https://github.com/soulmachine/machine-learning-cheat-sheet
soulmachine@gmail.com

Machine Learning Cheat Sheet

Classical equations, diagrams and tricks in machine learning

December 1, 2022

Preface

This cheat sheet is a condensed version of machine learning manual, which contains many classical equations and diagrams on machine learning, and aims to help you quickly recall knowledge and ideas in machine learning.

This cheat sheet has two significant advantages:

1. Clearer symbols. Mathematical formulas use quite a lot of confusing symbols. For example, X can be a set, a random variable, or a matrix. This is very confusing and makes it very difficult for readers to understand the meaning of math formulas. This cheat sheet tries to standardize the usage of symbols, and all symbols are clearly pre-defined, see section §.
2. Less thinking jumps. In many machine learning books, authors omit some intermediary steps of a mathematical proof process, which may save some space but causes difficulty for readers to understand this formula and readers get lost in the middle way of the derivation process. This cheat sheet tries to keep important intermediary steps as where as possible.

Contents

List of Contributors

Wei Zhang
PhD candidate at the Institute of Software, Chinese Academy of Sciences (ISCAS), Beijing, P.R.CHINA, e-mail: zh3feng@gmail.com, has written chapters of Naive Bayes and SVM.

Fei Pan
Master at Beijing University of Technology, Beijing, P.R.CHINA, e-mail: example@gmail.com, has written chapters of KMeans, AdaBoost.

Yong Li
PhD candidate at the Institute of Automation of the Chinese Academy of Sciences (CASIA), Beijing, P.R.CHINA, e-mail: liyong3forever@gmail.com, has written chapters of Logistic Regression.

Jiankou Li
PhD candidate at the Institute of Software, Chinese Academy of Sciences (ISCAS), Beijing, P.R.CHINA, e-mail: lijiankoucoco@163.com, has written chapters of BayesNet.

Notation

Introduction

It is very difficult to come up with a single, consistent notation to cover the wide variety of data, models and algorithms that we discuss. Furthermore, conventions difer between machine learning and statistics, and between different books and papers. Nevertheless, we have tried to be as consistent as possible. Below we summarize most of the notation used in this book, although individual sections may introduce new notation. Note also that the same symbol may have different meanings depending on the context, although we try to avoid this where possible.

General math notation

Symbol	Meaning
$\lfloor x \rfloor$	Floor of x, i.e., round down to nearest integer
$\lceil x \rceil$	Ceiling of x, i.e., round up to nearest integer
$\vec{x} \otimes \vec{y}$	Convolution of $\vec{x}$ and $\vec{y}$
$\vec{x} \odot \vec{y}$	Hadamard (elementwise) product of $\vec{x}$ and $\vec{y}$
$a \wedge b$	logical AND
$a \vee b$	logical OR
$\neg a$	logical NOT
$\mathbb{I}(x)$	Indicator function, $\mathbb{I}(x) = 1$ if x is true, else $\mathbb{I}(x) = 0$
∞	Infinity
$\rightarrow$	Tends towards, e.g., $n \rightarrow \infty$
$\propto$	Proportional to, so $y = ax$ can be written as $y \propto x$
$\|x\|$	Absolute value
$\|\mathcal{S}\|$	Size (cardinality) of a set
$n!$	Factorial function
∇	Vector of first derivatives
∇^2	Hessian matrix of second derivatives
$\triangleq$	Defined as
$O(\cdot)$	Big-O: roughly means order of magnitude
$\mathbb{R}$	The real numbers
$1:n$	Range (Matlab convention): $1:n = 1, 2, ..., n$
$\approx$	Approximately equal to
$\arg\max_x f(x)$	Argmax: the value x that maximizes f
$B(a,b)$	Beta function, $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$
$B(\vec{\alpha})$	Multivariate beta function, $\frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)}$
$\binom{n}{k}$	n choose k , equal to $n!/(k!(n-k)!)$
$\delta(x)$	Dirac delta function,$\delta(x) = \infty$ if $x = 0$, else $\delta(x) = 0$
$\exp(x)$	Exponential function e^x
$\Gamma(x)$	Gamma function, $\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} \mathrm{d}u$
$\Psi(x)$	Digamma function,$Psi(x) = \frac{d}{dx} \log \Gamma(x)$
$\mathcal{X}$	A set from which values are drawn (e.g.,$\mathcal{X} = \mathbb{R}^D$)

Linear algebra notation

We use boldface lower-case to denote vectors, such as $\vec{x}$, and boldface upper-case to denote matrices, such as $\vec{X}$. We denote entries in a matrix by non-bold upper case letters, such as X_{ij}.

Vectors are assumed to be column vectors, unless noted otherwise. We use $(x_1, \cdots, x_D)$ to denote a column vector created by stacking D scalars. If we write $\vec{X} = (\vec{x}_1, \cdots, \vec{x}_n)$, where the left hand side is a matrix, we mean to stack the $\vec{x}_i$ along the columns, creating a matrix.

Symbol	Meaning
$\vec{X} \succ 0$	$\vec{X}$ is a positive definite matrix
$tr(\vec{X})$	Trace of a matrix
$det(\vec{X})$	Determinant of matrix $\vec{X}$
$\left\|\vec{X}\right\|$	Determinant of matrix $\vec{X}$
$\vec{X}^{-1}$	Inverse of a matrix
$\vec{X}^{\dagger}$	Pseudo-inverse of a matrix
$\vec{X}^T$	Transpose of a matrix
$\vec{x}^T$	Transpose of a vector
$\text{diag}(x)$	Diagonal matrix made from vector $\vec{x}$
$\text{diag}(X)$	Diagonal vector extracted from matrix $\vec{X}$
$\vec{I}$ or $\vec{I}_d$	Identity matrix of size $d \times d$ (ones on diagonal, zeros of)
$\vec{1}$ or $\vec{1}_d$	Vector of ones (of length d)
$\vec{0}$ or $\vec{0}_d$	Vector of zeros (of length d)
$\|\|\vec{x}\|\| = \|\|\vec{x}\|\|_2$	Euclidean or ℓ_2 norm $\sqrt{\sum_{j=1}^{d} x_j^2}$
$\|\|\vec{x}\|\|_1$	ℓ_1 norm $\sum_{j=1}^{d} \left\|x_j\right\|$
$\vec{X}_{:,j}$	j'th column of matrix
$\vec{X}_{i,:}$	transpose of i'th row of matrix (a column vector)
$\vec{X}_{i,j}$	Element (i, j) of matrix $\vec{X}$
$\vec{x} \otimes \vec{y}$	Tensor product of $\vec{x}$ and $\vec{y}$

Probability notation

We denote random and fixed scalars by lower case, random and fixed vectors by bold lower case, and random and fixed matrices by bold upper case. Occasionally we use non-bold upper case to denote scalar random variables. Also, we use $p()$ for both discrete and continuous random variables

Symbol	Meaning
X, Y	Random variable
$P()$	Probability of a random event
$F()$	Cumulative distribution function(CDF), also called distribution function
$p(x)$	Probability mass function(PMF)
$f(x)$	probability density function(PDF)
$F(x, y)$	Joint CDF
$p(x, y)$	Joint PMF
$f(x, y)$	Joint PDF
$p(X\|Y)$	Conditional PMF, also called conditional probability
$f_{X\|Y}(x\|y)$	Conditional PDF
$X \perp Y$	X is independent of Y
$X \not\perp Y$	X is not independent of Y
$X \perp Y\|Z$	X is conditionally independent of Y given Z
$X \not\perp Y\|Z$	X is not conditionally independent of Y given Z
$X \sim p$	X is distributed according to distribution p
$\vec{\alpha}$	Parameters of a Beta or Dirichlet distribution

$\text{cov}[X]$	Covariance of X
$\mathbb{E}[X]$	Expected value of X
$\mathbb{E}_q[X]$	Expected value of X wrt distribution q
$\mathbb{H}(X)$ or $\mathbb{H}(p)$	Entropy of distribution $p(X)$
$\mathbb{I}(X;Y)$	Mutual information between X and Y
$\mathbb{KL}(p\|\|q)$	KL divergence from distribution p to q
$\ell(\vec{\theta})$	Log-likelihood function
$L(\theta,a)$	Loss function for taking action a when true state of nature is θ
λ	Precision (inverse variance) $\lambda = 1/\sigma^2$
Λ	Precision matrix $\Lambda = \Sigma^{-1}$
$\text{mode}[\vec{X}]$	Most probable value of $\vec{X}$
μ	Mean of a scalar distribution
$\vec{\mu}$	Mean of a multivariate distribution
Φ	cdf of standard normal
ϕ	pdf of standard normal
$\vec{\pi}$	multinomial parameter vector, Stationary distribution of Markov chain
ρ	Correlation coefficient
$\text{sigm}(x)$	Sigmoid (logistic) function, $\frac{1}{1+e^{-x}}$
σ^2	Variance
Σ	Covariance matrix
$\text{var}[x]$	Variance of x
ν	Degrees of freedom parameter
Z	Normalization constant of a probability distribution

Machine learning/statistics notation

In general, we use upper case letters to denote constants, such as C, K, M, N, T, etc. We use lower case letters as dummy indexes of the appropriate range, such as $c = 1 : C$ to index classes, $i = 1 : M$ to index data cases, $j = 1 : N$ to index input features, $k = 1 : K$ to index states or clusters, $t = 1 : T$ to index time, etc.

We use x to represent an observed data vector. In a supervised problem, we use y or $\vec{y}$ to represent the desired output label. We use $\vec{z}$ to represent a hidden variable. Sometimes we also use q to represent a hidden discrete variable.

Symbol	**Meaning**
C	Number of classes
D	Dimensionality of data vector (number of features)
N	Number of data cases
N_c	Number of examples of class c,$N_c = \sum_{i=1}^{N} \mathbb{I}(y_i = c)$
R	Number of outputs (response variables)
$\mathcal{D}$	Training data $\mathcal{D} = \{(\vec{x}_i, y_i) \| i = 1 : N\}$
$\mathcal{D}_{test}$	Test data
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
K	Number of states or dimensions of a variable (often latent)
$k(x,y)$	Kernel function
$\vec{K}$	Kernel matrix
$\mathcal{H}$	Hypothesis space
L	Loss function
$J(\vec{\theta})$	Cost function
$f(\vec{x})$	Decision function
$P(y\|\vec{x})$	Conditional probability
λ	Strength of ℓ_2 or ℓ_1 *regularizer*
$\phi(x)$	Basis function expansion of feature vector $\vec{x}$
Φ	Basis function expansion of design matrix $\vec{X}$
$q()$	Approximate or proposal distribution
$Q(\vec{\theta}, \vec{\theta}_{old})$	Auxiliary function in EM
T	Length of a sequence

$T(\mathcal{D})$	Test statistic for data
$\vec{T}$	Transition matrix of Markov chain
$\vec{\theta}$	Parameter vector
$\vec{\theta}^{(s)}$	s'th sample of parameter vector
$\hat{\vec{\theta}}$	Estimate (usually MLE or MAP) of $\vec{\theta}$
$\hat{\vec{\theta}}_{MLE}$	Maximum likelihood estimate of $\vec{\theta}$
$\hat{\vec{\theta}}_{MAP}$	MAP estimate of $\vec{\theta}$
$\bar{\vec{\theta}}$	Estimate (usually posterior mean) of $\vec{\theta}$
$\vec{w}$	Vector of regression weights (called $\vec{\beta}$ in statistics)
b	intercept (called ε in statistics)
$\vec{W}$	Matrix of regression weights
x_{ij}	Component (i.e., feature) j of data case i ,for $i = 1:N, j = 1:D$
$\vec{x}_i$	Training case, $i = 1:N$
$\vec{X}$	Design matrix of size $N \times D$
$\bar{\vec{x}}$	Empirical mean $\bar{\vec{x}} = \frac{1}{N}\sum_{i=1}^{N} \vec{x}_i$
$\tilde{\vec{x}}$	Future test case
$\vec{x}_*$	Feature test case
$\vec{y}$	Vector of all training labels $\vec{y} = (y_1, ..., y_N)$
z_{ij}	Latent component j for case i

Chapter 1
Introduction

1.1 Types of machine learning

- Supervised learning
  - Classification
  - Regression
- Unsupervised learning
  - Discovering clusters
  - Discovering latent factors
  - Discovering graph structure
  - Matrix completion

1.2 Three elements of a machine learning model

Model = Representation + Evaluation + Optimization[1]

1.2.1 Representation

In supervised learning, a model must be represented as a conditional probability distribution $P(y|\vec{x})$(usually we call it classifier) or a decision function $f(x)$. The set of classifiers(or decision functions) is called the hypothesis space of the model. Choosing a representation for a model is tantamount to choosing the hypothesis space that it can possibly learn.

1.2.2 Evaluation

In the hypothesis space, an evaluation function (also called objective function or risk function) is needed to distinguish good classifiers(or decision functions) from bad ones.

1.2.2.1 Loss function and risk function

Definition 1.1. In order to measure how well a function fits the training data, a **loss function** $L : Y \times Y \to R \geq 0$ is defined. For training example (x_i, y_i), the loss of predicting the value $\widehat{y}$ is $L(y_i, \widehat{y})$.

The following is some common loss functions:

1. 0-1 loss function
$$L(Y, f(X)) = \mathbb{I}(Y, f(X)) = \begin{cases} 1, & Y = f(X) \\ 0, & Y \neq f(X) \end{cases}$$
2. Quadratic loss function $L(Y, f(X)) = (Y - f(X))^2$
3. Absolute loss function $L(Y, f(X)) = |Y - f(X)|$
4. Logarithmic loss function
$L(Y, P(Y|X)) = -\log P(Y|X)$

Definition 1.2. The risk of function f is defined as the expected loss of f:

$$R_{\exp}(f) = E\left[L(Y, f(X))\right] = \int L(y, f(x))\, P(x, y)\mathrm{d}x\mathrm{d}y \quad (1.1)$$

which is also called expected loss or **risk function**.

Definition 1.3. The risk function $R_{\exp}(f)$ can be estimated from the training data as

$$R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) \quad (1.2)$$

which is also called empirical loss or **empirical risk**.

You can define your own loss function, but if you're a novice, you're probably better off using one from the literature. There are conditions that loss functions should meet[2]:

1. They should approximate the actual loss you're trying to minimize. As was said in the other answer, the standard loss functions for classification is zero-one-loss (misclassification rate) and the ones used for training classifiers are approximations of that loss.
2. The loss function should work with your intended optimization algorithm. That's why zero-one-loss is not used directly: it doesn't work with gradient-based optimization methods since it doesn't have a well-defined gradient (or even a subgradient, like the hinge loss for SVMs has).

   The main algorithm that optimizes the zero-one-loss directly is the old perceptron algorithm(chapter §**??**).

1.2.2.2 ERM and SRM

Definition 1.4. ERM(Empirical risk minimization)

$$\min_{f \in \mathcal{F}} R_{\text{emp}}(f) = \min_{f \in \mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) \quad (1.3)$$

Definition 1.5. Structural risk

$$R_{\text{smp}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \quad (1.4)$$

[1] Domingos, P. A few useful things to know about machine learning. Commun. ACM. 55(10):78–87 (2012).

[2] http://t.cn/zTrDxLO

Definition 1.6. SRM(Structural risk minimization)

$$\min_{f\in\mathcal{F}} R_{\text{srm}}(f) = \min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \tag{1.5}$$

1.2.3 Optimization

Finally, we need a **training algorithm**(also called **learning algorithm**) to search among the classifiers in the the hypothesis space for the highest-scoring one. The choice of optimization technique is key to the **efficiency** of the model.

1.3 Some basic concepts

1.3.1 Parametric vs non-parametric models

1.3.2 A simple non-parametric classifier: K-nearest neighbours

1.3.2.1 Representation

$$y = f(\vec{x}) = \arg\min_{c} \sum_{\vec{x}_i\in N_k(\vec{x})} \mathbb{I}(y_i = c) \tag{1.6}$$

where $N_k(\vec{x})$ is the set of k points that are closest to point $\vec{x}$.

Usually use **k-d tree** to accelerate the process of finding k nearest points.

1.3.2.2 Evaluation

No training is needed.

1.3.2.3 Optimization

No training is needed.

1.3.3 Overfitting

1.3.4 Cross validation

Definition 1.7. Cross validation, sometimes called *rotation estimation*, is a *model validation* technique for assessing how the results of a statistical analysis will generalize to an independent data set[3].

Common types of cross-validation:

1. K-fold cross-validation. In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k - 1 subsamples are used as training data.
2. 2-fold cross-validation. Also, called simple cross-validation or holdout method. This is the simplest variation of k-fold cross-validation, k=2.
3. Leave-one-out cross-validation(*LOOCV*). k=M, the number of original samples.

1.3.5 Model selection

When we have a variety of models of different complexity (e.g., linear or logistic regression models with different degree polynomials, or KNN classifiers with different values ofK), how should we pick the right one? A natural approach is to compute the **misclassification rate** on the training set for each method.

[3] http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Chapter 2
Probability

2.1 Frequentists vs. Bayesians

what is probability?

One is called the **frequentist** interpretation. In this view, probabilities represent long run frequencies of events. For example, the above statement means that, if we flip the coin many times, we expect it to land heads about half the time.

The other interpretation is called the **Bayesian** interpretation of probability. In this view, probability is used to quantify our **uncertainty** about something; hence it is fundamentally related to information rather than repeated trials (Jaynes 2003). In the Bayesian view, the above statement means we believe the coin is equally likely to land heads or tails on the next toss

One big advantage of the Bayesian interpretation is that it can be used to model our uncertainty about events that do not have long term frequencies. For example, we might want to compute the probability that the polar ice cap will melt by 2020 CE. This event will happen zero or one times, but cannot happen repeatedly. Nevertheless, we ought to be able to quantify our uncertainty about this event. To give another machine learning oriented example, we might have observed a "blip" on our radar screen, and want to compute the probability distribution over the location of the corresponding target (be it a bird, plane, or missile). In all these cases, the idea of repeated trials does not make sense, but the Bayesian interpretation is valid and indeed quite natural. We shall therefore adopt the Bayesian interpretation in this book. Fortunately, the basic rules of probability theory are the same, no matter which interpretation is adopted.

2.2 A brief review of probability theory

2.2.1 Basic concepts

We denote a random event by defining a **random variable** X.

Discrete random variable: X can take on any value from a finite or countably infinite set.

Continuous random variable: the value of X is real-valued.

2.2.1.1 CDF

$$F(x) \triangleq P(X \leq x) = \begin{cases} \sum_{u \leq x} p(u) & \text{, discrete} \\ \int_{-\infty}^{x} f(u)\mathrm{d}u & \text{, continuous} \end{cases} \tag{2.1}$$

2.2.1.2 PMF and PDF

For discrete random variable, We denote the probability of the event that $X = x$ by $P(X = x)$, or just $p(x)$ for short. Here $p(x)$ is called a **probability mass function** or **PMF**.A probability mass function is a function that gives the probability that a discrete random variable is exactly equal to some value[4]. This satisfies the properties $0 \leq p(x) \leq 1$ and $\sum_{x \in \mathcal{X}} p(x) = 1$.

For continuous variable, in the equation $F(x) = \int_{-\infty}^{x} f(u)\mathrm{d}u$, the function $f(x)$ is called a **probability density function** or **PDF**. A probability density function is a function that describes the relative likelihood for this random variable to take on a given value[5].This satisfies the properties $f(x) \geq 0$ and $\int_{-\infty}^{\infty} f(x)\mathrm{d}x = 1$.

2.2.2 Mutivariate random variables

2.2.2.1 Joint CDF

We denote joint CDF by $F(x,y) \triangleq P(X \leq x \cap Y \leq y) = P(X \leq x, Y \leq y)$.

$$F(x,y) \triangleq P(X \leq x, Y \leq y) = \begin{cases} \sum_{u \leq x, v \leq y} p(u,v) \\ \int_{-\infty}^{x} \int_{-\infty}^{y} f(u,v)\mathrm{d}u\mathrm{d}v \end{cases} \tag{2.2}$$

product rule:

$$p(X,Y) = P(X|Y)P(Y) \tag{2.3}$$

Chain rule:

$$p(X_{1:N}) = p(X_1)p(X_2|X_1)p(X_3|X_2,X_1)...p(X_N|X_{1:N-1}) \tag{2.4}$$

2.2.2.2 Marginal distribution

Marginal CDF:

[4] http://en.wikipedia.org/wiki/Probability_mass_function

[5] http://en.wikipedia.org/wiki/Probability_density_function

$$F_X(x) \triangleq F(x,+\infty) = \begin{cases} \sum_{x_i \le x} P(X=x_i) = \sum_{x_i \le x} \sum_{j=1}^{+\infty} P(X=x_i, Y=y_j) \\ \int_{-\infty}^{x} f_X(u)du = \int_{-\infty}^{x} \int_{-\infty}^{+\infty} f(u,v)\mathrm{d}u\mathrm{d}v \end{cases} \quad (2.5)$$

$$F_Y(y) \triangleq F(+\infty,y) = \begin{cases} \sum_{y_j \le y} p(Y=y_j) = \sum_{i=1}^{+\infty} \sum_{y_j \le y} P(X=x_i, Y=y_j) \\ \int_{-\infty}^{y} f_Y(v)dv = \int_{-\infty}^{+\infty} \int_{-\infty}^{y} f(u,v)\mathrm{d}u\mathrm{d}v \end{cases} \quad (2.6)$$

Marginal PMF and PDF:

$$\begin{cases} P(X=x_i) = \sum_{j=1}^{+\infty} P(X=x_i, Y=y_j) & \text{, discrete} \\ f_X(x) = \int_{-\infty}^{+\infty} f(x,y)\mathrm{d}y & \text{, continuous} \end{cases} \quad (2.7)$$

$$\begin{cases} p(Y=y_j) = \sum_{i=1}^{+\infty} P(X=x_i, Y=y_j) & \text{, discrete} \\ f_Y(y) = \int_{-\infty}^{+\infty} f(x,y)\mathrm{d}x & \text{, continuous} \end{cases} \quad (2.8)$$

2.2.2.3 Conditional distribution

Conditional PMF:

$$p(X=x_i|Y=y_j) = \frac{p(X=x_i, Y=y_j)}{p(Y=y_j)} \text{ if } p(Y)>0 \quad (2.9)$$

The pmf $p(X|Y)$ is called **conditional probability**.

Conditional PDF:

$$f_{X|Y}(x|y) = \frac{f(x,y)}{f_Y(y)} \quad (2.10)$$

2.2.3 Bayes rule

$$p(Y=y|X=x) = \frac{p(X=x,Y=y)}{p(X=x)} = \frac{p(X=x|Y=y)p(Y=y)}{\sum_{y'} p(X=x|Y=y')p(Y=y')} \quad (2.11)$$

2.2.4 Independence and conditional independence

We say X and Y are unconditionally independent or marginally independent, denoted $X \perp Y$, if we can represent the joint as the product of the two marginals, i.e.,

$$X \perp Y = P(X,Y) = P(X)P(Y) \quad (2.12)$$

We say X and Y are conditionally independent(CI) given Z if the conditional joint can be written as a product of conditional marginals:

$$X \perp Y|Z = P(X,Y|Z) = P(X|Z)P(Y|Z) \quad (2.13)$$

2.2.5 Quantiles

Since the cdf F is a monotonically increasing function, it has an inverse; let us denote this by F^{-1}. If F is the cdf of X , then $F^{-1}(\alpha)$ is the value of x_α such that $P(X \le x_\alpha) = \alpha$; this is called the α quantile of F. The value $F^{-1}(0.5)$ is the **median** of the distribution, with half of the probability mass on the left, and half on the right. The values $F^{-1}(0.25)$ and $F^{-1}(0.75)$are the lower and upper **quartiles**.

2.2.6 Mean and variance

The most familiar property of a distribution is its **mean**,or **expected value**, denoted by μ. For discrete rv's, it is defined as $\mathbb{E}[X] \triangleq \sum_{x \in \mathcal{X}} xp(x)$, and for continuous rv's, it is defined as $\mathbb{E}[X] \triangleq \int_{\mathcal{X}} xp(x)\mathrm{d}x$. If this integral is not finite, the mean is not defined (we will see some examples of this later).

The **variance** is a measure of the "spread" of a distribution, denoted by σ^2. This is defined as follows:

$$\begin{aligned} var[X] &= \mathbb{E}[(X-\mu)^2] && (2.14) \\ &= \int (x-\mu)^2 p(x)\mathrm{d}x \\ &= \int x^2 p(x)\mathrm{d}x + \mu^2 \int p(x)\mathrm{d}x - 2\mu \int xp(x)\mathrm{d}x \\ &= \mathbb{E}[X^2] - \mu^2 && (2.15) \end{aligned}$$

from which we derive the useful result

$$\mathbb{E}[X^2] = \sigma^2 + \mu^2 \quad (2.16)$$

The **standard deviation** is defined as

$$std[X] \triangleq \sqrt{var[X]} \quad (2.17)$$

This is useful since it has the same units as X itself.

https://en.wikipedia.org/wiki/Standardized_moment

2.3 Some common discrete distributions

In this section, we review some commonly used parametric distributions defined on discrete state spaces, both finite and countably infinite.

2.3.1 The Bernoulli and binomial distributions

Definition 2.1. Now suppose we toss a coin only once. Let $X \in \{0,1\}$ be a binary random variable, with probability of "success" or "heads" of θ. We say that X has a **Bernoulli distribution**. This is written as $X \sim \text{Ber}(\theta)$, where the pmf is defined as

$$\text{Ber}(x|\theta) \triangleq \theta^{\mathbb{I}(x=1)}(1-\theta)^{\mathbb{I}(x=0)} \tag{2.18}$$

Definition 2.2. Suppose we toss a coin n times. Let $X \in \{0,1,\cdots,n\}$ be the number of heads. If the probability of heads is θ, then we say X has a **binomial distribution**, written as $X \sim \text{Bin}(n,\theta)$. The pmf is given by

$$\text{Bin}(k|n,\theta) \triangleq \binom{n}{k}\theta^k(1-\theta)^{n-k} \tag{2.19}$$

2.3.2 The multinoulli and multinomial distributions

Definition 2.3. The Bernoulli distribution can be used to model the outcome of one coin tosses. To model the outcome of tossing a K-sided dice, let $\vec{x} = (\mathbb{I}(x=1),\cdots,\mathbb{I}(x=K)) \in \{0,1\}^K$ be a random vector(this is called **dummy encoding** or **one-hot encoding**), then we say X has a **multinoulli distribution**(or **categorical distribution**), written as $X \sim \text{Cat}(\theta)$. The pmf is given by:

$$p(\vec{x}) \triangleq \prod_{k=1}^{K} \theta_k^{\mathbb{I}(x_k=1)} \tag{2.20}$$

Definition 2.4. Suppose we toss a K-sided dice n times. Let $\vec{x} = (x_1,x_2,\cdots,x_K) \in \{0,1,\cdots,n\}^K$ be a random vector, where x_j is the number of times side j of the dice occurs, then we say X has a **multinomial distribution**, written as $X \sim \text{Mu}(n,\vec{\theta})$. The pmf is given by

$$p(\vec{x}) \triangleq \binom{n}{x_1 \cdots x_k} \prod_{k=1}^{K} \theta_k^{x_k} \tag{2.21}$$

where $\binom{n}{x_1 \cdots x_k} \triangleq \frac{n!}{x_1!x_2!\cdots x_K!}$

Bernoulli distribution is just a special case of a Binomial distribution with $n=1$, and so is multinoulli distribution as to multinomial distribution. See Table 2.1 for a summary.

Table 2.1: Summary of the multinomial and related distributions.

Name	K	n	X
Bernoulli	1	1	$x \in \{0,1\}$
Binomial	1	-	$\vec{x} \in \{0,1,\cdots,n\}$
Multinoulli	-	1	$\vec{x} \in \{0,1\}^K, \sum_{k=1}^K x_k = 1$
Multinomial	-	-	$\vec{x} \in \{0,1,\cdots,n\}^K, \sum_{k=1}^K x_k = n$

2.3.3 The Poisson distribution

Definition 2.5. We say that $X \in \{0,1,2,\cdots\}$ has a **Poisson distribution** with parameter $\lambda > 0$, written as $X \sim \text{Poi}(\lambda)$, if its pmf is

$$p(x|\lambda) = e^{-\lambda}\frac{\lambda^x}{x!} \tag{2.22}$$

The first term is just the normalization constant, required to ensure the distribution sums to 1.

The Poisson distribution is often used as a model for counts of rare events like radioactive decay and traffic accidents.

2.3.4 The empirical distribution

The **empirical distribution function**[6], or **empirical cdf**, is the cumulative distribution function associated with the empirical measure of the sample. Let $\mathcal{D} = \{x_1,x_2,\cdots,x_N\}$ be a sample set, it is defined as

$$F_n(x) \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(x_i \le x) \tag{2.23}$$

2.4 Some common continuous distributions

In this section we present some commonly used univariate (one-dimensional) continuous probability distributions.

2.4.1 Gaussian (normal) distribution

If $X \sim N(0,1)$,we say X follows a **standard normal** distribution.

The Gaussian distribution is the most widely used distribution in statistics. There are several reasons for this.

1. First, it has two parameters which are easy to interpret, and which capture some of the most basic properties of a distribution, namely its mean and variance.
2. Second,the central limit theorem (Section TODO) tells us that sums of independent random variables have an approximately Gaussian distribution, making it a good choice for modeling residual errors or "noise".
3. Third, the Gaussian distribution makes the least number of assumptions (has maximum entropy), subject to the constraint of having a specified mean and variance, as we show in Section TODO; this makes it a good default choice in many cases.
4. Finally, it has a simple mathematical form, which results in easy to implement, but often highly effective, methods, as we will see.

[6] http://en.wikipedia.org/wiki/Empirical_distribution_function

Table 2.2: Summary of Bernoulli, binomial multinoulli and multinomial distributions.

Name	Written as	X	$p(x)$(or $p(\vec{x})$)	$\mathbb{E}[X]$	var$[X]$
Bernoulli	$X \sim \text{Ber}(\theta)$	$x \in \{0,1\}$	$\theta^{\mathbb{I}(x=1)}(1-\theta)^{\mathbb{I}(x=0)}$	θ	$\theta(1-\theta)$
Binomial	$X \sim \text{Bin}(n,\theta)$	$x \in \{0,1,\cdots,n\}$	$\binom{n}{k}\theta^k(1-\theta)^{n-k}$	$n\theta$	$n\theta(1-\theta)$
Multinoulli	$X \sim \text{Cat}(\vec{\theta})$	$\vec{x} \in \{0,1\}^K, \sum_{k=1}^K x_k = 1$	$\prod_{k=1}^K \theta_j^{\mathbb{I}(x_j=1)}$	-	-
Multinomial	$X \sim \text{Mu}(n,\vec{\theta})$	$\vec{x} \in \{0,1,\cdots,n\}^K, \sum_{k=1}^K x_k = n$	$\binom{n}{x_1 \cdots x_k}\prod_{k=1}^K \theta_j^{x_j}$	-	-
Poisson	$X \sim \text{Poi}(\lambda)$	$x \in \{0,1,2,\cdots\}$	$e^{-\lambda}\frac{\lambda^x}{x!}$	λ	λ

Table 2.3: Summary of Gaussian distribution.

Written as	$f(x)$	$\mathbb{E}[X]$	mode	var$[X]$
$X \sim \mathcal{N}(\mu,\sigma^2)$	$\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$	μ	μ	σ^2

See (Jaynes 2003, ch 7) for a more extensive discussion of why Gaussians are so widely used.

2.4.2 Student's t-distribution

Table 2.4: Summary of Student's t-distribution.

Written as	$f(x)$	$\mathbb{E}[X]$	mode	var$[X]$
$X \sim \mathcal{T}(\mu,\sigma^2,\nu)$	$\frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}\left[1+\frac{1}{\nu}\left(\frac{x-\mu}{\nu}\right)^2\right]$	μ	μ	$\frac{\nu\sigma^2}{\nu-2}$

where $\Gamma(x)$ is the gamma function:

$$\Gamma(x) \triangleq \int_0^\infty t^{x-1}e^{-t}\mathrm{d}t \tag{2.24}$$

μ is the mean, $\sigma^2 > 0$ is the scale parameter, and $\nu > 0$ is called the **degrees of freedom**. See Figure 2.1 for some plots.

