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 2$_\varepsilon$ [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

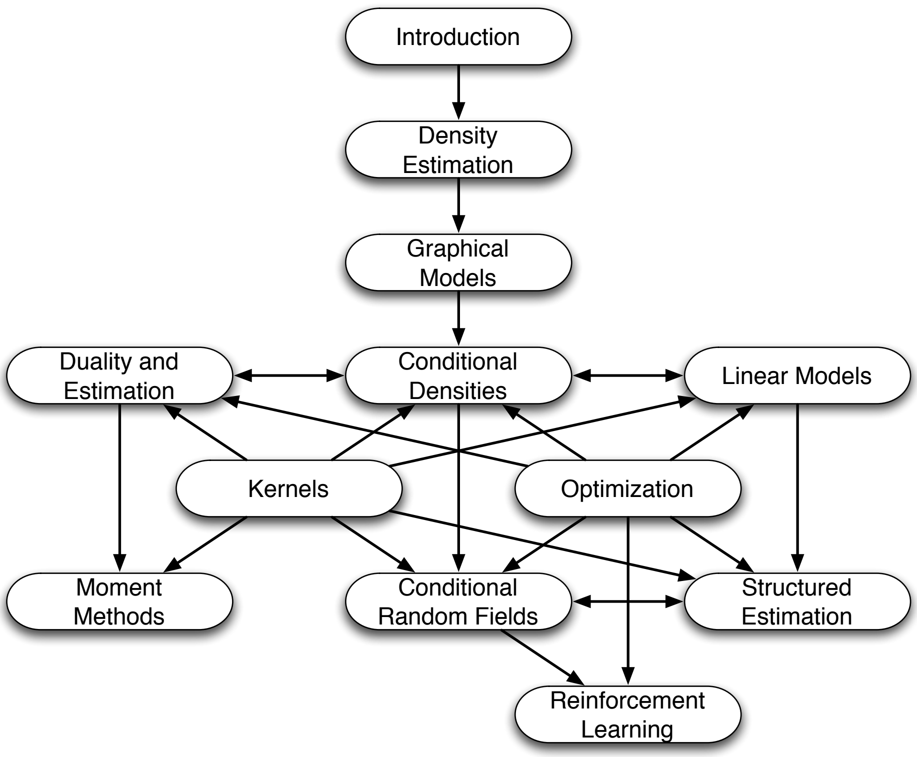
Author: vishy
Revision: 252
Timestamp: October 1, 2010
URL: svn://smola@repos.stat.purdue.edu/thebook/trunk/Book/thebook.tex

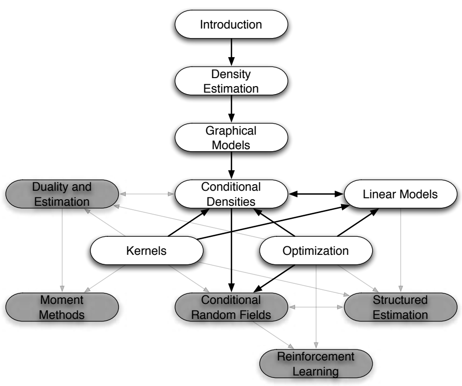
Contents

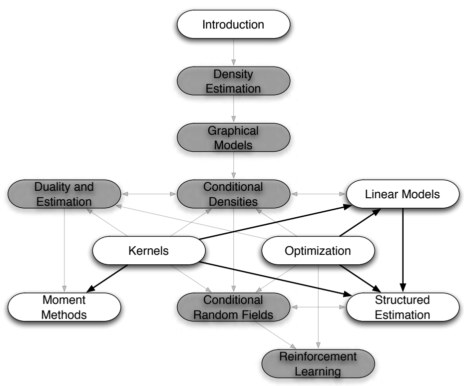
Preface

Since this is a textbook we biased our selection of references towards easily accessible work rather than the original references. While this may not be in the interest of the inventors of these concepts, it greatly simplifies access to those topics. Hence we encourage the reader to follow the references in the cited works should they be interested in finding out who may claim intellectual ownership of certain key ideas.

Structure of the Book







Canberra, August 2008

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

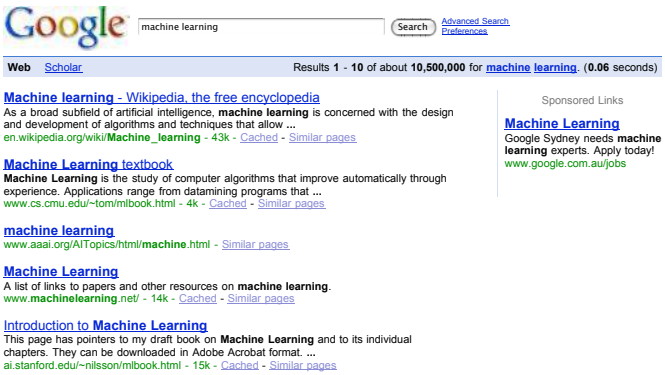


Fig. 1.1. The 5 top scoring webpages for the query "machine learning"

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 [**?**].

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'

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

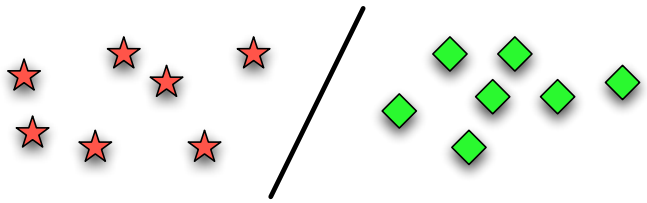


Fig. 1.5. Binary classification; separate stars from diamonds. In this example we are able to do so by drawing a straight line which separates both sets. We will see later that this is an important example of what is called a *linear classifier*.

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 → Animation → Anime → General Fan Pages → 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:

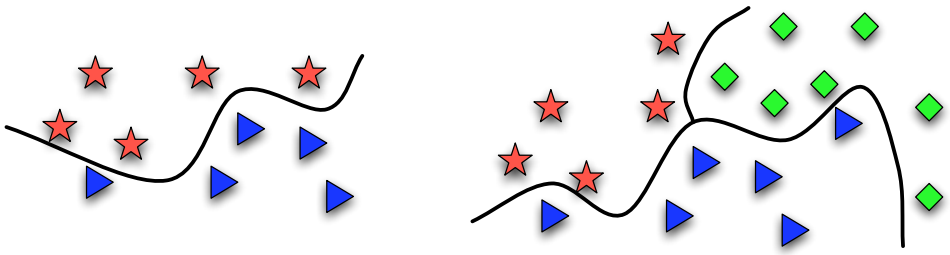


Fig. 1.6. Left: binary classification. Right: 3-class classification. Note that in the latter case we have much more degree for ambiguity. For instance, being able to distinguish stars from diamonds may not suffice to identify either of them correctly, since we also need to distinguish both of them from 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 $\mathbf{X}' = \{x'_1, \dots, x'_{m'}\}$. 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

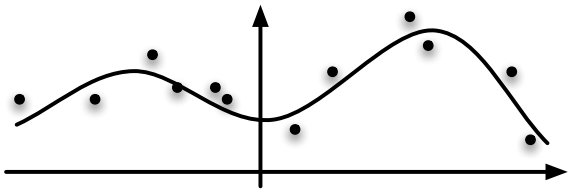


Fig. 1.7. Regression estimation. We are given a number of instances (indicated by black dots) and would like to find some function f mapping the observations $\mathcal{X}$ to $\mathbb{R}$ such that $f(x)$ is close to the observed values.

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 7.4). 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 [BT03].

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.

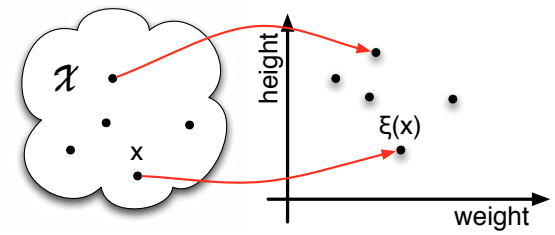


Fig. 1.9. The random variable ξ maps from the set of outcomes of an experiment (denoted here by $\mathcal{X}$) to real numbers. As an illustration here $\mathcal{X}$ consists of the patients a physician might encounter, and they are mapped via ξ to their weight and height.

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

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

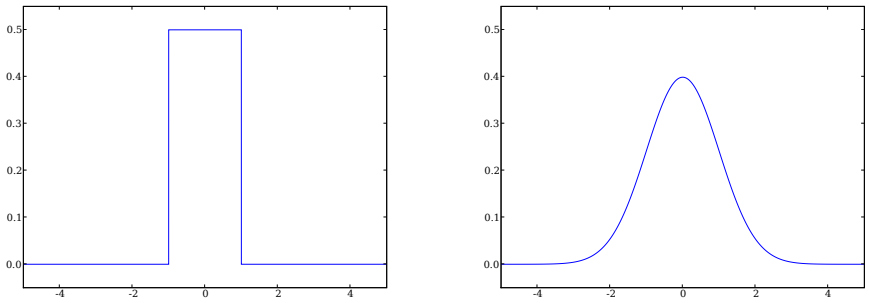


Fig. 1.10. Two common densities. Left: uniform distribution over the interval $[-1, 1]$. Right: Normal distribution with zero mean and unit variance.

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*

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

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

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

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 $\Pr(X < x') \leq q$ and $\Pr(X > x') \leq 1 - q$ is the q-quantile of the distribution p. Moreover, the value x' associated with $q = 0.5$ is called the median.*

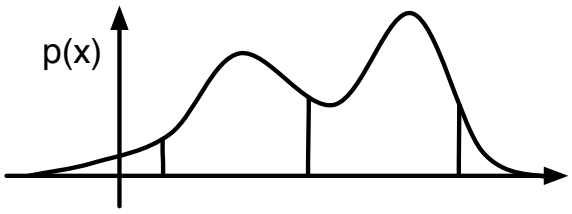


Fig. 1.11. Quantiles of a distribution correspond to the area under the integral of the density $p(x)$ for which the integral takes on a pre-specified value. Illustrated are the 0.1, 0.5 and 0.9 quantiles respectively.

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

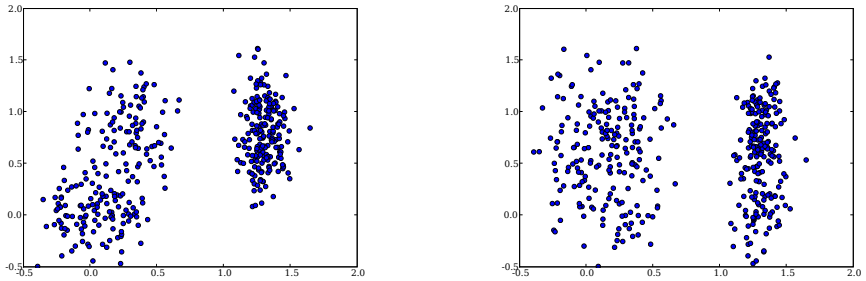


Fig. 1.12. Left: a sample from two dependent random variables. Knowing about first coordinate allows us to improve our guess about the second coordinate. Right: a sample drawn from two independent random variables, obtained by randomly permuting the dependent sample.

measuring the voltage of a device, we will typically assume that the individual measurements are drawn from the same distribution and that they are 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.

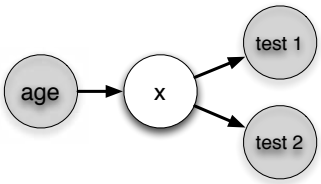


Fig. 1.13. A graphical description of our HIV testing scenario. Knowing the age of the patient influences our prior on whether the patient is HIV positive (the random variable X). The outcomes of the tests 1 and 2 are independent of each other given the status X. We observe the shaded random variables (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 **??**.

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 $p(X = \text{HIV+}|T = \text{HIV+}, A = 35)$ 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

