INTRODUCTION TO MACHINE LEARNING

Introduction to Machine Learning

Alex Smola and S.V.N. Vishwanathan

Yahoo! Labs
Santa Clara
–and–
Departments of Statistics and Computer Science
Purdue University
–and–
College of Engineering and Computer Science
Australian National University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011–4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2008

Printed in the United Kingdom at the University Press, Cambridge

Typeface Monotype Times 10/13pt *System* LaTeX 2ε [ALEXANDER J. SMOLA AND S.V.N. VISHWANATHAN]

A catalogue record for this book is available from the British Library

Library of Congress Cataloguing in Publication data available

ISBN 0 521 82583 0 hardback

AUTHOR: VISHY
REVISION: 256
TIMESTAMP: JANUARY 24, 2012
URL: SVN://SMOLA@REPOS.STAT.PURDUE.EDU/THEBOOK/TRUNK/BOOK/THEBOOK.TEX

Contents

1

Introduction

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress.

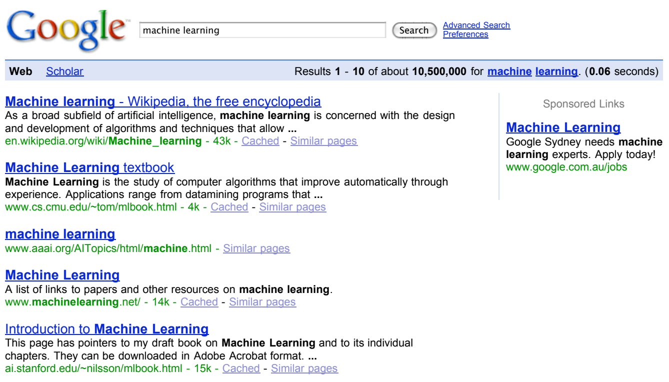
The purpose of this chapter is to provide the reader with an overview over the vast range of applications which have at their heart a machine learning problem and to bring some degree of order to the zoo of problems. After that, we will discuss some basic tools from statistics and probability theory, since they form the language in which many machine learning problems must be phrased to become amenable to solving. Finally, we will outline a set of fairly basic yet effective algorithms to solve an important problem, namely that of classification. More sophisticated tools, a discussion of more general problems and a detailed analysis will follow in later parts of the book.

1.1 A Taste of Machine Learning

Machine learning can appear in many guises. We now discuss a number of applications, the types of data they deal with, and finally, we formalize the problems in a somewhat more stylized fashion. The latter is key if we want to avoid reinventing the wheel for every new application. Instead, much of the *art* of machine learning is to reduce a range of fairly disparate problems to a set of fairly narrow prototypes. Much of the *science* of machine learning is then to solve those problems and provide good guarantees for the solutions.

1.1.1 Applications

Most readers will be familiar with the concept of web page **ranking**. That is, the process of submitting a query to a search engine, which then finds webpages relevant to the query and which returns them in their order of relevance. See e.g. Figure 1.1 for an example of the query results for “machine learning”. That is, the search engine returns a sorted list of webpages given a query. To achieve this goal, a search engine needs to ‘know’ which



Results **1** - **10** of about **10,500,000** for **machine** **learning**. (**0.06** seconds)

Machine learning - Wikipedia, the free encyclopedia
As a broad subfield of artificial intelligence, **machine learning** is concerned with the design and development of algorithms and techniques that allow **...**
en.wikipedia.org/wiki/**Machine_learning** - 43k - Cached - Similar pages

Machine Learning textbook
Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that **...**
www.cs.cmu.edu/~tom/mlbook.html - 4k - Cached - Similar pages

machine learning
www.aaai.org/AITopics/html/**machine**.html - Similar pages

Machine Learning
A list of links to papers and other resources on **machine learning**.
www.**machinelearning**.net/ - 14k - Cached - Similar pages

Introduction to **Machine Learning**
This page has pointers to my draft book on **Machine Learning** and to its individual chapters. They can be downloaded in Adobe Acrobat format. **...**
ai.stanford.edu/~nilsson/mlbook.html - 15k - Cached - Similar pages

Sponsored Links

Machine Learning
Google Sydney needs **machine learning** experts. Apply today!
www.google.com.au/jobs

Fig. 1.1. The 5 top scoring webpages for the query "machine learning"

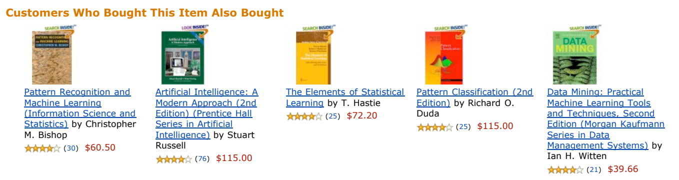
pages are relevant and which pages match the query. Such knowledge can be gained from several sources: the link structure of webpages, their content, the frequency with which users will follow the suggested links in a query, or from examples of queries in combination with manually ranked webpages. Increasingly machine learning rather than guesswork and clever engineering is used to *automate* the process of designing a good search engine [RPB06].

A rather related application is **collaborative filtering**. Internet bookstores such as Amazon, or video rental sites such as Netflix use this information extensively to entice users to purchase additional goods (or rent more movies). The problem is quite similar to the one of web page ranking. As before, we want to obtain a sorted list (in this case of articles). The key difference is that an explicit query is missing and instead we can only use past purchase and viewing decisions of the user to predict future viewing and purchase habits. The key side information here are the decisions made by *similar* users, hence the collaborative nature of the process. See Figure 1.2 for an example. It is clearly desirable to have an automatic system to solve this problem, thereby avoiding guesswork and time [BK07].

An equally ill-defined problem is that of **automatic translation** of documents. At one extreme, we could aim at fully *understanding* a text before translating it using a curated set of rules crafted by a computational linguist well versed in the two languages we would like to translate. This is a rather arduous task, in particular given that text is not always grammatically correct, nor is the document understanding part itself a trivial one. Instead, we could simply use *examples* of translated documents, such as the proceedings of the Canadian parliament or other multilingual entities (United Nations, European Union, Switzerland) to *learn* how to translate between the two languages. In other words, we could use examples of translations to learn how to translate. This machine learning approach proved quite successful [BPX+07].

Many security applications, e.g. for access control, use face recognition as one of its components. That is, given the photo (or video recording) of a person, recognize who this person is. In other words, the system needs to **classify** the faces into one of many categories (Alice, Bob, Charlie, . . .) or decide that it is an unknown face. A similar, yet conceptually quite different problem is that of verification. Here the goal is to verify whether the person in question is who he claims to be. Note that differently to before, this is now a yes/no question. To deal with different lighting conditions, facial expressions, whether a person is wearing glasses, hairstyle, etc., it is desirable to have a system which *learns* which features are relevant for identifying a person.

Another application where learning helps is the problem of **named entity recognition** (see Figure 1.4). That is, the problem of identifying entities, such as places, titles, names, actions, etc. from documents. Such steps are crucial in the automatic digestion and understanding of documents. Some modern e-mail clients, such as Apple's Mail.app nowadays ship with the ability to identify addresses in mails and filing them automatically in an address book. While systems using hand-crafted rules can lead to satisfactory results, it is far more efficient to use examples of marked-up documents to learn such dependencies automatically, in particular if we want to deploy our system in many languages. For instance, while 'bush' and 'rice'


Customers Who Bought This Item Also Bought

Pattern Recognition and Machine Learning (Information Science and Statistics) by Christopher M. Bishop (30) $60.50

Artificial Intelligence: A Modern Approach (2nd Edition) (Prentice Hall Series in Artificial Intelligence) by Stuart Russell (76) $115.00

The Elements of Statistical Learning by T. Hastie (25) $72.20

Pattern Classification (2nd Edition) by Richard O. Duda (25) $115.00

Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems) by Ian H. Witten (21) $39.66

Fig. 1.2. Books recommended by Amazon.com when viewing Tom Mitchell's Machine Learning Book [Mit97]. It is desirable for the vendor to recommend relevant books which a user might purchase.

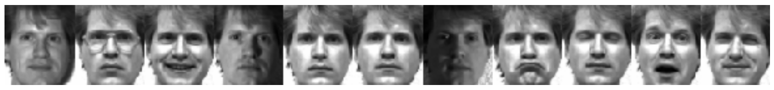


Fig. 1.3. 11 Pictures of the same person taken from the Yale face recognition database. The challenge is to recognize that we are dealing with the same person in all 11 cases.

```
HAVANA (Reuters) - The European Union's top development aid official
left Cuba on Sunday convinced that EU diplomatic sanctions against
the communist island should be dropped after Fidel Castro's
retirement, his main aide said.
```

```
<TYPE="ORGANIZATION">HAVANA</> (<TYPE="ORGANIZATION">Reuters</>) - The
<TYPE="ORGANIZATION">European Union</>'s top development aid official left
<TYPE="ORGANIZATION">Cuba</> on Sunday convinced that EU diplomatic sanctions
against the communist <TYPE="LOCATION">island</> should be dropped after
<TYPE="PERSON">Fidel Castro</>'s retirement, his main aide said.
```

Fig. 1.4. Named entity tagging of a news article (using LingPipe). The relevant locations, organizations and persons are tagged for further information extraction.

are clearly terms from agriculture, it is equally clear that in the context of contemporary politics they refer to members of the Republican Party.

Other applications which take advantage of learning are **speech recognition** (annotate an audio sequence with text, such as the system shipping with Microsoft Vista), the recognition of handwriting (annotate a sequence of strokes with text, a feature common to many PDAs), trackpads of computers (e.g. Synaptics, a major manufacturer of such pads derives its name from the synapses of a neural network), the detection of failure in jet engines, avatar behavior in computer games (e.g. Black and White), direct marketing (companies use past purchase behavior to guesstimate whether you might be willing to purchase even more) and floor cleaning robots (such as iRobot's Roomba). The overarching theme of learning problems is that there exists a nontrivial dependence between some observations, which we will commonly refer to as x and a desired response, which we refer to as y, for which a simple set of deterministic rules is not known. By using learning we can infer such a dependency between x and y in a systematic fashion.

We conclude this section by discussing the problem of **classification**, since it will serve as a prototypical problem for a significant part of this book. It occurs frequently in practice: for instance, when performing spam filtering, we are interested in a yes/no answer as to whether an e-mail contains relevant information or not. Note that this issue is quite user dependent: for a frequent traveller e-mails from an airline informing him about recent discounts might prove valuable information, whereas for many other recipients this might prove more of an nuisance (e.g. when the e-mail relates to products available only overseas). Moreover, the nature of annoying e-mails might change over time, e.g. through the availability of new products (Viagra, Cialis, Levitra, . . .), different opportunities for fraud (the Nigerian 419 scam which took a new twist after the Iraq war), or different data types (e.g. spam which consists mainly of images). To combat these problems we

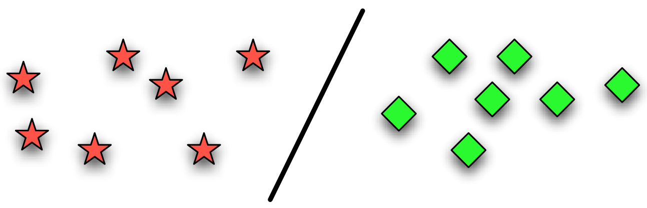


Fig. 1.5. Binary classification; separate stars from diamonds. In this example we are able to do so by drawing a straight line which separates both sets. We will see later that this is an important example of what is called a *linear classifier*.

want to build a system which is able to *learn* how to classify new e-mails. A seemingly unrelated problem, that of cancer diagnosis shares a common structure: given histological data (e.g. from a microarray analysis of a patient's tissue) infer whether a patient is healthy or not. Again, we are asked to generate a yes/no answer given a set of observations. See Figure 1.5 for an example.

1.1.2 Data

It is useful to characterize learning problems according to the type of data they use. This is a great help when encountering new challenges, since quite often problems on similar data types can be solved with very similar techniques. For instance natural language processing and bioinformatics use very similar tools for strings of natural language text and for DNA sequences. **Vectors** constitute the most basic entity we might encounter in our work. For instance, a life insurance company might be interesting in obtaining the vector of variables (blood pressure, heart rate, height, weight, cholesterol level, smoker, gender) to infer the life expectancy of a potential customer. A farmer might be interested in determining the ripeness of fruit based on (size, weight, spectral data). An engineer might want to find dependencies in (voltage, current) pairs. Likewise one might want to represent documents by a vector of counts which describe the occurrence of words. The latter is commonly referred to as bag of words features.

One of the challenges in dealing with vectors is that the *scales* and units of different coordinates may vary widely. For instance, we could measure the height in kilograms, pounds, grams, tons, stones, all of which would amount to multiplicative changes. Likewise, when representing temperatures, we have a full class of affine transformations, depending on whether we represent them in terms of Celsius, Kelvin or Farenheit. One way of dealing with those issues in an automatic fashion is to normalize the data. We will discuss means of doing so in an automatic fashion.

Lists: In some cases the vectors we obtain may contain a variable number of features. For instance, a physician might not necessarily decide to perform a full battery of diagnostic tests if the patient appears to be healthy.

Sets may appear in learning problems whenever there is a large number of potential causes of an effect, which are not well determined. For instance, it is relatively easy to obtain data concerning the toxicity of mushrooms. It would be desirable to use such data to infer the toxicity of a new mushroom given information about its chemical compounds. However, mushrooms contain a cocktail of compounds out of which one or more may be toxic. Consequently we need to infer the properties of an object given a *set* of features, whose composition and number may vary considerably.

Matrices are a convenient means of representing pairwise relationships. For instance, in collaborative filtering applications the rows of the matrix may represent users whereas the columns correspond to products. Only in some cases we will have knowledge about a given (user, product) combination, such as the rating of the product by a user.

A related situation occurs whenever we only have similarity information between observations, as implemented by a semi-empirical distance measure. Some homology searches in bioinformatics, e.g. variants of BLAST [AGML90], only return a similarity score which does not necessarily satisfy the requirements of a metric.

Images could be thought of as two dimensional arrays of numbers, that is, matrices. This representation is very crude, though, since they exhibit spatial coherence (lines, shapes) and (natural images exhibit) a multiresolution structure. That is, downsampling an image leads to an object which has very similar statistics to the original image. Computer vision and psychooptics have created a raft of tools for describing these phenomena.

Video adds a temporal dimension to images. Again, we could represent them as a three dimensional array. Good algorithms, however, take the temporal coherence of the image sequence into account.

Trees and Graphs are often used to describe relations between collections of objects. For instance the ontology of webpages of the DMOZ project (`www.dmoz.org`) has the form of a tree with topics becoming increasingly refined as we traverse from the root to one of the leaves (Arts $\rightarrow$ Animation $\rightarrow$ Anime $\rightarrow$ General Fan Pages $\rightarrow$ Official Sites). In the case of gene ontology the relationships form a directed acyclic graph, also referred to as the GO-DAG [ABB+00].

Both examples above describe estimation problems where our observations are vertices of a tree or graph. However, graphs themselves may be the observations. For instance, the DOM-tree of a webpage, the call-graph of a computer program, or the protein-protein interaction networks may form the basis upon which we may want to perform inference.

Strings occur frequently, mainly in the area of bioinformatics and natural language processing. They may be the input to our estimation problems, e.g. when classifying an e-mail as spam, when attempting to locate all names of persons and organizations in a text, or when modeling the topic structure of a document. Equally well they may constitute the output of a system. For instance, we may want to perform document summarization, automatic translation, or attempt to answer natural language queries.

Compound structures are the most commonly occurring object. That is, in most situations we will have a structured mix of different data types. For instance, a webpage might contain images, text, tables, which in turn contain numbers, and lists, all of which might constitute nodes on a graph of webpages linked among each other. Good statistical modelling takes such dependencies and structures into account in order to tailor sufficiently flexible models.

1.1.3 Problems

The range of learning problems is clearly large, as we saw when discussing applications. That said, researchers have identified an ever growing number of templates which can be used to address a large set of situations. It is those templates which make deployment of machine learning in practice easy and our discussion will largely focus on a choice set of such problems. We now give a by no means complete list of templates.

Binary Classification is probably the most frequently studied problem in machine learning and it has led to a large number of important algorithmic and theoretic developments over the past century. In its simplest form it reduces to the question: given a pattern x drawn from a domain $\mathcal{X}$, estimate which value an associated binary random variable $y \in \{\pm 1\}$ will assume. For instance, given pictures of apples and oranges, we might want to state whether the object in question is an apple or an orange. Equally well, we might want to predict whether a home owner might default on his loan, given income data, his credit history, or whether a given e-mail is spam or ham. The ability to solve this basic problem already allows us to address a large variety of practical settings.

There are many variants exist with regard to the protocol in which we are required to make our estimation:

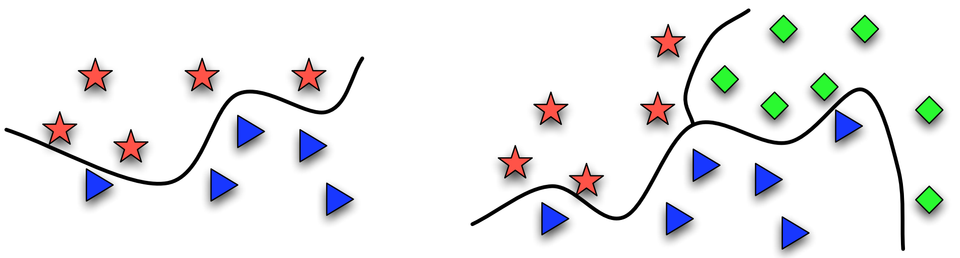


Fig. 1.6. Left: binary classification. Right: 3-class classification. Note that in the latter case we have much more degree for ambiguity. For instance, being able to distinguish stars from diamonds may not suffice to identify either of them correctly, since we also need to distinguish both of them from triangles.

- We might see a sequence of (x_i, y_i) pairs for which y_i needs to be estimated in an instantaneous online fashion. This is commonly referred to as online learning.
- We might observe a collection $\mathbf{X} := \{x_1, \dots x_m\}$ and $\mathbf{Y} := \{y_1, \dots y_m\}$ of pairs (x_i, y_i) which are then used to estimate y for a (set of) so-far unseen $\mathbf{X}' = \left\{x'_1, \dots, x'_{m'}\right\}$. This is commonly referred to as batch learning.
- We might be allowed to know $\mathbf{X}'$ already at the time of constructing the model. This is commonly referred to as transduction.
- We might be allowed to choose $\mathbf{X}$ for the purpose of model building. This is known as active learning.
- We might not have full information about $\mathbf{X}$, e.g. some of the coordinates of the x_i might be missing, leading to the problem of estimation with missing variables.
- The sets $\mathbf{X}$ and $\mathbf{X}'$ might come from different data sources, leading to the problem of covariate shift correction.
- We might be given observations stemming from two problems at the same time with the side information that both problems are somehow related. This is known as co-training.
- Mistakes of estimation might be penalized differently depending on the type of error, e.g. when trying to distinguish diamonds from rocks a very asymmetric loss applies.

Multiclass Classification is the logical extension of binary classification. The main difference is that now $y \in \{1, \dots, n\}$ may assume a range of different values. For instance, we might want to classify a document according to the language it was written in (English, French, German, Spanish, Hindi, Japanese, Chinese, . . .). See Figure 1.6 for an example. The main difference to before is that the cost of error may heavily depend on the type of

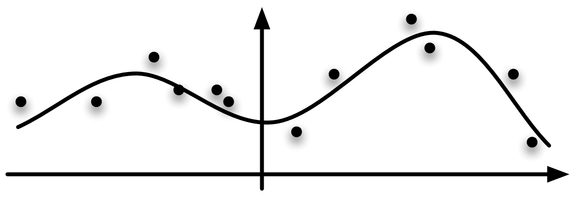


Fig. 1.7. Regression estimation. We are given a number of instances (indicated by black dots) and would like to find some function f mapping the observations $\mathcal{X}$ to $\mathbb{R}$ such that $f(x)$ is close to the observed values.

error we make. For instance, in the problem of assessing the risk of cancer, it makes a significant difference whether we mis-classify an early stage of cancer as healthy (in which case the patient is likely to die) or as an advanced stage of cancer (in which case the patient is likely to be inconvenienced from overly aggressive treatment).

Structured Estimation goes beyond simple multiclass estimation by assuming that the labels y have some additional structure which can be used in the estimation process. For instance, y might be a path in an ontology, when attempting to classify webpages, y might be a permutation, when attempting to match objects, to perform collaborative filtering, or to rank documents in a retrieval setting. Equally well, y might be an annotation of a text, when performing named entity recognition. Each of those problems has its own properties in terms of the set of y which we might consider admissible, or how to search this space. We will discuss a number of those problems in Chapter **??**.

Regression is another prototypical application. Here the goal is to estimate a real-valued variable $y \in \mathbb{R}$ given a pattern x (see e.g. Figure 1.7). For instance, we might want to estimate the value of a stock the next day, the yield of a semiconductor fab given the current process, the iron content of ore given mass spectroscopy measurements, or the heart rate of an athlete, given accelerometer data. One of the key issues in which regression problems differ from each other is the choice of a loss. For instance, when estimating stock values our loss for a put option will be decidedly one-sided. On the other hand, a hobby athlete might only care that our estimate of the heart rate matches the actual on average.

Novelty Detection is a rather ill-defined problem. It describes the issue of determining "unusual" observations given a set of past measurements. Clearly, the choice of what is to be considered unusual is very subjective. A commonly accepted notion is that unusual events occur rarely. Hence a possible goal is to design a system which assigns to each observation a rating

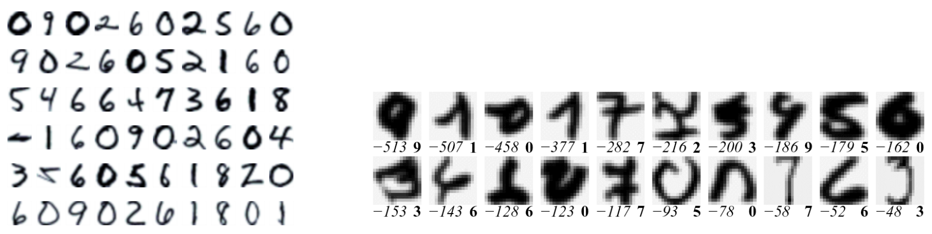


Fig. 1.8. Left: typical digits contained in the database of the US Postal Service. Right: unusual digits found by a novelty detection algorithm [SPST+01] (for a description of the algorithm see Section ??). The score below the digits indicates the degree of novelty. The numbers on the lower right indicate the class associated with the digit.

as to how novel it is. Readers familiar with density estimation might contend that the latter would be a reasonable solution. However, we neither need a score which sums up to 1 on the entire domain, nor do we care particularly much about novelty scores for *typical* observations. We will later see how this somewhat easier goal can be achieved directly. Figure 1.8 has an example of novelty detection when applied to an optical character recognition database.

1.2 Probability Theory

In order to deal with the instances of where machine learning can be used, we need to develop an adequate language which is able to describe the problems concisely. Below we begin with a fairly informal overview over probability theory. For more details and a very gentle and detailed discussion see the excellent book of [BT03b].

1.2.1 Random Variables

Assume that we cast a dice and we would like to know our chances whether we would see 1 rather than another digit. If the dice is fair all six outcomes $\mathcal{X} = \{1, \ldots, 6\}$ are equally likely to occur, hence we would see a 1 in roughly 1 out of 6 cases. Probability theory allows us to model uncertainty in the outcome of such experiments. Formally we state that 1 occurs with probability $\frac{1}{6}$.

In many experiments, such as the roll of a dice, the outcomes are of a numerical nature and we can handle them easily. In other cases, the outcomes may not be numerical, *e.g.,* if we toss a coin and observe heads or tails. In these cases, it is useful to associate numerical values to the outcomes. This is done via a random variable. For instance, we can let a random variable

X take on a value $+1$ whenever the coin lands heads and a value of -1 otherwise. Our notational convention will be to use uppercase letters, *e.g.,* X, Y etc to denote random variables and lower case letters, *e.g.,* x, y etc to denote the values they take.

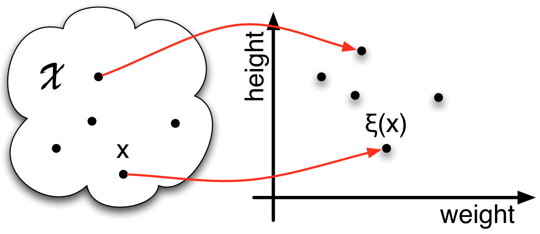


Fig. 1.9. The random variable ξ maps from the set of outcomes of an experiment (denoted here by $\mathcal{X}$) to real numbers. As an illustration here $\mathcal{X}$ consists of the patients a physician might encounter, and they are mapped via ξ to their weight and height.

1.2.2 Distributions

Perhaps the most important way to characterize a random variable is to associate probabilities with the values it can take. If the random variable is discrete, *i.e.,* it takes on a finite number of values, then this assignment of probabilities is called a *probability mass function* or PMF for short. A PMF must be, by definition, non-negative and must sum to one. For instance, if the coin is fair, *i.e.,* heads and tails are equally likely, then the random variable X described above takes on values of $+1$ and -1 with probability 0.5. This can be written as

$$Pr(X = +1) = 0.5 \text{ and } Pr(X = -1) = 0.5. \tag{1.1}$$

When there is no danger of confusion we will use the slightly informal notation $p(x) := Pr(X = x)$.

In case of a continuous random variable the assignment of probabilities results in a *probability density function* or PDF for short. With some abuse of terminology, but keeping in line with convention, we will often use density or distribution instead of probability density function. As in the case of the PMF, a PDF must also be non-negative and integrate to one. Figure 1.10 shows two distributions: the uniform distribution

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b] \\ 0 & \text{otherwise,} \end{cases} \tag{1.2}$$

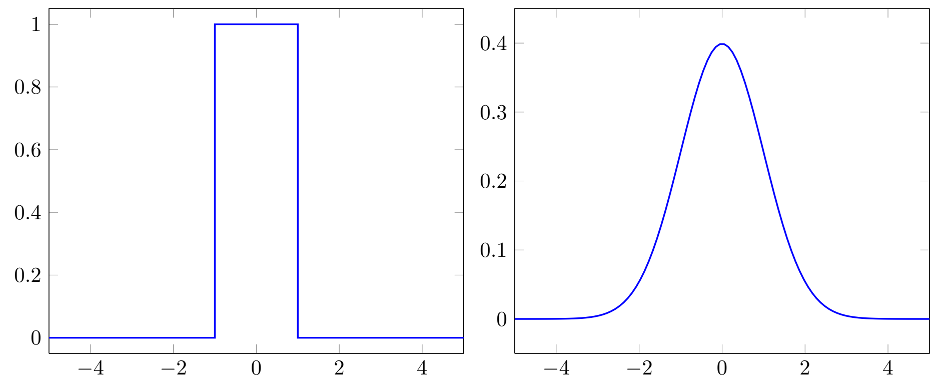


Fig. 1.10. Two common densities. Left: uniform distribution over the interval $[-1, 1]$. Right: Normal distribution with zero mean and unit variance.

and the Gaussian distribution (also called normal distribution)

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{1.3}$$

Closely associated with a PDF is the indefinite integral over p. It is commonly referred to as the cumulative distribution function (CDF).

Definition 1.1 (Cumulative Distribution Function) *For a real valued random variable X with PDF p the associated Cumulative Distribution Function F is given by*

$$F(x') := \Pr\left\{X \le x'\right\} = \int_{-\infty}^{x'} dp(x). \tag{1.4}$$

The CDF $F(x')$ allows us to perform range queries on p efficiently. For instance, by integral calculus we obtain

$$\Pr(a \le X \le b) = \int_a^b dp(x) = F(b) - F(a). \tag{1.5}$$

The values of x' for which $F(x')$ assumes a specific value, such as 0.1 or 0.5 have a special name. They are called the *quantiles* of the distribution p.

Definition 1.2 (Quantiles) *Let $q \in (0, 1)$. Then the value of x' for which $\Pr(X < x') \le q$ and $\Pr(X > x') \le 1 - q$ is the q-quantile of the distribution p. Moreover, the value x' associated with $q = 0.5$ is called the median.*

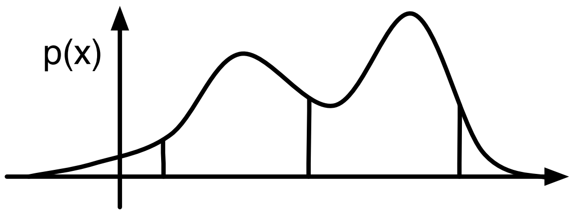


Fig. 1.11. Quantiles of a distribution correspond to the area under the integral of the density $p(x)$ for which the integral takes on a pre-specified value. Illustrated are the 0.1, 0.5 and 0.9 quantiles respectively.