The variance is only defined if $\nu > 2$. The mean is only defined if $\nu > 1$.

As an illustration of the robustness of the Student distribution, consider Figure 2.2. We see that the Gaussian is affected a lot, whereas the Student distribution hardly changes. This is because the Student has heavier tails, at least for small ν(see Figure 2.1).

If $\nu = 1$, this distribution is known as the **Cauchy** or **Lorentz** distribution. This is notable for having such heavy tails that the integral that defines the mean does not converge.

To ensure finite variance, we require $\nu > 2$. It is common to use $\nu = 4$, which gives good performance in a range of problems (Lange et al. 1989). For $\nu \gg 5$, the Student distribution rapidly approaches a Gaussian distribution and loses its robustness properties.

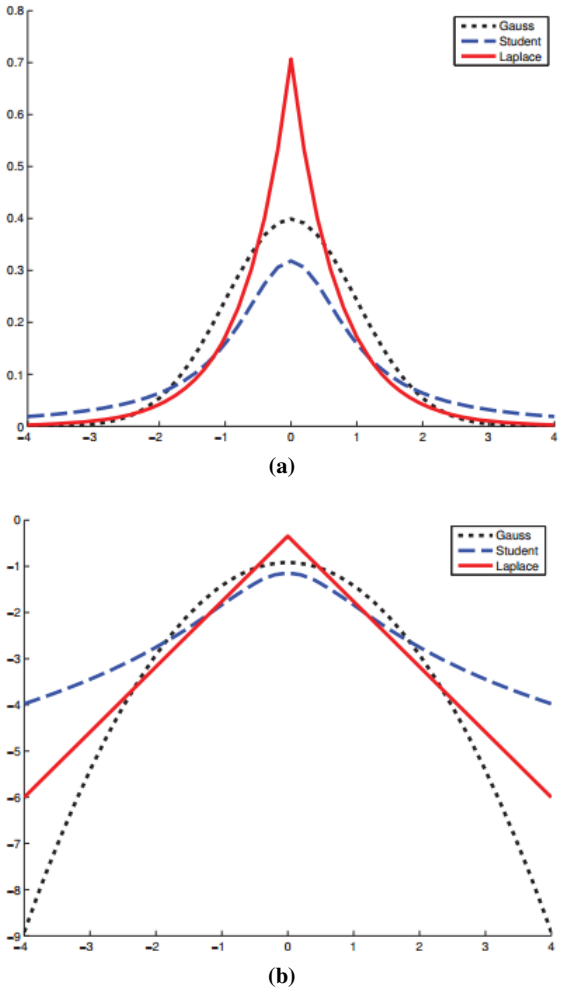


Fig. 2.1: (a) The pdf's for a $\mathcal{N}(0,1)$, $\mathcal{T}(0,1,1)$ and $Lap(0,1/\sqrt{2})$. The mean is 0 and the variance is 1 for both the Gaussian and Laplace. The mean and variance of the Student is undefined when $\nu = 1$.(b) Log of these pdf's. Note that the Student distribution is not log-concave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

2.4.3 The Laplace distribution

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

Its robustness to outliers is illustrated in Figure 2.2. It also put mores probability density at 0 than the Gaussian. This

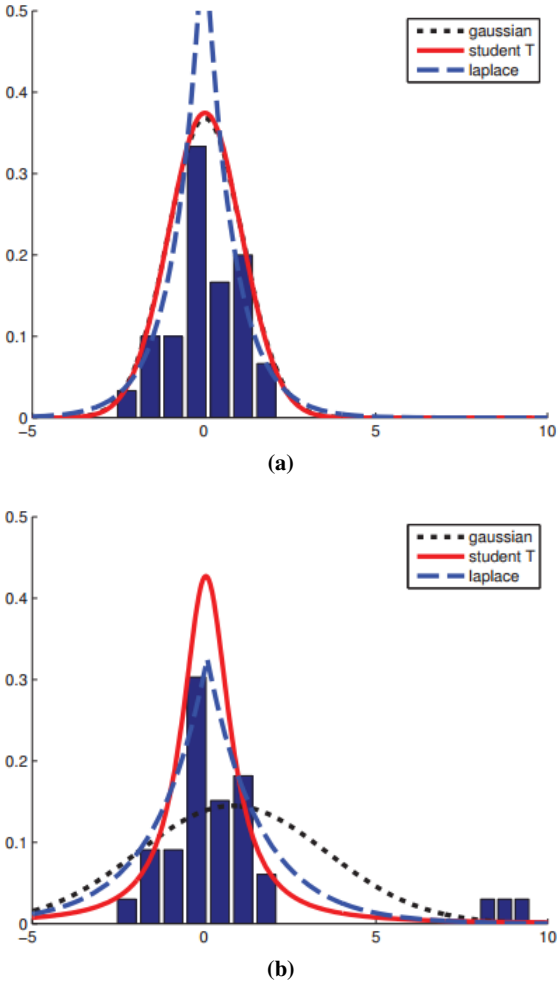


Fig. 2.2: Illustration of the effect of outliers on fitting Gaussian, Student and Laplace distributions. (a) No outliers (the Gaussian and Student curves are on top of each other). (b) With outliers. We see that the Gaussian is more affected by outliers than the Student and Laplace distributions.

Table 2.5: Summary of Laplace distribution.

Written as	$f(x)$	$\mathbb{E}[X]$	mode	var$[X]$
$X \sim \text{Lap}(\mu, b)$	$\frac{1}{2b}\exp\left(-\frac{\lvert x-\mu\rvert}{b}\right)$	μ	μ	$2b^2$

property is a useful way to encourage sparsity in a model, as we will see in Section TODO.

2.4.4 The gamma distribution

Table 2.6: Summary of gamma distribution

Written as	X	$f(x)$	$\mathbb{E}[X]$	mode	var$[X]$
$X \sim \text{Ga}(a,b)$	$x \in \mathbb{R}^+$	$\frac{b^a}{\Gamma(a)}x^{a-1}e^{-xb}$	$\frac{a}{b}$	$\frac{a-1}{b}$	$\frac{a}{b^2}$

Here $a > 0$ is called the shape parameter and $b > 0$ is called the rate parameter. See Figure 2.3 for some plots.

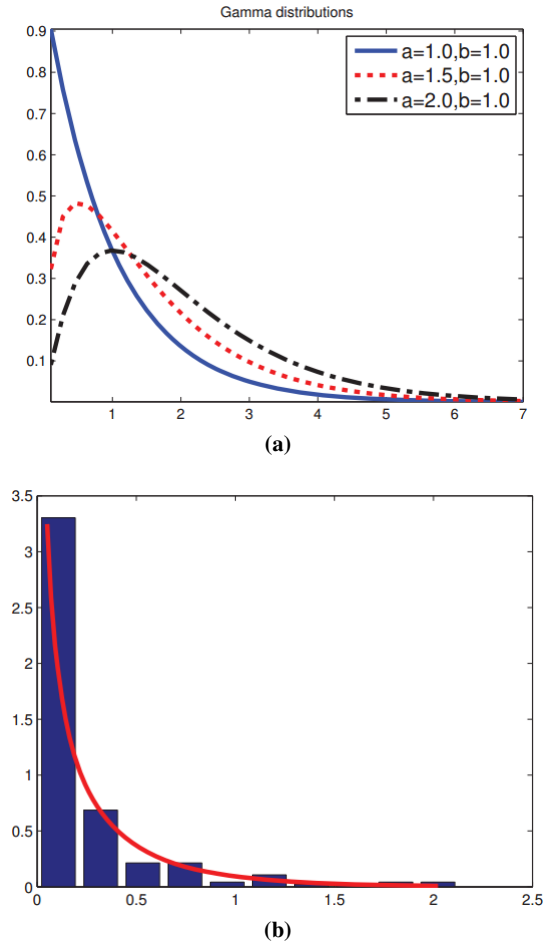


Fig. 2.3: Some Ga$(a, b = 1)$ distributions. If $a \leq 1$, the mode is at 0, otherwise it is > 0.As we increase the rate b, we reduce the horizontal scale, thus squeezing everything leftwards and upwards. (b) An empirical pdf of some rainfall data, with a fitted Gamma distribution superimposed.

2.4.5 The beta distribution

Here $B(a,b)$is the beta function,

$$B(a,b) \triangleq \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{2.25}$$

See Figure 2.4 for plots of some beta distributions. We require $a, b > 0$ to ensure the distribution is integrable (i.e., to ensure $B(a,b)$ exists). If $a = b = 1$, we get the uniform distirbution. If a and b are both less than 1, we get a bimodal distribution with "spikes" at 0 and 1; if a and b are both greater than 1, the distribution is unimodal.

2.4.6 Pareto distribution

The **Pareto distribution** is used to model the distribution of quantities that exhibit **long tails**, also called **heavy tails**.

As $k \to \infty$, the distribution approaches $\delta(x-m)$. See Figure 2.5(a) for some plots. If we plot the distribution on a log-log scale, it forms a straight line, of the form $\log p(x) =$

Table 2.7: Summary of Beta distribution

Name	Written as	X	$f(x)$	$\mathbb{E}[X]$	mode	var$[X]$
Beta distribution	$X \sim \text{Beta}(a,b)$	$x \in [0,1]$	$\frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$	$\frac{a}{a+b}$	$\frac{a-1}{a+b-2}$	$\frac{ab}{(a+b)^2(a+b+1)}$

Table 2.8: Summary of Pareto distribution

Name	Written as	X	$f(x)$	$\mathbb{E}[X]$	mode	var$[X]$
Pareto distribution	$X \sim \text{Pareto}(k,m)$	$x \geq m$	$km^k x^{-(k+1)} \mathbb{I}(x \geq m)$	$\frac{km}{k-1}$ if $k>1$	m	$\frac{m^2 k}{(k-1)^2(k-2)}$ if $k>2$

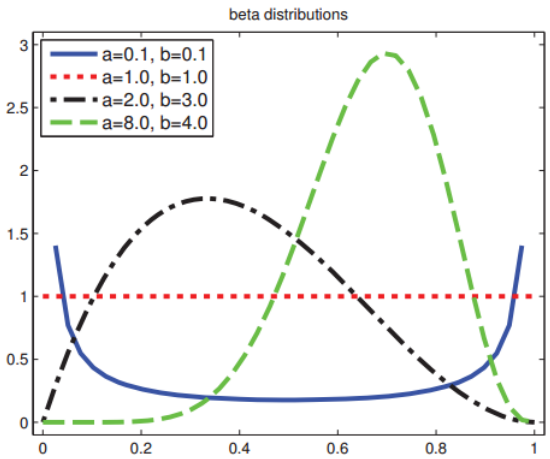


Fig. 2.4: Some beta distributions.

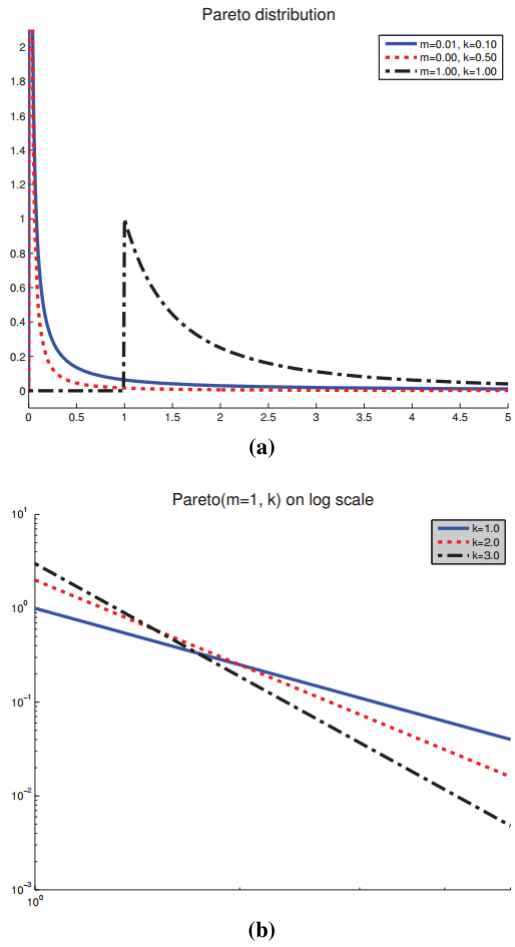


Fig. 2.5: (a) The Pareto distribution Pareto$(x|m,k)$ for $m=1$. (b) The pdf on a log-log scale.

$a \log x + c$ for some constants a and c. See Figure 2.5(b) for an illustration (this is known as a **power law**).

2.5 Joint probability distributions

Given a **multivariate random variable** or **random vector** [7] $X \in \mathbb{R}^D$, the **joint probability distribution**[8] is a probability distribution that gives the probability that each of $X_1, X_2, \cdots, X_D$ falls in any particular range or discrete set of values specified for that variable. In the case of only two random variables, this is called a **bivariate distribution**, but the concept generalizes to any number of random variables, giving a **multivariate distribution**.

The joint probability distribution can be expressed either in terms of a **joint cumulative distribution function** or in terms of a **joint probability density function** (in the case of continuous variables) or **joint probability mass function** (in the case of discrete variables).

2.5.1 Covariance and correlation

Definition 2.6. The **covariance** between two rv's X and Y measures the degree to which X and Y are (linearly) related. Covariance is defined as

$$\begin{aligned} \text{cov}[X,Y] &\triangleq \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right] \\ &= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] \end{aligned} \tag{2.26}$$

Definition 2.7. If X is a D-dimensional random vector, its **covariance matrix** is defined to be the following symmetric, positive definite matrix:

[7] http://en.wikipedia.org/wiki/Multivariate_random_variable

[8] http://en.wikipedia.org/wiki/Joint_probability_distribution

$$\text{cov}[X] \triangleq \mathbb{E}\left[(X-\mathbb{E}[X])(X-\mathbb{E}[X])^T\right] \tag{2.27}$$

$$= \begin{pmatrix} \text{var}[X_1] & \text{Cov}[X_1,X_2] & \cdots & \text{Cov}[X_1,X_D] \\ \text{Cov}[X_2,X_1] & \text{var}[X_2] & \cdots & \text{Cov}[X_2,X_D] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[X_D,X_1] & \text{Cov}[X_D,X_2] & \cdots & \text{var}[X_D] \end{pmatrix} \tag{2.28}$$

Definition 2.8. The (Pearson) **correlation coefficient** between X and Y is defined as

$$\text{corr}[X,Y] \triangleq \frac{\text{Cov}[X,Y]}{\sqrt{\text{var}[X],\text{var}[Y]}} \tag{2.29}$$

A **correlation matrix** has the form

$$\mathbf{R} \triangleq \begin{pmatrix} \text{corr}[X_1,X_1] & \text{corr}[X_1,X_2] & \cdots & \text{corr}[X_1,X_D] \\ \text{corr}[X_2,X_1] & \text{corr}[X_2,X_2] & \cdots & \text{corr}[X_2,X_D] \\ \vdots & \vdots & \ddots & \vdots \\ \text{corr}[X_D,X_1] & \text{corr}[X_D,X_2] & \cdots & \text{corr}[X_D,X_D] \end{pmatrix} \tag{2.30}$$

The correlation coefficient can viewed as a degree of linearity between X and Y, see Figure 2.6.

Uncorrelated does not imply independent. For example, let $X \sim U(-1,1)$ and $Y = X^2$. Clearly Y is dependent on X(in fact, Y is uniquely determined by X), yet one can show that $\text{corr}[X,Y] = 0$. Some striking examples of this fact are shown in Figure 2.6. This shows several data sets where there is clear dependence between X and Y, and yet the correlation coefficient is 0. A more general measure of dependence between random variables is mutual information, see Section TODO.

2.5.2 Multivariate Gaussian distribution

The **multivariate Gaussian** or **multivariate normal**(MVN) is the most widely used joint probability density function for continuous variables. We discuss MVNs in detail in Chapter 4; here we just give some definitions and plots.

The pdf of the MVN in D dimensions is defined by the following:

$$\mathcal{N}(\vec{x}|\vec{\mu},\Sigma) \triangleq \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right] \tag{2.31}$$

where $\vec{\mu} = \mathbb{E}[X] \in \mathbb{R}^D$ is the mean vector, and $\Sigma = \text{Cov}[X]$ is the $D \times D$ covariance matrix. The normalization constant $(2\pi)^{D/2}|\Sigma|^{1/2}$ just ensures that the pdf integrates to 1.

Figure 2.7 plots some MVN densities in 2d for three different kinds of covariance matrices. A full covariance matrix has A $D(D+1)/2$ parameters (we divide by 2 since Σ is symmetric). A diagonal covariance matrix has D parameters, and has 0s in the off-diagonal terms. A spherical or isotropic covariance,$\Sigma = \sigma^2\vec{I}_D$, has one free parameter.

2.5.3 Multivariate Student's t-distribution

A more robust alternative to the MVN is the multivariate Student's t-distribution, whose pdf is given by

$$\mathcal{T}(x|\vec{\mu},\Sigma,\nu) \triangleq \frac{\Gamma(\frac{\nu+D}{2})}{\Gamma(\frac{\nu}{2})}\frac{|\Sigma|^{-\frac{1}{2}}}{(\nu\pi)^{\frac{D}{2}}}\left[1+\frac{1}{\nu}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right]^{-\frac{\nu+D}{2}} \tag{2.32}$$

$$= \frac{\Gamma(\frac{\nu+D}{2})}{\Gamma(\frac{\nu}{2})}\frac{|\Sigma|^{-\frac{1}{2}}}{(\nu\pi)^{\frac{D}{2}}}\left[1+(\vec{x}-\vec{\mu})^T\vec{V}^{-1}(\vec{x}-\vec{\mu})\right]^{-\frac{\nu+D}{2}} \tag{2.33}$$

where Σ is called the scale matrix (since it is not exactly the covariance matrix) and $\vec{V} = \nu\Sigma$. This has fatter tails than a Gaussian. The smaller ν is, the fatter the tails. As $\nu \to \infty$, the distribution tends towards a Gaussian. The distribution has the following properties

$$\text{mean} = \vec{\mu}\ , \text{mode} = \vec{\mu}\ , \text{Cov} = \frac{\nu}{\nu-2}\Sigma \tag{2.34}$$

2.5.4 Dirichlet distribution

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

$$S_K = \left\{\vec{x} : 0 \le x_k \le 1, \sum_{k=1}^{K} x_k = 1\right\} \tag{2.35}$$

The pdf is defined as follows:

$$\text{Dir}(\vec{x}|\vec{\alpha}) \triangleq \frac{1}{B(\vec{\alpha})}\prod_{k=1}^{K} x_k^{\alpha_k-1}\mathbb{I}(\vec{x} \in S_K) \tag{2.36}$$

where $B(\alpha_1,\alpha_2,\cdots,\alpha_K)$ is the natural generalization of the beta function to K variables:

$$B(\alpha) \triangleq \frac{\prod_{k=1}^{K}\Gamma(\alpha_k)}{\Gamma(\alpha_0)} \text{ where } \alpha_0 \triangleq \sum_{k=1}^{K}\alpha_k \tag{2.37}$$

Figure 2.8 shows some plots of the Dirichlet when $K = 3$, and Figure 2.9 for some sampled probability vectors. We see that α_0 controls the strength of the distribution (how peaked it is), and the α_k control where the peak occurs. For example, $\text{Dir}(1,1,1)$ is a uniform distribution, $\text{Dir}(2,2,2)$ is a broad distribution centered at $(1/3,1/3,1/3)$, and $\text{Dir}(20,20,20)$ is a narrow distribution centered at $(1/3,1/3,1/3)$.If $\alpha_k < 1$ for all k, we get "spikes" at the corner of the simplex.

For future reference, the distribution has these properties

$$\mathbb{E}(x_k) = \frac{\alpha_k}{\alpha_0}, \text{mode}[x_k] = \frac{\alpha_k-1}{\alpha_0-K}, \text{var}[x_k] = \frac{\alpha_k(\alpha_0-\alpha_k)}{\alpha_0^2(\alpha_0+1)} \tag{2.38}$$

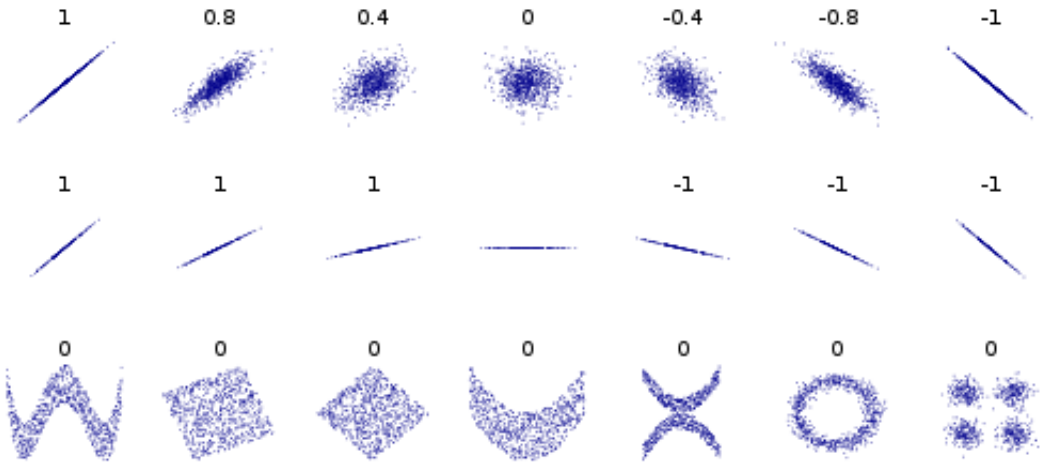


Fig. 2.6: Several sets of (x,y) points, with the Pearson correlation coefficient of x and y for each set. Note that the correlation reflects the noisiness and direction of a linear relationship (top row), but not the slope of that relationship (middle), nor many aspects of nonlinear relationships (bottom). N.B.: the figure in the center has a slope of 0 but in that case the correlation coefficient is undefined because the variance of Y is zero.Source:http://en.wikipedia.org/wiki/Correlation

2.6 Transformations of random variables

If $\vec{x} \sim P()$ is some random variable, and $\vec{y} = f(\vec{x})$, what is the distribution of Y? This is the question we address in this section.

2.6.1 Linear transformations

Suppose $g()$ is a linear function:

$$g(\vec{x}) = A\vec{x} + b \tag{2.39}$$

First, for the mean, we have

$$\mathbb{E}[\vec{y}] = \mathbb{E}[A\vec{x} + b] = A\mathbb{E}[\vec{x}] + b \tag{2.40}$$

this is called the **linearity of expectation**.

For the covariance, we have

$$\text{Cov}[\vec{y}] = \text{Cov}[A\vec{x} + b] = A\Sigma A^T \tag{2.41}$$

2.6.2 General transformations

If X is a discrete rv, we can derive the pmf for y by simply summing up the probability mass for all the x's such that $f(x) = y$:

$$p_Y(y) = \sum_{x:g(x)=y} p_X(x) \tag{2.42}$$

If X is continuous, we cannot use Equation 2.42 since $p_X(x)$ is a density, not a pmf, and we cannot sum up densities. Instead, we work with cdf's, and write

$$F_Y(y) = P(Y \leq y) = P(g(X) \leq y) = \int_{g(X)\leq y} f_X(x)\text{d}x \tag{2.43}$$

We can derive the pdf of Y by differentiating the cdf:

$$f_Y(y) = f_X(x)|\frac{dx}{dy}| \tag{2.44}$$

This is called **change of variables** formula. We leave the proof of this as an exercise.

For example, suppose $X \sim U(-1,1)$, and $Y = X^2$. Then $p_Y(y) = \frac{1}{2}y^{-\frac{1}{2}}$.

2.6.2.1 Multivariate change of variables *

Let f be a function $f : \mathbb{R}^n \to \mathbb{R}^n$, and let $\vec{y} = f(\vec{x})$. Then its Jacobian matrix $\vec{J}$ is given by

$$\vec{J}_{\vec{x}\to\vec{y}} \triangleq \frac{\partial \vec{y}}{\partial \vec{x}} \triangleq \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_n} \end{pmatrix} \tag{2.45}$$

$|\det(\vec{J})|$ measures how much a unit cube changes in volume when we apply f.

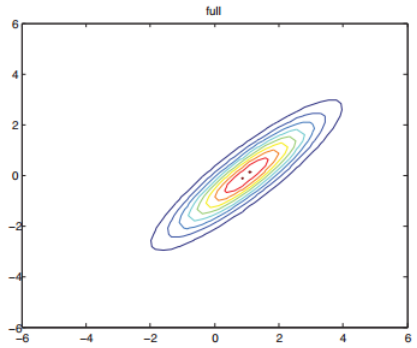
If f is an invertible mapping, we can define the pdf of the transformed variables using the Jacobian of the inverse mapping $\vec{y} \to \vec{x}$:

$$p_y(\vec{y}) = p_x(\vec{x})|\det(\frac{\partial \vec{x}}{\partial \vec{y}})| = p_x(\vec{x})|\det(\vec{J}_{\vec{y}\to\vec{x}})| \tag{2.46}$$

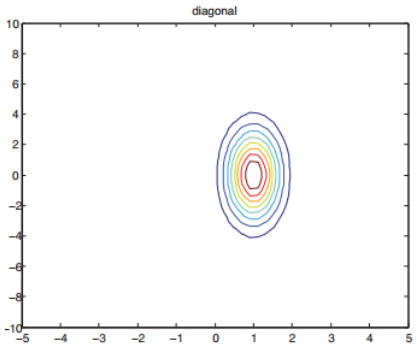
2.6.3 Central limit theorem

Given N random variables $X_1, X_2, \cdots, X_N$, each variable is **independent and identically distributed**[9](**iid** for short),

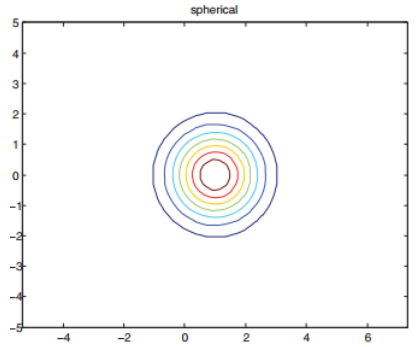
[9] http://en.wikipedia.org/wiki/Independent_identically_distributed



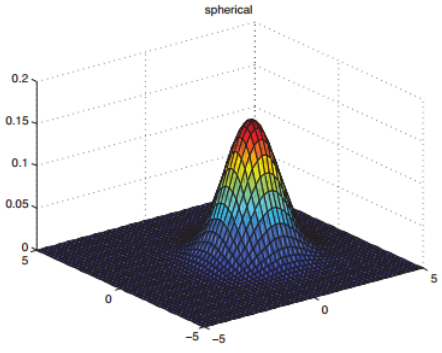
(a)



(b)

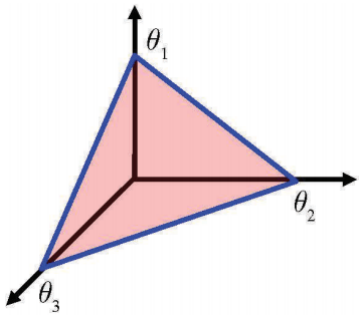


(c)

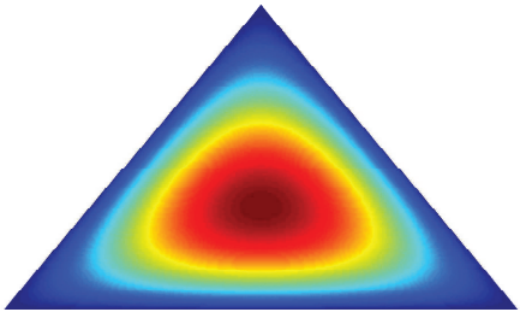


(d)

Fig. 2.7: We show the level sets for 2d Gaussians. (a) A full covariance matrix has elliptical contours.(b) A diagonal covariance matrix is an axis aligned ellipse. (c) A spherical covariance matrix has a circular shape. (d) Surface plot for the spherical Gaussian in (c).

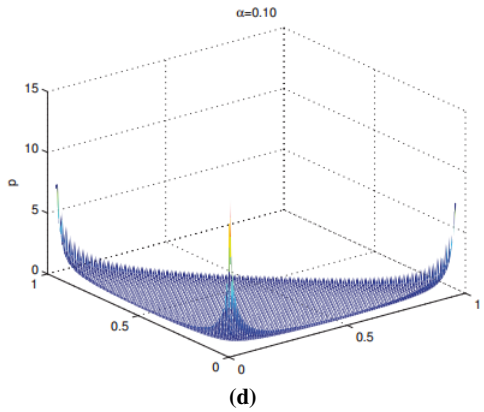


(a)



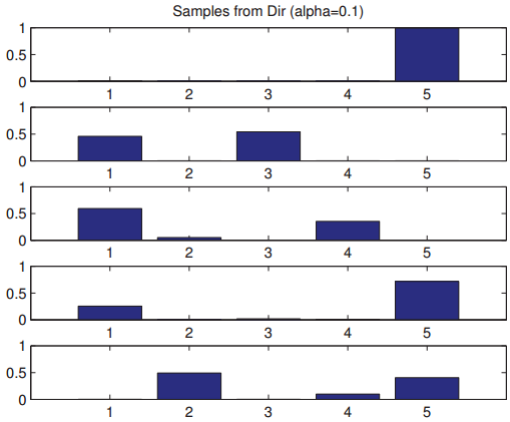
(b)

(c)

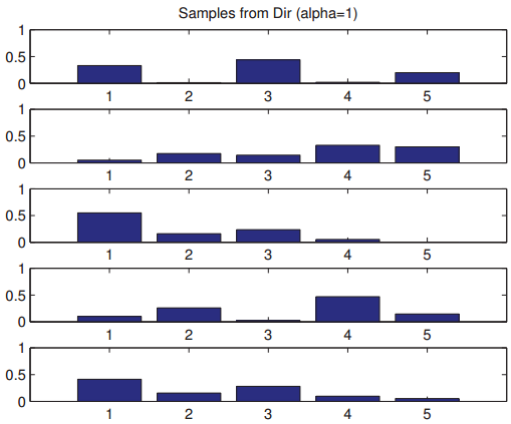


(d)

Fig. 2.8: (a) The Dirichlet distribution when $K = 3$ defines a distribution over the simplex, which can be represented by the triangular surface. Points on this surface satisfy $0 \leq \theta_k \leq 1$ and $\sum_{k=1}^{K} \theta_k = 1$. (b) Plot of the Dirichlet density when $\vec{\alpha} = (2,2,2)$. (c) $\vec{\alpha} = (20,2,2)$.



(a) $\vec{\alpha} = (0.1, \cdots, 0.1)$. This results in very sparse distributions, with many 0s.



(b) $\vec{\alpha} = (1, \cdots, 1)$. This results in more uniform (and dense) distributions.

Fig. 2.9: Samples from a 5-dimensional symmetric Dirichlet distribution for different parameter values.

and each has the same mean μ and variance σ^2, then

$$\frac{\sum_{i=1}^{n} X_i - N\mu}{\sqrt{N}\sigma} \sim \mathcal{N}(0,1) \tag{2.47}$$

this can also be written as

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{N}} \sim \mathcal{N}(0,1) \quad \text{, where } \bar{X} \triangleq \frac{1}{N}\sum_{i=1}^{n} X_i \tag{2.48}$$

2.7 Monte Carlo approximation

In general, computing the distribution of a function of an rv using the change of variables formula can be difficult. One simple but powerful alternative is as follows. First we generate S samples from the distribution, call them $x_1, \cdots, x_S$. (There are many ways to generate such samples; one popular method, for high dimensional distributions, is called Markov chain Monte Carlo or MCMC; this will be explained in Chapter TODO.) Given the samples, we can approximate the distribution of $f(X)$ by using the empirical distribution of $\{f(x_s)\}_{s=1}^{S}$. This is called a **Monte Carlo approximation**[10], named after a city in Europe known for its plush gambling casinos.