$p(t_2\|x)$	$x =$ 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

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

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 $\mathbf{X} := \{x_1, \ldots, x_m\}$

```
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.

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 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

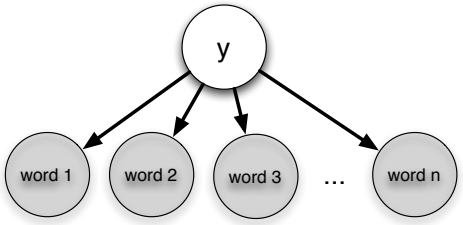


Fig. 1.16. Naive Bayes model. The occurrence of individual words is independent of each other, given the category of the text. For instance, the word *Viagra* is fairly frequent if $y =$ spam but it is considerably less frequent if $y =$ ham, except when considering the mailbox of a Pfizer sales representative.

$p(w|y)$ can be obtained, for instance, by simply counting the frequency occurrence 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_{ham} and m_{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 $y_i = \text{spam}$ **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

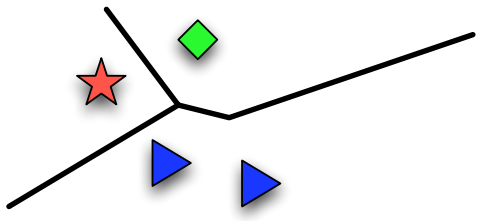


Fig. 1.17. 1 nearest neighbor classifier. Depending on whether the query point x is closest to the star, diamond or triangles, it uses one of the three labels for it.

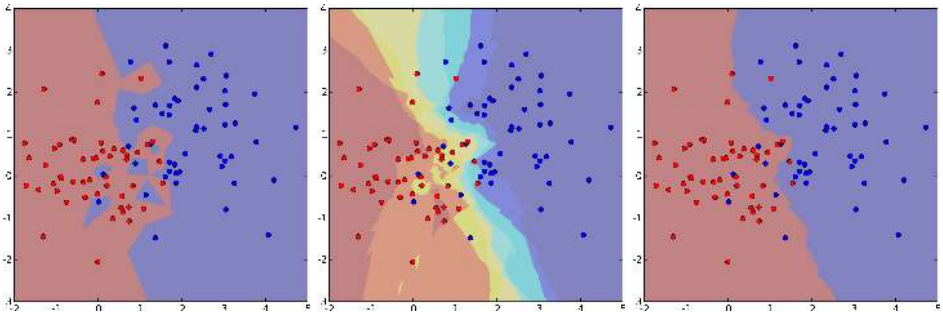


Fig. 1.18. k-Nearest neighbor classifiers using Euclidean distances. Left: decision boundaries obtained from a 1-nearest neighbor classifier. Middle: color-coded sets of where the number of red / blue points ranges between 7 and 0. Right: decision boundary determining where the blue or red dots are in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

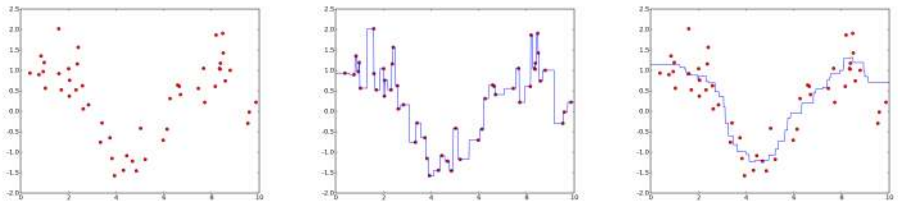


Fig. 1.19. k-Nearest neighbor regression estimator using Euclidean distances. Left: some points (x, y) drawn from a joint distribution. Middle: 1-nearest neighbour classifier. Right: 7-nearest neighbour classifier. Note that the regression estimate is much more smooth.

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].

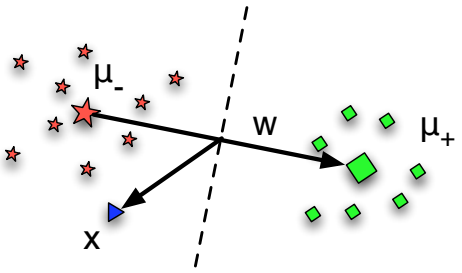


Fig. 1.20. A trivial classifier. Classification is carried out in accordance to which of the two means μ_- or μ_+ is closer to the test point x. Note that the sets of positive and negative labels respectively form a half space.

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

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

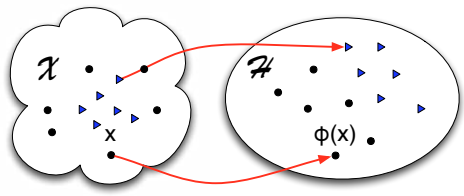


Fig. 1.21. The feature map ϕ maps observations x from $\mathcal{X}$ into a feature space $\mathcal{H}$. The map ϕ is a convenient way of encoding pre-processing steps systematically.

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 6, 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

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

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

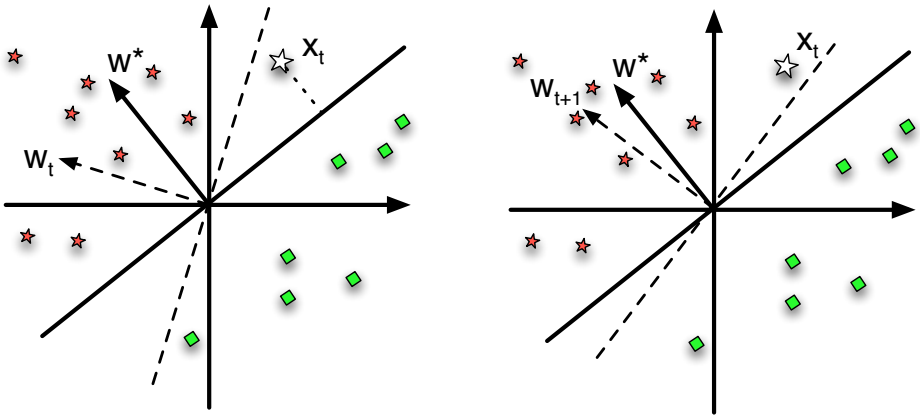


Fig. 1.22. The Perceptron without bias. Left: at time t we have a weight vector w_t denoted by the dashed arrow with corresponding separating plane (also dashed). For reference we include the linear separator w^* and its separating plane (both denoted by a solid line). As a new observation x_t arrives 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 7.

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 μ_j for $j = 1, \ldots, k$ randomly
  repeat
    for $i = 1$ **to** m **do**
      Compute $j' = \operatorname{argmin}_{j=1,\ldots,k} d(x_i, \mu_j)$
      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*

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

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* $\{1,\dots,6\}$. *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 $y^*(x) = \operatorname{argmax}_y p(y|x)$ 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

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

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

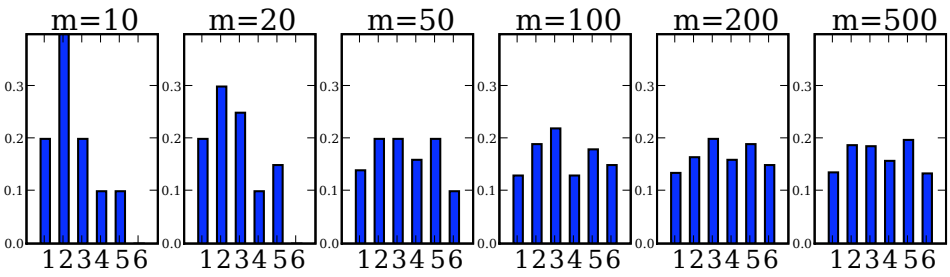


Fig. 2.1. Convergence of empirical means to expectations. From left to right: empirical frequencies of occurrence obtained by casting a dice 10, 20, 50, 100, 200, and 500 times respectively. Note that after 20 throws we still have not observed a single '6', an event which occurs with only $\left[\frac{5}{6}\right]^{20}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

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

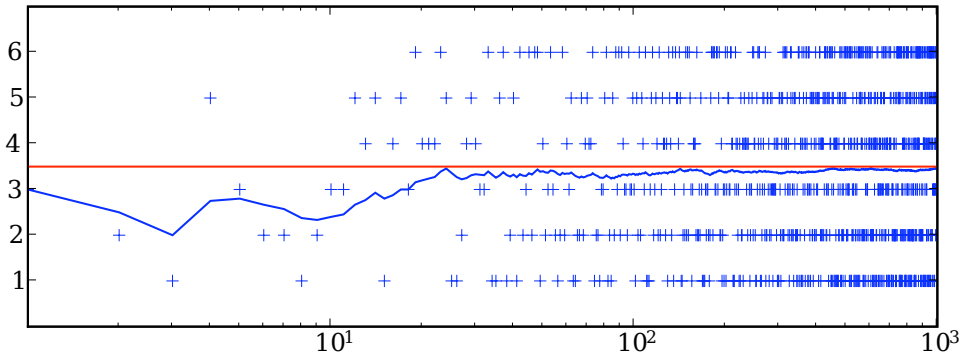


Fig. 2.2. The mean of a number of casts of a dice. The horizontal straight line denotes the mean 3.5. The uneven solid line denotes the actual mean $\bar{X}_n$ as a function of the number of draws, given as a semilogarithmic plot. The crosses denote the outcomes of the dice. Note how $\bar{X}_n$ ever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

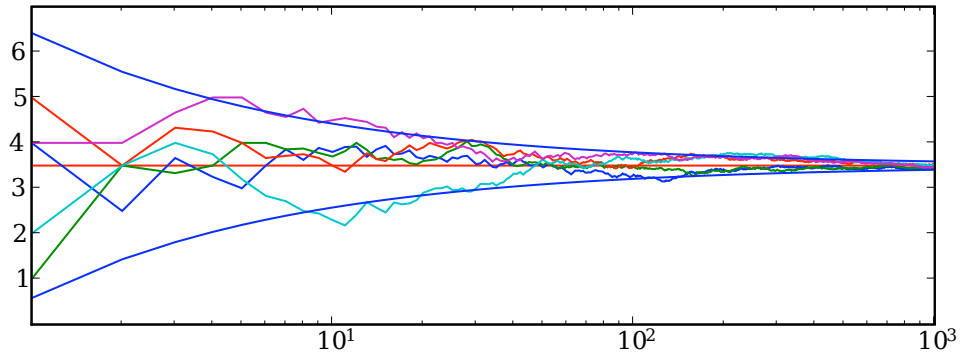


Fig. 2.3. Five instantiations of a running average over outcomes of a toss of a dice. Note that all of them converge to the mean 3.5. Moreover note that they all are well contained within the upper and lower envelopes given by $\mu \pm \sqrt{\mathrm{Var}_X[x]/m}$.

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* $2.92m^{-\frac{1}{2}}$.

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 $a_n^{-2}[g(X_n) - g(b)] \to [\nabla_x g(b)]^\top a_n^{-2}(X_n - b)$. 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 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

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

[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 $k(x, x')$.

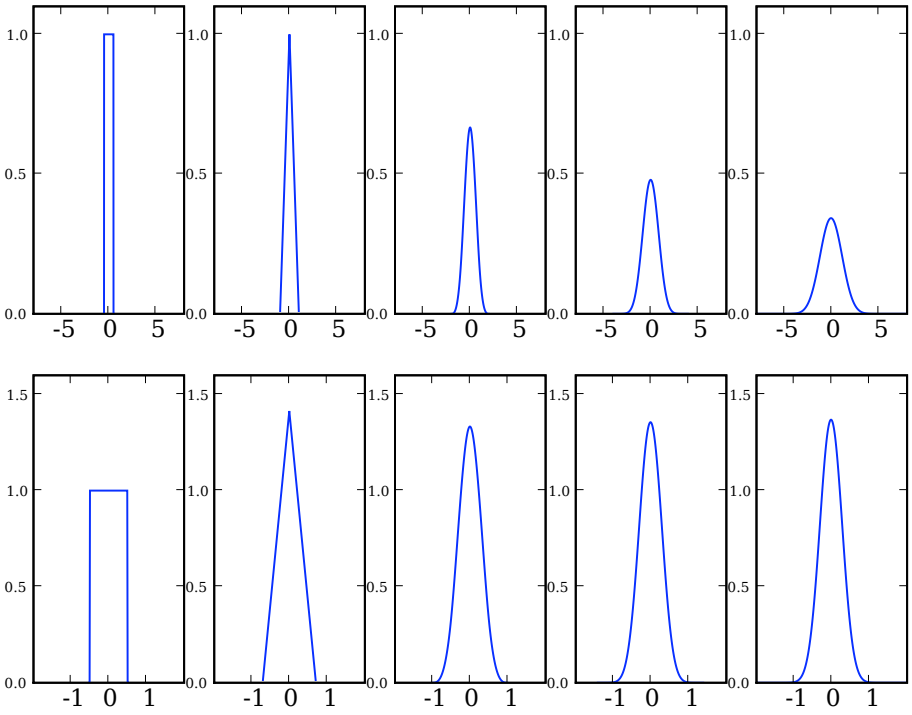


Fig. 2.4. A working example of the central limit theorem. The top row contains distributions of sums of uniformly distributed random variables on the interval $[0.5, 0.5]$. From left to right we have sums of $1, 2, 4, 8$ and 16 random variables. The bottom row contains the same distribution with the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

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

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

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

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

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

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

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

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

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.

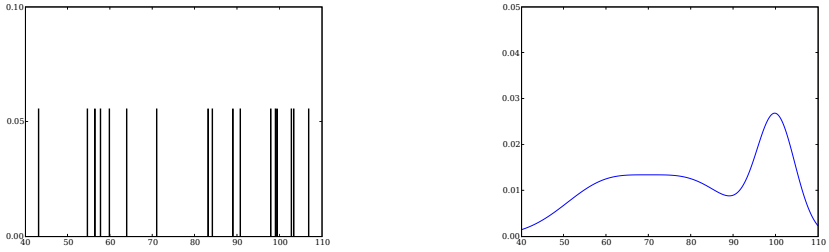


Fig. 2.5. Left: a naive density estimate given a sample of the weight of 18 persons. Right: the underlying weight distribution.

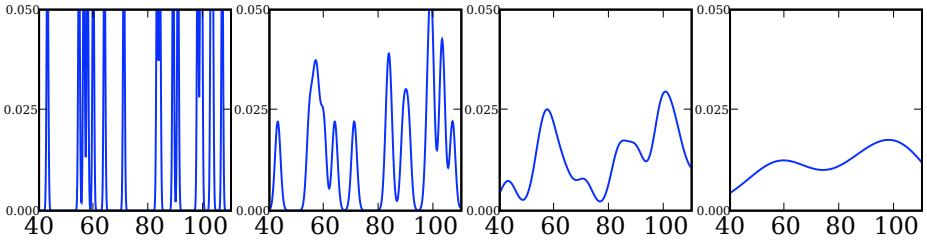


Fig. 2.6. Parzen windows density estimate associated with the 18 observations of the Figure above. From left to right: Gaussian kernel density estimate with kernel of width $0.3, 1, 3$, and 10 respectively.

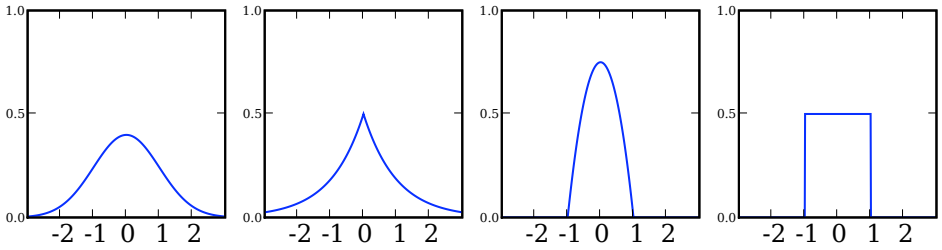


Fig. 2.7. Some kernels for Parzen windows density estimation. From left to right: Gaussian kernel, Laplace kernel, Epanechikov kernel, and uniform density.

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

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

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 $\mathbf{X} := \{x_1, \dots, x_n\}$ and $\mathbf{X}' := \{x_{n+1}, \dots, x_m\}$ 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 $\mathbf{X}_1, \dots, \mathbf{X}_n$ 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

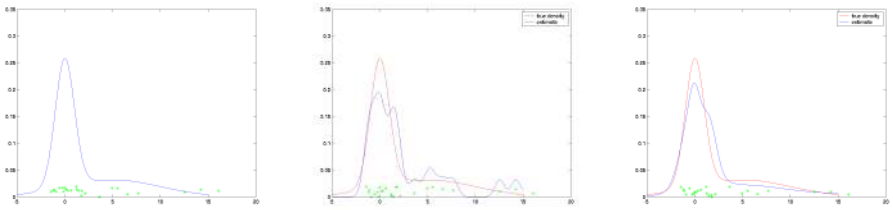


Fig. 2.8. Nonuniform density. Left: original density with samples drawn from the distribution. Middle: density estimate with a uniform kernel. Right: density estimate using Silverman's adjustment.

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

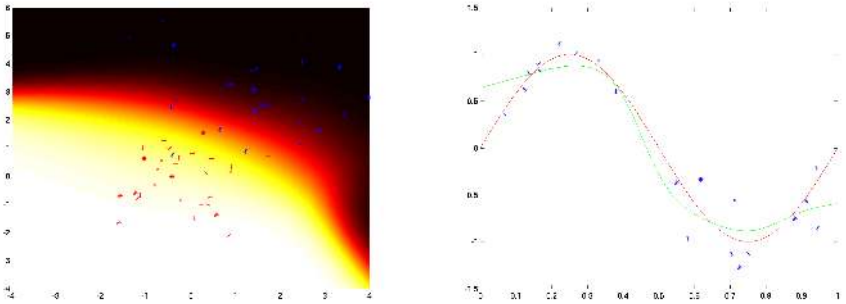


Fig. 2.9. Watson Nadaraya estimate. Left: a binary classifier. The optimal solution would be a straight line since both classes were drawn from a normal distribution with the same variance. Right: a regression estimator. The data was generated from a sinusoid with additive noise. The regression tracks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

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

Proof Note that $\nabla^2_\theta g(\theta) = \mathrm{Var}_x\left[\phi(x)\right]$ 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

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

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

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 $\nabla^n_\theta g(\theta)$ 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

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

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

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

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

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

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 distri-

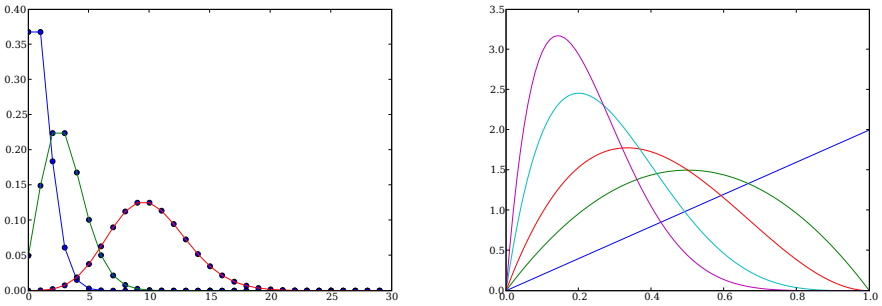


Fig. 2.10. Left: Poisson distributions with $\lambda = \{1, 3, 10\}$. Right: Beta distributions with $a = 2$ and $b \in \{1, 2, 3, 5, 7\}$. Note how with increasing b the distribution becomes more peaked close to the origin.

butions. It is given by

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

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.

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	$\mathfrak{C}_n$	$\lvert X\rvert^{-\frac{n+1}{2}}$	$\left(\log\lvert x\rvert, -\frac{1}{2}x\right)$	$-\theta_1\log\lvert\theta_2\rvert + \theta_1 n\log 2$ $+\sum_{i=1}^n \log\Gamma\left(\theta_1 + \frac{1-i}{2}\right)$	$\mathbb{R}\times\mathfrak{C}_n$
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 $\mathbb{R}^{n\times n}$.

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 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

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

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*

[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.

us to solve (2.56) in closed form using

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

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.

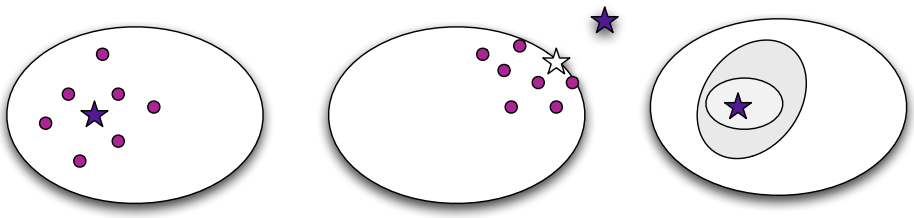


Fig. 2.11. Left: unbiased estimator; the estimates, denoted by circles have as mean the true parameter, as denoted by a star. Middle: consistent estimator. While the true model is not within the class we consider (as denoted by the ellipsoid), the estimates converge to the white star which is the best 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*

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

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 $\nabla_\theta^2 g(\theta)$.

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

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 $\nabla_\mu [\nabla_\theta g]^{-1}(\mu) = \left[\nabla_\theta^2 g(\theta)\right]^{-1}$. 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*

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

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

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

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 3. 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.

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

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

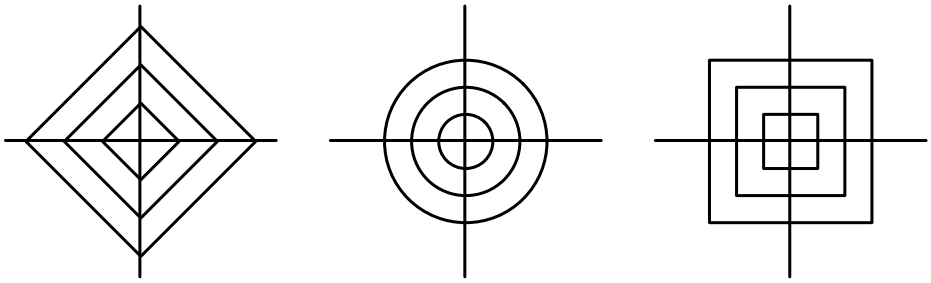


Fig. 2.12. From left to right: regions of equal prior probability in $\mathbb{R}^2$ for priors using the ℓ_1, ℓ_2 and ℓ_∞ norm. Note that only the ℓ_2 norm is invariant with regard to the coordinate system. As we shall see later, the ℓ_1 norm prior leads to solutions where only a small number of coordinates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

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 normal-

ization in the conjugate distribution, albeit $\phi(x)$ added. This yields

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

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 2. [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

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

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 3 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 3) 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 θ

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]

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

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*

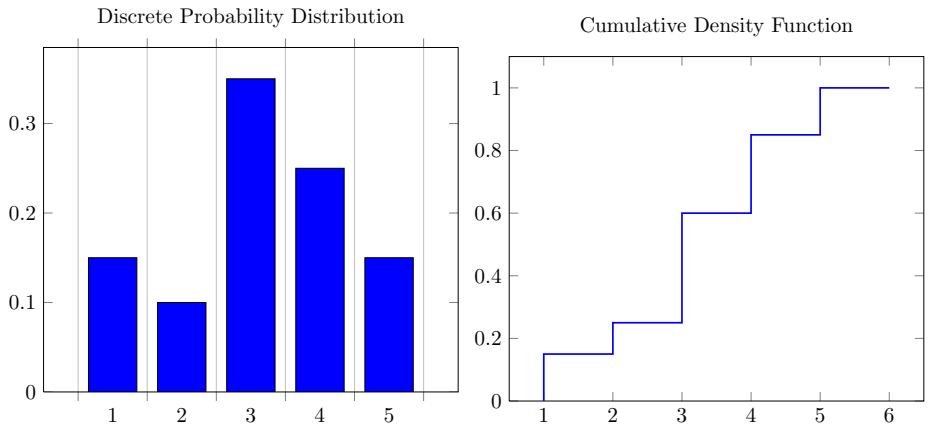


Fig. 2.13. Left: discrete probability distribution over 5 possible outcomes. Right: associated cumulative distribution function. When sampling, we draw x uniformly at random from $U[0,1]$ and compute the inverse of F.

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* $p(x)$.

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* $x = \phi(z) := \operatorname{argmin}_i \{f_i \geq z\}$. *See Figure 2.13 for an example of a distribution over* 5 *events.*

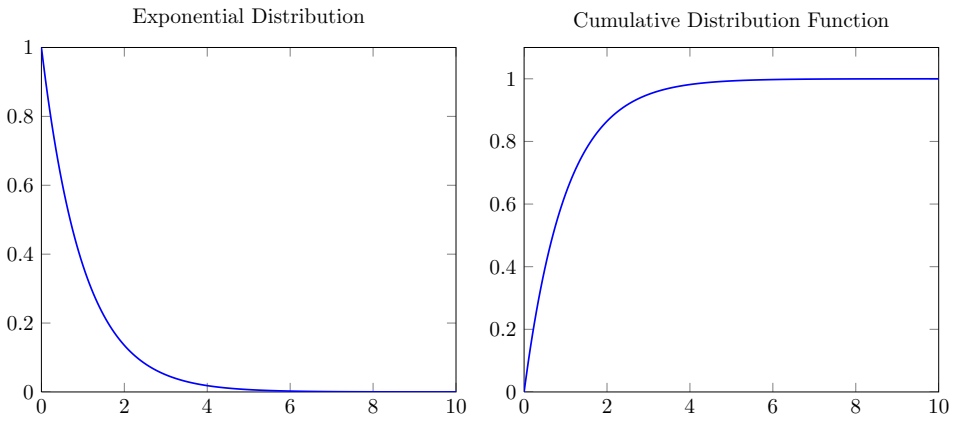


Fig. 2.14. Left: Exponential distribution with $\lambda = 1$. Right: associated cumulative distribution function. When sampling, we draw x uniformly at random from $U[0, 1]$ and compute the inverse.

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 $x = -\lambda^{-1} \log z$.

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*

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

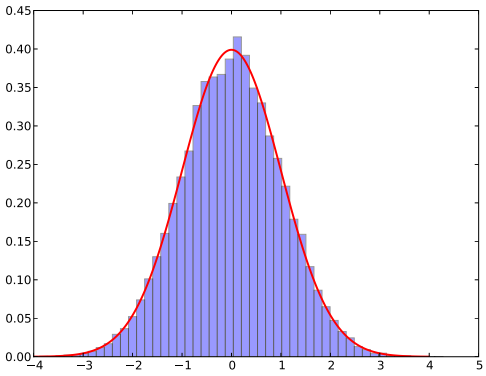


Fig. 2.15. Red: true density of the standard normal distribution (red line) is contrasted with the histogram of 20,000 random variables generated by the Box-Müller transform.

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* 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

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

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.*

Note that this time we did not need to carry out any sophisticated measure

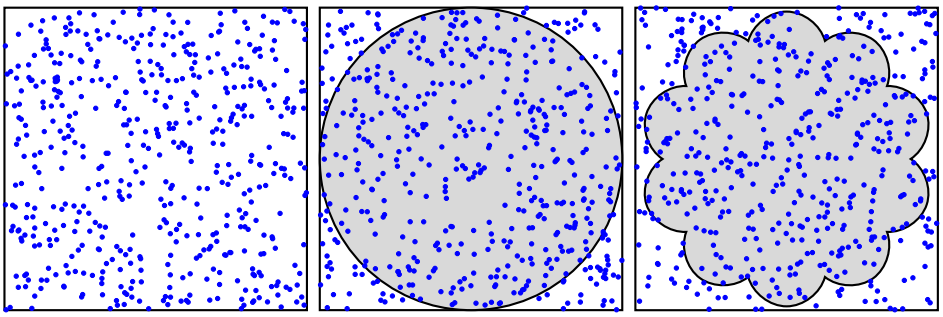


Fig. 2.16. Rejection sampler. Left: samples drawn from the uniform distribution on $[0,1]^2$. Middle: the samples drawn from the uniform distribution on the unit disc are all the points in the grey shaded area. Right: the same procedure allows us to sample uniformly from arbitrary sets.

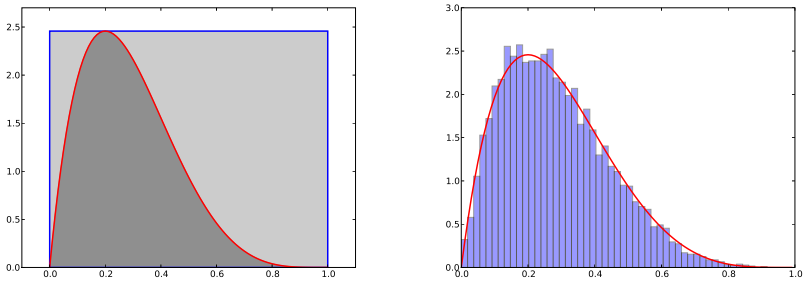


Fig. 2.17. Accept reject sampling for the Beta$(2,5)$ distribution. Left: Samples are generated uniformly from the blue rectangle (shaded area). Only those samples which fall under the red curve of the Beta$(2,5)$ distribution (darkly shaded area) are accepted. Right: The true density of the Beta$(2,5)$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

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* $\sqrt{\frac{\pi}{2e}}$.

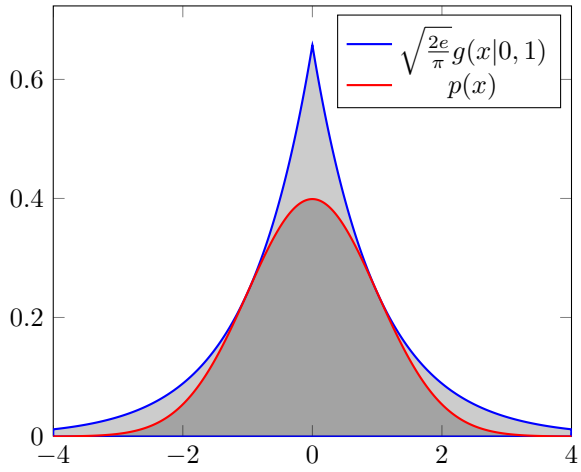


Fig. 2.18. Rejection sampling for the Normal distribution (red curve). Samples are generated uniformly from the Laplace distribution rescaled by $\sqrt{2e/\pi}$. Only those samples which fall under the red curve of the standard normal distribution (darkly shaded area) are accepted.

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

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

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})

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

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 $\theta \sim \text{Unif}[0, 2\pi]$.*

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

Problem 2.26 (Accept reject sampling for the unit disk {2}) *Give at least **TWO** 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 $U \sim \text{Unif}[0, 1]$ 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:* $\{(x, y) : \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} \leq 1\}$*.*

3

Optimization

Optimization plays an increasingly important role in machine learning. For instance, many machine learning algorithms minimize a regularized risk functional:

$$\min_f \; J(f) := \lambda\Omega(f) + R_{\text{emp}}(f), \tag{3.1}$$

with the empirical risk

$$R_{\text{emp}}(f) := \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i). \tag{3.2}$$

Here x_i are the training instances and y_i are the corresponding labels. l the loss function measures the discrepancy between y and the predictions $f(x_i)$. Finding the optimal f involves solving an optimization problem.

This chapter provides a self-contained overview of some basic concepts and tools from optimization, especially geared towards solving machine learning problems. In terms of concepts, we will cover topics related to convexity, duality, and Lagrange multipliers. In terms of tools, we will cover a variety of optimization algorithms including gradient descent, stochastic gradient descent, Newton's method, and Quasi-Newton methods. We will also look at some specialized algorithms tailored towards solving Linear Programming and Quadratic Programming problems which often arise in machine learning problems.

3.1 Preliminaries

Minimizing an arbitrary function is, in general, very difficult, but if the objective function to be minimized is convex then things become considerably simpler. As we will see shortly, the key advantage of dealing with convex functions is that a local optima is also the global optima. Therefore, well developed tools exist to find the global minima of a convex function. Consequently, many machine learning algorithms are now formulated in terms of convex optimization problems. We briefly review the concept of convex sets and functions in this section.

3.1.1 Convex Sets

Definition 3.1 (Convex Set) *A subset C of $\mathbb{R}^n$ is said to be convex if $(1-\lambda)x + \lambda y \in C$ whenever $x \in C, y \in C$ and $0 < \lambda < 1$.*

Intuitively, what this means is that the line joining any two points x and y from the set C lies inside C (see Figure 3.1). It is easy to see (Exercise 3.1) that intersections of convex sets are also convex.

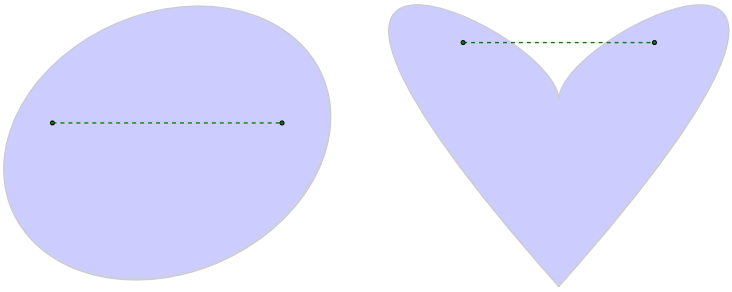


Fig. 3.1. The convex set (left) contains the line joining any two points that belong to the set. A non-convex set (right) does not satisfy this property.

A vector sum $\sum_i \lambda_i x_i$ is called a convex combination if $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$. Convex combinations are helpful in defining a convex hull:

Definition 3.2 (Convex Hull) *The convex hull,* $\mathrm{conv}(X)$*, of a finite subset $X = \{x_1, \ldots, x_n\}$ of $\mathbb{R}^n$ consists of all convex combinations of $x_1, \ldots, x_n$.*

3.1.2 Convex Functions

Let f be a real valued function defined on a set $X \subset \mathbb{R}^n$. The set

$$\{(x, \mu) : x \in X, \mu \in \mathbb{R}, \mu \geq f(x)\} \tag{3.3}$$

is called the *epigraph* of f. The function f is defined to be a convex function if its epigraph is a convex set in $\mathbb{R}^{n+1}$. An equivalent, and more commonly used, definition (Exercise 3.5) is as follows (see Figure 3.2 for geometric intuition):

Definition 3.3 (Convex Function) *A function f defined on a set X is called convex if, for any $x, x' \in X$ and any $0 < \lambda < 1$ such that $\lambda x + (1-\lambda)x' \in X$, we have*

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x'). \tag{3.4}$$

A function f is called strictly *convex if*

$$f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda)f(x') \tag{3.5}$$

whenever $x \neq x'$.

In fact, the above definition can be extended to show that if f is a convex function and $\lambda_i \geq 0$ with $\sum_i \lambda_i = 1$ then

$$f\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i f(x_i). \tag{3.6}$$

The above inequality is called the *Jensen's inequality* (problem).

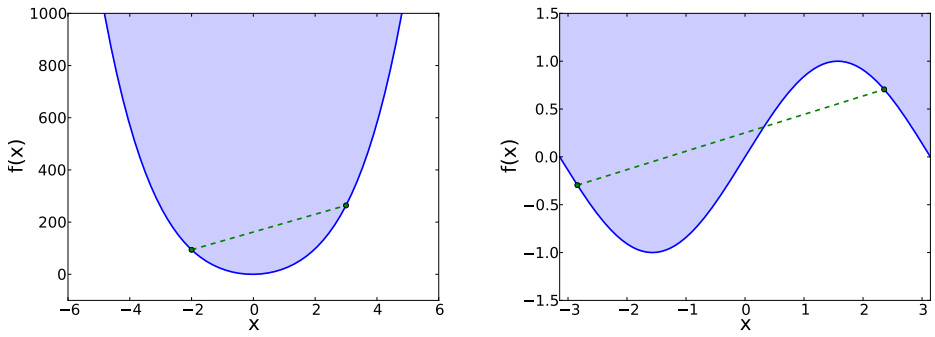


Fig. 3.2. A convex function (left) satisfies (3.4); the shaded region denotes its epigraph. A nonconvex function (right) does not satisfy (3.4).

If $f : X \to \mathbb{R}$ is differentiable, then f is convex if, and only if,

$$f(x') \geq f(x) + \langle x' - x, \nabla f(x) \rangle \text{ for all } x, x' \in X. \tag{3.7}$$

In other words, the first order Taylor approximation lower bounds the convex function universally (see Figure 3.4). Here and in the rest of the chapter $\langle x, y \rangle$ denotes the Euclidean dot product between vectors x and y, that is,

$$\langle x, y \rangle := \sum_i x_i y_i. \tag{3.8}$$

If f is twice differentiable, then f is convex if, and only if, its Hessian is positive semi-definite, that is,

$$\nabla^2 f(x) \succeq 0. \tag{3.9}$$

For twice differentiable strictly convex functions, the Hessian matrix is positive definite, that is, $\nabla^2 f(x) \succ 0$. We briefly summarize some operations which preserve convexity:

Addition	If f_1 and f_2 are convex, then $f_1 + f_2$ is also convex.
Scaling	If f is convex, then αf is convex for $\alpha > 0$.
Affine Transform	If f is convex, then $g(x) = f(Ax + b)$ for some matrix A and vector b is also convex.
Adding a Linear Function	If f is convex, then $g(x) = f(x) + \langle a, x \rangle$ for some vector a is also convex.
Subtracting a Linear Function	If f is convex, then $g(x) = f(x) - \langle a, x \rangle$ for some vector a is also convex.
Pointwise Maximum	If f_i are convex, then $g(x) = \max_i f_i(x)$ is also convex.
Scalar Composition	If $f(x) = h(g(x))$, then f is convex if a) g is convex, and h is convex, non-decreasing or b) g is concave, and h is convex, non-increasing.

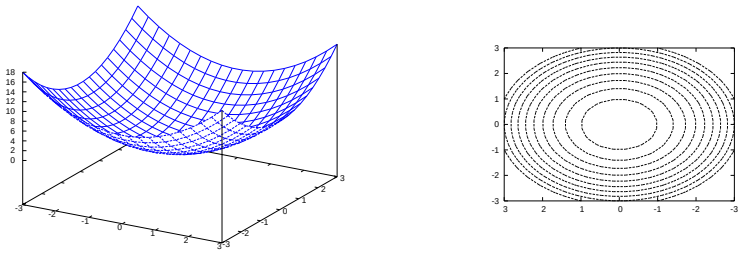


Fig. 3.3. Left: Convex Function in two variables. Right: the corresponding convex below-sets $\{x|f(x) \leq c\}$, for different values of c. This is also called a contour plot.

There is an intimate relation between convex functions and convex sets. For instance, the following lemma show that the *below sets* (level sets) of convex functions, sets for which $f(x) \leq c$, are convex.

Lemma 3.4 (Below-Sets of Convex Functions) *Denote by $f : X \to \mathbb{R}$ a convex function. Then the set*

$$X_c := \{x \,|\, x \in X \text{ and } f(x) \leq c\}, \text{ for all } c \in \mathbb{R}, \tag{3.10}$$

is convex.

Proof For any $x, x' \in X_c$, we have $f(x), f(x') \leq c$. Moreover, since f is convex, we also have

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x') \leq c \text{ for all } 0 < \lambda < 1. \tag{3.11}$$

Hence, for all $0 < \lambda < 1$, we have $(\lambda x + (1-\lambda)x') \in X_c$, which proves the claim. Figure 3.3 depicts this situation graphically. ■

As we hinted in the introduction of this chapter, minimizing an arbitrary function on a (possibly not even compact) set of arguments can be a difficult task, and will most likely exhibit many local minima. In contrast, minimization of a convex objective function on a convex set exhibits exactly one *global* minimum. We now prove this property.

Theorem 3.5 (Minima on Convex Sets) *If the convex function $f : X \to \mathbb{R}$ attains its minimum, then the set of $x \in X$, for which the minimum value is attained, is a convex set. Moreover, if f is strictly convex, then this set contains a single element.*

Proof Denote by c the minimum of f on X. Then the set $X_c := \{x | x \in X \text{ and } f(x) \leq c\}$ is clearly convex.

If f is strictly convex, then for any two distinct $x, x' \in X_c$ and any $0 < \lambda < 1$ we have

$$f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda) f(x') = \lambda c + (1-\lambda)c = c,$$

which contradicts the assumption that f attains its minimum on X_c. Therefore X_c must contain only a single element. ∎

As the following lemma shows, the minimum point can be characterized precisely.

Lemma 3.6 *Let $f : X \to \mathbb{R}$ be a differentiable convex function. Then x is a minimizer of f, if, and only if,*

$$\langle x' - x, \nabla f(x) \rangle \geq 0 \text{ for all } x'. \tag{3.12}$$

Proof To show the forward implication, suppose that x is the optimum but (3.12) does not hold, that is, there exists an x' for which

$$\langle x' - x, \nabla f(x) \rangle < 0.$$

Consider the line segment $z(\lambda) = (1-\lambda)x + \lambda x'$, with $0 < \lambda < 1$. Since X is convex, $z(\lambda)$ lies in X. On the other hand,

$$\frac{d}{d\lambda} f(z(\lambda))\Big|_{\lambda=0} = \langle x' - x, \nabla f(x) \rangle < 0,$$

which shows that for small values of λ we have $f(z(\lambda)) < f(x)$, thus showing that x is not optimal.

The reverse implication follows from (3.7) by noting that $f(x') \geq f(x)$, whenever (3.12) holds.

∎

One way to ensure that (3.12) holds is to set $\nabla f(x) = 0$. In other words, minimizing a convex function is equivalent to finding a x such that $\nabla f(x) = 0$. Therefore, the first order conditions are both necessary and sufficient when minimizing a convex function.

3.1.3 Subgradients

So far, we worked with differentiable convex functions. The subgradient is a generalization of gradients appropriate for convex functions, including those which are not necessarily smooth.

Definition 3.7 (Subgradient) *Suppose x is a point where a convex function f is finite. Then a subgradient is the normal vector of any tangential supporting hyperplane of f at x. Formally μ is called a subgradient of f at x if, and only if,*

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle \text{ for all } x'. \tag{3.13}$$

The set of all subgradients at a point is called the subdifferential, and is denoted by $\partial_x f(x)$. If this set is not empty then f is said to be *subdifferentiable at x*. On the other hand, if this set is a singleton then, the function is said to be *differentiable* at x. In this case we use $\nabla f(x)$ to denote the gradient of f. Convex functions are subdifferentiable everywhere in their domain. We now state some simple rules of subgradient calculus:

Addition	$\partial_x(f_1(x) + f_2(x)) = \partial_x f_1(x) + \partial_x f_2(x)$
Scaling	$\partial_x \alpha f(x) = \alpha \partial_x f(x)$, for $\alpha > 0$
Affine Transform	If $g(x) = f(Ax + b)$ for some matrix A and vector b, then $\partial_x g(x) = A^\top \partial_y f(y)$.
Pointwise Maximum	If $g(x) = \max_i f_i(x)$ then $\partial g(x) = \text{conv}(\partial_x f_{i'})$ where $i' \in \text{argmax}_i f_i(x)$.

The definition of a subgradient can also be understood geometrically. As illustrated by Figure 3.4, a differentiable convex function is always lower bounded by its first order Taylor approximation. This concept can be extended to non-smooth functions via subgradients, as Figure 3.5 shows.

By using more involved concepts, the proof of Lemma 3.6 can be extended to subgradients. In this case, minimizing a convex nonsmooth function entails finding a x such that $0 \in \partial f(x)$.

3.1.4 Strongly Convex Functions

When analyzing optimization algorithms, it is sometimes easier to work with strongly convex functions, which generalize the definition of convexity.

Definition 3.8 (Strongly Convex Function) *A convex function f is σ-strongly convex if, and only if, there exists a constant $\sigma > 0$ such that the function $f(x) - \frac{\sigma}{2}\|x\|^2$ is convex.*

The constant σ is called the modulus of strong convexity of f. If f is twice differentiable, then there is an equivalent, and perhaps easier, definition of strong convexity: f is strongly convex if there exists a σ such that

$$\nabla^2 f(x) \succeq \sigma I. \tag{3.14}$$

In other words, the smallest eigenvalue of the Hessian of f is *uniformly lower bounded* by σ everywhere. Some important examples of strongly convex functions include:

Example 3.1 (Squared Euclidean Norm) *The function $f(x) = \frac{\lambda}{2}\|x\|^2$ is λ-strongly convex.*

Example 3.2 (Negative Entropy) *Let $\Delta^n = \{x \text{ s.t. } \sum_i x_i = 1 \text{ and } x_i \geq 0\}$ be the n dimensional simplex, and $f : \Delta^n \to \mathbb{R}$ be the negative entropy:*

$$f(x) = \sum_i x_i \log x_i. \tag{3.15}$$

Then f is 1-strongly convex with respect to the $\|\cdot\|_1$ norm on the simplex (see Problem 3.7).

If f is a σ-strongly convex function then one can show the following properties (Exercise 3.8). Here x, x' are arbitrary and $\mu \in \partial f(x)$ and $\mu' \in \partial f(x')$.

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle + \frac{\sigma}{2}\|x' - x\|^2 \tag{3.16}$$

$$f(x') \leq f(x) + \langle x' - x, \mu \rangle + \frac{1}{2\sigma}\|\mu' - \mu\|^2 \tag{3.17}$$

$$\langle x - x', \mu - \mu' \rangle \geq \sigma\|x - x'\|^2 \tag{3.18}$$

$$\langle x - x', \mu - \mu' \rangle \leq \frac{1}{\sigma}\|\mu - \mu'\|^2. \tag{3.19}$$

3.1.5 Convex Functions with Lipschitz Continous Gradient

A somewhat symmetric concept to strong convexity is the Lipschitz continuity of the gradient. As we will see later they are connected by Fenchel duality.

Definition 3.9 (Lipschitz Continuous Gradient) *A differentiable convex function f is said to have a Lipschitz continuous gradient, if there exists a constant $L > 0$, such that*

$$\left\|\nabla f(x) - \nabla f(x')\right\| \leq L \left\|x - x'\right\| \quad \forall x, x'. \tag{3.20}$$

As before, if f is twice differentiable, then there is an equivalent, and perhaps easier, definition of Lipschitz continuity of the gradient: f has a Lipschitz continuous gradient strongly convex if there exists a L such that

$$LI \succeq \nabla^2 f(x). \tag{3.21}$$

In other words, the largest eigenvalue of the Hessian of f is *uniformly upper bounded* by L everywhere. If f has a Lipschitz continuous gradient with modulus L, then one can show the following properties (Exercise 3.9).

$$f(x') \leq f(x) + \left\langle x' - x, \nabla f(x)\right\rangle + \frac{L}{2}\left\|x - x'\right\|^2 \tag{3.22}$$

$$f(x') \geq f(x) + \left\langle x' - x, \nabla f(x)\right\rangle + \frac{1}{2L}\left\|\nabla f(x) - \nabla f(x')\right\|^2 \tag{3.23}$$

$$\left\langle x - x', \nabla f(x) - \nabla f(x')\right\rangle \leq L\left\|x - x'\right\|^2 \tag{3.24}$$

$$\left\langle x - x', \nabla f(x) - \nabla f(x')\right\rangle \geq \frac{1}{L}\left\|\nabla f(x) - \nabla f(x')\right\|^2. \tag{3.25}$$

3.1.6 Fenchel Duality

The Fenchel conjugate of a function f is given by

$$f^*(x^*) = \sup_x \left\{\langle x, x^*\rangle - f(x)\right\}. \tag{3.26}$$

Even if f is not convex, the Fechel conjugate which is written as a supremum over linear functions is always convex. Some rules for computing Fenchel duals are summarized in Table 3.1.6. If f is convex and its epigraph (3.3) is a closed convex set, then $f^{**} = f$. If f and f^* are convex, then they satisfy the so-called Fenchel-Young inequality

$$f(x) + f^*(x^*) \geq \langle x, x^*\rangle \text{ for all } x, x^*. \tag{3.27}$$

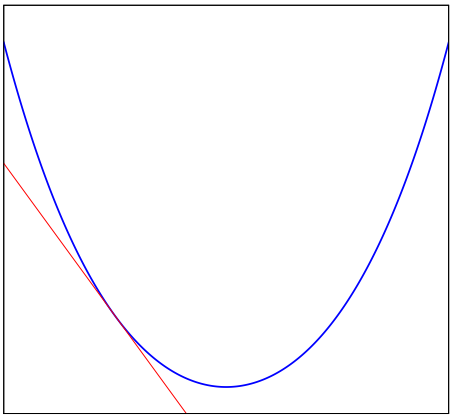


Fig. 3.4. A convex function is always lower bounded by its first order Taylor approximation. This is true even if the function is not differentiable (see Figure 3.5)

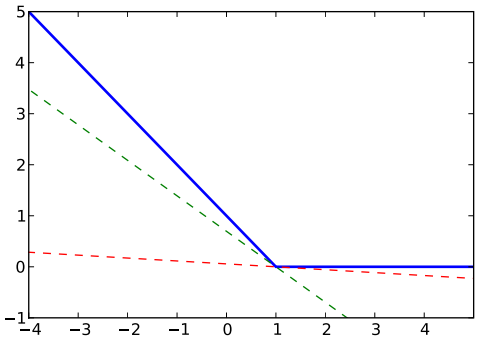


Fig. 3.5. Geometric intuition of a subgradient. The nonsmooth convex function (solid blue) is only subdifferentiable at the "kink" points. We illustrate two of its subgradients (dashed green and red lines) at a "kink" point which are tangential to the function. The normal vectors to these lines are subgradients. Observe that the first order Taylor approximations obtained by using the subgradients lower bounds the convex function.

This inequality becomes an equality whenever $x^* \in \partial f(x)$, that is,

$$f(x) + f^*(x^*) = \langle x, x^* \rangle \text{ for all } x \text{ and } x^* \in \partial f(x). \tag{3.28}$$

Strong convexity (Section 3.1.4) and Lipschitz continuity of the gradient

Table 3.1. *Rules for computing Fenchel Duals*

Scalar Addition	If $g(x) = f(x) + \alpha$ then $g^*(x^*) = f^*(x^*) - \alpha$.
Function Scaling	If $\alpha > 0$ and $g(x) = \alpha f(x)$ then $g^*(x^*) = \alpha f^*(x^*/\alpha)$.
Parameter Scaling	If $\alpha \neq 0$ and $g(x) = f(\alpha x)$ then $g^*(x^*) = f^*(x^*/\alpha)$
Linear Transformation	If A is an invertible matrix then $(f \circ A)^* = f^* \circ (A^{-1})^*$.
Shift	If $g(x) = f(x - x_0)$ then $g^*(x^*) = f^*(x^*) + \langle x^*, x_0 \rangle$.
Sum	If $g(x) = f_1(x) + f_2(x)$ then $g^*(x^*) = \inf \{f_1^*(x_1^*) + f_2^*(x_2^*)$ s.t. $x_1^* + x_2^* = x^*\}$.
Pointwise Infimum	If $g(x) = \inf f_i(x)$ then $g^*(x^*) = \sup_i f_i^*(x^*)$.

(Section 3.1.5) are related by Fenchel duality according to the following lemma, which we state without proof.

Lemma 3.10 (Theorem 4.2.1 and 4.2.2 [HUL93])

(i) *If f is σ-strongly convex, then f^* has a Lipschitz continuous gradient with modulus $\frac{1}{\sigma}$.*
(ii) *If f is convex and has a Lipschitz continuous gradient with modulus L, then f^* is $\frac{1}{L}$-strongly convex.*

Next we describe some convex functions and their Fenchel conjugates.

Example 3.3 (Squared Euclidean Norm) *Whenever $f(x) = \frac{1}{2}\|x\|^2$ we have $f^*(x^*) = \frac{1}{2}\|x^*\|^2$, that is, the squared Euclidean norm is its own conjugate.*

Example 3.4 (Negative Entropy) *The Fenchel conjugate of the negative entropy (3.15) is*

$$f^*(x^*) = \log \sum_i \exp(x_i^*).$$

3.1.7 Bregman Divergence

Let f be a differentiable convex function. The Bregman divergence defined by f is given by

$$\Delta_f(x, x') = f(x) - f(x') - \langle x - x', \nabla f(x') \rangle . \tag{3.29}$$

Also see Figure 3.6. Here are some well known examples.

Example 3.5 (Square Euclidean Norm) *Set $f(x) = \frac{1}{2}\|x\|^2$. Clearly, $\nabla f(x) = x$ and therefore*

$$\Delta_f(x, x') = \frac{1}{2}\|x\|^2 - \frac{1}{2}\|x'\|^2 - \langle x - x', x' \rangle = \frac{1}{2}\|x - x'\|^2 .$$

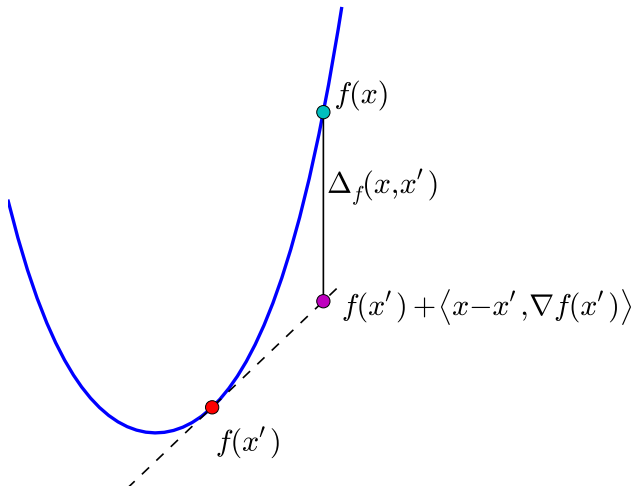


Fig. 3.6. $f(x)$ is the value of the function at x, while $f(x')+\langle x-x', \nabla f(x')\rangle$ denotes the first order Taylor expansion of f around x', evaluated at x. The difference between these two quantities is the Bregman divergence, as illustrated.

Example 3.6 (Relative Entropy) *Let f be the un-normalized entropy*

$$f(x) = \sum_i \left(x_i \log x_i - x_i\right). \tag{3.30}$$

One can calculate $\nabla f(x) = \log x$, where $\log x$ is the component wise logarithm of the entries of x, and write the Bregman divergence

$$\begin{aligned}\Delta_f(x,x') &= \sum_i x_i \log x_i - \sum_i x_i - \sum_i x'_i \log x'_i + \sum_i x'_i - \langle x - x', \log x' \rangle \\ &= \sum_i \left(x_i \log\left(\frac{x_i}{x'_i}\right) + x'_i - x_i \right).\end{aligned}$$

Example 3.7 (p-norm) *Let f be the square p-norm*

$$f(x) = \frac{1}{2}\|x\|_p^2 = \frac{1}{2}\left(\sum_i x_i^p\right)^{2/p}. \tag{3.31}$$

We say that the q-norm is dual to the p-norm whenever $\frac{1}{p}+\frac{1}{q}=1$. *One can verify (Problem 3.12) that the i-th component of the gradient* $\nabla f(x)$ *is*

$$\nabla_{x_i} f(x) = \frac{\operatorname{sign}(x_i)\,|x_i|^{p-1}}{\|x\|_p^{p-2}}. \tag{3.32}$$

The corresponding Bregman divergence is

$$\Delta_f(x,x') = \frac{1}{2}\|x\|_p^2 - \frac{1}{2}\|x'\|_p^2 - \sum_i (x_i - x_i')\frac{\operatorname{sign}(x_i')\,|x_i'|^{p-1}}{\|x'\|_p^{p-2}}.$$

The following properties of the Bregman divergence immediately follow:

- $\Delta_f(x,x')$ is convex in x.
- $\Delta_f(x,x') \geq 0$.
- Δ_f may not be symmetric, that is, in general $\Delta_f(x,x') \neq \Delta_f(x',x)$.
- $\nabla_x \Delta_f(x,x') = \nabla f(x) - \nabla f(x')$.

The next lemma establishes another important property.

Lemma 3.11 *The Bregman divergence (3.29) defined by a differentiable convex function* f *satisfies*

$$\Delta_f(x,y) + \Delta_f(y,z) - \Delta_f(x,z) = \langle \nabla f(z) - \nabla f(y), x - y\rangle\,. \tag{3.33}$$

Proof

$$\begin{aligned}\Delta_f(x,y) + \Delta_f(y,z) &= f(x) - f(y) - \langle x-y, \nabla f(y)\rangle + f(y) - f(z) - \langle y - z, \nabla f(z)\rangle \\ &= f(x) - f(z) - \langle x-y, \nabla f(y)\rangle - \langle y-z, \nabla f(z)\rangle \\ &= \Delta_f(x,z) + \langle \nabla f(z) - \nabla f(y), x - y\rangle\,.\end{aligned}$$

■

3.2 Unconstrained Smooth Convex Minimization

In this section we will describe various methods to minimize a smooth convex objective function.

3.2.1 Minimizing a One-Dimensional Convex Function

As a warm up let us consider the problem of minimizing a smooth one dimensional convex function $J : \mathbb{R} \to \mathbb{R}$ in the interval $[L, U]$. This seemingly

Algorithm 3.1 Interval Bisection

Input: L, U, precision ϵ
Set $t = 0$, $a_0 = L$ and $b_0 = U$
while $(b_t - a_t) \cdot J'(U) > \epsilon$ **do**
  if $J'(\frac{a_t+b_t}{2}) > 0$ **then**