1.2.3 Mean and Variance

A common question to ask about a random variable is what its expected value might be. For instance, when measuring the voltage of a device, we might ask what its typical values might be. When deciding whether to administer a growth hormone to a child a doctor might ask what a sensible range of height should be. For those purposes we need to define expectations and related quantities of distributions.

Definition 1.3 (Mean) *We define the mean of a random variable X as*

$$\mathbb{E}[X] := \int x dp(x) \tag{1.6}$$

More generally, if $f : \mathbb{R} \to \mathbb{R}$ is a function, then $f(X)$ is also a random variable. Its mean is mean given by

$$\mathbb{E}[f(X)] := \int f(x) dp(x). \tag{1.7}$$

Whenever X is a discrete random variable the integral in (1.6) can be replaced by a summation:

$$\mathbb{E}[X] = \sum_x x p(x). \tag{1.8}$$

For instance, in the case of a dice we have equal probabilities of $1/6$ for all 6 possible outcomes. It is easy to see that this translates into a mean of $(1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$.

The mean of a random variable is useful in assessing expected losses and benefits. For instance, as a stock broker we might be interested in the expected value of our investment in a year's time. In addition to that, however, we also might want to investigate the *risk* of our investment. That is, how likely it is that the value of the investment might deviate from its expectation since this might be more relevant for our decisions. This means that we need a variable to quantify the risk inherent in a random variable. One such measure is the *variance* of a random variable.

Definition 1.4 (Variance) *We define the variance of a random variable X as*

$$\mathrm{Var}[X] := \mathbb{E}\left[(X - \mathbf{E}[X])^2\right]. \tag{1.9}$$

As before, if $f : \mathbb{R} \to \mathbb{R}$ is a function, then the variance of $f(X)$ is given by

$$\mathrm{Var}[f(X)] := \mathbb{E}\left[(f(X) - \mathbf{E}[f(X)])^2\right]. \tag{1.10}$$

The variance measures by how much on average $f(X)$ deviates from its expected value. As we shall see in Section 2.1, an upper bound on the variance can be used to give guarantees on the probability that $f(X)$ will be within ϵ of its expected value. This is one of the reasons why the variance is often associated with the risk of a random variable. Note that often one discusses properties of a random variable in terms of its *standard deviation*, which is defined as the square root of the variance.

1.2.4 Marginalization, Independence, Conditioning, and Bayes Rule

Given two random variables X and Y, one can write their joint density $p(x, y)$. Given the joint density, one can recover $p(x)$ by integrating out y. This operation is called marginalization:

$$p(x) = \int_y dp(x, y). \tag{1.11}$$

If Y is a discrete random variable, then we can replace the integration with a summation:

$$p(x) = \sum_y p(x, y). \tag{1.12}$$

We say that X and Y are independent, *i.e.,* the values that X takes does not depend on the values that Y takes whenever

$$p(x, y) = p(x)p(y). \tag{1.13}$$

Independence is useful when it comes to dealing with large numbers of random variables whose behavior we want to estimate jointly. For instance, whenever we perform repeated measurements of a quantity, such as when measuring the voltage of a device, we will typically assume that the individual measurements are drawn from the same distribution and that they are

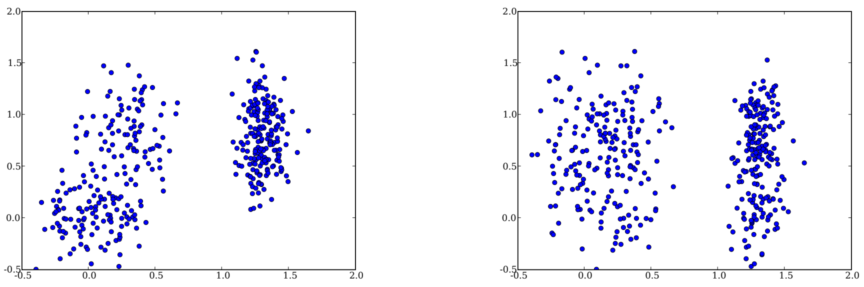


Fig. 1.12. Left: a sample from two dependent random variables. Knowing about first coordinate allows us to improve our guess about the second coordinate. Right: a sample drawn from two independent random variables, obtained by randomly permuting the dependent sample.

independent of each other. That is, having measured the voltage a number of times will not affect the value of the next measurement. We will call such random variables to be *independently and identically distributed*, or in short, *iid* random variables. See Figure 1.12 for an example of a pair of random variables drawn from dependent and independent distributions respectively.

Conversely, dependence can be vital in classification and regression problems. For instance, the traffic lights at an intersection are dependent of each other. This allows a driver to perform the inference that when the lights are green in his direction there will be no traffic crossing his path, i.e. the other lights will indeed be red. Likewise, whenever we are given a picture x of a digit, we hope that there will be dependence between x and its label y.

Especially in the case of dependent random variables, we are interested in conditional probabilities, *i.e.,* probability that X takes on a particular value given the value of Y. Clearly $Pr(X = rain|Y = cloudy)$ is higher than $Pr(X = rain|Y = sunny)$. In other words, knowledge about the value of Y significantly influences the distribution of X. This is captured via conditional probabilities:

$$p(x|y) := \frac{p(x, y)}{p(y)}. \tag{1.14}$$

Equation 1.14 leads to one of the key tools in statistical inference.

Theorem 1.5 (Bayes Rule) *Denote by X and Y random variables then the following holds*

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{1.15}$$

This follows from the fact that $p(x,y) = p(x|y)p(y) = p(y|x)p(x)$. The key consequence of (1.15) is that we may *reverse* the conditioning between a pair of random variables.

1.2.4.1 An Example

We illustrate our reasoning by means of a simple example — inference using an AIDS test. Assume that a patient would like to have such a test carried out on him. The physician recommends a test which is guaranteed to detect HIV-positive whenever a patient is infected. On the other hand, for healthy patients it has a 1% error rate. That is, with probability 0.01 it diagnoses a patient as HIV-positive even when he is, in fact, HIV-negative. Moreover, assume that 0.15% of the population is infected.

Now assume that the patient has the test carried out and the test returns 'HIV-negative'. In this case, logic implies that he is healthy, since the test has 100% detection rate. In the converse case things are not quite as straightforward. Denote by X and T the random variables associated with the health status of the patient and the outcome of the test respectively. We are interested in $p(X = \text{HIV+}|T = \text{HIV+})$. By Bayes rule we may write

$$p(X = \text{HIV+}|T = \text{HIV+}) = \frac{p(T = \text{HIV+}|X = \text{HIV+})p(X = \text{HIV+})}{p(T = \text{HIV+})}$$

While we know all terms in the numerator, $p(T = \text{HIV+})$ itself is unknown. That said, it can be computed via

$$\begin{aligned} p(T = \text{HIV+}) &= \sum_{x \in \{\text{HIV+},\text{HIV-}\}} p(T = \text{HIV+}, x) \\ &= \sum_{x \in \{\text{HIV+},\text{HIV-}\}} p(T = \text{HIV+}|x)p(x) \\ &= 1.0 \cdot 0.0015 + 0.01 \cdot 0.9985. \end{aligned}$$

Substituting back into the conditional expression yields

$$p(X = \text{HIV+}|T = \text{HIV+}) = \frac{1.0 \cdot 0.0015}{1.0 \cdot 0.0015 + 0.01 \cdot 0.9985} = 0.1306.$$

In other words, even though our test is quite reliable, there is such a low prior probability of having been infected with AIDS that there is not much evidence to accept the hypothesis even after this test.

Let us now think how we could improve the diagnosis. One way is to obtain further information about the patient and to use this in the diagnosis. For instance, information about his age is quite useful. Suppose the patient is 35 years old. In this case we would want to compute $p(X = \text{HIV+}|T =$

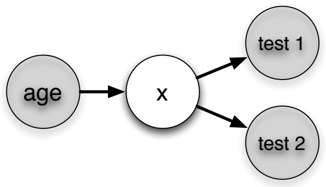


Fig. 1.13. A graphical description of our HIV testing scenario. Knowing the age of the patient influences our prior on whether the patient is HIV positive (the random variable X). The outcomes of the tests 1 and 2 are independent of each other given the status X. We observe the shaded random variables (age, test 1, test 2) and would like to infer the un-shaded random variable X. This is a special case of a graphical model which we will discuss in Chapter **??**.

HIV+, $A = 35$) where the random variable A denotes the age. The corresponding expression yields:

$$\frac{p(T = \text{HIV+}|X = \text{HIV+}, A)p(X = \text{HIV+}|A)}{p(T = \text{HIV+}|A)}$$

Here we simply *conditioned* all random variables on A in order to take additional information into account. We may assume that the test is *independent* of the age of the patient, i.e.

$$p(t|x, a) = p(t|x).$$

What remains therefore is $p(X = \text{HIV+}|A)$. Recent US census data pegs this number at approximately 0.9%. Plugging all data back into the conditional expression yields $\frac{1 \cdot 0.009}{1 \cdot 0.009 + 0.01 \cdot 0.991} = 0.48$. What has happened here is that by including additional observed random variables our estimate has become more reliable. Combination of evidence is a powerful tool. In our case it helped us make the classification problem of whether the patient is HIV-positive or not more reliable.

A second tool in our arsenal is the use of multiple measurements. After the first test the physician is likely to carry out a second test to confirm the diagnosis. We denote by T_1 and T_2 (and t_1, t_2 respectively) the two tests. Obviously, what we want is that T_2 will give us an "independent" second opinion of the situation. In other words, we want to ensure that T_2 does not make the same mistakes as T_1. For instance, it is probably a bad idea to repeat T_1 without changes, since it might perform the same diagnostic mistake as before. What we want is that the diagnosis of T_2 is independent of that of T_2 *given* the health status X of the patient. This is expressed as

$$p(t_1, t_2|x) = p(t_1|x)p(t_2|x). \tag{1.16}$$

See Figure 1.13 for a graphical illustration of the setting. Random variables satisfying the condition (1.16) are commonly referred to as *conditionally independent*. In shorthand we write $T_1, T_2 \perp\!\!\!\perp X$. For the sake of the argument we assume that the statistics for T_2 are given by

$p(t_2 \mid x)$	$x =$ HIV-	$x =$ HIV+
$t_2 =$ HIV-	0.95	0.01
$t_2 =$ HIV+	0.05	0.99

Clearly this test is less reliable than the first one. However, we may now combine both estimates to obtain a very reliable estimate based on the combination of both events. For instance, for $t_1 = t_2 =$ HIV+ we have

$$p(X = \text{HIV+} \mid T_1 = \text{HIV+}, T_2 = \text{HIV+}) = \frac{1.0 \cdot 0.99 \cdot 0.009}{1.0 \cdot 0.99 \cdot 0.009 + 0.01 \cdot 0.05 \cdot 0.991} = 0.95.$$

In other words, by combining two tests we can now confirm with very high confidence that the patient is indeed diseased. What we have carried out is a combination of evidence. Strong experimental evidence of two positive tests effectively overcame an initially very strong prior which suggested that the patient might be healthy.

Tests such as in the example we just discussed are fairly common. For instance, we might need to decide which manufacturing procedure is preferable, which choice of parameters will give better results in a regression estimator, or whether to administer a certain drug. Note that often our tests may not be conditionally independent and we would need to take this into account.

1.3 Basic Algorithms

We conclude our introduction to machine learning by discussing four simple algorithms, namely Naive Bayes, Nearest Neighbors, the Mean Classifier, and the Perceptron, which can be used to solve a binary classification problem such as that described in Figure 1.5. We will also introduce the K-means algorithm which can be employed when labeled data is not available. All these algorithms are readily usable and easily implemented from scratch in their most basic form.

For the sake of concreteness assume that we are interested in spam filtering. That is, we are given a set of m e-mails x_i, denoted by $\mathbf{X} := \{x_1, \ldots, x_m\}$ and associated labels y_i, denoted by $\mathbf{Y} := \{y_1, \ldots, y_m\}$. Here the labels satisfy $y_i \in \{\text{spam}, \text{ham}\}$. The key assumption we make here is that the pairs (x_i, y_i) are drawn jointly from some distribution $p(x, y)$ which represents the e-mail generating process for a user. Moreover, we assume that there

```
From: "LucindaParkison497072" <LucindaParkison497072@hotmail.com>
To: <kargr@earthlink.net>
Subject: we think ACGU is our next winner
Date: Mon, 25 Feb 2008 00:01:01 -0500
MIME-Version: 1.0
X-OriginalArrivalTime: 25 Feb 2008 05:01:01.0329 (UTC) FILETIME=[6A931810:01C8776B]
Return-Path: lucindaparkison497072@hotmail.com

(ACGU) .045 UP 104.5%

I do think that (ACGU) at it's current levels looks extremely attractive.

Asset Capital Group, Inc., (ACGU) announced that it is expanding the marketing of bio-remediation fluids and cleaning equipment. After
its recent acquisition of interest in American Bio-Clean Corporation and an 80

News is expected to be released next week on this growing company and could drive the price even higher. Buy (ACGU) Monday at open. I
believe those involved at this stage could enjoy a nice ride up.
```

Fig. 1.14. Example of a spam e-mail

x_1: `The quick brown fox jumped over the lazy dog.`
x_2: `The dog hunts a fox.`

	`the`	`quick`	`brown`	`fox`	`jumped`	`over`	`lazy`	`dog`	`hunts`	`a`
x_1	2	1	1	1	1	1	1	1	0	0
x_2	1	0	0	1	0	0	0	1	1	1

Fig. 1.15. Vector space representation of strings.

is sufficiently strong dependence between x and y that we will be able to estimate y given x and a set of labeled instances $\mathbf{X}, \mathbf{Y}$.

Before we do so we need to address the fact that e-mails such as Figure 1.14 are *text*, whereas the three algorithms we present will require data to be represented in a *vectorial* fashion. One way of converting text into a vector is by using the so-called *bag of words* representation [Mar61, Lew98]. In its simplest version it works as follows: Assume we have a list of all possible words occurring in $\mathbf{X}$, that is a dictionary, then we are able to assign a unique number with each of those words (e.g. the position in the dictionary). Now we may simply count for each document x_i the number of times a given word j is occurring. This is then used as the value of the j-th coordinate of x_i. Figure 1.15 gives an example of such a representation. Once we have the latter it is easy to compute distances, similarities, and other statistics directly from the vectorial representation.

1.3.1 Naive Bayes

In the example of the AIDS test we used the outcomes of the test to infer whether the patient is diseased. In the context of spam filtering the actual text of the e-mail x corresponds to the test and the label y is equivalent to the diagnosis. Recall Bayes Rule (1.15). We could use the latter to infer

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

We may have a good estimate of $p(y)$, that is, the probability of receiving a spam or ham mail. Denote by m_{ham} and m_{spam} the number of ham and spam e-mails in $\mathbf{X}$. In this case we can estimate

$$p(\text{ham}) \approx \frac{m_{\text{ham}}}{m} \text{ and } p(\text{spam}) \approx \frac{m_{\text{spam}}}{m}.$$

The key problem, however, is that we do not know $p(x|y)$ or $p(x)$. We may dispose of the requirement of knowing $p(x)$ by settling for a likelihood ratio

$$L(x) := \frac{p(\text{spam}|x)}{p(\text{ham}|x)} = \frac{p(x|\text{spam})p(\text{spam})}{p(x|\text{ham})p(\text{ham})}. \tag{1.17}$$

Whenever $L(x)$ exceeds a given threshold c we decide that x is spam and consequently reject the e-mail. If c is large then our algorithm is conservative and classifies an email as spam only if $p(\text{spam}|x) \gg p(\text{ham}|x)$. On the other hand, if c is small then the algorithm aggressively classifies emails as spam.

The key obstacle is that we have no access to $p(x|y)$. This is where we make our key approximation. Recall Figure 1.13. In order to model the distribution of the test outcomes T_1 and T_2 we made the assumption that they are conditionally independent of each other given the diagnosis. Analogously, we may now treat the occurrence of each word in a document as a separate test and combine the outcomes in a *naive* fashion by assuming that

$$p(x|y) = \prod_{j=1}^{\#\text{ of words in } x} p(w^j|y), \tag{1.18}$$

where w^j denotes the j-th word in document x. This amounts to the assumption that the probability of occurrence of a word in a document is independent of all other words given the category of the document. Even though this assumption does not hold in general – for instance, the word "York" is much more likely to after the word "New" – it suffices for our purposes (see Figure 1.16).

This assumption reduces the difficulty of knowing $p(x|y)$ to that of estimating the probabilities of occurrence of individual words w. Estimates for $p(w|y)$ can be obtained, for instance, by simply counting the frequency oc-

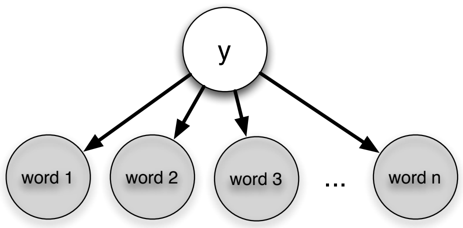


Fig. 1.16. Naive Bayes model. The occurrence of individual words is independent of each other, given the category of the text. For instance, the word *Viagra* is fairly frequent if $y =$ spam but it is considerably less frequent if $y =$ ham, except when considering the mailbox of a Pfizer sales representative.

currence of the word within documents of a given class. That is, we estimate

$$p(w|\text{spam}) \approx \frac{\sum_{i=1}^{m} \sum_{j=1}^{\#\text{ of words in } x_i} \left\{y_i = \text{spam and } w_i^j = w\right\}}{\sum_{i=1}^{m} \sum_{j=1}^{\#\text{ of words in } x_i} \{y_i = \text{spam}\}}$$

Here $\left\{y_i = \text{spam and } w_i^j = w\right\}$ equals 1 if and only if x_i is labeled as spam and w occurs as the j-th word in x_i. The denominator is simply the total number of words in spam documents. Similarly one can compute $p(w|\text{ham})$. In principle we could perform the above summation whenever we see a new document x. This would be terribly inefficient, since each such computation requires a full pass through $\mathbf{X}$ and $\mathbf{Y}$. Instead, we can perform a single pass through $\mathbf{X}$ and $\mathbf{Y}$ and store the resulting statistics as a good estimate of the conditional probabilities. Algorithm 1.1 has details of an implementation. Note that we performed a number of optimizations: Firstly, the normalization by m_{spam}^{-1} and m_{ham}^{-1} respectively is independent of x, hence we incorporate it as a fixed offset. Secondly, since we are computing a product over a large number of factors the numbers might lead to numerical overflow or underflow. This can be addressed by summing over the logarithm of terms rather than computing products. Thirdly, we need to address the issue of estimating $p(w|y)$ for words w which we might not have seen before. One way of dealing with this is to increment all counts by 1. This method is commonly referred to as Laplace smoothing. We will encounter a theoretical justification for this heuristic in Section 2.3.

This simple algorithm is known to perform surprisingly well, and variants of it can be found in most modern spam filters. It amounts to what is commonly known as "Bayesian spam filtering". Obviously, we may apply it to problems other than document categorization, too.

Algorithm 1.1 Naive Bayes

```
Train($\mathbf{X}, \mathbf{Y}$) {reads documents $\mathbf{X}$ and labels $\mathbf{Y}$}
  Compute dictionary $D$ of $\mathbf{X}$ with $n$ words.
  Compute $m, m_{\text{ham}}$ and $m_{\text{spam}}$.
  Initialize $b := \log c + \log m_{\text{ham}} - \log m_{\text{spam}}$ to offset the rejection threshold
  Initialize $p \in \mathbb{R}^{2 \times n}$ with $p_{ij} = 1$, $w_{\text{spam}} = n$, $w_{\text{ham}} = n$.
  {Count occurrence of each word}
  {Here $x_i^j$ denotes the number of times word $j$ occurs in document $x_i$}
  for $i = 1$ to $m$ do
    if $y_i =$ spam then
      for $j = 1$ to $n$ do
        $p_{0,j} \leftarrow p_{0,j} + x_i^j$
        $w_{\text{spam}} \leftarrow w_{\text{spam}} + x_i^j$
      end for
    else
      for $j = 1$ to $n$ do
        $p_{1,j} \leftarrow p_{1,j} + x_i^j$
        $w_{\text{ham}} \leftarrow w_{\text{ham}} + x_i^j$
      end for
    end if
  end for
  {Normalize counts to yield word probabilities}
  for $j = 1$ to $n$ do
    $p_{0,j} \leftarrow p_{0,j} / w_{\text{spam}}$
    $p_{1,j} \leftarrow p_{1,j} / w_{\text{ham}}$
  end for
Classify($x$) {classifies document $x$}
  Initialize score threshold $t = -b$
  for $j = 1$ to $n$ do
    $t \leftarrow t + x^j (\log p_{0,j} - \log p_{1,j})$
  end for
  if $t > 0$ return spam else return ham
```

1.3.2 Nearest Neighbor Estimators

An even simpler estimator than Naive Bayes is nearest neighbors. In its most basic form it assigns the label of its nearest neighbor to an observation x (see Figure 1.17). Hence, all we need to implement it is a distance measure $d(x, x')$ between pairs of observations. Note that this distance need not even be symmetric. This means that nearest neighbor classifiers can be extremely

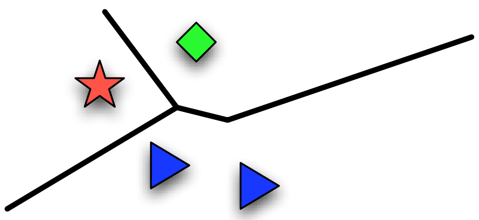


Fig. 1.17. 1 nearest neighbor classifier. Depending on whether the query point x is closest to the star, diamond or triangles, it uses one of the three labels for it.

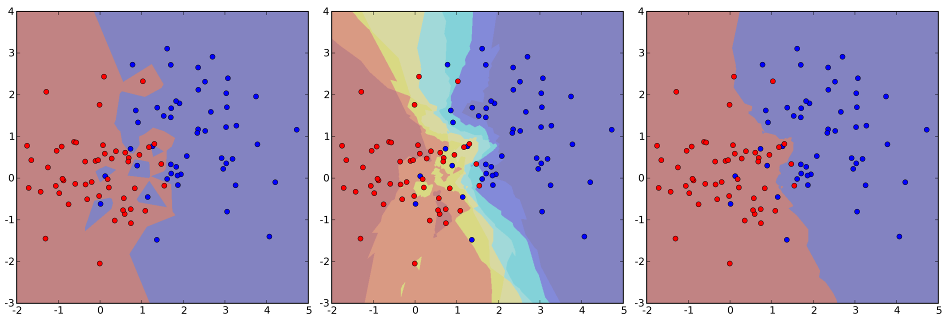


Fig. 1.18. k-Nearest neighbor classifiers using Euclidean distances. Left: decision boundaries obtained from a 1-nearest neighbor classifier. Middle: color-coded sets of where the number of red / blue points ranges between 7 and 0. Right: decision boundary determining where the blue or red dots are in the majority.

flexible. For instance, we could use string edit distances to compare two documents or information theory based measures.

However, the problem with nearest neighbor classification is that the estimates can be very noisy whenever the data itself is very noisy. For instance, if a spam email is erroneously labeled as nonspam then all emails which are similar to this email will share the same fate. See Figure 1.18 for an example. In this case it is beneficial to pool together a number of neighbors, say the k-nearest neighbors of x and use a majority vote to decide the class membership of x. Algorithm 1.2 has a description of the algorithm. Note that nearest neighbor algorithms can yield excellent performance when used with a good distance measure. For instance, the technology underlying the Netflix progress prize [BK07] was essentially nearest neighbours based.

Note that it is trivial to extend the algorithm to regression. All we need to change in Algorithm 1.2 is to return the average of the values y_i instead of their majority vote. Figure 1.19 has an example.

Note that the distance computation $d(x_i, x)$ for all observations can be-

Algorithm 1.2 k-Nearest Neighbor Classification

```
Classify($\mathbf{X}, \mathbf{Y}, x$) {reads documents $\mathbf{X}$, labels $\mathbf{Y}$ and query $x$}
  for $i = 1$ to $m$ do
    Compute distance $d(x_i, x)$
  end for
  Compute set $I$ containing indices for the $k$ smallest distances $d(x_i, x)$.
  return majority label of $\{y_i$ where $i \in I\}$.
```

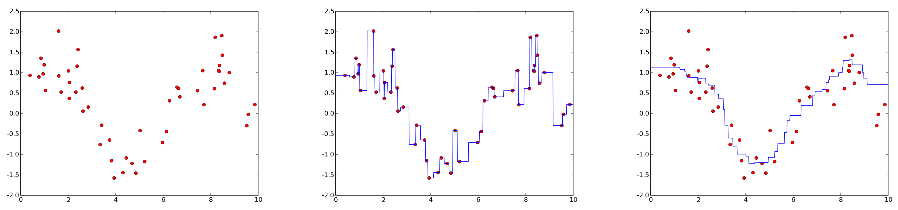


Fig. 1.19. k-Nearest neighbor regression estimator using Euclidean distances. Left: some points (x, y) drawn from a joint distribution. Middle: 1-nearest neighbour classifier. Right: 7-nearest neighbour classifier. Note that the regression estimate is much more smooth.

come extremely costly, in particular whenever the number of observations is large or whenever the observations x_i live in a very high dimensional space.

Random projections are a technique that can alleviate the high computational cost of Nearest Neighbor classifiers. A celebrated lemma by Johnson and Lindenstrauss [DG03] asserts that a set of m points in high dimensional Euclidean space can be projected into a $O(\log m/\epsilon^2)$ dimensional Euclidean space such that the distance between any two points changes only by a factor of $(1 \pm \epsilon)$. Since Euclidean distances are preserved, running the Nearest Neighbor classifier on this mapped data yields the same results but at a lower computational cost [GIM99].

The surprising fact is that the projection relies on a simple randomized algorithm: to obtain a d-dimensional representation of n-dimensional random observations we pick a matrix $R \in \mathbb{R}^{d \times n}$ where each element is drawn independently from a normal distribution with $n^{-\frac{1}{2}}$ variance and zero mean. Multiplying x with this projection matrix can be shown to achieve this property with high probability. For details see [DG03].

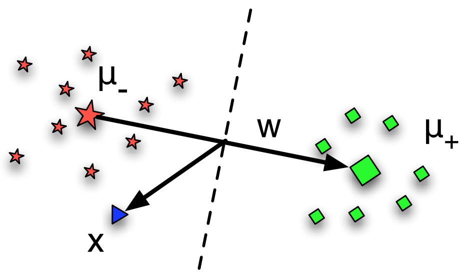


Fig. 1.20. A trivial classifier. Classification is carried out in accordance to which of the two means μ_- or μ_+ is closer to the test point x. Note that the sets of positive and negative labels respectively form a half space.

1.3.3 A Simple Classifier

We can use geometry to design another simple classification algorithm [SS02] for our problem. For simplicity we assume that the observations $x \in \mathbb{R}^d$, such as the bag-of-words representation of e-mails. We define the means μ_+ and μ_- to correspond to the classes $y \in \{\pm 1\}$ via

$$\mu_- := \frac{1}{m_-} \sum_{y_i=-1} x_i \text{ and } \mu_+ := \frac{1}{m_+} \sum_{y_i=1} x_i.$$

Here we used m_- and m_+ to denote the number of observations with label $y_i = -1$ and $y_i = +1$ respectively. An even simpler approach than using the nearest neighbor classifier would be to use the class label which corresponds to the mean closest to a new query x, as described in Figure 1.20.

For Euclidean distances we have

$$\|\mu_- - x\|^2 = \|\mu_-\|^2 + \|x\|^2 - 2\langle \mu_-, x\rangle \text{ and} \tag{1.19}$$
$$\|\mu_+ - x\|^2 = \|\mu_+\|^2 + \|x\|^2 - 2\langle \mu_+, x\rangle. \tag{1.20}$$

Here $\langle \cdot, \cdot \rangle$ denotes the standard dot product between vectors. Taking differences between the two distances yields

$$f(x) := \|\mu_+ - x\|^2 - \|\mu_- - x\|^2 = 2\langle \mu_- - \mu_+, x\rangle + \|\mu_-\|^2 - \|\mu_+\|^2. \tag{1.21}$$

This is a *linear* function in x and its sign corresponds to the labels we estimate for x. Our algorithm sports an important property: The classification rule can be expressed via dot products. This follows from

$$\|\mu_+\|^2 = \langle \mu_+, \mu_+\rangle = m_+^{-2} \sum_{y_i=y_j=1} \langle x_i, x_j\rangle \text{ and } \langle \mu_+, x\rangle = m_+^{-1} \sum_{y_i=1} \langle x_i, x\rangle.$$

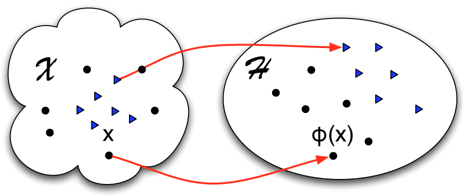


Fig. 1.21. The feature map ϕ maps observations x from $\mathcal{X}$ into a feature space $\mathcal{H}$. The map ϕ is a convenient way of encoding pre-processing steps systematically.