We can use Monte Carlo to approximate the expected value of any function of a random variable. We simply draw samples, and then compute the arithmetic mean of the function applied to the samples. This can be written as follows:

$$\mathbb{E}[g(X)] = \int g(x)p(x)\mathrm{d}x \approx \frac{1}{S}\sum_{s=1}^{S} f(x_s) \tag{2.49}$$

where $x_s \sim p(X)$.

This is called **Monte Carlo integration**[11], and has the advantage over numerical integration (which is based on evaluating the function at a fixed grid of points) that the function is only evaluated in places where there is non-negligible probability.

2.8 Information theory

2.8.1 Entropy

The entropy of a random variable X with distribution p, denoted by $\mathbb{H}(X)$ or sometimes $\mathbb{H}(p)$, is a measure of its uncertainty. In particular, for a discrete variable with K states, it is defined by

$$\mathbb{H}(X) \triangleq -\sum_{k=1}^{K} p(X=k)\log_2 p(X=k) \tag{2.50}$$

Usually we use log base 2, in which case the units are called **bits**(short for binary digits). If we use log base e , the units are called **nats**.

The discrete distribution with maximum entropy is the uniform distribution (see Section XXX for a proof). Hence for a K-ary random variable, the entropy is maximized if $p(x=k) = 1/K$; in this case, $\mathbb{H}(X) = \log_2 K$.

Conversely, the distribution with minimum entropy (which is zero) is any **delta-function** that puts all its mass on one state. Such a distribution has no uncertainty.

2.8.2 KL divergence

One way to measure the dissimilarity of two probability distributions, p and q , is known as the **Kullback-Leibler divergence**(**KL divergence**)or **relative entropy**. This is defined as follows:

$$\mathbb{KL}(P||Q) \triangleq \sum_{x} p(x)\log_2 \frac{p(x)}{q(x)} \tag{2.51}$$

[10] http://en.wikipedia.org/wiki/Monte_Carlo_method

[11] http://en.wikipedia.org/wiki/Monte_Carlo_integration

where the sum gets replaced by an integral for pdfs[12]. The KL divergence is only defined if P and Q both sum to 1 and if $q(x) = 0$ implies $p(x) = 0$ for all x(absolute continuity). If the quantity $0 \ln 0$ appears in the formula, it is interpreted as zero because $\lim_{x \to 0} x \ln x$. We can rewrite this as

$$\begin{aligned} \mathbb{KL}(p||q) &\triangleq \sum_x p(x) \log_2 p(x) - \sum_{k=1}^{K} p(x) \log_2 q(x) \\ &= \mathbb{H}(p,q) - \mathbb{H}(p) \end{aligned} \tag{2.52}$$

where $\mathbb{H}(p,q)$ is called the **cross entropy**,

$$\mathbb{H}(p,q) = -\sum_x p(x) \log_2 q(x) \tag{2.53}$$

One can show (Cover and Thomas 2006) that the cross entropy is the average number of bits needed to encode data coming from a source with distribution p when we use model q to define our codebook. Hence the "regular" entropy $\mathbb{H}(p) = \mathbb{H}(p,p)$, defined in section §2.8.1,is the expected number of bits if we use the true model, so the KL divergence is the diference between these. In other words, the KL divergence is the average number of *extra* bits needed to encode the data, due to the fact that we used distribution q to encode the data instead of the true distribution p.

The "extra number of bits" interpretation should make it clear that $\mathbb{KL}(p||q) \geq 0$, and that the KL is only equal to zero if $q = p$. We now give a proof of this important result.

Theorem 2.1. ***(Information inequality)*** $\mathbb{KL}(p||q) \geq 0$ *with equality iff* $p = q$.

One important consequence of this result is that *the discrete distribution with the maximum entropy is the uniform distribution*.

2.8.3 Mutual information

Definition 2.9. Mutual information or **MI**, is defined as follows:

$$\begin{aligned} \mathbb{I}(X;Y) &\triangleq \mathbb{KL}(P(X,Y)||P(X)P(Y)) \\ &= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \end{aligned} \tag{2.54}$$

We have $\mathbb{I}(X;Y) \geq 0$ with equality if $P(X,Y) = P(X)P(Y)$. That is, the MI is zero if the variables are independent.

To gain insight into the meaning of MI, it helps to re-express it in terms of joint and conditional entropies. One can show that the above expression is equivalent to the following:

$$\mathbb{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y) \tag{2.55}$$
$$= \mathbb{H}(Y) - \mathbb{H}(Y|X) \tag{2.56}$$
$$= \mathbb{H}(X) + \mathbb{H}(Y) - \mathbb{H}(X,Y) \tag{2.57}$$
$$= \mathbb{H}(X,Y) - \mathbb{H}(X|Y) - \mathbb{H}(Y|X) \tag{2.58}$$

where $\mathbb{H}(X)$ and $\mathbb{H}(Y)$ are the **marginal entropies**, $\mathbb{H}(X|Y)$ and $\mathbb{H}(Y|X)$ are the **conditional entropies**, and $\mathbb{H}(X,Y)$ is the **joint entropy** of X and Y, see Fig. 2.10[13].

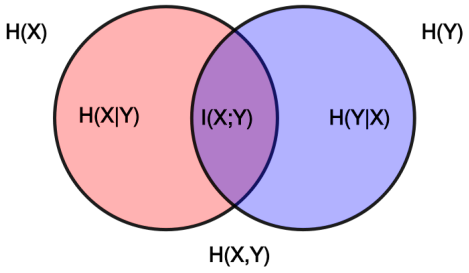


Fig. 2.10: Individual $\mathbb{H}(X), \mathbb{H}(Y)$, joint $\mathbb{H}(X,Y)$, and conditional entropies for a pair of correlated subsystems X,Y with mutual information $\mathbb{I}(X;Y)$.

Intuitively, we can interpret the MI between X and Y as the reduction in uncertainty about X after observing Y, or, by symmetry, the reduction in uncertainty about Y after observing X.

A quantity which is closely related to MI is the **pointwise mutual information** or **PMI**. For two events (not random variables) x and y, this is defined as

$$PMI(x,y) \triangleq \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)} \tag{2.59}$$

This measures the discrepancy between these events occuring together compared to what would be expected by chance. Clearly the MI of X and Y is just the expected value of the PMI. Interestingly, we can rewrite the PMI as follows:

$$PMI(x,y) = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)} \tag{2.60}$$

This is the amount we learn from updating the prior $p(x)$ into the posterior $p(x|y)$, or equivalently, updating the prior $p(y)$ into the posterior $p(y|x)$.

[12] The KL divergence is not a distance, since it is asymmetric. One symmetric version of the KL divergence is the **Jensen-Shannon divergence**, defined as $JS(p_1,p_2) = 0.5\mathbb{KL}(p_1||q) + 0.5\mathbb{KL}(p_2||q)$,where $q = 0.5p_1 + 0.5p_2$

[13] http://en.wikipedia.org/wiki/Mutual_information

Chapter 3
Generative models for discrete data

3.1 Generative classifier

$$p(y=c|\vec{x},\vec{\theta}) = \frac{p(y=c|\vec{\theta})p(\vec{x}|y=c,\vec{\theta})}{\sum_{c'} p(y=c'|\vec{\theta})p(\vec{x}|y=c',\vec{\theta})} \tag{3.1}$$

This is called a **generative classifier**, since it specifies how to generate the data using the **class conditional density** $p(\vec{x}|y=c)$ and the class prior $p(y=c)$. An alternative approach is to directly fit the class posterior, $p(y=c|\vec{x})$;this is known as a **discriminative classifier**.

3.2 Bayesian concept learning

Psychological research has shown that people can learn concepts from positive examples alone (Xu and Tenenbaum 2007).

We can think of learning the meaning of a word as equivalent to **concept learning**, which in turn is equivalent to binary classification. To see this, define $f(\vec{x}) = 1$ if x is an example of the concept C, and $f(\vec{x}) = 0$ otherwise. Then the goal is to learn the indicator function f, which just defines which elements are in the set C.

3.2.1 Likelihood

$$p(\mathcal{D}|h) \triangleq \left(\frac{1}{\text{size}(h)}\right)^N = \left(\frac{1}{|h|}\right)^N \tag{3.2}$$

This crucial equation embodies what Tenenbaum calls the **size principle**, which means the model favours the simplest (smallest) hypothesis consistent with the data. This is more commonly known as **Occam's razor**[14].

3.2.2 Prior

The prior is decided by human, not machines, so it is subjective. The subjectivity of the prior is controversial. For example, that a child and a math professor will reach different answers. In fact, they presumably not only have different priors, but also different hypothesis spaces. However, we can finesse that by defining the hypothesis space of the child and the math professor to be the same, and then setting the child's prior weight to be zero on certain "advanced" concepts. Thus there is no sharp distinction between the prior and the hypothesis space.

However, the prior is the mechanism by which background knowledge can be brought to bear on a problem. Without this, rapid learning (i.e., from small samples sizes) is impossible.

3.2.3 Posterior

The posterior is simply the likelihood times the prior, normalized.

$$p(h|\mathcal{D}) \triangleq \frac{p(\mathcal{D}|h)p(h)}{\sum_{h'\in\mathcal{H}} p(\mathcal{D}|h')p(h')} = \frac{\mathbb{I}(\mathcal{D}\in h)p(h)}{\sum_{h'\in\mathcal{H}} \mathbb{I}(\mathcal{D}\in h')p(h')} \tag{3.3}$$

where $\mathbb{I}(\mathcal{D}\in h)p(h)$ is 1 **iff**(iff and only if) all the data are in the extension of the hypothesis h.

In general, when we have enough data, the posterior $p(h|\mathcal{D})$ becomes peaked on a single concept, namely the MAP estimate, i.e.,

$$p(h|\mathcal{D}) \to \hat{h}^{MAP} \tag{3.4}$$

where $\hat{h}^{MAP}$ is the posterior mode,

$$\begin{aligned}\hat{h}^{MAP} &\triangleq \arg\max_h p(h|\mathcal{D}) = \arg\max_h p(\mathcal{D}|h)p(h) \\ &= \arg\max_h [\log p(\mathcal{D}|h) + \log p(h)]\end{aligned} \tag{3.5}$$

Since the likelihood term depends exponentially on N, and the prior stays constant, as we get more and more data, the MAP estimate converges towards the **maximum likelihood estimate** or **MLE**:

$$\hat{h}^{MLE} \triangleq \arg\max_h p(\mathcal{D}|h) = \arg\max_h \log p(\mathcal{D}|h) \tag{3.6}$$

In other words, if we have enough data, we see that the **data overwhelms the prior**.

3.2.4 Posterior predictive distribution

The concept of **posterior predictive distribution**[15] is normally used in a Bayesian context, where it makes use of the entire posterior distribution of the parameters given the observed data to yield a probability distribution over an interval rather than simply a point estimate.

[14] http://en.wikipedia.org/wiki/Occam%27s_razor

[15] http://en.wikipedia.org/wiki/Posterior_predictive_distribution

$$p(\tilde{\vec{x}}|\mathcal{D}) \triangleq \mathbb{E}_{h|\mathcal{D}}[p(\tilde{\vec{x}}|h)] = \begin{cases} \sum_h p(\tilde{\vec{x}}|h)p(h|\mathcal{D}) \\ \int p(\tilde{\vec{x}}|h)p(h|\mathcal{D})\mathrm{d}h \end{cases} \tag{3.7}$$

This is just a weighted average of the predictions of each individual hypothesis and is called **Bayes model averaging**(Hoeting et al. 1999).

3.3 The beta-binomial model

3.3.1 Likelihood

Given $X \sim \mathrm{Bin}(\theta)$, the likelihood of $\mathcal{D}$ is given by

$$p(\mathcal{D}|\theta) = \mathrm{Bin}(N_1|N,\theta) \tag{3.8}$$

3.3.2 Prior

$$\mathrm{Beta}(\theta|a,b) \propto \theta^{a-1}(1-\theta)^{b-1} \tag{3.9}$$

The parameters of the prior are called **hyper-parameters**.

3.3.3 Posterior

$$\begin{aligned} p(\theta|\mathcal{D}) &\propto \mathrm{Bin}(N_1|N_1+N_0,\theta)\mathrm{Beta}(\theta|a,b) \\ &= \mathrm{Beta}(\theta|N_1+a,N_0b) \end{aligned} \tag{3.10}$$

Note that updating the posterior sequentially is equivalent to updating in a single batch. To see this, suppose we have two data sets $\mathcal{D}_a$ and $\mathcal{D}_b$ with sufficient statistics N_1^a, N_0^a and N_1^b, N_0^b. Let $N_1 = N_1^a + N_1^b$ and $N_0 = N_0^a + N_0^b$ be the sufficient statistics of the combined datasets. In batch mode we have

$$\begin{aligned} p(\theta|\mathcal{D}_a,\mathcal{D}_b) &= p(\theta,\mathcal{D}_b|\mathcal{D}_a)p(\mathcal{D}_a) \\ &\propto p(\theta,\mathcal{D}_b|\mathcal{D}_a) \\ &= p(\mathcal{D}_b,\theta|\mathcal{D}_a) \\ &= p(\mathcal{D}_b|\theta)p(\theta|\mathcal{D}_a) \\ &\text{Combine Equation 3.10 and 2.19} \\ &= \mathrm{Bin}(N_1^b|\theta,N_1^b+N_0^b)\mathrm{Beta}(\theta|N_1^a+a,N_0^a+b) \\ &= \mathrm{Beta}(\theta|N_1^a+N_1^b+a,N_0^a+N_0^b+b) \end{aligned}$$

This makes Bayesian inference particularly well-suited to **online learning**, as we will see later.

3.3.3.1 Posterior mean and mode

From Table 2.7, the posterior mean is given by

$$\bar{\theta} = \frac{a+N_1}{a+b+N} \tag{3.11}$$

The mode is given by

$$\hat{\theta}_{MAP} = \frac{a+N_1-1}{a+b+N-2} \tag{3.12}$$

If we use a uniform prior, then the MAP estimate reduces to the MLE,

$$\hat{\theta}_{MLE} = \frac{N_1}{N} \tag{3.13}$$

We will now show that the posterior mean is convex combination of the prior mean and the MLE, which captures the notion that the posterior is a compromise between what we previously believed and what the data is telling us.

3.3.3.2 Posterior variance

The mean and mode are point estimates, but it is useful to know how much we can trust them. The variance of the posterior is one way to measure this. The variance of the Beta posterior is given by

$$\mathrm{var}(\theta|\mathcal{D}) = \frac{(a+N_1)(b+N_0)}{(a+N_1+b+N_0)^2(a+N_1+b+N_0+1)} \tag{3.14}$$

We can simplify this formidable expression in the case that $N \gg a,b$, to get

$$\mathrm{var}(\theta|\mathcal{D}) \approx \frac{N_1N_0}{NNN} = \frac{\hat{\theta}_{MLE}(1-\hat{\theta}_{MLE})}{N} \tag{3.15}$$

3.3.4 Posterior predictive distribution

So far, we have been focusing on inference of the unknown parameter(s). Let us now turn our attention to prediction of future observable data.

Consider predicting the probability of heads in a single future trial under a $\mathrm{Beta}(a,b)$posterior. We have

$$\begin{aligned} p(\tilde{x}|\mathcal{D}) &= \int_0^1 p(\tilde{x}|\theta)p(\theta|\mathcal{D})\mathrm{d}\theta \\ &= \int_0^1 \theta\mathrm{Beta}(\theta|a,b)\mathrm{d}\theta \\ &= \mathbb{E}[\theta|\mathcal{D}] = \frac{a}{a+b} \end{aligned} \tag{3.16}$$

3.3.4.1 Overfitting and the black swan paradox

Let us now derive a simple Bayesian solution to the problem. We will use a uniform prior, so $a=b=1$. In this case, plugging in the posterior mean gives **Laplace's rule of succession**

$$p(\tilde{x}|\mathcal{D}) = \frac{N_1+1}{N_0+N_1+1} \tag{3.17}$$

This justifies the common practice of adding 1 to the empirical counts, normalizing and then plugging them in, a technique known as **add-one smoothing**. (Note that plugging in

the MAP parameters would not have this smoothing effect, since the mode becomes the MLE if $a = b = 1$, see Section 3.3.3.1.)

3.3.4.2 Predicting the outcome of multiple future trials

Suppose now we were interested in predicting the number of heads, $\tilde{x}$, in M future trials. This is given by

$$p(\tilde{x}|\mathcal{D}) = \int_0^1 \text{Bin}(\tilde{x}|M,\theta)\text{Beta}(\theta|a,b)\text{d}\theta \tag{3.18}$$

$$= \binom{M}{\tilde{x}} \frac{1}{B(a,b)} \int_0^1 \theta^{\tilde{x}}(1-\theta)^{M-\tilde{x}}\theta^{a-1}(1-\theta)^{b-1}\text{d}\theta \tag{3.19}$$

We recognize the integral as the normalization constant for a $\text{Beta}(a+\tilde{x}, M-\tilde{x}+b)$ distribution. Hence

$$\int_0^1 \theta^{\tilde{x}}(1-\theta)^{M-\tilde{x}}\theta^{a-1}(1-\theta)^{b-1}\text{d}\theta = B(\tilde{x}+a, M-\tilde{x}+b) \tag{3.20}$$

Thus we find that the posterior predictive is given by the following, known as the (compound) **beta-binomial distribution**:

$$Bb(x|a,b,M) \triangleq \binom{M}{x} \frac{B(x+a, M-x+b)}{B(a,b)} \tag{3.21}$$

This distribution has the following mean and variance

$$\text{mean} = M\frac{a}{a+b}, \text{ var} = \frac{Mab}{(a+b)^2}\frac{a+b+M}{a+b+1} \tag{3.22}$$

This process is illustrated in Figure 3.1. We start with a $\text{Beta}(2,2)$ prior, and plot the posterior predictive density after seeing $N_1 = 3$ heads and $N_0 = 17$ tails. Figure 3.1(b) plots a plug-in approximation using a MAP estimate. We see that the Bayesian prediction has longer tails, spreading its probability mass more widely, and is therefore less prone to overfitting and blackswan type paradoxes.

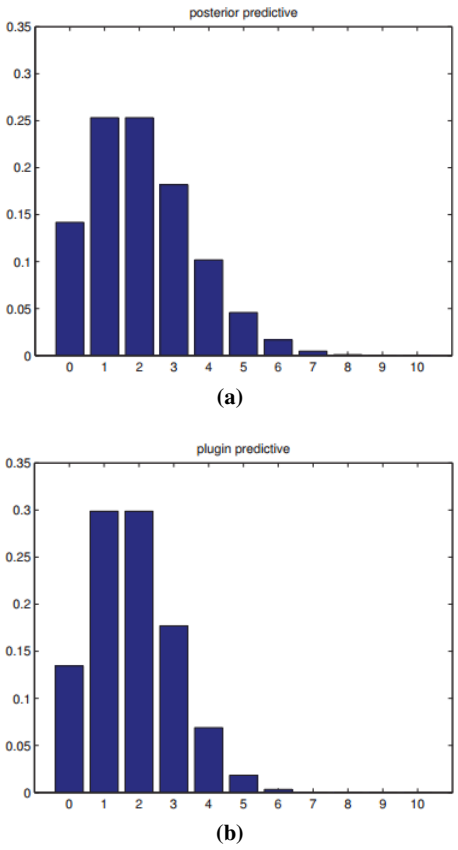


Fig. 3.1: (a) Posterior predictive distributions after seeing $N_1 = 3, N_0 = 17$. (b) MAP estimation.

3.4 The Dirichlet-multinomial model

In the previous section, we discussed how to infer the probability that a coin comes up heads. In this section, we generalize these results to infer the probability that a dice with K sides comes up as face k.

3.4.1 Likelihood

Suppose we observe N dice rolls, $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$, where $x_i \in \{1, 2, \cdots, K\}$. The likelihood has the form

$$p(\mathcal{D}|\vec{\theta}) = \binom{N}{N_1 \cdots N_k} \prod_{k=1}^K \theta_k^{N_k} \quad \text{where } N_k = \sum_{i=1}^N \mathbb{I}(y_i = k) \tag{3.23}$$

almost the same as Equation 2.21.

3.4.2 Prior

$$\text{Dir}(\vec{\theta}|\vec{\alpha}) = \frac{1}{B(\vec{\alpha})} \prod_{k=1}^K \theta_k^{\alpha_k - 1} \mathbb{I}(\vec{\theta} \in S_K) \tag{3.24}$$

3.4.3 Posterior

$$p(\vec{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\vec{\theta})p(\vec{\theta}) \tag{3.25}$$

$$\propto \prod_{k=1}^K \theta_k^{N_k}\theta_k^{\alpha_k - 1} = \prod_{k=1}^K \theta_k^{N_k + \alpha_k - 1} \tag{3.26}$$

$$= \text{Dir}(\vec{\theta}|\alpha_1 + N_1, \cdots, \alpha_K + N_K) \tag{3.27}$$

From Equation 2.38, the MAP estimate is given by

$$\hat{\theta}_k = \frac{N_k + \alpha_k - 1}{N + \alpha_0 - K} \tag{3.28}$$

If we use a uniform prior, $\alpha_k = 1$, we recover the MLE:

$$\hat{\theta}_k = \frac{N_k}{N} \tag{3.29}$$

3.4.4 Posterior predictive distribution

The posterior predictive distribution for a single multinoulli trial is given by the following expression:

$$p(X=j|\mathcal{D}) = \int p(X=j|\vec{\theta})p(\vec{\theta}|\mathcal{D})\mathrm{d}\vec{\theta} \tag{3.30}$$

$$= \int p(X=j|\theta_j)\left[\int p(\vec{\theta}_{-j},\theta_j|\mathcal{D})\mathrm{d}\vec{\theta}_{-j}\right]\mathrm{d}\theta_j \tag{3.31}$$

$$= \int \theta_j p(\theta_j|\mathcal{D})\mathrm{d}\theta_j = \mathbb{E}[\theta_j|\mathcal{D}] = \frac{\alpha_j + N_j}{\alpha_0 + N} \tag{3.32}$$

where $\vec{\theta}_{-j}$ are all the components of $\vec{\theta}$ except θ_j.

The above expression avoids the zero-count problem. In fact, this form of Bayesian smoothing is even more important in the multinomial case than the binary case, since the likelihood of data sparsity increases once we start partitioning the data into many categories.

3.5 Naive Bayes classifiers

Assume the features are **conditionally independent** given the class label, then the class conditional density has the following form

$$p(\vec{x}|y=c,\vec{\theta}) = \prod_{j=1}^{D} p(x_j|y=c,\vec{\theta}_{jc}) \tag{3.33}$$

The resulting model is called a **naive Bayes classifier**(NBC).

The form of the class-conditional density depends on the type of each feature. We give some possibilities below:

- In the case of real-valued features, we can use the Gaussian distribution: $p(\vec{x}|y,\vec{\theta}) = \prod_{j=1}^{D}\mathcal{N}(x_j|\mu_{jc},\sigma_{jc}^2)$, where μ_{jc} is the mean of feature j in objects of class c, and σ_{jc}^2 is its variance.
- In the case of binary features, $x_j \in \{0,1\}$, we can use the Bernoulli distribution: $p(\vec{x}|y,\vec{\theta}) = \prod_{j=1}^{D}\mathrm{Ber}(x_j|\mu_{jc})$, where μ_{jc} is the probability that feature j occurs in class c. This is sometimes called the **multivariate Bernoulli naive Bayes** model. We will see an application of this below.
- In the case of categorical features, $x_j \in \{a_{j1},a_{j2},\cdots,a_{jS_j}\}$, we can use the multinoulli distribution: $p(\vec{x}|y,\vec{\theta}) = \prod_{j=1}^{D}\mathrm{Cat}(x_j|\vec{\mu}_{jc})$, where $\vec{\mu}_{jc}$ is a histogram over the K possible values for x_j in class c.

Obviously we can handle other kinds of features, or use different distributional assumptions. Also, it is easy to mix and match features of different types.

3.5.1 Optimization

We now discuss how to "train" a naive Bayes classifier. This usually means computing the MLE or the MAP estimate for the parameters. However, we will also discuss how to compute the full posterior, $p(\vec{\theta}|\mathcal{D})$.

3.5.1.1 MLE for NBC

The probability for a single data case is given by

$$\begin{aligned} p(\vec{x}_i,y_i|\vec{\theta}) &= p(y_i|\vec{\pi})\prod_j p(x_{ij}|\vec{\theta}_j) \\ &= \prod_c \pi_c^{\mathbb{I}(y_i=c)}\prod_j\prod_c p(x_{ij}|\vec{\theta}_{jc})^{\mathbb{I}(y_i=c)} \end{aligned} \tag{3.34}$$

Hence the log-likelihood is given by

$$p(\mathcal{D}|\vec{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D}\sum_{c=1}^{C}\sum_{i:y_i=c}\log p(x_{ij}|\vec{\theta}_{jc}) \tag{3.35}$$

where $N_c \triangleq \sum_i \mathbb{I}(y_i=c)$ is the number of feature vectors in class c.

We see that this expression decomposes into a series of terms, one concerning $\vec{\pi}$, and DC terms containing the θ_{jc}'s. Hence we can optimize all these parameters separately.

From Equation 3.29, the MLE for the class prior is given by

$$\hat{\pi}_c = \frac{N_c}{N} \tag{3.36}$$

The MLE for θ_{jc}'s depends on the type of distribution we choose to use for each feature.

In the case of binary features, $x_j \in \{0,1\}$, $x_j|y=c \sim \mathrm{Ber}(\theta_{jc})$, hence

$$\hat{\theta}_{jc} = \frac{N_{jc}}{N_c} \tag{3.37}$$

where $N_{jc} \triangleq \sum_{i:y_i=c}\mathbb{I}(y_i=c)$ is the number that feature j occurs in class c.

In the case of categorical features, $x_j \in \{a_{j1},a_{j2},\cdots,a_{jS_j}\}$, $x_j|y=c \sim \mathrm{Cat}(\vec{\theta}_{jc})$, hence

$$\hat{\vec{\theta}}_{jc} = (\frac{N_{j1c}}{N_c},\frac{N_{j2c}}{N_c},\cdots,\frac{N_{jS_j}}{N_c})^T \tag{3.38}$$

where $N_{jkc} \triangleq \sum_{i=1}^{N}\mathbb{I}(x_{ij}=a_{jk},y_i=c)$ is the number that feature $x_j=a_{jk}$ occurs in class c.

3.5.1.2 Bayesian naive Bayes

Use a $\text{Dir}(\vec{\alpha})$ prior for $\vec{\pi}$.

In the case of binary features, use a $\text{Beta}(\beta 0, \beta 1)$ prior for each θ_{jc}; in the case of categorical features, use a $\text{Dir}(\vec{\alpha})$ prior for each $\vec{\theta}_{jc}$. Often we just take $\vec{\alpha} = \vec{1}$ and $\vec{\beta} = \vec{1}$, corresponding to **add-one** or **Laplace smoothing**.

3.5.2 Using the model for prediction

The goal is to compute

$$\begin{aligned} y = f(\vec{x}) &= \arg\max_c P(y=c|\vec{x},\vec{\theta}) \\ &= P(y=c|\vec{\theta}) \prod_{j=1}^{D} P(x_j|y=c,\vec{\theta}) \end{aligned} \tag{3.39}$$

We can the estimate parameters using MLE or MAP, then the posterior predictive density is obtained by simply plugging in the parameters $\hat{\vec{\theta}}$(MLE) or $\hat{\vec{\theta}}$(MAP).

Or we can use BMA, just integrate out the unknown parameters.

3.5.3 The log-sum-exp trick

when using generative classifiers of any kind, computing the posterior over class labels using Equation 3.1 can fail due to **numerical underflow**. The problem is that $p(\vec{x}|y=c)$ is often a very small number, especially if $\vec{x}$ is a high-dimensional vector. This is because we require that $\sum_{\vec{x}} p(\vec{x}|y) = 1$, so the probability of observing any particular high-dimensional vector is small. The obvious solution is to take logs when applying Bayes rule, as follows:

$$\log p(y=c|\vec{x},\vec{\theta}) = b_c - \log\left(\sum_{c'} e^{b_{c'}}\right) \tag{3.40}$$

where $b_c \triangleq \log p(\vec{x}|y=c,\vec{\theta}) + \log p(y=c|\vec{\theta})$.

We can factor out the largest term, and just represent the remaining numbers relative to that. For example,

$$\begin{aligned} \log(e^{-120} + e^{-121}) &= \log(e^{-120}(1+e^{-1})) \\ &= \log(1+e^{-1}) - 120 \end{aligned} \tag{3.41}$$

In general, we have

$$\sum_c e^{b_c} = \log\left[(\sum e^{b_c - B})e^B\right] = \log\left(\sum e^{b_c - B}\right) + B \tag{3.42}$$

where $B \triangleq \max\{b_c\}$.

This is called the **log-sum-exp** trick, and is widely used.

3.5.4 Feature selection using mutual information

Since an NBC is fitting a joint distribution over potentially many features, it can suffer from overfitting. In addition, the run-time cost is $O(D)$, which may be too high for some applications.

One common approach to tackling both of these problems is to perform **feature selection**, to remove "irrelevant" features that do not help much with the classification problem. The simplest approach to feature selection is to evaluate the relevance of each feature separately, and then take the top K,whereKis chosen based on some tradeoff between accuracy and complexity. This approach is known as **variable ranking**, **filtering**, or **screening**.

One way to measure relevance is to use mutual information (Section 2.8.3) between feature X_j and the class label Y

$$\mathbb{I}(X_j, Y) = \sum_{x_j} \sum_y p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)} \tag{3.43}$$

If the features are binary, it is easy to show that the MI can be computed as follows

$$\mathbb{I}_j = \sum_c \left[\theta_{jc}\pi_c \log \frac{\theta_{jc}}{\theta_j} + (1-\theta_{jc})\pi_c \log \frac{1-\theta_{jc}}{1-\theta_j}\right] \tag{3.44}$$

where $\pi_c = p(y=c)$, $\theta_{jc} = p(x_j = 1|y=c)$, and $\theta_j = p(x_j = 1) = \sum_c \pi_c \theta_{jc}$.

3.5.5 Classifying documents using bag of words

Document classification is the problem of classifying text documents into different categories.

3.5.5.1 Bernoulli product model

One simple approach is to represent each document as a binary vector, which records whether each word is present or not, so $x_{ij} = 1$ iff word j occurs in document i, otherwise $x_{ij} = 0$. We can then use the following class conditional density:

$$\begin{aligned} p(\vec{x}_i|y_i=c,\vec{\theta}) &= \prod_{j=1}^{D} \text{Ber}(x_{ij}|\theta_{jc}) \\ &= \prod_{j=1}^{D} \theta_{jc}^{x_{ij}} (1-\theta_{jc})^{1-x_{ij}} \end{aligned} \tag{3.45}$$

This is called the **Bernoulli product model**, or the **binary independence model**.

3.5.5.2 Multinomial document classifier

However, ignoring the number of times each word occurs in a document loses some information (McCallum and Nigam 1998). A more accurate representation counts the number of occurrences of each word. Specifically, let $\vec{x}_i$ be a vector of counts for document i, so $x_{ij} \in \{0, 1, \cdots, N_i\}$, where N_i is the number of terms in document i(so $\sum_{j=1}^{D} x_{ij} = N_i$). For the class conditional densities, we can use a multinomial distribution:

$$p(\vec{x}_i|y_i = c, \vec{\theta}) = \text{Mu}(\vec{x}_i|N_i, \vec{\theta}_c) = \frac{N_i!}{\prod_{j=1}^{D} x_{ij}!} \prod_{j=1}^{D} \theta_{jc}^{x_{ij}} \quad (3.46)$$

where we have implicitly assumed that the document length N_i is independent of the class. Here $heta_{jc}$ is the probability of generating word j in documents of class c; these parameters satisfy the constraint that $\sum_{j=1}^{D} \theta_{jc} = 1$ for each class c.