    $a_{t+1} = a_t$ and $b_{t+1} = \frac{a_t+b_t}{2}$
  else
    $a_{t+1} = \frac{a_t+b_t}{2}$ and $b_{t+1} = b_t$
  end if
  $t = t + 1$
end while
Return: $\frac{a_t+b_t}{2}$

simple problem has many applications. As we will see later, many optimization methods find a direction of descent and minimize the objective function along this direction[1]; this subroutine is called a line search. Algorithm 3.1 depicts a simple line search routine based on interval bisection.

Before we show that Algorithm 3.1 converges, let us first derive an important property of convex functions of one variable. For a differentiable one-dimensional convex function J (3.7) reduces to

$$J(w) \geq J(w') + (w - w') \cdot J'(w'), \tag{3.34}$$

where $J'(w)$ denotes the gradient of J. Exchanging the role of w and w' in (3.34), we can write

$$J(w') \geq J(w) + (w' - w) \cdot J'(w). \tag{3.35}$$

Adding the above two equations yields

$$(w - w') \cdot (J'(w) - J'(w')) \geq 0. \tag{3.36}$$

If $w \geq w'$, then this implies that $J'(w) \geq J'(w')$. In other words, the gradient of a one dimensional convex function is monotonically non-decreasing.

Recall that minimizing a convex function is equivalent to finding w^* such that $J'(w^*) = 0$. Furthermore, it is easy to see that the interval bisection maintains the invariant $J'(a_t) < 0$ and $J'(b_t) > 0$. This along with the monotonicity of the gradient suffices to ensure that $w^* \in (a_t, b_t)$. Setting $w = w^*$ in (3.34), and using the monotonicity of the gradient allows us to

[1] If the objective function is convex, then the one dimensional function obtained by restricting it along the search direction is also convex (Exercise 3.10).

write for any $w' \in (a_t, b_t)$

$$J(w') - J(w^*) \leq (w' - w^*) \cdot J'(w') \leq (b_t - a_t) \cdot J'(U). \tag{3.37}$$

Since we halve the interval (a_t, b_t) at every iteration, it follows that $(b_t - a_t) = (U - L)/2^t$. Therefore

$$J(w') - J(w^*) \leq \frac{(U - L) \cdot J'(U)}{2^t}, \tag{3.38}$$

for all $w' \in (a_t, b_t)$. In other words, to find an ϵ-accurate solution, that is, $J(w') - J(w^*) \leq \epsilon$ we only need $\log(U - L) + \log J'(U) + \log(1/\epsilon) < t$ iterations. An algorithm which converges to an ϵ accurate solution in $O(\log(1/\epsilon))$ iterations is said to be linearly convergent.

For multi-dimensional objective functions, one cannot rely on the monotonicity property of the gradient. Therefore, one needs more sophisticated optimization algorithms, some of which we now describe.

3.2.2 Coordinate Descent

Coordinate descent is conceptually the simplest algorithm for minimizing a multidimensional smooth convex function $J : \mathbb{R}^n \to \mathbb{R}$. At every iteration select a coordinate, say i, and update

$$w_{t+1} = w_t - \eta_t e_i. \tag{3.39}$$

Here e_i denotes the i-th basis vector, that is, a vector with one at the i-th coordinate and zeros everywhere else, while $\eta_t \in \mathbb{R}$ is a non-negative scalar step size. One could, for instance, minimize the one dimensional convex function $J(w_t - \eta e_i)$ to obtain the stepsize η_t. The coordinates can either be selected cyclically, that is, $1, 2, \ldots, n, 1, 2, \ldots$ or greedily, that is, the coordinate which yields the maximum reduction in function value.

Even though coordinate descent can be shown to converge if J has a Lipschitz continuous gradient [LT92], in practice it can be quite slow. However, if a high precision solution is not required, as is the case in some machine learning applications, coordinate descent is often used because a) the cost per iteration is very low and b) the speed of convergence may be acceptable especially if the variables are loosely coupled.

3.2.3 Gradient Descent

Gradient descent (also widely known as steepest descent) is an optimization technique for minimizing multidimensional smooth convex objective functions of the form $J : \mathbb{R}^n \to \mathbb{R}$. The basic idea is as follows: Given a location

w_t at iteration t, compute the gradient $\nabla J(w_t)$, and update

$$w_{t+1} = w_t - \eta_t \nabla J(w_t), \tag{3.40}$$

where η_t is a scalar stepsize. See Algorithm 3.2 for details. Different variants of gradient descent depend on how η_t is chosen:

Exact Line Search: Since $J(w_t - \eta \nabla J(w_t))$ is a one dimensional convex function in η, one can use the Algorithm 3.1 to compute:

$$\eta_t = \operatorname*{argmin}_{\eta} J(w_t - \eta \nabla J(w_t)). \tag{3.41}$$

Instead of the simple bisecting line search more sophisticated line searches such as the More-Thuente line search or the golden bisection rule can also be used to speed up convergence (see [NW99] Chapter 3 for an extensive discussion).

Inexact Line Search: Instead of minimizing $J(w_t - \eta \nabla J(w_t))$ we could simply look for a stepsize which results in sufficient decrease in the objective function value. One popular set of sufficient decrease conditions is the Wolfe conditions

$$J(w_{t+1}) \leq J(w_t) + c_1 \eta_t \langle \nabla J(w_t), w_{t+1} - w_t \rangle \text{ (sufficient decrease)} \tag{3.42}$$

$$\langle \nabla J(w_{t+1}), w_{t+1} - w_t \rangle \geq c_2 \langle \nabla J(w_t), w_{t+1} - w_t \rangle \text{ (curvature)} \tag{3.43}$$

with $0 < c_1 < c_2 < 1$ (see Figure 3.7). The Wolfe conditions are also called the Armijio-Goldstein conditions. If only sufficient decrease (3.42) alone is enforced, then it is called the Armijio rule.

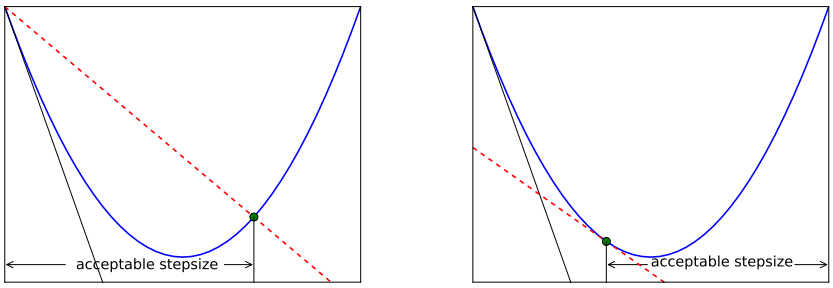


Fig. 3.7. The sufficient decrease condition (left) places an upper bound on the acceptable stepsizes while the curvature condition (right) places a lower bound on the acceptable stepsizes.

Algorithm 3.2 Gradient Descent

1: **Input:** Initial point w_0, gradient norm tolerance ϵ
2: Set $t = 0$
3: **while** $\|\nabla J(w_t)\| \geq \epsilon$ **do**
4: $\quad w_{t+1} = w_t - \eta_t \nabla J(w_t)$
5: $\quad t = t + 1$
6: **end while**
7: **Return:** w_t

Decaying Stepsize: Instead of performing a line search at every iteration, one can use a stepsize which decays according to a fixed schedule, for example, $\eta_t = 1/\sqrt{t}$. In Section 3.2.4 we will discuss the decay schedule and convergence rates of a generalized version of gradient descent.

Fixed Stepsize: Suppose J has a Lipschitz continuous gradient with modulus L. Using (3.22) and the gradient descent update $w_{t+1} = w_t - \eta_t \nabla J(w_t)$ one can write

$$J(w_{t+1}) \leq J(w_t) + \langle \nabla J(w_t), w_{t+1} - w_t \rangle + \frac{L}{2} \|w_{t+1} - w_t\| \tag{3.44}$$

$$= J(w_t) - \eta_t \|\nabla J(w_t)\|^2 + \frac{L\eta_t^2}{2} \|\nabla J(w_t)\|^2 . \tag{3.45}$$

Minimizing (3.45) as a function of η_t clearly shows that the upper bound on $J(w_{t+1})$ is minimized when we set $\eta_t = \frac{1}{L}$, which is the fixed stepsize rule.

Theorem 3.12 *Suppose J has a Lipschitz continuous gradient with modulus L. Then Algorithm 3.2 with a fixed stepsize $\eta_t = \frac{1}{L}$ will return a solution w_t with $\|\nabla J(w_t)\| \leq \epsilon$ in at most $O(1/\epsilon^2)$ iterations.*

Proof Plugging in $\eta_t = \frac{1}{L}$ and rearranging (3.45) obtains

$$\frac{1}{2L} \|\nabla J(w_t)\|^2 \leq J(w_t) - J(w_{t+1}) \tag{3.46}$$

Summing this inequality

$$\frac{1}{2L} \sum_{t=0}^{T} \|\nabla J(w_t)\|^2 \leq J(w_0) - J(w_T) \leq J(w_0) - J(w^*),$$

which clearly shows that $\|\nabla J(w_t)\| \to 0$ as $t \to \infty$. Furthermore, we can write the following simple inequality:

$$\|\nabla J(w_T)\| \leq \sqrt{\frac{2L(J(w_0) - J(w^*))}{T+1}}.$$

Solving for

$$\sqrt{\frac{2L(J(w_0) - J(w^*))}{T+1}} = \epsilon$$

shows that T is $O(1/\epsilon^2)$ as claimed. ■

If in addition to having a Lipschitz continuous gradient, if J is σ-strongly convex, then more can be said. First, one can translate convergence in $\|\nabla J(w_t)\|$ to convergence in function values. Towards this end, use (3.17) to write

$$J(w_t) \leq J(w^*) + \frac{1}{2\sigma} \|\nabla J(w_t)\|^2 .$$

Therefore, it follows that whenever $\|\nabla J(w_t)\| < \epsilon$ we have $J(w_t) - J(w^*) < \epsilon^2/2\sigma$. Furthermore, we can strengthen the rates of convergence.

Theorem 3.13 *Assume everything as in Theorem 3.12. Moreover assume that J is σ-strongly convex, and let $c := 1 - \frac{\sigma}{L}$. Then $J(w_t) - J(w^*) \leq \epsilon$ after at most*

$$\frac{\log((J(w_0) - J(w^*))/\epsilon)}{\log(1/c)} \tag{3.47}$$

iterations.

Proof Combining (3.46) with $\|\nabla J(w_t)\|^2 \geq 2\sigma(J(w_t) - J(w^*))$, and using the definition of c one can write

$$c(J(w_t) - J(w^*)) \geq J(w_{t+1}) - J(w^*).$$

Applying the above equation recursively

$$c^T(J(w_0) - J(w^*)) \geq J(w_T) - J(w^*).$$

Solving for

$$\epsilon = c^T(J(w_0) - J(w^*))$$

and rearranging yields (3.47). ■

When applied to practical problems which are not strongly convex gradient descent yields a low accuracy solution within a few iterations. However, as the iterations progress the method "stalls" and no further increase in accuracy is obtained because of the $O(1/\epsilon^2)$ rates of convergence. On the other hand, if the function is strongly convex, then gradient descent converges linearly, that is, in $O(\log(1/\epsilon))$ iterations. However, the number

of iterations depends inversely on $\log(1/c)$. If we approximate $\log(1/c) = -\log(1-\sigma/L) \approx \sigma/L$, then it shows that convergence depends on the ratio L/σ. This ratio is called the *condition number* of a problem. If the problem is well conditioned, *i.e.,* $\sigma \approx L$ then gradient descent converges extremely fast. In contrast, if $\sigma \ll L$ then gradient descent requires many iterations. This is best illustrated with an example: Consider the quadratic objective function

$$J(w) = \frac{1}{2} w^\top A w - bw, \tag{3.48}$$

where $A \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, and $b \in \mathbb{R}^n$ is any arbitrary vector.

Recall that a twice differentiable function is σ-strongly convex and has a Lipschitz continuous gradient with modulus L if and only if its Hessian satisfies $LI \succeq \nabla^2 J(w) \succeq \sigma I$ (see (3.14) and (3.21)). In the case of the quadratic function (3.48) $\nabla^2 J(w) = A$ and hence $\sigma = \lambda_{\min}$ and $L = \lambda_{\max}$, where $\lambda_{\min}$ (respectively $\lambda_{\max}$) denotes the minimum (respectively maximum) eigenvalue of A. One can thus change the condition number of the problem by varying the eigen-spectrum of the matrix A. For instance, if we set A to the $n \times n$ identity matrix, then $\lambda_{\max} = \lambda_{\min} = 1$ and hence the problem is well conditioned. In this case, gradient descent converges very quickly to the optimal solution. We illustrate this behavior on a two dimensional quadratic function in Figure 3.8 (right).

On the other hand, if we choose A such that $\lambda_{\max} \gg \lambda_{\min}$ then the problem (3.48) becomes ill-conditioned. In this case gradient descent exhibits zigzagging and slow convergence as can be seen in Figure 3.8 (left). Because of these shortcomings, gradient descent is not widely used in practice. A number of different algorithms we described below can be understood as explicitly or implicitly changing the condition number of the problem to accelerate convergence.

3.2.4 Mirror Descent

One way to motivate gradient descent is to use the following quadratic approximation of the objective function

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top (w - w_t), \tag{3.49}$$

where, as in the previous section, $\nabla J(\cdot)$ denotes the gradient of J. Minimizing this quadratic model at every iteration entails taking gradients with

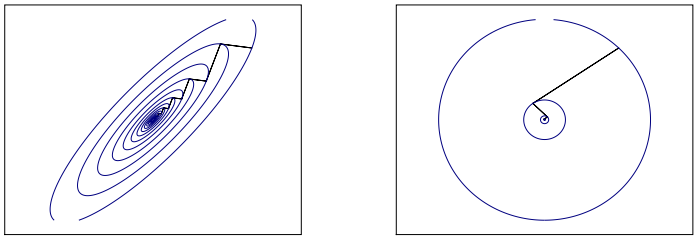


Fig. 3.8. Convergence of gradient descent with exact line search on two quadratic problems (3.48). The problem on the left is ill-conditioned, whereas the problem on the right is well-conditioned. We plot the contours of the objective function, and the steps taken by gradient descent. As can be seen gradient descent converges fast on the well conditioned problem, while it zigzags and takes many iterations to converge on the ill-conditioned problem.

respect to w and setting it to zero, which gives

$$w - w_t := -\nabla J(w_t). \tag{3.50}$$

Performing a line search along the direction $-\nabla J(w_t)$ recovers the familiar gradient descent update

$$w_{t+1} = w_t - \eta_t \nabla J(w_t). \tag{3.51}$$

The closely related mirror descent method replaces the quadratic penalty in (3.49) by a Bregman divergence defined by some convex function f to yield

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \Delta_f(w, w_t). \tag{3.52}$$

Computing the gradient, setting it to zero, and using $\nabla_w \Delta_f(w, w_t) = \nabla f(w) - \nabla f(w_t)$, the minimizer of the above model can be written as

$$\nabla f(w) - \nabla f(w_t) = -\nabla J(w_t). \tag{3.53}$$

As before, by using a stepsize η_t the resulting updates can be written as

$$w_{t+1} = \nabla f^{-1}(\nabla f(w_t) - \eta_t \nabla J(w_t)). \tag{3.54}$$

It is easy to verify that choosing $f(\cdot) = \frac{1}{2} \|\cdot\|^2$ recovers the usual gradient descent updates. On the other hand if we choose f to be the un-normalized entropy (3.30) then $\nabla f(\cdot) = \log$ and therefore (3.54) specializes to

$$w_{t+1} = \exp(\log(w_t) - \eta_t \nabla J(w_t)) = w_t \exp(-\eta_t \nabla J(w_t)), \tag{3.55}$$

which is sometimes called the Exponentiated Gradient (EG) update.

Theorem 3.14 *Let J be a convex function and $J(w^*)$ denote its minimum value. The mirror descent updates (3.54) with a σ-strongly convex function f satisfy*

$$\frac{\Delta_f(w^*, w_1) + \frac{1}{2\sigma}\sum_t \eta_t^2 \left\|\nabla J(w_t)\right\|^2}{\sum_t \eta_t} \geq \min_t J(w_t) - J(w^*).$$

Proof Using the convexity of J (see (3.7)) and (3.54) we can write

$$\begin{aligned} J(w^*) &\geq J(w_t) + \langle w^* - w_t, \nabla J(w_t)\rangle \\ &\geq J(w_t) - \frac{1}{\eta_t}\left\langle w^* - w_t, f(w_{t+1}) - f(w_t)\right\rangle. \end{aligned}$$

Now applying Lemma 3.11 and rearranging

$$\Delta_f(w^*, w_t) - \Delta_f(w^*, w_{t+1}) + \Delta_f(w_t, w_{t+1}) \geq \eta_t(J(w_t) - J(w^*)).$$

Summing over $t = 1, \ldots, T$

$$\Delta_f(w^*, w_1) - \Delta_f(w^*, w_{T+1}) + \sum_t \Delta_f(w_t, w_{t+1}) \geq \sum_t \eta_t(J(w_t) - J(w^*)).$$

Noting that $\Delta_f(w^*, w_{T+1}) \geq 0$, $J(w_t) - J(w^*) \geq \min_t J(w_t) - J(w^*)$, and rearranging it follows that

$$\frac{\Delta_f(w^*, w_1) + \sum_t \Delta_f(w_t, w_{t+1})}{\sum_t \eta_t} \geq \min_t J(w_t) - J(w^*). \tag{3.56}$$

Using (3.17) and (3.54)

$$\Delta_f(w_t, w_{t+1}) \leq \frac{1}{2\sigma}\left\|\nabla f(w_t) - \nabla f(w_{t+1})\right\|^2 = \frac{1}{2\sigma}\eta_t^2\left\|\nabla J(w_t)\right\|^2. \tag{3.57}$$

The proof is completed by plugging in (3.57) into (3.56). ■

Corollary 3.15 *If J has a Lipschitz continuous gradient with modulus L, and the stepsizes η_t are chosen as*

$$\eta_t = \frac{\sqrt{2\sigma\Delta_f(w^*, w_1)}}{L}\frac{1}{\sqrt{t}} \text{ then} \tag{3.58}$$

$$\min_{1\leq t\leq T} J(w_t) - J(w^*) \leq L\sqrt{\frac{2\Delta_f(w^*, w_1)}{\sigma}}\frac{1}{\sqrt{T}}.$$

Proof Since ∇J is Lipschitz continuous

$$\min_{1\leq t\leq T} J(w_t) - J(w^*) \leq \frac{\Delta_f(w^*, w_1) + \frac{1}{2\sigma}\sum_t \eta_t^2 L^2}{\sum_t \eta_t}.$$

Plugging in (3.58) and using Problem 3.15

$$\min_{1\le t\le T} J(w_t) - J(w^*) \le L\sqrt{\frac{\Delta_f(w^*, w_1)}{2\sigma}}\frac{(1+\sum_t \frac{1}{t})}{\sum_t \frac{1}{\sqrt{t}}} \le L\sqrt{\frac{\Delta_f(w^*, w_1)}{2\sigma}}\frac{1}{\sqrt{T}}.$$

■

3.2.5 Conjugate Gradient

Let us revisit the problem of minimizing the quadratic objective function (3.48). Since $\nabla J(w) = Aw - b$, at the optimum $\nabla J(w) = 0$ (see Lemma 3.6) and hence

$$Aw = b. \tag{3.59}$$

In fact, the Conjugate Gradient (CG) algorithm was first developed as a method to solve the above linear system.

As we already saw, updating w along the negative gradient direction may lead to zigzagging. Therefore CG uses the so-called *conjugate directions*.

Definition 3.16 (Conjugate Directions) *Non zero vectors p_t and $p_{t'}$ are said to be conjugate with respect to a symmetric positive definite matrix A if $p_{t'}^\top A p_t = 0$ if $t \neq t'$.*

Conjugate directions $\{p_0, \ldots, p_{n-1}\}$ are linearly independent and form a basis. To see this, suppose the p_t's are not linearly independent. Then there exists non-zero coefficients σ_t such that $\sum_t \sigma_t p_t = 0$. The p_t's are conjugate directions, therefore $p_{t'}^\top A(\sum_t \sigma_t p_t) = \sum_t \sigma_t p_{t'}^\top A p_t = \sigma_{t'} p_{t'}^\top A p_{t'} = 0$ for all t'. Since A is positive definite this implies that $\sigma_{t'} = 0$ for all t', a contradiction.

As it turns out, the conjugate directions can be generated iteratively as follows: Starting with any $w_0 \in \mathbb{R}^n$ define $p_0 = -g_0 = b - Aw_0$, and set

$$\alpha_t = -\frac{g_t^\top p_t}{p_t^\top A p_t} \tag{3.60a}$$

$$w_{t+1} = w_t + \alpha_t p_t \tag{3.60b}$$

$$g_{t+1} = Aw_{t+1} - b \tag{3.60c}$$

$$\beta_{t+1} = \frac{g_{t+1}^\top A p_t}{p_t^\top A p_t} \tag{3.60d}$$

$$p_{t+1} = -g_{t+1} + \beta_{t+1} p_t \tag{3.60e}$$

The following theorem asserts that the p_t generated by the above procedure are indeed conjugate directions.

Theorem 3.17 *Suppose the t-th iterate generated by the conjugate gradient method (3.60) is not the solution of (3.59), then the following properties hold:*

$$\operatorname{span}\{g_0, g_1, \ldots, g_t\} = \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}. \tag{3.61}$$

$$\operatorname{span}\{p_0, p_1, \ldots, p_t\} = \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}. \tag{3.62}$$

$$p_j^\top g_t = 0 \text{ for all } j < t \tag{3.63}$$

$$p_j^\top A p_t = 0 \text{ for all } j < t. \tag{3.64}$$

Proof The proof is by induction. The induction hypothesis holds trivially at $t = 0$. Assuming that (3.61) to (3.64) hold for some t, we prove that they continue to hold for $t + 1$.

Step 1: We first prove that (3.63) holds. Using (3.60c), (3.60b) and (3.60a)

$$\begin{aligned} p_j^\top g_{t+1} &= p_j^\top (Aw_{t+1} - b) \\ &= p_j^\top (Aw_t + \alpha_t p_t - b) \\ &= p_j^\top \left(Aw_t - \frac{g_t^\top p_t}{p_t^\top A p_t} A p_t - b \right) \\ &= p_j^\top g_t - \frac{p_j^\top A p_t}{p_t^\top A p_t} g_t^\top p_t. \end{aligned}$$

For $j = t$, both terms cancel out, while for $j < t$ both terms vanish due to the induction hypothesis.

Step 2: Next we prove that (3.61) holds. Using (3.60c) and (3.60b)

$$g_{t+1} = Aw_{t+1} - b = Aw_t + \alpha_t A p_t - b = g_t + \alpha_t A p_t.$$

By our induction hypothesis, $g_t \in \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}$, while $Ap_t \in \operatorname{span}\{Ag_0, A^2 g_0, \ldots, A^{t+1} g_0\}$. Combining the two we conclude that $g_{t+1} \in \operatorname{span}\{g_0, Ag_0, \ldots, A^{t+1} g_0\}$. On the other hand, we already showed that g_{t+1} is orthogonal to $\{p_0, p_1, \ldots, p_t\}$. Therefore, $g_{t+1} \notin \operatorname{span}\{p_0, p_1, \ldots, p_t\}$. Thus our induction assumption implies that $g_{t+1} \notin \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}$. This allows us to conclude that $\operatorname{span}\{g_0, g_1, \ldots, g_{t+1}\} = \operatorname{span}\{g_0, Ag_0, \ldots, A^{t+1} g_0\}$.

Step 3 We now prove (3.64) holds. Using (3.60e)

$$p_{t+1}^\top A p_j = -g_{t+1}^\top A p_j + \beta_{t+1} p_t^\top A p_j.$$

By the definition of β_{t+1} (3.60d) the above expression vanishes for $j = t$. For $j < t$, the first term is zero because $Ap_j \in \text{span}\{p_0, p_1, \ldots, p_{j+1}\}$, a subspace orthogonal to g_{t+1} as already shown in Step 1. The induction hypothesis guarantees that the second term is zero.

Step 4 Clearly, (3.61) and (3.60e) imply (3.62). This concludes the proof. ■

A practical implementation of (3.60) requires two more observations: First, using (3.60e) and (3.63)

$$-g_t^\top p_t = g_t^\top g_t - \beta_t g_t^\top p_{t-1} = g_t^\top g_t.$$

Therefore (3.60a) simplifies to

$$\alpha_t = \frac{g_t^\top g_t}{p_t^\top A p_t}. \tag{3.65}$$

Second, using (3.60c) and (3.60b)

$$g_{t+1} - g_t = A(w_{t+1} - w_t) = \alpha_t A p_t.$$

But $g_t \in \text{span}\{p_0, \ldots, p_t\}$, a subspace orthogonal to g_{t+1} by (3.63). Therefore $g_{t+1}^\top A p_t = \frac{1}{\alpha_t}(g_{t+1}^\top g_{t+1})$. Substituting this back into (3.60d) and using (3.65) yields

$$\beta_{t+1} = \frac{g_{t+1}^\top g_{t+1}}{g_t^\top g_t}. \tag{3.66}$$

We summarize the CG algorithm in Algorithm 3.3. Unlike gradient descent whose convergence rates for minimizing the quadratic objective function (3.48) depend upon the condition number of A, as the following theorem shows, the CG iterates converge in at most n steps.

Theorem 3.18 *The CG iterates* (3.60) *converge to the minimizer of* (3.48) *after at most* n *steps.*

Proof Let w denote the minimizer of (3.48). Since the p_t's form a basis

$$w - w_0 = \sigma_0 p_0 + \ldots + \sigma_{n-1} p_{n-1},$$

for some scalars σ_t. Our proof strategy will be to show that the coefficients

Algorithm 3.3 Conjugate Gradient

Input: Initial point w_0, residual norm tolerance ϵ
Set $t = 0$, $g_0 = Aw_0 - b$, and $p_0 = -g_0$
while $\|Aw_t - b\| \geq \epsilon$ **do**
  $\alpha_t = \frac{g_t^\top g_t}{p_t^\top A p_t}$
  $w_{t+1} = w_t + \alpha_t p_t$
  $g_{t+1} = g_t + \alpha_t A p_t$
  $\beta_{t+1} = \frac{g_{t+1}^\top g_{t+1}}{g_t^\top g_t}$
  $p_{t+1} = -g_{t+1} + \beta_{t+1} p_t$
  $t = t + 1$
end while
Return: w_t

σ_t coincide with α_t defined in (3.60a). Towards this end premultiply with $p_t^\top A$ and use conjugacy to obtain

$$\sigma_t = \frac{p_t^\top A(w - w_0)}{p_t^\top A p_t}. \tag{3.67}$$

On the other hand, following the iterative process (3.60b) from w_0 until w_t yields

$$w_t - w_0 = \alpha_0 p_0 + \ldots + \alpha_{t-1} p_{t-1}.$$

Again premultiplying with $p_t^\top A$ and using conjugacy

$$p_t^\top A(w_t - w_0) = 0. \tag{3.68}$$

Substituting (3.68) into (3.67) produces

$$\sigma_t = \frac{p_t^\top A(w - w_t)}{p_t^\top A p_t} = -\frac{g_t^\top p_t}{p_t^\top A p_t}, \tag{3.69}$$

thus showing that $\sigma_t = \alpha_t$. ■

Observe that the g_{t+1} computed via (3.60c) is nothing but the gradient of $J(w_{t+1})$. Furthermore, consider the following one dimensional optimization problem:

$$\min_{\alpha \in \mathbb{R}} \phi_t(\alpha) := J(w_t + \alpha p_t).$$

Differentiating ϕ_t with respect to α

$$\phi_t'(\alpha) = p_t^\top (Aw_t + \alpha A p_t - b) = p_t^\top (g_t + \alpha A p_t).$$

The gradient vanishes if we set $\alpha = -\frac{g_t^\top p_t}{p_t^\top A p_t}$, which recovers (3.60a). In other words, every iteration of CG minimizes $J(w)$ along a conjugate direction p_t. Contrast this with gradient descent which minimizes $J(w)$ along the negative gradient direction g_t at every iteration.

It is natural to ask if this idea of generating conjugate directions and minimizing the objective function along these directions can be applied to general convex functions. The main difficulty here is that Theorems 3.17 and 3.18 do not hold. In spite of this, extensions of CG are effective even in this setting. Basically the update rules for g_t and p_t remain the same, but the parameters α_t and β_t are computed differently. Table 3.2 gives an overview of different extensions. See [NW99, Lue84] for details.

Table 3.2. *Non-Quadratic modifications of Conjugate Gradient Descent*

Generic Method	Compute Hessian $K_t := \nabla^2 J(w_t)$ and update α_t and β_t with $\alpha_t = -\frac{g_t^\top p_t}{p_t^\top K_t p_t}$ and $\beta_t = -\frac{g_{t+1}^\top K_t p_t}{p_t^\top K_t p_t}$
Fletcher-Reeves	Set $\alpha_t = \operatorname{argmin}_\alpha J(w_t + \alpha p_t)$ and $\beta_t = \frac{g_{t+1}^\top g_{t+1}}{g_t^\top g_t}$.
Polak-Ribière	Set $\alpha_t = \operatorname{argmin}_\alpha J(w_t + \alpha p_t)$, $y_t = g_{t+1} - g_t$, and $\beta_t = \frac{y_t^\top g_{t+1}}{g_t^\top g_t}$. In practice, Polak-Ribière tends to be better than Fletcher-Reeves.
Hestenes-Stiefel	Set $\alpha_t = \operatorname{argmin}_\alpha J(w_t + \alpha p_t)$, $y_t = g_{t+1} - g_t$, and $\beta_t = \frac{y_t^\top g_{t+1}}{y_t^\top p_t}$.

3.2.6 Higher Order Methods

Recall the motivation for gradient descent as the minimizer of the quadratic model

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top (w - w_t),$$

The quadratic penalty in the above equation uniformly penalizes deviation from w_t in different dimensions. When the function is ill-conditioned one would intuitively want to penalize deviations in different directions differently. One way to achieve this is by using the Hessian, which results in the

Algorithm 3.4 Newton's Method

Input: Initial point w_0, gradient norm tolerance ϵ
Set $t = 0$
while $\|\nabla J(w_t)\| > \epsilon$ **do**
  Compute $p_t := -\nabla^2 J(w_t)^{-1} \nabla J(w_t)$
  Compute $\eta_t = \operatorname{argmin}_\eta J(w_t + \eta p_t)$ *e.g.,* via Algorithm 3.1.
  $w_{t+1} = w_t + \eta_t p_t$
  $t = t + 1$
end while
Return: w_t

following second order Taylor approximation:

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top \nabla^2 J(w_t)(w - w_t). \tag{3.70}$$

Of course, this requires that J be twice differentiable. We will also assume that J is strictly convex and hence its Hessian is positive definite and invertible. Minimizing Q_t by taking gradients with respect to w and setting it zero obtains

$$w - w_t := -\nabla^2 J(w_t)^{-1} \nabla J(w_t), \tag{3.71}$$

Since we are only minimizing a model of the objective function, we perform a line search along the descent direction (3.71) to compute the stepsize η_t, which yields the next iterate:

$$w_{t+1} = w_t - \eta_t \nabla^2 J(w_t)^{-1} \nabla J(w_t). \tag{3.72}$$

Details can be found in Algorithm 3.4.

Suppose w^* denotes the minimum of $J(w)$. We say that an algorithm exhibits quadratic convergence if the sequences of iterates $\{w_k\}$ generated by the algorithm satisfies:

$$\|w_{k+1} - w^*\| \leq C \|w_k - w_*\|^2 \tag{3.73}$$

for some constant $C > 0$. We now show that Newton's method exhibits quadratic convergence close to the optimum.

Theorem 3.19 (Quadratic convergence of Newton's Method) *Suppose J is twice differentiable, strongly convex, and the Hessian of J is bounded and Lipschitz continuous with modulus M in a neighborhood of the solution w^*. Furthermore, assume that $\left\|\nabla^2 J(w)^{-1}\right\| \leq N$. The iterations*

$w_{t+1} = w_t - \nabla^2 J(w_t)^{-1} \nabla J(w_t)$ converge quadratically to w^*, the minimizer of J.

Proof First notice that

$$\nabla J(w_t) - \nabla J(w^*) = \int_0^1 \nabla^2 J(w_t + t(w^* - w_t))(w_t - w^*) dt. \tag{3.74}$$

Next using the fact that $\nabla^2 J(w_t)$ is invertible and the gradient vanishes at the optimum ($\nabla J(w^*) = 0$), write

$$\begin{aligned} w_{t+1} - w^* &= w_t - w^* - \nabla^2 J(w_t)^{-1} \nabla J(w_t) \\ &= \nabla^2 J(w_t)^{-1} [\nabla^2 J(w_t)(w_t - w^*) - (\nabla J(w_t) - \nabla J(w^*))]. \end{aligned} \tag{3.75}$$

Using (3.75), (3.74), and the Lipschitz continuity of $\nabla^2 J$

$$\begin{aligned} &\left\| \nabla J(w_t) - \nabla J(w^*) - \nabla^2 J(w_t)(w_t - w^*) \right\| \\ &= \left\| \int_0^1 [\nabla^2 J(w_t + t(w_t - w^*)) - \nabla^2 J(w_t)](w_t - w^*) dt \right\| \\ &\leq \int_0^1 \left\| [\nabla^2 J(w_t + t(w_t - w^*)) - \nabla^2 J(w_t)] \right\| \left\| (w_t - w^*) \right\| dt \\ &\leq \left\| w_t - w^* \right\|^2 \int_0^1 M t \, dt = \frac{M}{2} \left\| w_t - w^* \right\|^2. \end{aligned} \tag{3.76}$$

Finally use (3.75) and (3.76) to conclude that

$$\|w_{t+1} - w^*\| \leq \frac{M}{2} \left\| \nabla^2 J(w_t)^{-1} \right\| \|w_t - w^*\|^2 \leq \frac{NM}{2} \|w_t - w^*\|^2.$$

■

Newton's method as we described it suffers from two major problems. First, it applies only to twice differentiable, strictly convex functions. Second, it involves computing and inverting of the $n \times n$ Hessian matrix at every iteration, thus making it computationally very expensive. Although Newton's method can be extended to deal with positive semi-definite Hessian matrices, the computational burden often makes it unsuitable for large scale applications. In such cases one resorts to Quasi-Newton methods.

3.2.6.1 Quasi-Newton Methods

Unlike Newton's method, which computes the Hessian of the objective function at every iteration, quasi-Newton methods never compute the Hessian; they approximate it from past gradients. Since they do not require the objective function to be twice differentiable, quasi-Newton methods are much

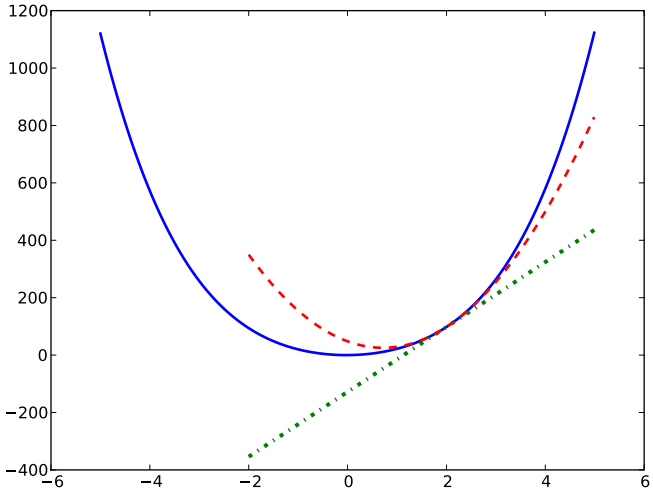


Fig. 3.9. The blue solid line depicts the one dimensional convex function $J(w) = w^4 + 20w^2 + w$. The green dotted-dashed line represents the first order Taylor approximation to $J(w)$, while the red dashed line represents the second order Taylor approximation, both evaluated at $w = 2$.

more widely applicable. They are widely regarded as the workhorses of smooth nonlinear optimization due to their combination of computational efficiency and good asymptotic convergence. The most popular quasi-Newton algorithm is BFGS, named after its discoverers Broyde, Fletcher, Goldfarb, and Shanno. In this section we will describe BFGS and its limited memory counterpart LBFGS.

Suppose we are given a smooth (not necessarily strictly) convex objective function $J : \mathbb{R}^n \to \mathbb{R}$ and a current iterate $w_t \in \mathbb{R}^n$. Just like Newton's method, BFGS forms a local quadratic model of the objective function, J:

$$Q_t(w) \;:=\; J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t (w - w_t). \tag{3.77}$$

Unlike Newton's method which uses the Hessian to build its quadratic model (3.70), BFGS uses the matrix $H_t \succ 0$, which is a positive-definite *estimate* of the Hessian. A quasi-Newton direction of descent is found by minimizing $Q_t(w)$:

$$w - w_t = -H_t^{-1} \nabla J(w_t). \tag{3.78}$$

The stepsize $\eta_t > 0$ is found by a line search obeying the Wolfe conditions

(3.42) and (3.43). The final update is given by

$$w_{t+1} = w_t - \eta_t H_t^{-1} \nabla J(w_t). \tag{3.79}$$

Given w_{t+1} we need to update our quadratic model (3.77) to

$$Q_{t+1}(w) \;:=\; J(w_{t+1}) + \langle \nabla J(w_{t+1}), w - w_{t+1} \rangle + \frac{1}{2}(w - w_{t+1})^\top H_{t+1}(w - w_{t+1}). \tag{3.80}$$

When updating our model it is reasonable to expect that the gradient of Q_{t+1} should match the gradient of J at w_t and w_{t+1}. Clearly,

$$\nabla Q_{t+1}(w) \;=\; \nabla J(w_{t+1}) + H_{t+1}(w - w_{t+1}), \tag{3.81}$$

which implies that $\nabla Q_{t+1}(w_{t+1}) = \nabla J(w_{t+1})$, and hence our second condition is automatically satisfied. In order to satisfy our first condition, we require

$$\nabla Q_{t+1}(w_t) \;=\; \nabla J(w_{t+1}) + H_{t+1}(w_t - w_{t+1}) = \nabla J(w_t). \tag{3.82}$$

By rearranging, we obtain the so-called *secant equation*:

$$H_{t+1} s_t = y_t, \tag{3.83}$$

where $s_t := w_{t+1} - w_t$ and $y_t := \nabla J(w_{t+1}) - \nabla J(w_t)$ denote the most recent step along the optimization trajectory in parameter and gradient space, respectively. Since H_{t+1} is a positive definite matrix, pre-multiplying the secant equation by s_t yields the *curvature condition*

$$s_t^\top y_t > 0. \tag{3.84}$$

If the curvature condition is satisfied, then there are an infinite number of matrices H_{t+1} which satisfy the secant equation (the secant equation represents n linear equations, but the symmetric matrix H_{t+1} has $n(n+1)/2$ degrees of freedom). To resolve this issue we choose the closest matrix to H_t which satisfies the secant equation. The key insight of the BFGS comes from the observation that the descent direction computation (3.78) involves the inverse matrix $B_t := H_t^{-1}$. Therefore, we choose a matrix $B_{t+1} := H_{t+1}^{-1}$ such that it is close to B_t and also satisfies the secant equation:

$$\min_B \|B - B_t\| \tag{3.85}$$

$$\text{s. t. } B = B^\top \text{ and } B y_t = s_t. \tag{3.86}$$

If the matrix norm $\|\cdot\|$ is appropriately chosen [NW99], then it can be shown that

$$B_{t+1} = (\mathbf{1} - \rho_t s_t y_t^\top) B_t (\mathbf{1} - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top, \tag{3.87}$$

Algorithm 3.5 LBFGS

Input: Initial point w_0, gradient norm tolerance $\epsilon > 0$
Set $t = 0$ and $B_0 = I$
while $\|\nabla J(w_t)\| > \epsilon$ **do**
  $p_t = -B_t \nabla J(w_t)$
  Find η_t that obeys (3.42) and (3.43)
  $s_t = \eta_t p_t$
  $w_{t+1} = w_t + s_t$
  $y_t := \nabla J(w_{t+1}) - \nabla J(w_t)$
  if $t = 0$: $B_t := \frac{s_t^\top y_t}{y_t^\top y_t} I$
  $\rho_t = (s_t^\top y_t)^{-1}$
  $B_{t+1} = (I - \rho_t s_t y_t^\top) B_t (I - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top$
  $t = t + 1$
end while
Return: w_t

where $\rho_t := (y_t^\top s_t)^{-1}$. In other words, the matrix B_t is modified via an incremental rank-two update, which is very efficient to compute, to obtain B_{t+1}.

There exists an interesting connection between the BFGS update (3.87) and the Hestenes-Stiefel variant of Conjugate gradient. To see this assume that an exact line search was used to compute w_{t+1}, and therefore $s_t^\top \nabla J(w_{t+1}) = 0$. Furthermore, assume that $B_t = \mathbf{1}$, and use (3.87) to write

$$p_{t+1} = -B_{t+1} \nabla J(w_{t+1}) = -\nabla J(w_{t+1}) + \frac{y_t^\top \nabla J(w_{t+1})}{y_t^\top s_t} s_t, \tag{3.88}$$

which recovers the Hestenes-Stiefel update (see (3.60e) and Table 3.2).

Limited-memory BFGS (LBFGS) is a variant of BFGS designed for solving large-scale optimization problems where the $O(d^2)$ cost of storing and updating B_t would be prohibitively expensive. LBFGS approximates the quasi-Newton direction (3.78) directly from the last m pairs of s_t and y_t via a matrix-free approach. This reduces the cost to $O(md)$ space and time per iteration, with m freely chosen. Details can be found in Algorithm 3.5.

3.2.6.2 Spectral Gradient Methods

Although spectral gradient methods do not use the Hessian explicitly, they are motivated by arguments very reminiscent of the Quasi-Newton methods. Recall the update rule (3.79) and secant equation (3.83). Suppose we want

a very simple matrix which approximates the Hessian. Specifically, we want

$$H_{t+1} = \alpha_{t+1} I \tag{3.89}$$

where α_{t+1} is a scalar and I denotes the identity matrix. Then the secant equation (3.83) becomes

$$\alpha_{t+1} s_t = y_t. \tag{3.90}$$

In general, the above equation cannot be solved. Therefore we use the α_{t+1} which minimizes $\|\alpha_{t+1} s_t - y_t\|^2$ which yields the Barzilai-Borwein (BB) stepsize

$$\alpha_{t+1} = \frac{s_t^\top y_t}{s_t^\top s_t}. \tag{3.91}$$

As it turns out, α_{t+1} lies between the minimum and maximum eigenvalue of the average Hessian in the direction s_t, hence the name Spectral Gradient method. The parameter update (3.79) is now given by

$$w_{t+1} = w_t - \frac{1}{\alpha_t} \nabla J(w_t). \tag{3.92}$$

A practical implementation uses safeguards to ensure that the stepsize α_{t+1} is neither too small nor too large. Given $0 < \alpha_{\min} < \alpha_{\max} < \infty$ we compute

$$\alpha_{t+1} = \min\left(\alpha_{\max}, \max\left(\alpha_{\min}, \frac{s_t^\top y_t}{s_t^\top s_t}\right)\right). \tag{3.93}$$

One of the peculiar features of spectral gradient methods is their use of a non-monotone line search. In all the algorithms we have seen so far, the stepsize is chosen such that the objective function J decreases at every iteration. In contrast, non-monotone line searches employ a parameter $M \geq 1$ and ensure that the objective function decreases in every M iterations. Of course, setting $M = 1$ results in the usual monotone line search. Details can be found in Algorithm 3.6.

3.2.7 Bundle Methods

The methods we discussed above are applicable for minimizing smooth, convex objective functions. Some regularized risk minimization problems involve a non-smooth objective function. In such cases, one needs to use bundle methods. In order to lay the ground for bundle methods we first describe their precursor the cutting plane method [Kel60]. Cutting plane method is based on a simple observation: A convex function is bounded from below by

Algorithm 3.6 Spectral Gradient Method

Input: w_0, $M \geq 1$, $\alpha_{\max} > \alpha_{\min} > 0$, $\gamma \in (0,1)$, $1 > \sigma_2 > \sigma_1 > 0$, $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$, and $\epsilon > 0$