Analogous expressions can be computed for μ_-. Consequently we may express the classification rule (1.21) as

$$f(x) = \sum_{i=1}^{m} \alpha_i \langle x_i, x \rangle + b \tag{1.22}$$

where $b = m_-^{-2} \sum_{y_i=y_j=-1} \langle x_i, x_j \rangle - m_+^{-2} \sum_{y_i=y_j=1} \langle x_i, x_j \rangle$ and $\alpha_i = y_i/m_{y_i}$.

This offers a number of interesting extensions. Recall that when dealing with documents we needed to perform pre-processing to map e-mails into a vector space. In general, we may pick arbitrary maps $\phi : \mathcal{X} \to \mathcal{H}$ mapping the space of observations into a *feature space* $\mathcal{H}$, as long as the latter is endowed with a dot product (see Figure 1.21). This means that instead of dealing with $\langle x, x' \rangle$ we will be dealing with $\langle \phi(x), \phi(x') \rangle$.

As we will see in Chapter **??**, whenever $\mathcal{H}$ is a so-called Reproducing Kernel Hilbert Space, the inner product can be abbreviated in the form of a kernel function $k(x, x')$ which satisfies

$$k(x, x') := \langle \phi(x), \phi(x') \rangle . \tag{1.23}$$

This small modification leads to a number of very powerful algorithm and it is at the foundation of an area of research called kernel methods. We will encounter a number of such algorithms for regression, classification, segmentation, and density estimation over the course of the book. Examples of suitable k are the polynomial kernel $k(x, x') = \langle x, x' \rangle^d$ for $d \in \mathbb{N}$ and the Gaussian RBF kernel $k(x, x') = e^{-\gamma \|x - x'\|^2}$ for $\gamma > 0$.

The upshot of (1.23) is that our basic algorithm can be *kernelized*. That is, we may rewrite (1.21) as

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x) + b \tag{1.24}$$

where as before $\alpha_i = y_i/m_{y_i}$ and the offset b is computed analogously. As

Algorithm 1.3 The Perceptron

```
Perceptron($\mathbf{X}, \mathbf{Y}$) {reads stream of observations $(x_i, y_i)$}
  Initialize $w = 0$ and $b = 0$
  while There exists some $(x_i, y_i)$ with $y_i(\langle w, x_i\rangle + b) \leq 0$ do
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end while
```

Algorithm 1.4 The Kernel Perceptron

```
KernelPerceptron($\mathbf{X}, \mathbf{Y}$) {reads stream of observations $(x_i, y_i)$}
  Initialize $f = 0$
  while There exists some $(x_i, y_i)$ with $y_i f(x_i) \leq 0$ do
    $f \leftarrow f + y_i k(x_i, \cdot) + y_i$
  end while
```

a consequence we have now moved from a fairly simple and pedestrian linear classifier to one which yields a nonlinear function $f(x)$ with a rather nontrivial decision boundary.

1.3.4 Perceptron

In the previous sections we assumed that our classifier had access to a training set of spam and non-spam emails. In real life, such a set might be difficult to obtain all at once. Instead, a user might want to have *instant* results whenever a new e-mail arrives and he would like the system to learn immediately from any corrections to mistakes the system makes.

To overcome both these difficulties one could envisage working with the following protocol: As emails arrive our algorithm classifies them as spam or non-spam, and the user provides feedback as to whether the classification is correct or incorrect. This feedback is then used to improve the performance of the classifier over a period of time.

This intuition can be formalized as follows: Our classifier maintains a parameter vector. At the t-th time instance it receives a data point x_t, to which it assigns a label $\hat{y}_t$ using its current parameter vector. The true label y_t is then revealed, and used to update the parameter vector of the classifier. Such algorithms are said to be *online*. We will now describe perhaps the simplest classifier of this kind namely the Perceptron [Heb49, Ros58].

Let us assume that the data points $x_t \in \mathbb{R}^d$, and labels $y_t \in \{\pm 1\}$. As before we represent an email as a bag-of-words vector and we assign $+1$ to spam emails and -1 to non-spam emails. The Perceptron maintains a weight

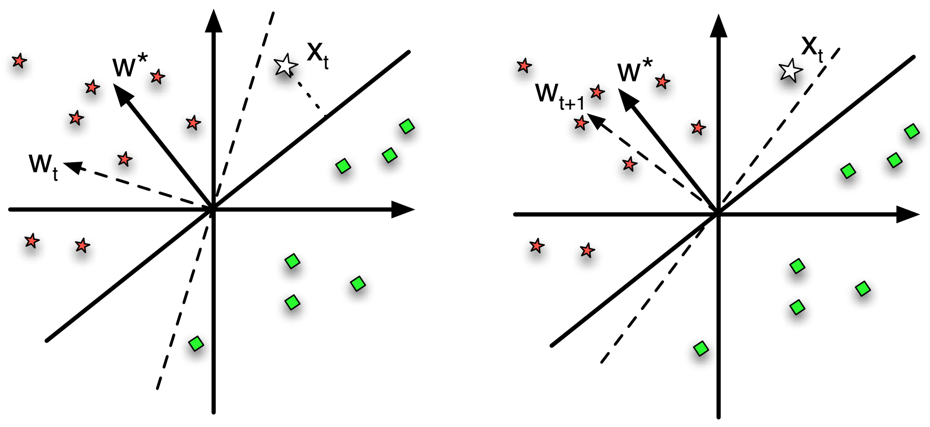


Fig. 1.22. The Perceptron without bias. Left: at time t we have a weight vector w_t denoted by the dashed arrow with corresponding separating plane (also dashed). For reference we include the linear separator w^* and its separating plane (both denoted by a solid line). As a new observation x_t arrives which happens to be mis-classified by the current weight vector w_t we perform an update. Also note the margin between the point x_t and the separating hyperplane defined by w^*. Right: This leads to the weight vector w_{t+1} which is more aligned with w^*.

vector $w \in \mathbb{R}^d$ and classifies x_t according to the rule

$$\hat{y}_t := \operatorname{sign}\{\langle w, x_t \rangle + b\}, \tag{1.25}$$

where $\langle w, x_t \rangle$ denotes the usual Euclidean dot product and b is an offset. Note the similarity of (1.25) to (1.21) of the simple classifier. Just as the latter, the Perceptron is a *linear* classifier which separates its domain $\mathbb{R}^d$ into two halfspaces, namely $\{x | \langle w, x \rangle + b > 0\}$ and its complement. If $\hat{y}_t = y_t$ then no updates are made. On the other hand, if $\hat{y}_t \neq y_t$ the weight vector is updated as

$$w \leftarrow w + y_t x_t \text{ and } b \leftarrow b + y_t. \tag{1.26}$$

Figure 1.22 shows an update step of the Perceptron algorithm. For simplicity we illustrate the case without bias, that is, where $b = 0$ and where it remains unchanged. A detailed description of the algorithm is given in Algorithm 1.3.

An important property of the algorithm is that it performs updates on w by multiples of the observations x_i on which it makes a mistake. Hence we may express w as $w = \sum_{i \in \text{Error}} y_i x_i$. Just as before, we can replace x_i and x by $\phi(x_i)$ and $\phi(x)$ to obtain a kernelized version of the Perceptron algorithm [FS99] (Algorithm 1.4).

If the dataset $(\mathbf{X}, \mathbf{Y})$ is linearly separable, then the Perceptron algorithm eventually converges and correctly classifies all the points in $\mathbf{X}$. The rate of convergence however depends on the margin. Roughly speaking, the margin quantifies how linearly separable a dataset is, and hence how easy it is to solve a given classification problem.

Definition 1.6 (Margin) *Let $w \in \mathbb{R}^d$ be a weight vector and let $b \in \mathbb{R}$ be an offset. The margin of an observation $x \in \mathbb{R}^d$ with associated label y is*

$$\gamma(x, y) := y\left(\langle w, x\rangle + b\right). \tag{1.27}$$

Moreover, the margin of an entire set of observations $\mathbf{X}$ with labels $\mathbf{Y}$ is

$$\gamma(\mathbf{X}, \mathbf{Y}) := \min_i \gamma(x_i, y_i). \tag{1.28}$$

Geometrically speaking (see Figure 1.22) the margin measures the distance of x from the hyperplane defined by $\{x| \langle w, x\rangle + b = 0\}$. Larger the margin, the more well separated the data and hence easier it is to find a hyperplane with correctly classifies the dataset. The following theorem asserts that if there exists a linear classifier which can classify a dataset with a large margin, then the Perceptron will also correctly classify the same dataset after making a small number of mistakes.

Theorem 1.7 (Novikoff's theorem) *Let $(\mathbf{X}, \mathbf{Y})$ be a dataset with at least one example labeled $+1$ and one example labeled -1. Let $R := \max_t \|x_t\|$, and assume that there exists (w^*, b^*) such that $\|w^*\| = 1$ and $\gamma_t := y_t(\langle w^*, x_t\rangle + b^*) \geq \gamma$ for all t. Then, the Perceptron will make at most $\frac{(1+R^2)(1+(b^*)^2)}{\gamma^2}$ mistakes.*

This result is remarkable since it does *not* depend on the dimensionality of the problem. Instead, it only depends on the *geometry* of the setting, as quantified via the margin γ and the radius R of a ball enclosing the observations. Interestingly, a similar bound can be shown for Support Vector Machines [Vap95] which we will be discussing in Chapter **??**.

Proof We can safely ignore the iterations where no mistakes were made and hence no updates were carried out. Therefore, without loss of generality assume that the t-th update was made after seeing the t-th observation and let w_t denote the weight vector after the update. Furthermore, for simplicity assume that the algorithm started with $w_0 = 0$ and $b_0 = 0$. By the update equation (1.26) we have

$$\begin{aligned}\langle w_t, w^*\rangle + b_t b^* &= \langle w_{t-1}, w^*\rangle + b_{t-1}b^* + y_t(\langle x_t, w^*\rangle + b^*)\\ &\geq \langle w_{t-1}, w^*\rangle + b_{t-1}b^* + \gamma.\end{aligned}$$

By induction it follows that $\langle w_t, w^* \rangle + b_t b^* \geq t\gamma$. On the other hand we made an update because $y_t(\langle x_t, w_{t-1} \rangle + b_{t-1}) < 0$. By using $y_t y_t = 1$,

$$\begin{aligned} \|w_t\|^2 + b_t^2 &= \|w_{t-1}\|^2 + b_{t-1}^2 + y_t^2 \|x_t\|^2 + 1 + 2y_t(\langle w_{t-1}, x_t \rangle + b_{t-1}) \\ &\leq \|w_{t-1}\|^2 + b_{t-1}^2 + \|x_t\|^2 + 1 \end{aligned}$$

Since $\|x_t\|^2 = R^2$ we can again apply induction to conclude that $\|w_t\|^2 + b_t^2 \leq t\left[R^2 + 1\right]$. Combining the upper and the lower bounds, using the Cauchy-Schwartz inequality, and $\|w^*\| = 1$ yields

$$\begin{aligned} t\gamma \leq \langle w_t, w^* \rangle + b_t b^* &= \left\langle \begin{bmatrix} w_t \\ b_t \end{bmatrix}, \begin{bmatrix} w^* \\ b^* \end{bmatrix} \right\rangle \\ &\leq \left\| \begin{bmatrix} w_t \\ b_t \end{bmatrix} \right\| \left\| \begin{bmatrix} w^* \\ b^* \end{bmatrix} \right\| = \sqrt{\|w_t\|^2 + b_t^2}\sqrt{1 + (b^*)^2} \\ &\leq \sqrt{t(R^2 + 1)}\sqrt{1 + (b^*)^2}. \end{aligned}$$

Squaring both sides of the inequality and rearranging the terms yields an upper bound on the number of updates and hence the number of mistakes. ■

The Perceptron was the building block of research on Neural Networks [Hay98, Bis95]. The key insight was to combine large numbers of such networks, often in a cascading fashion, to larger objects and to fashion optimization algorithms which would lead to classifiers with desirable properties. In this book we will take a complementary route. Instead of increasing the number of nodes we will investigate what happens when increasing the complexity of the feature map ϕ and its associated kernel k. The advantage of doing so is that we will reap the benefits from convex analysis and linear models, possibly at the expense of a slightly more costly function evaluation.

1.3.5 K-Means

All the algorithms we discussed so far are supervised, that is, they assume that labeled training data is available. In many applications this is too much to hope for; labeling may be expensive, error prone, or sometimes impossible. For instance, it is very easy to crawl and collect every page within the `www.purdue.edu` domain, but rather time consuming to assign a topic to each page based on its contents. In such cases, one has to resort to unsupervised learning. A prototypical unsupervised learning algorithm is K-means, which is clustering algorithm. Given $X = \{x_1, \ldots, x_m\}$ the goal of K-means is to partition it into k clusters such that each point in a cluster is similar to points from its own cluster than with points from some other cluster.

Towards this end, define prototype vectors $\mu_1, \ldots, \mu_k$ and an indicator vector r_{ij} which is 1 if, and only if, x_i is assigned to cluster j. To cluster our dataset we will minimize the following distortion measure, which minimizes the distance of each point from the prototype vector:

$$J(r, \mu) := \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{k} r_{ij} \|x_i - \mu_j\|^2, \tag{1.29}$$

where $r = \{r_{ij}\}$, $\mu = \{\mu_j\}$, and $\| \cdot \|^2$ denotes the usual Euclidean square norm.

Our goal is to find r and μ, but since it is not easy to jointly minimize J with respect to both r and μ, we will adapt a two stage strategy:

Stage 1 Keep the μ fixed and determine r. In this case, it is easy to see that the minimization decomposes into m independent problems. The solution for the i-th data point x_i can be found by setting:

$$r_{ij} = 1 \text{ if } j = \operatorname*{argmin}_{j'} \|x_i - \mu_{j'}\|^2, \tag{1.30}$$

and 0 otherwise.

Stage 2 Keep the r fixed and determine μ. Since the r's are fixed, J is an quadratic function of μ. It can be minimized by setting the derivative with respect to μ_j to be 0:

$$\sum_{i=1}^{m} r_{ij}(x_i - \mu_j) = 0 \text{ for all } j. \tag{1.31}$$

Rearranging obtains

$$\mu_j = \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}}. \tag{1.32}$$

Since $\sum_i r_{ij}$ counts the number of points assigned to cluster j, we are essentially setting μ_j to be the sample mean of the points assigned to cluster j.

The algorithm stops when the cluster assignments do not change significantly. Detailed pseudo-code can be found in Algorithm 1.5.

Two issues with K-Means are worth noting. First, it is sensitive to the choice of the initial cluster centers μ. A number of practical heuristics have been developed. For instance, one could randomly choose k points from the given dataset as cluster centers. Other methods try to pick k points from $\mathbf{X}$ which are farthest away from each other. Second, it makes a *hard* assignment of every point to a cluster center. Variants which we will encounter later in

Algorithm 1.5 K-Means

```
Cluster($\mathbf{X}$) {Cluster dataset $\mathbf{X}$}
  Initialize cluster centers $\mu_j$ for $j = 1, \ldots, k$ randomly
  **repeat**
    **for** $i = 1$ **to** $m$ **do**
      Compute $j' = \mathrm{argmin}_{j=1,\ldots,k} d(x_i, \mu_j)$
      Set $r_{ij'} = 1$ and $r_{ij} = 0$ for all $j' \neq j$
    **end for**
    **for** $j = 1$ **to** $k$ **do**
      Compute $\mu_j = \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}}$
    **end for**
  **until** Cluster assignments $r_{ij}$ are unchanged
  **return** $\{\mu_1, \ldots, \mu_k\}$ and $r_{ij}$
```

the book will relax this. Instead of letting $r_{ij} \in \{0, 1\}$ these *soft* variants will replace it with the probability that a given x_i belongs to cluster j.

The K-Means algorithm concludes our discussion of a set of basic machine learning methods for classification and regression. They provide a useful starting point for an aspiring machine learning researcher. In this book we will see many more such algorithms as well as connections between these basic algorithms and their more advanced counterparts.

Problems

Problem 1.1 (Eyewitness) *Assume that an eyewitness is 90% certain that a given person committed a crime in a bar. Moreover, assume that there were 50 people in the restaurant at the time of the crime. What is the posterior probability of the person actually having committed the crime.*

Problem 1.2 (DNA Test) *Assume the police have a DNA library of 10 million records. Moreover, assume that the false recognition probability is below* 0.00001% per record. *Suppose a match is found after a database search for an individual. What are the chances that the identification is correct? You can assume that the total population is 100 million people. Hint: compute the probability of no match occurring first.*

Problem 1.3 (Bomb Threat) *Suppose that the probability that one of a thousand passengers on a plane has a bomb is* $1 : 1,000,000$. *Assuming that the probability to have a bomb is evenly distributed among the passengers, the probability that two passengers have a bomb is roughly equal to* 10^{-12}.

Therefore, one might decide to take a bomb on a plane to decrease chances that somebody else has a bomb. What is wrong with this argument?

Problem 1.4 (Monty-Hall Problem) *Assume that in a TV show the candidate is given the choice between three doors. Behind two of the doors there is a pencil and behind one there is the grand prize, a car. The candidate chooses one door.* After that*, the showmaster opens another door behind which there is a pencil. Should the candidate switch doors after that? What is the probability of winning the car?*

Problem 1.5 (Mean and Variance for Random Variables) *Denote by X_i random variables. Prove that in this case*

$$\mathbf{E}_{X_1,\ldots X_N}\left[\sum_i x_i\right] = \sum_i \mathbf{E}_{X_i}[x_i] \text{ and } \mathrm{Var}_{X_1,\ldots X_N}\left[\sum_i x_i\right] = \sum_i \mathrm{Var}_{X_i}[x_i]$$

To show the second equality assume independence of the X_i.

Problem 1.6 (Two Dices) *Assume you have a game which uses the maximum of two dices. Compute the probability of seeing any of the events $\{1,\ldots,6\}$. Hint: prove first that the cumulative distribution function of the maximum of a pair of random variables is the square of the original cumulative distribution function.*

Problem 1.7 (Matching Coins) *Consider the following game: two players bring a coin each. the first player bets that when tossing the coins both will match and the second one bets that they will not match. Show that even if one of the players were to bring a tainted coin, the game still would be fair. Show that it is in the interest of each player to bring a fair coin to the game. Hint: assume that the second player knows that the first coin favors heads over tails.*

Problem 1.8 (Randomized Maximization) *How many observations do you need to draw from a distribution to ensure that the maximum over them is larger than 95% of all observations with at least 95% probability? Hint: generalize the result from Problem 1.6 to the maximum over n random variables.*

Application: Assume we have 1000 computers performing MapReduce [DG08] and the Reducers have to wait until all 1000 Mappers are finished with their job. Compute the quantile of the typical time to completion.

Problem 1.9 *Prove that the Normal distribution (1.3) has mean μ and variance σ^2. Hint: exploit the fact that p is symmetric around μ.*

Problem 1.10 (Cauchy Distribution) *Prove that for the density*

$$p(x) = \frac{1}{\pi(1+x^2)} \tag{1.33}$$

mean and variance are undefined. Hint: show that the integral diverges.

Problem 1.11 (Quantiles) *Find a distribution for which the mean exceeds the median. Hint: the mean depends on the value of the high-quantile terms, whereas the median does not.*

Problem 1.12 (Multicategory Naive Bayes) *Prove that for multicategory Naive Bayes the optimal decision is given by*

$$y^*(x) := \operatorname*{argmax}_y p(y) \prod_{i=1}^{n} p([x]_i|y) \tag{1.34}$$

where $y \in \mathcal{Y}$ is the class label of the observation x.

Problem 1.13 (Bayes Optimal Decisions) *Denote by $y^*(x) = \mathrm{argmax}_y\, p(y|x)$ the label associated with the largest conditional class probability. Prove that for $y^*(x)$ the probability of choosing the wrong label y is given by*

$$l(x) := 1 - p(y^*(x)|x).$$

Moreover, show that $y^*(x)$ is the label incurring the smallest misclassification error.

Problem 1.14 (Nearest Neighbor Loss) *Show that the expected loss incurred by the nearest neighbor classifier does not exceed twice the loss of the Bayes optimal decision.*

2

Density Estimation

2.1 Limit Theorems

Assume you are a gambler and go to a casino to play a game of dice. As it happens, it is your unlucky day and among the 100 times you toss the dice, you only see '6' eleven times. For a fair dice we know that each face should occur with equal probability $\frac{1}{6}$. Hence the expected value over 100 draws is $\frac{100}{6} \approx 17$, which is considerably more than the eleven times that we observed. Before crying foul you decide that some mathematical analysis is in order.

The probability of seeing a *particular* sequence of m trials out of which n are a '6' is given by $\frac{1}{6}^n \frac{5}{6}^{m-n}$. Moreover, there are $\binom{m}{n} = \frac{m!}{n!(m-n)!}$ different sequences of '6' and 'not 6' with proportions n and $m-n$ respectively. Hence we may compute the probability of seeing a '6' only 11 or less via

$$\Pr(X \leq 11) = \sum_{i=0}^{11} p(i) = \sum_{i=0}^{11} \binom{100}{i} \left[\frac{1}{6}\right]^i \left[\frac{5}{6}\right]^{100-i} \approx 7.0\% \qquad (2.1)$$

After looking at this figure you decide that things are probably reasonable. And, in fact, they are consistent with the convergence behavior of a simulated dice in Figure 2.1. In computing (2.1) we have learned something useful: the expansion is a special case of a *binomial* series. The first term

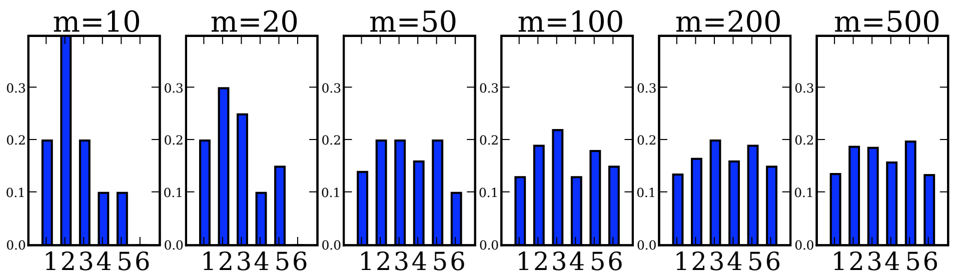


Fig. 2.1. Convergence of empirical means to expectations. From left to right: empirical frequencies of occurrence obtained by casting a dice 10, 20, 50, 100, 200, and 500 times respectively. Note that after 20 throws we still have not observed a single '6', an event which occurs with only $\left[\frac{5}{6}\right]^{20} \approx 2.6\%$ probability.

counts the number of configurations in which we could observe i times '6' in a sequence of 100 dice throws. The second and third term are the probabilities of seeing one particular instance of such a sequence.

Note that in general we may not be as lucky, since we may have considerably less information about the setting we are studying. For instance, we might not *know* the actual probabilities for each face of the dice, which would be a likely assumption when gambling at a casino of questionable reputation. Often the outcomes of the system we are dealing with may be continuous valued random variables rather than binary ones, possibly even with unknown range. For instance, when trying to determine the average wage through a questionnaire we need to determine how many people we need to ask in order to obtain a certain level of confidence.

To answer such questions we need to discuss limit theorems. They tell us by how much averages over a set of observations may deviate from the corresponding expectations and how many observations we need to draw to estimate a number of probabilities reliably. For completeness we will present proofs for some of the more fundamental theorems in Section 2.1.2. They are useful albeit non-essential for the understanding of the remainder of the book and may be omitted.

2.1.1 Fundamental Laws

The Law of Large Numbers developed by Bernoulli in 1713 is one of the fundamental building blocks of statistical analysis. It states that averages over a number of observations converge to their expectations given a sufficiently large number of observations and given certain assumptions on the independence of these observations. It comes in two flavors: the weak and the strong law.

Theorem 2.1 (Weak Law of Large Numbers) *Denote by $X_1, \ldots, X_m$ random variables drawn from $p(x)$ with mean $\mu = \mathbf{E}_{X_i}[x_i]$ for all i. Moreover let*

$$\bar{X}_m := \frac{1}{m} \sum_{i=1}^{m} X_i \tag{2.2}$$

be the empirical average over the random variables X_i. Then for any $\epsilon > 0$ the following holds

$$\lim_{m \to \infty} \Pr\left(\left|\bar{X}_m - \mu\right| \leq \epsilon\right) = 1. \tag{2.3}$$

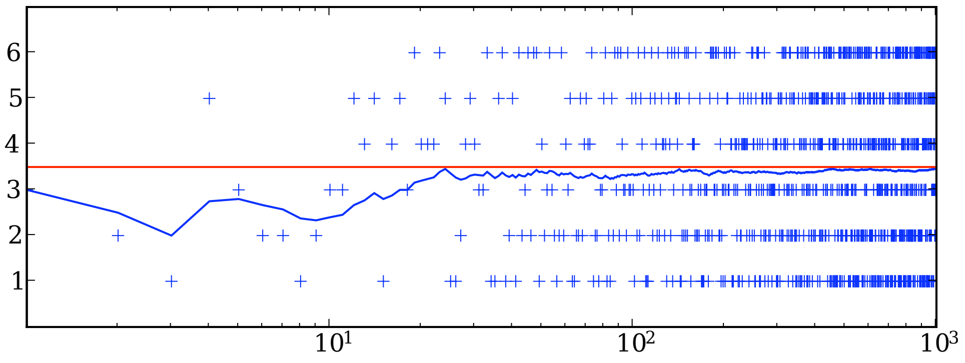


Fig. 2.2. The mean of a number of casts of a dice. The horizontal straight line denotes the mean 3.5. The uneven solid line denotes the actual mean $\bar{X}_n$ as a function of the number of draws, given as a semilogarithmic plot. The crosses denote the outcomes of the dice. Note how $\bar{X}_n$ ever more closely approaches the mean 3.5 are we obtain an increasing number of observations.

This establishes that, indeed, for large enough sample sizes, the average will converge to the expectation. The strong law strengthens this as follows:

Theorem 2.2 (Strong Law of Large Numbers) *Under the conditions of Theorem 2.1 we have* $\Pr\left(\lim_{m\to\infty} \bar{X}_m = \mu\right) = 1$.

The strong law implies that almost surely (in a measure theoretic sense) $\bar{X}_m$ converges to μ, whereas the weak law only states that for every ϵ the random variable $\bar{X}_m$ will be within the interval $[\mu-\epsilon, \mu+\epsilon]$. Clearly the strong implies the weak law since the measure of the events $\bar{X}_m = \mu$ converges to 1, hence any ϵ-ball around μ would capture this.

Both laws justify that we may take sample averages, e.g. over a number of events such as the outcomes of a dice and use the latter to estimate their means, their probabilities (here we treat the indicator variable of the event as a $\{0; 1\}$-valued random variable), their variances or related quantities. We postpone a proof until Section 2.1.2, since an effective way of proving Theorem 2.1 relies on the theory of characteristic functions which we will discuss in the next section. For the moment, we only give a pictorial illustration in Figure 2.2.

Once we established that the random variable $\bar{X}_m = m^{-1}\sum_{i=1}^m X_i$ converges to its mean μ, a natural second question is to establish how *quickly* it converges and what the properties of the limiting distribution of $\bar{X}_m - \mu$ are. Note in Figure 2.2 that the initial deviation from the mean is large whereas as we observe more data the empirical mean approaches the true one.

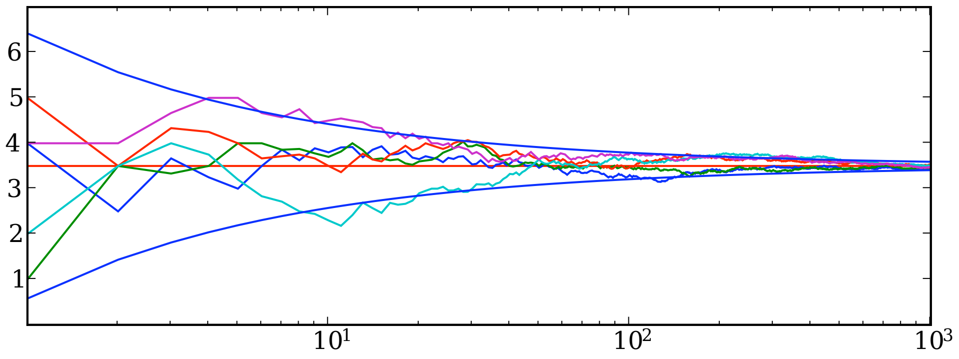


Fig. 2.3. Five instantiations of a running average over outcomes of a toss of a dice. Note that all of them converge to the mean 3.5. Moreover note that they all are well contained within the upper and lower envelopes given by $\mu \pm \sqrt{\mathrm{Var}_X[x]/m}$.