Although the multinomial classifier is easy to train and easy to use at test time, it does not work particularly well for document classification. One reason for this is that it does not take into account the **burstiness** of word usage. This refers to the phenomenon that most words never appear in any given document, but if they do appear once, they are likely to appear more than once, i.e., words occur in bursts.

The multinomial model cannot capture the burstiness phenomenon. To see why, note that Equation 3.46 has the form $\theta_{jc}^{x_{ij}}$, and since $\theta_{jc} \ll 1$ for rare words, it becomes increasingly unlikely to generate many of them. For more frequent words, the decay rate is not as fast. To see why intuitively, note that the most frequent words are function words which are not specific to the class, such as "and", "the", and "but"; the chance of the word "and" occuring is pretty much the same no matter how many time it has previously occurred (modulo document length), so the independence assumption is more reasonable for common words. However, since rare words are the ones that matter most for classification purposes, these are the ones we want to model the most carefully.

3.5.5.3 DCM model

Various ad hoc heuristics have been proposed to improve the performance of the multinomial document classifier (Rennie et al. 2003). We now present an alternative class conditional density that performs as well as these ad hoc methods, yet is probabilistically sound (Madsen et al. 2005).

Suppose we simply replace the multinomial class conditional density with the **Dirichlet Compound Multinomial** or **DCM** density, defined as follows:

$$\begin{aligned} p(\vec{x}_i|y_i = c, \vec{\alpha}) &= \int \text{Mu}(\vec{x}_i|N_i, \vec{\theta}_c)\text{Dir}(\vec{\theta}_c|\vec{\alpha}_c) \\ &= \frac{N_i!}{\prod_{j=1}^{D} x_{ij}!} \prod_{j=1}^{D} \frac{B(\vec{x}_i + \vec{\alpha}_c)}{B(\vec{\alpha}_c)} \end{aligned} \quad (3.47)$$

(This equation is derived in Equation TODO.) Surprisingly this simple change is all that is needed to capture the burstiness phenomenon. The intuitive reason for this is as follows: After seeing one occurence of a word, say wordj, the posterior counts on hetaj gets updated, making another occurence of wordjmore likely. By contrast, if hetaj is fixed, then the occurences of each word are independent. The multinomial model corresponds to drawing a ball from an urn with Kcolors of ball, recording its color, and then replacing it. By contrast, the DCM model corresponds to drawing a ball, recording its color, and then replacing it with one additional copy; this is called the **Polya urn**.

Using the DCM as the class conditional density gives much better results than using the multinomial, and has performance comparable to state of the art methods, as described in (Madsen et al. 2005). The only disadvantage is that fitting the DCM model is more complex; see (Minka 2000e; Elkan 2006) for the details.

Chapter 4
Gaussian Models

In this chapter, we discuss the **multivariate Gaussian** or **multivariate normal(MVN)**, which is the most widely used joint probability density function for continuous variables. It will form the basis for many of the models we will encounter in later chapters.

4.1 Basics

Recall from Section 2.5.2 that the pdf for an MVN in D dimensions is defined by the following:

$$\mathcal{N}(\vec{x}|\vec{\mu},\Sigma) \triangleq \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right] \tag{4.1}$$

The expression inside the exponent is the Mahalanobis distance between a data vector $\vec{x}$ and the mean vector $\vec{\mu}$, We can gain a better understanding of this quantity by performing an **eigendecomposition** of $\vec{\Sigma}$. That is, we write $\vec{\Sigma} = \vec{U}\vec{\Lambda}\vec{U}^T$, where $\vec{U}$ is an orthonormal matrix of eigenvectors satsifying $\vec{U}^T\vec{U} = \vec{I}$, and $\vec{\Lambda}$ is a diagonal matrix of eigenvalues. Using the eigendecomposition, we have that

$$\vec{\Sigma}^{-1} = \vec{U}^{-T}\vec{\Lambda}^{-1}\vec{U}^{-1} = \vec{U}\vec{\Lambda}^{-1}\vec{U}^T = \sum_{i=1}^{D}\frac{1}{\lambda_i}\vec{u}_i\vec{u}_i^T \tag{4.2}$$

where $\vec{u}_i$ is the i'th column of $\vec{U}$, containing the i'th eigenvector. Hence we can rewrite the Mahalanobis distance as follows:

$$(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu}) = (\vec{x}-\vec{\mu})^T\left(\sum_{i=1}^{D}\frac{1}{\lambda_i}\vec{u}_i\vec{u}_i^T\right)(\vec{x}-\vec{\mu}) \tag{4.3}$$

$$= \sum_{i=1}^{D}\frac{1}{\lambda_i}(\vec{x}-\vec{\mu})^T\vec{u}_i\vec{u}_i^T(\vec{x}-\vec{\mu}) \tag{4.4}$$

$$= \sum_{i=1}^{D}\frac{y_i^2}{\lambda_i} \tag{4.5}$$

where $y_i \triangleq \vec{u}_i^T(\vec{x}-\vec{\mu})$. Recall that the equation for an ellipse in 2d is

$$\frac{y_1^2}{\lambda_1} + \frac{y_2^2}{\lambda_2} = 1 \tag{4.6}$$

Hence we see that the contours of equal probability density of a Gaussian lie along ellipses. This is illustrated in Figure 4.1. The eigenvectors determine the orientation of the ellipse, and the eigenvalues determine how elogonated it is.

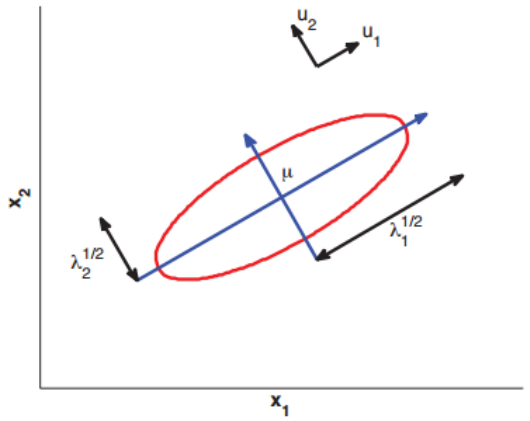


Fig. 4.1: Visualization of a 2 dimensional Gaussian density. The major and minor axes of the ellipse are defined by the first two eigenvectors of the covariance matrix, namely $\vec{u}_1$ and $\vec{u}_2$. Based on Figure 2.7 of (Bishop 2006a)

In general, we see that the Mahalanobis distance corresponds to Euclidean distance in a transformed coordinate system, where we shift by $\vec{\mu}$ and rotate by $\vec{U}$.

4.1.1 MLE for a MVN

Theorem 4.1. *(**MLE for a MVN**) If we have N iid samples $\vec{x}_i \sim \mathcal{N}(\vec{\mu},\vec{\Sigma})$, then the MLE for the parameters is given by*

$$\bar{\vec{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\vec{x}_i \triangleq \bar{\vec{x}} \tag{4.7}$$

$$\bar{\vec{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}(\vec{x}_i-\bar{\vec{x}})(\vec{x}_i-\bar{\vec{x}})^T \tag{4.8}$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N}\vec{x}_i\vec{x}_i^T\right) - \bar{\vec{x}}\bar{\vec{x}}^T \tag{4.9}$$

4.1.2 Maximum entropy derivation of the Gaussian *

In this section, we show that the multivariate Gaussian is the distribution with maximum entropy subject to having a specified mean and covariance (see also Section TODO). This is one reason the Gaussian is so widely used: the first two moments are usually all that we can reliably estimate from data, so we want a distribution that captures these properties, but otherwise makes as few addtional assumptions as possible.

To simplify notation, we will assume the mean is zero. The pdf has the form

$$f(\vec{x}) = \frac{1}{Z}\exp\left(-\frac{1}{2}\vec{x}^T\vec{\Sigma}^{-1}\vec{x}\right) \tag{4.10}$$

4.2 Gaussian discriminant analysis

One important application of MVNs is to define the the class conditional densities in a generative classifier, i.e.,

$$p(\vec{x}|y=c,\vec{\theta}) = \mathcal{N}(\vec{x}|\vec{\mu}_c,\vec{\Sigma}_c) \tag{4.11}$$

The resulting technique is called (Gaussian) **discriminant analysis** or **GDA** (even though it is a generative, not discriminative, classifier — see Section TODO for more on this distinction). If $\vec{\Sigma}_c$ is diagonal, this is equivalent to naive Bayes.

We can classify a feature vector using the following decision rule, derived from Equation 3.1:

$$y = \arg\max_c \left[\log p(y=c|\vec{\pi}) + \log p(\vec{x}|\vec{\theta})\right] \tag{4.12}$$

When we compute the probability of $\vec{x}$ under each class conditional density, we are measuring the distance from $\vec{x}$ to the center of each class, $\vec{\mu}_c$, using Mahalanobis distance. This can be thought of as a **nearest centroids classifier**.

As an example, Figure 4.2 shows two Gaussian class-conditional densities in 2d, representing the height and weight of men and women. We can see that the features are correlated, as is to be expected (tall people tend to weigh more). The ellipses for each class contain 95% of the probability mass. If we have a uniform prior over classes, we can classify a new test vector as follows:

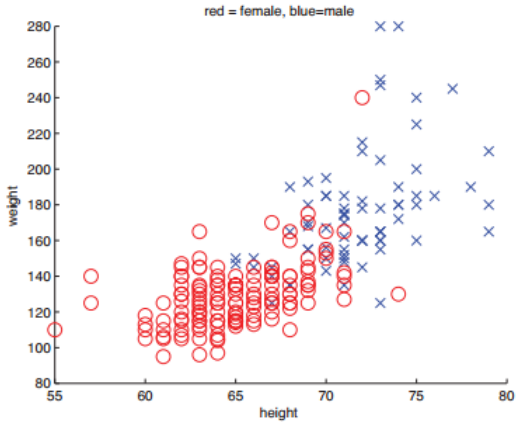
$$y = \arg\max_c (\vec{x}-\vec{\mu}_c)^T\vec{\Sigma}_c^{-1}(\vec{x}-\vec{\mu}_c) \tag{4.13}$$

4.2.1 Quadratic discriminant analysis (QDA)

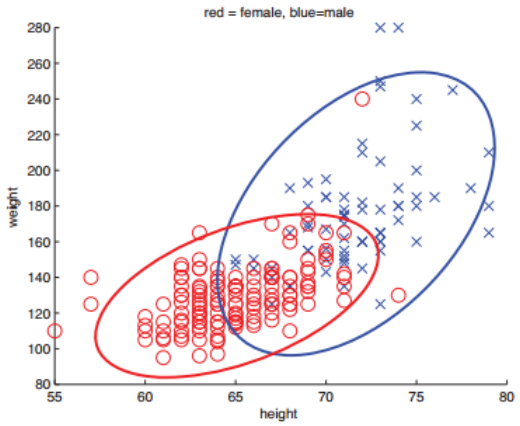
By plugging in the definition of the Gaussian density to Equation 3.1, we can get

$$p(y|\vec{x},\vec{\theta}) = \frac{\pi_c|2\pi\vec{\Sigma}_c|^{-\frac{1}{2}}\exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu})^T\vec{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right]}{\sum_{c'}\pi_{c'}|2\pi\vec{\Sigma}_{c'}|^{-\frac{1}{2}}\exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu})^T\vec{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right]} \tag{4.14}$$

Thresholding this results in a quadratic function of $\vec{x}$. The result is known as quadratic discriminant analysis(QDA). Figure 4.3 gives some examples of what the decision boundaries look like in 2D.



(a)



(b)

Fig. 4.2: (a) Height/weight data. (b) Visualization of 2d Gaussians fit to each class. 95% of the probability mass is inside the ellipse.

4.2.2 Linear discriminant analysis (LDA)

We now consider a special case in which the covariance matrices are **tied** or **shared** across classes,$\vec{\Sigma}_c = \vec{\Sigma}$. In this case, we can simplify Equation 4.14 as follows:

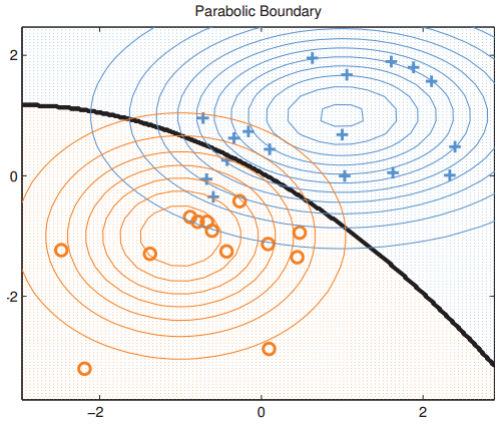
$$\begin{aligned} p(y|\vec{x},\vec{\theta}) &\propto \pi_c\exp\left(\vec{\mu}_c\vec{\Sigma}^{-1}\vec{x} - \frac{1}{2}\vec{x}^T\vec{\Sigma}^{-1}\vec{x} - \frac{1}{2}\vec{\mu}_c^T\vec{\Sigma}^{-1}\vec{\mu}_c\right) \\ &= \exp\left(\vec{\mu}_c\vec{\Sigma}^{-1}\vec{x} - \frac{1}{2}\vec{\mu}_c^T\vec{\Sigma}^{-1}\vec{\mu}_c + \log\pi_c\right) \\ &\quad \exp\left(-\frac{1}{2}\vec{x}^T\vec{\Sigma}^{-1}\vec{x}\right) \\ &\propto \exp\left(\vec{\mu}_c\vec{\Sigma}^{-1}\vec{x} - \frac{1}{2}\vec{\mu}_c^T\vec{\Sigma}^{-1}\vec{\mu}_c + \log\pi_c\right) \end{aligned} \tag{4.15}$$

Since the quadratic term $\vec{x}^T\vec{\Sigma}^{-1}\vec{x}$ is independent of c, it will cancel out in the numerator and denominator. If we define

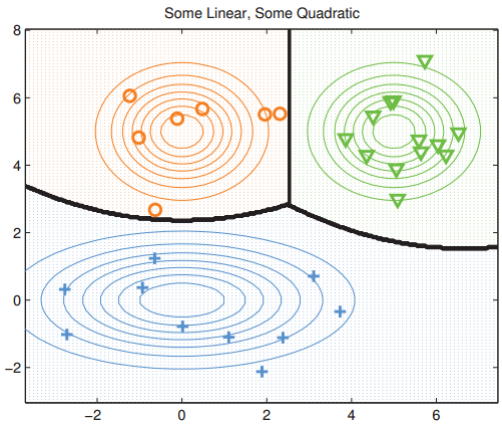
$$\gamma_c \triangleq -\frac{1}{2}\vec{\mu}_c^T\vec{\Sigma}^{-1}\vec{\mu}_c + \log\pi_c \tag{4.16}$$

$$\vec{\beta}_c \triangleq \vec{\Sigma}^{-1}\vec{\mu}_c \tag{4.17}$$

then we can write



(a)



(b)

Fig. 4.3: Quadratic decision boundaries in 2D for the 2 and 3 class case.

$$p(y|\vec{x},\vec{\theta}) = \frac{e^{\vec{\beta}_c^T \vec{x}+\gamma_c}}{\sum_{c'} e^{\vec{\beta}_{c'}^T \vec{x}+\gamma_{c'}}} \triangleq \sigma(\vec{\eta},c) \tag{4.18}$$

where $\vec{\eta} \triangleq (e^{\vec{\beta}_1^T \vec{x}}+\gamma_1,\cdots,e^{\vec{\beta}_C^T \vec{x}}+\gamma_C)$, $\sigma()$ is the **softmax activation function**[16], defined as follows:

$$\sigma(\vec{q},i) \triangleq \frac{\exp(q_i)}{\sum_{j=1}^n \exp(q_j)} \tag{4.19}$$

When parameterized by some constant, $\alpha > 0$, the following formulation becomes a smooth, differentiable approximation of the maximum function:

$$\mathcal{S}_\alpha(\vec{x}) = \frac{\sum_{j=1}^D x_j e^{\alpha x_j}}{\sum_{j=1}^D e^{\alpha x_j}} \tag{4.20}$$

$\mathcal{S}_\alpha$ has the following properties:

1. $\mathcal{S}_\alpha \to \max$ as $\alpha \to \infty$
2. $\mathcal{S}_0$ is the average of its inputs
3. $\mathcal{S}_\alpha \to \min$ as $\alpha \to -\infty$

Note that the softmax activation function comes from the area of statistical physics, where it is common to use the **Boltzmann distribution**, which has the same form as the softmax activation function.

An interesting property of Equation 4.18 is that, if we take logs, we end up with a linear function of $\vec{x}$. (The reason it is linear is because the $\vec{x}^T \vec{\Sigma}^{-1} \vec{x}$ cancels from the numerator and denominator.) Thus the decision boundary between any two classes, says c and c', will be a straight line. Hence this technique is called **linear discriminant analysis** or **LDA**.

An alternative to fitting an LDA model and then deriving the class posterior is to directly fit $p(y|\vec{x},\vec{W}) = \text{Cat}(y|\vec{W}\vec{x})$ for some $C \times D$ weight matrix $\vec{W}$. This is called **multi-class logistic regression**, or **multinomial logistic regression**. We will discuss this model in detail in Section TODO. The difference between the two approaches is explained in Section TODO.

4.2.3 Two-class LDA

To gain further insight into the meaning of these equations, let us consider the binary case. In this case, the posterior is given by

$$p(y=1|\vec{x},\vec{\theta}) = \frac{e^{\vec{\beta}_1^T \vec{x}+\gamma_1}}{e^{\vec{\beta}_0^T \vec{x}+\gamma_0}+e^{\vec{\beta}_1^T \vec{x}+\gamma_1}}) \tag{4.21}$$

$$= \frac{1}{1+e^{(\vec{\beta}_0-\vec{\beta}_1)^T \vec{x}+(\gamma_0-\gamma_1)}} \tag{4.22}$$

$$= \text{sigm}((\vec{\beta}_1-\vec{\beta}_0)^T \vec{x}+(\gamma_0-\gamma_1)) \tag{4.23}$$

where $\text{sigm}(x)$ refers to the sigmoid function[17].

Now

$$\gamma_1-\gamma_0 = -\frac{1}{2}\vec{\mu}_1^T \vec{\Sigma}^{-1}\vec{\mu}_1 + \frac{1}{2}\vec{\mu}_0^T \vec{\Sigma}^{-1}\vec{\mu}_0 + \log(\pi_1/\pi_0) \tag{4.24}$$

$$= -\frac{1}{2}(\vec{\mu}_1-\vec{\mu}_0)^T \vec{\Sigma}^{-1}(\vec{\mu}_1+\vec{\mu}_0) + \log(\pi_1/\pi_0) \tag{4.25}$$

So if we define

$$\vec{w} = \vec{\beta}_1-\vec{\beta}_0 = \vec{\Sigma}^{-1}(\vec{\mu}_1-\vec{\mu}_0) \tag{4.26}$$

$$\vec{x}_0 = \frac{1}{2}(\vec{\mu}_1+\vec{\mu}_0) - (\vec{\mu}_1-\vec{\mu}_0)\frac{\log(\pi_1/\pi_0)}{(\vec{\mu}_1-\vec{\mu}_0)^T \vec{\Sigma}^{-1}(\vec{\mu}_1-\vec{\mu}_0)} \tag{4.27}$$

then we have $\vec{w}^T \vec{x}_0 = -(\gamma_1-\gamma_0)$, and hence

$$p(y=1|\vec{x},\vec{\theta}) = \text{sigm}(\vec{w}^T(\vec{x}-\vec{x}_0)) \tag{4.28}$$

(This is closely related to logistic regression, which we will discuss in Section TODO.) So the final decision rule is as

[16] http://en.wikipedia.org/wiki/Softmax_activation_function

[17] http://en.wikipedia.org/wiki/Sigmoid_function

follows: shift $\vec{x}$ by $\vec{x}_0$, project onto the line $\vec{w}$, and see if the result is positive or negative.

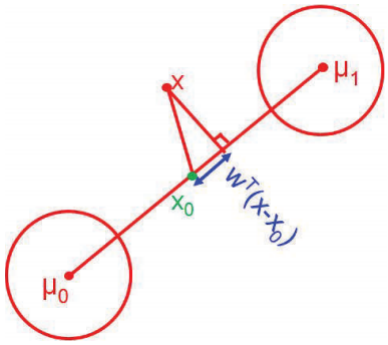


Fig. 4.4: Geometry of LDA in the 2 class case where $\vec{\Sigma}_1 = \vec{\Sigma}_2 = \vec{I}$.

If $\vec{\Sigma} = \sigma^2 \vec{I}$, then $\vec{w}$ is in the direction of $\vec{\mu}_1 - \vec{\mu}_0$. So we classify the point based on whether its projection is closer to $\vec{\mu}_0$ or $\vec{\mu}_1$. This is illustrated in Figure 4.4. Furthemore, if $\vec{\pi}_1 = \vec{\pi}_0$, then $\vec{x}_0 = \frac{1}{2}(\vec{\mu}_1 + \vec{\mu}_0)$, which is half way between the means. If we make $\vec{\pi}_1 > \vec{\pi}_0$, then $\vec{x}_0$ gets closer to $\vec{\mu}_0$, so more of the line belongs to class 1 a *priori*. Conversely if $\vec{\pi}_1 < \vec{\pi}_0$, the boundary shifts right. Thus we see that the class prior, π_c, just changes the decision threshold, and not the overall geometry, as we claimed above. (A similar argument applies in the multi-class case.)

The magnitude of $\vec{w}$ determines the steepness of the logistic function, and depends on how well-separated the means are, relative to the variance. In psychology and signal detection theory, it is common to define the **discriminability** of a signal from the background noise using a quantity called **d-prime**:

$$d' \triangleq \frac{\mu_1 - \mu_0}{\sigma} \tag{4.29}$$

where μ_1 is the mean of the signal and μ_0 is the mean of the noise, and σ is the standard deviation of the noise. If d' is large, the signal will be easier to discriminate from the noise.

4.2.4 MLE for discriminant analysis

The log-likelihood function is as follows:

$$p(\mathcal{D}|\vec{\theta}) = \sum_{c=1}^{C} \sum_{i:y_i=c} \log \pi_c + \sum_{c=1}^{C} \sum_{i:y_i=c} \log \mathcal{N}(\vec{x}_i|\vec{\mu}_c, \vec{\Sigma}_c) \tag{4.30}$$

The MLE for each parameter is as follows:

$$\hat{\vec{\mu}}_c = \frac{N_c}{N} \tag{4.31}$$

$$\hat{\vec{\mu}}_c = \frac{1}{N_c} \sum_{i:y_i=c} \vec{x}_i \tag{4.32}$$

$$\hat{\vec{\Sigma}}_c = \frac{1}{N_c} \sum_{i:y_i=c} (\vec{x}_i - \hat{\vec{\mu}}_c)(\vec{x}_i - \hat{\vec{\mu}}_c)^T \tag{4.33}$$

4.2.5 Strategies for preventing overfitting

The speed and simplicity of the MLE method is one of its greatest appeals. However, the MLE can badly overfit in high dimensions. In particular, the MLE for a full covariance matrix is singular if $N_c < D$. And even when $N_c > D$, the MLE can be ill-conditioned, meaning it is close to singular. There are several possible solutions to this problem:

- Use a diagonal covariance matrix for each class, which assumes the features are conditionally independent; this is equivalent to using a naive Bayes classifier (Section 3.5).
- Use a full covariance matrix, but force it to be the same for all classes,$\vec{\Sigma}_c = \vec{\Sigma}$. This is an example of **parameter tying** or **parameter sharing**, and is equivalent to LDA (Section 4.2.2).
- Use a diagonal covariance matrix and forced it to be shared. This is called diagonal covariance LDA, and is discussed in Section TODO.
- Use a full covariance matrix, but impose a prior and then integrate it out. If we use a conjugate prior, this can be done in closed form, using the results from Section TODO; this is analogous to the "Bayesian naive Bayes" method in Section 3.5.1.2. See (Minka 2000f) for details.
- Fit a full or diagonal covariance matrix by MAP estimation. We discuss two different kindsof prior below.
- Project the data into a low dimensional subspace and fit the Gaussians there. See Section TODO for a way to find the best (most discriminative) linear projection.

We discuss some of these options below.

4.2.6 Regularized LDA *

4.2.7 Diagonal LDA

4.2.8 Nearest shrunken centroids classifier *

One drawback of diagonal LDA is that it depends on all of the features. In high dimensional problems, we might prefer a method that only depends on a subset of the features, for reasons of accuracy and interpretability. One approach is to use a screening method, perhaps based on mutual information, as in Section 3.5.4. We now discuss another approach to this problem known as the **nearest shrunken centroids** classifier (Hastie et al. 2009, p652).

4.3 Inference in jointly Gaussian distributions

Given a joint distribution, $p(\vec{x}_1, \vec{x}_2)$, it is useful to be able to compute marginals $p(\vec{x}_1)$ and conditionals $p(\vec{x}_1|\vec{x}_2)$. We discuss how to do this below, and then give some applica-

tions. These operations take $O(D^3)$ time in the worst case. See Section TODO for faster methods.

4.3.1 Statement of the result

Theorem 4.2. ***(Marginals and conditionals of an MVN).*** *Suppose $X = (\vec{x}_1, \vec{x}_2)$ is jointly Gaussian with parameters*

$$\vec{\mu} = \begin{pmatrix} \vec{\mu}_1 \\ \vec{\mu}_2 \end{pmatrix}, \vec{\Sigma} = \begin{pmatrix} \vec{\Sigma}_{11} & \vec{\Sigma}_{12} \\ \vec{\Sigma}_{21} & \vec{\Sigma}_{22} \end{pmatrix}, \vec{\Lambda} = \vec{\Sigma}^{-1} = \begin{pmatrix} \vec{\Lambda}_{11} & \vec{\Lambda}_{12} \\ \vec{\Lambda}_{21} & \vec{\Lambda}_{22} \end{pmatrix}, \tag{4.34}$$

Then the marginals are given by

$$\begin{aligned} p(\vec{x}_1) &= \mathcal{N}(\vec{x}_1 | \vec{\mu}_1, \vec{\Sigma}_{11}) \\ p(\vec{x}_2) &= \mathcal{N}(\vec{x}_2 | \vec{\mu}_2, \vec{\Sigma}_{22}) \end{aligned} \tag{4.35}$$

and the posterior conditional is given by

$$\boxed{\begin{aligned} p(\vec{x}_1 | \vec{x}_2) &= \mathcal{N}(\vec{x}_1 | \vec{\mu}_{1|2}, \vec{\Sigma}_{1|2}) \\ \vec{\mu}_{1|2} &= \vec{\mu}_1 + \vec{\Sigma}_{12} \vec{\Sigma}_{22}^{-1} (\vec{x}_2 - \vec{\mu}_2) \\ &= \vec{\mu}_1 - \vec{\Lambda}_{11}^{-1} \vec{\Lambda}_{12} (\vec{x}_2 - \vec{\mu}_2) \\ &= \vec{\Sigma}_{1|2} (\vec{\Lambda}_{11} \vec{\mu}_1 - \vec{\Lambda}_{12} (\vec{x}_2 - \vec{\mu}_2)) \\ \vec{\Sigma}_{1|2} &= \vec{\Sigma}_{11} - \vec{\Sigma}_{12} \vec{\Sigma}_{22}^{-1} \vec{\Sigma}_{21} = \vec{\Lambda}_{11}^{-1} \end{aligned}} \tag{4.36}$$

Equation 4.36 is of such crucial importance in this book that we have put a box around it, so you can easily find it. For the proof, see Section TODO.

We see that both the marginal and conditional distributions are themselves Gaussian. For the marginals, we just extract the rows and columns corresponding to $\vec{x}_1$ or $\vec{x}_2$. For the conditional, we have to do a bit more work. However, it is not that complicated: the conditional mean is just a linear function of $\vec{x}_2$, and the conditional covariance is just a constant matrix that is independent of $\vec{x}_2$. We give three different (but equivalent) expressions for the posterior mean, and two different (but equivalent) expressions for the posterior covariance; each one is useful in different circumstances.

4.3.2 Examples

Below we give some examples of these equations in action, which will make them seem more intuitive.

4.3.2.1 Marginals and conditionals of a 2d Gaussian

4.4 Linear Gaussian systems

Suppose we have two variables, $\vec{x}$ and $\vec{y}$.Let $\vec{x} \in \mathbb{R}^{D_x}$ be a hidden variable, and $\vec{y} \in \mathbb{R}^{D_y}$ be a noisy observation of $\vec{x}$. Let us assume we have the following prior and likelihood:

$$\boxed{\begin{aligned} p(\vec{x}) &= \mathcal{N}(\vec{x} | \vec{\mu}_x, \vec{\Sigma}_x) \\ p(\vec{y} | \vec{x}) &= \mathcal{N}(\vec{y} | \vec{W}\vec{x} + \vec{\mu}_y, \vec{\Sigma}_y) \end{aligned}} \tag{4.37}$$

where $\vec{W}$ is a matrix of size $D_y \times D_x$. This is an example of a **linear Gaussian system**. We can represent this schematically as $\vec{x} \to \vec{y}$, meaning $\vec{x}$ generates $\vec{y}$. In this section, we show how to "invert the arrow", that is, how to infer $\vec{x}$ from $\vec{y}$. We state the result below, then give several examples, and finally we derive the result. We will see many more applications of these results in later chapters.

4.4.1 Statement of the result

Theorem 4.3. ***(Bayes rule for linear Gaussian systems).*** *Given a linear Gaussian system, as in Equation 4.37, the posterior $p(\vec{x}|\vec{y})$ is given by the following:*

$$\boxed{\begin{aligned} p(\vec{x}|\vec{y}) &= \mathcal{N}(\vec{x} | \vec{\mu}_{x|y}, \vec{\Sigma}_{x|y}) \\ \vec{\Sigma}_{x|y} &= \vec{\Sigma}_x^{-1} + \vec{W}^T \vec{\Sigma}_y^{-1} \vec{W} \\ \vec{\mu}_{x|y} &= \vec{\Sigma}_{x|y} \left[\vec{W}^T \vec{\Sigma}_y^{-1} (\vec{y} - \vec{\mu}_y) + \vec{\Sigma}_x^{-1} \vec{\mu}_x \right] \end{aligned}} \tag{4.38}$$

In addition, the normalization constant $p(\vec{y})$ is given by

$$\boxed{p(\vec{y}) = \mathcal{N}(\vec{y} | \vec{W}\vec{\mu}_x + \vec{\mu}_y, \vec{\Sigma}_y + \vec{W}\vec{\Sigma}_x\vec{W}^T)} \tag{4.39}$$

For the proof, see Section 4.4.3 TODO.

4.5 Digression: The Wishart distribution *

4.6 Inferring the parameters of an MVN

4.6.1 Posterior distribution of μ

4.6.2 Posterior distribution of Σ *

4.6.3 Posterior distribution of μ and Σ *

4.6.4 Sensor fusion with unknown precisions *

Chapter 5
Bayesian statistics

5.1 Introduction

Using the posterior distribution to summarize everything we know about a set of unknown variables is at the core of Bayesian statistics. In this chapter, we discuss this approach to statistics in more detail.

5.2 Summarizing posterior distributions

The posterior $p(\vec{\theta}|\mathcal{D})$ summarizes everything we know about the unknown quantities $\vec{\theta}$. In this section, we discuss some simple quantities that can be derived from a probability distribution, such as a posterior. These summary statistics are often easier to understand and visualize than the full joint.

5.2.1 MAP estimation

We can easily compute a **point estimate** of an unknown quantity by computing the posterior mean, median or mode. In Section 5.7, we discuss how to use decision theory to choose between these methods. Typically the posterior mean or median is the most appropriate choice for a realvalued quantity, and the vector of posterior marginals is the best choice for a discrete quantity. However, the posterior mode, aka the MAP estimate, is the most popular choice because it reduces to an optimization problem, for which efficient algorithms often exist. Futhermore, MAP estimation can be interpreted in non-Bayesian terms, by thinking of the log prior as a regularizer (see Section TODO for more details).

Although this approach is computationally appealing, it is important to point out that there are various drawbacks to MAP estimation, which we briefly discuss below. This will provide motivation for the more thoroughly Bayesian approach which we will study later in this chapter(and elsewhere in this book).