Initialize: $t = 0$
while $\|\nabla J(w_t)\| > \epsilon$ **do**
  $\lambda = 1$
  while TRUE do
    $d_t = -\frac{1}{\alpha_t}\nabla J(w_t)$
    $w_+ = w_t + \lambda d_t$
    $\delta = \langle d_t, \nabla J(w_t)\rangle$
    if $J(w_+) \leq \min_{0\leq j\leq \min(t,M-1)} J(x_{t-j}) + \gamma\lambda\delta$ **then**
      $w_{t+1} = w_+$
      $s_t = w_{t+1} - w_t$
      $y_t = \nabla J(w_{t+1}) - \nabla J(w_t)$
      break
    else
      $\lambda_{\text{tmp}} = -\frac{1}{2}\lambda^2\delta/(J(w_+) - J(w_t) - \lambda\delta)$
      if $\lambda_{\text{tmp}} > \sigma_1$ and $\lambda_{\text{tmp}} < \sigma_2\lambda$ **then**
        $\lambda = \lambda_{\text{tmp}}$
      else
        $\lambda = \lambda/2$
      end if
    end if
  end while
  $\alpha_{t+1} = \min(\alpha_{\max}, \max(\alpha_{\min}, \frac{s_t^\top y_t}{s_t^\top s_t}))$
  $t = t + 1$
end while
Return: w_t

its linearization (*i.e.,* first order Taylor approximation). See Figures 3.4 and 3.5 for geometric intuition, and recall (3.7) and (3.13):

$$J(w) \geq J(w') + \left\langle w - w', s' \right\rangle \quad \forall w \text{ and } s' \in \partial J(w'). \tag{3.94}$$

Given subgradients $s_1, s_2, \ldots, s_t$ evaluated at locations $w_0, w_1, \ldots, w_{t-1}$, we can construct a tighter (piecewise linear) lower bound for J as follows (also see Figure 3.10):

$$J(w) \geq J_t^{\text{CP}}(w) := \max_{1\leq i\leq t}\{J(w_{i-1}) + \langle w - w_{i-1}, s_i\rangle\}. \tag{3.95}$$

Given iterates $\{w_i\}_{i=0}^{t-1}$, the cutting plane method minimizes J_t^{CP} to obtain the next iterate w_t:

$$w_t := \operatorname*{argmin}_w J_t^{\mathrm{CP}}(w). \tag{3.96}$$

This iteratively refines the piecewise linear lower bound J^{CP} and allows us to get close to the minimum of J (see Figure 3.10 for an illustration).

If w^* denotes the minimizer of J, then clearly each $J(w_i) \geq J(w^*)$ and hence $\min_{0 \leq i \leq t} J(w_i) \geq J(w^*)$. On the other hand, since $J \geq J_t^{\mathrm{CP}}$ it follows that $J(w^*) \geq J_t^{\mathrm{CP}}(w_t)$. In other words, $J(w^*)$ is sandwiched between $\min_{0 \leq i \leq t} J(w_i)$ and $J_t^{\mathrm{CP}}(w_t)$ (see Figure 3.11 for an illustration). The cutting plane method monitors the monotonically decreasing quantity

$$\epsilon_t := \min_{0 \leq i \leq t} J(w_i) - J_t^{\mathrm{CP}}(w_t), \tag{3.97}$$

and terminates whenever ϵ_t falls below a predefined threshold ϵ. This ensures that the solution $J(w_t)$ is ϵ optimum, that is, $J(w_t) \leq J(w^*) + \epsilon$.

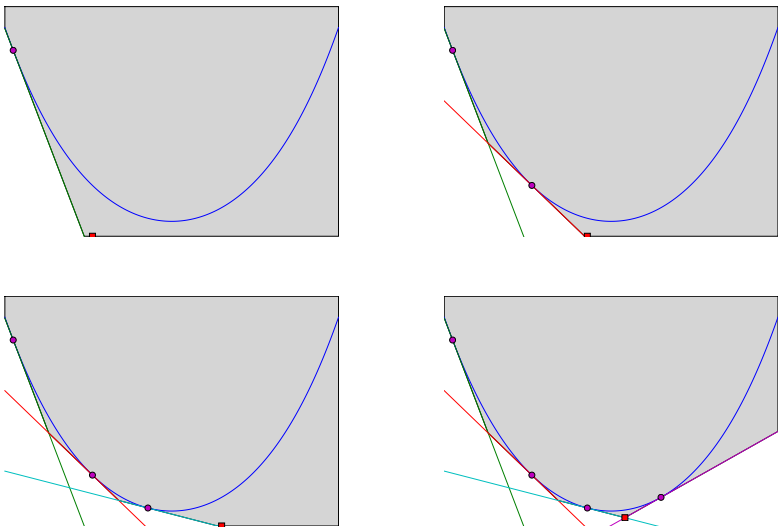


Fig. 3.10. A convex function (blue solid curve) is bounded from below by its linearizations (dashed lines). The gray area indicates the piecewise linear lower bound obtained by using the linearizations. We depict a few iterations of the cutting plane method. At each iteration the piecewise linear lower bound is minimized and a new linearization is added at the minimizer (red rectangle). As can be seen, adding more linearizations improves the lower bound.

Although cutting plane method was shown to be convergent [Kel60], it is

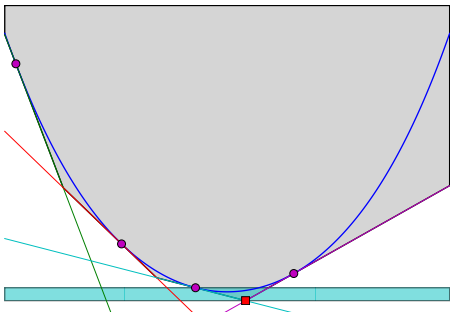


Fig. 3.11. A convex function (blue solid curve) with four linearizations evaluated at four different locations (magenta circles). The approximation gap ϵ_3 at the end of fourth iteration is indicated by the height of the cyan horizontal band *i.e.,* difference between lowest value of $J(w)$ evaluated so far and the minimum of $J_4^{\mathrm{CP}}(w)$ (red diamond).

well known (see *e.g.,* [LNN95, Bel05]) that it can be very slow when new iterates move too far away from the previous ones (*i.e.,* causing unstable "zig-zag" behavior in the iterates). In fact, in the worst case the cutting plane method might require exponentially many steps to converge to an ϵ optimum solution.

Bundle methods stabilize CPM by augmenting the piecewise linear lower (*e.g.,* $J_t^{\mathrm{CP}}(w)$ in (3.95)) with a prox-function (*i.e.,* proximity control function) which prevents overly large steps in the iterates [Kiw90]. Roughly speaking, there are 3 popular types of bundle methods, namely, *proximal* [Kiw90], *trust region* [SZ92], and *level set* [LNN95]. All three versions use $\frac{1}{2}\|\cdot\|^2$ as their prox-function, but differ in the way they compute the new iterate:

$$\text{proximal:} \quad w_t := \operatorname*{argmin}_w \{\frac{\zeta_t}{2}\|w - \hat{w}_{t-1}\|^2 + J_t^{\mathrm{CP}}(w)\}, \tag{3.98}$$

$$\text{trust region:} \quad w_t := \operatorname*{argmin}_w \{J_t^{\mathrm{CP}}(w) \mid \frac{1}{2}\|w - \hat{w}_{t-1}\|^2 \leq \kappa_t\}, \tag{3.99}$$

$$\text{level set:} \quad w_t := \operatorname*{argmin}_w \{\frac{1}{2}\|w - \hat{w}_{t-1}\|^2 \mid J_t^{\mathrm{CP}}(w) \leq \tau_t\}, \tag{3.100}$$

where $\hat{w}_{t-1}$ is the current prox-center, and ζ_t, κ_t, and τ_t are positive trade-off parameters of the stabilization. Although (3.98) can be shown to be equivalent to (3.99) for appropriately chosen ζ_t and κ_t, tuning ζ_t is rather difficult while a trust region approach can be used for automatically tuning

κ_t. Consequently the trust region algorithm BT of [SZ92] is widely used in practice.

3.3 Constrained Optimization

So far our focus was on unconstrained optimization problems. Many machine learning problems involve constraints, and can often be written in the following canonical form:

$$\min_w \quad J(w) \tag{3.101a}$$

$$\text{s. t.} \quad c_i(w) \leq 0 \text{ for } i \in \mathcal{I} \tag{3.101b}$$

$$e_i(w) = 0 \text{ for } i \in \mathcal{E} \tag{3.101c}$$

where both c_i and e_i are convex functions. We say that w is feasible if and only if it satisfies the constraints, that is, $c_i(w) \leq 0$ for $i \in \mathcal{I}$ and $e_i(w) = 0$ for $i \in \mathcal{E}$.

Recall that w is the minimizer of an unconstrained problem if and only if $\|\nabla J(w)\| = 0$ (see Lemma 3.6). Unfortunately, when constraints are present one cannot use this simple characterization of the solution. For instance, the w at which $\|\nabla J(w)\| = 0$ may not be a feasible point. To illustrate, consider the following simple minimization problem (see Figure 3.12):

$$\min_w \quad \frac{1}{2}w^2 \tag{3.102a}$$

$$\text{s. t.} \quad 1 \leq w \leq 2. \tag{3.102b}$$

Clearly, $\frac{1}{2}w^2$ is minimized at $w = 0$, but because of the presence of the constraints, the minimum of (3.102) is attained at $w = 1$ where $\nabla J(w) = w$ is equal to 1. Therefore, we need other ways to detect convergence. In Section 3.3.1 we discuss some general purpose algorithms based on the concept of orthogonal projection. In Section 3.3.2 we will discuss Lagrange duality, which can be used to further characterize the solutions of constrained optimization problems.

3.3.1 Projection Based Methods

Suppose we are interested in minimizing a smooth convex function of the following form:

$$\min_{w \in \Omega} J(w), \tag{3.103}$$

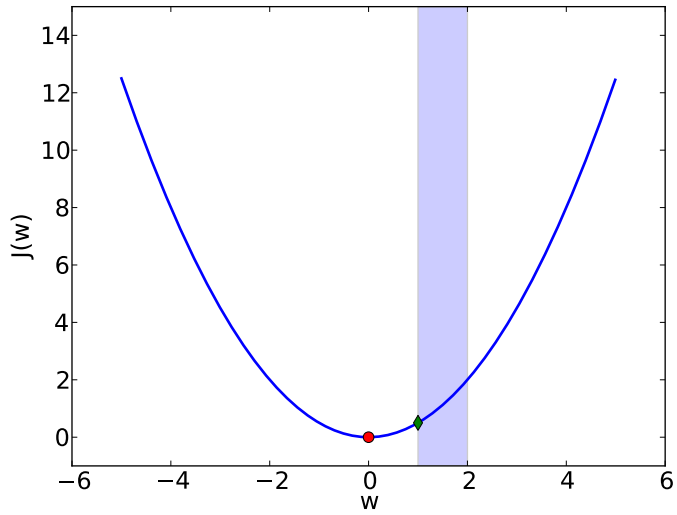


Fig. 3.12. The unconstrained minimum of the quadratic function $\frac{1}{2}w^2$ is attained at $w = 0$ (red circle). But, if we enforce the constraints $1 \leq w \leq 2$ (illustrated by the shaded area) then the minimizer is attained at $w = 1$ (green diamond).

where Ω is a convex feasible region. For instance, Ω may be described by convex functions c_i and e_i as in (3.101). The algorithms we describe in this section are applicable when Ω is a relatively simple set onto which we can compute an orthogonal projection. Given a point w' and a feasible region Ω, the orthogonal projection $P_\Omega(w')$ of w' on Ω is defined as

$$P_\Omega(w') := \operatorname*{argmin}_{w \in \Omega} \left\| w' - w \right\|^2 . \tag{3.104}$$

Geometrically speaking, $P_\Omega(w')$ is the closest point to w' in Ω. Of course, if $w' \in \Omega$ then $P_\Omega(w') = w'$.

We are interested in finding an approximate solution of (3.103), that is, a $w \in \Omega$ such that

$$J(w) - \min_{w \in \Omega} J(w) = J(w) - J^* \leq \epsilon, \tag{3.105}$$

for some pre-defined tolerance $\epsilon > 0$. Of course, J^* is unknown and hence the gap $J(w) - J^*$ cannot be computed in practice. Furthermore, as we showed in Section 3.3, for constrained optimization problems $\|\nabla J(w)\|$ does not vanish at the optimal solution. Therefore, we will use the following stopping

Algorithm 3.7 Basic Projection Based Method

1: **Input:** Initial point $w_0 \in \Omega$, and projected gradient norm tolerance $\epsilon > 0$
2: **Initialize:** $t = 0$
3: **while** $\|P_\Omega(w_t - \nabla J(w_t)) - w_t\| > \epsilon$ **do**
4: Find direction of descent d_t
5: $w_{t+1} = P_\Omega(w_t + \eta_t d_t)$
6: $t = t + 1$
7: **end while**
8: **Return:** w_t

criterion in our algorithms

$$\|P_\Omega(w_t - \nabla J(w_t)) - w_t\| \leq \epsilon. \tag{3.106}$$

The intuition here is as follows: If $w_t - \nabla J(w_t) \in \Omega$ then $P_\Omega(w_t - \nabla J(w_t)) = w_t$ if, and only if, $\nabla J(w_t) = 0$, that is, w_t is the global minimizer of $J(w)$. On the other hand, if $w_t - \nabla J(w_t) \notin \Omega$ but $P_\Omega(w_t - \nabla J(w_t)) = w_t$, then the constraints are preventing us from making any further progress along the descent direction $-\nabla J(w_t)$ and hence we should stop.

The basic projection based method is described in Algorithm 3.7. Any unconstrained optimization algorithm can be used to generate the direction of descent d_t. A line search is used to find the stepsize η_t. The updated parameter $w_t - \eta_t d_t$ is projected onto Ω to obtain w_{t+1}. If d_t is chosen to be the negative gradient direction $-\nabla J(w_t)$, then the resulting algorithm is called the projected gradient method. One can show that the rates of convergence of gradient descent with various line search schemes is also preserved by projected gradient descent.

3.3.2 Lagrange Duality

Lagrange duality plays a central role in constrained convex optimization. The basic idea here is to augment the objective function (3.101) with a weighted sum of the constraint functions by defining the Lagrangian:

$$L(w, \alpha, \beta) = J(w) + \sum_{i \in \mathcal{I}} \alpha_i c_i(w) + \sum_{i \in \mathcal{E}} \beta_i e_i(w) \tag{3.107}$$

for $\alpha_i \geq 0$ and $\beta_i \in \mathbb{R}$. In the sequel, we will refer to α (respectively β) as the Lagrange multipliers associated with the inequality (respectively equality) constraints. Furthermore, we will call α and β dual feasible if and only if

$\alpha_i \geq 0$ and $\beta_i \in \mathbb{R}$. The Lagrangian satisfies the following fundamental property, which makes it extremely useful for constrained optimization.

Theorem 3.20 *The Lagrangian* (3.107) *of* (3.101) *satisfies*

$$\max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) = \begin{cases} J(w) & \text{if } w \text{ is feasible} \\ \infty & \text{otherwise.} \end{cases}$$

In particular, if J^* *denotes the optimal value of* (3.101)*, then*

$$J^* = \min_w \max_{\alpha \geq 0, \beta} L(w, \alpha, \beta).$$

Proof First assume that w is feasible, that is, $c_i(w) \leq 0$ for $i \in \mathcal{I}$ and $e_i(w) = 0$ for $i \in \mathcal{E}$. Since $\alpha_i \geq 0$ we have

$$\sum_{i \in \mathcal{I}} \alpha_i c_i(w) + \sum_{i \in \mathcal{E}} \beta_i e_i(w) \leq 0, \tag{3.108}$$

with equality being attained by setting $\alpha_i = 0$ whenever $c_i(w) < 0$. Consequently,

$$\max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) = \max_{\alpha \geq 0, \beta} J(w) + \sum_{i \in \mathcal{I}} \alpha_i c_i(w) + \sum_{i \in \mathcal{E}} \beta_i e_i(w) = J(w)$$

whenever w is feasible. On the other hand, if w is not feasible then either $c_{i'}(w) > 0$ or $e_{i'}(w) \neq 0$ for some i'. In the first case simply let $\alpha_{i'} \to \infty$ to see that $\max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) \to \infty$. Similarly, when $e_{i'}(w) \neq 0$ let $\beta_{i'} \to \infty$ if $e_{i'}(w) > 0$ or $\beta_{i'} \to -\infty$ if $e_{i'}(w) < 0$ to arrive at the same conclusion. ■

If define the Lagrange dual function

$$D(\alpha, \beta) = \min_w L(w, \alpha, \beta), \tag{3.109}$$

for $\alpha \geq 0$ and β, then one can prove the following property, which is often called as *weak duality*.

Theorem 3.21 (Weak Duality) *The Lagrange dual function* (3.109) *satisfies*

$$D(\alpha, \beta) \leq J(w)$$

for all feasible w *and* $\alpha \geq 0$ *and* β*. In particular*

$$D^* := \max_{\alpha \geq 0, \beta} \min_w L(w, \alpha, \beta) \leq \min_w \max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) = J^*. \tag{3.110}$$

Proof As before, observe that whenever w is feasible

$$\sum_{i\in\mathcal{I}} \alpha_i c_i(w) + \sum_{i\in\mathcal{E}} \beta_i e_i(w) \leq 0.$$

Therefore

$$D(\alpha,\beta) = \min_w L(w,\alpha,\beta) = \min_w J(w) + \sum_{i\in\mathcal{I}} \alpha_i c_i(w) + \sum_{i\in\mathcal{E}} \beta_i e_i(w) \leq J(w)$$

for all feasible w and $\alpha \geq 0$ and β. In particular, one can choose w to be the minimizer of (3.101) and $\alpha \geq 0$ and β to be maximizers of $D(\alpha,\beta)$ to obtain (3.110). ■

Weak duality holds for any arbitrary function, not-necessarily convex. When the objective function and constraints are convex, and certain technical conditions, also known as Slater's conditions hold, then we can say more.

Theorem 3.22 (Strong Duality) *Supposed the objective function f and constraints c_i for $i \in \mathcal{I}$ and e_i for $i \in \mathcal{E}$ in (3.101) are convex and the following constraint qualification holds:*

There exists a w such that $c_i(w) < 0$ for all $i \in \mathcal{I}$.

Then the Lagrange dual function (3.109) satisfies

$$D^* := \max_{\alpha\geq 0,\beta} \min_w L(w,\alpha,\beta) = \min_w \max_{\alpha\geq 0,\beta} L(w,\alpha,\beta) = J^*. \tag{3.111}$$

The proof of the above theorem is quite technical and can be found in any standard reference (*e.g.,* [BV04]). Therefore we will omit the proof and proceed to discuss various implications of strong duality. First note that

$$\min_w \max_{\alpha\geq 0,\beta} L(w,\alpha,\beta) = \max_{\alpha\geq 0,\beta} \min_w L(w,\alpha,\beta). \tag{3.112}$$

In other words, one can switch the order of minimization over w with maximization over α and β. This is called the *saddle point property* of convex functions.

Suppose strong duality holds. Given any $\alpha \geq 0$ and β such that $D(\alpha,\beta) > -\infty$ and a feasible w we can immediately write the *duality gap*

$$J(w) - J^* = J(w) - D^* \leq J(w) - D(\alpha,\beta),$$

where J^* and D^* were defined in (3.111). Below we show that if w^* is primal optimal and (α^*,β^*) are dual optimal then $J(w^*) - D(\alpha^*,\beta^*) = 0$. This provides a non-heuristic stopping criterion for constrained optimization: stop when $J(w) - D(\alpha,\beta) \leq \epsilon$, where ϵ is a pre-specified tolerance.

Suppose the primal and dual optimal values are attained at w^* and (α^*, β^*) respectively, and consider the following line of argument:

$$\begin{aligned} J(w^*) &= D(\alpha^*, \beta^*) && \text{(3.113a)} \\ &= \min_w J(w) + \sum_{i \in \mathcal{I}} \alpha_i^* c_i(w) + \sum_{i \in \mathcal{E}} \beta_i^* e_j(w) && \text{(3.113b)} \\ &\leq J(w^*) + \sum_{i \in \mathcal{I}} \alpha_i^* c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* e_i(w^*) && \text{(3.113c)} \\ &\leq J(w^*). && \text{(3.113d)} \end{aligned}$$

To write (3.113a) we used strong duality, while (3.113c) obtains by setting $w = w^*$ in (3.113c). Finally, to obtain (3.113d) we used the fact that w^* is feasible and hence (3.108) holds. Since (3.113) holds with equality, one can conclude that the following *complementary slackness condition*:

$$\sum_{i \in \mathcal{I}} \alpha_i^* c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* e_i(w^*) = 0.$$

In other words, $\alpha_i^* c_i(w^*) = 0$ or equivalently $\alpha_i^* = 0$ whenever $c_i(w) < 0$. Furthermore, since w^* minimizes $L(w, \alpha^*, \beta^*)$ over w, it follows that its gradient must vanish at w^*, that is,

$$\nabla J(w^*) + \sum_{i \in \mathcal{I}} \alpha_i^* \nabla c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* \nabla e_i(w^*) = 0.$$

Putting everything together, we obtain

$$\begin{aligned} c_i(w^*) &\leq 0 \quad \forall i \in \mathcal{I} && \text{(3.114a)} \\ e_j(w^*) &= 0 \quad \forall i \in \mathcal{E} && \text{(3.114b)} \\ \alpha_i^* &\geq 0 && \text{(3.114c)} \\ \alpha_i^* c_i(w^*) &= 0 && \text{(3.114d)} \\ \nabla J(w^*) + \sum_{i \in \mathcal{I}} \alpha_i^* \nabla c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* \nabla e_i(w^*) &= 0. && \text{(3.114e)} \end{aligned}$$

The above conditions are called the KKT conditions. If the primal problem is convex, then the KKT conditions are both necessary and sufficient. In other words, if $\hat{w}$ and $(\hat{\alpha}, \hat{\beta})$ satisfy (3.114) then $\hat{w}$ and $(\hat{\alpha}, \hat{\beta})$ are primal and dual optimal with zero duality gap. To see this note that the first two conditions show that $\hat{w}$ is feasible. Since $\alpha_i \geq 0$, $L(w, \alpha, \beta)$ is convex in w. Finally the last condition states that $\hat{w}$ minimizes $L(w, \hat{\alpha}, \hat{\beta})$. Since $\hat{\alpha}_i c_i(\hat{w}) = 0$ and

$e_j(\hat{w}) = 0$, we have

$$\begin{aligned} D(\hat{\alpha}, \hat{\beta}) &= \min_w L(w, \hat{\alpha}, \hat{\beta}) \\ &= J(\hat{w}) + \sum_{i=1}^{n} \hat{\alpha}_i c_i(\hat{w}) + \sum_{j=1}^{m} \hat{\beta}_j e_j(\hat{w}) \\ &= J(\hat{w}). \end{aligned}$$

3.3.3 Linear and Quadratic Programs

So far we discussed general constrained optimization problems. Many machine learning problems have special structure which can be exploited further. We discuss the implication of duality for two such problems.

3.3.3.1 Linear Programming

An optimization problem with a linear objective function and (both equality and inequality) linear constraints is said to be a linear program (LP). A canonical linear program is of the following form:

$$\min_w \quad c^\top w \tag{3.115a}$$

$$\text{s. t. } Aw = b, w \geq 0. \tag{3.115b}$$

Here w and c are n dimensional vectors, while b is a m dimensional vector, and A is a $m \times n$ matrix with $m < n$.

Suppose we are given a LP of the form:

$$\min_w \quad c^\top w \tag{3.116a}$$

$$\text{s. t. } Aw \geq b, \tag{3.116b}$$

we can transform it into a canonical LP by introducing non-negative slack variables

$$\min_{w,\xi} \quad c^\top w \tag{3.117a}$$

$$\text{s. t. } Aw - \xi = b, \xi \geq 0. \tag{3.117b}$$

Next, we split w into its positive and negative parts w^+ and w^- respectively by setting $w_i^+ = \max(0, w_i)$ and $w_i^- = \max(0, -w_i)$. Using these new

variables we rewrite (3.117) as

$$\min_{w^+, w^-, \xi} \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^\top \begin{bmatrix} w^+ \\ w^- \\ \xi \end{bmatrix} \tag{3.118a}$$

$$\text{s. t. } \begin{bmatrix} A & -A & -I \end{bmatrix} \begin{bmatrix} w^+ \\ w^- \\ \xi \end{bmatrix} = b, \begin{bmatrix} w^+ \\ w^- \\ \xi \end{bmatrix} \geq 0, \tag{3.118b}$$

thus yielding a canonical LP (3.115) in the variables w^+, w^- and ξ.

By introducing non-negative Lagrange multipliers α and β one can write the Lagrangian of (3.115) as

$$L(w, \beta, s) = c^\top w + \beta^\top (Aw - b) - \alpha^\top w. \tag{3.119}$$

Taking gradients with respect to the primal and dual variables and setting them to zero obtains

$$A^\top \beta - \alpha = c \tag{3.120a}$$
$$Aw = b \tag{3.120b}$$
$$\alpha^\top w = 0 \tag{3.120c}$$
$$w \geq 0 \tag{3.120d}$$
$$\alpha \geq 0. \tag{3.120e}$$

Condition (3.120c) can be simplified by noting that both w and α are constrained to be non-negative, therefore $\alpha^\top w = 0$ if, and only if, $\alpha_i w_i = 0$ for $i = 1, \ldots, n$.

Using (3.120a), (3.120c), and (3.120b) we can write

$$c^\top w = (A^\top \beta - \alpha)^\top w = \beta^\top A w = \beta^\top b.$$

Substituting this into (3.115) and eliminating the primal variable w yields the following dual LP

$$\max_{\alpha, \beta} \; b^\top \beta \tag{3.121a}$$
$$\text{s.t. } A^\top \beta - \alpha = c, \alpha \geq 0. \tag{3.121b}$$

As before, we let $\beta^+ = \max(\beta, 0)$ and $\beta^- = \max(0, -\beta)$ and convert the

above LP into the following canonical LP

$$\max_{\alpha,\beta^+,\beta^-} \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix}^\top \begin{bmatrix} \beta^+ \\ \beta^- \\ \alpha \end{bmatrix} \tag{3.122a}$$

$$\text{s.t.} \begin{bmatrix} A^\top & -A^\top & -I \end{bmatrix} \begin{bmatrix} \beta^+ \\ \beta^- \\ \alpha \end{bmatrix} = c, \begin{bmatrix} \beta^+ \\ \beta^- \\ \alpha \end{bmatrix} \geq 0. \tag{3.122b}$$

It can be easily verified that the primal-dual problem is symmetric; by taking the dual of the dual we recover the primal (Problem 3.17). One important thing to note however is that the primal (3.115) involves n variables and $n + m$ constraints, while the dual (3.122) involves $2m + n$ variables and $4m + 2n$ constraints.

3.3.3.2 Quadratic Programming

An optimization problem with a convex quadratic objective function and linear constraints is said to be a convex quadratic program (QP). The canonical convex QP can be written as follows:

$$\min_w \ \frac{1}{2} w^\top G x + w^\top d \tag{3.123a}$$

$$\text{s.t. } a_i^\top w = b_i \text{ for } i \in \mathcal{E} \tag{3.123b}$$

$$a_i^\top w \leq b_i \text{ for } i \in \mathcal{I} \tag{3.123c}$$

Here $G \succeq 0$ is a $n \times n$ positive semi-definite matrix, $\mathcal{E}$ and $\mathcal{I}$ are finite set of indices, while d and a_i are n dimensional vectors, and b_i are scalars.

As a warm up let us consider the arguably simpler equality constrained quadratic programs. In this case, we can stack the a_i into a matrix A and the b_i into a vector b to write

$$\min_w \ \frac{1}{2} w^\top G w + w^\top d \tag{3.124a}$$

$$\text{s.t. } Aw = b \tag{3.124b}$$

By introducing non-negative Lagrange multipliers β the Lagrangian of the above optimization problem can be written as

$$L(w, \beta) = \frac{1}{2} w^\top G w + w^\top d + \beta(Aw - b). \tag{3.125}$$

To find the saddle point of the Lagrangian we take gradients with respect

to w and β and set them to zero. This obtains

$$\begin{aligned} Gw + d + A^\top \beta &= 0 \\ Aw &= b. \end{aligned}$$

Putting these two conditions together yields the following linear system of equations

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ \beta \end{bmatrix} = \begin{bmatrix} -d \\ b \end{bmatrix}. \tag{3.126}$$

The matrix in the above equation is called the KKT matrix, and we can use it to characterize the conditions under which (3.124) has a unique solution.

Theorem 3.23 *Let Z be a $n \times (n-m)$ matrix whose columns form a basis for the null space of A, that is, $AZ = 0$. If A has full row rank, and the reduced-Hessian matrix $Z^\top GZ$ is positive definite, then there exists a unique pair (w^*, β^*) which solves (3.126). Furthermore, w^* also minimizes (3.124).*

Proof Note that a unique (w^*, β^*) exists whenever the KKT matrix is non-singular. Suppose this is not the case, then there exist non-zero vectors a and b such that

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 0.$$

Since $Aa = 0$ this implies that a lies in the null space of A and hence there exists a u such that $a = Zu$. Therefore

$$\begin{bmatrix} Zu & 0 \end{bmatrix} \begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} Zu \\ 0 \end{bmatrix} = u^\top Z^\top GZu = 0.$$

Positive definiteness of $Z^\top GZ$ implies that $u = 0$ and hence $a = 0$. On the other hand, the full row rank of A and $A^\top b = 0$ implies that $b = 0$. In summary, both a and b are zero, a contradiction.

Let $w \neq w^*$ be any other feasible point and $\Delta w = w^* - w$. Since $Aw^* = Aw = b$ we have that $A\Delta w = 0$. Hence, there exists a non-zero u such that $\Delta w = Zu$. The objective function $J(w)$ can be written as

$$\begin{aligned} J(w) &= \frac{1}{2}(w^* - \Delta w)^\top G(w^* - \Delta w) + (w^* - \Delta w)^\top d \\ &= J(w^*) + \frac{1}{2}\Delta w^\top G \Delta w - (Gw^* + d)^\top \Delta w. \end{aligned}$$

First note that $\frac{1}{2}\Delta w^\top G \Delta w = \frac{1}{2}u^\top Z^\top GZu > 0$ by positive definiteness of the reduced Hessian. Second, since w^* solves (3.126) it follows that $(Gw^* +$

$d)^\top \Delta w = \beta^\top A \Delta w = 0$. Together these two observations imply that $J(w) > J(w^*)$. ■

If the technical conditions of the above theorem are met, then solving the equality constrained QP (3.124) is equivalent to solving the linear system (3.126). See [NW99] for a extensive discussion of algorithms that can be used for this task.

Next we turn our attention to the general QP (3.123) which also contains inequality constraints. The Lagrangian in this case can be written as

$$L(w, \beta) = \frac{1}{2} w^\top G w + w^\top d + \sum_{i \in \mathcal{I}} \alpha_i (a_i^\top w - b_i) + \sum_{i \in \mathcal{E}} \beta_i (a_i^\top w - b_i). \quad (3.127)$$

Let w^* denote the minimizer of (3.123). If we define the active set $\mathcal{A}(w^*)$ as

$$\mathcal{A}(w^*) = \left\{ i \text{ s.t. } i \in \mathcal{I} \text{ and } a_i^\top w^* = b_i \right\},$$

then the KKT conditions (3.114) for this problem can be written as

$$a_i^\top w - b_i < 0 \quad \forall i \in \mathcal{I} \setminus \mathcal{A}(w^*) \quad (3.128a)$$

$$a_i^\top w - b_i = 0 \quad \forall i \in \mathcal{E} \cup \mathcal{A}(w^*) \quad (3.128b)$$

$$\alpha_i^* \geq 0 \quad \forall i \in \mathcal{A}(w^*) \quad (3.128c)$$

$$G w^* + d + \sum_{i \in \mathcal{A}(w^*)} \alpha_i^* a_i + \sum_{i \in \mathcal{E}} \beta_i a_i = 0. \quad (3.128d)$$

Conceptually the main difficulty in solving (3.123) is in identifying the active set $\mathcal{A}(w^*)$. This is because $\alpha_i^* = 0$ for all $i \in \mathcal{I} \setminus \mathcal{A}(w^*)$. Most algorithms for solving (3.123) can be viewed as different ways to identify the active set. See [NW99] for a detailed discussion.

3.4 Stochastic Optimization

Recall that regularized risk minimization involves a data-driven optimization problem in which the objective function involves the summation of loss terms over a set of data to be modeled:

$$\min_f \; J(f) := \lambda \Omega(f) + \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i).$$

Classical optimization techniques must compute this sum in its entirety for each evaluation of the objective, respectively its gradient. As available data sets grow ever larger, such "batch" optimizers therefore become increasingly inefficient. They are also ill-suited for the incremental setting, where partial data must be modeled as it arrives.

Stochastic gradient-based methods, by contrast, work with gradient estimates obtained from small subsamples (mini-batches) of training data. This can greatly reduce computational requirements: on large, redundant data sets, simple stochastic gradient descent routinely outperforms sophisticated second-order batch methods by orders of magnitude.

The key idea here is that $J(w)$ is replaced by an instantaneous estimate J_t which is computed from a mini-batch of size k comprising of a subset of points (x_i^t, y_i^t) with $i = 1, \ldots, k$ drawn from the dataset:

$$J_t(w) = \lambda\Omega(w) + \frac{1}{k}\sum_{i=1}^{k} l(w, x_i^t, y_i^t). \tag{3.129}$$

Setting $k = 1$ obtains an algorithm which processes data points as they arrive.

3.4.1 Stochastic Gradient Descent

Perhaps the simplest stochastic optimization algorithm is Stochastic Gradient Descent (SGD). The parameter update of SGD takes the form:

$$w_{t+1} = w_t - \eta_t \nabla J_t(w_t). \tag{3.130}$$

If J_t is not differentiable, then one can choose an arbitrary subgradient from $\partial J_t(w_t)$ to compute the update. It has been shown that SGD asymptotically converges to the true minimizer of $J(w)$ if the stepsize η_t decays as $O(1/\sqrt{t})$. For instance, one could set

$$\eta_t = \sqrt{\frac{\tau}{\tau + t}}, \tag{3.131}$$

where $\tau > 0$ is a tuning parameter. See Algorithm 3.8 for details.

3.4.1.1 Practical Considerations

One simple yet effective rule of thumb to tune τ is to select a small subset of data, try various values of τ on this subset, and choose the τ that most reduces the objective function.

In some cases letting η_t to decay as $O(1/t)$ has been found to be more effective:

$$\eta_t = \frac{\tau}{\tau + t}. \tag{3.132}$$

The free parameter $\tau > 0$ can be tuned as described above. If $\Omega(w)$ is σ-strongly convex, then dividing the stepsize η_t by $\sigma\lambda$ yields good practical performance.

Algorithm 3.8 Stochastic Gradient Descent

Input: Maximum iterations T, batch size k, and τ
Set $t = 0$ and $w_0 = 0$
while $t < T$ **do**
  Choose a subset of k data points (x_i^t, y_i^t) and compute $\nabla J_t(w_t)$
  Compute stepsize $\eta_t = \sqrt{\frac{\tau}{\tau+t}}$
  $w_{t+1} = w_t - \eta_t \nabla J_t(w_t)$
  $t = t + 1$
end while
Return: w_T

3.5 Nonconvex Optimization

Our focus in the previous sections was on convex objective functions. Sometimes non-convex objective functions also arise in machine learning applications. These problems are significantly harder and tools for minimizing such objective functions are not as well developed. We briefly describe one algorithm which can be applied whenever we can write the objective function as a difference of two convex functions.

3.5.1 Concave-Convex Procedure

Any function with a bounded Hessian can be decomposed into the difference of two (non-unique) convex functions, that is, one can write

$$J(w) = f(w) - g(w), \tag{3.133}$$

where f and g are convex functions. Clearly, J is not convex, but there exists a reasonably simple algorithm namely the Concave-Convex Procedure (CCP) for finding a local minima of J. The basic idea is simple: In the t^{th} iteration replace g by its first order Taylor expansion at w_t, that is, $g(w_t) + \langle w - w_t, \nabla g(w_t) \rangle$ and minimize

$$J_t(w) = f(w) - g(w_t) - \langle w - w_t, \nabla g(w_t) \rangle . \tag{3.134}$$

Taking gradients and setting it to zero shows that J_t is minimized by setting

$$\nabla f(w_{t+1}) = \nabla g(w_t). \tag{3.135}$$

The iterations of CCP on a toy minimization problem is illustrated in Figure 3.13, while the complete algorithm listing can be found in Algorithm 3.9.

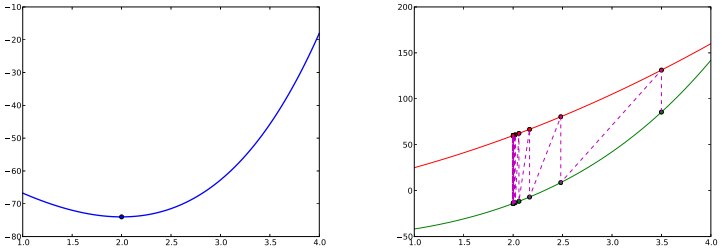


Fig. 3.13. Given the function on the left we decompose it into the difference of two convex functions depicted on the right panel. The CCP algorithm generates iterates by matching points on the two convex curves which have the same tangent vectors. As can be seen, the iterates approach the solution $x = 2.0$.

Algorithm 3.9 Concave-Convex Procedure

Input: Initial point w_0, maximum iterations T, convex functions f,g
Set $t = 0$
while $t < T$ **do**
  Set $w_{t+1} = \operatorname{argmin}_w f(w) - g(w_t) - \langle w - w_t, \nabla g(w_t) \rangle$
  $t = t + 1$
end while
Return: w_T

Theorem 3.24 *Let J be a function which can be decomposed into a difference of two convex functions* e.g., (3.133). *The iterates generated by* (3.135) *monotically decrease J. Furthermore, the stationary point of the iterates is a local minima of J.*

Proof Since f and g are convex

$$f(w_t) \geq f(w_{t+1}) + \langle w_t - w_{t+1}, \nabla f(w_{t+1}) \rangle$$
$$g(w_{t+1}) \geq g(w_t) + \langle w_{t+1} - w_t, \nabla g(w_t) \rangle .$$

Adding the two inequalities, rearranging, and using (3.135) shows that $J(w_t) = f(w_t) - g(w_t) \geq f(w_{t+1}) - g(w_{t+1}) = J(w_{t+1})$, as claimed.

Let w^* be a stationary point of the iterates. Then $\nabla f(w^*) = \nabla g(w^*)$, which in turn implies that w^* is a local minima of J because $\nabla J(w^*) = 0$. ■

There are a number of extensions to CCP. We mention only a few in the passing. First, it can be shown that all instances of the EM algorithm (Section **??**) can be shown to be special cases of CCP. Second, the rate of con-

vergence of CCP is related to the eigenvalues of the positive semi-definite matrix $\nabla^2(f+g)$. Third, CCP can also be extended to solve constrained problems of the form:

$$\begin{aligned} \min_w \quad & f_0(w) - g_0(w) \\ \text{s.t.} \quad & f_i(w) - g_i(w) \leq c_i \text{ for } i = 1, \ldots, n. \end{aligned}$$

where, as before, f_i and g_i for $i = 0, 1, \ldots, n$ are assumed convex. At every iteration, we replace g_i by its first order Taylor approximation and solve the following constrained convex problem:

$$\begin{aligned} \min_w \quad & f_0(w) - g_0(w_t) + \langle w - w_t, \nabla g_0(w_t) \rangle \\ \text{s.t.} \quad & f_i(w) - g_i(w_t) + \langle w - w_t, \nabla g_i(w_t) \rangle \leq c_i \text{ for } i = 1, \ldots, n. \end{aligned}$$

3.6 Some Practical Advice

The range of optimization algorithms we presented in this chapter might be somewhat intimidating for the beginner. Some simple rules of thumb can alleviate this anxiety

Code Reuse: Implementing an efficient optimization algorithm correctly is both time consuming and error prone. Therefore, as far as possible use existing libraries. A number of high class optimization libraries both commercial and open source exist.

Unconstrained Problems: For unconstrained minimization of a smooth convex function LBFGS (Section 3.2.6.1 is the algorithm of choice. In many practical situations the spectral gradient method (Section 3.2.6.2) is also very competitive. It also has the added advantage of being easy to implement. If the function to be minimized is non-smooth then Bundle methods (Section 3.2.7) are to be preferred. Amongst the different formulations, the Bundle Trust algorithm tends to be quite robust.

Constrained Problems: For constrained problems it is very important to understand the nature of the constraints. Simple equality ($Ax = b$) and box ($l \leq x \leq u$) constraints are easier to handle than general non-linear constraints. If the objective function is smooth, the constraint set Ω is simple, and orthogonal projections P_Ω are easy to compute, then spectral projected gradient (Section 3.3.1) is the method of choice. If the optimization problem is a QP or an LP then specialized solvers tend to be much faster than general purpose solvers.

Large Scale Problems: If your parameter vector is high dimensional then consider coordinate descent (Section 3.2.2) especially if the one dimensional line search along a coordinate can be carried out efficiently. If the objective function is made up of a summation of large number of terms, consider stochastic gradient descent (Section 3.4.1). Although both these algorithms do not guarantee a very accurate solution, practical experience shows that for large scale machine learning problems this is rarely necessary.

Duality: Sometimes problems which are hard to optimize in the primal may become simpler in the dual. For instance, if the objective function is strongly convex but non-smooth, its Fenchel conjugate is smooth with a Lipschitz continuous gradient.

Problems

Problem 3.1 (Intersection of Convex Sets {1}) *If C_1 and C_2 are convex sets, then show that $C_1 \cap C_2$ is also convex. Extend your result to show that $\bigcap_{i=1}^{n} C_i$ are convex if C_i are convex.*

Problem 3.2 (Linear Transform of Convex Sets {1}) *Given a set $C \subset \mathbb{R}^n$ and a linear transform $A \in \mathbb{R}^{m \times n}$, define $AC := \{y = Ax : x \in C\}$. If C is convex then show that AC is also convex.*

Problem 3.3 (Convex Combinations {1}) *Show that a subset of $\mathbb{R}^n$ is convex if and only if it contains all the convex combination of its elements.*

Problem 3.4 (Convex Hull {2}) *Show that the convex hull, $\operatorname{conv}(X)$ is the smallest convex set which contains X.*

Problem 3.5 (Epigraph of a Convex Function {2}) *Show that a function satisfies Definition 3.3 if, and only if, its epigraph is convex.*

Problem 3.6 *Prove the Jensen's inequality (3.6).*

Problem 3.7 (Strong convexity of the negative entropy {3}) *Show that the negative entropy (3.15) is 1-strongly convex with respect to the $\|\cdot\|_1$ norm on the simplex. Hint: First show that $\phi(t) := (t-1)\log t - 2\frac{(t-1)^2}{t+1} \geq 0$ for all $t \geq 0$. Next substitute $t = x_i/y_i$ to show that*

$$\sum_i (x_i - y_i) \log \frac{x_i}{y_i} \geq \|x - y\|_1^2 .$$

Problem 3.8 (Strongly Convex Functions {2}) *Prove 3.16, 3.17, 3.18 and 3.19.*

Problem 3.9 (Convex Functions with Lipschitz Continuous Gradient {2}) *Prove 3.22, 3.23, 3.24 and 3.25.*