The central limit theorem answers this question exactly by addressing a slightly more general question, namely whether the sum over a number of independent random variables where each of them arises from a *different* distribution might also have a well behaved limiting distribution. This is the case as long as the *variance* of each of the random variables is bounded. The limiting distribution of such a sum is Gaussian. This affirms the pivotal role of the Gaussian distribution.

Theorem 2.3 (Central Limit Theorem) *Denote by X_i independent random variables with means μ_i and standard deviation σ_i. Then*

$$Z_m := \left[\sum_{i=1}^m \sigma_i^2\right]^{-\frac{1}{2}} \left[\sum_{i=1}^m X_i - \mu_i\right] \tag{2.4}$$

converges to a Normal Distribution with zero mean and unit variance.

Note that just like the law of large numbers the central limit theorem (CLT) is an *asymptotic* result. That is, only in the limit of an infinite number of observations will it become exact. That said, it often provides an excellent approximation even for finite numbers of observations, as illustrated in Figure 2.4. In fact, the central limit theorem and related limit theorems build the foundation of what is known as asymptotic statistics.

Example 2.1 (Dice) *If we are interested in computing the mean of the values returned by a dice we may apply the CLT to the sum over m variables which have all mean $\mu = 3.5$ and variance (see Problem 2.1)*

$$\mathrm{Var}_X[x] = \mathbf{E}_X[x^2] - \mathbf{E}_X[x]^2 = (1+4+9+16+25+36)/6 - 3.5^2 \approx 2.92.$$

We now study the random variable $W_m := m^{-1} \sum_{i=1}^m [X_i - 3.5]$. Since each of the terms in the sum has zero mean, also W_m's mean vanishes. Moreover, W_m is a multiple of Z_m of (2.4). Hence we have that W_m converges to a normal distribution with zero mean and standard deviation $2.92 m^{-\frac{1}{2}}$.

Consequently the average of m tosses of the dice yields a random variable with mean 3.5 and it will approach a normal distribution with variance $m^{-\frac{1}{2}} 2.92$. In other words, the empirical mean converges to its average at rate $O(m^{-\frac{1}{2}})$. Figure 2.3 gives an illustration of the quality of the bounds implied by the CLT.

One remarkable property of functions of random variables is that in many conditions convergence properties of the random variables are bestowed upon the functions, too. This is manifest in the following two results: a variant of Slutsky's theorem and the so-called delta method. The former deals with limit behavior whereas the latter deals with an extension of the central limit theorem.

Theorem 2.4 (Slutsky's Theorem) *Denote by X_i, Y_i sequences of random variables with $X_i \to X$ and $Y_i \to c$ for $c \in \mathbb{R}$ in probability. Moreover, denote by $g(x, y)$ a function which is continuous for all (x, c). In this case the random variable $g(X_i, Y_i)$ converges in probability to $g(X, c)$.*

For a proof see e.g. [Bil68]. Theorem 2.4 is often referred to as the continuous mapping theorem (Slutsky only proved the result for affine functions). It means that for functions of random variables it is possible to pull the limiting procedure *into* the function. Such a device is useful when trying to prove asymptotic normality and in order to obtain characterizations of the limiting distribution.

Theorem 2.5 (Delta Method) *Assume that $X_n \in \mathbb{R}^d$ is asymptotically normal with $a_n^{-2}(X_n - b) \to \mathcal{N}(0, \Sigma)$ for $a_n^2 \to 0$. Moreover, assume that $g : \mathbb{R}^d \to \mathbb{R}^l$ is a mapping which is continuously differentiable at b. In this case the random variable $g(X_n)$ converges*

$$a_n^{-2} \left(g(X_n) - g(b)\right) \to \mathcal{N}(0, [\nabla_x g(b)] \Sigma [\nabla_x g(b)]^\top). \tag{2.5}$$

Proof Via a Taylor expansion we see that

$$a_n^{-2} \left[g(X_n) - g(b)\right] = [\nabla_x g(\xi_n)]^\top a_n^{-2}(X_n - b) \tag{2.6}$$

Here ξ_n lies on the line segment $[b, X_n]$. Since $X_n \to b$ we have that $\xi_n \to b$, too. Since g is continuously differentiable at b we may apply Slutsky's theorem to see that $a_n^{-2} \left[g(X_n) - g(b)\right] \to [\nabla_x g(b)]^\top a_n^{-2}(X_n - b)$. As a consequence, the transformed random variable is asymptotically normal with covariance $[\nabla_x g(b)]\Sigma[\nabla_x g(b)]^\top$. ■

We will use the delta method when it comes to investigating properties of maximum likelihood estimators in exponential families. There g will play the role of a mapping between expectations and the natural parametrization of a distribution.

2.1.2 The Characteristic Function

The Fourier transform plays a crucial role in many areas of mathematical analysis and engineering. This is equally true in statistics. For historic reasons its applications to distributions is called the characteristic function, which we will discuss in this section. At its foundations lie standard tools from functional analysis and signal processing [Rud73, Pap62]. We begin by recalling the basic properties:

Definition 2.6 (Fourier Transform) *Denote by $f : \mathbb{R}^n \to \mathbb{C}$ a function defined on a d-dimensional Euclidean space. Moreover, let $x, \omega \in \mathbb{R}^n$. Then the Fourier transform F and its inverse F^{-1} are given by*

$$F[f](\omega) := (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^n} f(x) \exp(-i \langle \omega, x \rangle) dx \tag{2.7}$$

$$F^{-1}[g](x) := (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^n} g(\omega) \exp(i \langle \omega, x \rangle) d\omega. \tag{2.8}$$

The key insight is that $F^{-1} \circ F = F \circ F^{-1} = \mathrm{Id}$. In other words, F and F^{-1} are inverses to each other for all functions which are L_2 integrable on $\mathbb{R}^d$, which includes probability distributions. One of the key advantages of Fourier transforms is that derivatives and convolutions on f translate into multiplications. That is $F[f \circ g] = (2\pi)^{\frac{d}{2}} F[f] \cdot F[g]$. The same rule applies to the inverse transform, i.e. $F^{-1}[f \circ g] = (2\pi)^{\frac{d}{2}} F^{-1}[f] F^{-1}[g]$.

The benefit for statistical analysis is that often problems are more easily expressed in the Fourier domain and it is easier to prove convergence results there. These results then carry over to the original domain. We will be exploiting this fact in the proof of the law of large numbers and the central limit theorem. Note that the definition of Fourier transforms can be extended to more general domains such as groups. See e.g. [BCR84] for further details. We next introduce the notion of a *characteristic function* of a distribution.[1]

Definition 2.7 (Characteristic Function) *Denote by $p(x)$ a distribution*

[1] In Chapter **??** we will discuss more general descriptions of distributions of which ϕ_X is a special case. In particular, we will replace the exponential $\exp(i \langle \omega, x \rangle)$ by a kernel function $k(x, x')$.

of a random variable $X \in \mathbb{R}^d$*. Then the characteristic function* $\phi_X(\omega)$ *with* $\omega \in \mathbb{R}^d$ *is given by*

$$\phi_X(\omega) := (2\pi)^{\frac{d}{2}} F^{-1}[p(x)] = \int \exp(i \langle \omega, x \rangle) dp(x). \tag{2.9}$$

In other words, $\phi_X(\omega)$ is the *inverse Fourier transform* applied to the probability measure $p(x)$. Consequently $\phi_X(\omega)$ *uniquely* characterizes $p(x)$ and moreover, $p(x)$ can be recovered from $\phi_X(\omega)$ via the forward Fourier transform. One of the key utilities of characteristic functions is that they allow us to deal in easy ways with *sums* of random variables.

Theorem 2.8 (Sums of random variables and convolutions) *Denote by* $X, Y \in \mathbb{R}$ *two independent random variables. Moreover, denote by* $Z := X + Y$ *the sum of both random variables. Then the distribution over* Z *satisfies* $p(z) = p(x) \circ p(y)$*. Moreover, the characteristic function yields:*

$$\phi_Z(\omega) = \phi_X(\omega)\phi_Y(\omega). \tag{2.10}$$

Proof Z is given by $Z = X + Y$. Hence, for a given $Z = z$ we have the freedom to choose $X = x$ freely provided that $Y = z - x$. In terms of distributions this means that the joint distribution $p(z, x)$ is given by

$$\begin{aligned} p(z, x) &= p(Y = z - x)p(x) \\ \text{and hence } p(z) &= \int p(Y = z - x) dp(x) = [p(x) \circ p(y)](z). \end{aligned}$$

The result for characteristic functions follows form the property of the Fourier transform. ∎

For sums of several random variables the characteristic function is the product of the individual characteristic functions. This allows us to prove both the weak law of large numbers and the central limit theorem (see Figure 2.4 for an illustration) by proving convergence in the Fourier domain.

Proof [Weak Law of Large Numbers] At the heart of our analysis lies a Taylor expansion of the exponential into

$$\begin{aligned} \exp(iwx) &= 1 + i \langle w, x \rangle + o(|w|) \\ \text{and hence } \phi_X(\omega) &= 1 + iw\mathbf{E}_X[x] + o(|w|). \end{aligned}$$

Given m random variables X_i with mean $\mathbf{E}_X[x] = \mu$ this means that their average $\bar{X}_m := \frac{1}{m}\sum_{i=1}^m X_i$ has the characteristic function

$$\phi_{\bar{X}_m}(\omega) = \left(1 + \frac{i}{m} w\mu + o(m^{-1}|w|)\right)^m \tag{2.11}$$

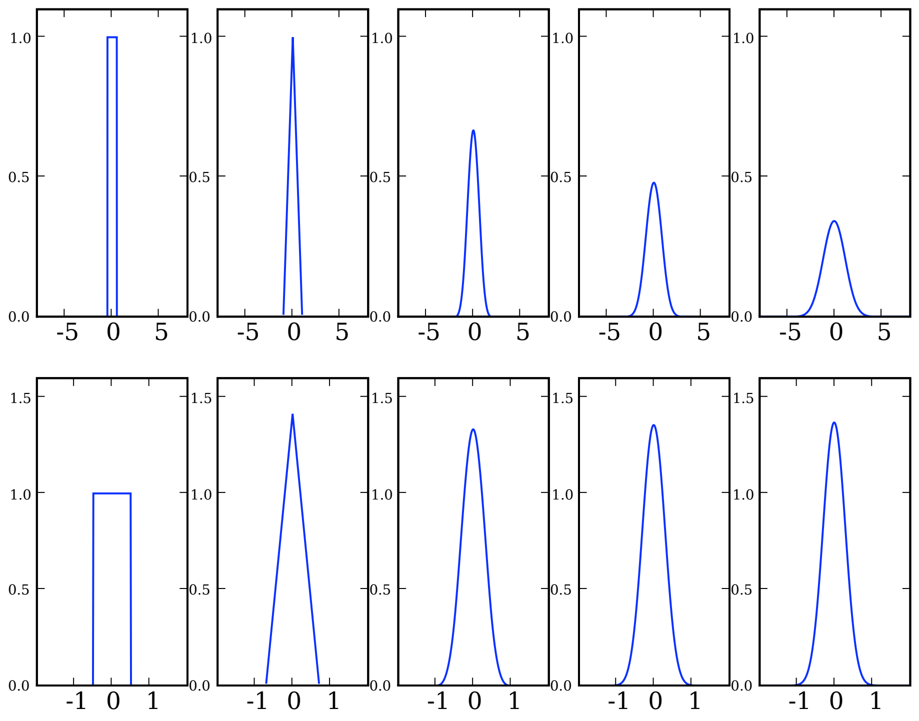


Fig. 2.4. A working example of the central limit theorem. The top row contains distributions of sums of uniformly distributed random variables on the interval $[0.5, 0.5]$. From left to right we have sums of $1, 2, 4, 8$ and 16 random variables. The bottom row contains the same distribution with the means rescaled by $\sqrt{m}$, where m is the number of observations. Note how the distribution converges increasingly to the normal distribution.

In the limit of $m \to \infty$ this converges to $\exp(iw\mu)$, the characteristic function of the constant distribution with mean μ. This proves the claim that in the large sample limit $\bar{X}_m$ is essentially constant with mean μ. ■

Proof [Central Limit Theorem] We use the same idea as above to prove the CLT. The main difference, though, is that we need to assume that the second moments of the random variables X_i *exist*. To avoid clutter we only prove the case of constant mean $\mathbf{E}_{X_i}[x_i] = \mu$ and variance $\mathrm{Var}_{X_i}[x_i] = \sigma^2$.

Let $Z_m := \frac{1}{\sqrt{m\sigma^2}} \sum_{i=1}^m (X_i - \mu)$. Our proof relies on showing convergence of the characteristic function of Z_m, i.e. ϕ_{Z_m} to that of a normally distributed random variable W with zero mean and unit variance. Expanding the exponential to second order yields:

$$\exp(iwx) = 1 + iwx - \frac{1}{2}w^2x^2 + o(|w|^2)$$

$$\text{and hence } \phi_X(\omega) = 1 + iw\mathbf{E}_X[x] - \frac{1}{2}w^2\mathrm{Var}_X[x] + o(|w|^2)$$

Since the mean of Z_m vanishes by centering $(X_i - \mu)$ and the variance per variable is m^{-1} we may write the characteristic function of Z_m via

$$\phi_{Z_m}(\omega) = \left(1 - \frac{1}{2m}w^2 + o(m^{-1}|w|^2)\right)^m$$

As before, taking limits $m \to \infty$ yields the exponential function. We have that $\lim_{m\to\infty}\phi_{Z_m}(\omega) = \exp(-\frac{1}{2}\omega^2)$ which is the characteristic function of the normal distribution with zero mean and variance 1. Since the characteristic function transform is injective this proves our claim. ■

Note that the characteristic function has a number of useful properties. For instance, it can also be used as moment generating function via the identity:

$$\nabla_\omega^n \phi_X(0) = i^{-n}\mathbf{E}_X[x^n]. \tag{2.12}$$

Its proof is left as an exercise. See Problem 2.2 for details. This connection also implies (subject to regularity conditions) that if we know the moments of a distribution we are able to reconstruct it directly since it allows us to reconstruct its characteristic function. This idea has been exploited in density estimation [Cra46] in the form of Edgeworth and Gram-Charlier expansions [Hal92].

2.1.3 Tail Bounds

In practice we never have access to an *infinite* number of observations. Hence the central limit theorem does not apply but is just an approximation to the real situation. For instance, in the case of the dice, we might want to state *worst case bounds* for *finite* sums of random variables to determine by how much the empirical mean may deviate from its expectation. Those bounds will not only be useful for simple averages but to quantify the behavior of more sophisticated estimators based on a set of observations.

The bounds we discuss below differ in the amount of knowledge they assume about the random variables in question. For instance, we might only know their mean. This leads to the Gauss-Markov inequality. If we know their mean and their variance we are able to state a stronger bound, the Chebyshev inequality. For an even stronger setting, when we know that each variable has bounded range, we will be able to state a Chernoff bound. Those bounds are progressively more tight and also more difficult to prove. We state them in order of technical sophistication.

Theorem 2.9 (Gauss-Markov) *Denote by $X \geq 0$ a random variable and let μ be its mean. Then for any $\epsilon > 0$ we have*

$$\Pr(X \geq \epsilon) \leq \frac{\mu}{\epsilon}. \tag{2.13}$$

Proof We use the fact that for nonnegative random variables

$$\Pr(X \geq \epsilon) = \int_{\epsilon}^{\infty} dp(x) \leq \int_{\epsilon}^{\infty} \frac{x}{\epsilon} dp(x) \leq \epsilon^{-1} \int_{0}^{\infty} x dp(x) = \frac{\mu}{\epsilon}.$$

This means that for random variables with a small mean, the proportion of samples with large value has to be small. ■

Consequently deviations from the mean are $O(\epsilon^{-1})$. However, note that this bound does *not* depend on the number of observations. A useful application of the Gauss-Markov inequality is Chebyshev's inequality. It is a statement on the range of random variables using its variance.

Theorem 2.10 (Chebyshev) *Denote by X a random variable with mean μ and variance σ^2. Then the following holds for $\epsilon > 0$:*

$$\Pr(|x - \mu| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}. \tag{2.14}$$

Proof Denote by $Y := |X - \mu|^2$ the random variable quantifying the deviation of X from its mean μ. By construction we know that $\mathbf{E}_Y[y] = \sigma^2$. Next let $\gamma := \epsilon^2$. Applying Theorem 2.9 to Y and γ yields $\Pr(Y > \gamma) \leq \sigma^2/\gamma$ which proves the claim. ■

Note the improvement to the Gauss-Markov inequality. Where before we had bounds whose confidence improved with $O(\epsilon^{-1})$ we can now state $O(\epsilon^{-2})$ bounds for deviations from the mean.

Example 2.2 (Chebyshev bound) *Assume that $\bar{X}_m := m^{-1} \sum_{i=1}^{m} X_i$ is the average over m random variables with mean μ and variance σ^2. Hence $\bar{X}_m$ also has mean μ. Its variance is given by*

$$\mathrm{Var}_{\bar{X}_m}[\bar{x}_m] = \sum_{i=1}^{m} m^{-2} \mathrm{Var}_{X_i}[x_i] = m^{-1}\sigma^2.$$

Applying Chebyshev's inequality yields that the probability of a deviation *of ϵ from the mean μ is bounded by $\frac{\sigma^2}{m\epsilon^2}$. For fixed failure probability $\delta = \Pr(|\bar{X}_m - \mu| > \epsilon)$ we have*

$$\delta \leq \sigma^2 m^{-1} \epsilon^{-2} \text{ and equivalently } \epsilon \leq \sigma/\sqrt{m\delta}.$$

This bound is quite reasonable for large δ but it means that for high levels of confidence we need a huge number of observations.

Much stronger results can be obtained if we are able to bound the *range* of the random variables. Using the latter, we reap an exponential improvement in the quality of the bounds in the form of the McDiarmid [McD89] inequality. We state the latter without proof:

Theorem 2.11 (McDiarmid) *Denote by $f : \mathcal{X}^m \to \mathbb{R}$ a function on $\mathcal{X}$ and let X_i be independent random variables. In this case the following holds:*

$$\Pr\left(|f(x_1, \ldots, x_m) - \mathbf{E}_{X_1,\ldots,X_m}[f(x_1, \ldots, x_m)]| > \epsilon\right) \leq 2 \exp\left(-2\epsilon^2 C^{-2}\right).$$

Here the constant C^2 is given by $C^2 = \sum_{i=1}^m c_i^2$ where

$$\left|f(x_1, \ldots, x_i, \ldots, x_m) - f(x_1, \ldots, x_i', \ldots, x_m)\right| \leq c_i$$

for all $x_1, \ldots, x_m, x_i'$ and for all i.

This bound can be used for averages of a number of observations when they are computed according to some algorithm as long as the latter can be encoded in f. In particular, we have the following bound [Hoe63]:

Theorem 2.12 (Hoeffding) *Denote by X_i iid random variables with bounded range $X_i \in [a, b]$ and mean μ. Let $\bar{X}_m := m^{-1} \sum_{i=1}^m X_i$ be their average. Then the following bound holds:*

$$\Pr\left(\left|\bar{X}_m - \mu\right| > \epsilon\right) \leq 2 \exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right). \tag{2.15}$$

Proof This is a corollary of Theorem 2.11. In $\bar{X}_m$ each individual random variable has range $[a/m, b/m]$ and we set $f(X_1, \ldots, X_m) := \bar{X}_m$. Straightforward algebra shows that $C^2 = m^{-2}(b-a)^2$. Plugging this back into McDiarmid's theorem proves the claim. ■

Note that (2.15) is *exponentially* better than the previous bounds. With increasing sample size the confidence level also increases exponentially.

Example 2.3 (Hoeffding bound) *As in example 2.2 assume that X_i are iid random variables and let $\bar{X}_m$ be their average. Moreover, assume that*

$X_i \in [a, b]$ for all i. As before we want to obtain guarantees on the probability that $|\bar{X}_m - \mu| > \epsilon$. For a given level of confidence $1 - \delta$ we need to solve

$$\delta \leq 2 \exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right) \tag{2.16}$$

for ϵ. Straightforward algebra shows that in this case ϵ needs to satisfy

$$\epsilon \geq |b - a| \sqrt{[\log 2 - \log \delta] / 2m} \tag{2.17}$$

In other words, while the confidence level only enters logarithmically into the inequality, the sample size m improves our confidence only with $\epsilon = O(m^{-\frac{1}{2}})$. That is, in order to improve our confidence interval from $\epsilon = 0.1$ to $\epsilon = 0.01$ we need 100 times as many observations.

While this bound is tight (see Problem 2.5 for details), it is possible to obtain better bounds if we know *additional* information. In particular knowing a bound on the *variance* of a random variable in addition to knowing that it has bounded range would allow us to strengthen the statement considerably. The Bernstein inequality captures this connection. For details see [BBL05] or works on empirical process theory [vdVW96, SW86, Vap82].

2.1.4 An Example

It is probably easiest to illustrate the various bounds using a concrete example. In a semiconductor fab processors are produced on a wafer. A typical 300mm wafer holds about 400 chips. A large number of processing steps are required to produce a finished microprocessor and often it is impossible to assess the effect of a design decision until the finished product has been produced.

Assume that the production manager wants to change some step from process 'A' to some other process 'B'. The goal is to increase the yield of the process, that is, the number of chips of the 400 potential chips on the wafer which can be sold. Unfortunately this number is a random variable, i.e. the number of working chips per wafer can vary widely between different wafers. Since process 'A' has been running in the factory for a very long time we may assume that the yield is well known, say it is $\mu_A = 350$ out of 400 processors on average. It is our goal to determine whether process 'B' is better and what its yield may be. Obviously, since production runs are expensive we want to be able to determine this number as quickly as possible, i.e. using as few wafers as possible. The production manager is risk averse and wants to ensure that the new process is really better. Hence he requires a confidence level of 95% before he will change the production.

A first step is to formalize the problem. Since we know process 'A' exactly we only need to concern ourselves with 'B'. We associate the random variable X_i with wafer i. A reasonable (and somewhat simplifying) assumption is to posit that all X_i are independent and identically distributed where all X_i have the mean μ_B. Obviously we do not know μ_B — otherwise there would be no reason for testing! We denote by $\bar{X}_m$ the average of the yields of m wafers using process 'B'. What we are interested in is the accuracy ϵ for which the probability

$$\delta = \Pr(|\bar{X}_m - \mu_B| > \epsilon) \text{ satisfies } \delta \leq 0.05.$$

Let us now discuss how the various bounds behave. For the sake of the argument assume that $\mu_B - \mu_A = 20$, i.e. the new process produces on average 20 additional usable chips.

Chebyshev In order to apply the Chebyshev inequality we need to bound the variance of the random variables X_i. The worst possible variance would occur if $X_i \in \{0; 400\}$ where both events occur with equal probability. In other words, with equal probability the wafer if fully usable or it is entirely broken. This amounts to $\sigma^2 = 0.5(200-0)^2 + 0.5(200-400)^2 = 40,000$. Since for Chebyshev bounds we have

$$\delta \leq \sigma^2 m^{-1} \epsilon^{-2} \tag{2.18}$$

we can solve for $m = \sigma^2/\delta\epsilon^2 = 40,000/(0.05 \cdot 400) = 20,000$. In other words, we would typically need 20,000 wafers to assess with reasonable confidence whether process 'B' is better than process 'A'. This is completely unrealistic.

Slightly better bounds can be obtained if we are able to make better assumptions on the variance. For instance, if we can be sure that the yield of process 'B' is at least 300, then the largest possible variance is $0.25(300-0)^2 + 0.75(300-400)^2 = 30,000$, leading to a minimum of 15,000 wafers which is not much better.

Hoeffding Since the yields are in the interval $\{0, \ldots, 400\}$ we have an explicit bound on the range of observations. Recall the inequality (2.16) which bounds the failure probably $\delta = 0.05$ by an exponential term. Solving this for m yields

$$m \geq 0.5|b-a|^2 \epsilon^{-2} \log(2/\delta) \approx 737.8 \tag{2.19}$$

In other words, we need at lest 738 wafers to determine whether process 'B' is better. While this is a significant improvement of almost two orders of magnitude, it still seems wasteful and we would like to do better.

Central Limit Theorem The central limit theorem is an *approximation*. This means that our reasoning is not accurate any more. That said, for large enough sample sizes, the approximation is good enough to use it for practical predictions. Assume for the moment that we knew the variance σ^2 exactly. In this case we know that $\bar{X}_m$ is approximately normal with mean μ_B and variance $m^{-1}\sigma^2$. We are interested in the interval $[\mu-\epsilon, \mu+\epsilon]$ which contains 95% of the probability mass of a normal distribution. That is, we need to solve the integral

$$\frac{1}{2\pi\sigma^2}\int_{\mu-\epsilon}^{\mu+\epsilon} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx = 0.95 \tag{2.20}$$

This can be solved efficiently using the cumulative distribution function of a normal distribution (see Problem 2.3 for more details). One can check that (2.20) is solved for $\epsilon = 2.96\sigma$. In other words, an interval of $\pm 2.96\sigma$ contains 95% of the probability mass of a normal distribution. The number of observations is therefore determined by

$$\epsilon = 2.96\sigma/\sqrt{m} \text{ and hence } m = 8.76\frac{\sigma^2}{\epsilon^2} \tag{2.21}$$

Again, our problem is that we do *not* know the variance of the distribution. Using the worst-case bound on the variance, i.e. $\sigma^2 = 40,000$ would lead to a requirement of at least $m = 876$ wafers for testing. However, while we do not *know* the variance, we may estimate it along with the mean and use the empirical estimate, possibly plus some small constant to ensure we do not underestimate the variance, instead of the upper bound.

Assuming that fluctuations turn out to be in the order of 50 processors, i.e. $\sigma^2 = 2500$, we are able to reduce our requirement to approximately 55 wafers. This is probably an acceptable number for a practical test.

Rates and Constants The astute reader will have noticed that all three confidence bounds had scaling behavior $m = O(\epsilon^{-2})$. That is, in all cases the number of observations was a fairly ill behaved function of the amount of confidence required. If we were just interested in convergence per se, a statement like that of the Chebyshev inequality would have been entirely sufficient. The various laws and bounds can often be used to obtain considerably better *constants* for statistical confidence guarantees. For more complex estimators, such as methods to classify, rank, or annotate data, a reasoning such as the one above can become highly nontrivial. See e.g. [MYA94, Vap98] for further details.

2.2 Parzen Windows

2.2.1 Discrete Density Estimation

The convergence theorems discussed so far mean that we can use empirical observations for the purpose of density estimation. Recall the case of the Naive Bayes classifier of Section 1.3.1. One of the key ingredients was the ability to use information about word counts for different document classes to estimate the probability $p(w^j|y)$, where w^j denoted the number of occurrences of word j in document x, given that it was labeled y. In the following we discuss an extremely simple and crude method for estimating probabilities. It relies on the fact that for random variables X_i drawn from distribution $p(x)$ with discrete values $X_i \in \mathcal{X}$ we have

$$\lim_{m \to \infty} \hat{p}_X(x) = p(x) \tag{2.22}$$

$$\text{where } \hat{p}_X(x) := m^{-1} \sum_{i=1}^{m} \{x_i = x\} \text{ for all } x \in \mathcal{X}. \tag{2.23}$$

Let us discuss a concrete case. We assume that we have 12 documents and would like to estimate the probability of occurrence of the word 'dog' from it. As raw data we have:

Document ID	1	2	3	4	5	6	7	8	9	10	11	12
Occurrences of ‘dog’	1	0	2	0	4	6	3	0	6	2	0	1

This means that the word ‘dog’ occurs the following number of times:

Occurrences of ‘dog’	0	1	2	3	4	5	6
Number of documents	4	2	2	1	1	0	2

Something unusual is happening here: for some reason we never observed 5 instances of the word dog in our documents, only 4 and less, or alternatively 6 times. So what about 5 times? It is reasonable to assume that the corresponding value should not be 0 either. Maybe we did not sample enough. One possible strategy is to add pseudo-counts to the observations. This amounts to the following estimate:

$$\hat{p}_X(x) := (m + |\mathcal{X}|)^{-1} \Big[1 + \sum_{i=1}^{m} \{x_i = x\} = p(x)\Big] \tag{2.24}$$

Clearly the limit for $m \to \infty$ is still $p(x)$. Hence, asymptotically we do not lose anything. This prescription is what we used in Algorithm 1.1 used a method called Laplace smoothing. Below we contrast the two methods:

Occurrences of ‘dog’	0	1	2	3	4	5	6
Number of documents	4	2	2	1	1	0	2
Frequency of occurrence	0.33	0.17	0.17	0.083	0.083	0	0.17
Laplace smoothing	0.26	0.16	0.16	0.11	0.11	0.05	0.16

The problem with this method is that as $|\mathcal{X}|$ increases we need increasingly more observations to obtain even a modicum of precision. On average, we will need at least one observation for every $x \in \mathcal{X}$. This can be infeasible for large domains as the following example shows.