5.2.1.1 No measure of uncertainty

The most obvious drawback of MAP estimation, and indeed of any other *point estimate* such as the posterior mean or median, is that it does not provide any measure of uncertainty. In many applications, it is important to know how much one can trust a given estimate. We can derive such confidence measures from the posterior, as we discuss in Section 5.2.2.

5.2.1.2 Plugging in the MAP estimate can result in overfitting

If we don't model the uncertainty in our parameters, then our predictive distribution will be overconfident. Overconfidence in predictions is particularly problematic in situations where we may be risk averse; see Section 5.7 for details.

5.2.1.3 The mode is an untypical point

Choosing the mode as a summary of a posterior distribution is often a very poor choice, since the mode is usually quite untypical of the distribution, unlike the mean or median. The basic problem is that the mode is a point of measure zero, whereas the mean and median take the volume of the space into account. See Figure 5.1.

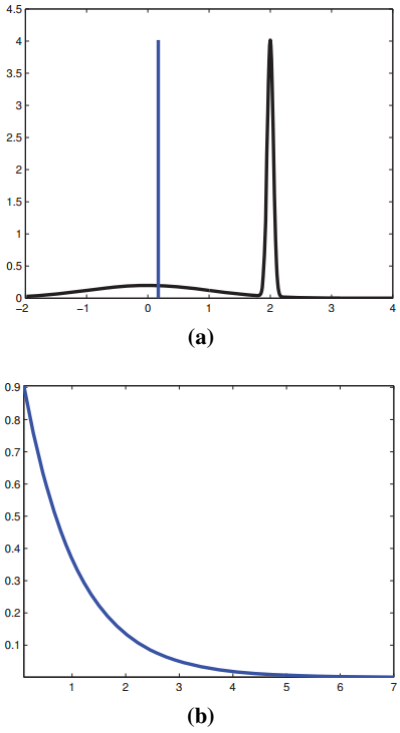


Fig. 5.1: (a) A bimodal distribution in which the mode is very untypical of the distribution. The thin blue vertical line is the mean, which is arguably a better summary of the distribution, since it is near the majority of the probability mass. (b) A skewed distribution in which the mode is quite different from the mean.

How should we summarize a posterior if the mode is not a good choice? The answer is to use decision theory, which we discuss in Section 5.7. The basic idea is to specify a loss function, where $L(\theta, \hat{\theta})$ is the loss you incur if the truth is θ and your estimate is $\hat{\theta}$. If we use 0-1 loss $L(\theta, \hat{\theta}) = \mathbb{I}(\theta \neq \hat{\theta})$(see section 1.2.2.1), then the optimal estimate is the posterior mode. 0-1 loss means you only get "points" if you make no errors, otherwise you get nothing: there is no "partial credit" under this loss function! For continuous-valued quantities, we often prefer to use squared error loss, $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$; the corresponding optimal estimator is then the posterior mean, as we show in Section 5.7. Or we can use a more robust loss function, $L(\theta, \hat{\theta}) = |\theta - \hat{\theta}|$, which gives rise to the posterior median.

5.2.1.4 MAP estimation is not invariant to reparameterization *

A more subtle problem with MAP estimation is that the result we get depends on how we parameterize the probability distribution. Changing from one representation to another equivalent representation changes the result, which is not very desirable, since the units of measurement are arbitrary (e.g., when measuring distance, we can use centimetres or inches).

To understand the problem, suppose we compute the posterior forx. If we define y=f(x), the distribution for yis given by Equation 2.44. The $\frac{dx}{dy}$ term is called the Jacobian, and it measures the change in size of a unit volume passed through f. Let $\hat{x} = \arg\max_x p_x(x)$ be the MAP estimate for x. In general it is not the case that $\hat{x} = \arg\max_x p_x(x)$ is given by $f(\hat{x})$. For example, let $X \sim \mathcal{N}(6, 1)$ and $y = f(x)$, where $f(x) = 1/(1 + \exp(-x + 5))$.

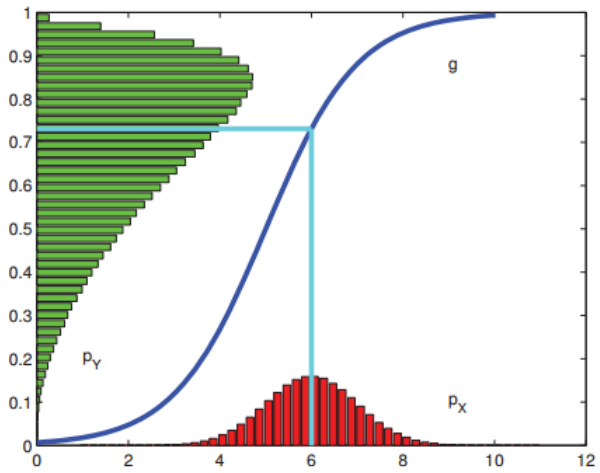


Fig. 5.2: Example of the transformation of a density under a nonlinear transform. Note how the mode of the transformed distribution is not the transform of the original mode. Based on Exercise 1.4 of (Bishop 2006b).

We can derive the distribution of y using Monte Carlo simulation (see Section 2.7). The result is shown in Figure **??**. We see that the original Gaussian has become "squashed" by the sigmoid nonlinearity. In particular, we see that the mode of the transformed distribution is not equal to the transform of the original mode.

The MLE does not suffer from this since the likelihood is a function, not a probability density. Bayesian inference does not suffer from this problem either, since the change of measure is taken into account when integrating over the parameter space.

5.2.2 Credible intervals

In addition to point estimates, we often want a measure of confidence. A standard measure of confidence in some (scalar) quantity θ is the "width" of its posterior distribution. This can be measured using a $100(1-\alpha)\%$ credible interval, which is a (contiguous) region $C = (\ell, u)$(standing for lower and upper) which contains $1 - \alpha$ of the posterior probability mass, i.e.,

$$C_\alpha(\mathcal{D}) \quad \text{where } P(\ell \leq \theta \leq u) = 1 - \alpha \tag{5.1}$$

There may be many such intervals, so we choose one such that there is $(1-\alpha)/2$ mass in each tail; this is called a **central interval**.

If the posterior has a known functional form, we can compute the posterior central interval using $\ell = F^{-1}(\alpha/2)$ and $u = F^{-1}(1-\alpha/2)$, where F is the cdf of the posterior.

If we don't know the functional form, but we can draw samples from the posterior, then we can use a Monte Carlo approximation to the posterior quantiles: we simply sort the $\mathcal{S}$ samples, and find the one that occurs at location $\alpha/\mathcal{S}$ along the sorted list. As $\mathcal{S} \to \infty$, this converges to the true quantile.

People often confuse Bayesian credible intervals with frequentist confidence intervals. However, they are not the same thing, as we discuss in Section TODO. In general, credible intervals are usually what people want to compute, but confidence intervals are usually what they actually compute, because most people are taught frequentist statistics but not Bayesian statistics. Fortunately, the mechanics of computing a credible interval is just as easy as computing a confidence interval.

5.2.3 Inference for a difference in proportions

Sometimes we have multiple parameters, and we are interested in computing the posterior distribution of some function of these parameters. For example, suppose you are about to buy something from Amazon.com, and there are two sellers offering it for the same price. Seller 1 has 90 positive reviews and 10 negative reviews. Seller 2 has 2 positive reviews and 0 negative reviews. Who should you buy from?[18].

[18] This example is from http://www.johndcook.com/blog/2011/09/27/bayesian-amazon/

On the face of it, you should pick seller 2, but we cannot be very confident that seller 2 is better since it has had so few reviews. In this section, we sketch a Bayesian analysis of this problem. Similar methodology can be used to compare rates or proportions across groups for a variety of other settings.

Let θ_1 and θ_2 be the unknown reliabilities of the two sellers. Since we don't know much about them, we'll endow them both with uniform priors, $\theta_i \sim \text{Beta}(1,1)$. The posteriors are $p(\theta_1|\mathcal{D}_1) = \text{Beta}(91,11)$ and $p(\theta_2|\mathcal{D}_2) = \text{Beta}(3,1)$.

We want to compute $p(\theta_1 > \theta_2|\mathcal{D})$. For convenience, let us define $\delta = \theta_1 - \theta_2$ as the difference in the rates. (Alternatively we might want to work in terms of the log-odds ratio.) We can compute the desired quantity using numerical integration

$$p(\delta > 0|\mathcal{D}) = \int_0^1 \int_0^1 \mathbb{I}(\theta_1 > \theta_2)\text{Beta}(\theta_1|91,11) \text{Beta}(\theta_2|3,1)\mathrm{d}\theta_1\mathrm{d}\theta_2 \tag{5.2}$$

We find $p(\delta > 0|\mathcal{D}) = 0.710$, which means you are better off buying from seller 1!

5.3 Bayesian model selection

In general, when faced with a set of models (i.e., families of parametric distributions) of different complexity, how should we choose the best one? This is called the **model selection** problem.

One approach is to use cross-validation to estimate the generalization error of all the candidate models, and then to pick the model that seems the best. However, this requires fitting each model K times, where K is the number of CV folds. A more efficient approach is to compute the posterior over models,

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)p(m)}{\sum_{m'} p(\mathcal{D}|m')p(m')} \tag{5.3}$$

From this, we can easily compute the MAP model, $\hat{m} = \arg\max_m p(m|\mathcal{D})$. This is called **Bayesian model selection**.

If we use a uniform prior over models, this amounts to picking the model which maximizes

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\vec{\theta})p(\vec{\theta}|m)\mathrm{d}\vec{\theta} \tag{5.4}$$

This quantity is called the **marginal likelihood**, the **integrated likelihood**, or the **evidence** for model m. The details on how to perform this integral will be discussed in Section 5.3.2. But first we give an intuitive interpretation of what this quantity means.

5.3.1 Bayesian Occam's razor

One might think that using $p(\mathcal{D}|m)$ to select models would always favour the model with the most parameters. This is true if we use $p(\mathcal{D}|\hat{\vec{\theta}}_m)$ to select models, where $\hat{\vec{\theta}}_m)$ is the MLE or MAP estimate of the parameters for model m, because models with more parameters will fit the data better, and hence achieve higher likelihood. However, if we integrate out the parameters, rather than maximizing them, we are automatically protected from overfitting: models with more parameters do not necessarily have higher *marginal likelihood*. This is called the **Bayesian Occam's razor** effect (MacKay 1995b; Murray and Ghahramani 2005), named after the principle known as **Occam's razor**, which says one should pick the simplest model that adequately explains the data.

One way to understand the Bayesian Occam's razor is to notice that the marginal likelihood can be rewritten as follows, based on the chain rule of probability (Equation 2.3):

$$\begin{aligned} p(D) = p((\vec{x}_1,y_1))&p((\vec{x}_2,y_2)|(\vec{x}_1,y_1)) \\ &p((\vec{x}_3,y_3)|(\vec{x}_1,y_1):(\vec{x}_2,y_2))\cdots \\ &p((\vec{x}_N,y_N)|(\vec{x}_1,y_1):(\vec{x}_{N-1},y_{N-1})) \end{aligned} \tag{5.5}$$

This is similar to a leave-one-out cross-validation estimate (Section 1.3.4) of the likelihood, since we predict each future point given all the previous ones. (Of course, the order of the data does not matter in the above expression.) If a model is too complex, it will overfit the "early" examples and will then predict the remaining ones poorly.

Another way to understand the Bayesian Occam's razor effect is to note that probabilities must sum to one. Hence $\sum_{p(\mathcal{D}')} p(m|\mathcal{D}') = 1$, where the sum is over all possible data sets. Complex models, which can predict many things, must spread their probability mass thinly, and hence will not obtain as large a probability for any given data set as simpler models. This is sometimes called the **conservation of probability mass** principle, and is illustrated in Figure 5.3.

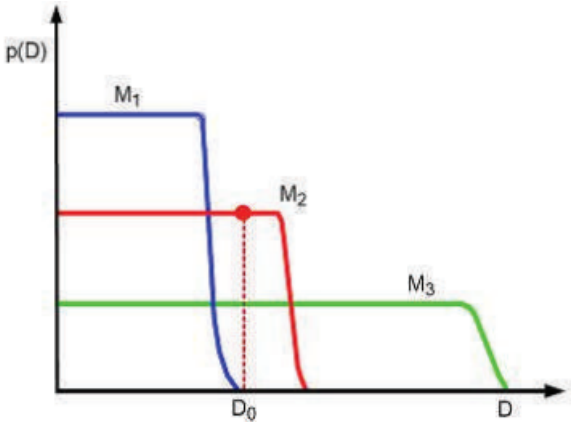


Fig. 5.3: A schematic illustration of the Bayesian Occam's razor. The broad (green) curve corresponds to a complex model, the narrow (blue) curve to a simple model, and the middle (red) curve is just right. Based on Figure 3.13 of (Bishop 2006a).

When using the Bayesian approach, we are not restricted to evaluating the evidence at a finite grid of values. Instead, we can use numerical optimization to find $\lambda^* = \arg\max_\lambda p(\mathcal{D}|\lambda)$. This technique is called **empirical Bayes** or **type II maximum likelihood** (see Section 5.6 for details). An example is shown in Figure TODO(b): we see that the curve has a similar shape to the CV estimate, but it can be computed more efficiently.

5.3.2 Computing the marginal likelihood (evidence)

When discussing parameter inference for a fixed model, we often wrote

$$p(\vec{\theta}|\mathcal{D},m) \propto p(\vec{\theta}|m)p(\mathcal{D}|\vec{\theta},m) \tag{5.6}$$

thus ignoring the normalization constant $p(\mathcal{D}|m)$. This is valid since $p(\mathcal{D}|m)$is constant wrt $\vec{\theta}$. However, when comparing models, we need to know how to compute the marginal likelihood, $p(\mathcal{D}|m)$. In general, this can be quite hard, since we have to integrate over all possible parameter values, but when we have a conjugate prior, it is easy to compute, as we now show.

Let $p(\vec{\theta}) = q(\vec{\theta})/Z_0$ be our prior, where $q(\vec{\theta})$ is an unnormalized distribution, and Z_0 is the normalization constant of the prior. Let $p(\mathcal{D}|\vec{\theta}) = q(\mathcal{D}|\vec{\theta})/Z_\ell$ be the likelihood, where Z_ℓ contains any constant factors in the likelihood. Finally let $p(\vec{\theta}|\mathcal{D}) = q(\vec{\theta}|\mathcal{D})/Z_N$ be our posterior , where $q(\vec{\theta}|\mathcal{D}) = q(\mathcal{D}|\vec{\theta})q(\vec{\theta})$ is the unnormalized posterior, and Z_N is the normalization constant of the posterior. We have

$$p(\vec{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\vec{\theta})p(\vec{\theta})}{p(\mathcal{D})} \tag{5.7}$$

$$\frac{q(\vec{\theta}|\mathcal{D})}{Z_N} = \frac{q(\mathcal{D}|\vec{\theta})q(\vec{\theta})}{Z_\ell Z_0 p(\mathcal{D})} \tag{5.8}$$

$$p(\mathcal{D}) = \frac{Z_N}{Z_0 Z_\ell} \tag{5.9}$$

So assuming the relevant normalization constants are tractable, we have an easy way to compute the marginal likelihood. We give some examples below.

5.3.2.1 Beta-binomial model

Let us apply the above result to the Beta-binomial model. Since we know $p(\vec{\theta}|\mathcal{D}) = \text{Beta}(\vec{\theta}|a',b')$, where $a' = a + N_1$, $b' = b + N_0$, we know the normalization constant of the posterior is $B(a',b')$. Hence

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{5.10}$$

$$= \frac{1}{p(\mathcal{D})}\left[\frac{1}{B(a,b)}\theta^{a-1}(1-\theta)^{b-1}\right] \left[\binom{N}{N_1}\theta^{N_1}(1-\theta)^{N_0}\right] \tag{5.11}$$

$$= \binom{N}{N_1}\frac{1}{p(\mathcal{D})}\frac{1}{B(a,b)}\left[\theta^{a+N_1-1}(1-\theta)^{b+N_0-1}\right] \tag{5.12}$$

So

$$\frac{1}{B(a+N_1,b+N_0)} = \binom{N}{N_1}\frac{1}{p(\mathcal{D})}\frac{1}{B(a,b)} \tag{5.13}$$

$$p(\mathcal{D}) = \binom{N}{N_1}\frac{B(a+N_1,b+N_0)}{B(a,b)} \tag{5.14}$$

The marginal likelihood for the Beta-Bernoulli model is the same as above, except it is missingthe $\binom{N}{N_1}$ term.

5.3.2.2 Dirichlet-multinoulli model

By the same reasoning as the Beta-Bernoulli case, one can show that the marginal likelihood for the Dirichlet-multinoulli model is given by

$$p(\mathcal{D}) = \frac{B(\vec{N}+\vec{\alpha})}{B(\vec{\alpha})} \tag{5.15}$$

$$= \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(N+\sum_k \alpha_k)}\prod_k \frac{\Gamma(N_k+\alpha_k)}{\Gamma(\alpha_k)} \tag{5.16}$$

5.3.2.3 Gaussian-Gaussian-Wishart model

Consider the case of an MVN with a conjugate NIW prior. Let Z_0 be the normalizer for the prior, Z_N be normalizer for the posterior, and let $Z_\ell(2\pi)^{ND/2} =$ be the normalizer for the likelihood. Then it is easy to see that

$$p(\mathcal{D}) = \frac{Z_N}{Z_0 Z_\ell} \tag{5.17}$$

$$= \frac{1}{(2\pi)^{ND/2}}\frac{\left(\frac{2\pi}{\kappa_N}\right)^{D/2}|\vec{S}_N|^{-\nu_N/2}2^{(\nu_0+N)D/2}\Gamma_D(\nu_N/2)}{\left(\frac{2\pi}{\kappa_0}\right)^{D/2}|\vec{S}_0|^{-\nu_0/2}2^{\nu_0 D/2}\Gamma_D(\nu_0/2)} \tag{5.18}$$

$$= \frac{1}{\pi^{ND/2}}\left(\frac{\kappa_0}{\kappa_N}\right)^{D/2}\frac{|\vec{S}_0|^{\nu_0/2}}{|\vec{S}_N|^{\nu_N/2}}\frac{\Gamma_D(\nu_N/2)}{\Gamma_D(\nu_0/2)} \tag{5.19}$$

5.3.2.4 BIC approximation to log marginal likelihood

In general, computing the integral in Equation 5.4 can be quite difficult. One simple but popular approximation is known as the **Bayesian information criterion** or **BIC**, which has the following form (Schwarz 1978):

$$\text{BIC} \triangleq \log p(\mathcal{D}|\hat{\vec{\theta}}) - \frac{\text{dof}(\hat{\vec{\theta}})}{2}\log N \tag{5.20}$$

where $\text{dof}(\hat{\vec{\theta}})$ is the number of **degrees of freedom** in the model, and $\hat{\vec{\theta}}$ is the MLE for the model. We see that this has the form of a **penalized log likelihood**, where the penalty term depends on the model's complexity. See Section TODO for the derivation of the BIC score.

As an example, consider linear regression. As we show in Section TODO, the MLE is given by $\hat{\vec{w}} = (\vec{X}^T\vec{X})^{-1}\vec{X}^T\vec{y}$ and $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{\vec{w}}^T\vec{x}_i)$. The corresponding log likelihood is given by

$$\log p(\mathcal{D}|\hat{\vec{\theta}}) = -\frac{N}{2}\log(2\pi\hat{\sigma}^2) - \frac{N}{2} \tag{5.21}$$

Hence the BIC score is as follows (dropping constant terms)

$$\text{BIC} = -\frac{N}{2}\log(\hat{\sigma}^2) - \frac{D}{2}\log N \tag{5.22}$$

where D is the number of variables in the model. In the statistics literature, it is common to use an alternative definition of BIC, which we call the BIC *cost*(since we want to minimize it):

$$\text{BIC-cost} \triangleq -2\log p(\mathcal{D}|\hat{\vec{\theta}}) - \text{dof}(\hat{\vec{\theta}})\log N \approx -2\log p(\mathcal{D}) \tag{5.23}$$

In the context of linear regression, this becomes

$$\text{BIC-cost} = N\log(\hat{\sigma}^2) + D\log N \tag{5.24}$$

The BIC method is very closely related to the **minimum description length** or **MDL** principle, which characterizes the score for a model in terms of how well it fits the data, minus how complex the model is to define. See (Hansen and Yu 2001) for details.

There is a very similar expression to BIC/ MDL called the **Akaike information criterion** or **AIC**, defined as

$$\text{AIC}(m, \mathcal{D}) = \log p(\mathcal{D}|\hat{\vec{\theta}}_{MLE}) - \text{dof}(m) \tag{5.25}$$

This is derived from a frequentist framework, and cannot be interpreted as an approximation to the marginal likelihood. Nevertheless, the form of this expression is very similar to BIC. We see that the penalty for AIC is less than for BIC. This causes AIC to pick more complex models. However, this can result in better predictive accuracy. See e.g., (Clarke et al. 2009, sec 10.2) for further discussion on such information criteria.

5.3.2.5 Effect of the prior

Sometimes it is not clear how to set the prior. When we are performing posterior inference, the details of the prior may not matter too much, since the likelihood often overwhelms the prior anyway. But when computing the marginal likelihood, the prior plays a much more important role, since we are averaging the likelihood over all possible parameter settings, as weighted by the prior.

If the prior is unknown, the correct Bayesian procedure is to put a prior on the prior. If the prior is unknown, the correct Bayesian procedure is to put a prior on the prior.

5.3.3 Bayes factors

Suppose our prior on models is uniform, $p(m) \propto 1$. Then model selection is equivalent to picking the model with the highest marginal likelihood. Now suppose we just have two models we are considering, call them the **null hypothesis**, M_0, and the **alternative hypothesis**, M_1. Define the **Bayes factor** as the ratio of marginal likelihoods:

$$\text{BF}_{1,0} \triangleq \frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_0)} = \frac{p(M_1|\mathcal{D})}{p(M_2|\mathcal{D})} \Big/ \frac{p(M_1)}{p(M_0)} \tag{5.26}$$

5.4 Priors

The most controversial aspect of Bayesian statistics is its reliance on priors. Bayesians argue this is unavoidable, since nobody is a **tabula rasa** or **blank slate**: all inference must be done conditional on certain assumptions about the world. Nevertheless, one might be interested in minimizing the impact of one's prior assumptions. We briefly discuss some ways to do this below.

5.4.1 Uninformative priors

If we don't have strong beliefs about what θ should be, it is common to use an **uninformative** or **non-informative** prior, and to "let the data speak for itself".

5.4.2 Robust priors

In many cases, we are not very confident in our prior, so we want to make sure it does not have an undue influence on the result. This can be done by using **robust priors**(Insua and Ruggeri 2000), which typically have heavy tails, which avoids forcing things to be too close to the prior mean.

5.4.3 Mixtures of conjugate priors

Robust priors are useful, but can be computationally expensive to use. Conjugate priors simplify the computation, but are often not robust, and not flexible enough to encode our prior knowledge. However, it turns out that a **mixture of conjugate priors** is also conjugate, and can approximate any kind of prior (Dallal and Hall 1983; Diaconis and

Ylvisaker 1985). Thus such priors provide a good compromise between computational convenience and flexibility.

5.5 Hierarchical Bayes

A key requirement for computing the posterior $p(\vec{\theta}|\mathcal{D})$ is the specification of a prior $p(\vec{\theta}|\vec{\eta})$, where $\vec{\eta}$ are the hyperparameters. What if we don't know how to set $\vec{\eta}$? In some cases, we can use uninformative priors, we we discussed above. A more Bayesian approach is to put a prior on our priors! In terms of graphical models (Chapter TODO), we can represent the situation as follows:

$$\vec{\eta} \rightarrow \vec{\theta} \rightarrow \mathcal{D} \tag{5.27}$$

This is an example of a **hierarchical Bayesian model**, also called a **multi-level** model, since there are multiple levels of unknown quantities.

5.6 Empirical Bayes

Method	Definition
Maximum likelihood	$\hat{\vec{\theta}} = \arg\max_{\vec{\theta}} p(\mathcal{D}\|\vec{\theta})$
MAP estimation	$\hat{\vec{\theta}} = \arg\max_{\vec{\theta}} p(\mathcal{D}\|\vec{\theta}) p(\vec{\theta}\|\vec{\eta})$
ML-II (**Empirical Bayes**)	$\hat{\vec{\eta}} = \arg\max_{\vec{\eta}} \int p(\mathcal{D}\|\vec{\theta}) p(\vec{\theta}\|\vec{\eta}) \mathrm{d}\vec{\theta}$ $= \arg\max_{\vec{\eta}} p(\mathcal{D}\|\vec{\eta})$
MAP-II	$\hat{\vec{\eta}} = \arg\max_{\vec{\eta}} \int p(\mathcal{D}\|\vec{\theta}) p(\vec{\theta}\|\vec{\eta}) p(\vec{\eta}) \mathrm{d}\vec{\theta}$ $= \arg\max_{\vec{\eta}} p(\mathcal{D}\|\vec{\eta}) p(\vec{\eta})$
Full Bayes	$p(\vec{\theta}, \vec{\eta}\|\mathcal{D}) \propto p(\mathcal{D}\|\vec{\theta}) p(\vec{\theta}\|\vec{\eta})$

5.7 Bayesian decision theory

We have seen how probability theory can be used to represent and updates our beliefs about the state of the world. However, ultimately our goal is to convert our beliefs into actions. In this section, we discuss the optimal way to do this.

Our goal is to devise a **decision procedure** or **policy**, $f(\vec{x}) : \mathcal{X} \rightarrow \mathcal{Y}$, which minimizes the **expected loss** $R_{\exp}(f)$(see Equation 1.1).

In the Bayesian approach to decision theory, the optimal output, having observed $\vec{x}$, is defined as the output a that minimizes the **posterior expected loss**:

$$\rho(f) = \mathbb{E}_{p(y|\vec{x})}[L(y, f(\vec{x}))] = \begin{cases} \sum_y L[y, f(\vec{x})] p(y|\vec{x}) \\ \int_y L[y, f(\vec{x})] p(y|\vec{x}) \mathrm{d}y \end{cases} \tag{5.28}$$

Hence the **Bayes estimator**, also called the **Bayes decision rule**, is given by

$$\delta(\vec{x}) = \arg\min_{f \in \mathcal{H}} \rho(f) \tag{5.29}$$

5.7.1 Bayes estimators for common loss functions

5.7.1.1 MAP estimate minimizes 0-1 loss

When $L(y, f(x))$ is **0-1 loss**(Section 1.2.2.1), we can proof that MAP estimate minimizes 0-1 loss,

$$\begin{aligned} \arg\min_{f \in \mathcal{H}} \rho(f) &= \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{K} L[C_k, f(\vec{x})] p(C_k|\vec{x}) \\ &= \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{K} \mathbb{I}(f(\vec{x}) \neq C_k) p(C_k|\vec{x}) \\ &= \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{K} p(f(\vec{x}) \neq C_k|\vec{x}) \\ &= \arg\min_{f \in \mathcal{H}} [1 - p(f(\vec{x}) = C_k|\vec{x})] \\ &= \arg\max_{f \in \mathcal{H}} p(f(\vec{x}) = C_k|\vec{x}) \end{aligned}$$

5.7.1.2 Posterior mean minimizes ℓ_2(quadratic) loss

For continuous parameters, a more appropriate loss function is **squared error**, ℓ_2 **loss**, or **quadratic loss**, defined as $L(y, f(\vec{x})) = [y - f(\vec{x})]^2$.

The posterior expected loss is given by

$$\begin{aligned} \rho(f) &= \int_y L[y, f(\vec{x})] p(y|\vec{x}) \mathrm{d}y \\ &= \int_y [y - f(\vec{x})]^2 p(y|\vec{x}) \mathrm{d}y \\ &= \int_y \left[y^2 - 2y f(\vec{x}) + f(\vec{x})^2 \right] p(y|\vec{x}) \mathrm{d}y \end{aligned} \tag{5.30}$$

Hence the optimal estimate is the posterior mean:

$$\begin{aligned} \frac{\partial \rho}{\partial f} &= \int_y [-2y + 2f(\vec{x})] p(y|\vec{x}) \mathrm{d}y = 0 \Rightarrow \\ \int_y f(\vec{x}) p(y|\vec{x}) \mathrm{d}y &= \int_y y p(y|\vec{x}) \mathrm{d}y \\ f(\vec{x}) \int_y p(y|\vec{x}) \mathrm{d}y &= \mathbb{E}_{p(y|\vec{x})}[y] \\ f(\vec{x}) &= \mathbb{E}_{p(y|\vec{x})}[y] \end{aligned} \tag{5.31}$$

This is often called the **minimum mean squared error** estimate or **MMSE** estimate.

5.7.1.3 Posterior median minimizes ℓ_1(absolute) loss

The ℓ_2 loss penalizes deviations from the truth quadratically, and thus is sensitive to outliers. A more robust alternative is the absolute or ℓ_1 loss. The optimal estimate is the posterior median, i.e., a value a such that $P(y < a|\vec{x}) = P(y \geq a|\vec{x}) = 0.5$.

Proof.

$$\begin{aligned}
\rho(f) &= \int_y L[y, f(\vec{x})] p(y|\vec{x}) \mathrm{d}y = \int_y |y - f(\vec{x})| p(y|\vec{x}) \mathrm{d}y \\
&= \int_y [f(\vec{x}) - y] p(y < f(\vec{x})|\vec{x}) + \\
&\quad [y - f(\vec{x})] p(y \geq f(\vec{x})|\vec{x}) \mathrm{d}y \\
&\frac{\partial \rho}{\partial f} = \int_y \left[p(y < f(\vec{x})|\vec{x}) - p(y \geq f(\vec{x})|\vec{x}) \right] \mathrm{d}y = 0 \Rightarrow \\
&p(y < f(\vec{x})|\vec{x}) = p(y \geq f(\vec{x})|\vec{x}) = 0.5 \\
&\therefore f(\vec{x}) = \text{median}
\end{aligned}$$

5.7.1.4 Reject option

In classification problems where $p(y|\vec{x})$ is very uncertain, we may prefer to choose a reject action, in which we refuse to classify the example as any of the specified classes, and instead say "don't know". Such ambiguous cases can be handled by e.g., a human expert. This is useful in **risk averse** domains such as medicine and finance.

We can formalize the reject option as follows. Let choosing $f(\vec{x}) = c_{K+1}$ correspond to picking the reject action, and choosing $f(\vec{x}) \in \{C_1, ..., C_k\}$ correspond to picking one of the classes. Suppose we define the loss function as

$$L(f(\vec{x}), y) = \begin{cases} 0 & \text{if } f(\vec{x}) = y \text{ and } f(\vec{x}), y \in \{C_1, ..., C_k\} \\ \lambda_s & \text{if } f(\vec{x}) \neq y \text{ and } f(\vec{x}), y \in \{C_1, ..., C_k\} \\ \lambda_r & \text{if } f(\vec{x}) = C_{K+1} \end{cases} \tag{5.32}$$

where λ_s is the cost of a substitution error, and λ_r is the cost of the reject action.

5.7.1.5 Supervised learning

We can define the loss incurred by $f(\vec{x})$ (i.e., using this predictor) when the unknown state of nature is $\vec{\theta}$(the parameters of the data generating mechanism) as follows:

$$L(\vec{\theta}, f) \triangleq \mathbb{E}_{p(\vec{x}, y|\vec{\theta})}[\ell(y - f(\vec{x}))] \tag{5.33}$$

This is known as the **generalization error**. Our goal is to minimize the posterior expected loss, given by

$$\rho(f|\mathcal{D}) = \int p(\vec{\theta}|\mathcal{D}) L(\vec{\theta}, f) \mathrm{d}\vec{\theta} \tag{5.34}$$

This should be contrasted with the frequentist risk which is defined in Equation TODO.

