Suppose Y' is a (possibly noisy) copy of Y. If we observe Y', we effectively observe Y as well, so the parents X and Z have to compete to explain this. So if we send a ball down $X \to Y \to Y'$, it should "bounce back" up along $Y' \to Y \to Z$, in order to pass information between the parents. However, if Y and *all* its children are hidden, the ball does not bounce back.

As an example of the CI statements encoded by a DAG, Table 4.1 shows some properties that follow from the student network in Figure 4.2.

4.2.4.2 Explaining away (Berkson's paradox)

In this section, we give some examples of the **explaining away** phenomenon, also called **Berkson's paradox**.

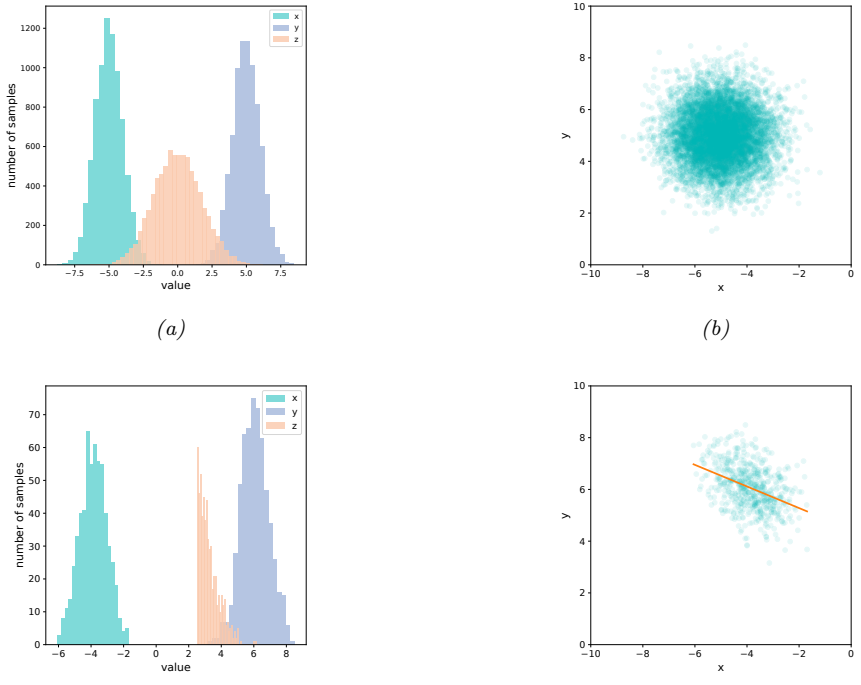


Figure 4.6: Samples from a jointly Gaussian DPGM, $p(x, y, z) = \mathcal{N}(x|-5, 1)\mathcal{N}(y|5, 1)\mathcal{N}(z|x + y, 1)$. (a) Unconditional marginal distributions, $p(x)$, $p(y)$, $p(z)$. (b) Unconditional joint distribution, $p(x, y)$. (c) Conditional marginal distribution, $p(x|z > 2.5)$, $p(y|z > 2.5)$, $p(z|z > 2.5)$. (d) Conditional joint distribution, $p(x, y|z > 2.5)$. Adapted from [Clo20]. Generated by berksons_gaussian.ipynb.

As a simple example (from [PM18b, p198]), consider tossing two coins 100 times. Suppose you only record the outcome of the experiment if at least one coin shows up heads. You should expect to record about 75 entries. You will see that every time coin 1 is recorded as tails, coin 2 will be recorded as heads. If we ignore the way in which the data was collected, we might infer from the fact that that coins 1 and 2 are correlated that there is a hidden common cause. However, the correct explanation is that the correlation is due to conditioning on a hidden common effect (namely the decision of whether to record the outcome or not, so we can censor tail-tail events). This is called **selection bias**.

As another example of this, consider a Gaussian DPGM of the form

$$p(x, y, z) = \mathcal{N}(x|-5, 1)\mathcal{N}(y|5, 1)\mathcal{N}(z|x + y, 1) \tag{4.25}$$

The graph structure is $X \rightarrow Z \leftarrow Y$, where Z is the child node. Some samples from the unconditional joint distribution $p(x, y, z)$ are shown in Figure 4.6(a); we see that X and Y are uncorrelated. Now suppose we only select samples where $z > 2.5$. Some samples from the conditional joint distribution

$p(x, y|z > 2.5)$ are shown in Figure 4.6(d); we see that now X and Y are correlated. This could cause us to erroneously conclude that there is a causal relationship, but in fact the dependency is caused by selection bias.

4.2.4.3 Markov blankets

The smallest set of nodes that renders a node i conditionally independent of all the other nodes in the graph is called i's **Markov blanket**; we will denote this by $\text{mb}(i)$. Below we show that the Markov blanket of a node in a DPGM is equal to the parents, the children, and the **co-parents**, i.e., other nodes who are also parents of its children:

$$\text{mb}(i) \triangleq \text{ch}(i) \cup \text{pa}(i) \cup \text{copa}(i) \tag{4.26}$$

See Figure 4.7 for an illustration.

To see why this is true, let us partition all the nodes into the target node X_i, its parents U, its children Y, its coparents Z, and the other variables O. Let X_{-i} be all the nodes except X_i. Then we have

$$p(X_i|X_{-i}) = \frac{p(X_i, X_{-i})}{\sum_x p(X_i = x, X_{-i})} \tag{4.27}$$

$$= \frac{p(X_i, U, Y, Z, O)}{\sum_x p(X_i = x, U, Y, Z, O)} \tag{4.28}$$

$$= \frac{p(X_i|U)[\prod_j p(Y_j|X_i, Z_j)]P(U, Z, O)}{\sum_x p(X_i = x|U)[\prod_j p(Y_j|X_i = x, Z_j)]P(U, Z, O)} \tag{4.29}$$

$$= \frac{p(X_i|U)[\prod_j p(Y_j|X_i, Z_j)]}{\sum_x p(X_i = x|U)[\prod_j p(Y_j|X_i = x, Z_j)]} \tag{4.30}$$

$$\propto p(X_i|\text{pa}(X_i)) \prod_{Y_j \in \text{ch}(X_i)} p(Y_j|\text{pa}(Y_j))) \tag{4.31}$$

where $\text{ch}(X_i)$ are the children of X_i and $\text{pa}(Y_j)$ are the parents of Y_j. We see that the terms that do not involve X_i cancel out from the numerator and denominator, so we are left with a product of terms that include X_i in their "scope". Hence the **full conditional** for node i becomes

$$p(x_i|\boldsymbol{x}_{-i}) = p(x_i|\boldsymbol{x}_{\text{mb}(i)}) \propto p(x_i|\boldsymbol{x}_{\text{pa}(i)}) \prod_{k \in \text{ch}(i)} p(x_k|\boldsymbol{x}_{\text{pa}(k)}) \tag{4.32}$$

We will see applications of this in Gibbs sampling (Equation (12.19)), and mean field variational inference (Equation (10.28)).

4.2.4.4 Other Markov properties

From the d-separation criterion, one can conclude that

$$i \perp \text{nd}(i) \setminus \text{pa}(i)|\text{pa}(i) \tag{4.33}$$

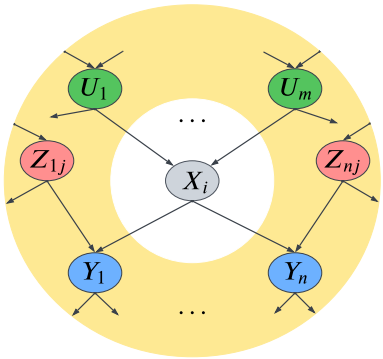


Figure 4.7: Illustration of the Markov blanket of a node in a directed graphical model. The target node X_i is shown in gray, its parents $U_{1:m}$ are shown in green, its children $Y_{1:n}$ are shown in blue, and its coparents $Z_{1:n,1:j}$ are shown in red. X_i is conditionally independent of all the other variables in the model given these variables. Adapted from Figure 13.4b of [RN19].

where the **non-descendants** of a node $\text{nd}(i)$ are all the nodes except for its descendants, $\text{nd}(i) = \{1, \ldots, N_G\} \setminus \{i \cup \text{desc}(i)\}$. Equation (4.33) is called the (directed) **local Markov property**. For example, in Figure 4.23(a), we have $\text{nd}(3) = \{1, 2, 4\}$, and $\text{pa}(3) = 1$, so $3 \perp 2, 4|1$.

A special case of this property is when we only look at predecessors of a node according to some topological ordering. We have

$$i \perp \text{pred}(i) \setminus \text{pa}(i)|\text{pa}(i) \tag{4.34}$$

which follows since $\text{pred}(i) \subseteq \text{nd}(i)$. This is called the **ordered Markov property**, which justifies Equation (4.2). For example, in Figure 4.23(a), if we use the ordering $1, 2, \ldots, 7$. we find $\text{pred}(3) = \{1, 2\}$ and $\text{pa}(3) = 1$, so $3 \perp 2|1$.

We have now described three Markov properties for DAGs: the directed global Markov property G in Equation (4.17), the directed local Markov property L in Equation (4.33), and the ordered Markov property O in Equation (4.34), It is obvious that $G \implies L \implies O$. What is less obvious, but nevertheless true, is that $O \implies L \implies G$ (see e.g., [KF09a] for the proof). Hence all these properties are equivalent.

Furthermore, any distribution p that is Markov wrt a graph can be factorized as in Equation (4.2); this is called the **factorization property** F. It is obvious that $O \implies F$, but one can show that the converse also holds (see e.g., [KF09a] for the proof).

4.2.5 Generation (sampling)

It is easy to generate prior samples from a DPGM: we simply visit the nodes in **topological order**, parents before children, and then sample a value for each node given the value of its parents. This will generate independent samples from the joint, $(x_1, \ldots, x_{N_G}) \sim p(\boldsymbol{x}|\boldsymbol{\theta})$. This is called **ancestral sampling**.

4.2.6 Inference

In the context of PGMs, the term "**inference**" refers to the task of computing the posterior over a set of **query nodes** Q given the observed values for a set of **visible nodes** V, while marginalizing over the irrelevant **nuisance variables**, $R = \{1, \ldots, N_G\} \setminus \{Q, V\}$:

$$p_{\boldsymbol{\theta}}(Q|V) = \frac{p_{\boldsymbol{\theta}}(Q, V)}{p_{\boldsymbol{\theta}}(V)} = \frac{\sum_R p_{\boldsymbol{\theta}}(Q, V, R)}{p_{\boldsymbol{\theta}}(V)} \tag{4.35}$$

(If the variables are continuous, we should replace sums with integrals.) If Q is a single node, then $p_{\boldsymbol{\theta}}(Q|V)$ is called the **posterior marginal** for node Q.

As an example, suppose $V = \boldsymbol{x}$ is a sequence of observed sound waves, $Q = \boldsymbol{z}$ is the corresponding set of unknown spoken words, and $R = \boldsymbol{r}$ are random "non-semantic" factors associated with the signal, such as prosody or background noise. Our goal is to compute the posterior over the words given the sounds, while being invariant to the irrelevant factors:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \sum_{\boldsymbol{r}} p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{r}|\boldsymbol{x}) = \sum_{\boldsymbol{r}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} = \sum_{\boldsymbol{r}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{x})}{\sum_{\boldsymbol{z}', \boldsymbol{r}'} p_{\boldsymbol{\theta}}(\boldsymbol{z}', \boldsymbol{r}', \boldsymbol{x})} \tag{4.36}$$

As a simplification, we can "lump" the random factors R into the query set Q to define the complete set of **hidden variables** $H = Q \cup R$. In this case, the tasks simpifies to

$$p_{\boldsymbol{\theta}}(\boldsymbol{h}|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{h}, \boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{h}, \boldsymbol{x})}{\sum_{\boldsymbol{h}'} p_{\boldsymbol{\theta}}(\boldsymbol{h}', \boldsymbol{x})} \tag{4.37}$$

The computational complexity of the inference task depends on the CI properties of the graph, as we discuss in Chapter 9. In general it is NP-hard (see Section 9.4.4), but for certain graph structures (such as chains, trees and other sparse graphs), it can be solved efficiently (in polynomial) time using dynamic programming (see Chapter 9). For cases where it is intractable, we can use standard methods for approximate Bayesian inference, which we review in Chapter 7.

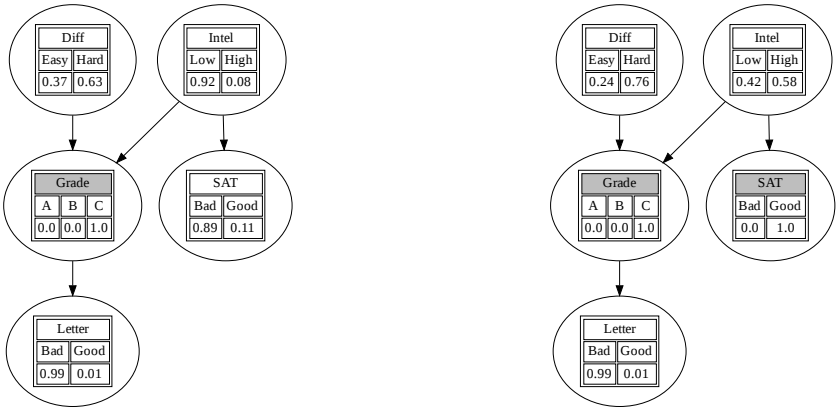
4.2.6.1 Example: inference in the Student network

As an example of inference in PGMs, consider the Student network from Section 4.2.2.2. Suppose we observe that the student gets a grade of C. The posterior marginals are shown in Figure 4.8a. We see that the low grade could be explained by the class being hard (since $p(D = \text{Hard}|G = C) = 0.63$), but is more likely explained by the student having low intelligence (since $p(I = \text{High}|G = C) = 0.08$).

However, now suppose we *also* observe that the student gets a good SAT score. The new posterior marginals are shown in Figure 4.8b. Now the posterior probability that the student is intelligent has jumped to $p(I = \text{High}|G = C, \text{SAT} = \text{Good}) = 0.58$, since otherwise it would be difficult to explain the good SAT score. Once we believe the student has high intelligence, we have to explain the C grade by assuming the class is hard, and indeed we find that the probability that the class is hard has increased to $p(D = \text{Hard}|G = C) = 0.76$. (This negative mutual interaction between multiple causes of some observations is called the explaining away effect, and is discussed in Section 4.2.4.2.)

4.2.7 Learning

So far, we have assumed that the structure G and parameters $\boldsymbol{\theta}$ of the PGM are known. However, it is possible to learn both of these from data. For details on how to learn G from data, see Section 30.3.



(a) *(b)*

Figure 4.8: Illustration of belief updating in the "Student" PGM. The histograms show the marginal distribution of each node. Nodes with shaded titles are clamped to an observed value. (a) Posterior after conditioning on Grade=C. (b) Posterior after also conditioning on SAT=Good. Generated by student_pgm.ipynb.

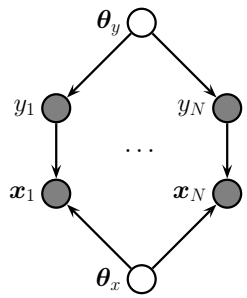


Figure 4.9: A DPGM representing the joint distribution $p(\boldsymbol{y}_{1:N}, \boldsymbol{x}_{1:N}, \boldsymbol{\theta}_y, \boldsymbol{\theta}_x)$. Here $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$ are global parameter nodes that are shared across the examples, whereas $\boldsymbol{x}_n$ and y_n are local variables.

Here we focus on **parameter learning**, i.e., computing the posterior $p(\boldsymbol{\theta}|\mathcal{D}, G)$. (Henceforth we will drop the conditioning on G, since we assume the graph structure is fixed.)