Problem 3.10 (One Dimensional Projection {1}) *If $f : \mathbb{R}^d \to \mathbb{R}$ is convex, then show that for an arbitrary x and p in $\mathbb{R}^d$ the one dimensional function $\Phi(\eta) := f(x + \eta p)$ is also convex.*

Problem 3.11 (Quasi-Convex Functions {2}) *In Section 3.1 we showed that the below-sets of a convex function $X_c := \{x \mid f(x) \leq c\}$ are convex. Give a counter-example to show that the converse is not true, that is, there exist non-convex functions whose below-sets are convex. This class of functions is called Quasi-Convex.*

Problem 3.12 (Gradient of the p-norm {1}) *Show that the gradient of the p-norm (3.31) is given by (3.32).*

Problem 3.13 *Derive the Fenchel conjugate of the following functions*

$$f(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise.} \end{cases} \quad \text{where } C \text{ is a convex set}$$

$$f(x) = ax + b$$

$$f(x) = \frac{1}{2} x^\top A x \text{ where } A \text{ is a positive definite matrix}$$

$$f(x) = -\log(x)$$

$$f(x) = \exp(x)$$

$$f(x) = x \log(x)$$

Problem 3.14 (Convergence of gradient descent {2}) *Suppose J has a Lipschitz continuous gradient with modulus L. Then show that Algorithm 3.2 with an inexact line search satisfying the Wolfe conditions (3.42) and (3.43) will return a solution w_t with $\|\nabla J(w_t)\| \leq \epsilon$ in at most $O(1/\epsilon^2)$ iterations.*

Problem 3.15 *Show that*

$$\frac{1 + \sum_{t=1}^{T} \frac{1}{t}}{\sum_{t=1}^{T} \frac{1}{\sqrt{t}}} \leq \frac{1}{\sqrt{T}}$$

Problem 3.16 (Coordinate Descent for Quadratic Programming {2}) *Derive a projection based method which uses coordinate descent to generate directions of descent for solving the following box constrained QP:*

$$\min_{w \in \mathbb{R}^n} \frac{1}{2} w^\top Q w + c^\top w$$
$$s.t.\ l \leq w \leq u.$$

You may assume that Q is positive definite and l and u are scalars.

Problem 3.17 (Dual of a LP {1}) *Show that the dual of the LP (3.122) is (3.115). In other words, we recover the primal by computing the dual of the dual.*

4

Online Learning and Boosting

So far the learning algorithms we considered assumed that all the training data is available before building a model for predicting labels on unseen data points. In many modern applications data is available only in a streaming fashion, and one needs to predict labels on the fly. To describe a concrete example, consider the task of spam filtering. As emails arrive the learning algorithm needs to classify them as spam or ham. Tasks such as these are tackled via online learning. Online learning proceeds in rounds. At each round a training example is revealed to the learning algorithm, which uses its current model to predict the label. The true label is then revealed to the learner which incurs a loss and updates its model based on the feedback provided. This protocol is summarized in Algorithm 4.1. The goal of online learning is to minimize the total loss incurred. By an appropriate choice of labels and loss functions, this setting encompasses a large number of tasks such as classification, regression, and density estimation. In our spam detection example, if an email is misclassified the user can provide feedback which is used to update the spam filter, and the goal is to minimize the number of misclassified emails.

4.1 Halving Algorithm

The halving algorithm is conceptually simple, yet it illustrates many of the concepts in online learning. Suppose we have access to a set of n experts, that is, functions f_i which map from the input space $\mathcal{X}$ to the output space $\mathcal{Y} = \{\pm 1\}$. Furthermore, assume that one of the experts is consistent, that is, there exists a $j \in \{1, \ldots, n\}$ such that $f_j(x_t) = y_t$ for $t = 1, \ldots, T$. The halving algorithm maintains a set $\mathcal{C}_t$ of consistent experts at time t. Initially $\mathcal{C}_0 = \{1, \ldots, n\}$, and it is updated recursively as

$$\mathcal{C}_{t+1} = \{i \in \mathcal{C}_t \text{ s.t. } f_i(x_{t+1}) = y_{t+1}\}\,. \tag{4.1}$$

The prediction on a new data point is computed via a majority vote amongst the consistent experts: $\hat{y}_t = \text{majority}(\mathcal{C}_t)$.

Lemma 4.1 *The Halving algorithm makes at most* $\log_2(n)$ *mistakes.*

Algorithm 4.1 Protocol of Online Learning

1: **for** $t = 1, \ldots, T$ do **do**
2: Get training instance x_t
3: Predict label $\hat{y}_t$
4: Get true label y_t
5: Incur loss $l(\hat{y}_t, x_t, y_t)$
6: Update model
7: **end for**

Proof Let M denote the total number of mistakes. The halving algorithm makes a mistake at iteration t if at least half the consistent experts $\mathcal{C}_t$ predict the wrong label. This in turn implies that

$$|\mathcal{C}_{t+1}| \leq \frac{|\mathcal{C}_t|}{2} \leq \frac{|\mathcal{C}_0|}{2^M} = \frac{n}{2^M}.$$

On the other hand, since one of the experts is consistent it follows that $1 \leq |\mathcal{C}_{t+1}|$. Therefore, $2^M \leq n$. Solving for M completes the proof. ■

4.2 Weighted Majority

We now turn to the scenario where none of the experts is consistent. Therefore, the aim here is not to minimize the number mistakes but to minimize regret.

In this chapter we will consider online methods for solving the following optimization problem:

$$\min_{w \in \Omega} J(w) \text{ where } J(w) = \sum_{t=1}^{T} f_t(w). \tag{4.2}$$

Suppose we have access to a function ψ which is continuously differentiable and strongly convex with modulus of strong convexity $\sigma > 0$ (see Section 3.1.4 for definition of strong convexity), then we can define the Bregman divergence (3.29) corresponding to ψ as

$$\Delta_\psi(w, w') = \psi(w) - \psi(w') - \left\langle w - w', \nabla\psi(w') \right\rangle.$$

We can also generalize the orthogonal projection (3.104) by replacing the square Euclidean norm with the above Bregman divergence:

$$P_{\psi,\Omega}(w') = \operatorname*{argmin}_{w \in \Omega} \Delta_\psi(w, w'). \tag{4.3}$$

Algorithm 4.2 Stochastic (sub)gradient Descent

1: **Input:** Initial point x_1, maximum iterations T
2: **for** $t = 1, \ldots, T$ **do**
3: Compute $\hat{w}_{t+1} = \nabla\psi^* \left(\nabla\psi(w_t) - \eta_t g_t\right)$ with $g_t = \partial_w f_t(w_t)$
4: Set $w_{t+1} = P_{\psi,\Omega}\left(\hat{w}_{t+1}\right)$
5: **end for**
6: **Return:** w_{T+1}

Denote $w^* = P_{\psi,\Omega}(w')$. Just like the Euclidean distance is non-expansive, the Bregman projection can also be shown to be non-expansive in the following sense:

$$\Delta_\psi(w, w') \geq \Delta_\psi(w, w^*) + \Delta_\psi(w^*, w') \tag{4.4}$$

for all $w \in \Omega$. The diameter of Ω as measured by Δ_ψ is given by

$$\operatorname{diam}_\psi(\Omega) = \max_{w,w' \in \Omega} \Delta_\psi(w, w'). \tag{4.5}$$

For the rest of this chapter we will make the following standard assumptions:

- Each f_t is convex and revealed at time instance t.
- Ω is a closed convex subset of $\mathbb{R}^n$ with non-empty interior.
- The diameter $\operatorname{diam}_\psi(\Omega)$ of Ω is bounded by $F < \infty$.
- The set of optimal solutions of (4.2) denoted by Ω^* is non-empty.
- The subgradient $\partial_w f_t(w)$ can be computed for every t and $w \in \Omega$.
- The Bregman projection (4.3) can be computed for every $w' \in \mathbb{R}^n$.
- The gradient $\nabla\psi$, and its inverse $(\nabla\psi)^{-1} = \nabla\psi^*$ can be computed.

The method we employ to solve (4.2) is given in Algorithm 4.2. Before analyzing the performance of the algorithm we would like to discuss three special cases. First, Euclidean distance squared which recovers projected stochastic gradient descent, second Entropy which recovers Exponentiated gradient descent, and third the p-norms for $p > 2$ which recovers the p-norm Perceptron. BUGBUG TODO.

Our key result is Lemma 4.3 given below. It can be found in various guises in different places most notably Lemma 2.1 and 2.2 in [**?**], Theorem 4.1 and Eq. (4.21) and (4.15) in [**?**], in the proof of Theorem 1 of [**?**], as well as Lemma 3 of [**?**]. We prove a slightly general variant; we allow for projections with an arbitrary Bregman divergence and also take into account a generalized version of strong convexity of f_t. Both these modifications will allow us to deal with general settings within a unified framework.

Definition 4.2 *We say that a convex function f is strongly convex with respect to another convex function ψ with modulus λ if*

$$f(w) - f(w') - \langle w - w', \mu \rangle \geq \lambda \Delta_\psi(w, w') \text{ for all } \mu \in \partial f(w'). \tag{4.6}$$

The usual notion of strong convexity is recovered by setting $\psi(\cdot) = \frac{1}{2}\|\cdot\|^2$.

Lemma 4.3 *Let f_t be strongly convex with respect to ψ with modulus $\lambda \geq 0$ for all t. For any $w \in \Omega$ the sequences generated by Algorithm 4.2 satisfy*

$$\Delta_\psi(w, w_{t+1}) \leq \Delta_\psi(w, w_t) - \eta_t \langle g_t, w_t - w \rangle + \frac{\eta_t^2}{2\sigma} \|g_t\|^2 \tag{4.7}$$

$$\leq (1 - \eta_t \lambda) \Delta_\psi(w, w_t) - \eta_t (f_t(w_t) - f_t(w)) + \frac{\eta_t^2}{2\sigma} \|g_t\|^2 . \tag{4.8}$$

Proof We prove the result in three steps. First we upper bound $\Delta_\psi(w, w_{t+1})$ by $\Delta_\psi(w, \hat{w}_{t+1})$. This is a consequence of (4.4) and the non-negativity of the Bregman divergence which allows us to write

$$\Delta_\psi(w, w_{t+1}) \leq \Delta_\psi(w, \hat{w}_{t+1}). \tag{4.9}$$

In the next step we use Lemma 3.11 to write

$$\Delta_\psi(w, w_t) + \Delta_\psi(w_t, \hat{w}_{t+1}) - \Delta_\psi(w, \hat{w}_{t+1}) = \langle \nabla\psi(\hat{w}_{t+1}) - \nabla\psi(w_t), w - w_t \rangle .$$

Since $\nabla\psi^* = (\nabla\psi)^{-1}$, the update in step 3 of Algorithm 4.2 can equivalently be written as $\nabla\psi(\hat{w}_{t+1}) - \nabla\psi(w_t) = -\eta_t g_t$. Plugging this in the above equation and rearranging

$$\Delta_\psi(w, \hat{w}_{t+1}) = \Delta_\psi(w, w_t) - \eta_t \langle g_t, w_t - w \rangle + \Delta_\psi(w_t, \hat{w}_{t+1}). \tag{4.10}$$

Finally we upper bound $\Delta_\psi(w_t, \hat{w}_{t+1})$. For this we need two observations: First, $\langle x, y \rangle \leq \frac{1}{2\sigma}\|x\|^2 + \frac{\sigma}{2}\|y\|^2$ for all $x, y \in \mathbb{R}^n$ and $\sigma > 0$. Second, the σ strong convexity of ψ allows us to bound $\Delta_\psi(\hat{w}_{t+1}, w_t) \geq \frac{\sigma}{2}\|w_t - \hat{w}_{t+1}\|^2$. Using these two observations

$$\begin{aligned}
\Delta_\psi(w_t, \hat{w}_{t+1}) &= \psi(w_t) - \psi(\hat{w}_{t+1}) - \langle \nabla\psi(\hat{w}_{t+1}), w_t - \hat{w}_{t+1} \rangle \\
&= -(\psi(\hat{w}_{t+1}) - \psi(w_t) - \langle \nabla\psi(w_t), \hat{w}_{t+1} - w_t \rangle) + \langle \eta_t g_t, w_t - \hat{w}_{t+1} \rangle \\
&= -\Delta_\psi(\hat{w}_{t+1}, w_t) + \langle \eta_t g_t, w_t - \hat{w}_{t+1} \rangle \\
&\leq -\frac{\sigma}{2}\|w_t - \hat{w}_{t+1}\|^2 + \frac{\eta_t^2}{2\sigma}\|g_t\|^2 + \frac{\sigma}{2}\|w_t - \hat{w}_{t+1}\|^2 \\
&= \frac{\eta_t^2}{2\sigma}\|g_t\|^2 .
\end{aligned} \tag{4.11}$$

Inequality (4.7) follows by putting together (4.9), (4.10), and (4.11), while (4.8) follows by using (4.6) with $f = f_t$ and $w' = w_t$ and substituting into

(4.7). ■

Now we are ready to prove regret bounds.

Lemma 4.4 *Let $w^* \in \Omega^*$ denote the best parameter chosen in hindsight, and let $\|g_t\| \leq L$ for all t. Then the regret of Algorithm 4.2 can be bounded via*

$$\sum_{t=1}^{T} f_t(w_t) - f_t(w^*) \leq F\left(\frac{1}{\eta_T} - T\lambda\right) + \frac{L^2}{2\sigma}\sum_{t=1}^{T}\eta_t. \tag{4.12}$$

Proof Set $w = w^*$ and rearrange (4.8) to obtain

$$f_t(w_t) - f_t(w^*) \leq \frac{1}{\eta_t}\left((1-\lambda\eta_t)\Delta_\psi(w^*, w_t) - \Delta_\psi(w^*, w_{t+1})\right) + \frac{\eta_t}{2\sigma}\|g_t\|^2.$$

Summing over t

$$\sum_{t=1}^{T} f_t(w_t) - f_t(w^*) \leq \underbrace{\sum_{t=1}^{T}\frac{1}{\eta_t}\left((1-\eta_t\lambda)\Delta_\psi(w^*, w_t) - \Delta_\psi(w^*, w_{t+1})\right)}_{T_1} + \underbrace{\sum_{t=1}^{T}\frac{\eta_t}{2\sigma}\|g_t\|^2}_{T_2}.$$

Since the diameter of Ω is bounded by F and Δ_ψ is non-negative

$$\begin{aligned}
T_1 &= \left(\frac{1}{\eta_1} - \lambda\right)\Delta_\psi(w^*, w_1) - \frac{1}{\eta_T}\Delta_\psi(w^*, w_{T+1}) + \sum_{t=2}^{T}\Delta_\psi(w^*, w_t)\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \lambda\right) \\
&\leq \left(\frac{1}{\eta_1} - \lambda\right)\Delta_\psi(w^*, w_1) + \sum_{t=2}^{T}\Delta_\psi(w^*, w_t)\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \lambda\right) \\
&\leq \left(\frac{1}{\eta_1} - \lambda\right)F + \sum_{t=2}^{T}F\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \lambda\right) = F\left(\frac{1}{\eta_T} - T\lambda\right).
\end{aligned}$$

On the other hand, since the subgradients are Lipschitz continuous with constant L it follows that

$$T_2 \leq \frac{L^2}{2\sigma}\sum_{t=1}^{T}\eta_t.$$

Putting together the bounds for T_1 and T_2 yields (4.12). ■

Corollary 4.5 *If $\lambda > 0$ and we set $\eta_t = \frac{1}{\lambda t}$ then*

$$\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \leq \frac{L^2}{2\sigma\lambda}(1 + \log(T)),$$

On the other hand, when $\lambda = 0$*, if we set* $\eta_t = \frac{1}{\sqrt{t}}$ *then*

$$\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \leq \left(F + \frac{L^2}{\sigma}\right)\sqrt{T}.$$

Proof First consider $\lambda > 0$ with $\eta_t = \frac{1}{\lambda t}$. In this case $\frac{1}{\eta_T} = T\lambda$, and consequently (4.12) specializes to

$$\sum_{t=1}^{T} f_t(w_t) - f_t(w^*) \leq \frac{L^2}{2\sigma\lambda}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{L^2}{2\sigma\lambda}(1 + \log(T)).$$

When $\lambda = 0$, and we set $\eta_t = \frac{1}{\sqrt{t}}$ and use problem 4.2 to rewrite (4.12) as

$$\sum_{t=1}^{T} f_t(w_t) - f_t(w^*) \leq F\sqrt{T} + \frac{L^2}{\sigma}\sum_{t=1}^{T}\frac{1}{2\sqrt{t}} \leq F\sqrt{T} + \frac{L^2}{\sigma}\sqrt{T}.$$

■

Problems

Problem 4.1 (Generalized Cauchy-Schwartz {1}) *Show that* $\langle x, y\rangle \leq \frac{1}{2\sigma}\|x\|^2 + \frac{\sigma}{2}\|y\|^2$ *for all* $x, y \in \mathbb{R}^n$ *and* $\sigma > 0$.

Problem 4.2 (Bounding sum of a series {1}) *Show that* $\sum_{t=a}^{b}\frac{1}{2\sqrt{t}} \leq \sqrt{b - a + 1}$. ***Hint:*** *Upper bound the sum by an integral.*

5

Conditional Densities

A number of machine learning algorithms can be derived by using conditional exponential families of distribution (Section 2.3). Assume that the training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ was drawn iid from some underlying distribution. Using Bayes rule (1.15) one can write the likelihood

$$p(\theta|X, Y) \propto p(\theta)p(Y|X, \theta) = p(\theta) \prod_{i=1}^{m} p(y_i|x_i, \theta), \tag{5.1}$$

and hence the negative log-likelihood

$$-\log p(\theta|X, Y) = -\sum_{i=1}^{m} \log p(y_i|x_i, \theta) - \log p(\theta) + \text{const.} \tag{5.2}$$

Because we do not have any prior knowledge about the data, we choose a zero mean unit variance isotropic normal distribution for $p(\theta)$. This yields

$$-\log p(\theta|X, Y) = \frac{1}{2} \|\theta\|^2 - \sum_{i=1}^{m} \log p(y_i|x_i, \theta) + \text{const.} \tag{5.3}$$

Finally, if we assume a conditional exponential family model for $p(y|x, \theta)$, that is,

$$p(y|x, \theta) = \exp\left(\langle \phi(x, y), \theta \rangle - g(\theta|x)\right), \tag{5.4}$$

then

$$-\log p(\theta|X, Y) = \frac{1}{2} \|\theta\|^2 + \sum_{i=1}^{m} g(\theta|x_i) - \langle \phi(x_i, y_i), \theta \rangle + \text{const.} \tag{5.5}$$

where

$$g(\theta|x) = \log \sum_{y \in \mathcal{Y}} \exp\left(\langle \phi(x, y), \theta \rangle\right), \tag{5.6}$$

is the log-partition function. Clearly, (5.5) is a smooth convex objective function, and algorithms for unconstrained minimization from Chapter 3

can be used to obtain the maximum aposteriori (MAP) estimate for θ. Given the optimal θ, the class label at any given x can be predicted using

$$y^* = \operatorname*{argmax}_y p(y|x,\theta). \tag{5.7}$$

In this chapter we will discuss a number of these algorithms that can be derived by specializing the above setup. Our discussion unifies seemingly disparate algorithms, which are often discussed separately in literature.

5.1 Logistic Regression

We begin with the simplest case namely binary classification[1]. The key observation here is that the labels $y \in \{\pm 1\}$ and hence

$$g(\theta|x) = \log\left(\exp\left(\langle\phi(x,+1),\theta\rangle\right) + \exp\left(\langle\phi(x,-1),\theta\rangle\right)\right). \tag{5.8}$$

Define $\hat{\phi}(x) := \phi(x,+1) - \phi(x,-1)$. Plugging (5.8) into (5.4), using the definition of $\hat{\phi}$ and rearranging

$$p(y=+1|x,\theta) = \frac{1}{1+\exp\left(\left\langle -\hat{\phi}(x),\theta\right\rangle\right)} \text{ and}$$
$$p(y=-1|x,\theta) = \frac{1}{1+\exp\left(\left\langle \hat{\phi}(x),\theta\right\rangle\right)},$$

or more compactly

$$p(y|x,\theta) = \frac{1}{1+\exp\left(\left\langle -y\hat{\phi}(x),\theta\right\rangle\right)}. \tag{5.9}$$

Since $p(y|x,\theta)$ is a logistic function, hence the name logistic regression. The classification rule (5.7) in this case specializes as follows: predict $+1$ whenever $p(y=+1|x,\theta) \geq p(y=-1|x,\theta)$ otherwise predict -1. However

$$\log\frac{p(y=+1|x,\theta)}{p(y=-1|x,\theta)} = \left\langle \hat{\phi}(x),\theta\right\rangle,$$

therefore one can equivalently use $\operatorname{sign}\left(\left\langle \hat{\phi}(x),\theta\right\rangle\right)$ as our prediction function. Using (5.9) we can write the objective function of logistic regression as

$$\frac{1}{2}\|\theta\|^2 + \sum_{i=1}^{m} \log\left(1+\exp\left(\left\langle -y_i\hat{\phi}(x_i),\theta\right\rangle\right)\right)$$

[1] The name logistic *regression* is a misnomer!

To minimize the above objective function we first compute the gradient.

$$\begin{aligned}\nabla J(\theta) &= \theta + \sum_{i=1}^{m} \frac{\exp\left(\left\langle -y_i \hat{\phi}(x_i), \theta \right\rangle\right)}{1 + \exp\left(\left\langle -y_i \hat{\phi}(x_i), \theta \right\rangle\right)} (-y_i \hat{\phi}(x_i)) \\ &= \theta + \sum_{i=1}^{m} (p(y_i|x_i, \theta) - 1) y_i \hat{\phi}(x_i).\end{aligned}$$

Notice that the second term of the gradient vanishes whenever $p(y_i|x_i, \theta) = 1$. Therefore, one way to interpret logistic regression is to view it as a method to maximize $p(y_i|x_i, \theta)$ for each point (x_i, y_i) in the training set. Since the objective function of logistic regression is twice differentiable one can also compute its Hessian

$$\nabla^2 J(\theta) = I - \sum_{i=1}^{m} p(y_i|x_i, \theta)(1 - p(y_i|x_i, \theta)) \hat{\phi}(x_i) \hat{\phi}(x_i)^\top,$$

where we used $y_i^2 = 1$. The Hessian can be used in the Newton method (Section 3.2.6) to obtain the optimal parameter θ.

5.2 Regression

5.2.1 Conditionally Normal Models

fixed variance

5.2.2 Posterior Distribution

integrating out vs. Laplace approximation, efficient estimation (sparse greedy)

5.2.3 Heteroscedastic Estimation

explain that we have two parameters. not too many details (do that as an assignment).

5.3 Multiclass Classification

5.3.1 Conditionally Multinomial Models

joint feature map

5.4 What is a CRF?

- Motivation with learning a digit example
- general definition
- Gaussian process + structure = CRF

5.4.1 Linear Chain CRFs

- Graphical model
- Applications
- Optimization problem

5.4.2 Higher Order CRFs

- 2-d CRFs and their applications in vision
- Skip chain CRFs
- Hierarchical CRFs (graph transducers, sutton et. al. JMLR etc)

5.4.3 Kernelized CRFs

- From feature maps to kernels
- The clique decomposition theorem
- The representer theorem
- Optimization strategies for kernelized CRFs

5.5 Optimization Strategies

5.5.1 Getting Started

- three things needed to optimize
  - MAP estimate
  - log-partition function
  - gradient of log-partition function
- Worked out example (linear chain?)

5.5.2 Optimization Algorithms

- Optimization algorithms (LBFGS, SGD, EG (Globerson et. al))

5.5.3 Handling Higher order CRFs

- How things can be done for higher order CRFs (briefly)

5.6 Hidden Markov Models

- Definition
- Discuss that they are modeling joint distribution $p(x, y)$
- The way they predict is by marginalizing out x
- Why they are wasteful and why CRFs generally outperform them

5.7 Further Reading

What we did not talk about:

- Details of HMM optimization
- CRFs applied to predicting parse trees via matrix tree theorem (collins, koo et al)
- CRFs for graph matching problems
- CRFs with Gaussian distributions (yes they exist)

5.7.1 Optimization

issues in optimization (blows up with number of classes). structure is not there. can we do better?

Problems

Problem 5.1 *Poisson models*

Problem 5.2 *Bayes Committee Machine*

Problem 5.3 *Newton / CG approach*

6

Kernels and Function Spaces

Kernels are measures of similarity. Broadly speaking, machine learning algorithms which rely only on the dot product between instances can be "kernelized" by replacing all instances of $\langle x, x' \rangle$ by a kernel function $k(x, x')$. We saw examples of such algorithms in Sections 1.3.3 and 1.3.4 and we will see many more examples in Chapter 7. Arguably, the design of a good kernel underlies the success of machine learning in many applications. In this chapter we will lay the ground for the theoretical properties of kernels and present a number of examples. Algorithms which use these kernels can be found in later chapters.

6.1 The Basics

Let $\mathcal{X}$ denote the space of inputs and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function which satisfies

$$k(x, x') = \langle \Phi(x), \Phi(x) \rangle \tag{6.1}$$

where Φ is a feature map which maps $\mathcal{X}$ into some dot product space $\mathcal{H}$. In other words, kernels correspond to dot products in some dot product space. The main advantage of using a kernel as a similarity measure are threefold: First, if the feature space is rich enough, then simple estimators such as hyperplanes and half-spaces may be sufficient. For instance, to classify the points in Figure BUGBUG, we need a nonlinear decision boundary, but once we map the points to a 3 dimensional space a hyperplane suffices. Second, kernels allow us to construct machine learning algorithms in the dot product space $\mathcal{H}$ without explicitly computing $\Phi(x)$. Third, we need not make any assumptions about the input space $\mathcal{X}$ other than for it to be a set. As we will see later in this chapter, this allows us to compute similarity between discrete objects such as strings, trees, and graphs. In the first half of this chapter we will present some examples of kernels, and discuss some theoretical properties of kernels in the second half.

6.1.1 Examples

6.1.1.1 Linear Kernel

Linear kernels are perhaps the simplest of all kernels. We assume that $x \in \mathbb{R}^n$ and define

$$k(x, x') = \langle x, x' \rangle = \sum_i x_i x'_i.$$

If x and x' are dense then computing the kernel takes $O(n)$ time. On the other hand, for sparse vectors this can be reduced to $O(|nnz(x) \cap nnz(x')|)$, where $nnz(\cdot)$ denotes the set of non-zero indices of a vector and $|\cdot|$ denotes the size of a set. Linear kernels are a natural representation to use for vectorial data. They are also widely used in text mining where documents are represented by a vector containing the frequency of occurrence of words (Recall that we encountered this so-called bag of words representation in Chapter 1). Instead of a simple bag of words, one can also map a text to the set of pairs of words that co-occur in a sentence for a richer representation.

6.1.1.2 Polynomial Kernel

Given $x \in \mathbb{R}^n$, we can compute a feature map Φ by taking all the d-th order products (also called the monomials) of the entries of x. To illustrate with a concrete example, let us consider $x = (x_1, x_2)$ and $d = 2$, in which case $\Phi(x) = \left(x_1^2, x_2^2, x_1 x_2, x_2 x_1\right)$. Although it is tedious to compute $\Phi(x)$ and $\Phi(x')$ explicitly in order to compute $k(x, x)$, there is a shortcut as the following proposition shows.

Proposition 6.1 *Let $\Phi(x)$ (resp. $\Phi(x')$) denote the vector whose entries are all possible d-th degree ordered products of the entries of x (resp. x'). Then*

$$k(x, x') = \left\langle \Phi(x), \Phi(x') \right\rangle = \left(\langle x, x' \rangle\right)^d. \tag{6.2}$$

Proof By direct computation

$$\begin{aligned}
\left\langle \Phi(x), \Phi(x') \right\rangle &= \sum_{j_1} \dots \sum_{j_d} x_{j_1} \dots x_{j_d} \cdot x'_{j_1} \dots x'_{j_d} \\
&= \sum_{j_1} x_{j_1} \cdot x'_{j_1} \dots \sum_{j_d} x_{j_d} \cdot x'_{j_d} = \left(\sum_j x_j \cdot x'_j \right)^d \\
&= \left(\langle x, x' \rangle\right)^d
\end{aligned}$$

■

The kernel (6.2) is called the polynomial kernel. An useful extension is the inhomogeneous polynomial kernel

$$k(x, x') = \left(\langle x, x' \rangle + c\right)^d, \tag{6.3}$$

which computes all monomials up to degree d (problem 6.2).

6.1.1.3 Radial Basis Function Kernels

6.1.1.4 Convolution Kernels

The framework of convolution kernels is a general way to extend the notion of kernels to structured objects such as strings, trees, and graphs. Let $x \in \mathcal{X}$ be a discrete object which can be decomposed into P parts $x_p \in \mathcal{X}_p$ in many different ways. As a concrete example consider the string $x = abc$ which can be split into two sets of substrings of size two namely $\{a, bc\}$ and $\{ab, c\}$. We denote the set of all such decompositions as $R(x)$, and use it to build a kernel on $\mathcal{X}$ as follows:

$$\left[k_1 \star \ldots \star k_P\right](x, x') = \sum_{\bar{x} \in R(x), \bar{x}' \in R(x')} \prod_{p=1}^{P} k_p(\bar{x}_p, \bar{x}'_p). \tag{6.4}$$

Here, the sum is over all possible ways in which we can decompose x and x' into $\bar{x}_1, \ldots, \bar{x}_P$ and $\bar{x}'_1, \ldots, \bar{x}'_P$ respectively. If the cardinality of $R(x)$ is finite, then it can be shown that (6.4) results in a valid kernel. Although convolution kernels provide the abstract framework, specific instantiations of this idea lead to a rich set of kernels on discrete objects. We will now discuss some of them in detail.

6.1.1.5 String Kernels

The basic idea behind string kernels is simple: Compare the strings by means of the subsequences they contain. More the number of common subsequences, the more similar two strings are. The subsequences need not have equal weights. For instance, the weight of a subsequence may be given by the inverse frequency of its occurrence. Similarly, if the first and last characters of a subsequence are rather far apart, then its contribution to the kernel must be down-weighted.

Formally, a string x is composed of characters from a finite alphabet Σ and $|x|$ denotes its length. We say that s is a subsequence of $x = x_1 x_2 \ldots x_{|x|}$ if $s = x_{i_1} x_{i_2} \ldots x_{i_{|s|}}$ for some $1 \leq i_1 < i_2 < \ldots < i_{|s|} \leq |x|$. In particular, if $i_{i+1} = i_i + 1$ then s is a substring of x. For example, acb is not a subsequence of $adbc$ while abc is a subsequence and adc is a substring. Assume that there exists a function $\#(x, s)$ which returns the number of times a subsequence

s occurs in x and a non-negative weighting function $w(s) \geq 0$ which returns the weight associated with s. Then the basic string kernel can be written as

$$k(x, x') = \sum_s \#(x, s)\ \#(x', s)\ w(s). \tag{6.5}$$

Different string kernels are derived by specializing the above equation:

All substrings kernel: If we restrict the summation in (6.5) to substrings then [VS04] provide a suffix tree based algorithm which allows one to compute for arbitrary $w(s)$ the kernel $k(x, x')$ in $O(|x| + |x'|)$ time and memory.

k-Spectrum kernel: The k-spectrum kernel is obtained by restricting the summation in (6.5) to substrings of length k. A slightly general variant considers all substrings of length up to k. Here k is a tuning parameter which is typically set to be a small number (*e.g.,* 5). A simple trie based algorithm can be used to compute the k-spectrum kernel in $O((|x| + |x'|)k)$ time (problem 6.3).

Inexact substring kernel: Sometimes the input strings might have measurement errors and therefore it is desirable to take into account inexact matches. This is done by replacing $\#(x, s)$ in (6.5) by another function $\#(x, s, \epsilon)$ which reports the number of approximate matches of s in x. Here ϵ denotes the number of mismatches allowed, typically a small number (*e.g.,* 3). By trading off computational complexity with storage the kernel can be computed efficiently. See [LK03] for details.

Mismatch kernel: Instead of simply counting the number of occurrences of a substring if we use a weighting scheme which down-weights the contributions of longer subsequences then this yields the so-called mismatch kernel. Given an index sequence $\mathcal{I} = (i_1, \ldots, i_k)$ with $1 \leq i_1 < i_2 < \ldots < i_k \leq |x|$ we can associate the subsequence $x(\mathcal{I}) = x_{i_1} x_{i_2} \ldots x_{i_k}$ with $\mathcal{I}$. Furthermore, define $|\mathcal{I}| = i_k - i_1 + 1$. Clearly, $|\mathcal{I}| > k$ if $\mathcal{I}$ is not contiguous. Let $\lambda \leq 1$ be a decay factor. Redefine

$$\#(x, s) = \sum_{s = x(\mathcal{I})} \lambda^{|\mathcal{I}|}, \tag{6.6}$$

that is, we count all occurrences of s in x but now the weight associated with a subsequence depends on its length. To illustrate, consider the subsequence *abc* which occurs in the string *abcebc* twice, namely, *abcebc* and *abcebc*. The first occurrence is counted with weight λ^3 while the second occurrence is counted with the weight λ^6. As it turns out, this kernel can be computed by a dynamic programming algorithm (problem BUGBUG) in $O(|x| \cdot |x'|)$ time.

6.1.1.6 Graph Kernels

There are two different notions of graph kernels. First, kernels *on* graphs are used to compare nodes of a single graph. In contrast, kernels *between* graphs focus on comparing two graphs. A random walk (or its continuous time limit, diffusion) underlie both types of kernels. The basic intuition is that two nodes are similar if there are a number of paths which connect them while two graphs are similar if they share many common paths. To describe these kernels formally we need to introduce some notation.

A graph G consists of an ordered set of n vertices $V = \{v_1, v_2, \ldots, v_n\}$, and a set of directed edges $E \subset V \times V$. A vertex v_i is said to be a neighbor of another vertex v_j if they are connected by an edge, *i.e.,* if $(v_i, v_j) \in E$; this is also denoted $v_i \sim v_j$. The adjacency matrix of a graph is the $n \times n$ matrix A with $A_{ij} = 1$ if $v_i \sim v_j$, and 0 otherwise. A walk of length k on G is a sequence of indices $i_0, i_1, \ldots i_k$ such that $v_{i_{r-1}} \sim v_{i_r}$ for all $1 \leq r \leq k$.

The adjacency matrix has a normalized cousin, defined $\tilde{A} := D^{-1}A$, which has the property that each of its rows sums to one, and it can therefore serve as the transition matrix for a stochastic process. Here, D is a diagonal matrix of node degrees, *i.e.,* $D_{ii} = d_i = \sum_j A_{ij}$. A random walk on G is a process generating sequences of vertices $v_{i_1}, v_{i_2}, v_{i_3}, \ldots$ according to $\mathbb{P}(i_{k+1}|i_1, \ldots i_k) = \tilde{A}_{i_k, i_{k+1}}$. The t^{th} power of $\tilde{A}$ thus describes t-length walks, *i.e.,* $(\tilde{A}^t)_{ij}$ is the probability of a transition from vertex v_j to vertex v_i via a walk of length t (problem BUGBUG). If p_0 is an initial probability distribution over vertices, then the probability distribution p_t describing the location of our random walker at time t is $p_t = \tilde{A}^t p_0$. The j^{th} component of p_t denotes the probability of finishing a t-length walk at vertex v_j. A random walk need not continue indefinitely; to model this, we associate every node v_{i_k} in the graph with a stopping probability q_{i_k}. The overall probability of stopping after t steps is given by $q^\top p_t$.

Given two graphs $G(V, E)$ and $G'(V', E')$, their direct product $G_\times$ is a graph with vertex set

$$V_\times = \{(v_i, v'_r) : v_i \in V,\ v'_r \in V'\}, \tag{6.7}$$

and edge set

$$E_\times = \{((v_i, v'_r), (v_j, v'_s)) : (v_i, v_j) \in E \wedge (v'_r, v'_s) \in E'\}. \tag{6.8}$$

In other words, $G_\times$ is a graph over pairs of vertices from G and G', and two vertices in $G_\times$ are neighbors if and only if the corresponding vertices in G and G' are both neighbors; see Figure 6.1 for an illustration. If A and A' are the respective adjacency matrices of G and G', then the adjacency

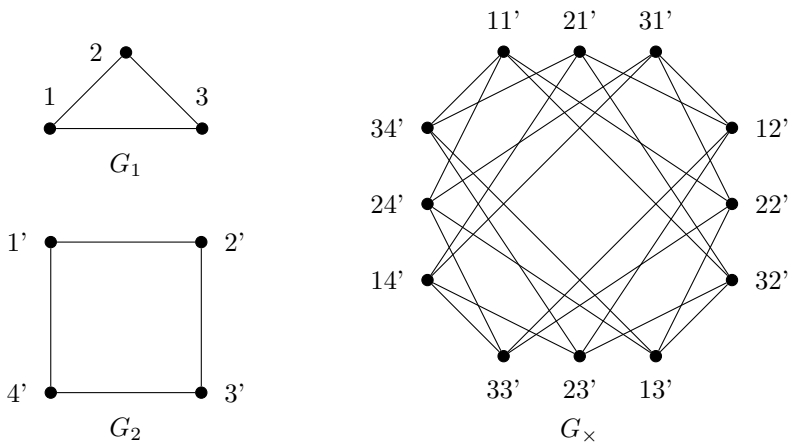


Fig. 6.1. Two graphs (G_1 & G_2) and their direct product ($G_\times$). Each node of the direct product graph is labeled with a pair of nodes (6.7); an edge exists in the direct product if and only if the corresponding nodes are adjacent in both original graphs (6.8). For instance, nodes $11'$ and $32'$ are adjacent because there is an edge between nodes 1 and 3 in the first, and $1'$ and $2'$ in the second graph.

matrix of $G_\times$ is $A_\times = A \otimes A'$. Similarly, $\tilde{A}_\times = \tilde{A} \otimes \tilde{A}'$. Performing a random walk on the direct product graph is equivalent to performing a simultaneous random walk on G and G'. If p and p' denote initial probability distributions over the vertices of G and G', then the corresponding initial probability distribution on the direct product graph is $p_\times := p \otimes p'$. Likewise, if q and q' are stopping probabilities (that is, the probability that a random walk ends at a given vertex), then the stopping probability on the direct product graph is $q_\times := q \otimes q'$.

To define a kernel which computes the similarity between G and G', one natural idea is to simply sum up $q_\times^\top \tilde{A}_\times^t p_\times$ for all values of t. However, this sum might not converge, leaving the kernel value undefined. To overcome this problem, we introduce appropriately chosen non-negative coefficients $\mu(t)$, and define the kernel between G and G' as

$$k(G, G') := \sum_{t=0}^{\infty} \mu(t)\, q_\times^\top \tilde{A}_\times^t p_\times. \tag{6.9}$$

This idea can be extended to graphs whose nodes are associated with labels by replacing the matrix $\tilde{A}_\times$ with a matrix of label similarities. For appropriate choices of $\mu(t)$ the above sum converges and efficient algorithms for computing the kernel can be devised. See [**?**] for details.

As it turns out, the simple idea of performing a random walk on the prod-

uct graph can be extended to compute kernels on Auto Regressive Moving Average (ARMA) models [VSV07]. Similarly, it can also be used to define kernels between transducers. Connections between the so-called rational kernels on transducers and the graph kernels defined via (6.9) are made explicit in [**?**].

6.2 Kernels

6.2.1 Feature Maps

give examples, linear classifier, nonlinear ones with r2-r3 map

6.2.2 The Kernel Trick

6.2.3 Examples of Kernels

gaussian, polynomial, linear, texts, graphs

- stress the fact that there is a difference between structure in the input space and structure in the output space

6.3 Algorithms

6.3.1 Kernel Perceptron

6.3.2 Trivial Classifier

6.3.3 Kernel Principal Component Analysis

6.4 Reproducing Kernel Hilbert Spaces

As it turns out, this class of functions coincides with the class of positive semi-definite functions. Intuitively, the notion of a positive semi-definite function is an extension of the familiar notion of a positive semi-definite matrix (also see Appendix BUGBUG):

Definition 6.2 *A real $n \times n$ symmetric matrix K satisfying*

$$\sum_{i,j} \alpha_i \alpha_j K_{i,j} \geq 0 \tag{6.10}$$

for all $\alpha_i, \alpha_j \in \mathbb{R}$ is called positive semi-definite. If equality in (6.10) occurs only when $\alpha_1, \ldots, \alpha_n = 0$, then K is said to be positive definite.

Definition 6.3 *Given a set of points $x_1, \ldots, x_n \in \mathcal{X}$ and a function k, the matrix*

$$K_{i,j} = k(x_i, x_j) \tag{6.11}$$

is called the Gram matrix or the kernel matrix of k with respect to $x_1, \ldots, x_n$.

Definition 6.4 *Let $\mathcal{X}$ be a nonempty set, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function. If k gives rise to a positive (semi-)definite Gram matrix for all $x_1, \ldots, x_n \in \mathcal{X}$ and $n \in \mathbb{N}$ then k is said to be positive (semi-)definite.*