5.7.2 The false positive vs false negative tradeoff

In this section, we focus on binary decision problems, such as hypothesis testing, two-class classification, object/ event detection, etc. There are two types of error we can make: a **false positive**(aka **false alarm**), or a **false negative**(aka **missed detection**). The 0-1 loss treats these two kinds of errors equivalently. However, we can consider the following more general loss matrix:

TODO

Chapter 6
Frequentist statistics

Attempts have been made to devise approaches to statistical inference that avoid treating parameters like random variables, and which thus avoid the use of priors and Bayes rule. Such approaches are known as **frequentist statistics**, **classical statistics** or **orthodox statistics**. Instead of being based on the posterior distribution, they are based on the concept of a sampling distribution.

6.1 Sampling distribution of an estimator

In frequentist statistics, a parameter estimate $\hat{\hat{\theta}}$ is computed by applying an **estimator** δ to some data $\mathcal{D}$, so $\hat{\hat{\theta}} = \delta(\mathcal{D})$. The parameter is viewed as fixed and the data as random, which is the exact opposite of the Bayesian approach. The uncertainty in the parameter estimate can be measured by computing the **sampling distribution** of the estimator. To understand this

6.1.1 Bootstrap

We might think of the bootstrap distribution as a "poor man's" Bayes posterior, see (Hastie et al. 2001, p235) for details.

6.1.2 Large sample theory for the MLE *

6.2 Frequentist decision theory

In frequentist or classical decision theory, there is a loss function and a likelihood, but there is no prior and hence no posterior or posterior expected loss. Thus there is no automatic way of deriving an optimal estimator, unlike the Bayesian case. Instead, in the frequentist approach, we are free to choose any estimator or decision procedure $f : \mathcal{X} \to \mathcal{Y}$ we want.

Having chosen an estimator, we define its expected loss or **risk** as follows:

$$\begin{aligned} R_{\text{exp}}(\theta, f) &\triangleq \mathbb{E}_{p(\tilde{\mathcal{D}}|\theta^*)}[L(\theta^*, f(\tilde{\mathcal{D}}))] \\ &= \int L(\theta^*, f(\tilde{\mathcal{D}})) p(\tilde{\mathcal{D}}|\theta^*) \mathrm{d}\tilde{\mathcal{D}} \end{aligned} \tag{6.1}$$

where $\tilde{\mathcal{D}}$ is data sampled from "nature's distribution", which is represented by parameter θ^*. In other words, the expectation is wrt the sampling distribution of the estimator. Compare this to the Bayesian posterior expected loss:

$$\rho(f|\mathcal{D},) \tag{6.2}$$

6.3 Desirable properties of estimators

6.4 Empirical risk minimization

6.4.1 Regularized risk minimization

6.4.2 Structural risk minimization

6.4.3 Estimating the risk using cross validation

6.4.4 Upper bounding the risk using statistical learning theory *

6.4.5 Surrogate loss functions

log-loss

$$L_{\text{nll}}(y, \eta) = -\log p(y|\vec{x}, \vec{w}) = \log(1 + e^{-y\eta}) \tag{6.3}$$

6.5 Pathologies of frequentist statistics *

Chapter 7
Linear Regression

7.1 Introduction

Linear regression is the "work horse" of statistics and (supervised) machine learning. When augmented with kernels or other forms of basis function expansion, it can model also nonlinear relationships. And when the Gaussian output is replaced with a Bernoulli or multinoulli distribution, it can be used for classification, as we will see below. So it pays to study this model in detail.

7.2 Representation

$$p(y|\vec{x},\vec{\theta}) = \mathcal{N}(y|\vec{w}^T\vec{x},\sigma^2) \tag{7.1}$$

where $\vec{w}$ and $\vec{x}$ are extended vectors, $\vec{x} = (1,x)$, $\vec{w} = (b,w)$.

Linear regression can be made to model non-linear relationships by replacing $\vec{x}$ with some non-linear function of the inputs, $\phi(\vec{x})$

$$p(y|\vec{x},\vec{\theta}) = \mathcal{N}(y|\vec{w}^T\phi(\vec{x}),\sigma^2) \tag{7.2}$$

This is known as **basis function expansion**. (Note that the model is still linear in the parameters $\vec{w}$, so it is still called linear regression; the importance of this will become clear below.) A simple example are polynomial basis functions, where the model has the form

$$\phi(x) = (1,x,\cdots,x^d) \tag{7.3}$$

7.3 MLE

Instead of maximizing the log-likelihood, we can equivalently minimize the **negative log likelihood** or **NLL**:

$$\text{NLL}(\vec{\theta}) \triangleq -\ell(\vec{\theta}) = -\log(\mathcal{D}|\vec{\theta}) \tag{7.4}$$

The NLL formulation is sometimes more convenient, since many optimization software packages are designed to find the minima of functions, rather than maxima.

Now let us apply the method of MLE to the linear regression setting. Inserting the definition of the Gaussian into the above, we find that the log likelihood is given by

$$\ell(\vec{\theta}) = \sum_{i=1}^{N}\log\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(y_i-\vec{w}^T\vec{x}_i)^2\right)\right] \tag{7.5}$$

$$= -\frac{1}{2\sigma^2}\text{RSS}(\vec{w}) - \frac{N}{2}\log(2\pi\sigma^2) \tag{7.6}$$

RSS stands for **residual sum of squares** and is defined by

$$\text{RSS}(\vec{w}) \triangleq \sum_{i=1}^{N}(y_i-\vec{w}^T\vec{x}_i)^2 \tag{7.7}$$

We see that the MLE for $\vec{w}$ is the one that minimizes the RSS, so this method is known as **least squares**.

Let's drop constants wrt $\vec{w}$ and NLL can be written as

$$\text{NLL}(\vec{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i-\vec{w}^T\vec{x}_i)^2 \tag{7.8}$$

There two ways to minimize $\text{NLL}(\vec{w})$.

7.3.1 OLS

Define $\vec{y} = (y_1,y_2,\cdots,y_N)$, $\vec{X} = \begin{pmatrix}\vec{x}_1^T\\ \vec{x}_2^T\\ \vdots\\ \vec{x}_N^T\end{pmatrix}$, then $\text{NLL}(\vec{w})$ can be written as

$$\text{NLL}(\vec{w}) = \frac{1}{2}(\vec{y}-\vec{X}\vec{w})^T(\vec{y}-\vec{X}\vec{w}) \tag{7.9}$$

When $\mathcal{D}$ is small(for example, $N < 1000$), we can use the following equation to compute $\vec{w}$ directly

$$\hat{\vec{w}}_{\text{OLS}} = (\vec{X}^T\vec{X})^{-1}\vec{X}^T\vec{y} \tag{7.10}$$

The corresponding solution $\hat{\vec{w}}_{\text{OLS}}$ to this linear system of equations is called the **ordinary least squares** or **OLS** solution.

Proof. We now state without proof some facts of matrix derivatives (we won't need all of these at this section).

$$trA \triangleq \sum_{i=1}^{n}A_{ii}$$

$$\frac{\partial}{\partial A}AB = B^T \tag{7.11}$$

$$\frac{\partial}{\partial A^T}f(A) = \left[\frac{\partial}{\partial A}f(A)\right]^T \tag{7.12}$$

$$\frac{\partial}{\partial A}ABA^TC = CAB + C^TAB^T \tag{7.13}$$

$$\frac{\partial}{\partial A}|A| = |A|(A^{-1})^T \tag{7.14}$$

Then,

$$\begin{aligned}
\text{NLL}(\vec{w}) &= \frac{1}{2N}(\vec{X}\vec{w}-\vec{y})^T(\vec{X}\vec{w}-\vec{y}) \\
\frac{\partial \text{NLL}}{\vec{w}} &= \frac{1}{2}\frac{\partial}{\vec{w}}(\vec{w}^T\vec{X}^T\vec{X}\vec{w}-\vec{w}^T\vec{X}^T\vec{y}-\vec{y}^T\vec{X}\vec{w}+\vec{y}^T\vec{y}) \\
&= \frac{1}{2}\frac{\partial}{\vec{w}}(\vec{w}^T\vec{X}^T\vec{X}\vec{w}-\vec{w}^T\vec{X}^T\vec{y}-\vec{y}^T\vec{X}\vec{w}) \\
&= \frac{1}{2}\frac{\partial}{\vec{w}}tr(\vec{w}^T\vec{X}^T\vec{X}\vec{w}-\vec{w}^T\vec{X}^T\vec{y}-\vec{y}^T\vec{X}\vec{w}) \\
&= \frac{1}{2}\frac{\partial}{\vec{w}}(tr\vec{w}^T\vec{X}^T\vec{X}\vec{w}-2tr\vec{y}^T\vec{X}\vec{w})
\end{aligned}$$

Combining Equations 7.12 and 7.13, we find that

$$\frac{\partial}{\partial A^T}ABA^TC = B^TA^TC^T + BA^TC$$

Let $A^T = \vec{w}, B = B^T = \vec{X}^T\vec{X}$, and $C = I$, Hence,

$$\begin{aligned}
\frac{\partial \text{NLL}}{\vec{w}} &= \frac{1}{2}(\vec{X}^T\vec{X}\vec{w}+\vec{X}^T\vec{X}\vec{w}-2\vec{X}^T\vec{y}) \\
&= \frac{1}{2}(\vec{X}^T\vec{X}\vec{w}-\vec{X}^T\vec{y}) \\
\frac{\partial \text{NLL}}{\vec{w}} &= 0 \Rightarrow \vec{X}^T\vec{X}\vec{w}-\vec{X}^T\vec{y}=0 \\
\vec{X}^T\vec{X}\vec{w} &= \vec{X}^T\vec{y} \\
\hat{\vec{w}}_{\text{OLS}} &= (\vec{X}^T\vec{X})^{-1}\vec{X}^T\vec{y}
\end{aligned} \tag{7.15}$$

Equation 7.15 is known as the **normal equation**.

7.3.1.1 Geometric interpretation

See Figure 7.1.

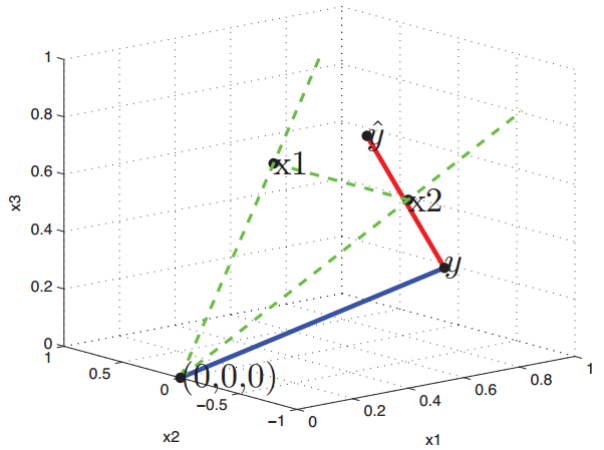


Fig. 7.1: Graphical interpretation of least squares for $N = 3$ examples and $D = 2$ features. $\tilde{\vec{x}}_1$ and $\tilde{\vec{x}}_2$ˇ are vectors in $\mathbb{R}^3$; together they define a 2D plane. $\vec{y}$ is also a vector in $\mathbb{R}^3$ but does not lie on this 2D plane. The orthogonal projection of $\vec{y}$ onto this plane is denoted $\hat{\vec{y}}$. The red line from $\vec{y}$ to $\hat{\vec{y}}$ is the residual, whose norm we want to minimize. For visual clarity, all vectors have been converted to unit norm.

To minimize the norm of the residual, $\vec{y}-\hat{\vec{y}}$, we want the residual vector to be orthogonal to every column of $\vec{X}$,soˇ $\tilde{\vec{x}}_j(\vec{y}-\hat{\vec{y}}) = 0$ for $j = 1 : D$. Hence

$$\begin{aligned}
\tilde{\vec{x}}_j(\vec{y}-\hat{\vec{y}}) = 0 &\Rightarrow \vec{X}^T(\vec{y}-\vec{X}\vec{w}) = 0 \\
&\Rightarrow \vec{w} = (\vec{X}^T\vec{X})^{-1}\vec{X}^T\vec{y}
\end{aligned} \tag{7.16}$$

7.3.2 SGD

When $\mathcal{D}$ is large, use stochastic gradient descent(SGD).

$$\because \frac{\partial}{\partial w_i}\text{NLL}(\vec{w}) = \sum_{i=1}^{N}(\vec{w}^T\vec{x}_i - y_i)x_{ij} \tag{7.17}$$

$$\begin{aligned}
\therefore w_j &= w_j - \alpha\frac{\partial}{\partial w_j}\text{NLL}(\vec{w}) \\
&= w_j - \sum_{i=1}^{N}\alpha(\vec{w}^T\vec{x}_i - y_i)x_{ij} \qquad (7.18) \\
\therefore \vec{w} &= \vec{w} - \alpha(\vec{w}^T\vec{x}_i - y_i)\vec{x} \qquad (7.19)
\end{aligned}$$

7.4 Ridge regression(MAP)

One problem with ML estimation is that it can result in overfitting. In this section, we discuss a way to ameliorate this problem by using MAP estimation with a Gaussian prior.

7.4.1 Basic idea

We can encourage the parameters to be small, thus resulting in a smoother curve, by using a zero-mean Gaussian prior:

$$p(\vec{w}) = \prod_j \mathcal{N}(w_j|0,\tau^2) \tag{7.20}$$

where $1/\tau^2$ controls the strength of the prior. The corresponding MAP estimation problem becomes

$$\arg\max_{\vec{w}} \sum_{i=1}^{N}\log\mathcal{N}(y_i|w_0+\vec{w}^T\vec{x}_i,\sigma^2) + \sum_{j=1}^{D}\log\mathcal{N}(w_j|0,\tau^2) \tag{7.21}$$

It is a simple exercise to show that this is equivalent to minimizing the following

$$J(\vec{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i-(w_0+\vec{w}^T\vec{x}_i))^2 + \lambda\|\vec{w}\|^2, \lambda \triangleq \frac{\sigma^2}{\tau^2} \tag{7.22}$$

Here the first term is the MSE/ NLL as usual, and the second term, $\lambda \geq 0$, is a complexity penalty. The corresponding solution is given by

$$\hat{\vec{w}}_{\text{ridge}} = (\lambda\vec{I}_D + \vec{X}^T\vec{X})^{-1}\vec{X}^T\vec{y} \tag{7.23}$$

This technique is known as **ridge regression**,or **penalized least squares**. In general, adding a Gaussian prior to the parameters of a model to encourage them to be small is called ℓ_2 **regularization** or **weight decay**. Note that the offset term w_0 is not regularized, since this just affects the height of the function, not its complexity.

We will consider a variety of different priors in this book. Each of these corresponds to a different form of **regularization**. This technique is very widely used to prevent overfitting.

7.4.2 Numerically stable computation *

$$\hat{w}_{\text{ridge}} = \vec{V}(\vec{Z}^T\vec{Z} + \lambda\vec{I}_N)^{-1}\vec{Z}^T\vec{y} \tag{7.24}$$

7.4.3 Connection with PCA *

7.4.4 Regularization effects of big data

Regularization is the most common way to avoid overfitting. However, another effective approach — which is not always available — is to use lots of data. It should be intuitively obvious that the more training data we have, the better we will be able to learn.

In domains with lots of data, simple methods can work surprisingly well (Halevy et al. 2009). However, there are still reasons to study more sophisticated learning methods, because there will always be problems for which we have little data. For example, even in such a data-rich domain as web search, as soon as we want to start personalizing the results, the amount of data available for any given user starts to look small again (relative to the complexity of the problem).

7.5 Bayesian linear regression

TODO

Chapter 8
Logistic Regression

8.1 Representation

Logistic regression can be binomial or multinomial. The **binomial logistic regression** model has the following form

$$p(y|\vec{x},\vec{w}) = \text{Ber}(y|\text{sigm}(\vec{w}^T\vec{x})) \tag{8.1}$$

where $\vec{w}$ and $\vec{x}$ are extended vectors, i.e., $\vec{w} = (b, w_1, w_2, \cdots, w_D), \vec{x} = (1, x_1, x_2, \cdots, x_D)$.

8.2 Optimization

8.2.1 MLE

$$\begin{aligned}
\ell(\vec{w}) &= \log\left\{\prod_{i=1}^{N}[\pi(\vec{x}_i)]^{y_i}[1-\pi(\vec{x}_i)]^{1-y_i}\right\} \\
&\quad \text{, where } \pi(\vec{x}) \triangleq P(y=1|\vec{x},\vec{w}) \\
&= \sum_{i=1}^{N}[y_i \log \pi(\vec{x}_i) + (1-y_i)\log(1-\pi(\vec{x}_i))] \\
&= \sum_{i=1}^{N}\left[y_i \log\frac{\pi(\vec{x}_i)}{1-\pi(\vec{x}_i)} + \log(1-\pi(\vec{x}_i))\right] \\
&= \sum_{i=1}^{N}[y_i(\vec{w}\cdot\vec{x}_i) - \log(1+\exp(\vec{w}\cdot\vec{x}_i))] \\
J(\vec{w}) &\triangleq \text{NLL}(\vec{w}) = -\ell(\vec{w}) \\
&= -\sum_{i=1}^{N}[y_i(\vec{w}\cdot\vec{x}_i) - \log(1+\exp(\vec{w}\cdot\vec{x}_i))]
\end{aligned} \tag{8.2}$$

Equation 8.2 is also called the **cross-entropy** error function (see Equation 2.53).

Unlike linear regression, we can no longer write down the MLE in closed form. Instead, we need to use an optimization algorithm to compute it, see Appendix A. For this, we need to derive the gradient and Hessian.

In the case of logistic regression, one can show that the gradient and Hessian of this are given by the following

$$\vec{g}(\vec{w}) = \frac{\text{d}J}{\text{d}\vec{w}} = \sum_{i=1}^{N}[\pi(\vec{x}_i) - y_i]\vec{x}_i = \vec{X}(\vec{\pi} - \vec{y}) \tag{8.3}$$

$$\begin{aligned}
\vec{H}(\vec{w}) &= \frac{\text{d}\vec{g}^T}{\text{d}\vec{w}} = \frac{\text{d}}{\text{d}\vec{w}}(\vec{\pi} - \vec{y})^T\vec{X}^T \\
&= \frac{\text{d}}{\text{d}\vec{w}}\vec{\pi}^T\vec{X}^T \\
&= (\pi(\vec{x}_i)(1-\pi(\vec{x}_i))\vec{x}_i, \cdots,)\vec{X}^T \\
&= \vec{X}\vec{S}\vec{X}^T, \quad \vec{S} \triangleq \text{diag}(\pi(\vec{x}_i)(1-\pi(\vec{x}_i))\vec{x}_i)
\end{aligned} \tag{8.4}$$

8.2.1.1 Iteratively reweighted least squares (IRLS)

TODO

8.2.2 MAP

Just as we prefer ridge regression to linear regression, so we should prefer MAP estimation for logistic regression to computing the MLE. ℓ_2 regularization

we can use ℓ_2 regularization, just as we did with ridge regression. We note that the new objective, gradient and Hessian have the following forms:

$$J'(\vec{w}) \triangleq \text{NLL}(\vec{w}) + \lambda\vec{w}^T\vec{w} \tag{8.5}$$

$$\vec{g}'(\vec{w}) = \vec{g}(\vec{w}) + \lambda\vec{w} \tag{8.6}$$

$$\vec{H}'(\vec{w}) = \vec{H}(\vec{w}) + \lambda\vec{I} \tag{8.7}$$

It is a simple matter to pass these modified equations into any gradient-based optimizer.

8.3 Multinomial logistic regression

8.3.1 Representation

Multinomial logistic regression model is also called a **maximum entropy classifier**, which has the following form

$$p(y=c|\vec{x},\vec{W}) = \frac{\exp(\vec{w}_c^T\vec{x})}{\sum_{c=1}^{C}\exp(\vec{w}_c^T\vec{x})} \tag{8.8}$$

8.3.2 MLE

Let $\vec{y}_i = (\mathbb{I}(y_i = 1), \mathbb{I}(y_i = 1), \cdots, \mathbb{I}(y_i = C))$, $\vec{\mu}_i = (p(y = 1|\vec{x}_i, \vec{W}), p(y = 2|\vec{x}_i, \vec{W}), \cdots, p(y = C|\vec{x}_i, \vec{W}))$, then the log-likelihood function can be written as

$$\ell(\vec{W}) = \log \prod_{i=1}^{N} \prod_{c=1}^{C} \mu_{ic}^{y_{ic}} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log \mu_{ic} \tag{8.9}$$

$$= \sum_{i=1}^{N} \left[\left(\sum_{c=1}^{C} y_{ic} \vec{w}_c^T \vec{x}_i \right) - \log \left(\sum_{c=1}^{C} \exp(\vec{w}_c^T \vec{x}_i) \right) \right] \tag{8.10}$$

Define the objective function as NLL

$$J(\vec{W}) = \text{NLL}(\vec{W}) = -\ell(\vec{W}) \tag{8.11}$$

Define $\vec{A} \otimes \vec{B}$ be the **kronecker product** of matrices $\vec{A}$ and $\vec{B}$.If $\vec{A}$ is an $m \times n$ matrix and $\vec{B}$ is a $p \times q$ matrix, then $\vec{A} \otimes \vec{B}$ is the $mp \times nq$ block matrix

$$\vec{A} \otimes \vec{B} \triangle \begin{pmatrix} a_{11}\vec{B} & \cdots & a_{1n}\vec{B} \\ \vdots & \vdots & \vdots \\ a_{m1}\vec{B} & \cdots & a_{mn}\vec{B} \end{pmatrix} \tag{8.12}$$

The gradient and Hessian are given by

$$\vec{g}(\vec{W}) = \sum_{i=1}^{N} (\vec{\mu} - \vec{y}_i) \otimes \vec{x}_i \tag{8.13}$$

$$\vec{H}(\vec{W}) = \sum_{i=1}^{N} (\text{diag}(\vec{\mu}_i) - \vec{\mu}_i \vec{\mu}_i^T) \otimes (\vec{x}_i \vec{x}_i^T) \tag{8.14}$$

where $\vec{y}_i = (\mathbb{I}(y_i = 1), \mathbb{I}(y_i = 1), \cdots, \mathbb{I}(y_i = C-1))$ and $\vec{\mu}_i = (p(y = 1|\vec{x}_i, \vec{W}), p(y = 2|\vec{x}_i, \vec{W}), \cdots, p(y = C-1|\vec{x}_i, \vec{W}))$ are column vectors of length $C-1$.

Pass them to any gradient-based optimizer.

8.3.3 MAP

The new objective

$$J'(\vec{W}) = \text{NLL}(\vec{w}) - \log p(\vec{W}) \tag{8.15}$$

$$\text{, where } p(\vec{W}) \triangleq \prod_{c=1}^{C} \mathcal{N}(\vec{w}_c | \vec{0}, \vec{V}_0)$$

$$= J(\vec{W}) + \frac{1}{2} \sum_{c=1}^{C} \vec{w}_c \vec{V}_0^{-1} \vec{w}_c \tag{8.16}$$

$$\tag{8.17}$$

Its gradient and Hessian are given by

$$\vec{g}'(\vec{w}) = \vec{g}(\vec{W}) + \vec{V}_0^{-1} \left(\sum_{c=1}^{C} \vec{w}_c \right) \tag{8.18}$$

$$\vec{H}'(\vec{w}) = \vec{H}(\vec{w}) + \vec{I}_C \otimes \vec{V}_0^{-1} \tag{8.19}$$

This can be passed to any gradient-based optimizer to find the MAP estimate. Note, however, that the Hessian has size $((CD)(CD))$, which is C times more row and columns than in the binary case, so limited memory BFGS is more appropriate than Newton's method.

8.4 Bayesian logistic regression

It is natural to want to compute the full posterior over the parameters, $p(\vec{w}|\mathcal{D})$, for logistic regression models. This can be useful for any situation where we want to associate confidence intervals with our predictions (e.g., this is necessary when solving contextual bandit problems, discussed in Section TODO).

Unfortunately, unlike the linear regression case, this cannot be done exactly, since there is no convenient conjugate prior for logistic regression. We discuss one simple approximation below; some other approaches include MCMC (Section TODO), variational inference (Section TODO), expectation propagation (Kuss and Rasmussen 2005), etc. For notational simplicity, we stick to binary logistic regression.

8.4.1 Laplace approximation

8.4.2 Derivation of the BIC

8.4.3 Gaussian approximation for logistic regression

8.4.4 Approximating the posterior predictive

8.4.5 Residual analysis (outlier detection) *

8.5 Online learning and stochastic optimization

Traditionally machine learning is performed **offline**, however, if we have **streaming data**, we need to perform **online learning**, so we can update our estimates as each new data point arrives rather than waiting until "the end" (which may never occur). And even if we have a batch of data, we might want to treat it like a stream if it is too large to hold in main memory. Below we discuss learning methods for this kind of scenario.

TODO

8.5.1 The perceptron algorithm

8.5.1.1 Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}(\vec{w}^T \vec{x} + b) \tag{8.20}$$

where $\text{sign}(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$, see Fig. 8.1[19].

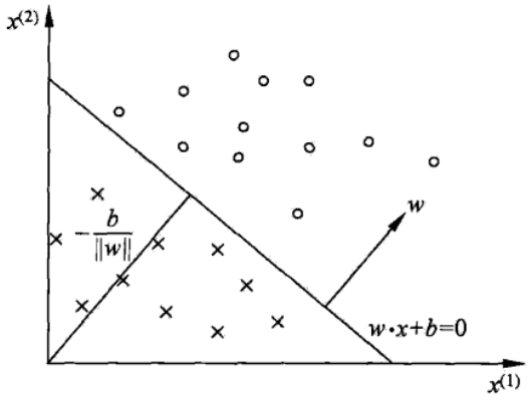


Fig. 8.1: Perceptron

8.5.1.2 Evaluation

$$L(\vec{w}, b) = -y_i(\vec{w}^T \vec{x}_i + b) \tag{8.21}$$
$$R_{emp}(f) = -\sum_i y_i(\vec{w}^T \vec{x}_i + b) \tag{8.22}$$
$$\tag{8.23}$$

8.5.1.3 Optimization

Primal form

$\vec{w} \leftarrow 0$; $b \leftarrow 0$; $k \leftarrow 0$;
while *no mistakes made within the for loop* **do**
  for $i \leftarrow 1$ **to** N **do**
    if $y_i(\vec{w} \cdot \vec{x}_i + b) \leq 0$ **then**
      $\vec{w} \leftarrow \vec{w} + \eta y_i \vec{x}_i$;
      $b \leftarrow b + \eta y_i$;
      $k \leftarrow k + 1$;
    end
  end
end

Algorithm 1: Perceptron learning algorithm, primal form, using SGD

Convergency

Theorem 8.1. ***(Novikoff)*** *If traning data set $\mathcal{D}$ is linearly separable, then*

1. *There exists a hyperplane denoted as $\widehat{\vec{w}}_{opt} \cdot \vec{x} + b_{opt} = 0$ which can correctly seperate all samples, and*

$$\exists \gamma > 0,\ \forall i,\ y_i(\vec{w}_{opt} \cdot \vec{x}_i + b_{opt}) \geq \gamma \tag{8.24}$$

2.

$$k \leq \left(\frac{R}{\gamma}\right)^2, \text{ where } R = \max_{1 \leq i \leq N} \left|\left|\widehat{\vec{x}}_i\right|\right| \tag{8.25}$$

Proof. (1) let $\gamma = \min_i y_i(\vec{w}_{opt} \cdot \vec{x}_i + b_{opt})$, then we get $y_i(\vec{w}_{opt} \cdot \vec{x}_i + b_{opt}) \geq \gamma$.

(2) The algorithm start from $\widehat{\vec{x}}_0 = 0$, if a instance is misclassified, then update the weight. Let $\widehat{\vec{w}}_{k-1}$ denotes the extended weight before the k-th misclassified instance, then we can get

$$y_i(\widehat{\vec{w}}_{k-1} \cdot \widehat{\vec{x}}_i) = y_i(\vec{w}_{k-1} \cdot \vec{x}_i + b_{k-1}) \leq 0 \tag{8.26}$$
$$\widehat{\vec{w}}_k = \widehat{\vec{w}}_{k-1} + \eta y_i \widehat{\vec{x}}_i \tag{8.27}$$

We could infer the following two equations, the proof procedure are omitted.

1. $\widehat{\vec{w}}_k \cdot \widehat{\vec{w}}_{opt} \geq k\eta\gamma$
2. $\left|\left|\widehat{\vec{w}}_k\right|\right|^2 \leq k\eta^2 R^2$

From above two equations we get

$$k\eta\gamma \leq \widehat{\vec{w}}_k \cdot \widehat{\vec{w}}_{opt} \leq \left|\left|\widehat{\vec{w}}_k\right|\right| \left|\left|\widehat{\vec{w}}_{opt}\right|\right| \leq \sqrt{k}\eta R$$
$$k^2\gamma^2 \leq kR^2$$
$$\text{i.e. } k \leq \left(\frac{R}{\gamma}\right)^2$$

Dual form

$$\vec{w} = \sum_{i=1}^N \alpha_i y_i \vec{x}_i \tag{8.28}$$
$$b = \sum_{i=1}^N \alpha_i y_i \tag{8.29}$$
$$f(\vec{x}) = \text{sign}\left(\sum_{j=1}^N \alpha_j y_j \vec{x}_j \cdot \vec{x} + b\right) \tag{8.30}$$

$\vec{\alpha} \leftarrow 0$; $b \leftarrow 0$; $k \leftarrow 0$;
while *no mistakes made within the for loop* **do**
  for $i \leftarrow 1$ **to** N **do**
    if $y_i\left(\sum_{j=1}^N \alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b\right) \leq 0$ **then**
      $\vec{\alpha} \leftarrow \vec{\alpha} + \eta$;
      $b \leftarrow b + \eta y_i$;
      $k \leftarrow k + 1$;
    end
  end
end

Algorithm 2: Perceptron learning algorithm, dual form

[19] https://en.wikipedia.org/wiki/Perceptron

8.6 Generative vs discriminative classifiers

8.6.1 Pros and cons of each approach

- **Easy to fit?** As we have seen, it is usually very easy to fit generative classifiers. For example, in Sections 3.5.1 and 4.2.4, we show that we can fit a naive Bayes model and an LDA model by simple counting and averaging. By contrast, logistic regression requires solving a convex optimization problem (see Section 8.2 for the details), which is much slower.
- **Fit classes separately?** In a generative classifier, we estimate the parameters of each class conditional density independently, so we do not have to retrain the model when we add more classes. In contrast, in discriminative models, all the parameters interact, so the whole model must be retrained if we add a new class. (This is also the case if we train a generative model to maximize a discriminative objective Salojarvi et al. (2005).)
- **Handle missing features easily?** Sometimes some of the inputs (components ofx) are not observed. In a generative classifier, there is a simple method for dealing with this, as we discuss in Section 8.6.2. However, in a discriminative classifier, there is no principled solution to this problem, since the model assumes that $\vec{x}$is always available to be conditioned on (although see (Marlin 2008) for some heuristic approaches).
- **Can handle unlabeled training data?** There is much interest in **semi-supervised learning**, which uses unlabeled data to help solve a supervised task. This is fairly easy to do using generative models (see e.g., (Lasserre et al. 2006; Liang et al. 2007)), but is much harder to do with discriminative models.
- **Symmetric in inputs and outputs?** We can run a generative model "backwards", and infer probable inputs given the output by computing $p(\vec{x}|y)$. This is not possible with a discriminative model. The reason is that a generative model defines a joint distribution on $\vec{x}$ and y, and hence treats both inputs and outputs symmetrically.
- **Can handle feature preprocessing?** A big advantage of discriminative methods is that they allow us to preprocess the input in arbitrary ways, e.g., we can replace $\vec{x}$ with $\phi(\vec{x})$, which could be some basis function expansion, etc. It is often hard to define a generative model on such pre-processed data, since the new features are correlated in complex ways.
- **Well-calibrated probabilities?** Some generative models, such as naive Bayes, make strong independence assumptions which are often not valid. This can result in very extreme posterior class probabilities (very near 0 or 1). Discriminative models, such as logistic regression, are usually better calibrated in terms of their probability estimates.