We can compute the parameter posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by treating $\boldsymbol{\theta}$ as "just another hidden variable", and then performing inference. However, in the machine learning community, it is more common to just compute a point estimate of the parameters, such as the posterior mode, $\hat{\boldsymbol{\theta}} = \operatorname{argmax} p(\boldsymbol{\theta}|\mathcal{D})$. This approximation is often reasonable, since the parameters depend on all the data, rather than just a single data point, and are therefore less uncertain than other hidden variables.

4.2.7.1 Learning from complete data

Figure 4.9 represents a graphical model for a typical supervised learning problem. We have N **local variables**, $\boldsymbol{x}_n$ and $\boldsymbol{y}_n$, and 2 **global variables**, corresponding to the parameters, which are shared across data samples. The local variables are observed (in the training set), so they are represented by solid (shaded) nodes. The global variables are not observed, and hence are represented by empty (unshaded) nodes. (The model represents a generative classifier, so the edge is from y_n to $\boldsymbol{x}_n$; if we are fitting a discriminative classifier, the edge would be from $\boldsymbol{x}_n$ to y_n, and there would be no $\boldsymbol{\theta}_y$ prior node.)

From the CI properties of Figure 4.9, it follows that the joint distribution factorizes into a product of terms, one per node:

$$p(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}_x)p(\boldsymbol{\theta}_y)\left[\prod_{n=1}^{N} p(y_n|\boldsymbol{\theta}_y)p(\boldsymbol{x}_n|y_n, \boldsymbol{\theta}_x)\right] \tag{4.38}$$

$$= \left[p(\boldsymbol{\theta}_y)\prod_{n=1}^{N} p(y_n|\boldsymbol{\theta}_y)\right]\left[p(\boldsymbol{\theta}_x)\prod_{n=1}^{N} p(\boldsymbol{x}_n|y_n, \boldsymbol{\theta}_x)\right] \tag{4.39}$$

$$= [p(\boldsymbol{\theta}_y)p(\mathcal{D}_y|\boldsymbol{\theta}_y)]\,[p(\boldsymbol{\theta}_x)p(\mathcal{D}_x|\boldsymbol{\theta}_x)] \tag{4.40}$$

where $\mathcal{D}_y = \{y_n\}_{n=1}^N$ is the data that is sufficient for estimating $\boldsymbol{\theta}_y$ and $\mathcal{D}_x = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ is the data that is sufficient for $\boldsymbol{\theta}_x$.

From Equation (4.40), we see that the prior, likelihood and posterior all **decompose** or factorize according to the graph structure. Thus we can compute the posterior for each parameter independently. In general, we have

$$p(\boldsymbol{\theta}, \mathcal{D}) = \prod_{i=1}^{N_G} p(\boldsymbol{\theta}_i)p(\mathcal{D}_i|\boldsymbol{\theta}_i) \tag{4.41}$$

Hence the likelihood and prior factorizes, and thus so does the posterior. If we just want to compute the MLE, we can compute

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{N_G} p(\mathcal{D}_i|\boldsymbol{\theta}_i) \tag{4.42}$$

We can solve this for each node independently, as we illustrate in Section 4.2.7.2.

4.2.7.2 Example: computing the MLE for CPTs

In this section, we illustrate how to compute the MLE for tabular CPDs. The likelihood is given by the following product of multinomials:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{i=1}^{N_G} p(x_{ni}|\boldsymbol{x}_{n,\mathrm{pa}(i)}, \boldsymbol{\theta}_i) \tag{4.43}$$

$$= \prod_{n=1}^{N}\prod_{i=1}^{N_G}\prod_{j=1}^{J_i}\prod_{k=1}^{K_i} \theta_{ijk}^{\mathbb{I}\left(x_{ni}=k, \boldsymbol{x}_{n,\mathrm{pa}(i)}=j\right)} \tag{4.44}$$

I	D	G	S	L
0	0	2	0	0
0	1	2	0	0
0	0	1	1	1
1	1	1	1	0
1	0	0	1	1
0	0	0	0	1
1	1	2	1	1

Table 4.2: Some fully observed training data for the student network.

I	D	$N_{i,j,k}$	$\hat{\theta}_{i,j,k}$	$\overline{\theta}_{i,j,k}$
0	0	$[1,1,1]$	$[\frac{1}{3},\frac{1}{3},\frac{1}{3}]$	$[\frac{2}{6},\frac{2}{6},\frac{2}{6}]$
0	1	$[0,0,1]$	$[\frac{0}{4},\frac{0}{4},\frac{1}{4}]$	$[\frac{1}{4},\frac{1}{4},\frac{2}{4}]$
1	0	$[1,0,0]$	$[\frac{1}{4},\frac{0}{4},\frac{0}{4}]$	$[\frac{2}{4},\frac{1}{4},\frac{1}{4}]$
1	1	$[0,1,1]$	$[0,\frac{1}{2},\frac{1}{2}]$	$[\frac{1}{5},\frac{2}{5},\frac{2}{5}]$

Table 4.3: Sufficient statistics N_{ijk} and corresponding MLE $\hat{\theta}_{ijk}$ and posterior mean $\overline{\theta}_{ijk}$ for node $i = G$ in the student network. Each row corresponds to a different joint configuration of its parent nodes, coresponding to state j. The index k refers to the 3 possible values of the child node G.

where

$$\theta_{ijk} \triangleq p(x_i = k | \boldsymbol{x}_{\mathrm{pa}(i)} = j) \tag{4.45}$$

Let us define the sufficient statistics for node i to be N_{ijk}, which is the number of times that node i is in state k while its parents are in joint state j:

$$N_{ijk} \triangleq \sum_{n=1}^{N} \mathbb{I}\left(x_{n,i} = k, x_{n,\mathrm{pa}(i)} = j\right) \tag{4.46}$$

The MLE for a multinomial is given by the normalized empirical frequencies:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{k'} N_{ijk'}} \tag{4.47}$$

For example, consider the student network from Section 4.2.2.2. In Table 4.2, we show some sample training data. For example, the last line in the tabel encodes a student who is smart ($I = 1$), who takes a hard class ($D = 1$), gets a C ($G = 2$), but who does well on the SAT ($S = 1$) and gets a good letter of recommendation ($L = 1$).

In Table 4.3, we list the sufficient statistics N_{ijk} and the MLE $\hat{\theta}_{ijk}$ for node $i = G$, with parents (I, D). A similar process can be used for the other nodes. Thus we see that fitting a DPGM with tabular CPDs reduces to a simple counting problem.

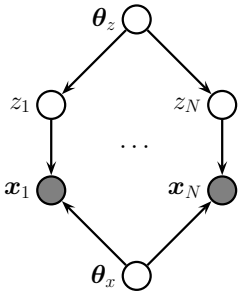


Figure 4.10: A DPGM representing the joint distribution $p(\boldsymbol{z}_{1:N}, \boldsymbol{x}_{1:N}, \boldsymbol{\theta}_z, \boldsymbol{\theta}_x)$. The local variables $\boldsymbol{z}_n$ are hidden, whereas $\boldsymbol{x}_n$ are observed. This is typical for learning unsupervised latent variable models.

However, we notice there are a lot of zeros in the sufficient statistics, due to the small sample size, resulting in extreme estimates for some of the probabilities $\hat{\theta}_{ijk}$. We discuss a (Bayesian) solution to this in Section 4.2.7.3.

4.2.7.3 Example: Computing the posterior for CPTs

In Section 4.2.7.2 we discussed how to compute the MLE for the CPTs in a discrete Bayes net. We also observed that this can suffer from the zero-count problem. In this section, we show how a Bayesian approach can solve this problem.

Let us put a separate Dirichlet prior on each row of each CPT, i.e., $\boldsymbol{\theta}_{ij} \sim \text{Dir}(\boldsymbol{\alpha}_{ij})$. Then we can compute the posterior by simply adding the pseudo counts to the empirical counts to get $\boldsymbol{\theta}_{ij}|\mathcal{D} \sim \text{Dir}(\mathbf{N}_{ij} + \boldsymbol{\alpha}_{ij})$, where $\mathbf{N}_{ij} = \{N_{ijk} : k = 1 : K_i\}$, and N_{ijk} is the number of times that node i is in state k while its parents are in state j. Hence the posterior mean estimate is given by

$$\bar{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{\sum_{k'}(N_{ijk'} + \alpha_{ijk'})} \tag{4.48}$$

The MAP estimate has the same form, except we use $\alpha_{ijk} - 1$ instead of α_{ijk}.

In Table 4.3, we illustrate this approach applied to the G node in the student network, where we use a uniform Dirichlet prior, $\alpha_{ijk} = 1$.

4.2.7.4 Learning from incomplete data

In Section 4.2.7.1, we explained that when we have complete data, the likelihood (and posterior) factorizes over CPDs, so we can estimate each CPD independently. Unfortunately, this is no longer the case when we have incomplete or missing data. To see this, consider Figure 4.10. The likelihood of the observed data can be written as follows:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{\boldsymbol{z}_{1:N}} \left[\prod_{n=1}^{N} p(\boldsymbol{z}_n|\boldsymbol{\theta}_z) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}_x) \right] \tag{4.49}$$

$$= \prod_{n=1}^{N} \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{\theta}_z) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}_x) \tag{4.50}$$

Thus the log likelihood is given by

$$\ell(\boldsymbol{\theta}) = \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{\theta}_z) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}_x) \tag{4.51}$$

The log function does not distribute over the $\sum_{\boldsymbol{z}_n}$ operation, so the objective does not decompose over nodes.[2] Consequently, we can no longer compute the MLE or the posterior by solving separate problems per node.

To solve this, we will resort to optimization methods. (We focus on the MLE case, and leave discussion of Bayesian inference for latent variable models to Part II.) In the sections below, we discuss how to use EM and SGD to find a local optimum of the (non convex) log likelihood objective.

4.2.7.5 Using EM to fit CPTs in the incomplete data case

A popular method for estimating the parameters of a DPGM in the presence of missing data is to the use the expectation maximization (EM) algorithm, as proposed in [Lau95]. We describe EM in detail in Section 6.6.3, but the basic idea is to alternate between inferring the latent variables $\boldsymbol{z}_n$ (the E or expectation step), and estimating the parameters given this completed dataset (the M or maximization step). Rather than returning the full posterior $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{\theta}^{(t)})$ in the E step, we instead return the expected sufficient statistics (ESS), which takes much less space. In the M step, we maximize the expected value of the log likelihood of the fully observed data using these ESS.

As an example, suppose all the CPDs are tabular, as in the example in Section 4.2.7.2. The log-likelihood of the complete data is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N_G} \sum_{j=1}^{J_i} \sum_{k=1}^{K_i} N_{ijk} \log \theta_{ijk} \tag{4.52}$$

and hence the expected complete data log-likelihood has the form

$$\mathbb{E}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] = \sum_i \sum_j \sum_k \overline{N}_{ijk} \log \theta_{ijk} \tag{4.53}$$

where

$$\overline{N}_{ijk} = \sum_{n=1}^{N} \mathbb{E}\left[\mathbb{I}\left(x_{ni} = k, \boldsymbol{x}_{n,\mathrm{pa}(i)} = j\right)\right] = \sum_{n=1}^{N} p(x_{ni} = k, \boldsymbol{x}_{n,\mathrm{pa}(i)} = j|\mathcal{D}_n, \boldsymbol{\theta}^{old}) \tag{4.54}$$

where $\mathcal{D}_n$ are all the visible variables in case n, and $\boldsymbol{\theta}^{old}$ are the parameters from the previous iteration. The quantity $p(x_{ni}, \boldsymbol{x}_{n,\mathrm{pa}(i)}|\mathcal{D}_n, \boldsymbol{\theta}^{old})$ is known as a **family marginal**, and can be computed using any GM inference algorithm. The $\overline{N}_{ijk}$ are the **expected sufficient statistics** (ESS), and constitute the output of the E step.

2. We can also see this from the graphical model: $\boldsymbol{\theta}_x$ is no longer independent of $\boldsymbol{\theta}_z$, because there is a path that connects them via the hidden nodes $\boldsymbol{z}_n$. (See Section 4.2.4 for an explanation of how to "read off" such CI properties from a DPGM.)

Given these ESS, the M step has the simple form

$$\hat{\theta}_{ijk} = \frac{\overline{N}_{ijk}}{\sum_{k'} \overline{N}_{ijk'}} \tag{4.55}$$

We can modify this to perform MAP estimation with a Dirichlet prior by simply adding pseudo counts to the expected counts.

The famous Baum-Welch algorithm is a special case of the above equations which arises when the DPGM is an HMM (Section 29.4.1)

4.2.7.6 Using SGD to fit CPTs in the incomplete data case

The EM algorithm is a batch algorithm. To scale up to large datasets, it is more common to use stochastic gradient descent or SGD (see e.g., [BC94; Bin+97]). To apply this, we need to compute the marginal likelihood of the observed data for each example:

$$p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{\theta}_z) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}_x) \tag{4.56}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_z, \boldsymbol{\theta}_x)$.) (We say that we have "**collapsed**" the model by marginalizing out $\boldsymbol{z}_n$.) We can then compute the log likelihood using

$$\ell(\boldsymbol{\theta}) = \log p(\mathcal{D}|\boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}_= \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{4.57}$$

The gradient of this objective can be computed as follows:

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \sum_n \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{4.58}$$

$$= \sum_n \frac{1}{p(\boldsymbol{x}_n|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{4.59}$$

$$= \sum_n \frac{1}{p(\boldsymbol{x}_n|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \left[\sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n, \boldsymbol{x}_n|\boldsymbol{\theta}) \right] \tag{4.60}$$

$$= \sum_n \sum_{\boldsymbol{z}_n} \frac{p(\boldsymbol{z}_n, \boldsymbol{x}_n|\boldsymbol{\theta})}{p(\boldsymbol{x}_n|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{z}_n, \boldsymbol{x}_n|\boldsymbol{\theta}) \tag{4.61}$$

$$= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{z}_n, \boldsymbol{x}_n|\boldsymbol{\theta}) \tag{4.62}$$

We can now apply a minibatch approximation to this in the usual way.

4.2.8 Plate notation

To make the parameters of a PGM explicit, we can add them as nodes to the graph, and treat them as hidden variables to be inferred. Figure 4.11(a) shows a simple example, in which we have N iid

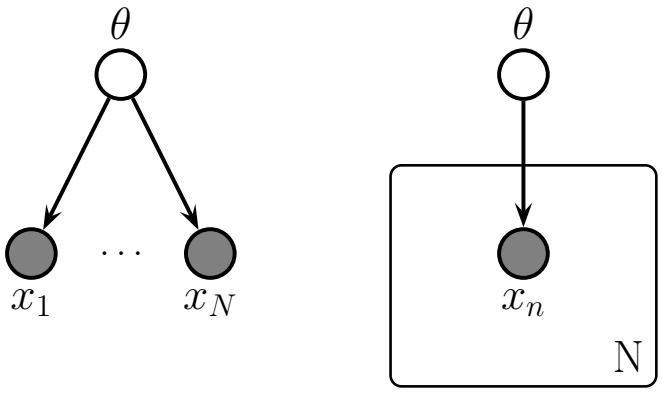


Figure 4.11: Left: data points $\boldsymbol{x}_n$ are conditionally independent given $\boldsymbol{\theta}$. Right: Same model, using plate notation. This represents the same model as the one on the left, except the repeated $\boldsymbol{x}_n$ nodes are inside a box, known as a plate; the number in the lower right hand corner, N, specifies the number of repetitions of the $\boldsymbol{x}_n$ node.

random variables, $\boldsymbol{x}_n$, all drawn from the same distribution with common parameter $\boldsymbol{\theta}$. We denote this by

$$\boldsymbol{x}_n \sim p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{4.63}$$

The corresponding joint distribution over the parameters and data $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ has the form

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) \tag{4.64}$$

where $p(\boldsymbol{\theta})$ is the prior distribution for the parameters, and $p(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood. By virtue of the iid assumption, the likelihood can be rewritten as follows:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{4.65}$$

Notice that the order of the data vectors is not important for defining this model, i.e., we can permute the leaves of the DPGM. When this property holds, we say that the data is **exchangeable**.