Clearly, every kernel function k of the form (6.1) is positive semi-definite. To see this simply write

$$\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) = \sum_{i,j} \alpha_i \alpha_j \langle x_i, x_j \rangle = \left\langle \sum_i \alpha_i x_i, \sum_j \alpha_j x_j \right\rangle \geq 0.$$

We now establish the converse, that is, we show that every positive semi-definite kernel function can be written as (6.1). Towards this end, define a map Φ from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ to $\mathbb{R}$ (denoted $\mathbb{R}^{\mathcal{X}}$) via $\Phi(x) = k(\cdot, x)$. In other words, $\Phi(x) : \mathcal{X} \to \mathbb{R}$ is a function which assigns the value $k(x', x)$ to $x' \in \mathcal{X}$. Next construct a vector space by taking all possible linear combinations of $\Phi(x)$

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \Phi(x_i) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i), \tag{6.12}$$

where $i \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$, and $x_i \in \mathcal{X}$ are arbitrary. This space can be endowed with a natural dot product

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(x_i, x'_j). \tag{6.13}$$

To see that the above dot product is well defined even though it contains the expansion coefficients (which need not be unique), note that $\langle f, g \rangle = \sum_{j=1}^{n'} \beta_j f(x'_j)$, independent of α_i. Similarly, for g, note that $\langle f, g \rangle = \sum_{i=1}^{n} \alpha_i f(x_i)$, this time independent of β_j. This also shows that $\langle f, g \rangle$ is bilinear. Symmetry follows because $\langle f, g \rangle = \langle g, f \rangle$, while the positive semi-definiteness of k implies that

$$\langle f, f \rangle = \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \geq 0. \tag{6.14}$$

Applying (6.13) shows that for all functions (6.12) we have

$$\langle f, k(\cdot, x) \rangle = f(x). \tag{6.15}$$

In particular

$$\langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x'). \tag{6.16}$$

In view of these properties, k is called a reproducing kernel. By using (6.15) and the following property of positive semi-definite functions (problem 6.1)

$$k(x, x')^2 \leq k(x, x) \cdot k(x', x') \tag{6.17}$$

we can now write

$$|f(x)|^2 = |\langle f, k(\cdot, x)\rangle| \leq k(x, x) \cdot \langle f, f\rangle . \tag{6.18}$$

From the above inequality, $f = 0$ whenever $\langle f, f\rangle = 0$, thus establishing $\langle \cdot, \cdot \rangle$ as a valid dot product. In fact, one can complete the space of functions (6.12) in the norm corresponding to the dot product (6.13), and thus get a Hilbert space $\mathcal{H}$, called the *reproducing kernel Hilbert Space (RKHS)*.

An alternate way to define a RKHS is as a Hilbert space $\mathcal{H}$ on functions from some input space $\mathcal{X}$ to $\mathbb{R}$ with the property that for any $f \in \mathcal{H}$ and $x \in \mathcal{X}$, the point evaluations $f \to f(x)$ are continuous (in particular, all points values $f(x)$ are well defined, which already distinguishes an RKHS from many L_2 Hilbert spaces). Given the point evaluation functional, one can then construct the reproducing kernel using the Riesz representation theorem. The Moore-Aronszajn theorem states that, for every positive semi-definite kernel on $\mathcal{X} \times \mathcal{X}$, there exists a unique RKHS and vice versa.

We finish this section by noting that $\langle \cdot, \cdot \rangle$ is a positive semi-definite function in the vector space of functions (6.12). This follows directly from the bilinearity of the dot product and (6.14) by which we can write for functions $f_1, \ldots, f_p$ and coefficients $\gamma_1, \ldots, \gamma_p$

$$\sum_i \sum_j \gamma_i \gamma_j \langle f_i, f_j \rangle = \left\langle \sum_i \gamma_i f_i, \sum_j \gamma_j f_j \right\rangle \geq 0. \tag{6.19}$$

6.4.1 Hilbert Spaces

evaluation functionals, inner products

6.4.2 Theoretical Properties

Mercer's theorem, positive semidefiniteness

6.4.3 Regularization

Representer theorem, regularization

6.5 Banach Spaces

6.5.1 Properties

6.5.2 Norms and Convex Sets

- smoothest function (L2) - smallest coefficients (L1) - structured priors (CAP formalism)

Problems

Problem 6.1 *Show that* (6.17) *holds for an arbitrary positive semi-definite function* k.

Problem 6.2 *Show that the inhomogeneous polynomial kernel* (6.3) *is a valid kernel and that it computes all monomials of degree up to* d.

Problem 6.3 (k-spectrum kernel {2}) *Given two strings* x *and* x' *show how one can compute the* k*-spectrum kernel (section 6.1.1.5) in* $O((|x| + |x'|)k)$ *time.* ***Hint:*** *You need to use a trie.*

7

Linear Models

A hyperplane in a space $\mathcal{H}$ endowed with a dot product $\langle \cdot, \cdot \rangle$ is described by the set

$$\{x \in \mathcal{H} | \, \langle w, x \rangle + b = 0\} \tag{7.1}$$

where $w \in \mathcal{H}$ and $b \in \mathbb{R}$. Such a hyperplane naturally divides $\mathcal{H}$ into two half-spaces: $\{x \in \mathcal{H} | \, \langle w, x \rangle + b \geq 0\}$ and $\{x \in \mathcal{H} | \, \langle w, x \rangle + b < 0\}$, and hence can be used as the decision boundary of a binary classifier. In this chapter we will study a number of algorithms which employ such linear decision boundaries. Although such models look restrictive at first glance, when combined with kernels (Chapter 6) they yield a large class of useful algorithms.

All the algorithms we will study in this chapter maximize the margin. Given a set $\mathcal{X} = \{x_1, \ldots, x_m\}$, the margin is the distance of the closest point in $\mathcal{X}$ to the hyperplane (7.1). Elementary geometric arguments (Problem 7.1) show that the distance of a point x_i to a hyperplane is given by $| \, \langle w, x_i \rangle + b \, | / \, \|w\|$, and hence the margin is simply

$$\min_{i=1,\ldots,m} \frac{| \, \langle w, x_i \rangle + b \, |}{\|w\|}. \tag{7.2}$$

Note that the parameterization of the hyperplane (7.1) is not unique; if we multiply both w and b by the same non-zero constant, then we obtain the same hyperplane. One way to resolve this ambiguity is to set

$$\min_{i=1,\ldots m} | \, \langle w, x_i \rangle + b | = 1.$$

In this case, the margin simply becomes $1/\|w\|$. We postpone justification of margin maximization for later and jump straight ahead to the description of various algorithms.

7.1 Support Vector Classification

Consider a binary classification task, where we are given a training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ with $x_i \in \mathcal{H}$ and $y_i \in \{\pm 1\}$. Our aim is to find a linear decision boundary parameterized by (w, b) such that $\langle w, x_i \rangle + b \geq 0$

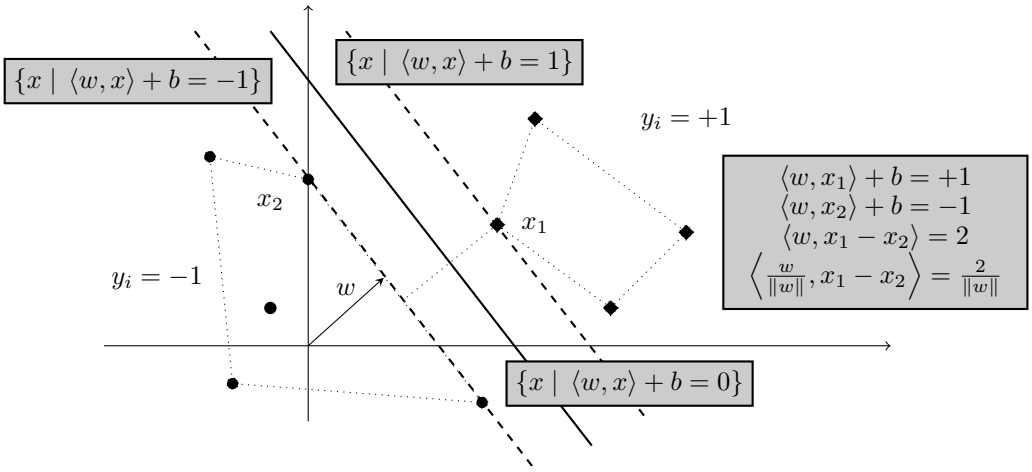


Fig. 7.1. A linearly separable toy binary classification problem of separating the diamonds from the circles. We normalize (w, b) to ensure that $\min_{i=1,\dots m} |\langle w, x_i\rangle + b| = 1$. In this case, the margin is given by $\frac{1}{\|w\|}$ as the calculation in the inset shows.

whenever $y_i = +1$ and $\langle w, x_i\rangle + b < 0$ whenever $y_i = -1$. Furthermore, as discussed above, we fix the scaling of w by requiring $\min_{i=1,\dots m} |\langle w, x_i\rangle + b| = 1$. A compact way to write our desiderata is to require $y_i(\langle w, x_i\rangle + b) \geq 1$ for all i (also see Figure 7.1). The problem of maximizing the margin therefore reduces to

$$\max_{w,b} \quad \frac{1}{\|w\|} \tag{7.3a}$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1 \text{ for all } i, \tag{7.3b}$$

or equivalently

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{7.4a}$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1 \text{ for all } i. \tag{7.4b}$$

This is a constrained convex optimization problem with a quadratic objective function and linear constraints (see Section 3.3). In deriving (7.4) we implicitly assumed that the data is linearly separable, that is, there is a hyperplane which correctly classifies the training data. Such a classifier is called a *hard margin classifier*. If the data is not linearly separable, then (7.4) does not have a solution. To deal with this situation we introduce

non-negative slack variables ξ_i to relax the constraints:

$$y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i.$$

Given any w and b the constraints can now be satisfied by making ξ_i large enough. This renders the whole optimization problem useless. Therefore, one has to penalize large ξ_i. This is done via the following modified optimization problem:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \tag{7.5a}$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \text{ for all } i \tag{7.5b}$$

$$\xi_i \geq 0, \tag{7.5c}$$

where $C > 0$ is a penalty parameter. The resultant classifier is said to be a *soft margin classifier*. By introducing non-negative Lagrange multipliers α_i and β_i one can write the Lagrangian (see Section 3.3)

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - \xi_i - y_i(\langle w, x_i\rangle + b)) - \sum_{i=1}^{m}\beta_i\xi_i.$$

Next take gradients with respect to w, b and ξ and set them to zero.

$$\nabla_w L = w - \sum_{i=1}^{m}\alpha_i y_i x_i = 0 \tag{7.6a}$$

$$\nabla_b L = -\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{7.6b}$$

$$\nabla_{\xi_i} L = \frac{C}{m} - \alpha_i - \beta_i = 0. \tag{7.6c}$$

Substituting (7.6) into the Lagrangian and simplifying yields the dual objective function:

$$-\frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle + \sum_{i=1}^{m}\alpha_i, \tag{7.7}$$

which needs to be maximized with respect to α. For notational convenience we will minimize the negative of (7.7) below. Next we turn our attention to the dual constraints. Recall that $\alpha_i \geq 0$ and $\beta_i \geq 0$, which in conjunction with (7.6c) immediately yields $0 \leq \alpha_i \leq \frac{C}{m}$. Furthermore, by (7.6b) $\sum_{i=1}^{m}\alpha_i y_i = 0$. Putting everything together, the dual optimization problem

boils down to

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i \tag{7.8a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{7.8b}$$

$$0 \leq \alpha_i \leq \frac{C}{m}. \tag{7.8c}$$

If we let H be a $m \times m$ matrix with entries $H_{ij} = y_i y_j \langle x_i, x_j \rangle$, while e, α, and y be m-dimensional vectors whose i-th components are one, α_i, and y_i respectively, then the above dual can be compactly written as the following Quadratic Program (QP) (Section 3.3.3):

$$\min_{\alpha} \quad \frac{1}{2}\alpha^\top H \alpha - \alpha^\top e \tag{7.9a}$$

$$\text{s.t.} \quad \alpha^\top y = 0 \tag{7.9b}$$

$$0 \leq \alpha_i \leq \frac{C}{m}. \tag{7.9c}$$

Before turning our attention to algorithms for solving (7.9), a number of observations are in order. First, note that computing H only requires computing dot products between training examples. If we map the input data to a Reproducing Kernel Hilbert Space (RKHS) via a feature map ϕ, then we can still compute the entries of H and solve for the optimal α. In this case, $H_{ij} = y_i y_j \langle \phi(x_i), \phi(x_j) \rangle = y_i y_j k(x_i, x_j)$, where k is the kernel associated with the RKHS. Given the optimal α, one can easily recover the decision boundary. This is a direct consequence of (7.6a), which allows us to write w as a linear combination of the training data:

$$w = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i),$$

and hence the decision boundary as

$$\langle w, x \rangle + b = \sum_{i=1}^{m} \alpha_i y_i k(x_i, x) + b. \tag{7.10}$$

By the KKT conditions (Section 3.3) we have

$$\alpha_i(1 - \xi_i - y_i(\langle w, x_i \rangle + b)) = 0 \text{ and } \beta_i \xi_i = 0.$$

We now consider three cases for $y_i(\langle w, x_i \rangle + b)$ and the implications of the KKT conditions (see Figure 7.2).

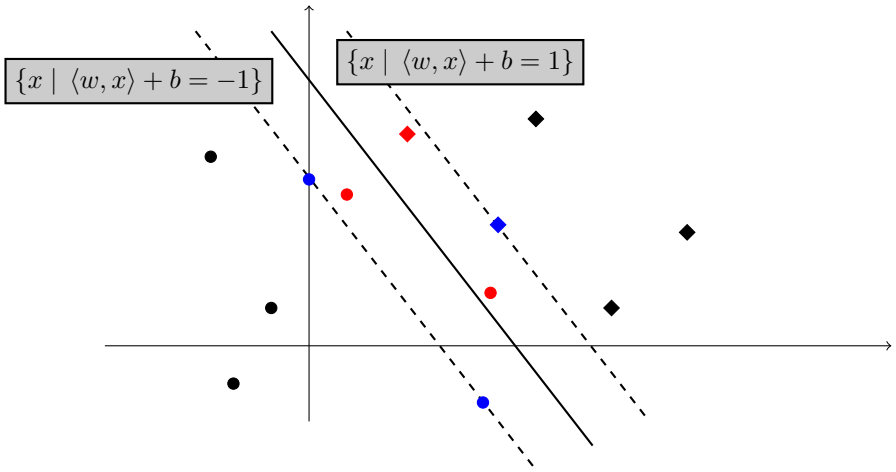


Fig. 7.2. The picture depicts the well classified points ($y_i(\langle w, x_i\rangle + b) > 1$ in black, the support vectors $y_i(\langle w, x_i\rangle + b) = 1$ in blue, and margin errors $y_i(\langle w, x_i\rangle + b) < 1$ in red.

$\mathbf{y_i(\langle w, x_i\rangle + b) < 1}$**:** In this case, $\xi_i > 0$, and hence the KKT conditions imply that $\beta_i = 0$. Consequently, $\alpha_i = \frac{C}{m}$ (see (7.6c)). Such points are said to be margin errors.

$\mathbf{y_i(\langle w, x_i\rangle + b) > 1}$**:** In this case, $\xi_i = 0$, $(1 - \xi_i - y_i(\langle w, x_i\rangle + b)) < 0$, and by the KKT conditions $\alpha_i = 0$. Such points are said to be well classified. It is easy to see that the decision boundary (7.10) does not change even if these points are removed from the training set.

$\mathbf{y_i(\langle w, x_i\rangle + b) = 1}$**:** In this case $\xi_i = 0$ and $\beta_i \geq 0$. Since α_i is non-negative and satisfies (7.6c) it follows that $0 \leq \alpha_i \leq \frac{C}{m}$. Such points are said to be on the margin. They are also sometimes called *support vectors*.

Since the support vectors satisfy $y_i(\langle w, x_i\rangle + b) = 1$ and $y_i \in \{\pm 1\}$ it follows that $b = y_i - \langle w, x_i\rangle$ for any support vector x_i. However, in practice to recover b we average

$$b = y_i - \sum_i \langle w, x_i\rangle \,. \tag{7.11}$$

over all support vectors, that is, points x_i for which $0 < \alpha_i < \frac{C}{m}$. Because it uses support vectors, the overall algorithm is called C-Support Vector classifier or C-SV classifier for short.

7.1.1 A Regularized Risk Minimization Viewpoint

A closer examination of (7.5) reveals that $\xi_i = 0$ whenever $y_i(\langle w, x_i\rangle + b) > 1$. On the other hand, $\xi_i = 1 - y_i(\langle w, x_i\rangle + b)$ whenever $y_i(\langle w, x_i\rangle + b) < 1$. In short, $\xi_i = \max(0, 1 - y_i(\langle w, x_i\rangle + b))$. Using this observation one can eliminate ξ_i from (7.5), and write it as the following unconstrained optimization problem:

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m} \max(0, 1 - y_i(\langle w, x_i\rangle + b)). \tag{7.12}$$

Writing (7.5) as (7.12) is particularly revealing because it shows that a support vector classifier is nothing but a regularized risk minimizer. Here the regularizer is the square norm of the decision hyperplane $\frac{1}{2}\|w\|^2$, and the loss function is the so-called binary hinge loss (Figure 7.3):

$$l(w, x, y) = \max(0, 1 - y(\langle w, x\rangle + b)). \tag{7.13}$$

It is easy to verify that the binary hinge loss (7.13) is convex but non-differentiable (see Figure 7.3) which renders the overall objective function (7.12) to be convex but non-smooth. There are two different strategies to minimize such an objective function. If minimizing (7.12) in the primal, one can employ non-smooth convex optimizers such as bundle methods (Section 3.2.7). This yields a d dimensional problem where d is the dimension of x. On the other hand, since (7.12) is strongly convex because of the presence of the $\frac{1}{2}\|w\|^2$ term, its Fenchel dual has a Lipschitz continuous gradient (see Lemma 3.10). The dual problem is m dimensional and contains linear constraints. This strategy is particularly attractive when the kernel trick is used or whenever $d \gg m$. In fact, the dual problem obtained via Fenchel duality is very related to the Quadratic programming problem (7.9) obtained via Lagrange duality (problem 7.4).

7.1.2 An Exponential Family Interpretation

Our motivating arguments for deriving the SVM algorithm have largely been geometric. We now show that an equally elegant probabilistic interpretation also exists. Assuming that the training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ was drawn iid from some underlying distribution, and using the Bayes rule (1.15) one can write the likelihood

$$p(\theta|X, Y) \propto p(\theta)p(Y|X, \theta) = p(\theta)\prod_{i=1}^{m} p(y_i|x_i, \theta), \tag{7.14}$$

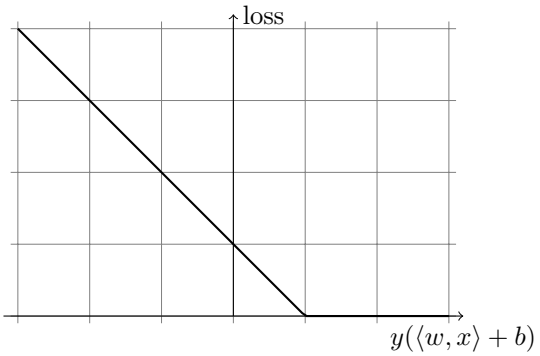


Fig. 7.3. The binary hinge loss. Note that the loss is convex but non-differentiable at the kink point. Furthermore, it increases linearly as the distance from the decision hyperplane $y(\langle w, x\rangle + b)$ decreases.

and hence the negative log-likelihood

$$-\log p(\theta|X, Y) = -\sum_{i=1}^{m} \log p(y_i|x_i, \theta) - \log p(\theta) + \text{const.} \tag{7.15}$$

In the absence of any prior knowledge about the data, we choose a zero mean unit variance isotropic normal distribution for $p(\theta)$. This yields

$$-\log p(\theta|X, Y) = \frac{1}{2}\|\theta\|^2 - \sum_{i=1}^{m} \log p(y_i|x_i, \theta) + \text{const.} \tag{7.16}$$

The maximum aposteriori (MAP) estimate for θ is obtained by minimizing (7.16) with respect to θ. Given the optimal θ, we can predict the class label at any given x via

$$y^* = \operatorname*{argmax}_y p(y|x, \theta). \tag{7.17}$$

Of course, our aim is not just to maximize $p(y_i|x_i, \theta)$ but also to ensure that $p(y|x_i, \theta)$ is small for all $y \neq y_i$. This, for instance, can be achieved by requiring

$$\frac{p(y_i|x_i, \theta)}{p(y|x_i, \theta)} \geq \eta, \text{ for all } y \neq y_i \text{ and some } \eta \geq 1. \tag{7.18}$$

As we saw in Section 2.3 exponential families of distributions are rather flexible modeling tools. We could, for instance, model $p(y_i|x_i, \theta)$ as a conditional exponential family distribution. Recall the definition:

$$p(y|x, \theta) = \exp\left(\langle \phi(x, y), \theta\rangle - g(\theta|x)\right). \tag{7.19}$$

Here $\phi(x, y)$ is a *joint* feature map which depends on both the input data x and the label y, while $g(\theta|x)$ is the log-partition function. Now (7.18) boils down to

$$\frac{p(y_i|x_i,\theta)}{\max_{y\neq y_i} p(y|x_i,\theta)} = \exp\left(\left\langle \phi(x_i,y_i) - \max_{y\neq y_i} \phi(x_i,y), \theta \right\rangle\right) \geq \eta. \tag{7.20}$$

If we choose η such that $\log \eta = 1$, set $\phi(x, y) = \frac{y}{2}\phi(x)$, and observe that $y \in \{\pm 1\}$ we can rewrite (7.20) as

$$\left\langle \frac{y_i}{2}\phi(x_i) - \left(-\frac{y_i}{2}\right)\phi(x_i), \theta \right\rangle = y_i \langle \phi(x_i), \theta \rangle \geq 1. \tag{7.21}$$

By replacing $-\log p(y_i|x_i, \theta)$ in (7.16) with the condition (7.21) we obtain the following objective function:

$$\min_{\theta} \quad \frac{1}{2}\|\theta\|^2 \tag{7.22a}$$

$$\text{s.t.} \quad y_i \langle \phi(x_i), \theta \rangle \geq 1 \text{ for all } i, \tag{7.22b}$$

which recovers (7.4), but without the bias b. The prediction function is recovered by noting that (7.17) specializes to

$$y^* = \operatorname*{argmax}_{y\in\{\pm 1\}} \langle \phi(x,y), \theta \rangle = \operatorname*{argmax}_{y\in\{\pm 1\}} \frac{y}{2} \langle \phi(x), \theta \rangle = \operatorname{sign}(\langle \phi(x), \theta \rangle). \tag{7.23}$$

As before, we can replace (7.21) by a linear penalty for constraint violation in order to recover (7.5). The quantity $\log \frac{p(y_i|x_i,\theta)}{\max_{y\neq y_i} p(y|x_i,\theta)}$ is sometimes called the *log-odds ratio*, and the above discussion shows that SVMs can be interpreted as maximizing the log-odds ratio in the exponential family. This interpretation will be developed further when we consider extensions of SVMs to tackle multiclass, multilabel, and structured prediction problems.

7.1.3 Specialized Algorithms for Training SVMs

The main task in training SVMs boils down to solving (7.9). The $m \times m$ matrix H is usually dense and cannot be stored in memory. Decomposition methods are designed to overcome these difficulties. The basic idea here is to identify and update a small *working set* B by solving a small subproblem at every iteration. Formally, let $B \subset \{1, \ldots, m\}$ be the working set and α_B be the corresponding sub-vector of α. Define $\bar{B} = \{1, \ldots, m\} \setminus B$ and $\alpha_{\bar{B}}$ analogously. In order to update α_B we need to solve the following

sub-problem of (7.9) obtained by freezing $\alpha_{\bar{B}}$:

$$\min_{\alpha_B} \ \frac{1}{2}\begin{bmatrix} \alpha_B^\top & \alpha_{\bar{B}}^\top \end{bmatrix}\begin{bmatrix} H_{BB} & H_{B\bar{B}} \\ H_{\bar{B}B} & H_{\bar{B}\bar{B}} \end{bmatrix}\begin{bmatrix} \alpha_B \\ \alpha_{\bar{B}} \end{bmatrix} - \begin{bmatrix} \alpha_B^\top & \alpha_{\bar{B}}^\top \end{bmatrix} e \tag{7.24a}$$

$$\text{s.t.} \ \begin{bmatrix} \alpha_B^\top & \alpha_{\bar{B}}^\top \end{bmatrix} y = 0 \tag{7.24b}$$

$$0 \le \alpha_i \le \frac{C}{m} \text{ for all } i \in B. \tag{7.24c}$$

Here, $\begin{bmatrix} H_{BB} & H_{B\bar{B}} \\ H_{\bar{B}B} & H_{\bar{B}\bar{B}} \end{bmatrix}$ is a permutation of the matrix H. By eliminating constant terms and rearranging, one can simplify the above problem to

$$\min_{\alpha_B} \ \frac{1}{2}\alpha_B^\top H_{BB}\alpha_B + \alpha_B^\top(H_{\bar{B}B}\alpha_{\bar{B}} - e) \tag{7.25a}$$

$$\text{s.t.} \ \alpha_B^\top y_B = -\alpha_{\bar{B}}^\top y_{\bar{B}} \tag{7.25b}$$

$$0 \le \alpha_i \le \frac{C}{m} \text{ for all } i \in B. \tag{7.25c}$$

An extreme case of a decomposition method is the Sequential Minimal Optimization (SMO) algorithm of Platt [Pla99], which updates only two coefficients per iteration. The advantage of this strategy as we will see below is that the resultant sub-problem can be solved analytically. Without loss of generality let $B = \{i, j\}$, and define $s = y_i/y_j$, $\begin{bmatrix} c_i & c_j \end{bmatrix} = (H_{\bar{B}B}\alpha_{\bar{B}} - e)^\top$ and $d = (-\alpha_{\bar{B}}^\top y_{\bar{B}}/y_j)$. Then (7.25) specializes to

$$\min_{\alpha_i,\alpha_j} \ \frac{1}{2}(H_{ii}\alpha_i^2 + H_{jj}\alpha_j^2 + 2H_{ij}\alpha_j\alpha_i) + c_i\alpha_i + c_j\alpha_j \tag{7.26a}$$

$$\text{s.t.} \ s\alpha_i + \alpha_j = d \tag{7.26b}$$

$$0 \le \alpha_i, \alpha_j \le \frac{C}{m}. \tag{7.26c}$$

This QP in two variables has an analytic solution.

Lemma 7.1 (Analytic solution of 2 variable QP) *Define bounds*

$$L = \begin{cases} \max(0, \frac{d-\frac{C}{m}}{s}) & \text{if } s > 0 \\ \max(0, \frac{d}{s}) & \text{otherwise} \end{cases} \tag{7.27}$$

$$H = \begin{cases} \min(\frac{C}{m}, \frac{d}{s}) & \text{if } s > 0 \\ \min(\frac{C}{m}, \frac{d-\frac{C}{m}}{s}) & \text{otherwise,} \end{cases} \tag{7.28}$$

and auxiliary variables

$$\chi = (H_{ii} + H_{jj}s^2 - 2sH_{ij}) \text{ and} \tag{7.29}$$

$$\rho = (c_j s - c_i - H_{ij}d + H_{jj}ds). \tag{7.30}$$

The optimal value of (7.26) *can be computed analytically as follows: If* $\chi = 0$ *then*

$$\alpha_i = \begin{cases} L & \text{if } \rho < 0 \\ H & \text{otherwise.} \end{cases}$$

If $\chi > 0$*, then* $\alpha_i = \max(L, \min(H, \rho/\chi))$*. In both cases,* $\alpha_j = (d - s\alpha_i)$*.*

Proof Eliminate the equality constraint by setting $\alpha_j = (d - s\alpha_i)$. Due to the constraint $0 \leq \alpha_j \leq \frac{C}{m}$ it follows that $s\alpha_i = d - \alpha_j$ can be bounded via $d - \frac{C}{m} \leq s\alpha_i \leq d$. Combining this with $0 \leq \alpha_i \leq \frac{C}{m}$ one can write $L \leq \alpha_i \leq H$ where L and H are given by (7.27) and (7.28) respectively.

Substituting $\alpha_j = (d - s\alpha_i)$ into the objective function, dropping the terms which do not depend on α_i, and simplifying by substituting χ and ρ yields the following optimization problem in α_i:

$$\begin{aligned} \min_{\alpha_i} \quad & \frac{1}{2}\alpha_i^2 \chi - \alpha_i \rho \\ \text{s.t.} \quad & L \leq \alpha_i \leq H. \end{aligned}$$

First consider the case when $\chi = 0$. In this case, $\alpha_i = L$ if $\rho < 0$ otherwise $\alpha_i = H$. On other hand, if $\chi > 0$ then the unconstrained optimum of the above optimization problem is given by ρ/χ. The constrained optimum is obtained by clipping appropriately: $\max(L, \min(H, \rho/\chi))$. This concludes the proof. ■

To complete the description of SMO we need a valid stopping criterion as well as a scheme for selecting the working set at every iteration. In order to derive a stopping criterion we will use the KKT gap, that is, the extent to which the KKT conditions are violated. Towards this end introduce non-negative Lagrange multipliers $b \in \mathbb{R}$, $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^m$ and write the Lagrangian of (7.9).

$$L(\alpha, b, \lambda, \mu) = \frac{1}{2}\alpha^\top H\alpha - \alpha^\top e + b\alpha^\top y - \lambda^\top \alpha + \mu^\top(\alpha - \frac{C}{m}e). \tag{7.31}$$

If we let $J(\alpha) = \frac{1}{2}\alpha^\top H\alpha - \alpha^\top e$ be the objective function and $\nabla J(\alpha) = H\alpha - e$ its gradient, then taking gradient of the Lagrangian with respect to α and setting it to 0 shows that

$$\nabla J(\alpha) + by = \lambda - \mu. \tag{7.32}$$

Furthermore, by the KKT conditions we have

$$\lambda_i \alpha_i = 0 \text{ and } \mu_i(\frac{C}{m} - \alpha_i) = 0, \tag{7.33}$$

with $\lambda_i \geq 0$ and $\mu_i \geq 0$. Equations (7.32) and (7.33) can be compactly rewritten as

$$\nabla J(\alpha)_i + by_i \geq 0 \text{ if } \alpha_i = 0 \tag{7.34a}$$

$$\nabla J(\alpha)_i + by_i \leq 0 \text{ if } \alpha_i = \frac{C}{m} \tag{7.34b}$$

$$\nabla J(\alpha)_i + by_i = 0 \text{ if } 0 < \alpha_i < \frac{C}{m}. \tag{7.34c}$$

Since $y_i \in \{\pm 1\}$, we can further rewrite (7.34) as

$$-y_i \nabla J(\alpha)_i \leq b \text{ for all } i \in I_{up}$$
$$-y_i \nabla J(\alpha)_i \geq b \text{ for all } i \in I_{down},$$

where the index sets I_{up} and I_{down} are defined as

$$I_{up} = \{i : \alpha_i < \frac{C}{m}, y_i = 1 \text{ or } \alpha_i > 0, y_i = -1\} \tag{7.35a}$$

$$I_{down} = \{i : \alpha_i < \frac{C}{m}, y_i = -1 \text{ or } \alpha_i > 0, y_i = 1\}. \tag{7.35b}$$

In summary, the KKT conditions imply that α is a solution of (7.9) if and only if

$$m(\alpha) \leq M(\alpha)$$

where

$$m(\alpha) = \max_{i \in I_{up}} -y_i \nabla J(\alpha)_i \text{ and } M(\alpha) = \min_{i \in I_{down}} -y_i \nabla J(\alpha)_i. \tag{7.36}$$

Therefore, a natural stopping criterion is to stop when the KKT gap falls below a desired tolerance ϵ, that is,

$$m(\alpha) \leq M(\alpha) + \epsilon. \tag{7.37}$$

Finally, we turn our attention to the issue of working set selection. The first order approximation to the objective function $J(\alpha)$ can be written as

$$J(\alpha + d) \approx J(\alpha) + \nabla J(\alpha)^\top d.$$

Since we are only interested in updating coefficients in the working set B we set $d^\top = \begin{bmatrix} d_B^\top & 0 \end{bmatrix}$, in which case we can rewrite the above first order

approximation as

$$\nabla J(\alpha)_B^\top d_B \approx J(\alpha + d) - J(\alpha).$$

From among all possible directions d_B we wish to choose one which decreases the objective function the most while maintaining feasibility. This is best expressed as the following optimization problem:

$$\begin{aligned}
\min_{d_B} \quad & \nabla J(\alpha)_B^\top d_B && \text{(7.38a)}\\
\text{s.t.} \quad & y_B^\top d_B = 0 && \text{(7.38b)}\\
& d_i \geq 0 \text{ if } \alpha_i = 0 \text{ and } i \in B && \text{(7.38c)}\\
& d_i \leq 0 \text{ if } \alpha_i = \frac{C}{m} \text{ and } i \in B && \text{(7.38d)}\\
& -1 \leq d_i \leq 1. && \text{(7.38e)}
\end{aligned}$$

Here (7.38b) comes from $y^\top(\alpha + d) = 0$ and $y^\top \alpha = 0$, while (7.38c) and (7.38d) comes from $0 \leq \alpha_i \leq \frac{C}{m}$. Finally, (7.38e) prevents the objective function from diverging to $-\infty$. If we specialize (7.38) to SMO, we obtain

$$\begin{aligned}
\min_{i,j} \quad & \nabla J(\alpha)_i d_i + \nabla J(\alpha)_j d_j && \text{(7.39a)}\\
\text{s.t.} \quad & y_i d_i + y_j d_j = 0 && \text{(7.39b)}\\
& d_k \geq 0 \text{ if } \alpha_k = 0 \text{ and } k \in \{i, j\} && \text{(7.39c)}\\
& d_k \leq 0 \text{ if } \alpha_k = \frac{C}{m} \text{ and } k \in \{i, j\} && \text{(7.39d)}\\
& -1 \leq d_k \leq 1 \text{ for } k \in \{i, j\}. && \text{(7.39e)}
\end{aligned}$$

At first glance, it seems that choosing the optimal i and j from the set $\{1, \ldots, m\} \times \{1, \ldots m\}$ requires $O(m^2)$ effort. We now show that $O(m)$ effort suffices.

Define new variables $\hat{d}_k = y_k d_k$ for $k \in \{i, j\}$, and use the observation $y_k \in \{\pm 1\}$ to rewrite the objective function as

$$(-y_i \nabla J(\alpha)_i + y_j \nabla J(\alpha)_j)\, \hat{d}_j.$$

Consider the case $-\nabla J(\alpha)_i y_i \geq -\nabla J(\alpha)_j y_j$. Because of the constraints (7.39c) and (7.39d) if we choose $i \in I_{up}$ and $j \in I_{down}$, then $\hat{d}_j = -1$ and $\hat{d}_i = 1$ is feasible and the objective function attains a negative value. For all other choices of i and j ($i, j \in I_{up}$; $i, j \in I_{down}$; $i \in I_{down}$ and $j \in I_{up}$) the objective function value of 0 is attained by setting $\hat{d}_i = \hat{d}_j = 0$. The case $-\nabla J(\alpha)_j y_j \geq -\nabla J(\alpha)_i y_i$ is analogous. In summary, the optimization

problem (7.39) boils down to

$$\min_{i\in I_{up}, j\in I_{down}} y_i \nabla J(\alpha)_i - y_j \nabla J(\alpha)_j = \min_{i\in I_{up}} y_i \nabla J(\alpha)_i - \max_{j\in I_{down}} y_j \nabla J(\alpha)_j,$$

which clearly can be solved in $O(m)$ time. Comparison with (7.36) shows that at every iteration of SMO we choose to update coefficients α_i and α_j which maximally violate the KKT conditions.

7.2 Extensions

7.2.1 The ν trick

In the soft margin formulation the parameter C is a trade-off between two conflicting requirements namely maximizing the margin and minimizing the training error. Unfortunately, this parameter is rather unintuitive and hence difficult to tune. The ν-SVM was proposed to address this issue. As Theorem 7.3 below shows, ν controls the number of support vectors and margin errors. The primal problem for the ν-SVM can be written as

$$\min_{w,b,\xi,\rho} \quad \frac{1}{2}\|w\|^2 - \rho + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i \tag{7.40a}$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq \rho - \xi_i \text{ for all } i \tag{7.40b}$$

$$\xi_i \geq 0, \text{ and } \rho \geq 0. \tag{7.40c}$$

As before, if we write the Lagrangian by introducing non-negative Lagrange multipliers, take gradients with respect to the primal variables and set them to zero, and substitute the result back into the Lagrangian we obtain the following dual:

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle \tag{7.41a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{7.41b}$$

$$\sum_{i=1}^{m} \alpha_i \geq 1 \tag{7.41c}$$

$$0 \leq \alpha_i \leq \frac{1}{\nu m}. \tag{7.41d}$$

It turns out that the dual can be further simplified via the following lemma.

Lemma 7.2 *Let $\nu \in [0,1]$ and (7.41) be feasible. Then there is at least one solution α which satisfies $\sum_i \alpha_i = 1$. Furthermore, if the final objective value of (7.41) is non-zero then all solutions satisfy $\sum_i \alpha_i = 1$.*

Proof The feasible region of (7.41) is bounded, therefore if it is feasible then there exists an optimal solution. Let α denote this solution and assume that $\sum_i \alpha_i > 1$. In this case we can define

$$\bar{\alpha} = \frac{1}{\sum_j \alpha_j} \alpha,$$

and easily check that $\bar{\alpha}$ is also feasible. As before, let H denote a $m \times m$ matrix with $H_{ij} = y_i y_j \langle x_i, x_j \rangle$. Since α is the optimal solution of (7.41) it follows that

$$\frac{1}{2}\alpha^\top H \alpha \leq \frac{1}{2}\bar{\alpha}^\top H \bar{\alpha} = \left(\frac{1}{\sum_j \alpha_j}\right)^2 \frac{1}{2}\alpha^\top H \alpha \leq \frac{1}{2}\alpha^\top H \alpha.$$

This implies that either $\frac{1}{2}\alpha^\top H \alpha = 0$, in which case $\bar{\alpha}$ is an optimal solution with the desired property or $\frac{1}{2}\alpha^\top H \alpha \neq 0$, in which case all optimal solutions satisfy $\sum_i \alpha_i = 1$. ∎

In view of the above theorem one can equivalently replace (7.41) by the following simplified optimization problem with two equality constraints

$$\min_{\alpha} \quad \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \tag{7.42a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{7.42b}$$

$$\sum_{i=1}^{m} \alpha_i = 1 \tag{7.42c}$$

$$0 \leq \alpha_i \leq \frac{1}{\nu m}. \tag{7.42d}$$

The following theorems, which we state without proof, explain the significance of ν and the connection between ν-SVM and the soft margin formulation.

Theorem 7.3 *Suppose we run ν-SVM with kernel k on some data and obtain $\rho > 0$. Then*

(i) *ν is an upper bound on the fraction of margin errors, that is points for which $y_i (\langle w, x_i \rangle + b_i) < \rho$.*

(ii) *ν is a lower bound on the fraction of support vectors, that is points for which $y_i\left(\langle w, x_i\rangle + b_i\right) = \rho$.*

(iii) *Suppose the data (X, Y) were generated iid from a distribution $p(x, y)$ such that neither $p(x, y = +1)$ or $p(x, y = -1)$ contain any discrete components. Moreover, assume that the kernel k is analytic and non-constant. With probability 1, asympotically, ν equals both the fraction of support vectors and fraction of margin errors.*

Theorem 7.4 *If (7.40) leads to a decision function with $\rho > 0$, then (7.5) with $C = \frac{1}{\rho}$ leads to the same decision function.*

7.2.2 Squared Hinge Loss

In binary classification, the actual loss which one would like to minimize is the so-called 0-1 loss

$$l(w, x, y) = \begin{cases} 0 & \text{if } y(\langle w, x\rangle + b) \geq 1 \\ 1 & \text{otherwise .} \end{cases} \tag{7.43}$$

This loss is difficult to work with because it is non-convex (see Figure 7.4). In

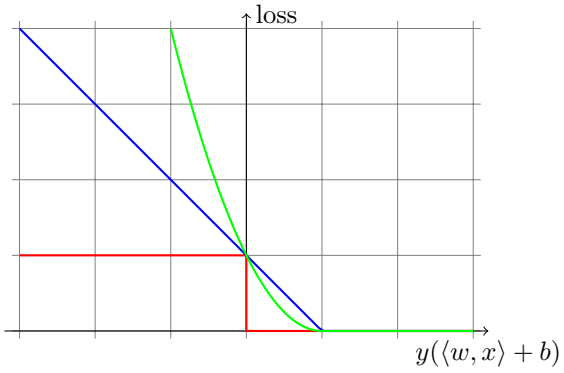


Fig. 7.4. The 0-1 loss which is non-convex and intractable is depicted in red. The hinge loss is a convex upper bound to the 0-1 loss and shown in blue. The square hinge loss is a differentiable convex upper bound to the 0-1 loss and is depicted in green.

fact, it has been shown that finding the optimal (w, b) pair which minimizes the 0-1 loss on a training dataset of m labeled points is NP hard [BDEL03]. Therefore various proxy functions such as the binary hinge loss (7.13) which we discussed in Section 7.1.1 are used. Another popular proxy is the square

hinge loss:

$$l(w, x, y) = \max(0, 1 - y(\langle w, x \rangle + b))^2. \tag{7.44}$$

Besides being a proxy for the 0-1 loss, the squared hinge loss, unlike the hinge loss, is also differentiable everywhere. This sometimes makes the optimization in the primal easier. Just like in the case of the hinge loss one can derive the dual of the regularized risk minimization problem and show that it is a quadratic programming problem (problem 7.5).

7.2.3 Ramp Loss

The ramp loss

$$l(w, x, y) = \min(1 - s, \max(0, 1 - y(\langle w, x \rangle + b))) \tag{7.45}$$

parameterized by $s \leq 0$ is another proxy for the 0-1 loss (see Figure 7.5). Although not convex, it can be expressed as the difference of two convex functions

$$l_{conc}(w, x, y) = \max(0, 1 - y(\langle w, x \rangle + b)) \text{ and}$$
$$l_{cave}(w, x, y) = \max(0, s - y(\langle w, x \rangle + b)).$$

Therefore the Convex-Concave procedure (CCP) we discussed in Section

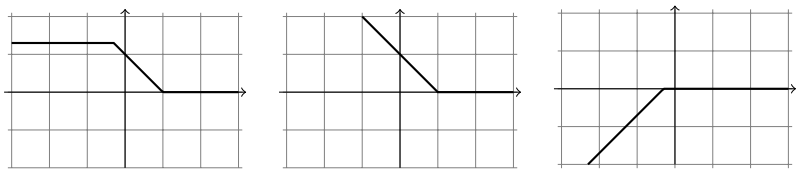


Fig. 7.5. The ramp loss depicted here with $s = -0.3$ can be viewed as the sum of a convex function namely the binary hinge loss (left) and a concave function $\min(0, 1 - y(\langle w, x \rangle + b))$ (right). Viewed alternatively, the ramp loss can be written as the difference of two convex functions.

3.5.1 can be used to solve the resulting regularized risk minimization problem with the ramp loss. Towards this end write

$$J(w) = \underbrace{\frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m} l_{conc}(w, x_i, y_i)}_{J_{conc}(w)} \underbrace{- \frac{C}{m}\sum_{i=1}^{m} l_{cave}(w, x_i, y_i)}_{J_{cave}(w)}. \tag{7.46}$$

Recall that at every iteration of the CCP we replace $J_{cave}(w)$ by its first order Taylor approximation, computing which requires

$$\partial_w J(w) = \frac{C}{m} \sum_{i=1}^{m} \partial_w l_{cave}(w, x_i, y_i). \tag{7.47}$$

This in turn can be computed as

$$\partial_w l_{cave}(w, x_i, y_i) = \delta_i y_i x_i \text{ with } \delta_i = \begin{cases} -1 & \text{if } s > y(\langle w, x \rangle + b) \\ 0 & \text{otherwise.} \end{cases} \tag{7.48}$$

Ignoring constant terms, each iteration of the CCP algorithm involves solving the following minimization problem (also see (3.134))