Example 2.4 (Curse of Dimensionality) *Assume that $\mathcal{X} = \{0, 1\}^d$, i.e. x consists of binary bit vectors of dimensionality d. As d increases the size of $\mathcal{X}$ increases exponentially, requiring an exponential number of observations to perform density estimation. For instance, if we work with images, a 100 × 100 black and white picture would require in the order of 10^{3010} observations to model such fairly low-resolution images accurately. This is clearly utterly infeasible — the number of particles in the known universe is in the order of 10^{80}. Bellman [Bel61] was one of the first to formalize this dilemma by coining the term 'curse of dimensionality'.*

This example clearly shows that we need better tools to deal with high-dimensional data. We will present one of such tools in the next section.

2.2.2 Smoothing Kernel

We now proceed to proper density estimation. Assume that we want to estimate the distribution of weights of a population. Sample data from a population might look as follows: $X = \{57, 88, 54, 84, 83, 59, 56, 43, 70, 63, 90, 98, 102, 97, 106, 99, 103, 112\}$. We could use this to perform a density estimate by placing discrete components at the locations $x_i \in X$ with weight $1/|X|$ as what is done in Figure 2.5. There is no reason to believe that weights are quantized in kilograms, or grams, or miligrams (or pounds and stones). And even if it were, we would expect that similar weights would have similar densities associated with it. Indeed, as the right diagram of Figure 2.5 shows, the corresponding density is continuous.

The key question arising is how we may transform X into a realistic estimate of the density $p(x)$. Starting with a 'density estimate' with only discrete terms

$$\hat{p}(x) = \frac{1}{m} \sum_{i=1}^{m} \delta(x - x_i) \tag{2.25}$$

we may choose to smooth it out by a smoothing kernel $h(x)$ such that the probability mass becomes somewhat more spread out. For a density estimate on $\mathcal{X} \subseteq \mathbb{R}^d$ this is achieved by

$$\hat{p}(x) = \frac{1}{m} \sum_{i=1}^{m} r^{-d} h\left(\tfrac{x-x_i}{r}\right). \tag{2.26}$$

This expansion is commonly known as the *Parzen windows* estimate. Note that obviously h must be chosen such that $h(x) \geq 0$ for all $x \in \mathcal{X}$ and moreover that $\int h(x)dx = 1$ in order to ensure that (2.26) is a proper probability distribution. We now formally justify this smoothing. Let R be a small region such that

$$q = \int_R p(x)\,dx.$$

Out of the m samples drawn from $p(x)$, the probability that k of them fall in region R is given by the binomial distribution

$$\binom{m}{k} q^k (1-q)^{m-k}.$$

The expected fraction of points falling inside the region can easily be computed from the expected value of the Binomial distribution: $\mathbb{E}[k/m] = q$. Similarly, the variance can be computed as $\mathrm{Var}[k/m] = q(1-q)/m$. As $m \to \infty$ the variance goes to 0 and hence the estimate peaks around the expectation. We can therefore set

$$k \approx mq.$$

If we assume that R is so small that $p(x)$ is constant over R, then

$$q \approx p(x) \cdot V,$$

where V is the volume of R. Rearranging we obtain

$$p(x) \approx \frac{k}{mV}. \tag{2.27}$$

Let us now set R to be a cube with side length r, and define a function

$$h(u) = \begin{cases} 1 \text{ if } |u_i| \leq \frac{1}{2} \\ 0 \text{ otherwise.} \end{cases}$$

Observe that $h\left(\frac{x-x_i}{r}\right)$ is 1 if and only if x_i lies inside a cube of size r centered around x. If we let

$$k = \sum_{i=1}^{m} h\left(\frac{x - x_i}{r}\right),$$

then one can use (2.27) to estimate p via

$$\hat{p}(x) = \frac{1}{m}\sum_{i=1}^{m} r^{-d} h\left(\frac{x - x_i}{r}\right),$$

where r^d is the volume of the hypercube of size r in d dimensions. By symmetry, we can interpret this equation as the sum over m cubes centered around m data points x_n. If we replace the cube by any smooth kernel function $h(\cdot)$ this recovers (2.26).

There exists a large variety of different kernels which can be used for the kernel density estimate. [Sil86] has a detailed description of the properties of a number of kernels. Popular choices are

$$h(x) = (2\pi)^{-\frac{1}{2}}\, e^{-\frac{1}{2}x^2} \qquad \text{Gaussian kernel} \tag{2.28}$$

$$h(x) = \tfrac{1}{2} e^{-|x|} \qquad \text{Laplace kernel} \tag{2.29}$$

$$h(x) = \tfrac{3}{4}\max(0, 1 - x^2) \qquad \text{Epanechnikov kernel} \tag{2.30}$$

$$h(x) = \tfrac{1}{2}\chi_{[-1,1]}(x) \qquad \text{Uniform kernel} \tag{2.31}$$

$$h(x) = \max(0, 1 - |x|) \qquad \text{Triangle kernel.} \tag{2.32}$$

Further kernels are the triweight and the quartic kernel which are basically powers of the Epanechnikov kernel. For practical purposes the Gaussian kernel (2.28) or the Epanechnikov kernel (2.30) are most suitable. In particular, the latter has the attractive property of compact support. This means that for any given density estimate at location x we will only need to evaluate terms $h(x_i - x)$ for which the distance $\|x_i - x\|$ is less than r. Such expansions are computationally much cheaper, in particular when we make use of fast nearest neighbor search algorithms [GIM99, IM98]. Figure 2.7 has some examples of kernels.

2.2.3 Parameter Estimation

So far we have not discussed the issue of parameter selection. It should be evident from Figure 2.6, though, that it is quite crucial to choose a good kernel width. Clearly, a kernel that is overly wide will oversmooth any fine detail that there might be in the density. On the other hand, a very narrow kernel will not be very useful, since it will be able to make statements only about the locations where we actually observed data.

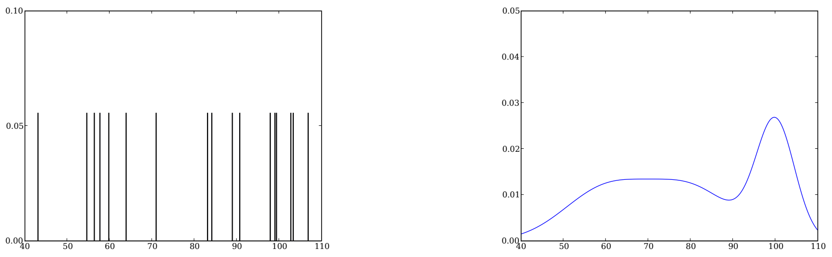


Fig. 2.5. Left: a naive density estimate given a sample of the weight of 18 persons. Right: the underlying weight distribution.

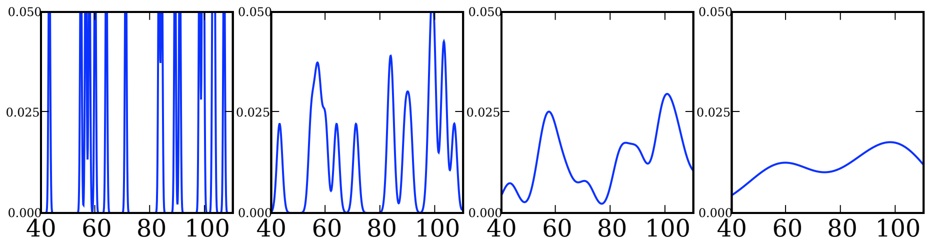


Fig. 2.6. Parzen windows density estimate associated with the 18 observations of the Figure above. From left to right: Gaussian kernel density estimate with kernel of width $0.3, 1, 3$, and 10 respectively.

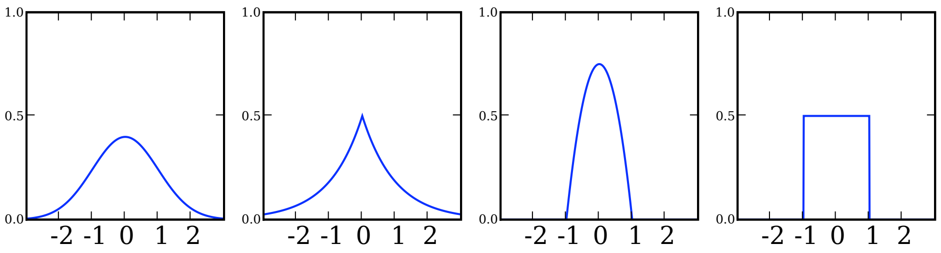


Fig. 2.7. Some kernels for Parzen windows density estimation. From left to right: Gaussian kernel, Laplace kernel, Epanechikov kernel, and uniform density.

Moreover, there is the issue of choosing a suitable kernel function. The fact that a large variety of them exists might suggest that this is a crucial issue. In practice, this turns out not to be the case and instead, the choice of a suitable kernel width is much more vital for good estimates. In other words, size matters, shape is secondary.

The problem is that we do not know which kernel width is best for the data. If the problem is one-dimensional, we might hope to be able to eyeball the size of r. Obviously, in higher dimensions this approach fails. A second option would be to choose r such that the log-likelihood of the data is maximized. It is given by

$$\log \prod_{i=1}^{m} p(x_i) = -m \log m + \sum_{i=1}^{m} \log \sum_{j=1}^{m} r^{-d} h\left(\frac{x_i - x_j}{r}\right) \tag{2.33}$$

Remark 2.13 (Log-likelihood) *We consider the logarithm of the likelihood for reasons of computational stability to prevent numerical underflow. While each term $p(x_i)$ might be within a suitable range, say 10^{-2}, the product of* 1000 *of such terms will easily exceed the exponent of floating point representations on a computer. Summing over the logarithm, on the other hand, is perfectly feasible even for large numbers of observations.*

Unfortunately computing the log-likelihood is equally infeasible: for decreasing r the only surviving terms in (2.33) are the functions $h((x_i - x_i)/r) = h(0)$, since the arguments of all other kernel functions diverge. In other words, the log-likelihood is maximized when $p(x)$ is peaked exactly at the locations where we observed the data. The graph on the left of Figure 2.6 shows what happens in such a situation.

What we just experienced is a case of *overfitting* where our model is too flexible. This led to a situation where our model was able to explain the observed data "unreasonably well", simply because we were able to adjust our parameters given the data. We will encounter this situation throughout the book. There exist a number of ways to address this problem.

Validation Set: We could use a subset of our set of observations as an *estimate* of the log-likelihood. That is, we could partition the observations into $\mathbf{X} := \{x_1, \ldots, x_n\}$ and $\mathbf{X}' := \{x_{n+1}, \ldots, x_m\}$ and use the second part for a likelihood score according to (2.33). The second set is typically called a *validation set.*

n-fold Cross-validation: Taking this idea further, note that there is no particular reason why any given x_i should belong to $\mathbf{X}$ or $\mathbf{X}'$ respectively. In fact, we could use all splits of the observations into sets $\mathbf{X}$ and $\mathbf{X}'$ to infer the quality of our estimate. While this is computationally infeasible, we could decide to split the observations into n equally sized subsets, say $\mathbf{X}_1, \ldots, \mathbf{X}_n$ and use each of them as a validation set at a time while the remainder is used to generate a density estimate.

Typically n is chosen to be 10, in which case this procedure is referred to as *10-fold cross-validation.* It is a computationally attractive procedure insofar as it does not require us to change the basic estimation algorithm. Nonetheless, computation can be costly.

Leave-one-out Estimator: At the extreme end of cross-validation we could choose $n = m$. That is, we only remove a single observation at a time and use the remainder of the data for the estimate. Using the average over the likelihood scores provides us with an even more fine-grained estimate. Denote by $p_i(x)$ the density estimate obtained by using $\mathbf{X} := \{x_1, \ldots, x_m\}$ without x_i. For a Parzen windows estimate this is given by

$$p_i(x_i) = (m-1)^{-1} \sum_{j \neq i} r^{-d} h\left(\tfrac{x_i - x_j}{r}\right) = \tfrac{m}{m-1}\left[p(x_i) - r^{-d} h(0)\right]. \tag{2.34}$$

Note that this is precisely the term $r^{-d}h(0)$ that is removed from the estimate. It is this term which led to divergent estimates for $r \to 0$. This means that the leave-one-out log-likelihood estimate can be computed easily via

$$L(\mathbf{X}) = m \log \tfrac{m}{m-1} + \sum_{i=1}^{m} \log \left[p(x_i) - r^{-d} h(0)\right]. \tag{2.35}$$

We then choose r such that $L(\mathbf{X})$ is maximized. This strategy is very robust and whenever it can be implemented in a computationally efficient manner, it is very reliable in performing model selection.

An alternative, probably more of theoretical interest, is to choose the scale r *a priori* based on the amount of data we have at our disposition. Intuitively, we need a scheme which ensures that $r \to 0$ as the number of observations increases $m \to \infty$. However, we need to ensure that this happens slowly enough that the number of observations within range r keeps on increasing in order to ensure good statistical performance. For details we refer the reader to [Sil86]. Chapter **??** discusses issues of model selection for estimators in general in considerably more detail.

2.2.4 Silverman's Rule

Assume you are an aspiring demographer who wishes to estimate the population density of a country, say Australia. You might have access to a limited census which, for a random portion of the population determines where they live. As a consequence you will obtain a relatively high number of samples

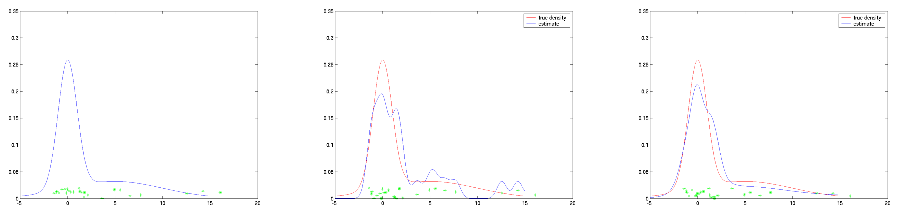


Fig. 2.8. Nonuniform density. Left: original density with samples drawn from the distribution. Middle: density estimate with a uniform kernel. Right: density estimate using Silverman's adjustment.

of city dwellers, whereas the number of people living in the countryside is likely to be very small.

If we attempt to perform density estimation using Parzen windows, we will encounter an interesting dilemma: in regions of high density (i.e. the cities) we will want to choose a narrow kernel width to allow us to model the variations in population density accurately. Conversely, in the outback, a very wide kernel is preferable, since the population there is very low. Unfortunately, this information is exactly what a density estimator itself could tell us. In other words we have a chicken and egg situation where having a good density estimate seems to be necessary to come up with a good density estimate.

Fortunately this situation can be addressed by realizing that we do not actually need to know the *density* but rather a rough estimate of the latter. This can be obtained by using information about the average distance of the k nearest neighbors of a point. One of Silverman's rules of thumb [Sil86] is to choose r_i as

$$r_i = \frac{c}{k} \sum_{x \in kNN(x_i)} \|x - x_i\| . \tag{2.36}$$

Typically c is chosen to be 0.5 and k is small, e.g. $k = 9$ to ensure that the estimate is computationally efficient. The density estimate is then given by

$$p(x) = \frac{1}{m} \sum_{i=1}^{m} r_i^{-d} h\left(\tfrac{x - x_i}{r_i}\right) . \tag{2.37}$$

Figure 2.8 shows an example of such a density estimate. It is clear that a locality dependent kernel width is better than choosing a uniformly constant kernel density estimate. However, note that this increases the computational complexity of performing a density estimate, since first the k nearest neighbors need to be found before the density estimate can be carried out.

2.2.5 Watson-Nadaraya Estimator

Now that we are able to perform density estimation we may use it to perform classification and regression. This leads us to an effective method for non-parametric data analysis, the Watson-Nadaraya estimator [Wat64, Nad65].

The basic idea is very simple: assume that we have a binary classification problem, i.e. we need to distinguish between two classes. Provided that we are able to compute density estimates $p(x)$ given a set of observations $\mathbf{X}$ we could appeal to Bayes rule to obtain

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{\frac{m_y}{m} \cdot \frac{1}{m_y} \sum_{i:y_i=y} r^{-d} h\left(\frac{x_i-x}{r}\right)}{\frac{1}{m}\sum_{i=1}^{m} r^{-d} h\left(\frac{x_i-x}{r}\right)}. \tag{2.38}$$

Here we only take the sum over all x_i with label $y_i = y$ in the numerator. The advantage of this approach is that it is very cheap to design such an estimator. After all, we only need to compute sums. The downside, similar to that of the k-nearest neighbor classifier is that it may require sums (or search) over a large number of observations. That is, evaluation of (2.38) is potentially an $O(m)$ operation. Fast tree based representations can be used to accelerate this [BKL06, KM00], however their behavior depends significantly on the dimensionality of the data. We will encounter computationally more attractive methods at a later stage.

For binary classification (2.38) can be simplified considerably. Assume that $y \in \{\pm 1\}$. For $p(y = 1|x) > 0.5$ we will choose that we should estimate $y = 1$ and in the converse case we would estimate $y = -1$. Taking the difference between twice the numerator and the denominator we can see that the function

$$f(x) = \frac{\sum_i y_i h\left(\frac{x_i-x}{r}\right)}{\sum_i h\left(\frac{x_i-x}{r}\right)} = \sum_i y_i \frac{h\left(\frac{x_i-x}{r}\right)}{\sum_i h\left(\frac{x_i-x}{r}\right)} =: \sum_i y_i w_i(x) \tag{2.39}$$

can be used to achieve the same goal since $f(x) > 0 \iff p(y = 1|x) > 0.5$. Note that $f(x)$ is a weighted combination of the labels y_i associated with weights $w_i(x)$ which depend on the proximity of x to an observation x_i. In other words, (2.39) is a smoothed-out version of the k-nearest neighbor classifier of Section 1.3.2. Instead of drawing a hard boundary at the k closest observation we use a soft weighting scheme with weights $w_i(x)$ depending on which observations are closest.

Note furthermore that the numerator of (2.39) is very similar to the simple classifier of Section 1.3.3. In fact, for kernels $k(x, x')$ such as the Gaussian RBF kernel, which are also kernels in the sense of a Parzen windows density estimate, i.e. $k(x, x') = r^{-d} h\left(\frac{x-x'}{r}\right)$ the two terms are identical. This

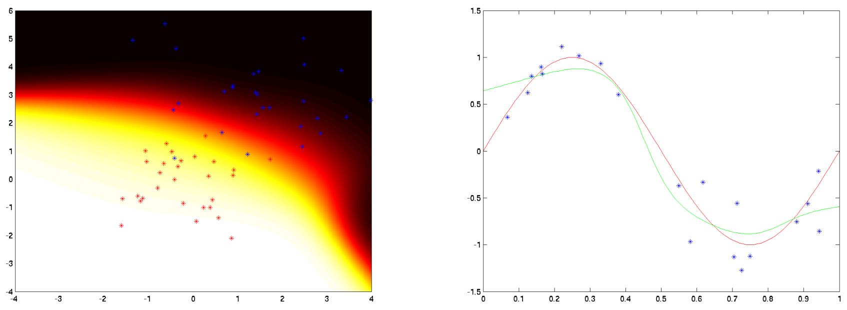


Fig. 2.9. Watson Nadaraya estimate. Left: a binary classifier. The optimal solution would be a straight line since both classes were drawn from a normal distribution with the same variance. Right: a regression estimator. The data was generated from a sinusoid with additive noise. The regression tracks the sinusoid reasonably well.

means that the Watson Nadaraya estimator provides us with an alternative explanation as to why (1.24) leads to a usable classifier.

In the same fashion as the Watson Nadaraya classifier extends the k-nearest neighbor classifier we also may construct a Watson Nadaraya regression estimator by replacing the binary labels y_i by real-valued values $y_i \in \mathbb{R}$ to obtain the regression estimator $\sum_i y_i w_i(x)$. Figure 2.9 has an example of the workings of both a regression estimator and a classifier. They are easy to use and they work well for moderately dimensional data.

2.3 Exponential Families

Distributions from the exponential family are some of the most versatile tools for statistical inference. Gaussians, Poisson, Gamma and Wishart distributions all form part of the exponential family. They play a key role in dealing with graphical models, classification, regression and conditional random fields which we will encounter in later parts of this book. Some of the reasons for their popularity are that they lead to convex optimization problems and that they allow us to describe probability distributions by linear models.

2.3.1 Basics

Densities from the exponential family are defined by

$$p(x;\theta) := p_0(x) \exp\left(\langle \phi(x), \theta \rangle - g(\theta)\right). \tag{2.40}$$

Here $p_0(x)$ is a density on $\mathcal{X}$ and is often called the base measure, $\phi(x)$ is a map from x to the sufficient statistics $\phi(x)$. θ is commonly referred to as the *natural* parameter. It lives in the space dual to $\phi(x)$. Moreover, $g(\theta)$ is a normalization constant which ensures that $p(x)$ is properly normalized. g is often referred to as the log-partition function. The name stems from physics where $Z = e^{g(\theta)}$ denotes the number of states of a physical ensemble. g can be computed as follows:

$$g(\theta) = \log \int_{\mathcal{X}} \exp\left(\langle \phi(x), \theta \rangle\right) dx. \tag{2.41}$$

Example 2.5 (Binary Model) *Assume that $\mathcal{X} = \{0; 1\}$ and that $\phi(x) = x$. In this case we have $g(\theta) = \log\left[e^0 + e^\theta\right] = \log\left[1 + e^\theta\right]$. It follows that $p(x = 0; \theta) = \frac{1}{1+e^\theta}$ and $p(x = 1; \theta) = \frac{e^\theta}{1+e^\theta}$. In other words, by choosing different values of θ one can recover different Bernoulli distributions.*

One of the convenient properties of exponential families is that the log-partition function g can be used to generate moments of the distribution itself simply by taking derivatives.

Theorem 2.14 (Log partition function) *The function $g(\theta)$ is convex. Moreover, the distribution $p(x; \theta)$ satisfies*

$$\nabla_\theta g(\theta) = \mathbf{E}_x\left[\phi(x)\right] \text{ and } \nabla_\theta^2 g(\theta) = \mathrm{Var}_x\left[\phi(x)\right]. \tag{2.42}$$

Proof Note that $\nabla_\theta^2 g(\theta) = \mathrm{Var}_x\left[\phi(x)\right]$ implies that g is convex, since the covariance matrix is positive semidefinite. To show (2.42) we expand

$$\nabla_\theta g(\theta) = \frac{\int_{\mathcal{X}} \phi(x) \exp\langle \phi(x), \theta \rangle\, dx}{\int_{\mathcal{X}} \exp\langle \phi(x), \theta \rangle} = \int \phi(x) p(x; \theta) dx = \mathbf{E}_x\left[\phi(x)\right]. \tag{2.43}$$

Next we take the second derivative to obtain

$$\nabla_\theta^2 g(\theta) = \int_{\mathcal{X}} \phi(x) \left[\phi(x) - \nabla_\theta g(\theta)\right]^\top p(x; \theta) dx \tag{2.44}$$

$$= \mathbf{E}_x\left[\phi(x)\phi(x)^\top\right] - \mathbf{E}_x\left[\phi(x)\right] \mathbf{E}_x\left[\phi(x)\right]^\top \tag{2.45}$$

which proves the claim. For the first equality we used (2.43). For the second line we used the definition of the variance. ■

One may show that higher derivatives $\nabla_\theta^n g(\theta)$ generate higher order cumulants of $\phi(x)$ under $p(x; \theta)$. This is why g is often also referred as the cumulant-generating function. Note that in general, computation of $g(\theta)$ is nontrivial since it involves solving a highdimensional integral. For many cases, in fact, the computation is NP hard, for instance when $\mathcal{X}$ is the domain of permutations [FJ95]. Throughout the book we will discuss a number of approximation techniques which can be applied in such a case.

Let us briefly illustrate (2.43) using the binary model of Example 2.5. We have that $\nabla_\theta = \frac{e^\theta}{1+e^\theta}$ and $\nabla^2_\theta = \frac{e^\theta}{(1+e^\theta)^2}$. This is exactly what we would have obtained from direct computation of the mean $p(x=1;\theta)$ and variance $p(x=1;\theta) - p(x=1;\theta)^2$ subject to the distribution $p(x;\theta)$.

2.3.2 Examples

A large number of densities are members of the exponential family. Note, however, that in statistics it is not common to express them in the dot product formulation for historic reasons and for reasons of notational compactness. We discuss a number of common densities below and show why they can be written in terms of an exponential family. A detailed description of the most commonly occurring types are given in a table.

Gaussian Let $x, \mu \in \mathbb{R}^d$ and let $\Sigma \in \mathbb{R}^{d\times d}$ where $\Sigma \succ 0$, that is, Σ is a positive definite matrix. In this case the normal distribution can be expressed via

$$\begin{aligned} p(x) &= (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \qquad (2.46)\\ &= \exp\left(x^\top \left[\Sigma^{-1}\mu\right] + \operatorname{tr}\left(\left[-\frac{1}{2}xx^\top\right]\left[\Sigma^{-1}\right]\right) - c(\mu,\Sigma)\right) \end{aligned}$$

where $c(\mu,\Sigma) = \frac{1}{2}\mu^\top\Sigma^{-1}\mu + \frac{d}{2}\log 2\pi + \frac{1}{2}\log|\Sigma|$. By combining the terms in x into $\phi(x) := (x, -\frac{1}{2}xx^\top)$ we obtain the sufficient statistics of x. The corresponding linear coefficients $(\Sigma^{-1}\mu, \Sigma^{-1})$ constitute the natural parameter θ. All that remains to be done to express $p(x)$ in terms of (2.40) is to rewrite $g(\theta)$ in terms of $c(\mu,\Sigma)$. The summary table on the following page contains details.

Multinomial Another popular distribution is one over k discrete events. In this case $\mathcal{X} = \{1,\ldots,k\}$ and we have in completely generic terms $p(x) = \pi_x$ where $\pi_x \geq 0$ and $\sum_x \pi_x = 1$. Now denote by $e_x \in \mathbb{R}^k$ the x-th unit vector of the canonical basis, that is $\langle e_x, e_{x'}\rangle = 1$ if $x = x'$ and 0 otherwise. In this case we may rewrite $p(x)$ via

$$p(x) = \pi_x = \exp\left(\langle e_x, \log\pi\rangle\right) \qquad (2.47)$$

where $\log\pi = (\log\pi_1,\ldots,\log\pi_k)$. In other words, we have succeeded in rewriting the distribution as a member of the exponential family where $\phi(x) = e_x$ and where $\theta = \log \pi$. Note that in this definition θ is restricted to a $k-1$ dimensional manifold (the k dimensional probability simplex). If we relax those constraints we need to ensure that $p(x)$ remains normalized. Details are given in the summary table.

Poisson This distribution is often used to model distributions over discrete events. For instance, the number of raindrops which fall on a given surface area in a given amount of time, the number of stars in a given volume of space, or the number of Prussian soldiers killed by horse-kicks in the Prussian cavalry all follow this distribution. It is given by

$$p(x) = \frac{e^{-\lambda}\lambda^x}{x!} = \frac{1}{x!}\exp\left(x \log \lambda - \lambda\right) \text{ where } x \in \mathbb{N}_0\,. \tag{2.48}$$

By defining $\phi(x) = x$ we obtain an exponential families model. Note that things are a bit less trivial here since $\frac{1}{x!}$ is the nonuniform counting *measure* on $\mathbb{N}_0$. The case of the uniform measure which leads to the exponential distribution is discussed in Problem 2.16.

The reason why many discrete processes follow the Poisson distribution is that it can be seen as the limit over the average of a large number of Bernoulli draws: denote by $z \in \{0, 1\}$ a random variable with $p(z = 1) = \alpha$. Moreover, denote by z_n the sum over n draws from this random variable. In this case z_n follows the multinomial distribution with $p(z_n = k) = \binom{n}{k}\alpha^k(1-\alpha)^{n-k}$. Now assume that we let $n \to \infty$ such that the expected value of z_n remains constant. That is, we rescale $\alpha = \frac{\lambda}{n}$. In this case we have

$$\begin{aligned} p(z_n = k) &= \frac{n!}{(n-k)!k!}\frac{\lambda^k}{n^k}\left(1 - \frac{\lambda}{n}\right)^{n-k} \\ &= \frac{\lambda^k}{k!}\left(1 - \frac{\lambda}{n}\right)^n\left[\frac{n!}{n^k(n-k)!}\left(1 - \frac{\lambda}{n}\right)^k\right] \end{aligned} \tag{2.49}$$

For $n \to \infty$ the second term converges to $e^{-\lambda}$. The third term converges to 1, since we have a product of only $2k$ terms, each of which converge to 1. Using the exponential families notation we may check that $\mathbf{E}[x] = \lambda$ and that moreover also $\mathrm{Var}[x] = \lambda$.