In Figure 4.11(a), we see that the $\boldsymbol{x}$ nodes are repeated N times. (The **shaded nodes** represent observed values, whereas the unshaded (hollow) nodes represent latent variables or parameters.) To avoid visual clutter, it is common to use a form of **syntactic sugar** called **plates**. This is a notational convention in which we draw a little box around the repeated variables, with the understanding that nodes within the box will get repeated when the model is **unrolled**. We often write the number of copies or repetitions in the bottom right corner of the box. This is illustrated in Figure 4.11(b).

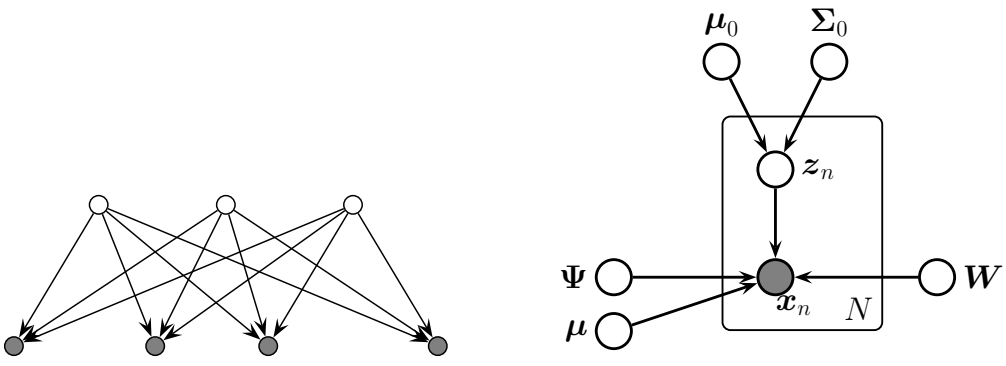


Figure 4.12: (a) Factor analysis model illustrated as a DPGM. We show the components of $\boldsymbol{z}$ (top row) and $\boldsymbol{x}$ (bottom row) as individual scalar nodes. (b) Equivalent model, where $\boldsymbol{z}$ and $\boldsymbol{x}$ are collapsed to vector-valued nodes, and parameters are added, using plate notation.

4.2.8.1 Example: factor analysis

In Section 28.3.1, we discuss the factor analysis model, which has the form

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{4.66}$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{4.67}$$

where $\mathbf{W}$ is a $D \times L$ matrix, known as the factor loading matrix, and $\boldsymbol{\Psi}$ is a diagonal $D \times D$ covariance matrix.

Note that $\boldsymbol{z}$ and $\boldsymbol{x}$ are both vectors. We can explicitly represent their components as scalar nodes as in Figure 4.12a. Here the directed edges correspond to non-zero entries in the $\mathbf{W}$ matrix.

We can also explicitly show the parameters of the model, using plate notation, as shown in Figure 4.12b.

4.2.8.2 Example: Naive Bayes classifier

In some models, we have doubly indexed variables. For example, consider a **naive Bayes classifier**. This is a simple generative classifier, defined as follows:

$$p(\boldsymbol{x}, y|\boldsymbol{\theta}) = p(y|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_d|y, \boldsymbol{\theta}_d) \tag{4.68}$$

The fact that the features $\boldsymbol{x}_{1:D}$ are considered conditionally independent given the class label y is where the term "naive" comes from. Nevertheless, this model often works surprisingly well, and is extremely east to fit.

We can represent the conditional independence assumption as shown in Figure 4.13a. We can represent the repetition over the dimension d with a plate. When we turn to infering the parameters

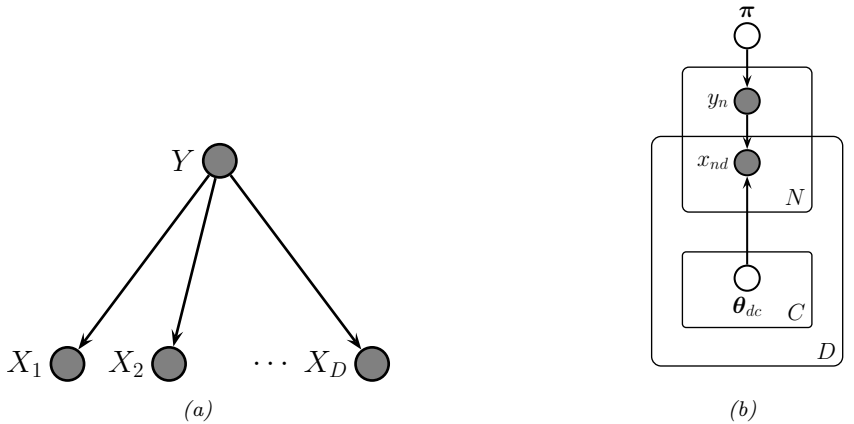


Figure 4.13: (a) Naive Bayes classifier as a DPGM. (b) Model augmented with plate notation.

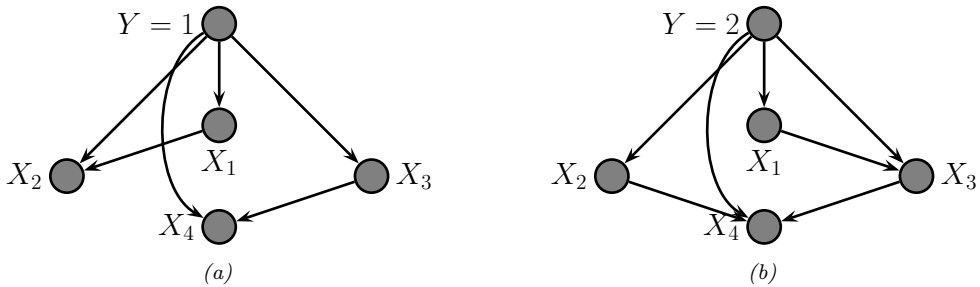


Figure 4.14: Tree-augmented naive Bayes classifier for $D = 4$ features. The tree topology can change depending on the value of y, as illustrated.

$\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\theta}_{1:D,1:C})$, we also need to represent the repetition over data cases n. This is shown in Figure 4.13b. Note that the parameter $\boldsymbol{\theta}_{dc}$ depends on d and c, whereas the feature $\boldsymbol{x}_{nd}$ depends on n and d. This is shown using **nested plates** to represent the shared d index.

4.2.8.3 Example: relaxing the naive Bayes assumption

We see from Figure 4.13a that the observed features are conditionally independent given the class label. We can of course allow for dependencies between the features, as illustrated in Figure 4.14. (We omit parameter nodes for simplicity.) If we enforce that the edges between the features forms a tree the model is known as a **tree-augmented naive Bayes classifier** [FGG97], or **TAN** model. (Trees are a restricted form of graphical that have various computational advantages that we discuss later.) Note that the topology of the tree can change depending on the value of the class node y; in this case, the model is known as a **Bayesian multi net**, and can be thought of as a supervised mixture of trees.

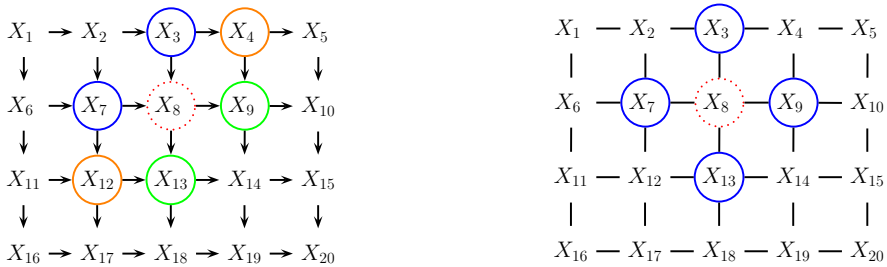


Figure 4.15: (a) A 2d lattice represented as a DAG. The dotted red node X_8 is independent of all other nodes (black) given its Markov blanket, which include its parents (blue), children (green) and co-parents (orange). (b) The same model represented as a UPGM. The red node X_8 is independent of the other black nodes given its neighbors (blue nodes).

4.3 Undirected graphical models (Markov random fields)

Directed graphical models (Section 4.2) are very useful. However, for some domains, being forced to choose a direction for the edges, as required by a DAG, is rather awkward. For example, consider modeling an image. It is reasonable to assume that the intensity values of neighboring pixels are correlated. We can model this using a DAG with a 2d lattice topology as shown in Figure 4.15(a). This is known as a **Markov mesh** [AHK65]. However, its conditional independence properties are rather unnatural.

An alternative is to use an undirected probabilistic graphical model (**UPGM**), also called a **Markov random field** (**MRF**) or **Markov network**. These do not require us to specify edge orientations, and are much more natural for some problems such as image analysis and spatial statistics. For example, an undirected 2d lattice is shown in Figure 4.15(b); now the Markov blanket of each node is just its nearest neighbors, as we show in Section 4.3.6.

Roughly speaking, the main advantages of UPGMs over DPGMs are: (1) they are symmetric and therefore more "natural" for certain domains, such as spatial or relational data; and (2) discriminative UPGMs (aka conditional random fields, or CRFs), which define conditional densities of the form $p(\boldsymbol{y}|\boldsymbol{x})$, work better than discriminative DGMs, for reasons we explain in Section 4.5.3. The main disadvantages of UPGMs compared to DPGMs are: (1) the parameters are less interpretable and less modular, for reasons we explain in Section 4.3.1; and (2) it is more computationally expensive to estimate the parameters, for reasons we explain in Section 4.3.9.1.

4.3.1 Representing the joint distribution

Since there is no topological ordering associated with an undirected graph, we can't use the chain rule to represent $p(\boldsymbol{x}_{1:N_G})$. So instead of associating CPDs with each node, we associate **potential functions** or **factors** with each **maximal clique** in the graph.[3] We will denote the potential

3. A **clique** is a set of nodes that are all neighbors of each other. A **maximal clique** is a clique which cannot be made any larger without losing the clique property.

function for clique c by $\psi_c(\boldsymbol{x}_c; \boldsymbol{\theta}_c)$, where $\boldsymbol{\theta}_c$ are its parameters. A potential function can be any non-negative function of its arguments (we give some examples below). We can use these functions to define the joint distribution as we explain in Section 4.3.1.1.

4.3.1.1 Hammersley-Clifford theorem

Suppose a joint distribution p satisfies the CI properties implied by the undirected graph G. (We discuss how to derive these properties in Section 4.3.6.) Then the **Hammersley-Clifford theorem** tells us that p can be written as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c; \boldsymbol{\theta}_c) \tag{4.69}$$

where $\mathcal{C}$ is the set of all the (maximal) cliques of the graph G, and $Z(\boldsymbol{\theta})$ is the **partition function** given by

$$Z(\boldsymbol{\theta}) \triangleq \sum_{\boldsymbol{x}} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c; \boldsymbol{\theta}_c) \tag{4.70}$$

Note that the partition function is what ensures the overall distribution sums to 1.[4]

The Hammersley-Clifford theorem was never published, but a proof can be found in [KF09a]. (Note that the theorem only holds for positive distributions, i.e., ones where $p(\boldsymbol{x}|\boldsymbol{\theta}) > 0$ for all configurations $\boldsymbol{x}$, which rules out some models with hard constraints.)

4.3.1.2 Gibbs distribution

The distribution in Equation (4.69) can be rewritten as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-\mathcal{E}(\boldsymbol{x}; \boldsymbol{\theta})) \tag{4.71}$$

where $\mathcal{E}(\boldsymbol{x}) > 0$ is the **energy** of state $\boldsymbol{x}$, defined by

$$\mathcal{E}(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_c \mathcal{E}(\boldsymbol{x}_c; \boldsymbol{\theta}_c) \tag{4.72}$$

where $\boldsymbol{x}_c$ are the variables in clique c. We can see the equivalence by defining the clique potentials as

$$\psi_c(\boldsymbol{x}_c; \boldsymbol{\theta}_c) = \exp(-\mathcal{E}(\boldsymbol{x}_c; \boldsymbol{\theta}_c)) \tag{4.73}$$

We see that low energy is associated with high probability states.

Equation (4.71) is known as the **Gibbs distribution**. This kind of probability model is also called an **energy-based model**. These are commonly used in physics and biochemistry. They are also used in ML to define generative models, as we discuss in Chapter 24. (See also Section 4.4, where we discuss conditional random fields (CRFs), which are models of the form $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$, where the potential functions are conditioned on input features $\boldsymbol{x}$.)

4. The partition function is denoted by Z because of the German word *Zustandssumme*, which means "sum over states". This reflects the fact that a lot of pioneering working on MRFs was done by German (and Austrian) physicists, such as Boltzmann.

4.3.2 Fully visible MRFs (Ising, Potts, Hopfield, etc)

In this section, we discuss some UPGMs for 2d grids, that are used in statistical physics and computer vision. We then discuss extensions to other graph structures, which are useful for biological modeling and pattern completion.

4.3.2.1 Ising models

Consider the 2d lattice in Figure 4.15(b). We can represent the joint distribution as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{i \sim j} \psi_{ij}(x_i, x_j; \boldsymbol{\theta}) \tag{4.74}$$

where $i \sim j$ means i and j are neighbors in the graph. This is called a **2d lattice model**.

An **Ising model** is a special case of the above, where the variables x_i are binary. Such models are often used to represent magnetic materials. In particular, each node represents an atom, which can have a magnetic dipole, or **spin**, which is in one of two states, $+1$ and -1. In some magnetic systems, neighboring spins like to be similar; in other systems, they like to be dissimilar. We can capture this interaction by defining the clique potentials as follows:

$$\psi_{ij}(x_i, x_j; \boldsymbol{\theta}) = \begin{cases} e^{J_{ij}} & \text{if } x_i = x_j \\ e^{-J_{ij}} & \text{if } x_i \neq x_j \end{cases} \tag{4.75}$$

where J_{ij} is the coupling strength between nodes i and j. This is known as the **Ising model**. If two nodes are not connected in the graph, we set $J_{ij} = 0$. We assume that the weight matrix is symmetric, so $J_{ij} = J_{ji}$. Often we also assume all edges have the same strength, so $J_{ij} = J$ for each (i, j) edge. Thus

$$\psi_{ij}(x_i, x_j; J) = \begin{cases} e^{J} & \text{if } x_i = x_j \\ e^{-J} & \text{if } x_i \neq x_j \end{cases} \tag{4.76}$$

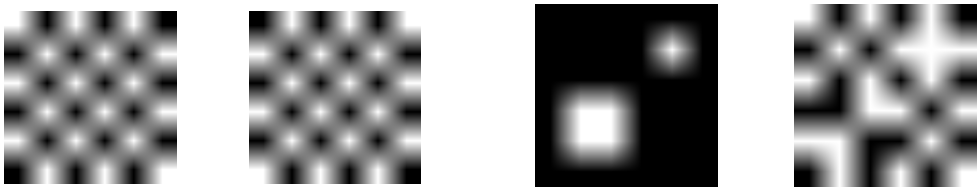
It is more common to define the Ising model as an energy-based model, as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(J)} \exp(-\mathcal{E}(\boldsymbol{x}; J)) \tag{4.77}$$

$$\mathcal{E}(\boldsymbol{x}; J) = -J \sum_{i \sim j} x_i x_j \tag{4.78}$$

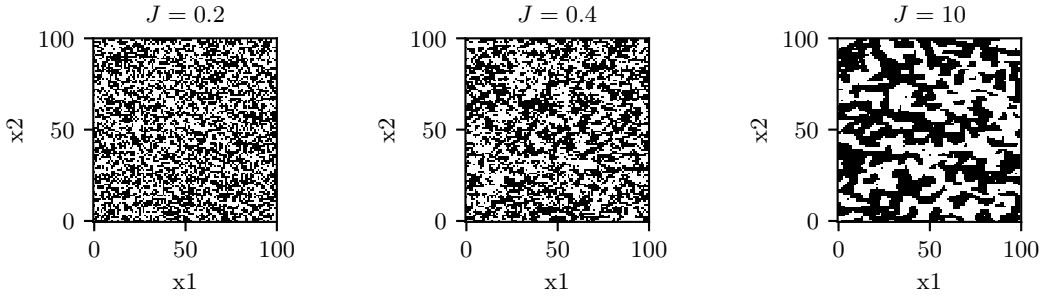
where $\mathcal{E}(\boldsymbol{x}; J)$ is the energy, and where we exploited the fact that $x_i x_j = -1$ if $x_i \neq x_j$, and $x_i x_j = +1$ if $x_i = x_j$. The magnitude of J controls the degree of coupling strength between neighboring sites, which depends on the (inverse) temperature of the system (colder = more tightly coupled = larger magnitude J).