$$J(w) = \frac{1}{2}\|w\|^2 + \frac{C}{m} \sum_{i=1}^{m} l_{conc}(w, x_i, y_i) - \left(\frac{C}{m} \sum_{i=1}^{m} \delta_i y_i x_i \right) w. \tag{7.49}$$

Let δ denote a vector in $\mathbb{R}^m$ with components δ_i. Using the same notation as in (7.9) we can write the following dual optimization problem which is very closely related to the standard SVM dual (7.9) (see problem 7.6)

$$\min_{\alpha} \quad \frac{1}{2}\alpha^\top H \alpha - \alpha^\top e \tag{7.50a}$$

$$\text{s.t.} \quad \alpha^\top y = 0 \tag{7.50b}$$

$$-\frac{C}{m}\delta \leq \alpha_i \leq \frac{C}{m}(e - \delta). \tag{7.50c}$$

In fact, this problem can be solved by a SMO solver with minor modifications. Putting everything together yields Algorithm 7.1.

Algorithm 7.1 CCP for Ramp Loss

1: Initialize δ^0 and α^0
2: **repeat**
3: Solve (7.50) to find α^{t+1}
4: Compute δ^{t+1} using (7.48)
5: **until** $\delta^{t+1} = \delta^t$

7.3 Support Vector Regression

As opposed to classification where the labels y_i are binary valued, in regression they are real valued. Given a tolerance ϵ, our aim here is to find a

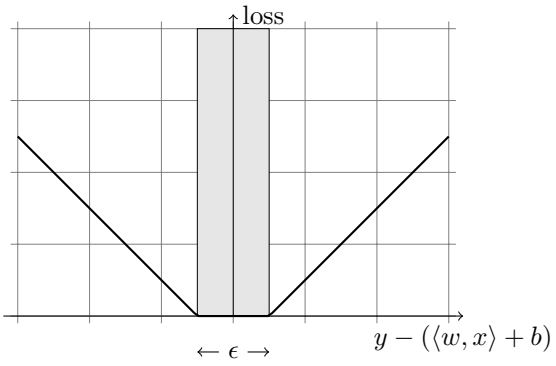


Fig. 7.6. The ϵ insensitive loss. All points which lie within the ϵ tube shaded in gray incur zero loss while points outside incur a linear loss.

hyperplane parameterized by (w, b) such that

$$|y_i - (\langle w, x_i \rangle + b)| \leq \epsilon. \tag{7.51}$$

In other words, we want to find a hyperplane such that all the training data lies within an ϵ tube around the hyperplane. We may not always be able to find such a hyperplane, hence we relax the above condition by introducing slack variables ξ_i^+ and ξ_i^- and write the corresponding primal problem as

$$\begin{aligned}
\min_{w,b,\xi^+,\xi^-} \quad & \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-) && \text{(7.52a)}\\
\text{s.t.} \quad & y_i - (\langle w, x_i \rangle + b) \leq \epsilon + \xi_i^+ \text{ for all } i && \text{(7.52b)}\\
& (\langle w, x_i \rangle + b) - y_i \leq \epsilon + \xi_i^- \text{ for all } i && \text{(7.52c)}\\
& \xi_i^+ \geq 0, \text{ and } \xi_i^- \geq 0. && \text{(7.52d)}
\end{aligned}$$

The Lagrangian can be written by introducing non-negative Lagrange multipliers α_i^+, α_i^-, β_i^+ and β_i^-:

$$\begin{aligned}
L(w, b, \xi^+, \xi^-, \alpha^+, \alpha^-, \beta^+, \beta^-) =& \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-) - \sum_{i=1}^{m}(\beta_i^+\xi_i^+ + \beta_i^-\xi_i^-)\\
&+ \sum_{i=1}^{m}\alpha_i^+(y_i - (\langle w, x_i \rangle + b) - \epsilon - \xi^+)\\
&+ \sum_{i=1}^{m}\alpha_i^-((\langle w, x_i \rangle + b) - y_i - \epsilon - \xi^-).
\end{aligned}$$

Taking gradients with respect to the primal variables and setting them to 0, we obtain the following conditions:

$$w = \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-) x_i \tag{7.53}$$

$$\sum_{i=1}^{m} \alpha_i^+ = \sum_{i=1}^{m} \alpha_i^- \tag{7.54}$$

$$\alpha_i^+ + \beta_i^+ = \frac{C}{m} \tag{7.55}$$

$$\alpha_i^- + \beta_i^- = \frac{C}{m}. \tag{7.56}$$

Noting that $\alpha_i^{\{+,-\}}, \beta_i^{\{+,-\}} \geq 0$ and substituting the above conditions into the Lagrangian yields the dual

$$\min_{\alpha^+,\alpha^-} \quad \frac{1}{2} \sum_{i,j} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) \langle x_i, x_j \rangle \tag{7.57a}$$

$$+ \epsilon \sum_{i=1}^{m} (\alpha_i^+ + \alpha_i^-) - \sum_{i=1}^{m} y_i (\alpha_i^+ - \alpha_i^-)$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i^+ = \sum_{i=1}^{m} \alpha_i^- \tag{7.57b}$$

$$0 \leq \alpha_i^+ \leq \frac{C}{m} \tag{7.57c}$$

$$0 \leq \alpha_i^- \leq \frac{C}{m}. \tag{7.57d}$$

This is a quadratic programming problem with one equality constraint, and hence a SMO like decomposition method can be derived for finding the optimal coefficients α^+ and α^- (Problem 7.7).

As a consequence of (7.53), analogous to the classification case, one can map the data via a feature map ϕ into an RKHS with kernel k and recover the decision boundary $f(x) = \langle w, \phi(x) \rangle + b$ via

$$f(x) = \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-) \langle \phi(x)_i, \phi(x) \rangle + b = \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-) k(x_i, x) + b. \tag{7.58}$$

Finally, the KKT conditions

$$\left(\frac{C}{m} - \alpha_i^+\right) \xi_i^+ = 0 \quad \left(\frac{C}{m} - \alpha_i^-\right) \xi_i^- = 0 \text{ and}$$

$$\alpha_i^- ((\langle w, x_i \rangle + b) - y_i - \epsilon - \xi^-) = 0 \quad \alpha_i^+ (y_i - (\langle w, x_i \rangle + b) - \epsilon - \xi^+) = 0,$$

allow us to draw many useful conclusions:

- Whenever $|y_i - (\langle w, x_i \rangle + b)| < \epsilon$, this implies that $\xi_i^+ = \xi_i^- = \alpha_i^+ = \alpha_i^- = 0$. In other words, points which lie inside the ϵ tube around the hyperplane $\langle w, x \rangle + b$ do not contribute to the solution thus leading to sparse expansions in terms of α.
- If $(\langle w, x_i \rangle + b) - y_i > \epsilon$ we have $\xi_i^- > 0$ and therefore $\alpha_i^- = \frac{C}{m}$. On the other hand, $\xi^+ = 0$ and $\alpha_i^+ = 0$. The case $y_i - (\langle w, x_i \rangle + b) > \epsilon$ is symmetric and yields $\xi^- = 0$, $\xi_i^+ > 0$, $\alpha_i^+ = \frac{C}{m}$, and $\alpha_i^- = 0$.
- Finally, if $(\langle w, x_i \rangle + b) - y_i = \epsilon$ we have $\xi_i^- = 0$ and $0 \leq \alpha_i^- \leq \frac{C}{m}$, while $\xi^+ = 0$ and $\alpha_i^+ = 0$. Similarly, when $y_i - (\langle w, x_i \rangle + b) = \epsilon$ we obtain $\xi_i^+ = 0$, $0 \leq \alpha_i^+ \leq \frac{C}{m}$, $\xi^- = 0$ and $\alpha_i^- = 0$.

Note that α_i^+ and α_i^- are never simultaneously non-zero.

7.3.1 Incorporating General Loss Functions

Using the same reasoning as in Section 7.1.1 we can deduce from (7.52) that the loss function of support vector regression is given by

$$l(w, x, y) = \max(0, |y - \langle w, x \rangle| - \epsilon). \tag{7.59}$$

It turns out that the support vector regression framework can be easily extended to handle other, more general, convex loss functions such as the ones found in Table 7.1. Different losses have different properties and hence lead to different estimators. For instance, the square loss leads to penalized least squares (LS) regression, while the Laplace loss leads to the penalized least absolute deviations (LAD) estimator. Huber's loss on the other hand is a combination of the penalized LS and LAD estimators, and the pinball loss with parameter $\tau \in [0, 1]$ is used to estimate τ-quantiles. Setting $\tau = 0.5$ in the pinball loss leads to a scaled version of the Laplace loss. If we define $\xi = y - \langle w, x \rangle$, then it is easily verified that all these losses can all be written as

$$l(w, x, y) = \begin{cases} l^+(\xi - \epsilon) & \text{if } \xi > \epsilon \\ l^-(-\xi - \epsilon) & \text{if } \xi < \epsilon \\ 0 & \text{if } \xi \in [-\epsilon, \epsilon]. \end{cases} \tag{7.60}$$

For all these different loss functions, the support vector regression formu-

lation can be written in a unified fashion as follows

$$\min_{w,b,\xi^+,\xi^-} \quad \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m} l^+(\xi_i^+) + l^-(\xi_i^-) \tag{7.61a}$$

$$\text{s.t.} \quad y_i - (\langle w, x_i\rangle + b) \leq \epsilon + \xi_i^+ \text{ for all } i \tag{7.61b}$$

$$(\langle w, x_i\rangle + b) - y_i \leq \epsilon + \xi_i^- \text{ for all } i \tag{7.61c}$$

$$\xi_i^+ \geq 0, \text{ and } \xi_i^- \geq 0. \tag{7.61d}$$

The dual in this case is given by

$$\min_{\alpha^+,\alpha^-} \quad \frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\langle x_i, x_j\rangle \tag{7.62a}$$

$$- \frac{C}{m}\sum_{i=1}^{m} T^+(\xi^+) + T^-(\xi^-) + \epsilon\sum_{i=1}^{m}(\alpha_i^+ + \alpha_i^-) - \sum_{i=1}^{m} y_i(\alpha_i^+ - \alpha_i^-)$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\alpha_i^+ = \sum_{i=1}^{m}\alpha_i^- \tag{7.62b}$$

$$0 \leq \alpha_i^{\{+,-\}} \leq \frac{C}{m}\partial_\xi l^{\{+,-\}}(\xi_i^{\{+,-\}}) \tag{7.62c}$$

$$0 \leq \xi_i^{\{+,-\}} \tag{7.62d}$$

$$\xi_i^{\{+,-\}} = \inf\left\{\xi^{\{+,-\}} \mid \frac{C}{m}\partial_\xi l^{\{+,-\}} \geq \alpha_i^{\{+,-\}}\right\}. \tag{7.62e}$$

Here $T^+(\xi) = l^+(\xi) - \xi\partial_\xi l^+(\xi)$ and $T^-(\xi) = l^-(\xi) - \xi\partial_\xi l^-(\xi)$. We now show how (7.62) can be specialized to the pinball loss. Clearly, $l^+(\xi) = \tau\xi$ while $l^-(-\xi) = (\tau-1)\xi$, and hence $l^-(\xi) = (1-\tau)\xi$. Therefore, $T^+(\xi) = (\tau-1)\xi - \xi(\tau-1) = 0$. Similarly $T^-(\xi) = 0$. Since $\partial_\xi l^+(\xi) = \tau$ and $\partial_\xi l^-(\xi) = (1-\tau)$ for all $\xi \geq 0$, it follows that the bounds on $\alpha^{\{+,-\}}$ can be computed as $0 \leq \alpha_i^+ \leq \frac{C}{m}\tau$ and $0 \leq \alpha_i^- \leq \frac{C}{m}(1-\tau)$. If we denote $\alpha = \alpha^+ - \alpha^-$ and

Table 7.1. *Various loss functions which can be used in support vector regression. For brevity we denote* $y - \langle w, x\rangle$ *as* ξ *and write the loss* $l(w, x, y)$ *in terms of* ξ.

ϵ-insensitive loss	$\max(0, \lvert\xi\rvert - \epsilon)$
Laplace loss	$\lvert\xi\rvert$
Square loss	$\frac{1}{2}\lvert\xi\rvert^2$
Huber's robust loss	$\begin{cases}\frac{1}{2\sigma}\xi^2 & \text{if } \lvert\xi\rvert \leq \sigma \\ \lvert\xi\rvert - \frac{\sigma}{2} & \text{otherwise}\end{cases}$
Pinball loss	$\begin{cases}\tau\xi & \text{if } \xi \geq 0 \\ (\tau-1)\xi & \text{otherwise.}\end{cases}$

observe that $\epsilon = 0$ for the pinball loss then (7.62) specializes as follows:

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\left\langle x_i, x_j\right\rangle - \sum_{i=1}^{m} y_i\alpha_i \tag{7.63a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\alpha_i = 0 \tag{7.63b}$$

$$\frac{C}{m}(\tau - 1) \leq \alpha_i \leq \frac{C}{m}\tau. \tag{7.63c}$$

Similar specializations of (7.62) for other loss functions in Table 7.1 can be derived.

7.3.2 Incorporating the ν Trick

One can also incorporate the ν trick into support vector regression. The primal problem obtained after incorporating the ν trick can be written as

$$\min_{w,b,\xi^+,\xi^-,\epsilon} \quad \frac{1}{2}\|w\|^2 + \left(\epsilon + \frac{1}{\nu m}\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-)\right) \tag{7.64a}$$

$$\text{s.t.} \quad (\langle w, x_i\rangle + b) - y_i \leq \epsilon + \xi_i^+ \text{ for all } i \tag{7.64b}$$

$$y_i - (\langle w, x_i\rangle + b) \leq \epsilon + \xi_i^- \text{ for all } i \tag{7.64c}$$

$$\xi_i^+ \geq 0, \xi_i^- \geq 0, \text{ and } \epsilon \geq 0. \tag{7.64d}$$

Proceeding as before we obtain the following simplified dual

$$\min_{\alpha^+,\alpha^-} \quad \frac{1}{2}\sum_{i,j}(\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+)\left\langle x_i, x_j\right\rangle - \sum_{i=1}^{m} y_i(\alpha_i^- - \alpha_i^+) \tag{7.65a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}(\alpha_i^- - \alpha_i^+) = 0 \tag{7.65b}$$

$$\sum_{i=1}^{m}(\alpha_i^- + \alpha_i^+) = 1 \tag{7.65c}$$

$$0 \leq \alpha_i^+ \leq \frac{1}{\nu m} \tag{7.65d}$$

$$0 \leq \alpha_i^- \leq \frac{1}{\nu m}. \tag{7.65e}$$

7.4 Novelty Detection

The large margin approach can also be adapted to perform novelty detection or quantile estimation. Novelty detection is an unsupervised task where one

is interested in flagging a small fraction of the input $X = \{x_1, \ldots, x_m\}$ as atypical or novel. It can be viewed as a special case of the quantile estimation task, where we are interested in estimating a *simple* set $\mathcal{C}$ such that $Pr(x \in \mathcal{C}) \geq \mu$ for some $\mu \in [0, 1]$. One way to measure simplicity is to use the volume of the set. Formally, if $|\mathcal{C}|$ denotes the volume of a set, then the quantile estimation task is to estimate

$$\operatorname{arginf}\{|\mathcal{C}| \text{ s.t. } Pr(x \in \mathcal{C}) \geq \mu\}. \tag{7.66}$$

Given the input data X one can compute the empirical density

$$\hat{p}(x) = \begin{cases} \frac{1}{m} & \text{if } x \in X \\ 0 & \text{otherwise,} \end{cases}$$

and estimate its (not necessarily unique) μ-quantiles. Unfortunately, such estimates are very brittle and do not generalize well to unseen data. One possible way to address this issue is to restrict $\mathcal{C}$ to be simple subsets such as spheres or half spaces. In other words, we estimate simple sets which contain μ fraction of the dataset. For our purposes, we specifically work with half-spaces defined by hyperplanes. While half-spaces may seem rather restrictive remember that the kernel trick can be used to map data into a high-dimensional space; half-spaces in the mapped space correspond to non-linear decision boundaries in the input space. Furthermore, instead of explicitly identifying $\mathcal{C}$ we will learn an indicator function for $\mathcal{C}$, that is, a function f which takes on values -1 inside $\mathcal{C}$ and -1 elsewhere.

With $\frac{1}{2}\|w\|^2$ as a regularizer, the problem of estimating a hyperplane such that a large fraction of the points in the input data X lie on one of its sides can be written as:

$$\min_{w,\xi,\rho} \quad \frac{1}{2}\|w\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i - \rho \tag{7.67a}$$

$$\text{s.t.} \quad \langle w, x_i \rangle \geq \rho - \xi_i \text{ for all } i \tag{7.67b}$$

$$\xi_i \geq 0. \tag{7.67c}$$

Clearly, we want ρ to be as large as possible so that the volume of the half-space $\langle w, x \rangle \geq \rho$ is minimized. Furthermore, $\nu \in [0, 1]$ is a parameter which is analogous to ν we introduced for the ν-SVM earlier. Roughly speaking, it denotes the fraction of input data for which $\langle w, x_i \rangle \leq \rho$. An alternative interpretation of (7.67) is to assume that we are separating the data set X from the origin (See Figure 7.7 for an illustration). Therefore, this method is also widely known as the one-class SVM.

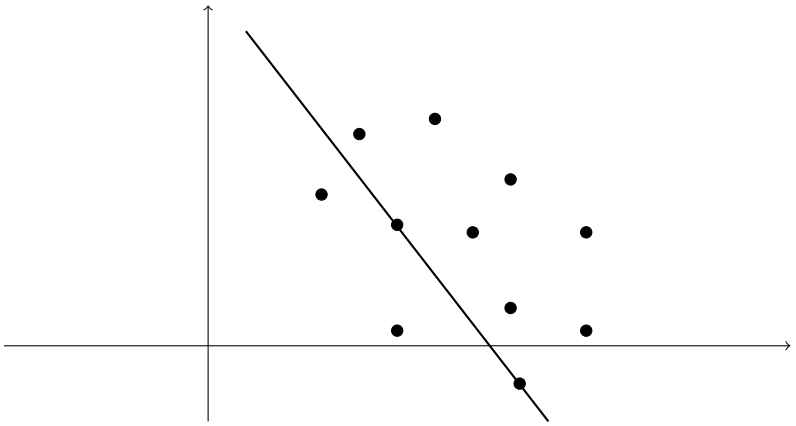


Fig. 7.7. The novelty detection problem can be viewed as finding a large margin hyperplane which separates ν fraction of the data points away from the origin.

The Lagrangian of (7.67) can be written by introducing non-negative Lagrange multipliers α_i, and β_i:

$$L(w, \xi, \rho, \alpha, \beta) = \frac{1}{2}\|w\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i - \rho + \sum_{i=1}^{m}\alpha_i(\rho - \xi_i - \langle w, x_i\rangle) - \sum_{i=1}^{m}\beta_i\xi_i.$$

By taking gradients with respect to the primal variables and setting them to 0 we obtain

$$w = \sum_{i=1}^{m}\alpha_i x_i \tag{7.68}$$

$$\alpha_i = \frac{1}{\nu m} - \beta_i \leq \frac{1}{\nu m} \tag{7.69}$$

$$\sum_{i=1}^{m}\alpha_i = 1. \tag{7.70}$$

Noting that $\alpha_i, \beta_i \geq 0$ and substituting the above conditions into the Lagrangian yields the dual

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\langle x_i, x_j\rangle \tag{7.71a}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq \frac{1}{\nu m} \tag{7.71b}$$

$$\sum_{i=1}^{m}\alpha_i = 1. \tag{7.71c}$$

This can easily be solved by a straightforward modification of the SMO algorithm (see Section 7.1.3 and Problem 7.7). Like in the previous sections, an analysis of the KKT conditions shows that $0 < \alpha$ if and only if $\langle w, x_i \rangle \leq \rho$; such points are called support vectors. As before, we can replace $\langle x_i, x_j \rangle$ by a kernel $k(x_i, x_j)$ to transform half-spaces in the feature space to non-linear shapes in the input space. The following theorem explains the significance of the parameter ν.

Theorem 7.5 *Assume that the solution of (7.71) satisfies $\rho \neq 0$, then the following statements hold:*

(i) *ν is an upper bound on the fraction of support vectors, that is points for which $\langle w, x_i \rangle \leq \rho$.*
(ii) *Suppose the data X were generated independently from a distribution $p(x)$ which does not contain discrete components. Moreover, assume that the kernel k is analytic and non-constant. With probability 1, asympotically, ν equals the fraction of support vectors.*

7.5 Margins and Probability

discuss the connection between probabilistic models and linear classifiers. issues of consistency, optimization, efficiency, etc.

7.6 Beyond Binary Classification

In contrast to binary classification where there are only two possible ways to label a training sample, in some of the extensions we discuss below each training sample may be associated with one or more of k possible labels. Therefore, we will use the decision function

$$y^* = \underset{y \in \{1,\ldots,k\}}{\operatorname{argmax}} f(x, y) \text{ where } f(x, y) = \langle \phi(x, y), w \rangle . \tag{7.72}$$

Recall that the joint feature map $\phi(x, y)$ was introduced in section 7.1.2. One way to interpret the above equation is to view $f(x, y)$ as a compatibility score between instance x and label y; we assign the label with the highest compatibility score to x. There are a number of extensions of the binary hinge loss (7.13) which can be used to estimate this score function. In all these cases the objective function is written as

$$\min_w J(w) := \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} l(w, x_i, y_i). \tag{7.73}$$

Here λ is a scalar which trades off the regularizer $\frac{1}{2}\|w\|^2$ with the empirical risk $\frac{1}{m}\sum_{i=1}^{m} l(w, x_i, y_i)$. Plugging in different loss functions yields classifiers for different settings. Two strategies exist for finding the optimal w. Just like in the binary SVM case, one can compute and maximize the dual of (7.73). However, the number of dual variables becomes $m|\mathcal{Y}|$, where m is the number of training points and $|\mathcal{Y}|$ denotes the size of the label set. The second strategy is to optimize (7.73) directly. However, the loss functions we discuss below are non-smooth, therefore non-smooth optimization algorithms such as bundle methods (section 3.2.7) need to be used.

7.6.1 Multiclass Classification

In multiclass classification a training example is labeled with one of k possible labels, that is, $\mathcal{Y} = \{1, \ldots, k\}$. We discuss two different extensions of the binary hinge loss to the multiclass setting. It can easily be verified that setting $\mathcal{Y} = \{\pm 1\}$ and $\phi(x, y) = \frac{y}{2}\phi(x)$ recovers the binary hinge loss in both cases.

7.6.1.1 Additive Multiclass Hinge Loss

A natural generalization of the binary hinge loss is to penalize all labels which have been misclassified. The loss can now be written as

$$l(w, x, y) = \sum_{y' \neq y} \max\left(0, 1 - \left(\left\langle \phi(x, y) - \phi(x, y'), w \right\rangle\right)\right). \tag{7.74}$$

7.6.1.2 Maximum Multiclass Hinge Loss

Another variant of (7.13) penalizes only the maximally violating label:

$$l(w, x, y) := \max\left(0, \max_{y' \neq y}(1 - \left\langle \phi(x, y) - \phi(x, y'), w \right\rangle)\right). \tag{7.75}$$

Note that both (7.74) and (7.75) are zero whenever

$$f(x, y) = \langle \phi(x, y), w \rangle \geq 1 + \max_{y' \neq y} \left\langle \phi(x, y'), w \right\rangle = 1 + \max_{y' \neq y} f(x, y'). \tag{7.76}$$

In other words, they both ensure an adequate margin of separation, in this case 1, between the score of the true label $f(x, y)$ and every other label $f(x, y')$. However, they differ in the way they penalize violators, that is, labels $y' \neq y$ for which $f(x, y) \leq 1 + f(x, y')$. In one case we linearly penalize the violators and sum up their contributions while in the other case we linearly penalize only the maximum violator. In fact, (7.75) can be interpreted

as the log odds ratio in the exponential family. Towards this end choose η such that $\log \eta = 1$ and rewrite (7.20):

$$\log \frac{p(y|x,w)}{\max_{y' \neq y} p(y'|x,w)} = \left\langle \phi(x,y) - \max_{y' \neq y} \phi(x,y'), w \right\rangle \geq 1.$$

Rearranging yields (7.76).

7.6.2 Multilabel Classification

In multilabel classification one or more of k possible labels are assigned to a training example. Just like in the multiclass case two different losses can be defined.

7.6.2.1 Additive Multilabel Hinge Loss

If we let $\mathcal{Y}_x \subseteq \mathcal{Y}$ denote the labels assigned to x, and generalize the hinge loss to penalize all labels $y' \notin \mathcal{Y}_x$ which have been assigned higher score than some $y \in \mathcal{Y}_x$, then the loss can be written as

$$l(w,x,y) = \sum_{y \in \mathcal{Y}_x \text{ and } y' \notin \mathcal{Y}_x} \max\left(0, 1 - \left(\left\langle \phi(x,y) - \phi(x,y'), w \right\rangle\right)\right). \tag{7.77}$$

7.6.2.2 Maximum Multilabel Hinge Loss

Another variant only penalizes the maximum violating pair. In this case the loss can be written as

$$l(w,x,y) = \max\left(0, \max_{y \in \mathcal{Y}_x, y' \notin \mathcal{Y}_x} \left[1 - \left(\left\langle \phi(x,y) - \phi(x,y'), w \right\rangle\right)\right]\right). \tag{7.78}$$

One can immediately verify that specializing the above losses to the multiclass case recovers (7.74) and (7.75) respectively, while the binary case recovers (7.13). The above losses are zero only when

$$\min_{y \in \mathcal{Y}_x} f(x,y) = \min_{y \in \mathcal{Y}_x} \langle \phi(x,y), w \rangle \geq 1 + \max_{y' \notin \mathcal{Y}_x} \langle \phi(x,y'), w \rangle = 1 + \max_{y' \notin \mathcal{Y}_x} f(x,y').$$

This can be interpreted as follows: The losses ensure that all the labels assigned to x have larger scores compared to labels not assigned to x with the margin of separation of at least 1.

Although the above loss functions are compatible with multiple labels, the prediction function $\operatorname{argmax}_y f(x,y)$ only takes into account the label with the highest score. This is a significant drawback of such models, which can be overcome by using a multiclass approach instead. Let $|\mathcal{Y}|$ be the size of the label set and $z \in \mathbb{R}^{|\mathcal{Y}|}$ denote a vector with ± 1 entries. We set

$z_y = +1$ if the $y \in \mathcal{Y}_x$ and $z_y = -1$ otherwise, and use the multiclass loss (7.75) on z. To predict we compute $z^* = \operatorname{argmax}_z f(x, z)$ and assign to x the labels corresponding to components of z^* which are $+1$. Since z can take on $2^{|\mathcal{Y}|}$ possible values, this approach is not feasible if $|\mathcal{Y}|$ is large. To tackle such problems, and to further reduce the computational complexity we assume that the labels correlations are captured via a $|\mathcal{Y}| \times |\mathcal{Y}|$ positive semi-definite matrix P, and $\phi(x, y)$ can be written as $\phi(x) \otimes Py$. Here $\otimes$ denotes the Kronecker product. Furthermore, we express the vector w as a $n \times |\mathcal{Y}|$ matrix W, where n denotes the dimension of $\phi(x)$. With these assumptions $\langle \phi(x) \otimes P(z - z'), w \rangle$ can be rewritten as

$$\left\langle \phi(x)^\top WP, (z - z') \right\rangle = \sum_i \left[\phi(x)^\top WP \right]_i (z_i - z'_i),$$

and (7.78) specializes to

$$l(w, x, z) := \max \left(0, \left(1 - \sum_i \min_{z'_i \neq z_i} \left[\phi(x)^\top WP \right]_i (z_i - z'_i) \right) \right). \tag{7.79}$$

A analogous specialization of (7.77) can also be derived wherein the minimum is replaced by a summation. Since the minimum (or summation as the case may be) is over $|\mathcal{Y}|$ possible labels, computing the loss is tractable even if the set of labels $\mathcal{Y}$ is large.

7.6.3 Ordinal Regression and Ranking

We can generalize our above discussion to consider slightly more general ranking problems. Denote by $\mathcal{Y}$ the set of all directed acyclic graphs on N nodes. The presence of an edge (i, j) in $y \in \mathcal{Y}$ indicates that i is preferred to j. The goal is to find a function $f(x, i)$ which imposes a total order on $\{1, \ldots, N\}$ which is in close agreement with y. Specifically, if the estimation error is given by the number of subgraphs of y which are in disagreement with the total order imposed by f, then the additive version of the loss can be written as

$$l(w, x, y) = \sum_{G \in \mathcal{A}(y)} \max_{(i,j) \in G} \left(0, 1 - (f(x, i) - f(x, j)) \right), \tag{7.80}$$

where $\mathcal{A}(y)$ denotes the set of all possible subgraphs of y. The maximum margin version, on the other hand, is given by

$$l(w, x, y) = \max_{G \in \mathcal{A}(y)} \max_{(i,j) \in G} \left(0, 1 - (f(x, i) - f(x, j)) \right). \tag{7.81}$$

In other words, we test for each subgraph G of y if the ranking imposed by G is satisfied by f. Selecting specific types of directed acyclic graphs recovers the multiclass and multilabel settings (problem 7.9).

7.7 Large Margin Classifiers with Structure

7.7.1 Margin

define margin pictures

7.7.2 Penalized Margin

different types of loss, rescaling

7.7.3 Nonconvex Losses

the max - max loss

7.8 Applications

7.8.1 Sequence Annotation

7.8.2 Matching

7.8.3 Ranking

7.8.4 Shortest Path Planning

7.8.5 Image Annotation

7.8.6 Contingency Table Loss

7.9 Optimization

7.9.1 Column Generation

subdifferentials

7.9.2 Bundle Methods

7.9.3 Overrelaxation in the Dual

when we cannot do things exactly

7.10 CRFs vs Structured Large Margin Models

7.10.1 Loss Function

7.10.2 Dual Connections

7.10.3 Optimization

Problems

Problem 7.1 (Deriving the Margin {1}) *Show that the distance of a point x_i to a hyperplane $\mathcal{H} = \{x | \langle w, x\rangle + b = 0\}$ is given by $|\langle w, x_i\rangle + b| / \|w\|$.*

Problem 7.2 (SVM without Bias {1}) *A homogeneous hyperplane is one which passes through the origin, that is,*

$$\mathcal{H} = \{x | \langle w, x\rangle = 0\}. \tag{7.82}$$

If we devise a soft margin classifier which uses the homogeneous hyperplane as a decision boundary, then the corresponding primal optimization problem can be written as follows:

$$\min_{w,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \tag{7.83a}$$

$$s.t. \quad y_i \langle w, x_i\rangle \geq 1 - \xi_i \text{ for all } i \tag{7.83b}$$

$$\xi_i \geq 0, \tag{7.83c}$$

Derive the dual of (7.83) and contrast it with (7.9). What changes to the SMO algorithm would you make to solve this dual?

Problem 7.3 (Deriving the simplified ν-SVM dual {2}) *In Lemma 7.2 we used (7.41) to show that the constraint $\sum_i \alpha_i \geq 1$ can be replaced by $\sum_i \alpha_i = 1$. Show that an equivalent way to arrive at the same conclusion is by arguing that the constraint $\rho \geq 0$ is redundant in the primal (7.40). **Hint:** Observe that whenever $\rho < 0$ the objective function is always non-negative. On the other hand, setting $w = \xi = b = \rho = 0$ yields an objective function value of 0.*

Problem 7.4 (Fenchel and Lagrange Duals {2}) *We derived the Lagrange dual of (7.12) in Section 7.1 and showed that it is (7.9). Derive the Fenchel dual of (7.12) and relate it to (7.9). **Hint:** See theorem 3.3.5 of [BL00].*

Problem 7.5 (Dual of the square hinge loss {1}) *The analog of* (7.5) *when working with the square hinge loss is the following*

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i^2 \tag{7.84a}$$

$$s.t. \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \text{ for all } i \tag{7.84b}$$

$$\xi_i \geq 0, \tag{7.84c}$$

Derive the Lagrange dual of the above optimization problem and show that it a Quadratic Programming problem.

Problem 7.6 (Dual of the ramp loss {1}) *Derive the Lagrange dual of* (7.49) *and show that it the Quadratic Programming problem* (7.50)*.*

Problem 7.7 (SMO for various SVM formulations {2}) *Derive an SMO like decomposition algorithm for solving the dual of the following problems:*

- *ν-SVM* (7.41)*.*
- *SV regression* (7.57)*.*
- *SV novelty detection* (7.71)*.*

Problem 7.8 (Novelty detection with Balls {2}) *In Section 7.4 we assumed that we wanted to estimate a halfspace which contains a major fraction of the input data. An alternative approach is to use balls, that is, we estimate a ball of small radius in feature space which encloses a majority of the input data. Write the corresponding optimization problem and its dual. Show that if the kernel is translation invariant, that is, $k(x, x')$ depends only on $\|x - x'\|$ then the optimization problem with balls is equivalent to* (7.71)*. Explain why this happens geometrically.*

Problem 7.9 (Multiclass and Multilabel loss from Ranking Loss {1}) *Show how the multiclass (resp. multilabel) losses* (7.74) *and* (7.75) *(resp.* (7.77) *and* (7.79)*) can be derived as special cases of* (7.80) *and* (7.81) *respectively.*

Problem 7.10 *Invariances (basic loss)*

Problem 7.11 *Polynomial transformations - SDP constraints*

Appendix 1

Linear Algebra and Functional Analysis

A1.1 Johnson Lindenstrauss Lemma

Lemma 1.1 (Johnson Lindenstrauss) *Let X be a set of n points in $\mathbb{R}^d$ represented as a $n \times d$ matrix A. Given $\epsilon, \beta > 0$ let*

$$k \geq \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n \tag{1.1}$$

be a positive integer. Construct a $d \times k$ random matrix R with independent standard normal random variables, that is, $R_{ij} \sim \mathcal{N}(0, 1)$, and let

$$E = \frac{1}{\sqrt{k}} AR. \tag{1.2}$$

Define $f : \mathbb{R}^d \to \mathbb{R}^k$ as the function which maps the rows of A to the rows of E. With probability at least $1 - n^{-\beta}$, for all $u, v \in X$ we have

$$(1 - \epsilon) \|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon) \|u - v\|^2 . \tag{1.3}$$

Our proof presentation by and large follows [**?**]. We first show that

Lemma 1.2 *For any arbitrary vector $\alpha \in \mathbb{R}^d$ let q_i denote the i-th component of $f(\alpha)$. Then $q_i \sim \mathcal{N}(0, \|\alpha\|^2 / k)$ and hence*

$$\mathbb{E}\left[\|f(\alpha)\|^2\right] = \sum_{i=1}^{k} \mathbb{E}\left[q_i^2\right] = \|\alpha\|^2 . \tag{1.4}$$

In other words, the expected length of vectors are preserved even after embedding them in a k dimensional space. Next we show that the lengths of the embedded vectors are tightly concentrated around their mean.

Lemma 1.3 *For any $\epsilon > 0$ and any unit vector $\alpha \in \mathbb{R}^d$ we have*

$$Pr\left(\|f(\alpha)\|^2 > 1 + \epsilon\right) < \exp\left(-\frac{k}{2}\left(\epsilon^2/2 - \epsilon^3/3\right)\right) \tag{1.5}$$

$$Pr\left(\|f(\alpha)\|^2 < 1 - \epsilon\right) < \exp\left(-\frac{k}{2}\left(\epsilon^2/2 - \epsilon^3/3\right)\right). \tag{1.6}$$

Corollary 1.4 *If we choose k as in (1.1) then for any $\alpha \in \mathbb{R}^d$ we have*

$$Pr\left((1-\epsilon)\|\alpha\|^2 \leq \|f(\alpha)\|^2 \leq (1+\epsilon)\|\alpha\|^2\right) \geq 1 - \frac{2}{n^{2+\beta}}. \tag{1.7}$$

Proof Follows immediately from Lemma 1.3 by setting

$$2\exp\left(-\frac{k}{2}\left(\epsilon^2/2 - \epsilon^3/3\right)\right) \leq \frac{2}{n^{2+\beta}},$$

and solving for k. ■

There are $\binom{n}{2}$ pairs of vectors u, v in X, and their corresponding distances $\|u-v\|$ are preserved within $1 \pm \epsilon$ factor as shown by the above lemma. Therefore, the probability of not satisfying (1.3) is bounded by $\binom{n}{2} \cdot \frac{2}{n^{2+\beta}} < 1/n^{\beta}$ as claimed in the Johnson Lindenstrauss Lemma. All that remains is to prove Lemma 1.2 and 1.3.

Proof (Lemma 1.2). Since $q_i = \frac{1}{\sqrt{k}} \sum_j R_{ij}\alpha_j$ is a linear combination of standard normal random variables R_{ij} it follows that q_i is normally distributed. To compute the mean note that

$$\mathbb{E}[q_i] = \frac{1}{\sqrt{k}} \sum_j \alpha_j \mathbb{E}[R_{ij}] = 0.$$

Since R_{ij} are independent zero mean unit variance random variables, $\mathbb{E}[R_{ij}R_{il}] = 1$ if $j = l$ and 0 otherwise. Using this

$$\mathbb{E}\left[q_i^2\right] = \frac{1}{k}\mathbb{E}\left(\sum_{j=1}^{d} R_{ij}\alpha_j\right)^2 = \frac{1}{k}\sum_{j=1}^{d}\sum_{l=1}^{d}\alpha_j\alpha_l\,\mathbb{E}[R_{ij}R_{il}] = \frac{1}{k}\sum_{j=1}^{d}\alpha_j^2 = \frac{1}{k}\|\alpha\|^2.$$

■

Proof (Lemma 1.3). Clearly, for all λ

$$Pr\left[\|f(\alpha)\|^2 > 1+\epsilon\right] = Pr\left[\exp\left(\lambda\|f(\alpha)\|^2\right) > \exp(\lambda(1+\epsilon))\right].$$

Using Markov's inequality ($Pr[X \geq a] \leq \mathbb{E}[X]/a$) we obtain

$$\begin{aligned} Pr\left[\exp\left(\lambda \left\|f(\alpha)\right\|^2\right) > \exp(\lambda(1+\epsilon))\right] &\leq \frac{\mathbb{E}\left[\exp\left(\lambda \left\|f(\alpha)\right\|^2\right)\right]}{\exp(\lambda(1+\epsilon))} \\ &= \frac{\mathbb{E}\left[\exp\left(\lambda \sum_{i=1}^{k} q_i^2\right)\right]}{\exp(\lambda(1+\epsilon))} \\ &= \frac{\mathbb{E}\left[\prod_{i=1}^{k} \exp\left(\lambda q_i^2\right)\right]}{\exp(\lambda(1+\epsilon))} \\ &= \left(\frac{\mathbb{E}\left[\exp\left(\lambda q_i^2\right)\right]}{\exp\left(\frac{\lambda}{k}(1+\epsilon)\right)}\right)^k. \end{aligned} \tag{1.8}$$

The last equality is because the q_i's are i.i.d. Since α is a unit vector, from the previous lemma $q_i \sim \mathcal{N}(0, 1/k)$. Therefore, kq_i^2 is a χ^2 random variable with moment generating function

$$\mathbb{E}\left[\exp\left(\lambda q_i^2\right)\right] = \mathbb{E}\left[\exp\left(\frac{\lambda}{k} k q_i^2\right)\right] = \frac{1}{\sqrt{1-\frac{2\lambda}{k}}}.$$

Plugging this into (1.8)

$$Pr\left[\exp\left(\lambda \left\|f(\alpha)\right\|^2\right) > \exp\left(\lambda(1+\epsilon)\right)\right] \leq \left(\frac{\exp\left(-\frac{\lambda}{k}(1+\epsilon)\right)}{\sqrt{1-\frac{2\lambda}{k}}}\right)^k.$$

Setting $\lambda = \frac{k\epsilon}{2(1+\epsilon)}$ in the above inequality and simplifying

$$Pr\left[\exp\left(\lambda \left\|f(\alpha)\right\|^2\right) > \exp(\lambda(1+\epsilon))\right] \leq \left(\exp(-\epsilon)(1+\epsilon)\right)^{k/2}.$$

Using the inequality

$$\log(1+\epsilon) < \epsilon - \epsilon^2/2 + \epsilon^3/3$$

we can write

$$Pr\left[\exp\left(\lambda \left\|f(\alpha)\right\|^2\right) > \exp(\lambda(1+\epsilon))\right] \leq \exp\left(-\frac{k}{2}\left(\epsilon^2/2 - \epsilon^3/3\right)\right).$$

This proves (1.5). To prove (1.6) we need to repeat the above steps and use the inequality

$$\log(1-\epsilon) < -\epsilon - \epsilon^2/2.$$

This is left as an exercise to the reader. ■

A1.2 Spectral Properties of Matrices

A1.2.1 Basics

A1.2.2 Special Matrices

unitary, hermitean, positive semidefinite

A1.2.3 Normal Forms

Jacobi

A1.3 Functional Analysis

A1.3.1 Norms and Metrics

vector space, norm, triangle inequality

A1.3.2 Banach Spaces

normed vector space, evaluation functionals, examples, dual space

A1.3.3 Hilbert Spaces

symmetric inner product

A1.3.4 Operators

spectrum, norm, bounded, unbounded operators

A1.4 Fourier Analysis

A1.4.1 Basics

A1.4.2 Operators

Appendix 2

Conjugate Distributions

Binomial — Beta

$$\begin{aligned}\phi(x) &= x \\ e^{h(n\nu,n)} &= \frac{\Gamma(n\nu+1)\Gamma(n(1-\nu)+1)}{\Gamma(n+2)} = B(n\nu+1, n(1-\nu)+1)\end{aligned}$$

In traditional notation one represents the conjugate as

$$p(z;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1-z)^{\beta-1}$$

where $\alpha = n\nu + 1$ and $\beta = n(1-b\nu)+1$.

Multinomial — Dirichlet

$$\begin{aligned}\phi(x) &= e_x \\ e^{h(n\nu,n)} &= \frac{\prod_{i=1}^{d}\Gamma(n\nu_i+1)}{\Gamma(n+d)}\end{aligned}$$

In traditional notation one represents the conjugate as

$$p(z;\alpha) = \frac{\Gamma(\sum_{i=1}^{d}\alpha_i)}{\prod_{i=1}^{d}\Gamma(\alpha_i)} \prod_{i=1}^{d} z_i^{\alpha_i-1}$$

where $\alpha_i = n\nu_i + 1$

Poisson — Gamma

$$\begin{aligned}\phi(x) &= x \\ e^{h(n\nu,n)} &= n^{-n\nu}\Gamma(n\nu)\end{aligned}$$

In traditional notation one represents the conjugate as

$$p(z;\alpha) = \beta^{-\alpha}\Gamma(\alpha) z^{\alpha-1} e^{-\beta x}$$

where $\alpha = n\nu$ and $\beta = n$.

- Multinomial / Binomial
- Gaussian
- Laplace
- Poisson
- Dirichlet
- Wishart
- Student-t
- Beta
- Gamma

Appendix 3

Loss Functions

A3.1 Loss Functions

A multitude of loss functions are commonly used to derive seemingly different algorithms. This often blurs the similarities as well as subtle differences between them, often for historic reasons: Each new loss is typically accompanied by at least one publication dedicated to it. In many cases, the loss is not spelled out explicitly either but instead, it is only given by means of a constrained optimization problem. A case in point are the papers introducing (binary) hinge loss [BM92, CV95] and structured loss [TGK04, TJHA05]. Likewise, a geometric description obscures the underlying loss function, as in novelty detection [SPST+01].