Beta This is a distribution on the unit interval $\mathcal{X} = [0, 1]$ which is very versatile when it comes to modelling unimodal and bimodal distributions. It is given by

$$p(x) = x^{a-1}(1-x)^{b-1}\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}. \tag{2.50}$$

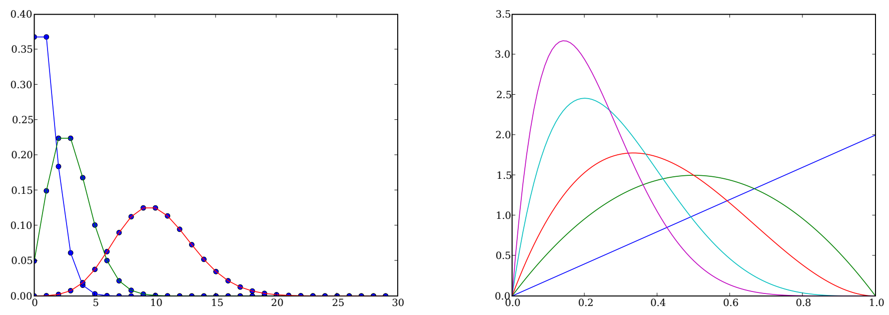


Fig. 2.10. Left: Poisson distributions with $\lambda = \{1, 3, 10\}$. Right: Beta distributions with $a = 2$ and $b \in \{1, 2, 3, 5, 7\}$. Note how with increasing b the distribution becomes more peaked close to the origin.

Taking logarithms we see that this, too, is an exponential families distribution, since $p(x) = \exp((a-1)\log x + (b-1)\log(1-x) + \log\Gamma(a+b) - \log\Gamma(a) - \log\Gamma(b))$.

Figure 2.10 has a graphical description of the Poisson distribution and the Beta distribution. For a more comprehensive list of exponential family distributions see the table below and [Fel71, FT94, MN83]. In principle any map $\phi(x)$, domain $\mathcal{X}$ with underlying measure μ are suitable, as long as the log-partition function $g(\theta)$ can be computed efficiently.

Theorem 2.15 (Convex feasible domain) *The domain of definition Θ of $g(\theta)$ is convex.*

Proof By construction g is convex and differentiable everywhere. Hence the below-sets for all values c with $\{x|g(x) \leq c\}$ exist. Consequently the domain of definition is convex. ■

Having a convex function is very valuable when it comes to parameter inference since convex minimization problems have unique minimum values and global minima. We will discuss this notion in more detail when designing maximum likelihood estimators.

2.4 Estimation

In many statistical problems the challenge is to estimate parameters of interest. For instance, in the context of exponential families, we may want to estimate a parameter $\hat{\theta}$ such that it is close to the "true" parameter θ^* in the distribution. While the problem is fully general, we will describe the

Name	Domain $\mathcal{X}$	Measure	$\phi(x)$	$g(\theta)$	Domain Θ
Bernoulli	$\{0,1\}$	Counting	x	$\log\left(1+e^{\theta}\right)$	$\mathbb{R}$
Multinomial	$\{1..N\}$	Counting	e_x	$\log\sum_{i=1}^{N} e^{\theta_i}$	$\mathbb{R}^N$
Exponential	$\mathbb{N}_0^+$	Counting	x	$-\log\left(1-e^{\theta}\right)$	$(-\infty,0)$
Poisson	$\mathbb{N}_0^+$	$\frac{1}{x!}$	x	e^{θ}	$\mathbb{R}$
Laplace	$[0,\infty)$	Lebesgue	x	$\log\theta$	$(-\infty,0)$
Gaussian	$\mathbb{R}$	Lebesgue	$\left(x,-\frac{1}{2}x^2\right)$	$\frac{1}{2}\log 2\pi-\frac{1}{2}\log\theta_2+\frac{1}{2}\frac{\theta_1^2}{\theta_2}$	$\mathbb{R}\times(0,\infty)$
	$\mathbb{R}^n$	Lebesgue	$\left(x,-\frac{1}{2}xx^\top\right)$	$\frac{n}{2}\log 2\pi-\frac{1}{2}\log\lvert\theta_2\rvert+\frac{1}{2}\theta_1^\top\theta_2^{-1}\theta_1$	$\mathbb{R}^n\times\mathfrak{C}_n$
Inverse Normal	$[0,\infty)$	$x^{-\frac{3}{2}}$	$\left(-x,-\frac{1}{x}\right)$	$\frac{1}{2}\log\pi-2\sqrt{\theta_1\theta_2}-\frac{1}{2}\log\theta_2$	$(0,\infty)^2$
Beta	$[0,1]$	$\frac{1}{x(1-x)}$	$(\log x,\log(1-x))$	$\log\frac{\Gamma(\theta_1)\Gamma(\theta_2)}{\Gamma(\theta_1+\theta_2)}$	$\mathbb{R}^2$
Gamma	$[0,\infty)$	$\frac{1}{x}$	$(\log x,-x)$	$\log\Gamma(\theta_1)-\theta_1\log\theta_2$	$(0,\infty)^2$
Wishart	$\mathfrak{C}_n$	$\lvert X\rvert^{-\frac{n+1}{2}}$	$\left(\log\lvert x\rvert,-\frac{1}{2}x\right)$	$-\theta_1\log\lvert\theta_2\rvert+\theta_1 n\log 2$	$\mathbb{R}\times\mathfrak{C}_n$
				$+\sum_{i=1}^{n}\log\Gamma\left(\theta_1+\frac{1-i}{2}\right)$	
Dirichlet	S_n	$\left(\prod_{i=1}^{n} x_i\right)^{-1}$	$(\log x_1,\ldots,\log x_n)$	$\sum_{i=1}^{n}\log\Gamma(\theta_i)-\log\Gamma\left(\sum_{i=1}^{n}\theta_i\right)$	$(\mathbb{R}^+)^n$
Inverse χ^2	$\mathbb{R}^+$	$e^{-\frac{1}{2x}}$	$-\log x$	$(\theta-1)\log 2+\log(\theta-1)$	$(0,\infty)$
Logarithmic	$\mathbb{N}$	$\frac{1}{x}$	x	$\log(-\log(1-e^{\theta}))$	$(-\infty,0)$
Conjugate	Θ	Lebesgue	$(\theta,-g(\theta))$	generic	

$\mathfrak{S}_n$ denotes the probability simplex in n dimensions. $\mathfrak{C}_n$ is the cone of positive semidefinite matrices in $\mathbb{R}^{n\times n}$.

relevant steps in obtaining estimates for the special case of the exponential family. This is done for two reasons — firstly, exponential families are an important special case and we will encounter slightly more complex variants on the reasoning in later chapters of the book. Secondly, they are of a sufficiently simple form that we are able to show a *range* of different techniques. In more advanced applications only a small subset of those methods may be practically feasible. Hence exponential families provide us with a working example based on which we can compare the consequences of a number of different techniques.

2.4.1 Maximum Likelihood Estimation

Whenever we have a distribution $p(x;\theta)$ parametrized by some parameter θ we may use data to find a value of θ which maximizes the *likelihood* that the data would have been generated by a distribution with this choice of parameter.

For instance, assume that we observe a set of temperature measurements $\mathbf{X} = \{x_1, \ldots, x_m\}$. In this case, we could try finding a normal distribution such that the likelihood $p(\mathbf{X};\theta)$ of the data under the assumption of a normal distribution is maximized. Note that this does *not* imply in any way that the temperature measurements are actually drawn from a normal distribution. Instead, it means that we are attempting to find the Gaussian which fits the data in the best fashion.

While this distinction may appear subtle, it is critical: we do *not* assume that our model accurately reflects reality. Instead, we simply try doing the best possible job at modeling the data given a specified model class. Later we will encounter alternative approaches at estimation, namely Bayesian methods, which *make* the assumption that our model ought to be able to describe the data accurately.

Definition 2.16 (Maximum Likelihood Estimator) *For a model $p(\cdot;\theta)$ parametrized by θ and observations $\mathbf{X}$ the maximum likelihood estimator (MLE) is*

$$\hat{\theta}_{\mathrm{ML}}[\mathbf{X}] := \operatorname*{argmax}_{\theta} p(\mathbf{X};\theta). \tag{2.51}$$

In the context of exponential families this leads to the following procedure: given m observations drawn iid from some distribution, we can express the joint likelihood as

$$p(\mathbf{X};\theta) = \prod_{i=1}^{m} p(x_i;\theta) = \prod_{i=1}^{m} \exp\left(\langle\phi(x_i),\theta\rangle - g(\theta)\right) \tag{2.52}$$

$$= \exp\left(m\left(\langle\mu[\mathbf{X}],\theta\rangle - g(\theta)\right)\right) \tag{2.53}$$

$$\text{where } \mu[\mathbf{X}] := \frac{1}{m}\sum_{i=1}^{m}\phi(x_i). \tag{2.54}$$

Here $\mu[\mathbf{X}]$ is the empirical average of the map $\phi(x)$. Maximization of $p(\mathbf{X};\theta)$ is equivalent to minimizing the negative log-likelihood $-\log p(\mathbf{X};\theta)$. The latter is a common practical choice since for independently drawn data, the product of probabilities decomposes into the sum of the logarithms of individual likelihoods. This leads to the following objective function to be minimized

$$-\log p(\mathbf{X};\theta) = m\left[g(\theta) - \langle\theta,\mu[\mathbf{X}]\rangle\right] \tag{2.55}$$

Since $g(\theta)$ is convex and $\langle\theta,\mu[\mathbf{X}]\rangle$ is linear in θ, it follows that minimization of (2.55) is a convex optimization problem. Using Theorem 2.14 and the first order optimality condition $\nabla_\theta g(\theta) = \mu[\mathbf{X}]$ for (2.55) implies that

$$\theta = \left[\nabla_\theta g\right]^{-1}(\mu[\mathbf{X}]) \text{ or equivalently } \mathbf{E}_{x\sim p(x;\theta)}[\phi(x)] = \nabla_\theta g(\theta) = \mu[\mathbf{X}]. \tag{2.56}$$

Put another way, the above conditions state that we aim to find the distribution $p(x;\theta)$ which has the same expected value of $\phi(x)$ as what we observed empirically via $\mu[\mathbf{X}]$. Under very mild technical conditions a solution to (2.56) exists.

In general, (2.56) cannot be solved analytically. In certain special cases, though, this is easily possible. We discuss two such choices in the following: Multinomial and Poisson distributions.

Example 2.6 (Poisson Distribution) *For the Poisson distribution*[1] *where* $p(x;\theta) = \frac{1}{x!}\exp(\theta x - e^\theta)$ *it follows that* $g(\theta) = e^\theta$ *and* $\phi(x) = x$. *This allows us to solve (2.56) in closed form using*

$$\nabla_\theta g(\theta) = e^\theta = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ and hence } \theta = \log\sum_{i=1}^{m} x_i - \log m. \tag{2.57}$$

[1] Often the Poisson distribution is specified using $\lambda := \log\theta$ as its rate parameter. In this case we have $p(x;\lambda) = \lambda^x e^{-\lambda}/x!$ as its parametrization. The advantage of the *natural* parametrization using θ is that we can directly take advantage of the properties of the log-partition function as generating the cumulants of x.

Example 2.7 (Multinomial Distribution) *For the multinomial distribution the log-partition function is given by $g(\theta) = \log \sum_{i=1}^{N} e^{\theta_i}$, hence we have that*

$$\nabla_i g(\theta) = \frac{e^{\theta_i}}{\sum_{j=1}^{N} e^{\theta_j}} = \frac{1}{m} \sum_{j=1}^{m} \{x_j = i\} . \tag{2.58}$$

It is easy to check that (2.58) is satisfied for $e^{\theta_i} = \sum_{j=1}^{m} \{x_j = i\}$. In other words, the MLE for a discrete distribution simply given by the empirical frequencies of occurrence.

The multinomial setting also exhibits two rather important aspects of exponential families: firstly, choosing $\theta_i = c + \log \sum_{i=1}^{m} \{x_j = i\}$ for any $c \in \mathbb{R}$ will lead to an equivalent distribution. This is the case since the sufficient statistic $\phi(x)$ is not minimal. In our context this means that the coordinates of $\phi(x)$ are linearly dependent — for any x we have that $\sum_j [\phi(x)]_j = 1$, hence we could eliminate one dimension. This is precisely the additional degree of freedom which is reflected in the scaling freedom in θ.

Secondly, for data where some events do not occur at all, the expression $\log \left[\sum_{j=1}^{m} \{x_j = i\}\right] = \log 0$ is ill defined. This is due to the fact that this particular set of counts occurs on the boundary of the convex set within which the natural parameters θ are well defined. We will see how different types of priors can alleviate the issue.

Using the MLE is not without problems. As we saw in Figure 2.1, convergence can be slow, since we are not using any side information. The latter can provide us with problems which are both numerically better conditioned and which show better convergence, *provided that our assumptions are accurate*. Before discussing a Bayesian approach to estimation, let us discuss basic statistical properties of the estimator.

2.4.2 Bias, Variance and Consistency

When designing any estimator $\hat{\theta}(\mathbf{X})$ we would like to obtain a number of desirable properties: in general it should not be biased towards a particular solution unless we have good reason to believe that this solution should be preferred. Instead, we would like the estimator to recover, at least on average, the "correct" parameter, should it exist. This can be formalized in the notion of an *unbiased* estimator.

Secondly, we would like that, even if no correct parameter can be found, e.g. when we are trying to fit a Gaussian distribution to data which is not normally distributed, that we will converge to the best possible parameter choice as we obtain more data. This is what is understood by *consistency*.

Finally, we would like the estimator to achieve low bias and near-optimal estimates as quickly as possible. The latter is measured by the *efficiency* of an estimator. In this context we will encounter the Cramér-Rao bound which controls the best possible rate at which an estimator can achieve this goal. Figure 2.11 gives a pictorial description.

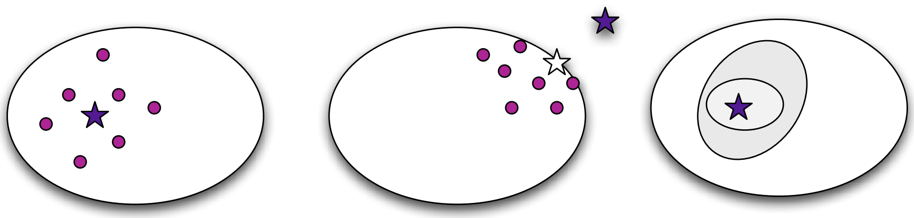


Fig. 2.11. Left: unbiased estimator; the estimates, denoted by circles have as mean the true parameter, as denoted by a star. Middle: consistent estimator. While the true model is not within the class we consider (as denoted by the ellipsoid), the estimates converge to the white star which is the best model within the class that approximates the true model, denoted by the solid star. Right: different estimators have different regions of uncertainty, as made explicit by the ellipses around the true parameter (solid star).

Definition 2.17 (Unbiased Estimator) *An estimator $\hat{\theta}[\mathbf{X}]$ is unbiased if for all θ where $\mathbf{X} \sim p(\mathbf{X}; \theta)$ we have $\mathbf{E}_{\mathbf{X}}[\hat{\theta}[\mathbf{X}]] = \theta$.*

In other words, in expectation the parameter estimate matches the true parameter. Note that this only makes sense if a true parameter actually *exists*. For instance, if the data is Poisson distributed and we attempt modeling it by a Gaussian we will obviously not obtain unbiased estimates.

For finite sample sizes MLE is often *biased*. For instance, for the normal distribution the variance estimates carry bias $O(m^{-1})$. See problem 2.19 for details. In general, under fairly mild conditions, MLE is asymptotically unbiased [DGL96]. We prove this for exponential families. For more general settings the proof depends on the dimensionality and smoothness of the family of densities that we have at our disposition.

Theorem 2.18 (MLE for Exponential Families) *Assume that $\mathbf{X}$ is an m-sample drawn iid from $p(x; \theta)$. The estimate $\hat{\theta}[\mathbf{X}] = g^{-1}(\mu[\mathbf{X}])$ is asymptotically normal with*

$$m^{-\frac{1}{2}}[\hat{\theta}[\mathbf{X}] - \theta] \to \mathcal{N}(0, \left[\nabla^2_{\theta} g(\theta)\right]^{-1}). \tag{2.59}$$

In other words, the estimate $\hat{\theta}[\mathbf{X}]$ is asymptotically normal, it converges to the true parameter θ, and moreover, the variance at the correct parameter is given by the inverse of the covariance matrix of the data, as given by the second derivative of the log-partition function $\nabla^2_\theta g(\theta)$.

Proof Denote by $\mu = \nabla_\theta g(\theta)$ the true mean. Moreover, note that $\nabla^2_\theta g(\theta)$ is the covariance of the data drawn from $p(x;\theta)$. By the central limit theorem (Theorem 2.3) we have that $n^{-\frac{1}{2}}[\mu[\mathbf{X}] - \mu] \to \mathcal{N}(0, \nabla^2_\theta g(\theta))$.

Now note that $\hat{\theta}[\mathbf{X}] = [\nabla_\theta g]^{-1}(\mu[\mathbf{X}])$. Therefore, by the delta method (Theorem 2.5) we know that $\hat{\theta}[\mathbf{X}]$ is also asymptotically normal. Moreover, by the inverse function theorem the Jacobian of g^{-1} satisfies $\nabla_\mu [\nabla_\theta g]^{-1}(\mu) = \left[\nabla^2_\theta g(\theta)\right]^{-1}$. Applying Slutsky's theorem (Theorem 2.4) proves the claim. ■

Now that we established the asymptotic properties of the MLE for exponential families it is only natural to ask how much variation one may expect in $\hat{\theta}[\mathbf{X}]$ when performing estimation. The Cramer-Rao bound governs this.

Theorem 2.19 (Cramér and Rao [Rao73]) *Assume that $\mathbf{X}$ is drawn from $p(\mathbf{X};\theta)$ and let $\hat{\theta}[\mathbf{X}]$ be an asymptotically unbiased estimator. Denote by I the Fisher information matrix and by B the variance of $\hat{\theta}[\mathbf{X}]$ where*

$$I := \operatorname{Cov}\left[\nabla_\theta \log p(\mathbf{X};\theta)\right] \text{ and } B := \operatorname{Var}\left[\hat{\theta}[\mathbf{X}]\right]. \tag{2.60}$$

In this case $\det IB \geq 1$ for all estimators $\hat{\theta}[\mathbf{X}]$.

Proof We prove the claim for the scalar case. The extension to matrices is straightforward. Using the Cauchy-Schwarz inequality we have

$$\operatorname{Cov}^2\left[\nabla_\theta \log p(\mathbf{X};\theta), \hat{\theta}[\mathbf{X}]\right] \leq \operatorname{Var}\left[\nabla_\theta \log p(\mathbf{X};\theta)\right] \operatorname{Var}\left[\hat{\theta}[\mathbf{X}]\right] = IB. \tag{2.61}$$

Note that at the true parameter the expected log-likelihood score vanishes

$$\mathbf{E}_\mathbf{X}[\nabla_\theta \log p(\mathbf{X};\theta)] = \nabla_\theta \int p(\mathbf{X};\theta) d\mathbf{X} = \nabla_\theta 1 = 0. \tag{2.62}$$

Hence we may simplify the covariance formula by dropping the means via

$$\begin{aligned}
\operatorname{Cov}\left[\nabla_\theta \log p(\mathbf{X};\theta), \hat{\theta}[\mathbf{X}]\right] &= \mathbf{E}_\mathbf{X}\left[\nabla_\theta \log p(\mathbf{X};\theta)\hat{\theta}[\mathbf{X}]\right] \\
&= \int p(\mathbf{X};\theta)\hat{\theta}(\mathbf{X})\nabla_\theta \log p(\mathbf{X};\theta) d\theta \\
&= \nabla_\theta \int p(\mathbf{X};\theta)\hat{\theta}(\mathbf{X}) d\mathbf{X} = \nabla_\theta \theta = 1.
\end{aligned}$$

Here the last equality follows since we may interchange integration by $\mathbf{X}$ and the derivative with respect to θ. ■

The Cramér-Rao theorem implies that there is a limit to how well we may estimate a parameter given finite amounts of data. It is also a yardstick by which we may measure how efficiently an estimator uses data. Formally, we define the efficiency as the quotient between actual performance and the Cramér-Rao bound via

$$e := 1/\det IB. \tag{2.63}$$

The closer e is to 1, the lower the variance of the corresponding estimator $\hat{\theta}(\mathbf{X})$. Theorem 2.18 implies that for exponential families MLE is asymptotically efficient. It turns out to be generally true.

Theorem 2.20 (Efficiency of MLE [Cra46, GW92, Ber85]) *The maximum likelihood estimator is asymptotically efficient ($e = 1$).*

So far we only discussed the behavior of $\hat{\theta}[\mathbf{X}]$ whenever there *exists* a true θ generating $p(\theta; \mathbf{X})$. If this is not true, we need to settle for less: how well $\hat{\theta}[\mathbf{X}]$ approaches the best possible choice of within the given model class. Such behavior is referred to as consistency. Note that it is not possible to define consistency *per se*. For instance, we may ask whether $\hat{\theta}$ converges to the optimal parameter θ^*, or whether $p(x; \hat{\theta})$ converges to the optimal density $p(x; \theta^*)$, and with respect to which norm. Under fairly general conditions this turns out to be true for finite-dimensional parameters and smoothly parametrized densities. See [DGL96, vdG00] for proofs and further details.

2.4.3 A Bayesian Approach

The analysis of the Maximum Likelihood method might suggest that inference is a solved problem. After all, in the limit, MLE is unbiased and it exhibits as small variance as possible. Empirical results using a *finite* amount of data, as present in Figure 2.1 indicate otherwise.

While not making any assumptions can lead to interesting and general theorems, it ignores the fact that in practice we almost always have some idea about what to expect of our solution. It would be foolish to ignore such additional information. For instance, when trying to determine the voltage of a battery, it is reasonable to expect a measurement in the order of 1.5V or less. Consequently such *prior* knowledge should be incorporated into the estimation process. In fact, the use of side information to guide estimation turns out to be *the* tool to building estimators which work well in high dimensions.

Recall Bayes' rule (1.15) which states that $p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$. In our context this means that if we are interested in the posterior probability of θ

assuming a particular value, we may obtain this using the likelihood (often referred to as evidence) of x having been generated by θ via $p(x|\theta)$ and our prior belief $p(\theta)$ that θ might be chosen in the distribution generating x. Observe the subtle but important difference to MLE: instead of treating θ as a parameter of a density model, we treat θ as an unobserved random variable which we may attempt to infer given the observations $\mathbf{X}$.

This can be done for a number of different purposes: we might want to infer the most likely value of the parameter given the posterior distribution $p(\theta|\mathbf{X})$. This is achieved by

$$\hat{\theta}_{\mathrm{MAP}}(\mathbf{X}) := \operatorname*{argmax}_{\theta} p(\theta|\mathbf{X}) = \operatorname*{argmin}_{\theta} - \log p(\mathbf{X}|\theta) - \log p(\theta). \tag{2.64}$$

The second equality follows since $p(\mathbf{X})$ does not depend on θ. This estimator is also referred to as the *Maximum a Posteriori*, or MAP estimator. It differs from the maximum likelihood estimator by adding the negative log-prior to the optimization problem. For this reason it is sometimes also referred to as Penalized MLE. Effectively we are penalizing unlikely choices θ via $-\log p(\theta)$.

Note that using $\hat{\theta}_{\mathrm{MAP}}(\mathbf{X})$ as the parameter of choice is not quite accurate. After all, we can only infer a distribution over θ and in general there is no guarantee that the posterior is indeed concentrated around its mode. A more accurate treatment is to use the *distribution* $p(\theta|\mathbf{X})$ directly via

$$p(x|\mathbf{X}) = \int p(x|\theta)p(\theta|\mathbf{X})d\theta. \tag{2.65}$$

In other words, we integrate out the unknown parameter θ and obtain the density estimate directly. As we will see, it is generally impossible to solve (2.65) exactly, an important exception being conjugate priors. In the other cases one may resort to sampling from the posterior distribution to approximate the integral.

While it is possible to design a wide variety of prior distributions, this book focuses on two important families: norm-constrained prior and conjugate priors. We will encounter them throughout, the former sometimes in the guise of regularization and Gaussian Processes, the latter in the context of exchangeable models such as the Dirichlet Process.

Norm-constrained priors take on the form

$$p(\theta) \propto \exp(-\lambda \left\|\theta - \theta_0\right\|_p^d) \text{ for } p, d \geq 1 \text{ and } \lambda > 0. \tag{2.66}$$

That is, they restrict the deviation of the parameter value θ from some guess θ_0. The intuition is that extreme values of θ are much less likely than more moderate choices of θ which will lead to more smooth and even distributions $p(x|\theta)$.

A popular choice is the Gaussian prior which we obtain for $p = d = 1$ and $\lambda = 1/2\sigma^2$. Typically one sets $\theta_0 = 0$ in this case. Note that in (2.66) we did not spell out the normalization of $p(\theta)$ — in the context of MAP estimation this is not needed since it simply becomes a constant offset in the optimization problem (2.64). We have

$$\hat{\theta}_{\text{MAP}}[\mathbf{X}] = \operatorname*{argmin}_{\theta} m\left[g(\theta) - \langle \theta, \mu[\mathbf{X}] \rangle\right] + \lambda \left\|\theta - \theta_0\right\|_p^d \tag{2.67}$$

For $d, p \geq 1$ and $\lambda \geq 0$ the resulting optimization problem is *convex* and it has a unique solution. Moreover, very efficient algorithms exist to solve this problem. We will discuss this in detail in Chapter **??**. Figure 2.12 shows the regions of equal prior probability for a range of different norm-constrained priors.

As can be seen from the diagram, the choice of the norm can have profound consequences on the solution. That said, as we will show in Chapter **??**, the estimate $\hat{\theta}_{\text{MAP}}$ is well concentrated and converges to the optimal solution under fairly general conditions.

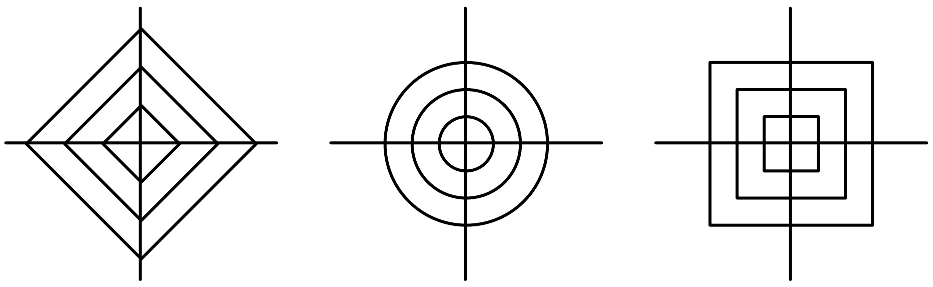


Fig. 2.12. From left to right: regions of equal prior probability in $\mathbb{R}^2$ for priors using the ℓ_1, ℓ_2 and ℓ_∞ norm. Note that only the ℓ_2 norm is invariant with regard to the coordinate system. As we shall see later, the ℓ_1 norm prior leads to solutions where only a small number of coordinates is nonzero.

An alternative to norm-constrained priors are *conjugate* priors. They are designed such that the posterior $p(\theta|\mathbf{X})$ has the same functional form as the prior $p(\theta)$. In exponential families such priors are defined via

$$p(\theta|n, \nu) = \exp\left(\langle n\nu, \theta\rangle - ng(\theta) - h(\nu, n)\right) \text{ where} \tag{2.68}$$

$$h(\nu, n) = \log \int \exp\left(\langle n\nu, \theta\rangle - ng(\theta)\right) d\theta. \tag{2.69}$$

Note that $p(\theta|n,\nu)$ itself is a member of the exponential family with the feature map $\phi(\theta) = (\theta, -g(\theta))$. Hence $h(\nu, n)$ is *convex* in $(n\nu, n)$. Moreover, the posterior distribution has the form

$$p(\theta|\mathbf{X}) \propto p(\mathbf{X}|\theta)p(\theta|n,\nu) \propto \exp\left(\langle m\mu[\mathbf{X}] + n\nu, \theta\rangle - (m+n)g(\theta)\right). \quad (2.70)$$

That is, the posterior distribution has the same form as a conjugate prior with parameters $\frac{m\mu[\mathbf{X}]+n\nu}{m+n}$ and $m+n$. In other words, n acts like a phantom sample size and ν is the corresponding mean parameter. Such an interpretation is reasonable given our desire to design a prior which, when combined with the likelihood remains in the same model class: we treat prior knowledge as having observed virtual data beforehand which is then added to the actual set of observations. In this sense data and prior become completely equivalent — we obtain our knowledge either from actual observations or from virtual observations which describe our belief into how the data generation process is supposed to behave.