If all the edge weights are positive, $J > 0$, then neighboring spins are likely to be in the same state, since if $x_i = x_j$, the energy term gets a contribution of $-J < 0$, and lower energy corresponds to higher probability. In the machine learning literature, this is called an **associative Markov network**. In the physics literature, this is called a **ferromagnetic** model. If the weights are



(a) (b)

Figure 4.16: (a) The two ground states for a small ferromagnetic Ising model where $J = 1$. (b) Two different states for a small Ising model which have the same energy. Left: $J = 1$, so neighboring pixels have similar values. Right: $J = -1$, so neighboring pixels have different values. From Figures 31.7 and 31.8 of [Mac03].



(a) (b) (c)

Figure 4.17: Samples from an associative Ising model with varying $J > 0$. Generated by gibbs_demo_ising.ipynb.

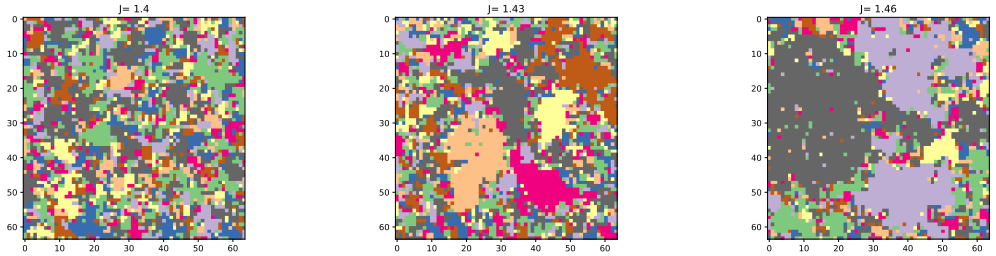
sufficiently strong, the corresponding probability distribution will have two modes, corresponding to the two checkerboard patterns in Figure 4.16a. These are called the **ground states** of the system.

If all of the weights are negative, $J < 0$, then the spins want to be different from their neighbors (see Figure 4.16b). This is called an **antiferromagnetic** system, and results in a **frustrated system**, since it is not possible for all neighbors to be different from each other in a 2d lattice. Thus the corresponding probability distribution will have multiple modes, corresponding to different "solutions" to the problem.

Figure 4.17 shows some samples from the Ising model for varying $J > 0$. (The samples were created using the Gibbs sampling method discussed in Section 12.3.3.) As the temperature reduces, the distribution becomes less entropic, and the "clumpiness" of the samples increases. One can show that, as the lattice size goes to infinity, there is a **critical temperature** J_c below which many large clusters occur, and above which many small clusters occur. In the case of an isotropic square lattice model, one can show [Geo88] that

$$J_c = \frac{1}{2}\log(1+\sqrt{2}) \approx 0.44 \tag{4.79}$$

This rapid change in global behavior as we vary a parameter of the system is called a **phase transition**. This can be used to explain how natural systems, such as water, can suddenly go from



(a) (b) (c)

Figure 4.18: Visualizing a sample from a 10-state Potts model of size 128×128. *The critical value is* $J_c = \log(1 + \sqrt{10}) = 1.426$. *for different association strengths: (a)* $J = 1.40$, *(b)* $J = 1.43$, *(c)* $J = 1.46$. *Generated by gibbs_ demo_ potts.ipynb.*

solid to liquid, or from liquid to gas, when the temperature changes slightly. See e.g., [Mac03, ch 31] for further details on the statistical mechanics of Ising models.

In addition to pairwise terms, it is standard to add **unary terms**, $\psi_i(x_i)$. In statistical physics, this is called an **external field**. The resulting model is as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_i \psi_i(x_i; \boldsymbol{\theta}) \prod_{i \sim j} \psi_{ij}(x_i, x_j; \boldsymbol{\theta}) \tag{4.80}$$

The ψ_i terms can be thought of as a local bias term, that are independent of the contributions of the neighboring nodes. For binary nodes, we can define this as follows:

$$\psi_i(x_i) = \begin{cases} e^{\alpha} & \text{if } x_i = +1 \\ e^{-\alpha} & \text{if } x_i = -1 \end{cases} \tag{4.81}$$

If we write this as an energy-based model, we have

$$\mathcal{E}(\boldsymbol{x}|\boldsymbol{\theta}) = -\alpha \sum_i x_i - J \sum_{i \sim j} x_i x_j \tag{4.82}$$

4.3.2.2 Potts models

In Section 4.3.2.1, we discussed the Ising model, which is a simple 2d MRF for defining distributions over binary variables. It is easy to generalize the Ising model to multiple discrete states, $x_i \in \{1, 2, \ldots, K\}$. if we use the same potential function for every edge, we can write

$$\psi_{ij}(x_i = k, x_j = k') = e^{J_{ij}(k,k')} \tag{4.83}$$

where $J_{ij}(k, k')$ is the energy if one node has state k and its neighbor has state k'. A common special case is

$$\psi_{ij}(x_i = k, x_j = k') = \begin{cases} e^{J} & \text{if } k = k' \\ e^{0} & \text{if } k \neq k' \end{cases} \tag{4.84}$$

This is called the **Potts model**. The Potts model reduces to the Ising model if we define $J_{\text{potts}} = 2J_{\text{ising}}$.

If $J > 0$, then neighboring nodes are encouraged to have the same label; this is an example of an associative Markov model. Some samples from this model are shown in Figure 4.18. The phase transition for a 2d Potts model occurs at the following value (see [MS96]):

$$J_c = \log(1 + \sqrt{K}) \tag{4.85}$$

We can extend this model to have local evidence for each node. If we write this as an energy-based model, we have

$$\mathcal{E}(\boldsymbol{x}|\boldsymbol{\theta}) = -\sum_i \sum_{k=1}^{K} \alpha_k \mathbb{I}(x_i = k) - J \sum_{i \sim j} \mathbb{I}(x_i = x_j) \tag{4.86}$$

4.3.2.3 Potts models for protein structure prediction

One interesting application of Potts models arises in the area of **protein structure prediction**. The goal is to predict the 3d shape of a protein from its 1d sequence of amino acids. A common approach to this is known as **direct coupling analysis** (DCA). We give a brief summary below; for details, see [Mor+11].

First we compute a **multiple sequence alignment** (MSA) from a set of related amino acid sequences from the same protein family; this can be done using HMMs, as explained in Section 29.3.2. The MSA can be represented by an $N \times T$ matrix $\mathbf{X}$, where N is the number of sequences, T is the length of each sequence, and $X_{ni} \in \{1, \ldots, V\}$ is the identity of the letter at location i in sequence n. For protein sequences, $V = 21$, representing the 20 amino acids plus the gap character.

Once we have the MSA matrix $\mathbf{X}$, we fit the Potts model using maximum likelihood estimation, or some approximation, such as pseudo likelihood [Eke+13]; see Section 4.3.9 for details.[5] After fitting the model, we select the edges with the highest J_{ij} coefficients, where $i, j \in \{1, \ldots, T\}$ are locations or **residues** in the protein. Since these locations are highly coupled, they are likely to be in physical contact, since interacting residues must coevolve to avoid destroying the function of the protein (see e.g., [LHF17] for a review). This graph is called a **contact map**.

Once the contact map is established, it can be used as input to a 3d structural prediction algorithm, such as [Xu18] or the **alphafold** system [Eva+18], which won the 2018 CASP competition. Such methods use neural networks to learn functions of the form $p(d(i,j)|\{c(i,j)\})$, where $d(i,j)$ is the 3d distance between residues i and j, and $c(i,j)$ is the contact map.

4.3.2.4 Hopfield networks

A **Hopfield network** [Hop82] is a fully connected Ising model (Section 4.3.2.1) with a symmetric weight matrix, $\mathbf{W} = \mathbf{W}^\mathsf{T}$. The corresponding energy function has the form

$$\mathcal{E}(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathbf{W}\boldsymbol{x} \tag{4.87}$$

5. To encourage the model to learn sparse connectivity, we can also compute a MAP estimate with a sparsity promoting prior, as discussed in [IM17].

where $x_i \in \{-1, +1\}$.

The main application of Hopfield networks is as an **associative memory** or **content addressable memory**. The idea is this: suppose we train on a set of fully observed bit vectors, corresponding to patterns we want to memorize. (We discuss how to do this below). Then, at test time, we present a partial pattern to the network. We would like to estimate the missing variables; this is called **pattern completion**. That is, we want to compute

$$\boldsymbol{x}^* = \operatorname*{argmin}_{\boldsymbol{x}} \mathcal{E}(\boldsymbol{x}) \tag{4.88}$$

We can solve this optimization problem using **iterative conditional modes** (**ICM**), in which we set each hidden variable to its most likely state given its neighbors. Picking the most probable state amounts to using the rule

$$\boldsymbol{x}^{t+1} = \text{sgn}(\mathbf{W}\boldsymbol{x}^t) \tag{4.89}$$

This can be seen as a deterministic version of Gibbs sampling (see Section 12.3.3).

We illustrate this process in Figure 4.19. In the top row, we show some training examples. In the middle row, we show a corrupted input, corresponding to the initial state $\boldsymbol{x}^0$. In the bottom row, we show the final state after 30 iterations of ICM. The overall process can be thought of as retrieving a complete example from memory based on a piece of the example.

To learn the weights $\mathbf{W}$, we could use the maximum likelihood estimate method described in Section 4.3.9.1. (See also [HSDK12].) However, a simpler heuristic method, proposed in [Hop82], is to use the following **outer product method**:

$$\mathbf{W} = \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\mathsf{T} \right) - \mathbf{I} \tag{4.90}$$

This normalizes the output product matrix by N, and then sets the diagonal to 0. This ensures the energy is low for patterns that match any of the examples in the training set. This is the technique we used in Figure 4.19. Note, however, that this method not only stores the original patterms but also their inverses, and other linear combinations. Consequently there is a limit to how many examples the model can store before they start to "collide" in the memory. Hopfield proved that, for random patterns, the network capacity is $\sim 0.14N$.

4.3.3 MRFs with latent variables (Boltzmann machines, etc)

In this section, we discuss MRFs which contain latent variables, as a way to represent high dimensional joint distributions in discrete spaces.

4.3.3.1 Vanilla Boltzmann machines

MRFs in which all the variables are visible are limited in their expressive power, since the only way to model correlation between the variables is by directly adding an edge. An alternative approach is to introduce latent variables. A **Boltzmann machine** [AHS85] is like an Ising model (Section 4.3.2.1) with latent variables. In addition, the graph structure can be arbitrary (not just a lattice), and the binary states are $x_i \in \{0, 1\}$ instead of $x_i \in \{-1, +1\}$. We usually partition the nodes into hidden nodes $\boldsymbol{z}$ and visible nodes $\boldsymbol{x}$, as shown in Figure 4.20(a).

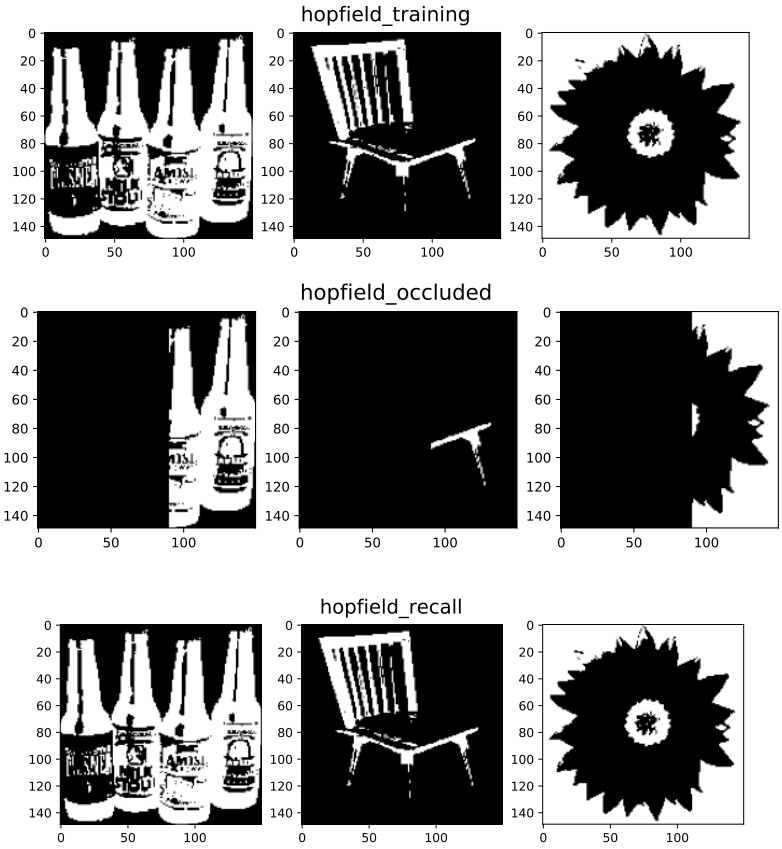
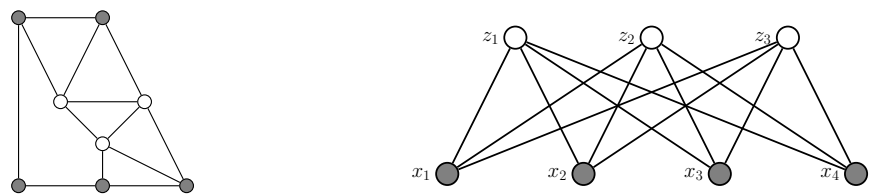


Figure 4.19: Examples of how an associative memory can reconstruct images. These are binary images of size 150×150 pixels. Top: training images. Middle row: partially visible test images. Bottom row: final state estimate. Adapted from Figure 2.1 of [HKP91]. Generated by hopfield_demo.ipynb.



(a) *(b)*

Figure 4.20: (a) A general Boltzmann machine, with an arbitrary graph structure. The shaded (visible) nodes are partitioned into input and output, although the model is actually symmetric and defines a joint distribution on all the nodes. (b) A restricted Boltzmann machine with a bipartite structure. Note the lack of intra-layer connections.