In this section we give an expository yet unifying presentation of many of those loss functions. Many of them are well known, while others, such as multivariate ranking, hazard regression, or Poisson regression are not commonly used in machine learning. Tables A3.1 and A3.1 contain a choice subset of simple scalar and vectorial losses. Our aim is to put the multitude of loss functions in an unified framework, and to show how these losses and their (sub)gradients can be computed efficiently for use in our solver framework.

Note that not all losses, while convex, are continuously differentiable. In this situation we give *a* subgradient. While this may not be optimal, the convergence rates of our algorithm do not depend on which element of the subdifferential we provide: in all cases the first order Taylor approximation is a lower bound which is tight at the point of expansion.

In this setion, with little abuse of notation, v_i is understood as the i-th component of vector v when v is clearly not an element of a sequence or a set.

A3.1.1 Scalar Loss Functions

It is well known [Wah97] that the convex optimization problem

$$\min_{\xi} \ \xi \text{ subject to } y \langle w, x \rangle \geq 1 - \xi \text{ and } \xi \geq 0 \tag{3.1}$$

Logistic [CSS00]	$\log(1+\exp(-yf))$	$-y/(1+\exp(-yf))$
Novelty [SPST+01]	$\max(0, \rho - f)$	0 if $f \geq \rho$ and -1 otherwise
Least mean squares [Wil98]	$\frac{1}{2}(f-y)^2$	$f - y$
Least absolute deviation	$\lvert f-y \rvert$	$\operatorname{sign}(f-y)$
Quantile regression [Koe05]	$\max(\tau(f-y), (1-\tau)(y-f))$	τ if $f > y$ and $\tau - 1$ otherwise
ϵ-insensitive [VGS97]	$\max(0, \lvert f-y \rvert - \epsilon)$	0 if $\lvert f-y \rvert \leq \epsilon$, else $\operatorname{sign}(f-y)$
Huber's robust loss [MSR+97]	$\frac{1}{2}(f-y)^2$ if $\lvert f-y \rvert \leq 1$, else $\lvert f-y \rvert - \frac{1}{2}$	$f-y$ if $\lvert f-y \rvert \leq 1$, else $\operatorname{sign}(f-y)$
Poisson regression [Cre93]	$\exp(f) - yf$	$\exp(f) - y$

Vectorial loss functions and their derivatives, depending on the vector $f := Wx$ and on y.

	Loss	Derivative
Soft-Margin Multiclass [TGK04] [CS03]	$\max_{y'}(f_{y'} - f_y + \Delta(y, y'))$	$e_{y^*} - e_y$ where y^* is the argmax of the loss
Scaled Soft-Margin Multiclass [TJHA05]	$\max_{y'} \Gamma(y, y')(f_{y'} - f_y + \Delta(y, y'))$	$\Gamma(y, y')(e_{y^*} - e_y)$ where y^* is the argmax of the loss
Softmax Multiclass [CDLS99]	$\log \sum_{y'} \exp(f_{y'}) - f_y$	$\left[\sum_{y'} e_{y'} \exp(f'_y)\right] / \sum_{y'} \exp(f'_y) - e_y$
Multivariate Regression	$\frac{1}{2}(f-y)^\top M(f-y)$ where $M \succeq 0$	$M(f-y)$

takes on the value $\max(0, 1 - y \langle w, x \rangle)$. The latter is a convex function in w and x. Likewise, we may rewrite the ϵ-insensitive loss, Huber's robust loss, the quantile regression loss, and the novelty detection loss in terms of loss functions rather than a constrained optimization problem. In all cases, $\langle w, x \rangle$ will play a key role insofar as the loss is convex in terms of the *scalar* quantity $\langle w, x \rangle$. A large number of loss functions fall into this category, as described in Table A3.1. Note that not all functions of this type are continuously differentiable. In this case we adopt the convention that

$$\partial_x \max(f(x), g(x)) = \begin{cases} \partial_x f(x) & \text{if } f(x) \geq g(x) \\ \partial_x g(x) & \text{otherwise .} \end{cases} \tag{3.2}$$

Since we are only interested in obtaining an arbitrary element of the subdifferential this convention is consistent with our requirements.

Let us discuss the issue of efficient computation. For all scalar losses we may write $l(x, y, w) = \bar{l}(\langle w, x \rangle, y)$, as described in Table A3.1. In this case a simple application of the chain rule yields that $\partial_w l(x, y, w) = \bar{l}'(\langle w, x \rangle, y) \cdot x$. For instance, for squared loss we have

$$\bar{l}(\langle w, x \rangle, y) = \tfrac{1}{2}(\langle w, x \rangle - y)^2 \text{ and } \bar{l}'(\langle w, x \rangle, y) = \langle w, x \rangle - y.$$

Consequently, the derivative of the empirical risk term is given by

$$\partial_w R_{\text{emp}}(w) = \frac{1}{m} \sum_{i=1}^{m} \bar{l}'(\langle w, x_i \rangle, y_i) \cdot x_i. \tag{3.3}$$

This means that if we want to compute l and $\partial_w l$ on a large number of observations x_i, represented as matrix X, we can make use of fast linear algebra routines to pre-compute the vectors

$$f = Xw \text{ and } g^\top X \text{ where } g_i = \bar{l}'(f_i, y_i). \tag{3.4}$$

This is possible for any of the loss functions listed in Table A3.1, and many other similar losses. The advantage of this unified representation is that implementation of each individual loss can be done in very little time. The computational infrastructure for computing Xw and $g^\top X$ is shared. Evaluating $\bar{l}(f_i, y_i)$ and $\bar{l}'(f_i, y_i)$ for all i can be done in $O(m)$ time and it is not time-critical in comparison to the remaining operations. Algorithm 3.1 describes the details.

An important but often neglected issue is worth mentioning. Computing f requires us to *right* multiply the matrix X with the vector w while computing g requires the *left* multiplication of X with the vector $g^\top$. If X is stored in a row major format then Xw can be computed rather efficiently while $g^\top X$ is

Algorithm 3.1 ScalarLoss(w, X, y)

1: **input:** Weight vector w, feature matrix X, and labels y
2: Compute $f = Xw$
3: Compute $r = \sum_i \bar{l}(f_i, y_i)$ and $g = \bar{l}'(f, y)$
4: $g \leftarrow g^\top X$
5: **return** Risk r and gradient g

expensive. This is particularly true if X cannot fit in main memory. Converse is the case when X is stored in column major format. Similar problems are encountered when X is a sparse matrix and stored in either compressed row format or in compressed column format.

A3.1.2 Structured Loss

In recent years structured estimation has gained substantial popularity in machine learning [TJHA05, TGK04, BHS+07]. At its core it relies on two types of convex loss functions: logistic loss:

$$l(x, y, w) = \log \sum_{y' \in \mathcal{Y}} \exp\left(\langle w, \phi(x, y') \rangle\right) - \langle w, \phi(x, y) \rangle \,, \tag{3.5}$$

and soft-margin loss:

$$l(x, y, w) = \max_{y' \in \mathcal{Y}} \Gamma(y, y') \left\langle w, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'). \tag{3.6}$$

Here $\phi(x, y)$ is a *joint* feature map, $\Delta(y, y') \geq 0$ describes the cost of misclassifying y by y', and $\Gamma(y, y') \geq 0$ is a scaling term which indicates by how much the large margin property should be enforced. For instance, [TGK04] choose $\Gamma(y, y') = 1$. On the other hand [TJHA05] suggest $\Gamma(y, y') = \Delta(y, y')$, which reportedly yields better performance. Finally, [McA07] recently suggested generic functions $\Gamma(y, y')$.

The logistic loss can also be interpreted as the negative log-likelihood of a conditional exponential family model:

$$p(y|x; w) := \exp(\langle w, \phi(x, y) \rangle - g(w|x)), \tag{3.7}$$

where the normalizing constant $g(w|x)$, often called the log-partition function, reads

$$g(w|x) := \log \sum_{y' \in \mathcal{Y}} \exp\left(\langle w, \phi(x, y') \rangle\right). \tag{3.8}$$

As a consequence of the Hammersley-Clifford theorem [Jor08] every exponential family distribution corresponds to a undirected graphical model. In our case this implies that the labels y factorize according to an undirected graphical model. A large number of problems have been addressed by this setting, amongst them named entity tagging [LMP01], sequence alignment [TJHA05], segmentation [RSS+07] and path planning [RBZ06]. It is clearly impossible to give examples of all settings in this section, nor would a brief summary do this field any justice. We therefore refer the reader to the edited volume [BHS+07] and the references therein.

If the underlying graphical model is tractable then efficient inference algorithms based on dynamic programming can be used to compute (3.5) and (3.6). We discuss intractable graphical models in Section A3.1.2.1, and now turn our attention to the derivatives of the above structured losses.

When it comes to computing derivatives of the logistic loss, (3.5), we have

$$\partial_w l(x, y, w) = \frac{\sum_{y'} \phi(x, y') \exp \langle w, \phi(x, y') \rangle}{\sum_{y'} \exp \langle w, \phi(x, y') \rangle} - \phi(x, y) \tag{3.9}$$

$$= \mathbf{E}_{y' \sim p(y'|x)} \left[\phi(x, y') \right] - \phi(x, y). \tag{3.10}$$

where $p(y|x)$ is the exponential family model (3.7). In the case of (3.6) we denote by $\bar{y}(x)$ the argmax of the RHS, that is

$$\bar{y}(x) := \operatorname*{argmax}_{y'} \Gamma(y, y') \left\langle w, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'). \tag{3.11}$$

This allows us to compute the derivative of $l(x, y, w)$ as

$$\partial_w l(x, y, w) = \Gamma(y, \bar{y}(x)) \left[\phi(x, \bar{y}(x)) - \phi(x, y) \right]. \tag{3.12}$$

In the case where the loss is maximized for more than one distinct value $\bar{y}(x)$ we may average over the individual values, since any convex combination of such terms lies in the subdifferential.

Note that (3.6) majorizes $\Delta(y, y^*)$, where $y^* := \operatorname{argmax}_{y'} \langle w, \phi(x, y') \rangle$ [TJHA05]. This can be seen via the following series of inequalities:

$$\Delta(y, y^*) \leq \Gamma(y, y^*) \left\langle w, \phi(x, y^*) - \phi(x, y) \right\rangle + \Delta(y, y^*) \leq l(x, y, w).$$

The first inequality follows because $\Gamma(y, y^*) \geq 0$ and y^* maximizes $\langle w, \phi(x, y') \rangle$ thus implying that $\Gamma(y, y^*) \left\langle w, \phi(x, y^*) - \phi(x, y) \right\rangle \geq 0$. The second inequality follows by definition of the loss.

We conclude this section with a simple lemma which is at the heart of several derivations of [Joa05]. While the proof in the original paper is far from trivial, it is straightforward in our setting:

Lemma 3.1 *Denote by $\delta(y, y')$ a loss and let $\phi(x_i, y_i)$ be a feature map for observations (x_i, y_i) with $1 \leq i \leq m$. Moreover, denote by X, Y the set of all m patterns and labels respectively. Finally let*

$$\Phi(X, Y) := \sum_{i=1}^{m} \phi(x_i, y_i) \text{ and } \Delta(Y, Y') := \sum_{i=1}^{m} \delta(y_i, y_i'). \tag{3.13}$$

Then the following two losses are equivalent:

$$\sum_{i=1}^{m} \max_{y'} \left\langle w, \phi(x_i, y') - \phi(x_i, y_i) \right\rangle + \delta(y_i, y') \text{ and } \max_{Y'} \left\langle w, \Phi(X, Y') - \Phi(X, Y) \right\rangle + \Delta(Y, Y').$$

This is immediately obvious, since both feature map and loss decompose, which allows us to perform maximization over Y' by maximizing each of its m components. In doing so, we showed that aggregating all data and labels into a single feature map and loss yields results identical to minimizing the sum over all individual losses. This holds, in particular, for the sample error loss of [Joa05]. Also note that this equivalence does *not* hold whenever $\Gamma(y, y')$ is not constant.

A3.1.2.1 Intractable Models

We now discuss cases where computing $l(x, y, w)$ itself is too expensive. For instance, for intractable graphical models, the computation of $\sum_y \exp \langle w, \phi(x, y) \rangle$ cannot be computed efficiently. [WJ03] propose the use of a convex majorization of the log-partition function in those cases. In our setting this means that instead of dealing with

$$l(x, y, w) = g(w|x) - \langle w, \phi(x, y) \rangle \text{ where } g(w|x) := \log \sum_{y} \exp \langle w, \phi(x, y) \rangle \tag{3.14}$$

one uses a more easily computable convex upper bound on g via

$$\sup_{\mu \in \mathrm{MARG}(x)} \langle w, \mu \rangle + H_{\mathrm{Gauss}}(\mu|x). \tag{3.15}$$

Here $\mathrm{MARG}(x)$ is an outer bound on the conditional marginal polytope associated with the map $\phi(x, y)$. Moreover, $H_{\mathrm{Gauss}}(\mu|x)$ is an upper bound on the entropy by using a Gaussian with identical variance. More refined tree decompositions exist, too. The key benefit of our approach is that the solution μ of the optimization problem (3.15) can immediately be used as a gradient of the upper bound. This is computationally rather efficient.

Likewise note that [TGK04] use relaxations when solving structured estimation problems of the form

$$l(x, y, w) = \max_{y'} \Gamma(y, y') \left\langle w, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'), \tag{3.16}$$

by enlarging the domain of maximization with respect to y'. For instance, instead of an integer programming problem we might relax the setting to a linear program which is much cheaper to solve. This, again, provides an upper bound on the original loss function.

In summary, we have demonstrated that convex relaxation strategies are well applicable for bundle methods. In fact, the results of the corresponding optimization procedures can be used directly for further optimization steps.

A3.1.3 Scalar Multivariate Performance Scores

We now discuss a series of structured loss functions and how they can be implemented efficiently. For the sake of completeness, we give a concise representation of previous work on multivariate performance scores and ranking methods. All these loss functions rely on having access to $\langle w, x \rangle$, which can be computed efficiently by using the same operations as in Section A3.1.1.

A3.1.3.1 ROC Score

Denote by $f = Xw$ the vector of function values on the training set. It is well known that the area under the ROC curve is given by

$$\mathrm{AUC}(x, y, w) = \frac{1}{m_+ m_-} \sum_{y_i < y_j} \mathbf{I}(\langle w, x_i \rangle < \langle w, x_j \rangle), \tag{3.17}$$

where m_+ and m_- are the numbers of positive and negative observations respectively, and $\mathbf{I}(\cdot)$ is indicator function. Directly optimizing the cost $1 - \mathrm{AUC}(x, y, w)$ is difficult as it is not continuous in w. By using $\max(0, 1 + \langle w, x_i - x_j \rangle)$ as the surrogate loss function for all pairs (i, j) for which $y_i < y_j$ we have the following convex multivariate empirical risk

$$R_{\mathrm{emp}}(w) = \frac{1}{m_+ m_-} \sum_{y_i < y_j} \max(0, 1 + \langle w, x_i - x_j \rangle) = \frac{1}{m_+ m_-} \sum_{y_i < y_j} \max(0, 1 + f_i - f_j). \tag{3.18}$$

Obviously, we could compute $R_{\mathrm{emp}}(w)$ and its derivative by an $O(m^2)$ operation. However [Joa05] showed that both can be computed in $O(m \log m)$ time using a sorting operation, which we now describe.

Denote by $c = f - \frac{1}{2} y$ an auxiliary variable and let i and j be indices such

Algorithm 3.2 ROCScore(X, y, w)

input: Feature matrix X, labels y, and weight vector w
initialization: $s_- = m_-$ and $s_+ = 0$ and $l = \mathbf{0}_m$ and $c = Xw - \frac{1}{2}y$
$\pi \leftarrow \{1, \ldots, m\}$ sorted in ascending order of c
for $i = 1$ **to** m **do**
  if $y_{\pi_i} = -1$ **then**
    $l_{\pi_i} \leftarrow s_+$ and $s_- \leftarrow s_- - 1$
  else
    $l_{\pi_i} \leftarrow -s_-$ and $s_+ \leftarrow s_+ + 1$
  end if
end for
Rescale $l \leftarrow l/(m_+ m_-)$ and compute $r = \langle l, c \rangle$ and $g = l^\top X$.
return Risk r and subgradient g

that $y_i = -1$ and $y_j = 1$. It follows that $c_i - c_j = 1 + f_i - f_j$. The efficient algorithm is due to the observation that there are at most m distinct terms $c_k,\ k = 1, \ldots, m$, each with different frequency l_k and sign, appear in (3.18). These frequencies l_k can be determined by first sorting c in ascending order then scanning through the labels according to the sorted order of c and keeping running statistics such as the number s_- of negative labels yet to encounter, and the number s_+ of positive labels encountered. When visiting y_k, we know c_k should appears s_+ (or s_-) times with positive (or negative) sign in (3.18) if $y_k = -1$ (or $y_k = 1$). Algorithm 3.2 spells out explicitly how to compute $R_{\text{emp}}(w)$ and its subgradient.

A3.1.3.2 Ordinal Regression

Essentially the same preference relationships need to hold for ordinal regression. The only difference is that y_i need not take on binary values any more. Instead, we may have an arbitrary number of different values y_i (*e.g.,* 1 corresponding to 'strong reject' up to 10 corresponding to 'strong accept', when it comes to ranking papers for a conference). That is, we now have $y_i \in \{1, \ldots, n\}$ rather than $y_i \in \{\pm 1\}$. Our goal is to find some w such that $\langle w, x_i - x_j \rangle < 0$ whenever $y_i < y_j$. Whenever this relationship is not satisfied, we incur a cost $C(y_i, y_j)$ for preferring x_i to x_j. For examples, $C(y_i, y_j)$ could be constant *i.e.,* $C(y_i, y_j) = 1$ [Joa06] or linear *i.e.,* $C(y_i, y_j) = y_j - y_i$.

Denote by m_i the number of x_j for which $y_j = i$. In this case, there are $\bar{M} = m^2 - \sum_{i=1}^n m_i^2$ pairs (y_i, y_j) for which $y_i \neq y_j$; this implies that there are $M = \bar{M}/2$ pairs (y_i, y_j) such that $y_i < y_j$. Normalizing by the total

number of comparisons we may write the overall cost of the estimator as

$$\frac{1}{M}\sum_{y_i<y_j} C(y_i,y_j)\mathbf{I}(\langle w,x_i\rangle > \langle w,x_j\rangle) \text{ where } M=\frac{1}{2}\left[m^2-\sum_i^n m_i^2\right]. \quad (3.19)$$

Using the same convex majorization as above when we were maximizing the ROC score, we obtain an empirical risk of the form

$$R_{\text{emp}}(w)=\frac{1}{M}\sum_{y_i<y_j} C(y_i,y_j)\max(0,1+\langle w,x_i-x_j\rangle) \quad (3.20)$$

Now the goal is to find an efficient algorithm for obtaining the number of times when the individual losses are nonzero such as to compute both the value and the gradient of $R_{\text{emp}}(w)$. The complication arises from the fact that observations x_i with label y_i may appear in either side of the inequality depending on whether $y_j < y_i$ or $y_j > y_i$. This problem can be solved as follows: sort $f = Xw$ in ascending order and traverse it while keeping track of how many items with a lower value y_j are no more than 1 apart in terms of their value of f_i. This way we may compute the count statistics efficiently. Algorithm 3.3 describes the details, generalizing the results of [Joa06]. Again, its runtime is $O(m \log m)$, thus allowing for efficient computation.

A3.1.3.3 Preference Relations

In general, our loss may be described by means of a set of preference relations $j \succeq i$ for arbitrary pairs $(i,j) \in \{1,\dots m\}^2$ associated with a cost $C(i,j)$ which is incurred whenever i is ranked above j. This set of preferences may or may not form a partial or a total order on the domain of all observations. In these cases efficient computations along the lines of Algorithm 3.3 exist. In general, this is not the case and we need to rely on the fact that the set P containing all preferences is sufficiently small that it can be enumerated efficiently. The risk is then given by

$$\frac{1}{|P|}\sum_{(i,j)\in P} C(i,j)\mathbf{I}(\langle w,x_i\rangle > \langle w,x_j\rangle) \quad (3.21)$$

Algorithm 3.3 OrdinalRegression(X, y, w, C)

input: Feature matrix X, labels y, weight vector w, and score matrix C
initialization: $l = \mathbf{0}_n$ and $u_i = m_i \; \forall i \in [n]$ and $r = 0$ and $g = \mathbf{0}_m$
Compute $f = Xw$ and set $c = [f - \frac{1}{2}, f + \frac{1}{2}] \in \mathbb{R}^{2m}$ (concatenate the vectors)
Compute $M = (m^2 - \sum_{i=1}^{n} m_i^2)/2$
Rescale $C \leftarrow C/M$
$\pi \leftarrow \{1, \ldots, 2m\}$ sorted in ascending order of c
for $i = 1$ **to** $2m$ **do**
  $j = \pi_i \bmod m$
  if $\pi_i \leq m$ **then**
    for $k = 1$ **to** $y_j - 1$ **do**
      $r \leftarrow r - C(k, y_j) u_k c_j$
      $g_j \leftarrow g_j - C(k, y_j) u_k$
    end for
    $l_{y_j} \leftarrow l_{y_j} + 1$
  else
    for $k = y_j + 1$ **to** n **do**
      $r \leftarrow r + C(y_j, k) l_k c_{j+m}$
      $g_j \leftarrow g_j + C(y_j, k) l_k$
    end for
    $u_{y_j} \leftarrow u_{y_j} - 1$
  end if
end for
$g \leftarrow g^\top X$
return: Risk r and subgradient g

Again, the same majorization argument as before allows us to write a convex upper bound

$$R_{\text{emp}}(w) = \frac{1}{|P|} \sum_{(i,j) \in P} C(i, j) \max\left(0, 1 + \langle w, x_i \rangle - \langle w, x_j \rangle\right) \tag{3.22}$$

$$\text{where } \partial_w R_{\text{emp}}(w) = \frac{1}{|P|} \sum_{(i,j) \in P} C(i, j) \begin{cases} 0 & \text{if } \langle w, x_j - x_i \rangle \geq 1 \\ x_i - x_j & \text{otherwise} \end{cases} \tag{3.23}$$

The implementation is straightforward, as given in Algorithm 3.4.

Algorithm 3.4 Preference(X, w, C, P)

input: Feature matrix X, weight vector w, score matrix C, and preference set P
initialization: $r = 0$ and $g = \mathbf{0}_m$
Compute $f = Xw$
while $(i, j) \in P$ **do**
  if $f_j - f_i < 1$ **then**
    $r \leftarrow r + C(i, j)(1 + f_i - f_j)$
    $g_i \leftarrow g_i + C(i, j)$ and $g_j \leftarrow g_j - C(i, j)$
  end if
end while
$g \leftarrow g^\top X$
return Risk r and subgradient g

A3.1.3.4 Ranking

In webpage and document ranking we are often in a situation similar to that described in Section A3.1.3.2, however with the difference that we do not only care about objects x_i being ranked according to scores y_i but moreover that different degrees of importance are placed on different documents.

The information retrieval literature is full with a large number of different scoring functions. Examples are criteria such as *Normalized Discounted Cumulative Gain (NDCG)*, *Mean Reciprocal Rank (MRR)*, *Precision@n*, or *Expected Rank Utility (ERU)*. They are used to address the issue of evaluating rankers, search engines or recommender sytems [Voo01, JK02, BHK98, BH04]. For instance, in webpage ranking only the first k retrieved documents that matter, since users are unlikely to look beyond the first k, say 10, retrieved webpages in an internet search. [LS07] show that these scores can be optimized directly by minimizing the following loss:

$$l(X, y, w) = \max_{\pi} \sum_i c_i \left\langle w, x_{\pi(i)} - x_i \right\rangle + \left\langle a - a(\pi), b(y) \right\rangle . \tag{3.24}$$

Here c_i is a monotonically decreasing sequence, the documents are assumed to be arranged in order of decreasing relevance, π is a permutation, the vectors a and $b(y)$ depend on the choice of a particular ranking measure, and $a(\pi)$ denotes the permutation of a according to π. Pre-computing $f = Xw$ we may rewrite (3.24) as

$$l(f, y) = \max_{\pi} \left[c^\top f(\pi) - a(\pi)^\top b(y) \right] - c^\top f + a^\top b(y) \tag{3.25}$$

Algorithm 3.5 Ranking(X, y, w)

input: Feature matrix X, relevances y, and weight vector w
Compute vectors a and $b(y)$ according to some ranking measure
Compute $f = Xw$
Compute elements of matrix $C_{ij} = c_i f_j - b_i a_j$
$\pi = \text{LinearAssignment}(C)$
$r = c^\top (f(\pi) - f) + (a - a(\pi))^\top b$
$g = c(\pi^{-1}) - c$ and $g \leftarrow g^\top X$
return Risk r and subgradient g

and consequently the derivative of $l(X, y, w)$ with respect to w is given by

$$\partial_w l(X, y, w) = (c(\bar{\pi}^{-1}) - c)^\top X \text{ where } \bar{\pi} = \operatorname*{argmax}_\pi c^\top f(\pi) - a(\pi)^\top b(y). \tag{3.26}$$

Here π^{-1} denotes the inverse permutation, such that $\pi \circ \pi^{-1} = 1$. Finding the permutation maximizing $c^\top f(\pi) - a(\pi)^\top b(y)$ is a linear assignment problem which can be easily solved by the Hungarian Marriage algorithm, that is, the Kuhn-Munkres algorithm.

The original papers by [Kuh55] and [Mun57] implied an algorithm with $O(m^3)$ cost in the number of terms. Later, [Kar80] suggested an algorithm with expected quadratic time in the size of the assignment problem (ignoring log-factors). Finally, [OL93] propose a linear time algorithm for large problems. Since in our case the number of pages is fairly small (in the order of 50 to 200 *per query*) the scaling behavior per query is not too important. We used an existing implementation due to [JV87].

Note also that training sets consist of a *collection* of ranking problems, that is, we have several ranking problems of size 50 to 200. By means of parallelization we are able to distribute the work onto a cluster of workstations, which is able to overcome the issue of the rather costly computation per collection of queries. Algorithm 3.5 spells out the steps in detail.

A3.1.3.5 Contingency Table Scores

[Joa05] observed that F_β scores and related quantities dependent on a contingency table can also be computed efficiently by means of structured estimation. Such scores depend in general on the number of true and false positives and negatives alike. Algorithm 3.6 shows how a corresponding empirical risk and subgradient can be computed efficiently. As with the previous losses, here again we use convex majorization to obtain a tractable optimization problem.

Given a set of labels y and an estimate y', the numbers of true positives (T_+), true negatives (T_-), false positives (F_+), and false negatives (F_-) are determined according to a contingency table as follows:

	$y > 0$	$y < 0$
$y' > 0$	T_+	F_+
$y' < 0$	F_-	T_-

In the sequel, we denote by $m_+ = T_+ + F_-$ and $m_- = T_- + F_+$ the numbers of positives and negative labels in y, respectively. We note that F_β score can be computed based on the contingency table [Joa05] as

$$F_\beta(T_+, T_-) = \frac{(1+\beta^2)T_+}{T_+ + m_- - T_- + \beta^2 m_+}. \tag{3.27}$$

If we want to use $\langle w, x_i \rangle$ to estimate the label of observation x_i, we may use the following structured loss to "directly" optimize w.r.t. F_β score [Joa05]:

$$l(X, y, w) = \max_{y'} \left[(y' - y)^\top f + \Delta(T_+, T_-) \right], \tag{3.28}$$

where $f = Xw$, $\Delta(T_+, T_-) := 1 - F_\beta(T_+, T_-)$, and (T_+, T_-) is determined by using y and y'. Since Δ does not depend on the specific choice of (y, y') but rather just on which sets they disagree, l can be maximized as follows: Enumerating all possible $m_+ m_-$ contingency tables in a way such that given a configuration (T_+, T_-), T_+ (T_-) positive (negative) observations x_i with largest (lowest) value of $\langle w, x_i \rangle$ are labeled as positive (negative). This is effectively implemented as a nested loop hence run in $O(m^2)$ time. Algorithm 3.6 describes the procedure in details.

A3.1.4 Vector Loss Functions

Next we discuss "vector" loss functions, *i.e.,* functions where w is best described as a matrix (denoted by W) and the loss depends on Wx. Here, we have feature vector $x \in \mathbb{R}^d$, label $y \in \mathbb{R}^k$, and weight matrix $W \in \mathbb{R}^{d \times k}$. We also denote feature matrix $X \in \mathbb{R}^{m \times d}$ as a matrix of m feature vectors x_i, and stack up the columns W_i of W as a vector w.

Some of the most relevant cases are multiclass classification using both the exponential families model and structured estimation, hierarchical models, *i.e.,* ontologies, and multivariate regression. Many of those cases are summarized in Table A3.1.

Algorithm 3.6 $\mathrm{F}_\beta(X, y, w)$

input: Feature matrix X, labels y, and weight vector w
Compute $f = Xw$
$\pi^+ \leftarrow \{i : y_i = 1\}$ sorted in descending order of f
$\pi^- \leftarrow \{i : y_i = -1\}$ sorted in ascending order of f
Let $p_0 = 0$ and $p_i = 2\sum_{k=i}^{m_+} f_{\pi_k^+},\ i = 1, \ldots, m_+$
Let $n_0 = 0$ and $n_i = 2\sum_{k=i}^{m_-} f_{\pi_k^-},\ i = 1, \ldots, m_-$
$y' \leftarrow -y$ and $r \leftarrow -\infty$
for $i = 0$ **to** m_+ **do**
  for $j = 0$ **to** m_- **do**
    $r_{\text{tmp}} = \Delta(i, j) - p_i + n_j$
    if $r_{\text{tmp}} > r$ **then**
      $r \leftarrow r_{\text{tmp}}$
      $T_+ \leftarrow i$ and $T_- \leftarrow j$
    end if
  end for
end for
$y'_{\pi_i^+} \leftarrow 1,\ i = 1, \ldots, T_+$
$y'_{\pi_i^-} \leftarrow -1,\ i = 1, \ldots, T_-$
$g \leftarrow (y' - y)^\top X$
return Risk r and subgradient g

A3.1.4.1 Unstructured Setting

The simplest loss is multivariate regression, where $l(x, y, W) = \frac{1}{2}(y - x^\top W)^\top M(y - x^\top W)$. In this case it is clear that by pre-computing XW subsequent calculations of the loss and its gradient are significantly accelerated.

A second class of important losses is given by plain multiclass classification problems, *e.g.,* recognizing digits of a postal code or categorizing high-level document categories. In this case, $\phi(x, y)$ is best represented by $e_y \otimes x$ (using a linear model). Clearly we may view $\langle w, \phi(x, y)\rangle$ as an operation which chooses a column indexed by y from xW, since all labels y correspond to a different weight vector W_y. Formally we set $\langle w, \phi(x, y)\rangle = [xW]_y$. In this case, structured estimation losses can be rewritten as

$$l(x, y, W) = \max_{y'} \Gamma(y, y') \left\langle W_{y'} - W_y, x \right\rangle + \Delta(y, y') \tag{3.29}$$

$$\text{and } \partial_W l(x, y, W) = \Gamma(y, y^*)(e_{y^*} - e_y) \otimes x. \tag{3.30}$$

Here Γ and Δ are defined as in Section A3.1.2 and y^* denotes the value of y'

for which the RHS of (3.29) is maximized. This means that for unstructured multiclass settings we may simply compute xW. Since this needs to be performed for all observations x_i we may take advantage of fast linear algebra routines and compute $f = XW$ for efficiency. Likewise note that computing the gradient over m observations is now a matrix-matrix multiplication, too: denote by G the matrix of rows of gradients $\Gamma(y_i, y_i^*)(e_{y_i^*} - e_{y_i})$. Then $\partial_W R_{\text{emp}}(X, y, W) = G^\top X$. Note that G is very sparse with at most two nonzero entries per row, which makes the computation of $G^\top X$ essentially as expensive as two matrix vector multiplications. Whenever we have many classes, this may yield significant computational gains.

Log-likelihood scores of exponential families share similar expansions. We have

$$l(x, y, W) = \log \sum_{y'} \exp \langle w, \phi(x, y') \rangle - \langle w, \phi(x, y) \rangle = \log \sum_{y'} \exp \langle W_{y'}, x \rangle - \langle W_y, x \rangle \tag{3.31}$$

$$\partial_W l(x, y, W) = \frac{\sum_{y'} (e_{y'} \otimes x) \exp \langle W_{y'}, x \rangle}{\sum_{y'} \exp \langle W_{y'}, x \rangle} - e_y \otimes x. \tag{3.32}$$

The main difference to the soft-margin setting is that the gradients are *not* sparse in the number of classes. This means that the computation of gradients is slightly more costly.

A3.1.4.2 Ontologies

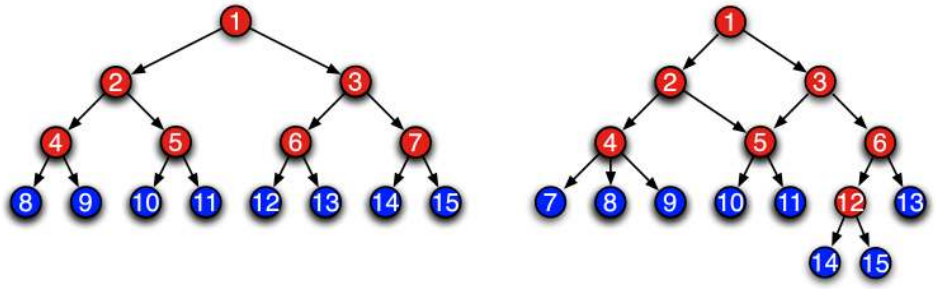


Fig. A3.1. Two ontologies. **Left:** a binary hierarchy with internal nodes $\{1, \ldots, 7\}$ and labels $\{8, \ldots 15\}$. **Right:** a generic directed acyclic graph with internal nodes $\{1, \ldots, 6, 12\}$ and labels $\{7, \ldots, 11, 13, \ldots, 15\}$. Note that node 5 has two parents, namely nodes 2 and 3. Moreover, the labels need not be found at the same level of the tree: nodes 14 and 15 are one level lower than the rest of the nodes.

Assume that the labels we want to estimate can be found to belong to a directed acyclic graph. For instance, this may be a gene-ontology graph

[ABB+00] a patent hierarchy [CH04], or a genealogy. In these cases we have a hierarchy of categories to which an element x may belong. Figure A3.1 gives two examples of such directed acyclic graphs (DAG). The first example is a binary tree, while the second contains nodes with different numbers of children (*e.g.,* node 4 and 12), nodes at different levels having children (*e.g.,* nodes 5 and 12), and nodes which have more than one parent (*e.g.,* node 5). It is a well known fundamental property of trees that they have at most as many internal nodes as they have leaf nodes.

It is now our goal to build a classifier which is able to categorize observations according to which leaf node they belong to (each leaf node is assigned a label y). Denote by $k+1$ the number of nodes in the DAG including the root node. In this case we may design a feature map $\phi(y) \in \mathbb{R}^k$ [CH04] by associating with every label y the vector describing the path from the root node to y, ignoring the root node itself. For instance, for the first DAG in Figure A3.1 we have

$$\phi(8) = (1,0,1,0,0,0,1,0,0,0,0,0,0,0) \text{ and } \phi(13) = (0,1,0,0,1,0,0,0,0,0,0,1,0,0)$$

Whenever several paths are admissible, as in the right DAG of Figure A3.1 we average over all possible paths. For example, we have

$$\phi(10) = (0.5,0.5,0,1,0,0,0,0,1,0,0,0,0,0) \text{ and } \phi(15) = (0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,1).$$

Also note that the lengths of the paths need not be the same (*e.g.,* to reach 15 it takes a longer path than to reach 13). Likewise, it is natural to assume that $\Delta(y,y')$, *i.e.,* the cost for mislabeling y as y' will depend on the similarity of the path. In other words, it is likely that the cost for placing x into the wrong sub-sub-category is less than getting the main category of the object wrong.

To complete the setting, note that for $\phi(x,y) = \phi(y) \otimes x$ the cost of computing all labels is k inner products, since the value of $\langle w, \phi(x,y)\rangle$ for a particular y can be obtained by the sum of the contributions for the segments of the path. This means that the values for *all* terms can be computed by a simple breadth first traversal through the graph. As before, we may make use of vectorization in our approach, since we may compute $xW \in \mathbb{R}^k$ to obtain the contributions on all segments of the DAG before performing the graph traversal. Since we have m patterns x_i we may vectorize matters by pre-computing XW.

Also note that $\phi(y) - \phi(y')$ is nonzero only for those edges where the paths for y and y' differ. Hence we only change weights on those parts of the graph where the categorization differs. Algorithm 3.7 describes the subgradient and loss computation for the soft-margin type of loss function.

Algorithm 3.7 Ontology(X, y, W)

input: Feature matrix $X \in \mathbb{R}^{m \times d}$, labels y, and weight matrix $W \in \mathbb{R}^{d \times k}$
initialization: $G = \mathbf{0} \in \mathbb{R}^{m \times k}$ and $r = 0$
Compute $f = XW$ and let $f_i = x_i W$
for $i = 1$ **to** m **do**
  Let D_i be the DAG with edges annotated with the values of f_i
  Traverse D_i to find node y^* that maximize sum of f_i values on the path plus $\Delta(y_i, y')$
  $G_i = \phi(y^*) - \phi(y_i)$
  $r \leftarrow r + z_{y^*} - z_{y_i}$
end for
$g = G^\top X$
return Risk r and subgradient g

The same reasoning applies to estimation when using an exponential families model. The only difference is that we need to compute a *soft-max* over paths rather than exclusively choosing the best path over the ontology. Again, a breadth-first recursion suffices: each of the leaves y of the DAG is associated with a probability $p(y|x)$. To obtain $\mathbf{E}_{y \sim p(y|x)} [\phi(y)]$ all we need to do is perform a bottom-up traversal of the DAG summing over all probability weights on the path. Wherever a node has more than one parent, we distribute the probability weight equally over its parents.

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.

[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.

[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.

[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.

[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.

[BT03] 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.

[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.

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

[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.

[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.

[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.

[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.

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

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

[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 30^{th} Symposium on Theory of Computing, 1998, pp. 604–613.

[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.

[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.

[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.

[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.

[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] Q. Le and A.J. Smola, *Direct optimization of ranking measures*, J. Mach. Learn. Res. (2007), submitted.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. 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.

[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.

[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.