Eq. (2.70) has the added benefit of allowing us to provide an exact normalized version of the posterior. Using (2.68) we obtain that

$$p(\theta|\mathbf{X}) = \exp\left(\langle m\mu[\mathbf{X}] + n\nu, \theta\rangle - (m+n)g(\theta) - h\left(\tfrac{m\mu[\mathbf{X}]+n\nu}{m+n}, m+n\right)\right).$$

The main remaining challenge is to compute the normalization h for a range of important conjugate distributions. The table on the following page provides details. Besides attractive algebraic properties, conjugate priors also have a second advantage — the integral (2.65) can be solved exactly:

$$p(x|\mathbf{X}) = \int \exp\left(\langle\phi(x), \theta\rangle - g(\theta)\right) \times \\ \exp\left(\langle m\mu[\mathbf{X}] + n\nu, \theta\rangle - (m+n)g(\theta) - h\left(\tfrac{m\mu[\mathbf{X}]+n\nu}{m+n}, m+n\right)\right) d\theta$$

Combining terms one may check that the integrand amounts to the normalization in the conjugate distribution, albeit $\phi(x)$ added. This yields

$$p(x|\mathbf{X}) = \exp\left(h\left(\tfrac{m\mu[\mathbf{X}]+n\nu+\phi(x)}{m+n+1}, m+n+1\right) - h\left(\tfrac{m\mu[\mathbf{X}]+n\nu}{m+n}, m+n\right)\right)$$

Such an expansion is very useful whenever we would like to draw x from $p(x|\mathbf{X})$ without the need to obtain an instantiation of the latent variable θ. We provide explicit expansions in appendix **??**. [GS04] use the fact that θ can be integrated out to obtain what is called a collapsed Gibbs sampler for topic models [BNJ03].

2.4.4 An Example

Assume we would like to build a language model based on available documents. For instance, a linguist might be interested in estimating the frequency of words in Shakespeare's collected works, or one might want to compare the change with respect to a collection of webpages. While models describing documents by treating them as bags of words which all have been obtained independently of each other are exceedingly simple, they are valuable for quick-and-dirty content filtering and categorization, e.g. a spam filter on a mail server or a content filter for webpages.

Hence we model a document d as a multinomial distribution: denote by w_i for $i \in \{1, \ldots, m_d\}$ the words in d. Moreover, denote by $p(w|\theta)$ the probability of occurrence of word w, then under the assumption that the words are independently drawn, we have

$$p(d|\theta) = \prod_{i=1}^{m_d} p(w_i|\theta). \tag{2.71}$$

It is our goal to find parameters θ such that $p(d|\theta)$ is accurate. For a given collection D of documents denote by m_w the number of counts for word w in the entire collection. Moreover, denote by m the total number of words in the entire collection. In this case we have

$$p(D|\theta) = \prod_i p(d_i|\theta) = \prod_w p(w|\theta)^{m_w}. \tag{2.72}$$

Finding suitable parameters θ given D proceeds as follows: In a maximum likelihood model we set

$$p(w|\theta) = \frac{m_w}{m}. \tag{2.73}$$

In other words, we use the empirical frequency of occurrence as our best guess and the sufficient statistic of D is $\phi(w) = e_w$, where e_w denotes the unit vector which is nonzero only for the "coordinate" w. Hence $\mu[D]_w = \frac{m_w}{m}$.

We know that the conjugate prior of the multinomial model is a Dirichlet model. It follows from (2.70) that the posterior mode is obtained by replacing $\mu[D]$ by $\frac{m\mu[D]+n\nu}{m+n}$. Denote by $n_w := \nu_w \cdot n$ the pseudo-counts arising from the conjugate prior with parameters (ν, n). In this case we will estimate the probability of the word w as

$$p(w|\theta) = \frac{m_w + n_w}{m+n} = \frac{m_w + n\nu_w}{m+n}. \tag{2.74}$$

In other words, we add the pseudo counts n_w to the actual word counts m_w. This is particularly useful when the document we are dealing with is brief, that is, whenever we have little data: it is quite unreasonable to infer from a webpage of approximately 1000 words that words not occurring in this page have zero probability. This is exactly what is mitigated by means of the conjugate prior (ν, n).

Finally, let us consider norm-constrained priors of the form (2.66). In this case, the integral required for

$$\begin{aligned} p(D) &= \int p(D|\theta)p(\theta)d\theta \\ &\propto \int \exp\left(-\lambda \left\|\theta - \theta_0\right\|_p^d + m\left\langle \mu[D], \theta\right\rangle - mg(\theta)\right) d\theta \end{aligned}$$

is *intractable* and we need to resort to an approximation. A popular choice is to replace the integral by $p(D|\theta^*)$ where θ^* maximizes the integrand. This is precisely the MAP approximation of (2.64). Hence, in order to perform estimation we need to solve

$$\underset{\theta}{\text{minimize}}\, g(\theta) - \langle \mu[D], \theta\rangle + \frac{\lambda}{m}\left\|\theta - \theta_0\right\|_p^d. \tag{2.75}$$

A very simple strategy for minimizing (2.75) is gradient descent. That is for a given value of θ we compute the gradient of the objective function and take a fixed step towards its minimum. For simplicity assume that $d = p = 2$ and $\lambda = 1/2\sigma^2$, that is, we assume that θ is normally distributed with variance σ^2 and mean θ_0. The gradient is given by

$$\nabla_\theta\left[-\log p(D, \theta)\right] = \mathbf{E}_{x \sim p(x|\theta)}[\phi(x)] - \mu[D] + \frac{1}{m\sigma^2}[\theta - \theta_0] \tag{2.76}$$

In other words, it depends on the discrepancy between the mean of $\phi(x)$ with respect to our current model and the empirical average $\mu[X]$, and the difference between θ and the prior mean θ_0.

Unfortunately, convergence of the procedure $\theta \leftarrow \theta - \eta\nabla_\theta\left[\ldots\right]$ is usually very slow, even if we adjust the steplength η efficiently. The reason is that the gradient need not point towards the minimum as the space is most likely distorted. A better strategy is to use Newton's method (see Chapter **??** for a detailed discussion and a convergence proof). It relies on a second order Taylor approximation

$$-\log p(D, \theta + \delta) \approx -\log p(D, \theta) + \langle \delta, G\rangle + \frac{1}{2}\delta^\top H\delta \tag{2.77}$$

where G and H are the first and second derivatives of $-\log p(D, \theta)$ with respect to θ. The quadratic expression can be minimized with respect to δ by choosing $\delta = -H^{-1}G$ and we can fashion an update algorithm from this by letting $\theta \leftarrow \theta - H^{-1}G$. One may show (see Chapter **??**) that Algorithm 2.1

is quadratically convergent. Note that the prior on θ ensures that H is well conditioned even in the case where the variance of $\phi(x)$ is not. In practice this means that the prior ensures fast convergence of the optimization algorithm.

Algorithm 2.1 Newton method for MAP estimation

```
NewtonMAP($D$)
  Initialize $\theta = \theta_0$
  **while** not converged **do**
    Compute $G = \mathbf{E}_{x \sim p(x|\theta)}[\phi(x)] - \mu[D] + \frac{1}{m\sigma^2}[\theta - \theta_0]$
    Compute $H = \mathrm{Var}_{x \sim p(x|\theta)}[\phi(x)] + \frac{1}{m\sigma^2}\mathbf{1}$
    Update $\theta \leftarrow \theta - H^{-1}G$
  **end while**
  **return** $\theta$
```

2.5 Sampling

So far we considered the problem of estimating the underlying probability density, given a set of samples drawn from that density. Now let us turn to the converse problem, that is, how to generate random variables given the underlying probability density. In other words, we want to design a random variable generator. This is useful for a number of reasons:

We may encounter probability distributions where optimization over suitable model parameters is essentially impossible and where it is equally impossible to obtain a closed form expression of the distribution. In these cases it may still be possible to perform sampling to draw examples of the kind of data we expect to see from the model. Chapter **??** discusses a number of graphical models where this problem arises.

Secondly, assume that we are interested in testing the performance of a network router under different load conditions. Instead of introducing the under-development router in a live network and wreaking havoc, one could estimate the probability density of the network traffic under various load conditions and build a model. The behavior of the network can then be simulated by using a probabilistic model. This involves drawing random variables from an estimated probability distribution.

Carrying on, suppose that we generate data packets by sampling and see an anomalous behavior in your router. In order to reproduce and debug this problem one needs access to the same set of random packets which caused the problem in the first place. In other words, it is often convenient if our random variable generator is reproducible; At first blush this seems like a contradiction. After all, our random number generator is supposed to generate random variables. This is less of a contradiction if we consider how random numbers are generated in a computer — given a particular initialization (which typically depends on the state of the system, e.g. time, disk size, bios checksum, etc.) the random number algorithm produces a sequence of numbers which, for all practical purposes, can be treated as iid. A simple method is the linear congruential generator [PTVF94]

$$x_{i+1} = (a x_i + b) \bmod c.$$

The performance of these iterations depends significantly on the choice of the constants a, b, c. For instance, the GNU C compiler uses $a = 1103515245, b = 12345$ and $c = 2^{32}$. In general b and c need to be relatively prime and $a - 1$ needs to be divisible by all prime factors of c and by 4. It is very much advisable *not* to attempt implementing such generators on one's own unless it is absolutely necessary.

Useful desiderata for a pseudo random number generator (PRNG) are that for practical purposes it is statistically indistinguishable from a sequence of iid data. That is, when applying a number of statistical tests, we will accept the null-hypothesis that the random variables are iid. See Chapter **??** for a detailed discussion of statistical testing procedures for random variables. In the following we assume that we have access to a *uniform* RNG $U[0, 1]$ which draws random numbers uniformly from the range $[0, 1]$.

2.5.1 Inverse Transformation

We now consider the scenario where we would like to draw from some distinctively non-uniform distribution. Whenever the latter is relatively simple this can be achieved by applying an inverse transform:

Theorem 2.21 *For $z \sim p(z)$ with $z \in \mathcal{Z}$ and an injective transformation $\phi : \mathcal{Z} \to \mathcal{X}$ with inverse transform ϕ^{-1} on $\phi(\mathcal{Z})$ it follows that the random variable $x := \phi(z)$ is drawn from $\left|\nabla_x \phi^{-1}(x)\right| \cdot p(\phi^{-1}(x))$. Here $\left|\nabla_x \phi^{-1}(x)\right|$ denotes the determinant of the Jacobian of ϕ^{-1}.*

This follows immediately by applying a variable transformation for a measure, i.e. we change $dp(z)$ to $dp(\phi^{-1}(x)) \left|\nabla_x \phi^{-1}(x)\right|$. Such a conversion strategy is particularly useful for univariate distributions.

Corollary 2.22 *Denote by $p(x)$ a distribution on $\mathbb{R}$ with cumulative distribution function $F(x') = \int_{-\infty}^{x'} dp(x)$. Then the transformation $x = \phi(z) = F^{-1}(z)$ converts samples $z \sim U[0, 1]$ to samples drawn from $p(x)$.*

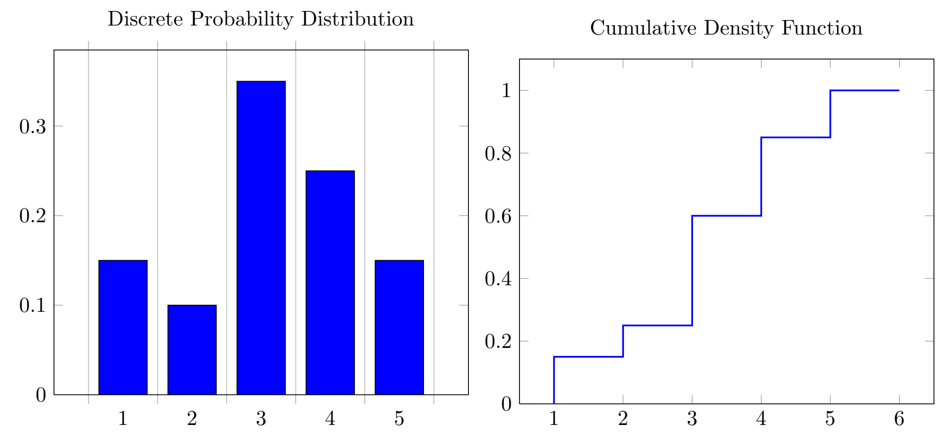


Fig. 2.13. Left: discrete probability distribution over 5 possible outcomes. Right: associated cumulative distribution function. When sampling, we draw x uniformly at random from $U[0, 1]$ and compute the inverse of F.

We now apply this strategy to a number of univariate distributions. One of the most common cases is sampling from a discrete distribution.

Example 2.8 (Discrete Distribution) *In the case of a discrete distribution over $\{1, \ldots, k\}$ the cumulative distribution function is a step-function with steps at $\{1, \ldots, k\}$ where the height of each step is given by the corresponding probability of the event.*

The implementation works as follows: denote by $p \in [0, 1]^k$ the vector of probabilities and denote by $f \in [0, 1]^k$ with $f_i = f_{i-1} + p_i$ and $f_1 = p_1$ the steps of the cumulative distribution function. Then for a random variable z drawn from $U[0, 1]$ we obtain $x = \phi(z) := \mathrm{argmin}_i \{f_i \geq z\}$. See Figure 2.13 for an example of a distribution over 5 events.

Example 2.9 (Exponential Distribution) *The density of a Exponential-distributed random variable is given by*

$$p(x|\lambda) = \lambda \exp(-\lambda x) \text{ if } \lambda > 0 \text{ and } x \geq 0. \tag{2.78}$$

This allows us to compute its cdf as

$$F(x|\lambda) = 1 - \exp(-\lambda x) \text{if } \lambda > 0 \text{ for } x \geq 0. \tag{2.79}$$

Therefore to generate a Exponential random variable we draw $z \sim U[0, 1]$ and solve $x = \phi(z) = F^{-1}(z|\lambda) = -\lambda^{-1} \log(1 - z)$. Since z and $1 - z$ are drawn from $U[0, 1]$ we can simplify this to $x = -\lambda^{-1} \log z$.

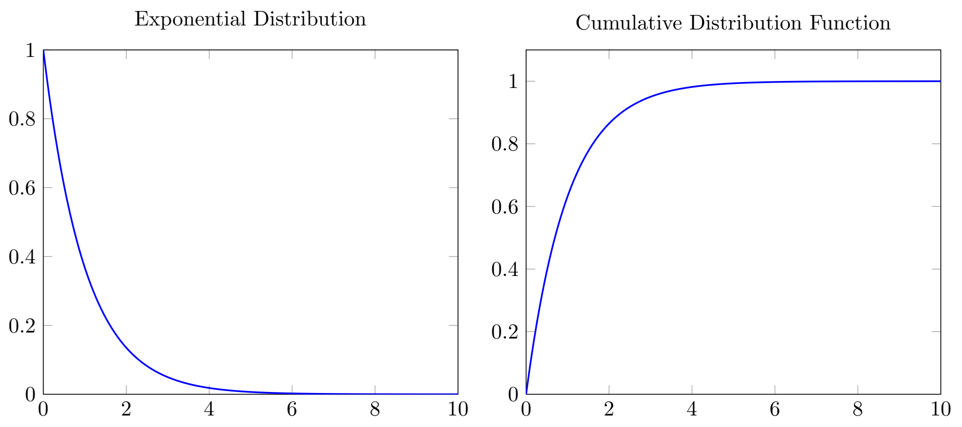


Fig. 2.14. Left: Exponential distribution with $\lambda = 1$. Right: associated cumulative distribution function. When sampling, we draw x uniformly at random from $U[0, 1]$ and compute the inverse.

We could apply the same reasoning to the normal distribution in order to draw Gaussian random variables. Unfortunately, the cumulative distribution function of the Gaussian is not available in closed form and we would need resort to rather nontrivial numerical techniques. It turns out that there exists a much more elegant algorithm which has its roots in Gauss' proof of the normalization constant of the Normal distribution. This technique is known as the Box-Müller transform.

Example 2.10 (Box-Müller Transform) *Denote by X, Y independent Gaussian random variables with zero mean and unit variance. We have*

$$p(x, y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2} = \frac{1}{2\pi} e^{-\frac{1}{2}(x^2+y^2)} \tag{2.80}$$

The key observation is that the joint *distribution $p(x, y)$ is radially symmetric, i.e. it only depends on the radius $r^2 = x^2 + y^2$. Hence we may perform a variable substitution in polar coordinates via the map ϕ where*

$$x = r \cos\theta \text{ and } y = r \sin\theta \text{ hence } (x, y) = \phi^{-1}(r, \theta). \tag{2.81}$$

This allows us to express the density in terms of (r, θ) via

$$p(r, \theta) = p(\phi^{-1}(r, \theta)) \left|\nabla_{r,\theta}\phi^{-1}(r, \theta)\right| = \frac{1}{2\pi} e^{-\frac{1}{2}r^2} \left| \begin{bmatrix} \cos\theta & \sin\theta \\ -r\sin\theta & r\cos\theta \end{bmatrix} \right| = \frac{r}{2\pi} e^{-\frac{1}{2}r^2}.$$

The fact that $p(r, \theta)$ is constant *in θ means that we can easily sample $\theta \in [0, 2\pi]$ by drawing a random variable, say z_θ from $U[0, 1]$ and rescaling it with*

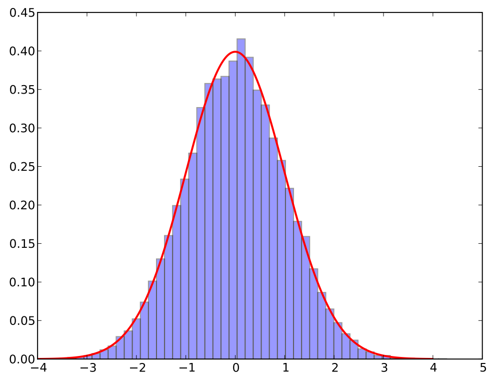


Fig. 2.15. Red: true density of the standard normal distribution (red line) is contrasted with the histogram of 20,000 random variables generated by the Box-Müller transform.

2π. *To obtain a sampler for* r *we need to compute the cumulative distribution function for* $p(r) = re^{-\frac{1}{2}r^2}$*:*

$$F(r') = \int_0^{r'} re^{-\frac{1}{2}r^2} dr = 1 - e^{-\frac{1}{2}r'^2} \text{ and hence } r = F^{-1}(z) = \sqrt{-2\log(1-z)}. \tag{2.82}$$

Observing that $z \sim U[0,1]$ *implies that* $1 - z \sim U[0,1]$ *yields the following sampler: draw* $z_\theta, z_r \sim U[0,1]$ *and compute* x *and* y *by*

$$x = \sqrt{-2\log z_r}\cos 2\pi z_\theta \text{ and } y = \sqrt{-2\log z_r}\sin 2\pi z_\theta.$$

Note that the Box-Müller transform yields two independent *Gaussian random variables. See Figure 2.15 for an example of the sampler.*

Example 2.11 (Uniform distribution on the disc) *A similar strategy can be employed when sampling from the unit disc. In this case the closed-form expression of the distribution is simply given by*

$$p(x,y) = \begin{cases} \frac{1}{\pi} & \text{if } x^2 + y^2 \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.83}$$

Using the variable transform (2.81) yields

$$p(r,\theta) = p(\phi^{-1}(r,\theta)) \left|\nabla_{r,\theta}\phi^{-1}(r,\theta)\right| = \begin{cases} \frac{r}{\pi} & \text{if } r \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.84}$$

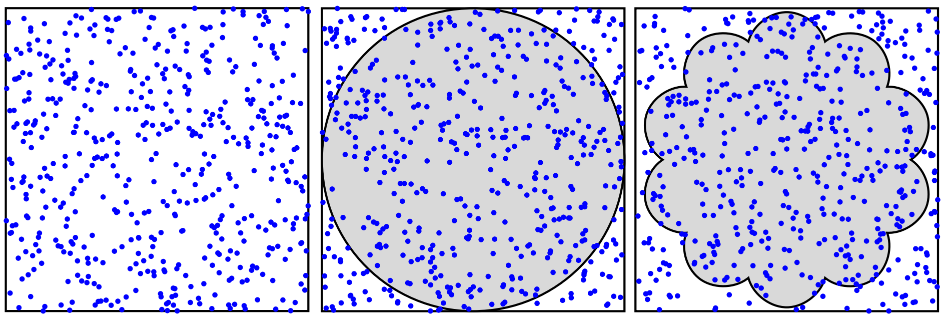


Fig. 2.16. Rejection sampler. Left: samples drawn from the uniform distribution on $[0,1]^2$. Middle: the samples drawn from the uniform distribution on the unit disc are all the points in the grey shaded area. Right: the same procedure allows us to sample uniformly from arbitrary sets.

Integrating out θ yields $p(r) = 2r$ for $r \in [0,1]$ with corresponding CDF $F(r) = r^2$ for $r \in [0,1]$. Hence our sampler draws $z_r, z_\theta \sim U[0,1]$ and then computes $x = \sqrt{z_r} \cos 2\pi z_\theta$ and $y = \sqrt{z_r} \sin 2\pi z_\theta$.

2.5.2 Rejection Sampler

All the methods for random variable generation that we looked at so far require intimate knowledge about the pdf of the distribution. We now describe a general purpose method, which can be used to generate samples from an arbitrary distribution. Let us begin with sampling from a set:

Example 2.12 (Rejection Sampler) *Denote by $X \subseteq \mathcal{X}$ a set and let p be a density on $\mathcal{X}$. Then a sampler for drawing from $p_X(x) \propto p(x)$ for $x \in X$ and $p_X(x) = 0$ for $x \notin X$, that is, $p_X(x) = p(x|x \in X)$ is obtained by the procedure:*

```
repeat
   draw $x \sim p(x)$
until $x \in X$
return $x$
```

That is, the algorithm keeps on drawing from p until the random variable is contained in X. The probability that this occurs is clearly $p(X)$. Hence the larger $p(X)$ the higher the efficiency of the sampler. See Figure 2.16.

Example 2.13 (Uniform distribution on a disc) *The procedure works trivially as follows: draw $x, y \sim U[0,1]$. Accept if $(2x-1)^2 + (2y-1)^2 \leq 1$ and return sample $(2x-1, 2y-1)$. This sampler has efficiency $\frac{4}{\pi}$ since this is the surface ratio between the unit square and the unit ball.*

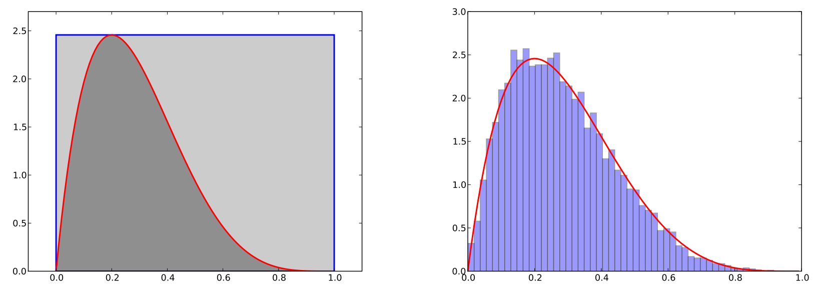


Fig. 2.17. Accept reject sampling for the Beta(2, 5) distribution. Left: Samples are generated uniformly from the blue rectangle (shaded area). Only those samples which fall under the red curve of the Beta(2, 5) distribution (darkly shaded area) are accepted. Right: The true density of the Beta(2, 5) distribution (red line) is contrasted with the histogram of 10,000 samples drawn by the rejection sampler.

Note that this time we did not need to carry out any sophisticated measure transform. This mathematical convenience came at the expense of a slightly less efficient sampler — about 21% of all samples are rejected.

The same reasoning that we used to obtain a hard accept/reject procedure can be used for a considerably more sophisticated rejection sampler. The basic idea is that if, for a given distribution p we can find another distribution q which, after rescaling, becomes an upper envelope on p, we can use q to sample from and reject depending on the ratio between q and p.

Theorem 2.23 (Rejection Sampler) *Denote by p and q distributions on $\mathcal{X}$ and let c be a constant such that such that $cq(x) \geq p(x)$ for all $x \in \mathcal{X}$. Then the algorithm below draws from p with acceptance probability c^{-1}.*

```
repeat
    draw $x \sim q(x)$ and $t \sim U[0,1]$
until $ct \leq \frac{p(x)}{q(x)}$
return $x$
```

Proof Denote by Z the event that the sample drawn from q is accepted. Then by Bayes rule the probability $\Pr(x|Z)$ can be written as follows

$$\Pr(x|Z) = \frac{\Pr(Z|x)\Pr(x)}{\Pr(Z)} = \frac{\frac{p(x)}{cq(x)} \cdot q(x)}{c^{-1}} = p(x) \tag{2.85}$$

Here we used that $\Pr(Z) = \int \Pr(Z|x)q(x)dx = \int c^{-1}p(x)dx = c^{-1}$. ■

Note that the algorithm of Example 2.12 is a special case of such a rejection sampler — we majorize p_X by the uniform distribution rescaled by $\frac{1}{p(X)}$.

Example 2.14 (Beta distribution) *Recall that the* $\mathrm{Beta}(a, b)$ *distribution, as a member of the Exponential Family with sufficient statistics* $(\log x, \log(1-x))$*, is given by*

$$p(x|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \tag{2.86}$$

For given (a, b) *one can verify (problem 2.25) that*

$$M := \operatorname*{argmax}_x p(x|a, b) = \frac{a-1}{a+b-2}. \tag{2.87}$$

provided $a > 1$*. Hence, if we use as proposal distribution the uniform distribution* $U[0, 1]$ *with scaling factor* $c = p(M|a, b)$ *we may apply Theorem 2.23. As illustrated in Figure 2.17, to generate a sample from* $\mathrm{Beta}(a, b)$ *we first generate a pair* (x, t)*, uniformly at random from the shaded rectangle. A sample is retained if* $ct \leq p(x|a, b)$*, and rejected otherwise. The acceptance rate of this sampler is* $\frac{1}{c}$*.*

Example 2.15 (Normal distribution) *We may use the Laplace distribution to generate samples from the Normal distribution. That is, we use*

$$q(x|\lambda) = \frac{\lambda}{2} e^{-\lambda|x|} \tag{2.88}$$

as the proposal distribution. For a normal distribution $p = \mathcal{N}(0, 1)$ *with zero mean and unit variance it turns out that choosing* $\lambda = 1$ *yields the most efficient sampling scheme (see Problem 2.27) with*

$$p(x) \leq \sqrt{\frac{2e}{\pi}}\, q(x|\lambda = 1)$$

As illustrated in Figure 2.18, we first generate $x \sim q(x|\lambda = 1)$ *using the inverse transform method (see Example 2.9 and Problem 2.21) and* $t \sim U[0, 1]$*. If* $t \leq \sqrt{2e/\pi}p(x)$ *we accept* x*, otherwise we reject it. The efficiency of this scheme is* $\sqrt{\frac{\pi}{2e}}$*.*

While rejection sampling is fairly efficient in low dimensions its efficiency is unsatisfactory in high dimensions. This leads us to an instance of the curse of dimensionality [Bel61]: the pdf of a d-dimensional Gaussian random variable centered at 0 with variance $\sigma^2\,\mathbf{1}$ is given by

$$p(x|\sigma^2) = (2\pi)^{-\frac{d}{2}} \sigma^{-d} e^{-\frac{1}{2\sigma^2}\|x\|^2}$$

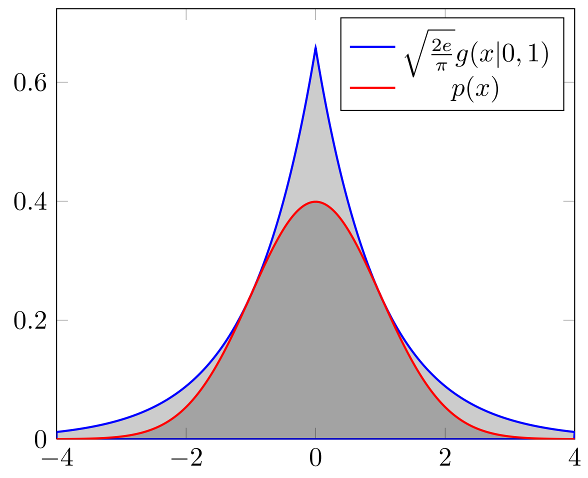


Fig. 2.18. Rejection sampling for the Normal distribution (red curve). Samples are generated uniformly from the Laplace distribution rescaled by $\sqrt{2e/\pi}$. Only those samples which fall under the red curve of the standard normal distribution (darkly shaded area) are accepted.