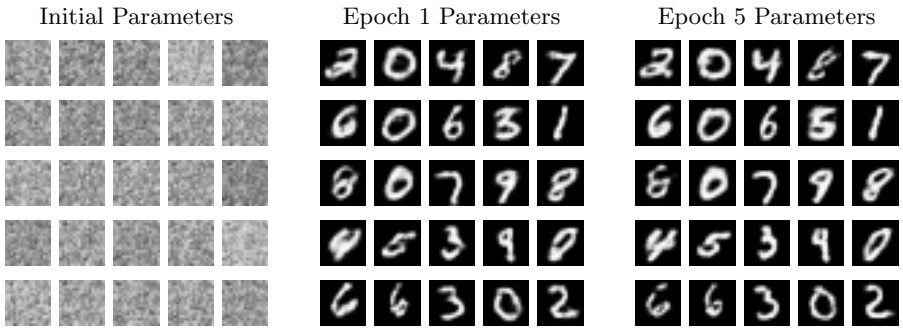


Figure 4.21: Some reconstructed images generated by a binary RBM fit to MNIST. Generated by rbm_ contrastive_ divergence.ipynb.

4.3.3.2 Restricted Boltzmann machines (RBMs)

Unfortunately, exact inference (and hence learning) in Boltzmann machines is intractable, and even approximate inference (e.g., Gibbs sampling, Section 12.3) can be slow. However, suppose we restrict the architecture so that the nodes are arranged in two layers, and so that there are no connections between nodes within the same layer (see Figure 4.20(b)). This model is known as a **restricted Boltzmann machine** (**RBM**) [HT01; HS06a], or a **harmonium** [Smo86]. The RBM supports efficient approximate inference, since the hidden nodes are conditionally independent given the visible nodes, i.e., $p(\boldsymbol{z}|\boldsymbol{x}) = \prod_{k=1}^{K} p(z_k|\boldsymbol{x})$. Note this is in contrast to a directed two-layer models, where the explaining away effect causes the latent variables to become "entangled" in the posterior even if they are independent in the prior.

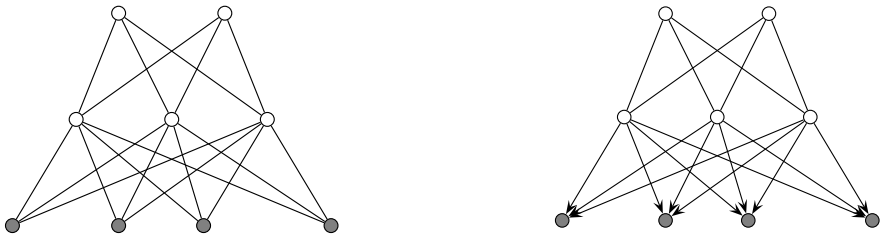
Typically the hidden and visible nodes in an RBM are binary, so the energy terms have the form $w_{dk} x_d z_k$. If $z_k = 1$, then the k'th hidden unit adds a term of the form $\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{x}$ to the energy; this can be thought of as a "soft constraint". If $z_k = 0$, the hidden unit is not active, and does not have an opinion about this data example. By turning on different combinations of constraints, we can create complex distributions on the visible data. This is an example of a **product of experts** (Section 24.1.1), since $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k:z_k=1} \exp(\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{x})$.

This can be thought of as a mixture model with an exponential number of hidden components, corresponding to 2^H settings of $\boldsymbol{z}$. That is, $\boldsymbol{z}$ is a **distributed representation**, whereas a standard mixture model uses a **localist representation**, where $z \in \{1, K\}$, and each setting of z corresponds to a complete prototype or exemplar $\boldsymbol{w}_k$ to which $\boldsymbol{x}$ is compared, giving rise to a model of the form $p(\boldsymbol{x}|z = k) \propto \exp(\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{x})$.

Many different kinds of RBMs have been defined, which use different pairwise potential functions. See Table 4.4 for a summary. (Figure 4.21 gives an example of some images generated from an RBM fit to the binarized MNIST dataset.) All of these are special cases of the **exponential family harmonium** [WRZH04]. See Supplementary Section 4.3 for more details.

Visible	Hidden	Name	Reference
Binary	Binary	Binary RBM	[HS06a]
Gaussian	Binary	Gaussian RBM	[WS05]
Categorical	Binary	Categorical RBM	[SMH07]
Multiple categorical	Binary	Replicated softmax/ undirected LDA	[SH10]
Gaussian	Gaussian	Undirected PCA	[MM01]
Binary	Gaussian	Undirected binary PCA	[WS05]

Table 4.4: Summary of different kinds of RBM.



(a) *(b)*

Figure 4.22: (a) Deep Boltzmann machine. (b) Deep belief network. The top two layers define the prior in terms on an RBM. The remaining layers are a directed graphical model that "decodes" the prior into observable data.

4.3.3.3 Deep Boltzmann machines

We can make a "deep" version of an RBM by stacking multiple layers; this is called a **deep Boltzmann machine** [SH09]. For example, the two layer model in Figure 4.22(a) has the form

$$p(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2|\boldsymbol{\theta}) = \frac{1}{Z(\mathbf{W}_1, \mathbf{W}_2)} \exp\left(\boldsymbol{x}^\mathsf{T}\mathbf{W}_1\boldsymbol{z}_1 + \boldsymbol{z}_1^\mathsf{T}\mathbf{W}_2\boldsymbol{z}_2\right) \tag{4.91}$$

where $\boldsymbol{x}$ are the visible nodes at the bottom, and we have dropped bias terms for brevity.

4.3.3.4 Deep belief networks (DBNs)

We can use an RBM as a prior over a latent distributed code, and then use a DPGM "decoder" to convert this into the observed data, as shown in Figure 4.22(b). The corresponding joint distribution has the form

$$p(\boldsymbol{x}, \boldsymbol{z}_1, \boldsymbol{z}_2|\boldsymbol{\theta}) = p(\boldsymbol{x}|\boldsymbol{z}_1, \mathbf{W}_1)\frac{1}{Z(\mathbf{W}_2)} \exp\left(\boldsymbol{z}_1^\mathsf{T}\mathbf{W}_2\boldsymbol{z}_2\right) \tag{4.92}$$

In other words, it is an RBM on top of a DPGM. This combination has been called a **deep belief network** (**DBN**) [HOT06a]. However, this name is confusing, since it is not actually a belief net. We will therefore call it a **deep Boltzmann network** (which conveniently has the same DBN abbreviation).

DBNs can be trained in a simple greedy fashion, and support fast bottom-up inference (see [HOT06a] for details). DBNs played an important role in the history of deep learning, since they were one of the first deep models that could be successfully trained. However, they are no longer widely used, since the advent of better ways to train fully supervised DNNs (such as using ReLU units and the Adam optimizer), and the advent of efficient ways to train deep DPGMs, such as the VAE (Section 21.2).

4.3.4 Maximum entropy models

In Section 2.3.7, we show that the exponential family is the distribution with maximum entropy, subject to the constraints that the expected value of the features (sufficient statistics) $\boldsymbol{\phi}(\boldsymbol{x})$ match the empirical expectations. Thus the model has the form

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\boldsymbol{\theta}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x})\right) \tag{4.93}$$

If the features $\boldsymbol{\phi}(\boldsymbol{x})$ decompose according to a graph structure, we get a kind of MRF known as a **maximum entropy model**. We give some examples below.

4.3.4.1 Log-linear models

Suppose the potential functions have the following log-linear form:

$$\psi_c(\boldsymbol{x}_c; \boldsymbol{\theta}_c) = \exp(\boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}_c)) \tag{4.94}$$

where $\boldsymbol{\phi}(\boldsymbol{x}_c)$ is a feature vector derived from the variables in clique c. Then the overall model is given by

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}_c)\right) \tag{4.95}$$

For example, in a Gaussian graphical model (GGM), we have

$$\boldsymbol{\phi}([x_i, x_j]) = [x_i, x_j, x_i x_j] \tag{4.96}$$

for $x_i \in \mathbb{R}$. And in an Ising model, we have

$$\boldsymbol{\phi}([x_i, x_j]) = [x_i, x_j, x_i x_j] \tag{4.97}$$

for $x_i \in \{-1, +1\}$. Thus both of these are maxent models. However, there are two key differences: first, in a GGM, the variables are real-valued, not binary; second, in a GGM, the partition function $Z(\boldsymbol{\theta})$ can be computed in $O(D^3)$ time, whereas in a Boltzmann machine, computing the partition function can take $O(2^D)$ time (see Section 9.4.4 for details).

If the features $\boldsymbol{\phi}$ are structured in a hierarchical way (capturing first order interactions, and second order interactions, etc.), and all the variables $\boldsymbol{x}$ are categorical, the resulting model is known in statistics as a **log-linear model**. However, in the ML community, the term "log-linear model" is often used to describe any model of the form Equation (4.95).

4.3.4.2 Feature induction for a maxent spelling model

In some applications, we assume the features $\boldsymbol{\phi}(\boldsymbol{x})$ are known. However, it is possible to learn the features in a maxent model in an unsupervised way; this is known as **feature induction**.

A common approach to feature induction, first proposed in [DDL97; ZWM97], is to start with a base set of features, and then to continually create new feature combinations out of old ones, greedily adding the best ones to the model.

As an example of this approach, [DDL97] describe how to build models to represent English spelling. This can be formalized as a probability distribution over variable length strings, $p(\boldsymbol{x}|\boldsymbol{\theta})$, where x_t is a letter in the English alphabet. Initially the model has no features, which represents the uniform distribution. The algorithm starts by choosing to add the feature

$$\phi_1(\boldsymbol{x}) = \sum_i \mathbb{I}\left(x_i \in \{a, \ldots, z\}\right) \tag{4.98}$$

which checks if any letter is lower case or not. After the feature is added, the parameters are (re)-fit by maximum likelihood (a computationally difficult problem, which we discuss in Section 4.3.9.1). For this feature, it turns out that $\hat{\theta}_1 = 1.944$, which means that a word with a lowercase letter in any position is about $e^{1.944} \approx 7$ times more likely than the same word without a lowercase letter in that position. Some samples from this model, generated using (annealed) Gibbs sampling (described in Section 12.3), are shown below.[6]

```
m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga, msmGh, pcp, d, oziVlal, hzagh, yzop, io,
advzmxnv, ijv_bolft, x, emx, kayerf, mlj, rawzyb, jp, ag, ctdnnnbg, wgdw, t, kguv, cy,
spxcq, uzflbbf, dxtkkn, cxwx, jpd, ztzh, lv, zhpkvnu, l^, r, qee, nynrx, atze4n, ik, se, w,
lrh, hp+, yrqyka'h, zcngotcnx, igcump, zjcjs, lqpWiqu, cefmfhc, o, lb, fdcY, tzby, yopxmvk,
by, fz,, t, govyccm, ijyiduwfzo, 6xr, duh, ejv, pk, pjw, l, fl, w
```

The second feature added by the algorithm checks if two adjacent characters are lower case:

$$\phi_2(\boldsymbol{x}) = \sum_{i \sim j} \mathbb{I}\left(x_i \in \{a, \ldots, z\}, x_j \in \{a, \ldots, z\}\right) \tag{4.99}$$

Now the model has the form

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp(\theta_1 \phi_1(\boldsymbol{x}) + \theta_2 \phi_2(\boldsymbol{x})) \tag{4.100}$$

Continuing in this way, the algorithm adds features for the strings `s>` and `ing>`, where `>` represents the end of word, and for various regular expressions such as `[0-9]`, etc. Some samples from the model with 1000 features, generated using (annealed) Gibbs sampling, are shown below.

```
was, reaser, in, there, to, will, ,, was, by, homes, thing, be, reloverated, ther, which,
conists, at, fores, anditing, with, Mr., proveral, the, ,, ***, on't, prolling, prothere, ,,
mento, at, yaou, 1, chestraing, for, have, to, intrally, of, qut, ., best, compers, ***,
cluseliment, uster, of, is, deveral, this, thise, of, offect, inatever, thifer,
constranded, stater, vill, in, thase, in, youse, menttering, and, ., of, in, verate, of,
to
```

6. We thank John Lafferty for sharing this example.

If we define a feature for every possible combination of letters, we can represent any probability distribution. However, this will overfit. The power of maxent approach is that we can choose which features matter for the domain.

An alternative approach is to introduce latent variables, that implicitly model correlations amongst the visible nodes, rather than explicitly having to learn feature functions. See Section 4.3.3 for an example of such a model.

4.3.5 Gaussian MRFs

In Section 4.2.3, we showed how to represent a multivariate Gaussian using a DPGM. In this section, we show how to represent a multivariate Gaussian using an UPGM. (For further details on GMRFs, see e.g., [RH05].)

4.3.5.1 Standard GMRFs

A **Gaussian graphical model** (or **GGM**), also called a **Gaussian MRF**, is a pairwise MRF of the following form:

$$p(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{i \sim j} \psi_{ij}(x_i, x_j) \prod_i \psi_i(x_i) \tag{4.101}$$

$$\psi_{ij}(x_i, x_j) = \exp(-\frac{1}{2} x_i \Lambda_{ij} x_j) \tag{4.102}$$

$$\psi_i(x_i) = \exp(-\frac{1}{2} \Lambda_{ii} x_i^2 + \eta_i x_i) \tag{4.103}$$

$$Z(\boldsymbol{\theta}) = (2\pi)^{D/2} |\boldsymbol{\Lambda}|^{-\frac{1}{2}} \tag{4.104}$$

The ψ_{ij} are **edge potentials** (pairwise terms), each the ψ_i are **node potentials** or **unary terms**. (We could absorb the unary terms into the pairwise terms, but we have kept them separate for clarity.)

The joint distribution can be rewritten in a more familiar form as follows:

$$p(\boldsymbol{x}) \propto \exp[\boldsymbol{\eta}^\mathsf{T} \boldsymbol{x} - \frac{1}{2} \boldsymbol{x}^\mathsf{T} \boldsymbol{\Lambda} \boldsymbol{x}] \tag{4.105}$$

This is called the **information form** of a Gaussian; $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$ are called the **canonical parameters**.

If $\Lambda_{ij} = 0$, there is no pairwise term connecting x_i and x_j, and hence $x_i \perp x_i | \boldsymbol{x}_{-ij}$, where $\boldsymbol{x}_{-ij}$ are all the nodes except for x_i and x_j. Hence the zero entries in $\boldsymbol{\Lambda}$ are called **structural zeros**. This means we can use ℓ_1 regularization on the weights to learn a sparse graph, a method known as **graphical lasso** (see Supplementary Section 30.3.2).

Note that the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1}$ can be dense even if the precision matrix $\boldsymbol{\Lambda}$ is sparse. For example, consider an AR(1) process with correlation parameter ρ.[7] The precision matrix (for a

7. This example is from https://dansblog.netlify.app/posts/2022-03-22-a-linear-mixed-effects-model/.

graph with $T = 7$ nodes) looks like this:

$$\boldsymbol{\Lambda} = \frac{1}{\tau^2}\begin{pmatrix} 1 & -\rho & & & & & \\ -\rho & 1+\rho^2 & -\rho & & & & \\ & -\rho & 1+\rho^2 & -\rho & & & \\ & & -\rho & 1+\rho^2 & -\rho & & \\ & & & -\rho & 1+\rho^2 & -\rho & \\ & & & & -\rho & 1+\rho^2 & -\rho \\ & & & & & -\rho & 1 \end{pmatrix} \tag{4.106}$$

But the covariance matrix is fully dense:

$$\boldsymbol{\Lambda}^{-1} = \tau^2 \begin{pmatrix} \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 & \rho^6 & \rho^7 \\ \rho^2 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 & \rho^6 \\ \rho^3 & \rho^2 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 \\ \rho^4 & \rho^3 & \rho^2 & \rho & \rho^2 & \rho^3 & \rho^4 \\ \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & \rho^2 & \rho^3 \\ \rho^6 & \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & \rho^2 \\ \rho^7 & \rho^6 & \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho \end{pmatrix} \tag{4.107}$$

This follows because, in a chain structured UPGM, every pair of nodes is marginally correlated, even if they may be conditionally independent given a separator.