See Table 8.1 for a summary of the classification and regression techniques we cover in this book.

8.6.2 Dealing with missing data

Sometimes some of the inputs (components of $\vec{x}$) are not observed; this could be due to a sensor failure, or a failure to complete an entry in a survey, etc. This is called the **missing data problem** (Little. and Rubin 1987). The ability to handle missing data in a principled way is one of the biggest advantages of generative models.

To formalize our assumptions, we can associate a binary response variable $r_i \in \{0,1\}$ that specifies whether each value $\vec{x}_i$ is observed or not. The joint model has the form $p(\vec{x}_i, r_i|\vec{\theta}, \vec{\phi}) = p(r_i|\vec{x}_i, \vec{\phi})p(\vec{x}_i|\vec{\theta})$, where $\vec{\phi}$ are the parameters controlling whether the item is observed or not.

- If we assume $p(r_i|\vec{x}_i, \vec{\phi}) = p(r_i|\vec{\phi})$, we say the data is **missing completely at random** or **MCAR**.
- If we assume $p(r_i|\vec{x}_i, \vec{\phi}) = p(r_i|\vec{x}_i^o, \vec{\phi})$, where $\vec{x}_i^o$ is the observed part of $\vec{x}_i$, we say the data is **missing at random** or **MAR**.
- If neither of these assumptions hold, we say the data is **not missing at random** or **NMAR**. In this case, we have to model the missing data mechanism, since the pattern of missingness is informative about the values of the missing data and the corresponding parameters. This is the case in most collaborative filtering problems, for example.

See e.g., (Marlin 2008) for further discussion. We will henceforth assume the data is MAR.

When dealing with missing data, it is helpful to distinguish the cases when there is missingness only at test time (so the training data is **complete data**), from the harder case when there is missingness also at training time. We will discuss these two cases below. Note that the class label is always missing at test time, by definition; if the class label is also sometimes missing at training time, the problem is called semi-supervised learning.

8.6.2.1 Missing data at test time

In a generative classifier, we can handle features that are MAR by marginalizing them out. For example, if we are missing the value ofx1, we can compute

$$p(y=c|\vec{x}_{2:D}, \vec{\theta}) \propto p(y=c|\vec{\theta})p(\vec{x}_{2:D}|y=c, \vec{\theta}) \tag{8.31}$$

$$=\propto p(y=c|\vec{\theta}) \sum_{x_1} p(x_1, \vec{x}_{2:D}|y=c, \vec{\theta}) \tag{8.32}$$

Similarly, in discriminant analysis, no matter what regularization method was used to estimate the parameters, we can always analytically marginalize out the missing variables (see Section 4.3):

$$p(\vec{x}_{2:D}|y=c, \vec{\theta}) = \mathcal{N}(\vec{x}_{2:D}|\vec{\mu}_{c,2:D}, \vec{\Sigma}_{c,2:D}) \tag{8.33}$$

Model	Classif/regr	Gen/Discr	Param/Non	Section
Discriminant analysis	Classif	Gen	Param	Sec. 4.2.2, 4.2.4
Naive Bayes classifier	Classif	Gen	Param	Sec. 3.5, 3.5.1.2
Tree-augmented Naive Bayes classifier	Classif	Gen	Param	Sec. 10.2.1
Linear regression	Regr	Discrim	Param	Sec. 1.4.5, 7.3, 7.6
Logistic regression	Classif	Discrim	Param	Sec. 1.4.6, 8.2.1.1, 8.4.3, 21.8.1.1
Sparse linear/ logistic regression	Both	Discrim	Param	Ch. 13
Mixture of experts	Both	Discrim	Param	Sec. 11.2.4
Multilayer perceptron (MLP)/ Neural network	Both	Discrim	Param	Ch. 16
Conditional random field (CRF)	Classif	Discrim	Param	Sec. 19.6
K nearest neighbor classifier	Classif	Gen	Non	Sec. TODO, TODO
(Infinite) Mixture Discriminant analysis	Classif	Gen	Non	Sec. 14.7.3
Classification and regression trees (CART)	Both	Discrim	Non	Sec. 16.2
Boosted model	Both	Discrim	Non	Sec. 16.4
Sparse kernelized lin/logreg (SKLR)	Both	Discrim	Non	Sec. 14.3.2
Relevance vector machine (RVM)	Both	Discrim	Non	Sec. 14.3.2
Support vector machine (SVM)	Both	Discrim	Non	Sec. 14.5
Gaussian processes (GP)	Both	Discrim	Non	Ch. 15
Smoothing splines	Regr	Discrim	Non	Section 15.4.6

Table 8.1: List of various models for classification and regression which we discuss in this book. Columns are as follows: Model name; is the model suitable for classification, regression, or both; is the model generative or discriminative; is the model parametric or non-parametric; list of sections in book which discuss the model. See also http://pmtk3.googlecode.com/svn/trunk/docs/tutorial/html/tutSupervised.html for the PMTK equivalents of these models. Any generative probabilistic model (e.g., HMMs, Boltzmann machines, Bayesian networks, etc.) can be turned into a classifier by using it as a class conditional density

8.6.2.2 Missing data at training time

Missing data at training time is harder to deal with. In particular, computing the MLE or MAP estimate is no longer a simple optimization problem, for reasons discussed in Section TODO. However, soon we will study are a variety of more sophisticated algorithms (such as EM algorithm, in Section 11.4) for finding approximate ML or MAP estimates in such cases.

8.6.3 Fisher's linear discriminant analysis (FLDA) *

TODO

Chapter 9
Generalized linear models and the exponential family

9.1 The exponential family

Before defining the exponential family, we mention several reasons why it is important:

- It can be shown that, under certain regularity conditions, the exponential family is the only family of distributions with finite-sized sufficient statistics, meaning that we can compress the data into a fixed-sized summary without loss of information. This is particularly useful for online learning, as we will see later.
- The exponential family is the only family of distributions for which conjugate priors exist, which simplifies the computation of the posterior (see Section 9.1.5).
- The exponential family can be shown to be the family of distributions that makes the least set of assumptions subject to some user-chosen constraints (see Section 9.1.6).
- The exponential family is at the core of generalized linear models, as discussed in Section 9.2.
- The exponential family is at the core of variational inference, as discussed in Section TODO.

9.1.1 Definition

A pdf or pmf $p(\vec{x}|\vec{\theta})$,for $\vec{x} \in \mathbb{R}^m$ and $\vec{\theta} \in \mathbb{R}^D$, is said to be in the **exponential family** if it is of the form

$$p(\vec{x}|\vec{\theta}) = \frac{1}{Z(\vec{\theta})} h(\vec{x}) \exp[\vec{\theta}^T \phi(\vec{x})] \tag{9.1}$$

$$= h(\vec{x}) \exp[\vec{\theta}^T \phi(\vec{x}) - A(\vec{\theta})] \tag{9.2}$$

where

$$Z(\vec{\theta}) = \int h(\vec{x}) \exp[\vec{\theta}^T \phi(\vec{x})] \mathrm{d}\vec{x} \tag{9.3}$$

$$A(\vec{\theta}) = \log Z(\vec{\theta}) \tag{9.4}$$

Here $\vec{\theta}$ are called the **natural parameters** or **canonical parameters**, $\phi(\vec{x}) \in \mathbb{R}^D$ is called a vector of **sufficient statistics**, $Z(\vec{\theta})$ is called the **partition function**, $A(\vec{\theta})$ is called the **log partition function** or **cumulant function**, and $h(\vec{x})$ is the a scaling constant, often 1. If $\phi(\vec{x}) = \vec{x}$, we say it is a **natural exponential family**.

Equation 9.2 can be generalized by writing

$$p(\vec{x}|\vec{\theta}) = h(\vec{x}) \exp[\eta(\vec{\theta})^T \phi(\vec{x}) - A(\eta(\vec{\theta}))] \tag{9.5}$$

where η is a function that maps the parameters $\vec{\theta}$ to the canonical parameters $\vec{\eta} = \eta(\vec{\theta})$.If $\dim(\vec{\theta}) < \dim(\eta(\vec{\theta}))$, it is called a **curved exponential family**, which means we have more sufficient statistics than parameters. If $\eta(\vec{\theta}) = \vec{\theta}$, the model is said to be in **canonical form**. We will assume models are in canonical form unless we state otherwise.

9.1.2 Examples

9.1.2.1 Bernoulli

The Bernoulli for $x \in \{0,1\}$ can be written in exponential family form as follows:

$$\begin{aligned} \mathrm{Ber}(x|\mu) &= \mu^x (1-\mu)^{1-x} \\ &= \exp[x \log \mu + (1-x)\log(1-\mu)] \end{aligned} \tag{9.6}$$

where $\phi(x) = (\mathbb{I}(x=0), \mathbb{I}(x=1))$ and $\vec{\theta} = (\log \mu, \log(1-\mu))$.

However, this representation is **over-complete** since $\vec{1}^T \phi(x) = \mathbb{I}(x=0) + \mathbb{I}(x=1) = 1$. Consequently $\vec{\theta}$ is not uniquely identifiable. It is common to require that the representation be **minimal**, which means there is a unique θ associated with the distribution. In this case, we can just define

$$\mathrm{Ber}(x|\mu) = (1-\mu)\exp\left(x \log \frac{\mu}{1-\mu}\right) \tag{9.7}$$

$$\text{where } \phi(x) = x, \theta = \log \frac{\mu}{1-\mu}, Z = \frac{1}{1-\mu}$$

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

$$\mu = \mathrm{sigm}(\theta) = \frac{1}{1+e^{-\theta}} \tag{9.8}$$

9.1.2.2 Multinoulli

We can represent the multinoulli as a minimal exponential family as follows:

$$\begin{aligned}
\text{Cat}(\vec{x}|\vec{\mu}) &= \prod_{k=1}^{K} = \exp\left(\sum_{k=1}^{K} x_k \log \mu_k\right) \\
&= \exp\left[\sum_{k=1}^{K-1} x_k \log \mu_k + (1 - \sum_{k=1}^{K-1} x_k) \log(1 - \sum_{k=1}^{K-1} \mu_k)\right] \\
&= \exp\left[\sum_{k=1}^{K-1} x_k \log \frac{\mu_k}{1 - \sum_{k=1}^{K-1} \mu_k} + \log(1 - \sum_{k=1}^{K-1} \mu_k)\right] \\
&= \exp\left[\sum_{k=1}^{K-1} x_k \log \frac{\mu_k}{\mu_K} + \log \mu_K\right], \text{where } \mu_K \triangleq 1 - \sum_{k=1}^{K-1} \mu_k
\end{aligned}$$

We can write this in exponential family form as follows:

$$\text{Cat}(\vec{x}|\vec{\mu}) = \exp[\vec{\theta}^T \phi(\vec{x}) - A(\vec{\theta})] \tag{9.9}$$

$$\vec{\theta} \triangleq (\log \frac{\mu_1}{\mu_K}, \cdots, \log \frac{\mu_{K-1}}{\mu_K}) \tag{9.10}$$

$$\phi(\vec{x}) \triangleq (x_1, \cdots, x_{K-1}) \tag{9.11}$$

We can recover the mean parameters from the canonical parameters using

$$\mu_k = \frac{e^{\theta_k}}{1 + \sum_{j=1}^{K-1} e^{\theta_j}} \tag{9.12}$$

$$\mu_K = 1 - \frac{\sum_{j=1}^{K-1} e^{\theta_j}}{1 + \sum_{j=1}^{K-1} e^{\theta_j}} = \frac{1}{1 + \sum_{j=1}^{K-1} e^{\theta_j}} \tag{9.13}$$

and hence

$$A(\vec{\theta}] = -\log \mu_K = \log(1 + \sum_{j=1}^{K-1} e^{\theta_j}) \tag{9.14}$$

9.1.2.3 Univariate Gaussian

The univariate Gaussian can be written in exponential family form as follows:

$$\begin{aligned}
\mathcal{N}(x|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right] \\
&= \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2\right] \\
&= \frac{1}{Z(\vec{\theta})} \exp[\vec{\theta}^T \phi(x)]
\end{aligned} \tag{9.15}$$

where

$$\vec{\theta} = (\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}) \tag{9.16}$$

$$\phi(x) = (x, x^2) \tag{9.17}$$

$$Z(\vec{\theta}) = \sqrt{2\pi}\sigma \exp(\frac{\mu^2}{2\sigma^2}) \tag{9.18}$$

9.1.2.4 Non-examples

Not all distributions of interest belong to the exponential family. For example, the uniform distribution,$X \sim U(a,b)$, does not, since the support of the distribution depends on the parameters. Also, the Student T distribution (Section TODO) does not belong, since it does not have the required form.

9.1.3 Log partition function

An important property of the exponential family is that derivatives of the log partition function can be used to generate **cumulants** of the sufficient statistics.[20] For this reason, $A(\vec{\theta})$ is sometimes called a **cumulant function**. We will prove this for a 1-parameter distribution; this can be generalized to a K-parameter distribution in a straightforward way. For the first derivative we have

For the second derivative we have

$$\begin{aligned}
\frac{\mathrm{d}A}{\mathrm{d}\theta} &= \frac{\mathrm{d}}{\mathrm{d}\theta}\left\{\log \int \exp[\theta\phi(x)]\, h(x)\mathrm{d}x\right\} \\
&= \frac{\frac{\mathrm{d}}{\mathrm{d}\theta}\int \exp[\theta\phi(x)]\, h(x)\mathrm{d}x}{\int \exp[\theta\phi(x)]\, h(x)\mathrm{d}x} \\
&= \frac{\int \phi(x) exp[\theta\phi(x)]\, h(x)\mathrm{d}x}{\exp(A(\theta))} \\
&= \int \phi(x) \exp[\theta\phi(x) - A(\theta)]\, h(x)\mathrm{d}x \\
&= \int \phi(x) p(x)\mathrm{d}x = \mathbb{E}[\phi(x)]
\end{aligned} \tag{9.19}$$

For the second derivative we have

$$\begin{aligned}
\frac{\mathrm{d}^2 A}{\mathrm{d}\theta^2} &= \int \phi(x) \exp[\theta\phi(x) - A(\theta)]\, h(x) \left[\phi(x) - A'(\theta)\right] \mathrm{d}x \\
&= \int \phi(x) p(x) \left[\phi(x) - A'(\theta)\right] \mathrm{d}x \\
&= \int \phi^2(x) p(x)\mathrm{d}x - A'(\theta) \int \phi(x) p(x)\mathrm{d}x \\
&= \mathbb{E}[\phi^2(x)] - \mathbb{E}[\phi(x)]^2 = \text{var}[\phi(x)]
\end{aligned} \tag{9.20}$$

In the multivariate case, we have that

$$\frac{\partial^2 A}{\partial\theta_i \partial\theta_j} = \mathbb{E}[\phi_i(x)\phi_j(x)] - \mathbb{E}[\phi_i(x)]\mathbb{E}[\phi_j(x)] \tag{9.21}$$

and hence

$$\nabla^2 A(\vec{\theta}) = \text{cov}[\phi(\vec{x})] \tag{9.22}$$

Since the covariance is positive definite, we see that $A(\vec{\theta})$ is a convex function (see Section A.1).

9.1.4 MLE for the exponential family

The likelihood of an exponential family model has the form

[20] The first and second cumulants of a distribution are its mean $\mathbb{E}[X]$ and variance $\text{var}[X]$, whereas the first and second moments are its mean $\mathbb{E}[X]$ and $\mathbb{E}[X^2]$.

$$p(\mathcal{D}|\vec{\theta}) = \left[\prod_{i=1}^{N} h(\vec{x}_i)\right] g(\vec{\theta})^N \exp\left[\vec{\theta}^T \left(\sum_{i=1}^{N} \phi(\vec{x}_i)\right)\right] \tag{9.23}$$

We see that the sufficient statistics are N and

$$\phi(\mathcal{D}) = \sum_{i=1}^{N} \phi(\vec{x}_i) = (\sum_{i=1}^{N} \phi_1(\vec{x}_i), \cdots, \sum_{i=1}^{N} \phi_K(\vec{x}_i)) \tag{9.24}$$

The **Pitman-Koopman-Darmois theorem** states that, under certain regularity conditions, the exponential family is the only family of distributions with finite sufficient statistics. (Here, finite means of a size independent of the size of the data set.)

One of the conditions required in this theorem is that the support of the distribution not be dependent on the parameter.

9.1.5 Bayes for the exponential family

TODO

9.1.5.1 Likelihood

9.1.6 Maximum entropy derivation of the exponential family *

9.2 Generalized linear models (GLMs)

Linear and logistic regression are examples of **generalized linear models**, or **GLM**s (McCullagh and Nelder 1989). These are models in which the output density is in the exponential family (Section 9.1), and in which the mean parameters are a linear combination of the inputs, passed through a possibly nonlinear function, such as the logistic function. We describe GLMs in more detail below. We focus on scalar outputs for notational simplicity. (This excludes multinomial logistic regression, but this is just to simplify the presentation.)

9.2.1 Basics

9.3 Probit regression

9.4 Multi-task learning

Chapter 10
Directed graphical models (Bayes nets)

10.1 Introduction

10.1.1 Chain rule

$$p(x_{1:V}) = p(x_1)p(x_2|x_1)p(x_3|x_{1:2})\cdots p(x_N|x_{1:V-1}) \quad (10.1)$$

10.1.2 Conditional independence

X and Y are **conditionally independent** given Z, denoted $X \perp Y|Z$, iff the conditional joint can be written as a product of conditional marginals, i.e.

$$X \perp Y|Z \iff p(X,Y|Z) = p(X|Z)p(Y|Z) \quad (10.2)$$

first order **Markov assumption**: the future is independent of the past given the present,

$$x_{t+1} \perp x_{1:t-1}|x_t \quad (10.3)$$

first-order **Markov chain**

$$p(x_{1:V}) = p(x_1)\prod_{t=2}^{V} p(x_t|x_{t-1}) \quad (10.4)$$

10.1.3 Graphical models

A **graphical model**(GM) is a way to represent a joint distribution by making CI assumptions. In particular, the nodes in the graph represent random variables, and the (lack of) edges represent CI assumptions.

There are several kinds of graphical model, depending on whether the graph is directed, undirected, or some combination of directed and undirected. In this chapter, we just study directed graphs. We consider undirected graphs in Chapter 19.

10.1.4 Directed graphical model

A **directed graphical model**or **DGM** is a GM whose graph is a DAG. These are more commonly known as **Bayesian networks**. However, there is nothing inherently "Bayesian" about Bayesian networks: they are just a way of defining probability distributions. These models are also called **belief networks**. The term "belief" here refers to subjective probability. Once again, there is nothing inherently subjective about the kinds of probability distributions represented by DGMs.

Ordered Markov property

$$x_s \perp x_{\text{pred}(s)\setminus\text{pa}(s)} \perp x_{\text{pa}(s)} \quad (10.5)$$

where $\text{pa}(s)$ are the parents of nodes, and $\text{pred}(s)$ are the predecessors of nodes in the DAG.

Markov chain on a DGM

$$p(x_{1:V}|G) = \prod_{t=1}^{V} p(x_t|x_{\text{pa}(t)}) \quad (10.6)$$

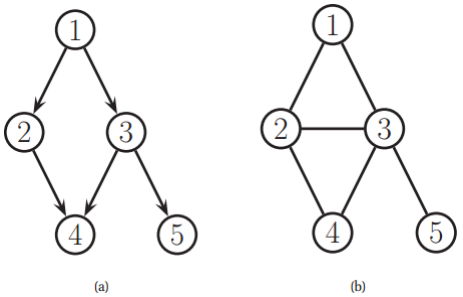


Fig. 10.1: (a) A simple DAG on 5 nodes, numbered in topological order. Node 1 is the root, nodes 4 and 5 are the leaves. (b) A simple undirected graph, with the following maximal cliques: 1,2,3, 2,3,4, 3,5.

10.2 Examples

10.2.1 Naive Bayes classifiers

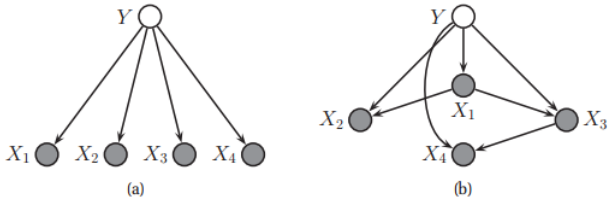


Fig. 10.2: (a) A naive Bayes classifier represented as a DGM. We assume there are $D = 4$ features, for simplicity. Shaded nodes are observed, unshaded nodes are hidden. (b) Tree-augmented naive Bayes classifier for $D = 4$ features. In general, the tree topology can change depending on the value of y.

10.2.2 Markov and hidden Markov models

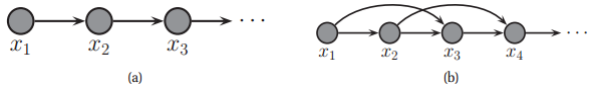


Fig. 10.3: A first and second order Markov chain.

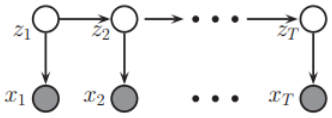


Fig. 10.4: A first-order HMM.

10.3 Inference

Suppose we have a set of correlated random variables with joint distribution $p(\vec{x}_{1:V}|\vec{\theta})$. Let us partition this vector into the **visible variables** $\vec{x}_v$, which are observed, and the **hidden variables**, $\vec{x}_h$, which are unobserved. Inference refers to computing the posterior distribution of the unknowns given the knowns:

$$p(\vec{x}_h|\vec{x}_v,\theta) = \frac{p(\vec{x}_h,\vec{x}_v|\vec{\theta})}{p(\vec{x}_v|\vec{\theta})} = \frac{p(\vec{x}_h,\vec{x}_v|\vec{\theta})}{\sum_{\vec{x}'_h} p(\vec{x}'_h,\vec{x}_v|\vec{\theta})} \tag{10.7}$$

Sometimes only some of the hidden variables are of interest to us. So let us partition the hidden variables into **query variables**, $\vec{x}_q$, whose value we wish to know, and the remaining **nuisance variables**, $\vec{x}_n$, which we are not interested in. We can compute what we are interested in by **marginalizing out** the nuisance variables:

$$p(\vec{x}_q|\vec{x}_v,\vec{\theta}) = \sum_{\vec{x}_n} p(\vec{x}_q,\vec{x}_n|\vec{x}_v,\vec{\theta}) \tag{10.8}$$

10.4 Learning

MAP estimate:

$$\hat{\vec{\theta}} = \arg\max_{\vec{\theta}} \sum_{i=1}^{N} \log p(\vec{x}_{i,v}|\vec{\theta}) + \log p(\vec{\theta}) \tag{10.9}$$

10.4.1 Learning from complete data

If all the variables are fully observed in each case, so there is no missing data and there are no hidden variables, we say the data is **complete**. For a DGM with complete data, the likelihood is given by

$$\begin{aligned} p(\mathcal{D}|\vec{\theta}) &= \prod_{i=1}^{N} p(\vec{x}_i|\vec{\theta}) \\ &= \prod_{i=1}^{N}\prod_{t=1}^{V} p(\vec{x}_{it}|\vec{x}_{i,\mathrm{pa}(t)},\vec{\theta}_t) \\ &= \prod_{t=1}^{V} p(\mathcal{D}_t|\vec{\theta}_t) \end{aligned} \tag{10.10}$$

where $\mathcal{D}_t$ is the data associated with node t and its parents, i.e., the t'th family.

Now suppose that the prior factorizes as well:

$$p(\vec{\theta}) = \prod_{t=1}^{V} p(\vec{\theta}_t) \tag{10.11}$$

Then clearly the posterior also factorizes:

$$p(\vec{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\vec{\theta})p(\vec{\theta}) = \prod_{t=1}^{V} p(\mathcal{D}_t|\vec{\theta}_t)p(\vec{\theta}_t) \tag{10.12}$$

10.4.2 Learning with missing and/or latent variables

If we have missing data and/or hidden variables, the likelihood no longer factorizes, and indeed it is no longer convex, as we explain in detail in Section TODO. This means we will usually can only compute a locally optimal ML or MAP estimate. Bayesian inference of the parameters is even harder. We discuss suitable approximate inference techniques in later chapters.

10.5 Conditional independence properties of DGMs

10.5.1 d-separation and the Bayes Ball algorithm (global Markov properties)

1. P contains a chain

$$\begin{aligned} p(x,z|y) &= \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(y|x)p(z|y)}{p(y)} \\ &= \frac{p(x,y)p(z|y)}{p(y)} = p(x|y)p(z|y) \end{aligned} \tag{10.13}$$

2. P contains a fork

$$p(x,z|y) = \frac{p(x,y,z)}{p(y)} = \frac{p(y)p(x|y)p(z|y)}{p(y)} = p(x|y)p(z|y) \tag{10.14}$$

3. P contains v-structure

$$p(x,z|y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(z)p(y|x,z)}{p(y)} \neq p(x|y)p(z|y) \tag{10.15}$$

10.5.2 Other Markov properties of DGMs

10.5.3 Markov blanket and full conditionals

$$mb(t) = ch(t) \cup pa(t) \cup copa(t) \tag{10.16}$$

10.5.4 Multinoulli Learning

Multinoulli Distribution:

$$Cat(x|\mu) = \prod_{k=1}^{K} \mu_k^{x_k} \tag{10.17}$$

then from **??** and 10.17:

$$p(x|G,\theta) = \prod_{v=1}^{V} \prod_{c=1}^{C_v} \prod_{k=1}^{K} \theta_{vck}^{y_{vck}} \tag{10.18}$$

Likelihood

$$p(D|G,\theta) = \prod_{n=1}^{N} p(x_n|G,\theta) = \prod_{n=1}^{N} \prod_{v=1}^{V} \prod_{c=1}^{C_{nv}} \prod_{k=1}^{K} \theta_{vck}^{y_{nvck}} \tag{10.19}$$

where $y_{nv} = f(pa(x_{nv}))$, f(x) is a map from x to a vector, there is only one element in the vector is 1.

10.6 Influence (decision) diagrams *

Chapter 11
Mixture models and the EM algorithm

11.1 Latent variable models

In Chapter 10 we showed how graphical models can be used to define high-dimensional joint probability distributions. The basic idea is to model dependence between two variables by adding an edge between them in the graph. (Technically the graph represents conditional independence, but you get the point.)

An alternative approach is to assume that the observed variables are correlated because they arise from a hidden common "cause". Model with hidden variables are also known as **latent variable models** or **LVM**s. As we will see in this chapter, such models are harder to fit than models with no latent variables. However, they can have significant advantages, for two main reasons.

- First, LVMs often have fewer parameters than models that directly represent correlation in the visible space.
- Second, the hidden variables in an LVM can serve as a **bottleneck**, which computes a compressed representation of the data. This forms the basis of unsupervised learning, as we will see. Figure 11.1 illustrates some generic LVM structures that can be used for this purpose.

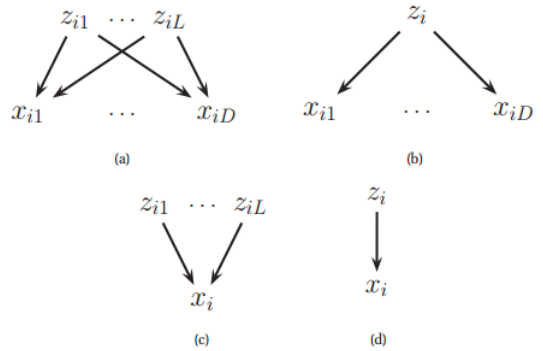


Fig. 11.1: A latent variable model represented as a DGM. (a) Many-to-many. (b) One-to-many. (c) Many-to-one. (d) One-to-one.

11.2 Mixture models

The simplest form of LVM is when $z_i \in \{1, \cdots, K\}$, representing a discrete latent state. We will use a discrete prior for this, $p(zi) = \text{Cat}(\pi)$. For the likelihood, we use $p(\vec{x}_i|z_i = k) = p_k(\vec{x}_i)$, where p_k is the k'th **base distribution** for the observations; this can be of any type. The overall model is known as a **mixture model**, since we are mixing together the K base distributions as follows:

$$p(\vec{x}_i|\vec{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\vec{x}_i|\vec{\theta}) \tag{11.1}$$

Depending on the form of the likelihood $p(\vec{x}_i|\vec{z}_i)$ and the prior $p(\vec{z}_i)$, we can generate a variety of different models, as summarized in Table 11.1.

$p(\vec{x}_i\|\vec{z}_i)$	$p(\vec{z}_i)$	**Name**	**Section**
MVN	Discrete	Mixture of Gaussians	11.2.1
Prod. Discrete	Discrete	Mixture of multinomials	11.2.2
Prod. Gaussian	Prod. Gaussian	Factor analysis/ probabilistic PCA	12.1.5
Prod. Gaussian	Prod. Laplace	Probabilistic ICA/ sparse coding	12.6
Prod. Discrete	Prod. Gaussian	Multinomial PCA	27.2.3
Prod. Discrete	Dirichlet	Latent Dirichlet allocation	27.3
Prod. Noisy-OR	Prod. Bernoulli	BN20/ QMR	10.2.3
Prod. Bernoulli	Prod. Bernoulli	Sigmoid belief net	27.7

Table 11.1: Summary of some popular directed latent variable models. Here "Prod" means product, so "Prod. Discrete" in the likelihood means a factored distribution of the form $\prod_j \text{Cat}(x_{ij}|\vec{z}_i)$, and "Prod. Gaussian" means a factored distribution of the form $\prod_j \mathcal{N}(x_{ij}|\vec{z}_i)$.

11.2.1 Mixtures of Gaussians

$$p_k(\vec{x}_i|\vec{\theta}) = \mathcal{N}(\vec{x}_i|\vec{\mu}_k, \Sigma_k) \tag{11.2}$$

11.2.2 Mixtures of multinoullis

$$p_k(\vec{x}_i|\vec{\theta}) = \prod_{j=1}^{D} \text{Ber}(x_{ij}|\mu_{jk}) = \prod_{j=1}^{D} \mu_{jk}^{x_{ij}} (1-\mu_{jk})^{1-x_{ij}} \tag{11.3}$$

where μ_{jk} is the probability that bit j turns on in cluster k.

The latent variables do not have to any meaning, we might simply introduce latent variables in order to make the model more powerful. For example, one can show that the mean and covariance of the mixture distribution are given by

$$\mathbb{E}[\vec{x}] = \sum_{k=1}^{K} \pi_k \vec{\mu}_k \tag{11.4}$$

$$\text{Cov}[\vec{x}] = \sum_{k=1}^{K} \pi_k (\Sigma_k + \vec{\mu}_k \vec{\mu}_k^T) - \mathbb{E}[\vec{x}]\mathbb{E}[\vec{x}]^T \tag{11.5}$$

where $\Sigma_k = \text{diag}(\mu_{jk}(1-\mu_{jk}))$. So although the component distributions are factorized, the joint distribution is not. Thus the mixture distribution can capture correlations between variables, unlike a single product-of-Bernoullis model.

11.2.3 Using mixture models for clustering

There are two main applications of mixture models, black-box density model(see Section 14.7.3 TODO) and clustering(see Chapter 25 TODO).

Soft clustering

$$r_{ik} \triangleq p(z_i = k|\vec{x}_i, \vec{\theta}) = \frac{p(z_i = k, \vec{x}_i|\vec{\theta})}{p(\vec{x}_i|\vec{\theta})} = \frac{p(z_i = k|\vec{\theta})p(\vec{x}_i|z_i = k, \vec{\theta})}{\sum_{k'=1}^{K} p(z_i = k'|\vec{\theta})p(\vec{x}_i|z_i = k', \vec{\theta})} \tag{11.6}$$

where r_{ik} is known as the **responsibility** of cluster k for point i.

Hard clustering

$$z_i^* \triangleq \arg\max_k r_{ik} = \arg\max_k p(z_i = k|\vec{x}_i, \vec{\theta}) \tag{11.7}$$

The difference between generative classifiers and mixture models only arises at training time: in the mixture case, we never observe z_i, whereas with a generative classifier, we do observe y_i(which plays the role of z_i).