Now suppose that we want to draw from $p(x|\sigma^2)$ by sampling from another Gaussian q with slightly larger variance $\rho^2 > \sigma^2$. In this case the ratio between both distributions is maximized at 0 and it yields

$$c = \frac{q(0|\sigma^2)}{p(0|\rho^2)} = \left[\frac{\rho}{\sigma}\right]^d$$

If suppose $\frac{\rho}{\sigma} = 1.01$, and $d = 1000$, we find that $c \approx 20960$. In other words, we need to generate approximately 21,000 samples on the average from q to draw a single sample from p. We will discuss a more sophisticated sampling algorithms, namely Gibbs Sampling, in Section **??**. It allows us to draw from rather nontrivial distributions as long as the distributions in small subsets of random variables are simple enough to be tackled directly.

Problems

Problem 2.1 (Bias Variance Decomposition {1}) *Prove that the variance* $\mathrm{Var}_X[x]$ *of a random variable can be written as* $\mathbf{E}_X[x^2] - \mathbf{E}_X[x]^2$.

Problem 2.2 (Moment Generating Function {2}) *Prove that the characteristic function can be used to generate moments as given in (2.12). Hint: use the Taylor expansion of the exponential and apply the differential operator before the expectation.*

Problem 2.3 (Cumulative Error Function {2})

$$\text{erf}(x) = \sqrt{2/\pi} \int_0^x e^{-x^2} dx. \tag{2.89}$$

Problem 2.4 (Weak Law of Large Numbers {2}) *In analogy to the proof of the central limit theorem prove the weak law of large numbers. Hint: use a first order Taylor expansion of $e^{i\omega t} = 1 + i\omega t + o(t)$ to compute an approximation of the characteristic function. Next compute the limit $m \to \infty$ for $\phi_{\bar{X}_m}$. Finally, apply the inverse Fourier transform to associate the constant distribution at the mean μ with it.*

Problem 2.5 (Rates and confidence bounds {3}) *Show that the rate of hoeffding is tight — get bound from central limit theorem and compare to the hoeffding rate.*

Problem 2.6 *Why can't we just use each chip on the wafer as a random variable? Give a counterexample. Give bounds if we actually were allowed to do this.*

Problem 2.7 (Union Bound) *Work on many bounds at the same time. We only have logarithmic penalty.*

Problem 2.8 (Randomized Rounding {4}) *Solve the linear system of equations $Ax = b$ for integral x.*

Problem 2.9 (Randomized Projections {3}) *Prove that the randomized projections converge.*

Problem 2.10 (The Count-Min Sketch {5}) *Prove the projection trick*

Problem 2.11 (Parzen windows with triangle kernels {1}) *Suppose you are given the following data: $X = \{2, 3, 3, 5, 5\}$. Plot the estimated density using a kernel density estimator with the following kernel:*

$$k(u) = \begin{cases} 0.5 - 0.25 * |u| & \text{if } |u| \leq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Problem 2.12 *Gaussian process link with Gaussian prior on natural parameters*

Problem 2.13 *Optimization for Gaussian regularization*

Problem 2.14 *Conjugate prior (student-t and wishart).*

Problem 2.15 (Multivariate Gaussian {1}) *Prove that $\Sigma \succ 0$ is a necessary and sufficient condition for the normal distribution to be well defined.*

Problem 2.16 (Discrete Exponential Distribution {2}) *$\phi(x) = x$ and uniform measure.*

Problem 2.17 *Exponential random graphs.*

Problem 2.18 (Maximum Entropy Distribution) *Show that exponential families arise as the solution of the maximum entropy estimation problem.*

Problem 2.19 (Maximum Likelihood Estimates for Normal Distributions) *Derive the maximum likelihood estimates for a normal distribution, that is, show that they result in*

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ and } \hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \hat{\mu})^2 \tag{2.90}$$

using the exponential families parametrization. Next show that while the mean estimate $\hat{\mu}$ is unbiased, the variance estimate has a slight bias of $O(\frac{1}{m})$. To see this, take the expectation with respect to $\hat{\sigma}^2$.

Problem 2.20 (cdf of Logistic random variable {1}) *Show that the cdf of the Logistic random variable (**??**) is given by (**??**).*

Problem 2.21 (Double-exponential (Laplace) distribution {1}) *Use the inverse-transform method to generate a sample from the double-exponential (Laplace) distribution (2.88).*

Problem 2.22 (Normal random variables in polar coordinates {1}) *If X_1 and X_2 are standard normal random variables and let (R, θ) denote the polar coordinates of the pair (X_1, X_2). Show that $R^2 \sim \chi^2_2$ and $\theta \sim \mathrm{Unif}[0, 2\pi]$.*

Problem 2.23 (Monotonically increasing mappings {1}) *A mapping $T : \mathbb{R} \to \mathbb{R}$ is one-to-one if, and only if, T is monotonically increasing, that is, $x > y$ implies that $T(x) > T(y)$.*

Problem 2.24 (Monotonically increasing multi-maps {2}) *Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be one-to-one. If $X \sim p_X(x)$, then show that the distribution $p_Y(y)$ of $Y = T(X)$ can be obtained via (**??**).*

Problem 2.25 (Argmax of the $\mathrm{Beta}(a, b)$ distribution {1}) *Show that the mode of the $\mathrm{Beta}(a, b)$ distribution is given by (2.87).*

Problem 2.26 (Accept reject sampling for the unit disk {2}) *Give at least **<u>TWO</u>** different accept-reject based sampling schemes to generate samples uniformly at random from the unit disk. Compute their efficiency.*

Problem 2.27 (Optimizing Laplace for Standard Normal {1}) *Optimize the ratio $p(x)/g(x|\mu,\sigma)$, with respect to μ and σ, where $p(x)$ is the standard normal distribution (**??**), and $g(x|\mu,\sigma)$ is the Laplace distribution (2.88).*

Problem 2.28 (Normal Random Variable Generation {2}) *The aim of this problem is to write code to generate standard normal random variables (**??**) by using different methods. To do this generate $U \sim \mathrm{Unif}[0,1]$ and apply*

(i) *the Box-Muller transformation outlined in Section **??**.*
(ii) *use the following approximation to the inverse CDF*

$$\Phi^{-1}(\alpha) \approx t - \frac{a_0 + a_1 t}{1 + b_1 t + b_2 t^2}, \tag{2.91}$$

where $t^2 = \log(\alpha^{-2})$ and

$$a_0 = 2.30753, a_1 = 0.27061, b_1 = 0.99229, b_2 = 0.04481$$

(iii) *use the method outlined in example 2.15.*

Plot a histogram of the samples you generated to confirm that they are normally distributed. Compare these different methods in terms of the time needed to generate 1000 random variables.

Problem 2.29 (Non-standard Normal random variables {2}) *Describe a scheme based on the Box-Muller transform to generate d dimensional normal random variables $p(x|0,I)$. How can this be used to generate arbitrary normal random variables $p(x|\mu,\Sigma)$.*

Problem 2.30 (Uniform samples from a disk {2}) *Show how the ideas described in Section **??** can be generalized to draw samples uniformly at random from an axis parallel ellipse: $\{(x,y) : \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} \leq 1\}$.*

Bibliography

[ABB$^+$00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, *Gene ontology: tool for the unification of biology. the gene ontology consortium*, Nat Genet **25** (2000), 25–29.

[Ach03] D. Achlioptas, *Database-friendly random projections: Johnson-lindenstrauss with binary coins*, J. Comput. Syst. Sci **66** (2003), no. 4, 671–687.

[ADSM06] J. Anemuller, J.-R. Duann, T. J. Sejnowski, and S. Makeig, *Spatio-temporal dynamics in fmri recordings revealed with complex independent component analysis*, Neurocomputing **69** (2006), 1502–1512.

[AGML90] S. F. Altschul, W. Gish, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool*, Journal of Molecular Biology **215** (1990), no. 3, 403–410.

[BBL05] O. Bousquet, S. Boucheron, and G. Lugosi, *Theory of classification: a survey of recent advances*, ESAIM: Probab. Stat. **9** (2005), 323– 375.

[BCR84] C. Berg, J. P. R. Christensen, and P. Ressel, *Harmonic analysis on semigroups*, Springer, New York, 1984.

[BDEL03] S. Ben-David, N. Eiron, and P.M. Long, *On the difficulty of approximately maximizing agreements*, J. Comput. System Sci. **66** (2003), no. 3, 496–514.

[Bel61] R. E. Bellman, *Adaptive control processes*, Princeton University Press, Princeton, NJ, 1961.

[Bel05] Alexandre Belloni, *Introduction to bundle methods*, Tech. report, Operation Research Center, M.I.T., 2005.

[Ber85] J. O. Berger, *Statistical decision theory and Bayesian analysis*, Springer, New York, 1985.

[Bes74] J. Besag, *Spatial interaction and the statistical analysis of lattice systems (with discussion)*, Journal of the Royal Statistical Society. Series B **36** (1974), no. 2, 192–236.

[BH04] J. Basilico and T. Hofmann, *Unifying collaborative and content-based filtering*, Proc. Intl. Conf. Machine Learning (New York, NY), ACM Press, 2004, pp. 65–72.

[BHK98] J. S. Breese, D. Heckerman, and C. Kardie, *Empirical analysis of predictive algorithms for collaborative filtering*, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998, pp. 43–52.

[BHS$^+$07] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan, *Predicting structured data*, MIT Press, Cambridge, Massachusetts, 2007.

[Bil68] Patrick Billingsley, *Convergence of probability measures*, John Wiley and Sons, 1968.

[Bis95] C. M. Bishop, *Neural networks for pattern recognition*, Clarendon Press, Oxford, 1995.

[Bis06] Christopher Bishop, *Pattern recognition and machine learning*, Springer, 2006.

[BK07] R. M. Bell and Y. Koren, *Lessons from the netflix prize challenge*, SIGKDD Explorations **9** (2007), no. 2, 75–79.

[BKL06] A. Beygelzimer, S. Kakade, and J. Langford, *Cover trees for nearest neighbor*, International Conference on Machine Learning, 2006.

[BL00] J. M. Borwein and A. S. Lewis, *Convex analysis and nonlinear optimization: Theory and examples*, CMS books in Mathematics, Canadian Mathematical Society, 2000.

[BM90] H. Bourlard and N. Morgan, *A continuous speech recognition system embedding MLP into HMM*, Advances in Neural Information Processing Systems 2 (San Mateo, CA) (D. S. Touretzky, ed.), Morgan Kaufmann Publishers, 1990, pp. 186–193.

[BM92] K. P. Bennett and O. L. Mangasarian, *Robust linear programming discrimination of two linearly inseparable sets*, Optim. Methods Softw. **1** (1992), 23–34.

[BNJ03] D. Blei, A. Ng, and M. Jordan, *Latent Dirichlet allocation*, Journal of Machine Learning Research **3** (2003), 993–1022.

[BPX$^+$07] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean, *Large language models in machine translation*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007, pp. 858–867.

[BT03a] Amir Beck and Marc Teboulle, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Operations Research Letters **31** (2003), no. 3, 167–175.

[BT03b] D.P. Bertsekas and J.N. Tsitsiklis, *Introduction to probability*, Athena Scientific, 2003.

[BV04] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, Cambridge, England, 2004.

[Car97] J.-F. Cardoso, *Infomax and maximum likelihood for blind source separation*, IEEE Letters on Signal Processing **4** (1997), 112–114.

[Car98] ———, *Blind signal separation: statistical principles*, Proceedings of the IEEE **90** (1998), no. 8, 2009–2026.

[CDLS99] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter, *Probabilistic networks and expert sytems*, Springer, New York, 1999.

[CH04] Lijuan Cai and T. Hofmann, *Hierarchical document categorization with support vector machines*, Proceedings of the Thirteenth ACM conference on Information and knowledge management (New York, NY, USA), ACM Press, 2004, pp. 78–87.

[Cra46] H. Cramér, *Mathematical methods of statistics*, Princeton University Press, 1946.

[Cre93] N. A. C. Cressie, *Statistics for spatial data*, John Wiley and Sons, New York, 1993.

[CS03] K. Crammer and Y. Singer, *Ultraconservative online algorithms for multiclass problems*, Journal of Machine Learning Research **3** (2003), 951–991.

[CSS00] M. Collins, R. E. Schapire, and Y. Singer, *Logistic regression, AdaBoost and Bregman distances*, Proc. 13th Annu. Conference on Comput. Learning Theory, Morgan Kaufmann, San Francisco, 2000, pp. 158–169.

[CT91] T. M. Cover and J. A. Thomas, *Elements of information theory*, John Wiley and Sons, New York, 1991.

[CV95] C. Cortes and V. Vapnik, *Support vector networks*, Machine Learning **20** (1995), no. 3, 273–297.

[DdFG01] Arnaud Doucet, Nando de Freitas, and Neil Gordon, *Sequential monte carlo methods in practice*, Springer-Verlag, 2001.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, Cambridge University Press, 1998.

[DG03] S. Dasgupta and A. Gupta, *An elementary proof of a theorem of johnson and lindenstrauss*, Random Struct. Algorithms **22** (2003), no. 1, 60–65.

[DG08] J. Dean and S. Ghemawat, *MapReduce: simplified data processing on large clusters*, CACM **51** (2008), no. 1, 107–113.

[DGL96] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic theory of pattern recognition*, Applications of mathematics, vol. 31, Springer, New York, 1996.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society B **39** (1977), no. 1, 1–22.

[Fel71] W. Feller, *An introduction to probability theory and its applications*, 2 ed., John Wiley and Sons, New York, 1971.

[FJ95] A. Frieze and M. Jerrum, *An analysis of a monte carlo algorithm for estimating the permanent*, Combinatorica **15** (1995), no. 1, 67–83.

[FMT96] J. Franklin, T. Mitchell, and S. Thrun (eds.), *Recent advances in robot learning*, Kluwer International Series in Engineering and Computer Science, no. 368, Kluwer Academic Publishers, 1996.

[FS99] Y. Freund and R. E. Schapire, *Large margin classification using the perceptron algorithm*, Machine Learning **37** (1999), no. 3, 277–296.

[FT94] L. Fahrmeir and G. Tutz, *Multivariate statistical modelling based on generalized linear models*, Springer, 1994.

[GIM99] A. Gionis, P. Indyk, and R. Motwani, *Similarity search in high dimensions via hashing*, Proceedings of the 25th VLDB Conference (Edinburgh, Scotland) (M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, eds.), Morgan Kaufmann, 1999, pp. 518–529.

[GS04] T.L. Griffiths and M. Steyvers, *Finding scientific topics*, Proceedings of the National Academy of Sciences **101** (2004), 5228–5235.

[GVP90] D. Geiger, T. Verma, and J. Pearl, *Recognizing independence in Bayesian networks*, Networks **20** (1990), 507–534.

[GW92] P. Groeneboom and J. A. Wellner, *Information bounds and nonparametric maximum likelihood estimation*, DMV, vol. 19, Springer, 1992.

[Hal92] P. Hall, *The bootstrap and edgeworth expansions*, Springer, New York, 1992.

[Hay91] S. Haykin, *Adaptive filter theory*, Prentice-Hall, Englewood Cliffs, NJ, 1991, Second edition.

[Hay98] ———, *Neural networks : A comprehensive foundation*, Macmillan, New York, 1998, 2nd edition.

[HC71] J. M. Hammersley and P. E. Clifford, *Markov fields on finite graphs and lattices*, unpublished manuscript, 1971.

[Heb49] D. O. Hebb, *The organization of behavior*, John Wiley and Sons, New York, 1949.

[HF08] J.C. Huang and B.J. Frey, *Cumulative distribution networks and the derivative-sum-product algorithm*, UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, July 9-12, 2008, Helsinki, Finland (D.A. McAllester and P. Myllymäki, eds.), AUAI Press, 2008, pp. 290–297.

[HKO01] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent component analysis*, John Wiley and Sons, New York, 2001.

[Hoe63] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association **58** (1963), 13–30.

[HUL93] J.B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms, I and II*, vol. 305 and 306, Springer-Verlag, 1993.

[IM98] P. Indyk and R. Motawani, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, Proceedings of the 30th Symposium on Theory of Computing, 1998, pp. 604–613.

[JGJS99] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, *An introduction to variational methods for graphical models*, Machine Learning **37** (1999), no. 2, 183–233.

[JK02] K. Jarvelin and J. Kekalainen, *IR evaluation methods for retrieving highly relevant documents*, ACM Special Interest Group in Information Retrieval (SIGIR), New York: ACM, 2002, pp. 41–48.

[Joa05] T. Joachims, *A support vector method for multivariate performance measures*, Proc. Intl. Conf. Machine Learning (San Francisco, California), Morgan Kaufmann Publishers, 2005, pp. 377–384.

[Joa06] ———, *Training linear SVMs in linear time*, Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD), ACM, 2006.

[Jor08] M. I. Jordan, *An introduction to probabilistic graphical models*, MIT Press, 2008, To Appear.

[JV87] R. Jonker and A. Volgenant, *A shortest augmenting path algorithm for dense and sparse linear assignment problems*, Computing **38** (1987), 325–340.

[Kar80] R.M. Karp, *An algorithm to solve the $m \times n$ assignment problem in expected time $O(mn \log n)$*, Networks **10** (1980), no. 2, 143–152.

[KD05] S. S. Keerthi and D. DeCoste, *A modified finite Newton method for fast solution of large scale linear SVMs*, J. Mach. Learn. Res. **6** (2005), 341–361.

[Kel60] J. E. Kelly, *The cutting-plane method for solving convex programs*, Journal of the Society for Industrial and Applied Mathematics **8** (1960), no. 4, 703–712.

[KF09] D. Koller and N. Friedman, *Probabilistic graphical models: Principles and techniques*, MIT Press, 2009.

[Kiw90] Krzysztof C. Kiwiel, *Proximity control in bundle methods for convex nondifferentiable minimization*, Mathematical Programming **46** (1990), 105–122.

[KM00] Paul Komarek and Andrew Moore, *A dynamic adaptation of AD-trees for efficient machine learning on large data sets*, Proc. Intl. Conf. Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, pp. 495–502.

[KMH94] A. Krogh, I. S. Mian, and D. Haussler, *A hidden Markov model that finds genes in* e. coli *DNA*, Nucleic Acids Research **22** (1994), 4768–4778.

[Koe05] R. Koenker, *Quantile regression*, Cambridge University Press, 2005.

[Kuh55] H.W. Kuhn, *The Hungarian method for the assignment problem*, Naval Research Logistics Quarterly **2** (1955), 83–97.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

[Lew98] D. D. Lewis, *Naive (Bayes) at forty: The independence assumption in information retrieval*, Proceedings of ECML-98, 10th European Conference on Machine Learning (Chemnitz, DE) (C. Nédellec and C. Rouveirol, eds.), no. 1398, Springer Verlag, Heidelberg, DE, 1998, pp. 4–15.

[LGBS00] T.-W. Lee, M. Girolami, A. Bell, and T. Sejnowski, *A unifying framework* *for independent component analysis*, Comput. Math. Appl. **39** (2000), 1–21.

[LK03] C. Leslie and R. Kuang, *Fast kernels for inexact string matching*, Proc. Annual Conf. Computational Learning Theory, 2003.

[LMP01] J. D. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data*, Proceedings of International Conference on Machine Learning (San Francisco, CA), vol. 18, Morgan Kaufmann, 2001, pp. 282–289.

[LNN95] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov, *New variants of bundle methods*, Mathematical Programming **69** (1995), 111–147.

[LS07] Quoc V. Le and Alexander J. Smola, *Direct optimization of ranking measures*, Tech. Report 0704.3359, arXiv, April 2007, http://arxiv.org/abs/0704.3359.

[LT92] Z. Q. Luo and P. Tseng, *On the convergence of coordinate descent method for convex differentiable minimization*, Journal of Optimization Theory and Applications **72** (1992), no. 1, 7–35.

[Lue84] D. G. Luenberger, *Linear and nonlinear programming*, second ed., Addison-Wesley, Reading, May 1984.

[Mac67] J. MacQueen, *Some methods of classification and analysis of multivariate observations*, Proc. 5th Berkeley Symposium on Math., Stat., and Prob. (L. M. LeCam and J. Neyman, eds.), U. California Press, Berkeley, CA, 1967, p. 281.

[Mar61] M.E. Maron, *Automatic indexing: An experimental inquiry*, Journal of the Association for Computing Machinery **8** (1961), 404–417.

[McA07] David McAllester, *Generalization bounds and consistency for structured labeling*, Predicting Structured Data (Cambridge, Massachusetts), MIT Press, 2007.

[McD89] C. McDiarmid, *On the method of bounded differences*, Survey in Combinatorics, Cambridge University Press, 1989, pp. 148–188.

[Mit97] T. M. Mitchell, *Machine learning*, McGraw-Hill, New York, 1997.

[MN83] P. McCullagh and J. A. Nelder, *Generalized linear models*, Chapman and Hall, London, 1983.

[MSR$^+$97] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, *Predicting time series with support vector machines*, Artificial Neural Networks ICANN'97 (Berlin) (W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, eds.), Lecture Notes in Comput. Sci., vol. 1327, Springer-Verlag, 1997, pp. 999–1004.

[Mun57] J. Munkres, *Algorithms for the assignment and transportation problems*, Journal of SIAM **5** (1957), no. 1, 32–38.

[MYA94] N. Murata, S. Yoshizawa, and S. Amari, *Network information criterion — determining the number of hidden units for artificial neural network models*, IEEE Transactions on Neural Networks **5** (1994), 865–872.

[Nad65] E. A. Nadaraya, *On nonparametric estimates of density functions and regression curves*, Theory of Probability and its Applications **10** (1965), 186–190.

[Ned02] Angelia Nedic, *Subgradient methods for convex minimization*, Ph.D. thesis, MIT, 2002.

[Nel06] R. B. Nelsen, *An introduction to copulas*, Springer, 2006.

[NW99] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer Series in Operations Research, Springer, 1999.

[OL93] J.B. Orlin and Y. Lee, *Quickmatch: A very fast algorithm for the assignment problem*, Working Paper 3547-93, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, March 1993.

[Pap62] A. Papoulis, *The fourier integral and its applications*, McGraw-Hill, New York, 1962.

[Pea01] J. Pearl, *Causality: Models, reasoning and inference*, Cambridge University Press, 2001.

[Pla99] J. Platt, *Fast training of support vector machines using sequential minimal optimization*, Advances in Kernel Methods — Support Vector Learning (Cambridge, MA) (B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds.), MIT Press, 1999, pp. 185–208.

[PTVF94] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in c. the art of scientific computation*, Cambridge University Press, Cambridge, UK, 1994.

[Rab89] L. R. Rabiner, *A tutorial on hidden Markov models and selected applications in speech recognition*, Proceedings of the IEEE **77** (1989), no. 2, 257–285.

[Rao73] C. R. Rao, *Linear statistical inference and its applications*, John Wiley and Sons, New York, 1973.

[RBZ06] N. Ratliff, J. Bagnell, and M. Zinkevich, *Maximum margin planning*, International Conference on Machine Learning, July 2006.

[RG99] S. Roweis and Z. Ghahramani, *A unifying review of linear Gaussian models*, Neural Computation **11** (1999), no. 2.

[Ros58] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review **65** (1958), no. 6, 386–408.

[RPB06] M. Richardson, A. Prakash, and E. Brill, *Beyond pagerank: machine learning for static ranking*, Proceedings of the 15th international conference on World Wide Web, WWW (L. Carr, D. De Roure, A. Iyengar, C.A. Goble, and M. Dahlin, eds.), ACM, 2006, pp. 707–715.

[RSS$^+$07] G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. J. Sommer, and B. Schölkopf, *Improving the Caenorhabditis elegans genome annotation using machine learning*, PLoS Computational Biology **3** (2007), no. 2, e20 doi:10.1371/journal.pcbi.0030020.

[Rud73] W. Rudin, *Functional analysis*, McGraw-Hill, New York, 1973.

[Sha48] C. E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal **27** (1948), 379–423, 623–656.

[Sha98] R.D. Shachter, *Bayes-ball: The rational pasttime*, Fourteenth Conference on Uncertainty in Artificial Intelligence (Wisconsin, USA) (G.F. Cooper and S. Moral, eds.), Morgan Kaufmann Publishers, Inc., San Francisco, June 1998.

[Sil86] B. W. Silverman, *Density estimation for statistical and data analysis*, Monographs on statistics and applied probability, Chapman and Hall, London, 1986.

[SPST$^+$01] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the support of a high-dimensional distribution*, Neural Comput. **13** (2001), no. 7, 1443–1471.

[SS02] B. Schölkopf and A. Smola, *Learning with kernels*, MIT Press, Cambridge, MA, 2002.

[SSS07] S. Shalev-Shwartz and Y. Singer, *Logarithmic regret algorithms for strongly convex repeated games*, Tech. report, School of Computer Science, Hebrew University, 2007.

[SW86] G.R. Shorack and J.A. Wellner, *Empirical processes with applications to statistics*, Wiley, New York, 1986.

[SZ92] Helga Schramm and Jochem Zowe, *A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results*, SIAM J. Optimization **2** (1992), 121–152.

[TGK04] B. Taskar, C. Guestrin, and D. Koller, *Max-margin Markov networks*, Advances in Neural Information Processing Systems 16 (Cambridge, MA) (S. Thrun, L. Saul, and B. Schölkopf, eds.), MIT Press, 2004, pp. 25–32.

[TJHA05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun, *Large margin methods for structured and interdependent output variables*, J. Mach. Learn. Res. **6** (2005), 1453–1484.

[Vap82] V. Vapnik, *Estimation of dependences based on empirical data*, Springer, Berlin, 1982.

[Vap95] ———, *The nature of statistical learning theory*, Springer, New York, 1995.

[Vap98] ———, *Statistical learning theory*, John Wiley and Sons, New York, 1998.

[vdG00] S. van de Geer, *Empirical processes in M-estimation*, Cambridge University Press, 2000.

[vdVW96] A. W. van der Vaart and J. A. Wellner, *Weak convergence and empirical processes*, Springer, 1996.

[VGS97] V. Vapnik, S. Golowich, and A. J. Smola, *Support vector method for function approximation, regression estimation, and signal processing*, Advances in Neural Information Processing Systems 9 (Cambridge, MA) (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), MIT Press, 1997, pp. 281–287.

[Voo01] E. Voorhees, *Overview of the TRECT 2001 question answering track*, TREC, 2001.

[VS04] S. V. N. Vishwanathan and A. J. Smola, *Fast kernels for string and tree matching*, Kernel Methods in Computational Biology (Cambridge, MA) (B. Schölkopf, K. Tsuda, and J. P. Vert, eds.), MIT Press, 2004, pp. 113–130.

[VSKB10] S. V. N. Vishwanathan, Nicol N. Schraudolph, Imre Risi Kondor, and Karsten M. Borgwardt, *Graph kernels*, J. Mach. Learn. Res. (2010), In press.

[VSV07] S. V. N. Vishwanathan, A. J. Smola, and R. Vidal, *Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes*, International Journal of Computer Vision **73** (2007), no. 1, 95–119.

[Wah97] G. Wahba, *Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV*, Tech. Report 984, Department of Statistics, University of Wisconsin, Madison, 1997.

[Wat64] G. S. Watson, *Smooth regression analysis*, Sankhya A **26** (1964), 359–372.

[Wat99] C. Watkins, *Dynamic alignment kernels*, CSD-TR-98- 11, Royal Holloway, University of London, Egham, Surrey, UK, 1999.

[WB06] G. Welch and G. Bishop, *An introduction to the kalman filter*, Tech. Report TR-95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 2006.

[Wil98] C. K. I. Williams, *Prediction with Gaussian processes: From linear regression to linear prediction and beyond*, Learning and Inference in Graphical Models (M. I. Jordan, ed.), Kluwer Academic, 1998, pp. 599–621.

[WJ03] M. J. Wainwright and M. I. Jordan, *Graphical models, exponential families, and variational inference*, Tech. Report 649, UC Berkeley, Department of Statistics, September 2003.

[WJ08] ———, *Graphical models, exponential families, and variational inference*, Foundations and Trends in Machine Learning **1** (2008), no. 1 − 2, 1 − 305.

[YAC98] H. H. Yang, S.-I. Amari, and A. Cichocki, *Information theoretic approach to blind separation of sources in non-linear mixture*, Signal Processing **64** (1998), no. 3, 291–300.

[Zin03] M. Zinkevich, *Online convex programming and generalised infinitesimal gradient ascent*, Proc. Intl. Conf. Machine Learning, 2003, pp. 928–936.