4.3.5.2 Nonlinear Gaussian MRFs

In this section, we consider a generalization of GGMs to handle the case of nonlinear models. Suppose the joint is given by a product of local factors, or clique potentials, ψ_c, each of which is defined on a set or clique variables $\boldsymbol{x}_c$ as follows:

$$p(\boldsymbol{x}) = \frac{1}{Z}\prod_c \psi_c(\boldsymbol{x}_c) \tag{4.108}$$

$$\psi_c(\boldsymbol{x}_c) = \exp(-E_c(\boldsymbol{x}_c)) \tag{4.109}$$

$$E_c(\boldsymbol{x}_c) = \frac{1}{2}(f_c(\boldsymbol{x}_c) - \boldsymbol{d}_c)^\mathsf{T}\boldsymbol{\Sigma}_c^{-1}(f_c(\boldsymbol{x}_c) - \boldsymbol{d}_c) \tag{4.110}$$

where $\boldsymbol{d}_c$ is an optional local evidence term for the c'th clique, and f_c is some measurement function.

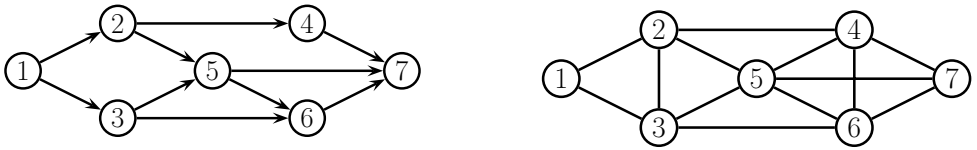
Suppose the measurent function f_c is linear, i.e.,

$$f_c(\boldsymbol{x}) = \mathbf{J}_c\boldsymbol{x} + \boldsymbol{b}_c \tag{4.111}$$

In this case, the energy for clique c becomes

$$E_c(\boldsymbol{x}_c) = \frac{1}{2}\boldsymbol{x}_c^\mathsf{T}\underbrace{\mathbf{J}_c^\mathsf{T}\boldsymbol{\Sigma}_c^{-1}\mathbf{J}_c}_{\boldsymbol{\Lambda}_c}\boldsymbol{x}_c + \boldsymbol{x}_c^\mathsf{T}\underbrace{\mathbf{J}_c^\mathsf{T}\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{b}_c - \boldsymbol{d}_c)}_{-\boldsymbol{\eta}_c} + \underbrace{\frac{1}{2}(\boldsymbol{b}_c - \boldsymbol{d}_c)\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{b}_c - \boldsymbol{d}_c)}_{k_c} \tag{4.112}$$

$$= \frac{1}{2}\boldsymbol{x}_c^\mathsf{T}\boldsymbol{\Lambda}_c\boldsymbol{x}_c - \boldsymbol{\eta}_c^\mathsf{T}\boldsymbol{x}_c + k_c \tag{4.113}$$



(a) *(b)*

Figure 4.23: (a) A DPGM. (b) Its moralized version, represented as a UPGM.

which is a standard Gaussian factor. If f_c is nonlinear, it is common to linearize the model around the current estimate $\boldsymbol{x}_c^0$ to get

$$f_c(\boldsymbol{x}_c) \approx f_c(\boldsymbol{x}_c^0) + \mathbf{J}_c(\boldsymbol{x}_c - \boldsymbol{x}_c^0) = \mathbf{J}_c\boldsymbol{x}_c + \underbrace{(f_c(\boldsymbol{x}_c^0) - \mathbf{J}_c\boldsymbol{x}_c^0)}_{\boldsymbol{b}_c} \tag{4.114}$$

where $\mathbf{J}_c$ is the Jacobian of $f_c(\boldsymbol{x}_c)$ wrt $\boldsymbol{x}_c$. This gives us a "temporary" Gaussian factor that we can use for inference. This process can be iterated for improved accuracy.

4.3.6 Conditional independence properties

In this section, we explain how UPGMs encode conditional independence assumptions.

4.3.6.1 Basic results

UPGMs define CI relationships via simple graph separation as follows: given 3 sets of nodes A, B, and C, we say $\mathbf{X}_A \perp_G \mathbf{X}_B|\mathbf{X}_C$ iff C separates A from B in the graph G. This means that, when we remove all the nodes in C, if there are no paths connecting any node in A to any node in B, then the CI property holds. This is called the **global Markov property** for UPGMs. For example, in Figure 4.23(b), we have that $\{X_1, X_2\} \perp \{X_6, X_7\}|\{X_3, X_4, X_5\}$.

The smallest set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's **Markov blanket**; we will denote this by $\text{mb}(t)$. Formally, the Markov blanket satisfies the following property:

$$t \perp \mathcal{V} \setminus \text{cl}(t)|\text{mb}(t) \tag{4.115}$$

where $\text{cl}(t) \triangleq \text{mb}(t) \cup \{t\}$ is the **closure** of node t, and $\mathcal{V} = \{1, \ldots, N_G\}$ is the set of all nodes. One can show that, in a UPGM, a node's Markov blanket is its set of immediate neighbors. This is called the **undirected local Markov property**. For example, in Figure 4.23(b), we have $\text{mb}(X_5) = \{X_2, X_3, X_4, X_6, X_7\}$.

From the local Markov property, we can easily see that two nodes are conditionally independent given the rest if there is no direct edge between them. This is called the **pairwise Markov property**. In symbols, this is written as

$$s \perp t|\mathcal{V} \setminus \{s, t\} \iff G_{st} = 0 \tag{4.116}$$

where $G_{st} = 0$ means there is no edge between s and t (so there is a 0 in the adjaceny matrix).

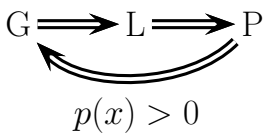


Figure 4.24: Relationship between Markov properties of UPGMs.

(a) *(b)*

Figure 4.25: (a) The ancestral graph induced by the DAG in Figure 4.23(a) wrt $U = \{X_2, X_4, X_5\}$. (b) The moralized version of (a).

Using the three Markov properties we have discussed, we can derive the following CI properties (amongst others) from the UPGM in Figure 4.23(b): $X_1 \perp X_7|\text{rest}$ (pairwise); $X_1 \perp \text{rest}|X_2, X_3$ (local); $X_1, X_2 \perp X_6, X_7|X_3, X_4, X_5$ (global).

It is obvious that global Markov implies local Markov which implies pairwise Markov. What is less obvious is that pairwise implies global, and hence that all these Markov properties are the same, as illustrated in Figure 4.24 (see e.g., [KF09a, p119] for a proof).[8] The importance of this result is that it is usually easier to empirically assess pairwise conditional independence; such pairwise CI statements can be used to construct a graph from which global CI statements can be extracted.

4.3.6.2 An undirected alternative to d-separation

We have seen that determinining CI relationships in UPGMs is much easier than in DPGMs, because we do not have to worry about the directionality of the edges. That is, we can use simple graph separation, instead of d-separation.

In this section, we show how to convert a DPGM to a UPGM, so that we can infer CI relationships for the DPGM using simple graph separation. It is tempting to simply convert the DPGM to a UPGM by dropping the orientation of the edges, but this is clearly incorrect, since a v-structure $A \rightarrow B \leftarrow C$ has quite different CI properties than the corresponding undirected chain $A - B - C$ (e.g., the latter graph incorrectly states that $A \perp C|B$). To avoid such incorrect CI statements, we can add edges between the "unmarried" parents A and C, and then drop the arrows from the

8. This assumes $p(\boldsymbol{x}) > 0$ for all $\boldsymbol{x}$, i.e., that p is a positive density. The restriction to positive densities arises because deterministic constraints can result in independencies present in the distribution that are not explicitly represented in the graph. See e.g., [KF09a, p120] for some examples. Distributions with non-graphical CI properties are said to be **unfaithful** to the graph, so $I(p) \neq I(G)$.

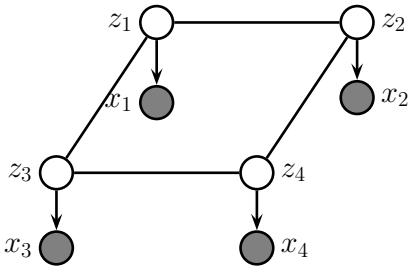


Figure 4.26: A grid-structured MRF with hidden nodes z_i and local evidence nodes x_i. The prior $p(\boldsymbol{z})$ is an undirected Ising model, and the likelihood $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_i p(x_i|z_i)$ is a directed fully factored model.

edges, forming (in this case) a fully connected undirected graph. This process is called **moralization**. Figure 4.23 gives a larger example of moralization: we interconnect 2 and 3, since they have a common child 5, and we interconnect 4, 5 and 6, since they have a common child 7.

Unfortunately, moralization loses some CI information, and therefore we cannot use the moralized UPGM to determine CI properties of the DPGM. For example, in Figure 4.23(a), using d-separation, we see that $X_4 \perp X_5|X_2$. Adding a moralization arc $X_4 - X_5$ would lose this fact (see Figure 4.23(b)). However, notice that the 4-5 moralization edge, due to the common child 7, is not needed if we do not observe 7 or any of its descendants. This suggests the following approach to determining if $A \perp B|C$. First we form the **ancestral graph** of DAG G with respect to $U = A \cup B \cup C$. This means we remove all nodes from G that are not in U or are not ancestors of U. We then moralize this ancestral graph, and apply the simple graph separation rules for UPGMs. For example, in Figure 4.25(a), we show the ancestral graph for Figure 4.23(a) using $U = \{X_2, X_4, X_5\}$. In Figure 4.25(b), we show the moralized version of this graph. It is clear that we now correctly conclude that $X_4 \perp X_5|X_2$.

4.3.7 Generation (sampling)

Unlike with DPGMs, it can be quite slow to sample from an UPGM, even from the unconditional prior, because there is no ordering of the variables. Furthermore, we cannot easily compute the probability of any configuration unless we know the value of Z. Consequently it is common to use MCMC methods for generating from an UPGM (see Chapter 12).

In the special case of UPGMs with low treewidth and discrete or Gaussian potentials, it is possible to use the junction tree algorithm to draw samples using dynamic programming (see Supplementary Section 9.2.3).

4.3.8 Inference

We discuss inference in graphical models in detail in Chapter 9. In this section, we just give an example.

Suppose we have an image composed of binary pixels, z_i, but we only observe noisy versions of the

(a) *(b)*

Figure 4.27: Example of image denoising using mean field variational inference. We use an Ising prior with $W_{ij} = 1$ and a Gaussian noise model with $\sigma = 2$. (a) Noisy image. (b) Result of inference. Generated by ising_image_denoise_demo.ipynb.

pixels, x_i. We assume the joint model has the form

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}) = \left[\frac{1}{Z}\sum_{i\sim j}\psi_{ij}(z_i, z_j)\right]\prod_i p(x_i|z_i) \tag{4.117}$$

where $p(\boldsymbol{z})$ is an Ising model prior, and $p(x_i|z_i) = \mathcal{N}(x_i|z_i, \sigma^2)$, for $z_i \in \{-1, +1\}$. This model uses a UPGM as a prior, and has directed edges for the likelihood, as shown in Figure 4.26; such a hybrid undirected-directed model is called a **chain graph** (even though it is not chain-structured).

The inference task is to compute the posterior marginals $p(z_i|\boldsymbol{x})$, or the posterior MAP estimate, $\text{argmax}_{\boldsymbol{z}}\, p(\boldsymbol{z}|\boldsymbol{x})$. The exact computation is intractable for large grids (for reasons explained in Section 9.4.4), so we must use approximate methods. There are many algorithms that we can use, including mean field variational inference (Section 10.2.2), Gibbs sampling (Section 12.3.3), loopy belief propagation (Section 9.3), etc. In Figure 4.27, we show the results of variational inference.

4.3.9 Learning

In this section, we discuss how to estimate the parameters for an MRF. As we will see, computing the MLE can be computationally expensive, even in the fully observed case, because of the need to deal with the partition function $Z(\boldsymbol{\theta})$. And computing the posterior over the parameters, $p(\boldsymbol{\theta}|\mathcal{D})$, is even harder, because of the additional normalizing constant $p(\mathcal{D})$ — this case has been called **doubly intractable** [MGM06]. Consequently we will focus on point estimation methods such as MLE and MAP. (For one approach to Bayesian parameter inference in an MRF, based on persistent variational inference, see [IM17].)

4.3.9.1 Learning from complete data

We will start by assuming there are no hidden variables or missing data during training (this is known as the **complete data** setting). For simplicity of presentation, we restrict our discusssion to the case of MRFs with log-linear potential functions. (See Section 24.2 for the general nonlinear case, where we discuss MLE for energy-based models.)

In particular, we assume the distribution has the following form:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\phi}_c(\boldsymbol{x})\right) \tag{4.118}$$

where c indexes the cliques. The (averaged) log-likelihood of the full dataset becomes

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \frac{1}{N} \sum_n \left[\sum_c \boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\phi}_c(\boldsymbol{x}_n) - \log Z(\boldsymbol{\theta})\right] \tag{4.119}$$

Its gradient is given by

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N} \sum_n \left[\boldsymbol{\phi}_c(\boldsymbol{x}_n) - \frac{\partial}{\partial \boldsymbol{\theta}_c} \log Z(\boldsymbol{\theta})\right] \tag{4.120}$$

We know from Section 2.3.3 that the derivative of the log partition function wrt $\boldsymbol{\theta}_c$ is the expectation of the c'th feature vector under the model, i.e.,

$$\frac{\partial \log Z(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_c} = \mathbb{E}\left[\boldsymbol{\phi}_c(\boldsymbol{x})|\boldsymbol{\theta}\right] = \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{\theta}) \boldsymbol{\phi}_c(\boldsymbol{x}) \tag{4.121}$$

Hence the gradient of the log likelihood is

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N} \sum_n \left[\boldsymbol{\phi}_c(\boldsymbol{x}_n)\right] - \mathbb{E}\left[\boldsymbol{\phi}_c(\boldsymbol{x})\right] \tag{4.122}$$

When the expected value of the features according to the data is equal to the expected value of the features according to the model, the gradient will be zero, so we get

$$\mathbb{E}_{p_\mathcal{D}}\left[\boldsymbol{\phi}_c(\boldsymbol{x})\right] = \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{\theta})}\left[\boldsymbol{\phi}_c(\boldsymbol{x})\right] \tag{4.123}$$

This is called **moment matching**. Evaluating the $\mathbb{E}_{p_\mathcal{D}}\left[\boldsymbol{\phi}_c(\boldsymbol{x})\right]$ term is called the **clamped phase** or **positive phase**, since $\boldsymbol{x}$ is set to the observed values $\boldsymbol{x}_n$; evaluating the $\mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{\theta})}\left[\boldsymbol{\phi}_c(\boldsymbol{x})\right]$ term is called the **unclamped phase** or **negative phase**, since $\boldsymbol{x}$ is free to vary, and is generated by the model.

In the case of MRFs with tabular potentials (i.e., one feature per entry in the clique table), we can use an algorithm called **iterative proportional fitting** or **IPF** [Fie70; BFH75; JP95] to solve these equations in an iterative fashion.[9] But in general, we must use gradient methods to perform parameter estimation.

9. In the case of decomposable graphs, IPF converges in a single iteration. Intuitively, this is because a decomposable graph can be converted to a DAG without any loss of information, as explained in Section 4.5, and we know that we can compute the MLE for tabular CPDs in closed form, just by normalizing the counts.

(a) *(b)*

Figure 4.28: (a) A small 2d lattice. (b) The representation used by pseudo likelihood. Solid nodes are observed neighbors. Adapted from Figure 2.2 of [Car03].

4.3.9.2 Computational issues

The biggest computational bottleneck in fitting MRFs and CRFs using MLE is the cost of computing the derivative of the log partition function, $\log Z(\boldsymbol{\theta})$, which is needed to compute the derivative of the log likelihood, as we saw in Section 4.3.9.1. To see why this is slow to compute, note that

$$\nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} Z(\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} = \frac{1}{Z(\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \int \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} = \frac{1}{Z(\boldsymbol{\theta})} \int \nabla_{\boldsymbol{\theta}} \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} \tag{4.124}$$

$$= \frac{1}{Z(\boldsymbol{\theta})} \int \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} = \int \frac{\tilde{p}(\boldsymbol{x};\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x} \tag{4.125}$$

$$= \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) \right] \tag{4.126}$$

where in Equation (4.125) we used the fact that $\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{\tilde{p}(\boldsymbol{x};\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \tilde{p}(\boldsymbol{x};\boldsymbol{\theta})$ (this is known as the **log-derivative trick**). Thus we see that we need to draw samples from the model at each step of SGD training, just to estimate the gradient

In Section 24.2.1, we discuss various efficient sampling methods. However, it is also possible to use alternative estimators which do not use the principle of maximum likelihood. For example, in Section 24.2.2 we discuss the technique of contrastive divergence. And in Section 4.3.9.3, we discuss the technique of pseudo likelihood. (See also [Sto17] for a review of many methods for parameter estimation in MRFs.)

4.3.9.3 Maximum pseudo-likelihood estimation

When fitting fully visible MRFs (or CRFs), a simple alternative to maximizing the likelihood is to maximize the **pseudo likelihood** [Bes75], defined as follows:

$$\ell_{PL}(\boldsymbol{\theta}) \triangleq \frac{1}{N_{\mathcal{D}}} \sum_{n=1}^{N_{\mathcal{D}}} \sum_{d=1}^{D} \log p(x_{nd} | \boldsymbol{x}_{n,-d}, \boldsymbol{\theta}) \tag{4.127}$$

That is, we optimize the product of the full conditionals, also known as the **composite likelihood** [Lin88a; DL10; VRF11]. Compare this to the objective for maximum likelihood:

$$\ell_{ML}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{D}}} \sum_{n=1}^{N_{\mathcal{D}}} \log p(\boldsymbol{x}_n | \boldsymbol{\theta}) \tag{4.128}$$

In the case of Gaussian MRFs, PL is equivalent to ML [Bes75], although this is not true in general. Nevertheless, it is a consistent estimator in the large sample limit [LJ08].

The PL approach is illustrated in Figure 4.28 for a 2d grid. We learn to predict each node, given all of its neighbors. This objective is generally fast to compute since each full conditional $p(x_d|\boldsymbol{x}_{-d}, \boldsymbol{\theta})$ only requires summing over the states of a single node, x_d, in order to compute the local normalization constant. The PL approach is similar to fitting each full conditional separately, except that, in PL, the parameters are tied between adjacent nodes.

Experiments in [PW05; HT09] suggest that PL works as well as exact ML for fully observed Ising models, but is much faster. In [Eke+13], they use PL to fitt Potts models to (aligned) protein sequence data. However, when fitting RBMs, [Mar+10] found that PL is worse than some of the stochastic ML methods we discuss in Section 24.2.

Another more subtle problem is that each node assumes that its neighbors have known values during training. If node $j \in \text{nbr}(i)$ is a perfect predictor for node i (where $\text{nbr}(i)$ is the set of neighbors), then j will learn to rely completely on node i, even at the expense of ignoring other potentially useful information, such as its local evidence, say y_i. At test time, the neighboring nodes will not be observed, and performance will suffer.[10]

4.3.9.4 Learning from incomplete data

In this section, we consider parameter estimation for MRFs (and CRFs) with hidden variables. Such **incomplete data** can arise for several reasons. For example, we may want to learn a model of the form $p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$ which lets us infer a "clean" image $\boldsymbol{z}$ from a noisy or corrupted version $\boldsymbol{x}$. If we only observe $\boldsymbol{x}$, the model is called a **hidden Gibbs random field**. See Section 10.2.2 for an example. As another example, we may have a CRF in which the hidden variables are used to encode an unknown alignment between the inputs and outputs [Qua+07], or to model missing parts of the input [SRS10].

We now discuss how to compute the MLE in such cases. For notational simplicity, we focus on unconditional models (MRFs, not CRFs), and we assume all the potentials are log-linear. In this case, the model has the following form:

$$p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{z}))}{Z(\boldsymbol{\theta})} = \frac{\tilde{p}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \tag{4.129}$$

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}, \boldsymbol{z}} \exp(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{z})) \tag{4.130}$$

where $\tilde{p}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$ is the unnormalized distribution. We have dropped the sum over cliques c for brevity.

10. Geoff Hinton has an analogy for this problem. Suppose we want to learn to denoise images of symmetric shapes, such as Greek vases. Each hidden pixel x_i depends on its spatial neighbors, as well the noisy observation y_i. Since its symmetric counterpart x_j will perfectly predict x_i, the model will ignore y_i and just rely on x_j, even though x_j will not be available at test time.

The log likelihood is now given by

$$\ell(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\log\left(\sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})\right) \tag{4.131}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{1}{Z(\boldsymbol{\theta})}\sum_{\boldsymbol{z}_n}\tilde{p}(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})\right) \tag{4.132}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left[\log\sum_{\boldsymbol{z}_n}\tilde{p}(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})\right] - \log Z(\boldsymbol{\theta}) \tag{4.133}$$

Note that

$$\log\sum_{\boldsymbol{z}_n}\tilde{p}(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) = \log\sum_{\boldsymbol{z}_n}\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{z}_n)) \triangleq \log Z(\boldsymbol{\theta}, \boldsymbol{x}_n) \tag{4.134}$$

where $Z(\boldsymbol{\theta}, \boldsymbol{x}_n)$ is the same as the partition function for the whole model, except that $\boldsymbol{x}$ is fixed at $\boldsymbol{x}_n$. Thus the log likelihood is a difference of two partition functions, one where $\boldsymbol{x}$ is clamped to $\boldsymbol{x}_n$ and $\boldsymbol{z}$ is unclamped, and one where both $\boldsymbol{x}$ and $\boldsymbol{z}$ are unclamped. The gradient of these log partition functions corresponds to the expected features, where (in the clamped case) we condition on $\boldsymbol{x} = \boldsymbol{x}_n$. Hence

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \frac{1}{N}\sum_{n}\left[\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}|\boldsymbol{x}_n,\boldsymbol{\theta})}\left[\boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{z})\right]\right] - \mathbb{E}_{(\boldsymbol{z},\boldsymbol{x})\sim p(\boldsymbol{z},\boldsymbol{x}|\boldsymbol{\theta})}\left[\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{z})\right] \tag{4.135}$$

4.4 Conditional random fields (CRFs)

A **conditional random field** or **CRF** [LMP01] is a Markov random field defined on a set of related label nodes $\boldsymbol{y}$, whose joint probability is predicted conditional on a fixed set of input nodes $\boldsymbol{x}$. More precisely, it corresponds to a model of the following form:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\theta})}\prod_{c}\psi_c(\boldsymbol{y}_c; \boldsymbol{x}, \boldsymbol{\theta}) \tag{4.136}$$

(Note how the partition function now depends on the inputs $\boldsymbol{x}$ as well as the parameters $\boldsymbol{\theta}$.) Now suppose the potential functions are log-linear and have the form

$$\psi_c(\boldsymbol{y}_c; \boldsymbol{x}, \boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{\phi}_c(\boldsymbol{x}, \boldsymbol{y}_c)) \tag{4.137}$$

This is a conditional version of the maxent models we discussed in Section 4.3.4. Of course, we can also use nonlinear potential functions, such as DNNs.

CRFs are useful because they capture dependencies amongst the output labels. They can therefore be used for **structured prediction**, where the output $\boldsymbol{y} \in \mathcal{Y}$ that we want to predict given the input $\boldsymbol{x}$ lives in some structured space, such as a sequence of labels, or labels associated with nodes on a graph. In such problems, there are often constraints on the set of valid values of the output $\boldsymbol{y}$. For example, if we want to perform sentence parsing, the output should satisfy the rules of grammar

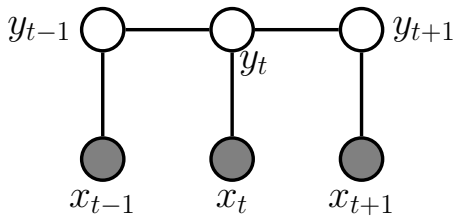


Figure 4.29: A 1d conditional random field (CRF) for sequence labeling.

(e.g., noun phrase must precede verb phrase). See Section 4.4.1 for details on the application of CRFs to NLP. In some cases, the "constraints" are "soft", rather than "hard". For example, if we want to associate a label with each pixel in an image (a task called semantic segmentation), we might want to "encourage" the label at one location to be the same as its neighbors, unless the visual input strongly suggests a change in semantic content at this location (e.g., at the edge of an object). See Section 4.4.2 for details on the applications of CRFs to computer vision tasks.

4.4.1 1d CRFs

In this section, we focus on 1d CRFs defined on chain-structured graphical models. The graphical model is shown in Figure 4.29. This defines a joint distribution over sequences, $\boldsymbol{y}_{1:T}$, given a set of inputs, $\boldsymbol{x}_{1:T}$, as follows:

$$p(\boldsymbol{y}_{1:T}|\boldsymbol{x},\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{\theta})} \prod_{t=1}^{T} \psi(y_t, \boldsymbol{x}_t; \boldsymbol{\theta}) \prod_{t=2}^{T} \psi(y_{t-1}, y_t; \boldsymbol{\theta}) \tag{4.138}$$

where $\psi(y_t, \boldsymbol{x}_t; \boldsymbol{\theta})$ are the node potentials and $\psi(y_t, y_{t+1}; \boldsymbol{\theta})$ are the edge potentials. (We have assumed that the edge potentials are independent of the input $\boldsymbol{x}$, but this assumption is not required.)

Note that one could also consider an alternative way to define this conditional distribution, by using a discriminative directed Markov chain:

$$p(\boldsymbol{y}_{1:T}|\boldsymbol{x},\boldsymbol{\theta}) = p(y_1|\boldsymbol{x}_1;\boldsymbol{\theta}) \prod_{t=2}^{T} p(y_t|y_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}) \tag{4.139}$$

This is called a **maximum entropy Markov model** [MFP00]. However, it suffers from a subtle flaw compared to the CRF. In particular, in the directed model, each conditional $p(y_t|y_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta})$, is **locally normalized**, whereas in the CRF, the model is **globally normalized** due to the $Z(\boldsymbol{x}, \boldsymbol{\theta})$ term. The latter allows information to propagate through the entire sequence, as we discuss in more detail in Section 4.5.3.

CRFs were widely used in the natural language processing (NLP) community in the 1980s–2010s (see e.g., [Smi11]), although recently they have been mostly replaced by RNNs and transformers (see e.g., [Gol17]). Fortunately, we can get the best of both words by combining CRFs with DNNs, which

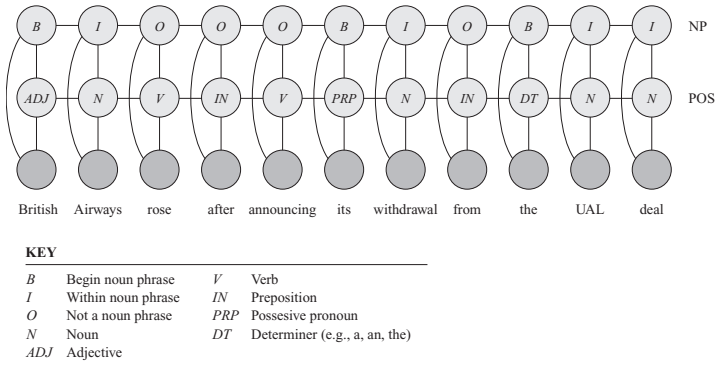


Figure 4.30: A CRF for joint part of speech tagging and noun phrase segmentation. From Figure 4.E.1 of [KF09a]. Used with kind permission of Daphne Koller.

allows us to combine data driven techniques with prior knowledge about constraints on the label space. We give some examples below.

4.4.1.1 Noun phrase chunking

A common task in NLP is **information extraction**, in which we try to parse a sentence into **noun phrases** (NP), such as names and addresses of people or businesses, as well as **verb phrases**, which describe who is doing what to whom (e.g., "British Airways rose"). In order to tackle this task, we can assign a **part of speech** tag to each word, where the tags correspond to Noun, Verb, Adjective, etc. In addition, to extract the span of each noun phrase, we can annotate words as being at the beginning (B) or inside (I) a noun phrase, or outside (O) of one. See Figure 4.30 for an example.

The connections between adjacent labels can encode constraints such as the fact that B (begin) must preceed I (inside). For example, the sequences `OBIIO` and `OBIOBIO` are valid (corresponding to one NP of 3 words, and two adjacent NPs of 2 words), but `OIBIO` is not. This prior information can be encoded by defining $\psi(y_{t-1}^{\text{BIO}} = *, y_t^{\text{BIO}} = B, x_t; \boldsymbol{\theta})$ to be 0 for any value of * except O. We can encode similar grammatical rules for the POS tags.

Given this model, we can compute the MAP sequence of labels, and thereby extract the spans that are labeled as noun phrases. This is called **noun phrase chunking**.

4.4.1.2 Named entity recognition

In this section we consider the task of **named entity extraction**, in which we not only tag the noun phrases, but also classify them into different types. A simple approach to this is to extend the BIO notation to {B-Per, I-Per, B-Loc, I-Loc, B-Org, I-Org, Other }. However, sometimes it is ambiguous whether a word is a person, location, or something else. Proper nouns are particularly difficult to deal with because they belong to an **open class**, that is, there is an unbounded number of possible names, unlike the set of nouns and verbs, which is large but essentially fixed. For example, "British Airways" is an organization, but "British Virgin Islands" is a location.

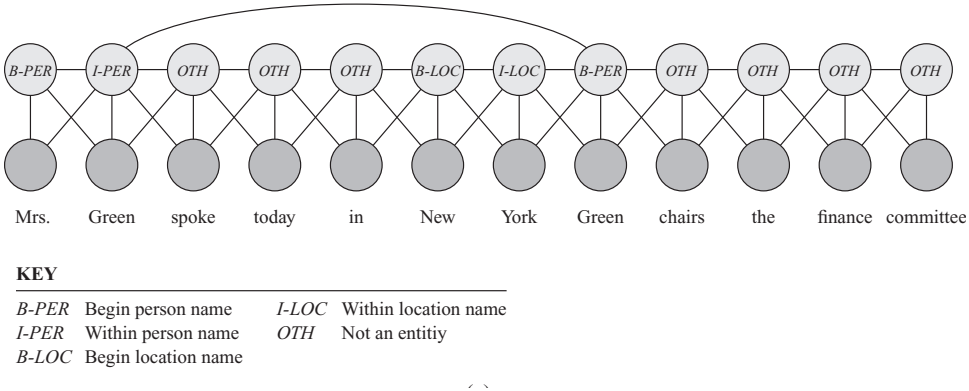


Figure 4.31: A skip-chain CRF for named entity recognition. From Figure 4.E.1 of [KF09a]. Used with kind permission of Daphne Koller.

We can get better performance by considering long-range correlations between words. For example, we might add a link between all occurrences of the same word, and force the word to have the same tag in each occurence. (The same technique can also be helpful for resolving the identity of pronouns.) This is known as a **skip-chain CRF**. See Figure 4.31 for an illustration, where we show that the word "Green" is interpeted as a person in both occurrences within the same sentence.

We see that the graph structure itself changes depending on the input, which is an additional advantage of CRFs over generative models. Unfortunately, inference in this model is generally more expensive than in a simple chain with local connections because of the larger treewdith (see Section 9.4.2).

4.4.1.3 Natural language parsing

A generalization of chain-structured models for language is to use probabilistic grammars. In particular, a probabilistic **context free grammar** or **PCFG** in Chomsky normal form is a set of re-write or production rules of the form $\sigma \rightarrow \sigma'\sigma''$ or $\sigma \rightarrow x$, where $\sigma, \sigma', \sigma'' \in \Sigma$ are non-terminals (analogous to parts of speech), and $x \in \mathcal{X}$ are terminals, i.e., words. Each such rule has an associated probability. The resulting model defines a probability distribution over sequences of words. We can compute the probability of observing a particular sequence $\boldsymbol{x} = x_1 \dots x_T$ by summing over all trees that generate it. This can be done in $O(T^3)$ time using the **inside-outside algorithm**; see e.g., [JM08; MS99; Eis16] for details.

PCFGs are generative models. It is possible to make discriminative versions which encode the probability of a labeled tree, $\boldsymbol{y}$, given a sequence of words, $\boldsymbol{x}$, by using a CRF of the form $p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp(\boldsymbol{w}^\mathsf{T} \Psi(\boldsymbol{x}, \boldsymbol{y}))$. For example, we might define $\Psi(\boldsymbol{x}, \boldsymbol{y})$ to count the number of times each production rule was used (which is analogous to the number of state transitions in a chain-structured model), as illustrated in Figure 4.32. We can also use a deep neural net to define the features, as in the **neural CRF parser** method of [DK15b].

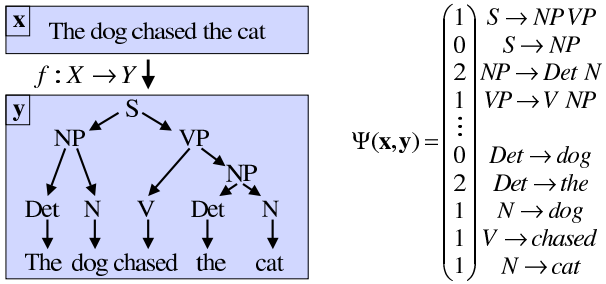


Figure 4.32: Illustration of a simple parse tree based on a context free grammar in Chomsky normal form. The feature vector $\Psi(\boldsymbol{x}, \boldsymbol{y})$ counts the number of times each production rule was used, and is used to define the energy of a particular tree structure, $E(\boldsymbol{y}|\boldsymbol{x}) = -\boldsymbol{w}^{\mathsf{T}}\Psi(\boldsymbol{x}, \boldsymbol{y})$. The probability distribution over trees is given by $p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp(-E(\boldsymbol{y}|\boldsymbol{x}))$. From Figure 5.2 of [AHT07]. Used with kind permission of Yasemin Altun.

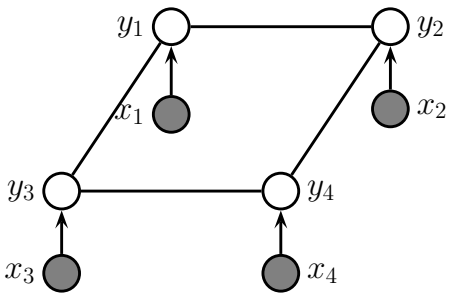


Figure 4.33: A grid-structured CRF with label nodes y_i and local evidence nodes x_i.

4.4.2 2d CRFs

It is also possible to apply CRFs to image processing problems, which are usually defined on 2d grids, as illustrated in Figure 4.33. (Compare this to the generative model in Figure 4.26.) This corresponds to the following conditional model:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\left[\sum_{i\sim j}\psi_{ij}(y_i, y_j)\right]\prod_i p(y_i|\boldsymbol{x}_i) \tag{4.140}$$

In the sections below, we discuss some applications of this and other CRF models in computer vision.

4.4.2.1 Semantic segmentation

The task of **semantic segmentation** is to assign a label to every pixel in an image. We can easily solve this problem using a CNN with one softmax output node per pixel. However, this may fail to capture long-range dependencies, since convolution is a local operation.

One way to get better results is to feed the output of the CNN into a CRF. Since the CNN already uses convolution, its outputs will usually already be locally smooth, so the benefits from using a CRF

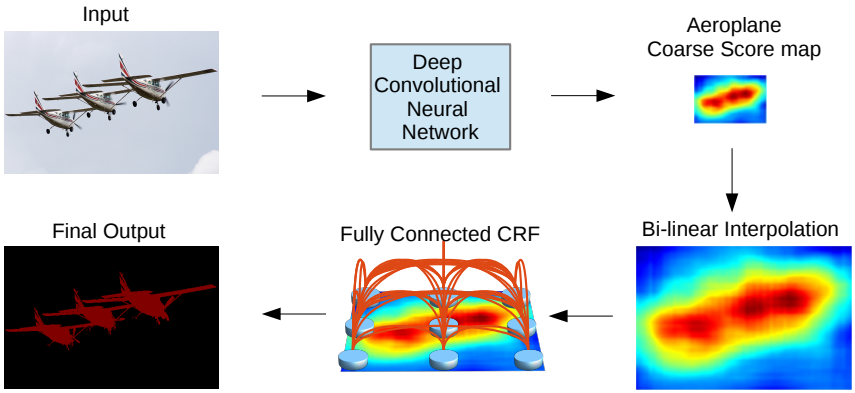


Figure 4.34: A fully connected CRF is added to the output of a CNN, in order to increase the sharpness of the segmentation boundaries. From Figure 3 of [Che+15]. Used with kind permission of Jay Chen.

with a local grid structure may be quite small. However, we can somtimes get better results if we use a **fully connected CRF**, which has connections between all the pixels. This can capture long range connections which the grid-structured CRF cannot. See Figure 4.34 for an illustration, and [Che+17a] for details.

Unfortunately, exact inference in a fully connected CRF is intractable, but in the case of Gaussian potentials, it is possible to devise an efficient mean field algorithm, as described in [KK11]. Interestingly, [Zhe+15] showed how the mean field update equations can be implemented using a recurrent neural network (see Section 16.3.4), allowing end-to-end training. Alternatively, if we are willing to use a finite number of iterations, we can just "unroll" the computation graph and treat it as a fixed-sized feedforward circuit. The result is a graph-structured neural network, where the topology of the GNN is derived from the graphical model (c.f., Section 9.3.10). The advantage of this compared to standard CRF methods is that we can train this entire model end-to-end using standard gradient descent methods; we no longer have to worry about the partition function (see Section 4.4.3), or the lack of convergence that can arise when combining approximate inference with standard CRF learning.

4.4.2.2 Deformable parts models

Consider the problem of **object detection**, i.e., finding the location(s) of an object of a given class (e.g., a person or a car) in an image. One way to tackle this is to train a binary classifier that takes as input an image patch and specifies if the patch contains the object or not. We can then apply this to every image patch, and return the locations where the classifier has high confidence detections; this is known as a **sliding window detector**, and works quite well for rigid objects such as cars or frontal faces. Such an approach can be made efficient by using convolutional neural networks (CNNs); see Section 16.3.2 for details.

However, such methods can work poorly when there is occlusion, or when the shape is deformable, such as a person's or animal's body, because there is too much variation in the overall appearance. A natural strategy to deal with such problems is break the object into parts, and then to detect

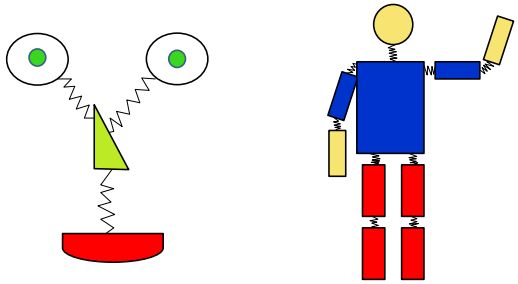


Figure 4.35: Pictorial structures model for a face and body. Each body part corresponds to a node in the CRF whose state space represents the location of that part. The edges (springs) represent pairwise spatial constraints. The local evidence nodes are not shown. Adapted from a figure by Pedro Felzenszwalb.

each part separately. But we still need to enforce spatial coherence of the parts. This can be done using a pairwise CRF, where node y_i specifies the location of part i in the image (assuming it is present), and where we connect adjacent parts by a potential function that encourages them to be close together. For example, we can use a pairwise potential of the form $\psi(y_i, y_j|\boldsymbol{x}) = \exp(-d(y_i, y_j))$, where $y_i \in \{1, \ldots, K\}$ is the location of part i (a discretization of the 2d image plane), and $d(y_i, y_i)$ is the distance between parts i and j. (We can make this "distance" also depend on the inputs $\boldsymbol{x}$ if we want, for example we may relax the distance penalty if we detect an edge.) In addition we will have a local evidence term of the form $p(y_i|\boldsymbol{x})$, which can be any kind of discriminative classifier, such as a CNN, which predicts the distribution over locations for part i given the image $\boldsymbol{x}$. The overall model has the form

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\left[\prod_i p(y_i|f(\boldsymbol{x})_i)\right]\left[\prod_{(i,j)\in E} \psi(y_i, y_j|\boldsymbol{x})\right] \tag{4.141}$$

where E is the set of edges in the CRF, and $f(\boldsymbol{x})_i$ is the i'th output of the CNN.

We can think of this CRF as a series of parts connected by springs, where the energy of the system increases if the parts are moved too far from their expected relative distance. This is illustrated in Figure 4.35. The resulting model is known as a **pictorial structure** [FE73], or **deformable parts model** [Fel+10]. Furthermore, since this is a conditional model, we can make the spring strengths be image dependent.

We can find the globally optimal joint configuration $\boldsymbol{y}^* = \text{argmax}_{\boldsymbol{y}}\, p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ using brute force enumeration in $O(K^T)$ time, where T is the number of nodes and K is the number of states (locations) per node. While T is often small, (e.g., just 10 body parts in Figure 4.35), K is often very large, since there are millions of possible locations in an image. By using tree-structured graphs, exact inference can be done in $O(TK^2)$ time, as we explain in Section 9.2.2. Furthermore, by exploiting the fact that the discrete states are ordinal, inference time can be further reduced to $O(TK)$, as explained in [Fel+10].

Note that by "augmenting" standard deep neural network libaries with a dynamic programming inference "module", we can represent DPMs as a kind of CNN, as shown in [Gir+15]. The key property is that we can backpropagate gradients through the inference algorithm.

4.4.3 Parameter estimation

In this section, we discuss how to perform maximum likelihood estimation for CRFs. This is a small extension of the MRF case in Section 4.3.9.1.

4.4.3.1 Log-linear potentials

In this section we assume the log potential functions are linear in the parameters, i.e.,

$$\psi_c(\boldsymbol{y}_c; \boldsymbol{x}, \boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\phi}_c(\boldsymbol{x}, \boldsymbol{y}_c)) \tag{4.142}$$

Hence the log likelihood becomes

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_n \log p(\boldsymbol{y}_n | \boldsymbol{x}_n, \boldsymbol{\theta}) = \frac{1}{N} \sum_n \left[\sum_c \boldsymbol{\theta}_c^\mathsf{T} \boldsymbol{\phi}_c(\boldsymbol{y}_{nc}, \boldsymbol{x}_n) - \log Z(\boldsymbol{x}_n; \boldsymbol{\theta}) \right] \tag{4.143}$$

where

$$Z(\boldsymbol{x}_n; \boldsymbol{\theta}) = \sum_{\boldsymbol{y}} \exp(\boldsymbol{\theta}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{y}, \boldsymbol{x}_n)) \tag{4.144}$$

is the partition function for example n.

We know from Section 2.3.3 that the derivative of the log partition function yields the expected sufficient statistics, so the gradient of the log likelihood can be written as follows:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N} \sum_n \left[\boldsymbol{\phi}_c(\boldsymbol{y}_{nc}, \boldsymbol{x}_n) - \frac{\partial}{\partial \boldsymbol{\theta}_c} \log Z(\boldsymbol{x}_n; \boldsymbol{\theta}) \right] \tag{4.145}$$

$$= \frac{1}{N} \sum_n \left[\boldsymbol{\phi}_c(\boldsymbol{y}_{nc}, \boldsymbol{x}_n) - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}_n, \boldsymbol{\theta})} \left[\boldsymbol{\phi}_c(\boldsymbol{y}, \boldsymbol{x}_n) \right] \right] \tag{4.146}$$

Since the objective is convex, we can use a variety of solvers to find the MLE, such as the stochastic meta descent method of [Vis+06], which is a variant of SGD where the stepsize is adapted automatically.

4.4.3.2 General case

In the general case, a CRF can be written as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp(f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}))}{Z(\boldsymbol{x}; \boldsymbol{\theta})} = \frac{\exp(f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}))}{\sum_{\boldsymbol{y}'} \exp(f(\boldsymbol{x}, \boldsymbol{y}'; \boldsymbol{\theta}))} \tag{4.147}$$

where $f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ is a scoring (negative energy) function, where high scores correspond to probable configurations. The gradient of the log likelihood is

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_n, \boldsymbol{y}_n; \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{x}_n; \boldsymbol{\theta}) \tag{4.148}$$

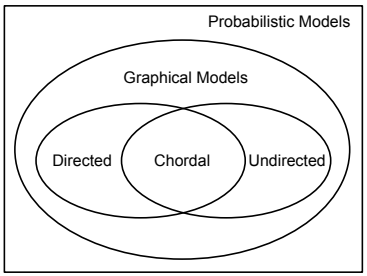


Figure 4.36: DPGMs and UPGMs can perfectly represent different sets of distributions. Some distributions can be perfectly represented by either DPGM's or UPGMs; the corresponding graph must be chordal.

Computing derivatives of the log partition function is tractable provided we can compute the corresponding expectations, as we discuss in Section 4.3.9.2. Note, however, that we need to compute these derivatives for every training example, which is slower than the MRF case, where the log partition function is a constant independent of the observed data (but dependent on the model parameters).

4.4.4 Other approaches to structured prediction

Many other approaches to structured prediction have been proposed, going beyond CRFs. For example, **max margin Markov networks** [TGK03], and the closely relayed **structural support vector machine** [Tso+05], can be seen as non-probabilistic alternatives to CRFs. More recently, [BYM17] proposed **Structured Prediction Energy Networks**, which are a form of energy based model (Chapter 24), where we predict using an optimization procedure, $\hat{\boldsymbol{y}}(\boldsymbol{x}) = \operatorname{argmin} \mathcal{E}(\boldsymbol{x}, \boldsymbol{y})$. In addition, it is common to use graph neural networks (Section 16.3.6) and sequence-to-sequence models such as transformers (Section 16.3.5) for this task.

4.5 Comparing directed and undirected PGMs

In this section, we compare DPGMs and UPGMs in terms of their modeling power, we discuss how to convert from one to the other, and we and present a unified representation.

4.5.1 CI properties

Which model has more "expressive power", a DPGM or a UPGM? To formalize the question, recall from Section 4.2.4 that G is an I-map of a distribution p if $I(G) \subseteq I(p)$, meaning that all the CI statements encoded by the graph G are true of the distribution p. Now define G to be **perfect map** of p if $I(G) = I(p)$, in other words, the graph can represent all (and only) the CI properties of the distribution. It turns out that DPGMs and UPGMs are perfect maps for different sets of distributions (see Figure 4.36). In this sense, neither is more powerful than the other as a representation language.

As an example of some CI relationships that can be perfectly modeled by a DPGM but not a UPGM, consider a v-structure $A \rightarrow C \leftarrow B$. This asserts that $A \perp B$, and $A \not\perp B|C$. If we drop