11.2.4 Mixtures of experts

Section 14.7.3 TODO described how to use mixture models in the context of generative classifiers. We can also use them to create discriminative models for classification and regression. For example, consider the data in Figure 11.2(a). It seems like a good model would be three different linear regression functions, each applying to a different part of the input space. We can model this by allowing the mixing weights and the mixture densities to be input-dependent:

$$p(y_i|\vec{x}_i, z_i = k, \vec{\theta}) = \mathcal{N}(y_i|\vec{w}_k^T\vec{x}, \sigma_k^2) \tag{11.8}$$

$$p(z_i|\vec{x}_i, \vec{\theta}) = \text{Cat}(z_i|\mathcal{S}(\vec{V}^T\vec{x}_i)) \tag{11.9}$$

See Figure 11.3(a) for the DGM.

This model is called a **mixture of experts** or MoE (Jordan and Jacobs 1994). The idea is that each submodel is considered to be an "expert" in a certain region of input space. The function $p(z_i|\vec{x}_i, \vec{\theta})$ is called a **gating function**, and decides which expert to use, depending on the input values. For example, Figure 11.2(b) shows how the three experts have "carved up" the 1d input space, Figure 11.2(a) shows the predictions of each expert individually (in this case, the experts are just linear regression models), and Figure 11.2(c)

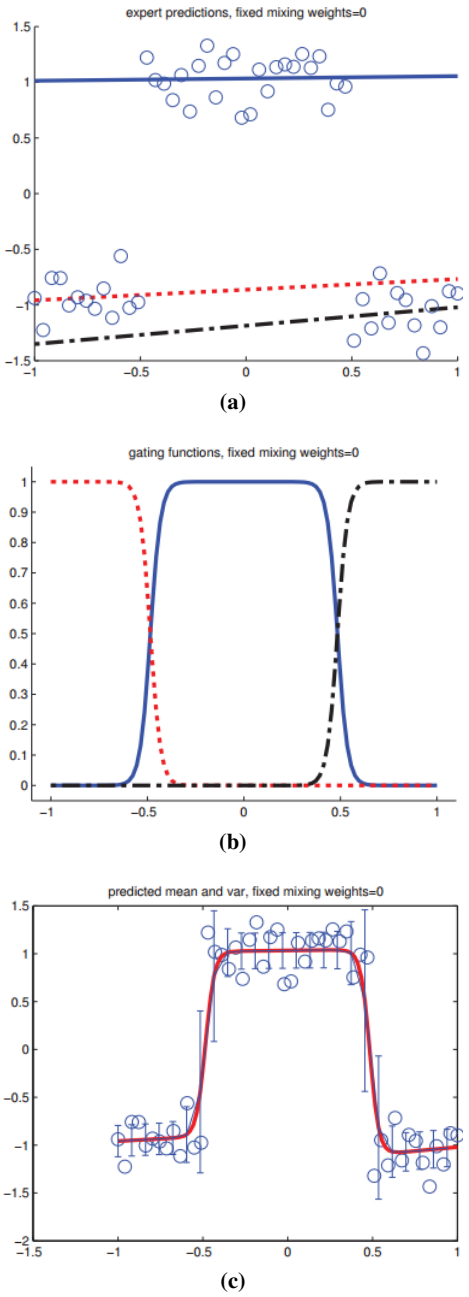


Fig. 11.2: (a) Some data fit with three separate regression lines. (b) Gating functions for three different "experts". (c) The conditionally weighted average of the three expert predictions.

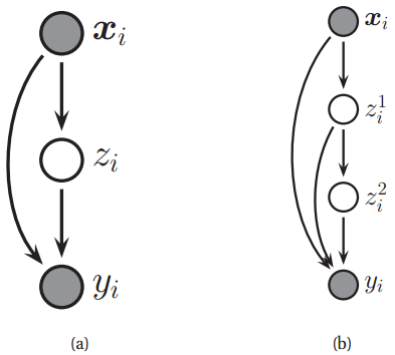


Fig. 11.3: (a) A mixture of experts. (b) A hierarchical mixture of experts.

shows the overall prediction of the model, obtained using

$$p(y_i|\vec{x}_i,\vec{\theta}) = \sum_{k=1}^{K} p(z_i = k|\vec{x}_i,\vec{\theta})p(y_i|\vec{x}_i,z_i = k,\vec{\theta}) \quad (11.10)$$

We discuss how to fit this model in Section TODO 11.4.3.

11.3 Parameter estimation for mixture models

11.3.1 Unidentifiability

11.3.2 Computing a MAP estimate is non-convex

11.4 The EM algorithm

11.4.1 Introduction

For many models in machine learning and statistics, computing the ML or MAP parameter estimate is easy provided we observe all the values of all the relevant random variables, i.e., if we have complete data. However, if we have missing data and/or latent variables, then computing the ML/MAP estimate becomes hard.

One approach is to use a generic gradient-based optimizer to find a local minimum of the $\text{NLL}(\vec{\theta})$. However, we often have to enforce constraints, such as the fact that covariance matrices must be positive definite, mixing weights must sum to one, etc., which can be tricky. In such cases, it is often much simpler (but not always faster) to use an algorithm called **expectation maximization**,or **EM** for short (Dempster et al. 1977; Meng and van Dyk 1997; McLachlan and Krishnan 1997). This is is an efficient iterative algorithm to compute the ML or MAP estimate in the presence of missing or hidden data, often with closed-form updates at each step. Furthermore, the algorithm automatically enforce the required constraints.

See Table 11.2 for a summary of the applications of EM in this book.

Table 11.2: Some models discussed in this book for which EM can be easily applied to find the ML/ MAP parameter estimate.

Model	Section
Mix. Gaussians	11.4.2
Mix. experts	11.4.3
Factor analysis	12.1.5
Student T	11.4.5
Probit regression	11.4.6
DGM with hidden variables	11.4.4
MVN with missing data	11.6.1
HMMs	17.5.2
Shrinkage estimates of Gaussian means	Exercise 11.13

11.4.2 Basic idea

EM exploits the fact that if the data were fully observed, then the ML/ MAP estimate would be easy to compute. In particular, each iteration of the EM algorithm consists of two processes: The E-step, and the M-step.

- In the **E-step**, the missing data are inferred given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation, explaining the choice of terminology.
- In the **M-step**, the likelihood function is maximized under the assumption that the missing data are known. The missing data inferred from the E-step are used in lieu of the actual missing data.

Let $\vec{x}_i$ be the visible or observed variables in case i, and let $\vec{z}_i$ be the hidden or missing variables. The goal is to maximize the log likelihood of the observed data:

$$\ell(\vec{\theta}) = \log p(\mathcal{D}|\vec{\theta}) = \sum_{i=1}^{N} \log p(\vec{x}_i|\vec{\theta}) = \sum_{i=1}^{N} \log \sum_{\vec{z}_i} p(\vec{x}_i,\vec{z}_i|\vec{\theta}) \quad (11.11)$$

Unfortunately this is hard to optimize, since the log cannot be pushed inside the sum.

EM gets around this problem as follows. Define the **complete data log likelihood** to be

$$\ell_c(\vec{\theta}) = \sum_{i=1}^{N} \log p(\vec{x}_i,\vec{z}_i|\vec{\theta}) \quad (11.12)$$

This cannot be computed, since $\vec{z}_i$ is unknown. So let us define the **expected complete data log likelihood** as follows:

$$Q(\vec{\theta},\vec{\theta}^{t-1}) \triangleq \mathbb{E}_{\vec{z}|\mathcal{D},\theta^{t-1}}\left[\ell_c(\vec{\theta})\right] = \mathbb{E}\left[\ell_c(\vec{\theta})|\mathcal{D},\theta^{t-1}\right] \quad (11.13)$$

where t is the current iteration number. Q is called the **auxiliary function**(see Section 11.4.9 for derivation). The expectation is taken wrt the old parameters, $\vec{\theta}^{t-1}$, and the observed data $\mathcal{D}$. The goal of the E-step is to compute $Q(\vec{\theta},\vec{\theta}^{t-1})$, or rather, the parameters inside of it which the MLE(or MAP) depends on; these are known as the **expected sufficient statistics** or **ESS**. In the M-step, we optimize the Q function wrt $\vec{\theta}$:

$$\vec{\theta}^t = \arg\max_{\vec{\theta}} Q(\vec{\theta},\vec{\theta}^{t-1}) \quad (11.14)$$

To perform MAP estimation, we modify the M-step as follows:

$$\vec{\theta}^t = \arg\max_{\vec{\theta}} Q(\vec{\theta},\vec{\theta}^{t-1}) + \log p(\vec{\theta}) \quad (11.15)$$

The E step remains unchanged.

In summary, the EM algorithm's pseudo code is as follows

Below we explain how to perform the E and M steps for several simple models, that should make things clearer.

input : observed data $\mathcal{D} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N\}$,joint distribution $P(\vec{x}, \vec{z}|\vec{\theta})$
output: model's parameters $\vec{\theta}$
// 1. identify hidden variables $\vec{z}$, write out the log likelihood function $\ell(\vec{x}, \vec{z}|\vec{\theta})$
$\vec{\theta}^{(0)}$ = ... // initialize
while *(!convergency)* **do**
  // 2. E-step: plug in $P(\vec{x}, \vec{z}|\vec{\theta})$, derive the formula of $Q(\vec{\theta}, \vec{\theta}^{t-1})$
  $Q(\vec{\theta}, \vec{\theta}^{t-1}) = \mathbb{E}\left[\ell_c(\vec{\theta})|\mathcal{D}, \theta^{t-1}\right]$
  // 3. M-step: find $\vec{\theta}$ that maximizes the value of $Q(\vec{\theta}, \vec{\theta}^{t-1})$
  $\vec{\theta}^t = \arg\max_{\vec{\theta}} Q(\vec{\theta}, \vec{\theta}^{t-1})$
end

Algorithm 3: EM algorithm

11.4.3 EM for GMMs

11.4.3.1 Auxiliary function

$$\begin{aligned} Q(\vec{\theta}, \vec{\theta}^{t-1}) &= \mathbb{E}_{z|\mathcal{D},\theta^{t-1}}\left[\ell_c(\vec{\theta})\right] \\ &= \mathbb{E}_{z|\mathcal{D},\theta^{t-1}}\left[\sum_{i=1}^{N} \log p(\vec{x}_i, z_i|\vec{\theta})\right] \\ &= \sum_{i=1}^{N} \mathbb{E}_{z|\mathcal{D},\theta^{t-1}}\left\{\log\left[\prod_{k=1}^{K}\left(\pi_k p(\vec{x}_i|\vec{\theta}_k)\right)^{\mathbb{I}(z_i=k)}\right]\right\} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}[\mathbb{I}(z_i=k)] \log\left[\pi_k p(\vec{x}_i|\vec{\theta}_k)\right] \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} p(z_i=k|\vec{x}_i, \vec{\theta}^{t-1}) \log\left[\pi_k p(\vec{x}_i|\vec{\theta}_k)\right] \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\log\pi_k + \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\log p(\vec{x}_i|\vec{\theta}_k) \end{aligned} \tag{11.16}$$

where $r_{ik} \triangleq \mathbb{E}[\mathbb{I}(z_i=k)] = p(z_i=k|\vec{x}_i, \vec{\theta}^{t-1})$ is the **responsibility** that cluster k takes for data point i. This is computed in the E-step, described below.

11.4.3.2 E-step

The E-step has the following simple form, which is the same for any mixture model:

$$\begin{aligned} r_{ik} = p(z_i=k|\vec{x}_i, \vec{\theta}^{t-1}) &= \frac{p(z_i=k, \vec{x}_i|\vec{\theta}^{t-1})}{p(\vec{x}_i|\vec{\theta}^{t-1})} \\ &= \frac{\pi_k p(\vec{x}_i|\vec{\theta}_k^{t-1})}{\sum_{k'=1}^{K} \pi_{k'} p(\vec{x}_i|\vec{\theta}_k^{t-1})} \end{aligned} \tag{11.17}$$

11.4.3.3 M-step

In the M-step, we optimize Q wrt $\vec{\pi}$ and $\vec{\theta}_k$.

For $\vec{\pi}$, grouping together only the terms that depend on π_k, we find that we need to maximize $\sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\log\pi_k$. However, there is an additional constraint $\sum_{k=1}^{K}\pi_k = 1$, since they represent the probabilities $\vec{\pi}_k = P(z_i=k)$. To deal with the constraint we construct the Lagrangian

$$\mathcal{L}(\vec{\pi}) = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\log\pi_k + \beta\left(\sum_{k=1}^{K}\pi_k - 1\right)$$

where β is the Lagrange multiplier. Taking derivatives, we find

$$\hat{\vec{\pi}}_k = \frac{\sum_{i=1}^{N}\hat{r}_{ik}}{N} \tag{11.18}$$

This is the same for any mixture model, whereas $\vec{\theta}_k$ depends on the form of $p(\vec{x}|\vec{\theta}_k)$.

For $\vec{\theta}_k$, plug in the pdf to Equation 11.16

$$\begin{aligned} Q(\vec{\theta}, \vec{\theta}^{t-1}) = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\log\pi_k - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\Big[\log|\vec{\Sigma}_k| + \\ (\vec{x}_i - \vec{\mu}_k)^T\vec{\Sigma}_k^{-1}(\vec{x}_i - \vec{\mu}_k)\Big] \end{aligned}$$

Take partial derivatives of Q wrt $\vec{\mu}_k$, $\vec{\Sigma}_k$ and let them equal to 0, we can get

$$\begin{aligned} \frac{\partial Q}{\partial\vec{\mu}_k} &= -\frac{1}{2}\sum_{i=1}^{N} r_{ik}\left[(\vec{\Sigma}_k^{-1} + \vec{\Sigma}_k^{-T})(\vec{x}_i - \vec{\mu}_k)\right] \\ &= -\sum_{i=1}^{N} r_{ik}\left[\vec{\Sigma}_k^{-1}(\vec{x}_i - \vec{\mu}_k)\right] = 0 \Rightarrow \\ \hat{\vec{\mu}}_k &= \frac{\sum_{i=1}^{N}\hat{r}_{ik}\vec{x}_i}{\sum_{i=1}^{N}\hat{r}_{ik}} \end{aligned} \tag{11.19}$$

$$\begin{aligned} \frac{\partial Q}{\partial\vec{\Sigma}_k} &= -\frac{1}{2}\sum_{i=1}^{N} r_{ik}\left[\frac{1}{\vec{\Sigma}_k} - \frac{1}{\vec{\Sigma}_k^2}(\vec{x}_i - \vec{\mu}_k)(\vec{x}_i - \vec{\mu}_k)^T\right] = 0 \Rightarrow \\ \hat{\vec{\Sigma}}_k &= \frac{\sum_{i=1}^{N}\hat{r}_{ik}(\vec{x}_i - \vec{\mu}_k)(\vec{x}_i - \vec{\mu}_k)^T}{\sum_{i=1}^{N}\hat{r}_{ik}} \qquad (11.20) \\ &= \frac{\sum_{i=1}^{N}\hat{r}_{ik}\vec{x}_i\vec{x}_i^T}{\sum_{i=1}^{N}\hat{r}_{ik}} - \vec{\mu}_k\vec{\mu}_k^T \qquad (11.21) \end{aligned}$$

11.4.3.4 Algorithm pseudo code

11.4.3.5 MAP estimation

As usual, the MLE may overfit. The overfitting problem is particularly severe in the case of GMMs. An easy solution to this is to perform MAP estimation. The new auxiliary function is the expected complete data log-likelihood plus the log prior:

input : observed data $\mathcal{D} = \{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N\}$,GMM
output: GMM's parameters $\vec{\pi}, \vec{\mu}, \vec{\Sigma}$
// 1. initialize
$\vec{\pi}^{(0)}$ = ...
$\vec{\mu}^{(0)}$ = ...
$\vec{\Sigma}^{(0)}$ = ...
t = 0
while *(!convergency)* **do**
  // 2. E-step
  $\hat{r}_{ik} = \frac{\pi_k p(\vec{x}_i|\vec{\theta}_k^{t-1})}{\sum_{k'=1}^{K} \pi_{k'} p(\vec{x}_i|\vec{\mu}_k^{t-1}, \vec{\Sigma}_k^{t-1})}$
  // 3. M-step
  $\hat{\vec{\pi}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik}}{N}$
  $\hat{\vec{\mu}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik} \vec{x}_i}{\sum_{i=1}^{N} \hat{r}_{ik}}$
  $\hat{\vec{\Sigma}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik} \vec{x}_i \vec{x}_i^T}{\sum_{i=1}^{N} \hat{r}_{ik}} - \vec{\mu}_k \vec{\mu}_k^T$
  ++t
end

Algorithm 4: EM algorithm for GMM

$$Q(\vec{\theta}, \vec{\theta}^{t-1}) = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \log \pi_k + \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \log p(\vec{x}_i|\vec{\theta}_k) + \log p(\vec{\pi}) + \sum_{k=1}^{K} \log p(\vec{\theta}_k) \tag{11.22}$$

It is natural to use conjugate priors.

$$p(\vec{\pi}) = \mathrm{Dir}(\vec{\pi}|\vec{\alpha})$$
$$p(\vec{\mu}_k, \vec{\Sigma}_k) = \mathrm{NIW}(\vec{\mu}_k, \vec{\Sigma}_k|\vec{m}_0, \kappa_0, \nu_0, \vec{S}_0)$$

From Equation 3.28 and Section 4.6.3, the MAP estimate is given by

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N} r_{ik} + \alpha_k - 1}{N + \sum_{k=1}^{K} \alpha_k - K} \tag{11.23}$$

$$\hat{\vec{\mu}}_k = \frac{\sum_{i=1}^{N} r_{ik} \vec{x}_i + \kappa_0 \vec{m}_0}{\sum_{i=1}^{N} r_{ik} + \kappa_0} \tag{11.24}$$

$$\hat{\vec{\Sigma}}_k = \frac{\vec{S}_0 + \vec{S}_k + \frac{\kappa_0 r_k}{\kappa_0 + r_k} (\bar{\vec{x}}_k - \vec{m}_0)(\bar{\vec{x}}_k - \vec{m}_0)^T}{\nu_0 + r_k + D + 2} \tag{11.25}$$

where $r_k \triangleq \sum_{i=1}^{N} r_{ik}, \bar{\vec{x}}_k \triangleq \frac{\sum_{i=1}^{N} r_{ik} \vec{x}_i}{r_k}$,

$$\vec{S}_k \triangleq \sum_{i=1}^{N} r_{ik} (\vec{x}_i - \bar{\vec{x}}_k)(\vec{x}_i - \bar{\vec{x}}_k)^T$$

11.4.4 EM for K-means

11.4.4.1 Representation

$$y_j = k \text{ if } \|\vec{x}_j - \vec{\mu}_k\|_2^2 \text{ is minimal} \tag{11.26}$$

where $\vec{\mu}_k$ is the centroid of cluster k.

11.4.4.2 Evaluation

$$\arg\min_{\vec{\mu}} \sum_{j=1}^{N} \sum_{k=1}^{K} \gamma_{jk} \|\vec{x}_j - \vec{\mu}_k\|_2^2 \tag{11.27}$$

The hidden variable is γ_{jk}, which's meanining is:

$$\gamma_{jk} = \begin{cases} 1, & \text{if } \|\vec{x}_j - \vec{\mu}_k\|_2 \text{ is minimal for } \vec{\mu}_k \\ 0, & \text{otherwise} \end{cases}$$

11.4.4.3 Optimization

E-Step:

$$\gamma_{jk}^{(i+1)} = \begin{cases} 1, & \text{if } \|\vec{x}_j - \vec{\mu}_k^{(i)}\|_2 \text{ is minimal for } \vec{\mu}_k^{(i)} \\ 0, & \text{otherwise} \end{cases} \tag{11.28}$$

M-Step:

$$\vec{\mu}_k^{(i+1)} = \frac{\sum_{j=1}^{N} \gamma_{jk}^{(i+1)} \vec{x}_j}{\sum \gamma_{jk}^{(i+1)}} \tag{11.29}$$

11.4.4.4 Tricks

Choosing k

TODO

Choosing the initial centroids(seeds)

1. **K-means++**.

   The intuition that spreading out the k initial cluster centers is a good thing is behind this approach: the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center[21].

   The exact algorithm is as follows:

   a. Choose one center uniformly at random from among the data points.

   b. For each data point $\vec{x}$, compute $D(\vec{x})$, the distance between $\vec{x}$ and the nearest center that has already been chosen.

   c. Choose one new data point at random as a new center, using a weighted probability distribution where a point $\vec{x}$ is chosen with probability proportional to $D(\vec{x})^2$.

   d. Repeat Steps 2 and 3 until k centers have been chosen.

   e. Now that the initial centers have been chosen, proceed using standard k-means clustering.

2. TODO

[21] http://en.wikipedia.org/wiki/K-means++

11.4.5 EM for mixture of experts

11.4.6 EM for DGMs with hidden variables

11.4.7 EM for the Student distribution *

11.4.8 EM for probit regression *

11.4.9 Derivation of the Q function

Theorem 11.1. ***(Jensen's inequality)*** *Let f be a convex function(see Section A.1) defined on a convex set $\mathcal{S}$. If $\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n \in \mathcal{S}$ and $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ with $\sum_{i=1}^{n} \lambda_i = 1$,*

$$f\left(\sum_{i=1}^{n} \lambda_i \vec{x}_i\right) \leq \sum_{i=1}^{n} \lambda_i f(\vec{x}_i) \tag{11.30}$$

Proposition 11.1.

$$\log\left(\sum_{i=1}^{n} \lambda_i \vec{x}_i\right) \geq \sum_{i=1}^{n} \lambda_i \log(\vec{x}_i) \tag{11.31}$$

Now let's proof why the Q function should look like Equation 11.13:

$$\begin{aligned}
\ell(\vec{\theta}) &= \log P(\mathcal{D}|\vec{\theta}) \\
&= \log \sum_{\vec{z}} P(\mathcal{D}, \vec{z}|\vec{\theta}) \\
&= \log \sum_{\vec{z}} P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta}) \\
\ell(\vec{\theta}) - \ell(\vec{\theta}^{t-1}) &= \log \left[\sum_{\vec{z}} P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})\right] - \log P(\mathcal{D}|\vec{\theta}^{t-1}) \\
&= \log \left[\sum_{\vec{z}} P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta}) \frac{P(\vec{z}|\mathcal{D}, \theta^{t-1})}{P(\vec{z}|\mathcal{D}, \theta^{t-1})}\right] \\
&\quad - \log P(\mathcal{D}|\vec{\theta}^{t-1}) \\
&= \log \left[\sum_{\vec{z}} P(\vec{z}|\mathcal{D}, \theta^{t-1}) \frac{P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})}{P(\vec{z}|\mathcal{D}, \theta^{t-1})}\right] \\
&\quad - \log P(\mathcal{D}|\vec{\theta}^{t-1}) \\
&\geq \sum_{\vec{z}} P(\vec{z}|\mathcal{D}, \theta^{t-1}) \log \left[\frac{P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})}{P(\vec{z}|\mathcal{D}, \theta^{t-1})}\right] \\
&\quad - \log P(\mathcal{D}|\vec{\theta}^{t-1}) \\
&= \sum_{\vec{z}} \{P(\vec{z}|\mathcal{D}, \theta^{t-1}) \\
&\quad \log \left[\frac{P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})}{P(\vec{z}|\mathcal{D}, \theta^{t-1}) P(\mathcal{D}|\vec{\theta}^{t-1})}\right]\}
\end{aligned}$$

$$\begin{aligned}
B(\vec{\theta}, \vec{\theta}^{t-1}) &\triangleq \ell(\vec{\theta}^{t-1}) + \\
&\sum_{\vec{z}} P(\vec{z}|\mathcal{D}, \theta^{t-1}) \log \left[\frac{P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})}{P(\vec{z}|\mathcal{D}, \theta^{t-1}) P(\mathcal{D}|\vec{\theta}^{t-1})}\right] \\
&\Rightarrow
\end{aligned}$$

$$\begin{aligned}
\vec{\theta}^t &= \arg\max_{\vec{\theta}} B(\vec{\theta}, \vec{\theta}^{t-1}) \\
&= \arg\max_{\vec{\theta}} \left\{\ell(\vec{\theta}^{t-1}) + \right. \\
&\quad \left. \sum_{\vec{z}} P(\vec{z}|\mathcal{D}, \theta^{t-1}) \log \left[\frac{P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})}{P(\vec{z}|\mathcal{D}, \theta^{t-1}) P(\mathcal{D}|\vec{\theta}^{t-1})}\right]\right\}
\end{aligned}$$

Now drop terms which are constant w.r.t. $\vec{\theta}$

$$\begin{aligned}
&= \arg\max_{\vec{\theta}} \left\{\sum_{\vec{z}} P(\vec{z}|\mathcal{D}, \theta^{t-1}) \log \left[P(\mathcal{D}|\vec{z}, \vec{\theta}) P(\vec{z}|\vec{\theta})\right]\right\} \\
&= \arg\max_{\vec{\theta}} \left\{\sum_{\vec{z}} P(\vec{z}|\mathcal{D}, \theta^{t-1}) \log \left[P(\mathcal{D}, \vec{z}|\vec{\theta})\right]\right\} \\
&= \arg\max_{\vec{\theta}} \left\{\mathbb{E}_{\vec{z}|\mathcal{D}, \theta^{t-1}} \log \left[P(\mathcal{D}, \vec{z}|\vec{\theta})\right]\right\} \qquad (11.32) \\
&\triangleq \arg\max_{\vec{\theta}} Q(\vec{\theta}, \vec{\theta}^{t-1}) \qquad (11.33)
\end{aligned}$$

11.4.10 Convergence of the EM Algorithm *

11.4.10.1 Expected complete data log likelihood is a lower bound

Note that $\ell(\vec{\theta}) \geq B(\vec{\theta}, \vec{\theta}^{t-1})$, and $\ell(\vec{\theta}^{t-1}) \geq B(\vec{\theta}^{t-1}, \vec{\theta}^{t-1})$, which means $B(\vec{\theta}, \vec{\theta}^{t-1})$ is an lower bound of $\ell(\vec{\theta})$. If we maximize $B(\vec{\theta}, \vec{\theta}^{t-1})$, then $\ell(\vec{\theta})$ gets maximized, see Figure 11.4.

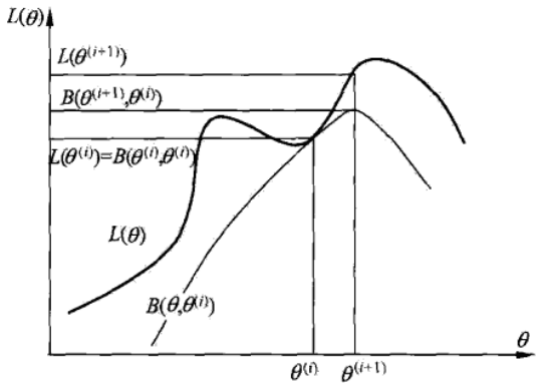


Fig. 11.4: Graphical interpretation of a single iteration of the EM algorithm: The function $B(\vec{\theta}, \vec{\theta}^{t-1})$ is bounded above by the log likelihood function $\ell(\vec{\theta})$. The functions are equal at $\vec{\theta} = \vec{\theta}^{t-1}$. The EM algorithm chooses $\vec{\theta}^{t-1}$ as the value of $\vec{\theta}$ for which $B(\vec{\theta}, \vec{\theta}^{t-1})$ is a maximum. Since $\ell(\vec{\theta}) \geq B(\vec{\theta}, \vec{\theta}^{t-1})$ increasing $B(\vec{\theta}, \vec{\theta}^{t-1})$ ensures that the value of the log likelihood function $\ell(\vec{\theta})$ is increased at each step.

Since the expected complete data log likelihood Q is derived from $B(\vec{\theta}, \vec{\theta}^{t-1})$ by dropping terms which are constant w.r.t. $\vec{\theta}$, so it is also a lower bound to a lower bound of $\ell(\vec{\theta})$.

11.4.10.2 EM monotonically increases the observed data log likelihood

11.4.11 Generalization of EM Algorithm *

EM algorithm can be interpreted as F function's maximization-maximization algorithm, based on this interpretation there are many variations and generalization, e.g., generalized EM Algorithm(GEM).

11.4.11.1 F function's maximization-maximization algorithm

Definition 11.1. Given the probability distribution of the hidden variable Z is $\tilde{P}(Z)$, define **F function** as the following:

$$F(\tilde{P}, \vec{\theta}) = \mathbb{E}_{\tilde{P}}[\log P(X,Z|\theta)] + H(\tilde{P}) \tag{11.34}$$

Where $H(\tilde{P}) = -\mathbb{E}_{\tilde{P}} \log \tilde{P}(Z)$, which is $\tilde{P}(Z)$'s entropy. Usually we assume that $P(X,Z|\theta)$ is continuous w.r.t. $\vec{\theta}$, therefore $F(\tilde{P}, \vec{\theta})$ is continuous w.r.t. $\tilde{P}$ and $\vec{\theta}$.

Lemma 11.1. *For a fixed $\vec{\theta}$, there is only one distribution $\tilde{P}_\theta$ which maximizes $F(\tilde{P}, \vec{\theta})$*

$$\tilde{P}_\theta(Z) = P(Z|X, \vec{\theta}) \tag{11.35}$$

and $\tilde{P}_\theta$ is continuous w.r.t. $\vec{\theta}$.

Proof. Given a fixed $\vec{\theta}$, we can get $\tilde{P}_\theta$ which maximizes $F(\tilde{P}, \vec{\theta})$. we construct the Lagrangian

$$\begin{aligned}\mathcal{L}(\tilde{P}, \vec{\theta}) = \mathbb{E}_{\tilde{P}}[\log P(X,Z|\theta)] - \mathbb{E}_{\tilde{P}} \log \tilde{P}_\theta(Z) \\ + \lambda \left[1 - \sum_Z \tilde{P}(Z)\right]\end{aligned} \tag{11.36}$$

Take partial derivative with respect to $\tilde{P}_\theta(Z)$ then we get

$$\frac{\partial \mathcal{L}}{\partial \tilde{P}_\theta(Z)} = \log P(X,Z|\theta) - \log \tilde{P}_\theta(Z) - 1 - \lambda$$

Let it equal to 0, we can get

$$\lambda = \log P(X,Z|\theta) - \log \tilde{P}_\theta(Z) - 1$$

Then we can derive that $\tilde{P}_\theta(Z)$ is proportional to $P(X,Z|\theta)$

$$\begin{aligned}\frac{P(X,Z|\theta)}{\tilde{P}_\theta(Z)} &= e^{1+\lambda} \\ \Rightarrow \tilde{P}_\theta(Z) &= \frac{P(X,Z|\vec{\theta})}{e^{1+\lambda}}\end{aligned}$$

$$\sum_Z \tilde{P}_\theta(Z) = 1 \Rightarrow \sum_Z \frac{P(X,Z|\vec{\theta})}{e^{1+\lambda}} = 1 \Rightarrow P(X|\vec{\theta}) = e^{1+\lambda}$$

$$\tilde{P}_\theta(Z) = \frac{P(X,Z|\vec{\theta})}{e^{1+\lambda}} = \frac{P(X,Z|\vec{\theta})}{P(X|\vec{\theta})} = P(Z|X,\vec{\theta})$$

Lemma 11.2. *If $\tilde{P}_\theta(Z) = P(Z|X,\vec{\theta})$, then*

$$F(\tilde{P}, \vec{\theta}) = \log P(X|\vec{\theta}) \tag{11.37}$$

Theorem 11.2. *One iteration of EM algorithm can be implemented as F function's maximization-maximization.*

Assume $\vec{\theta}^{t-1}$ is the estimation of $\vec{\theta}$ in the $(t-1)$-th iteration, $\tilde{P}^{t-1}$ is the estimation of $\tilde{P}$ in the $(t-1)$-th iteration. Then in the t-th iteration two steps are:

1. for fixed $\vec{\theta}^{t-1}$, find $\tilde{P}^t$ that maximizes $F(\tilde{P}, \vec{\theta}^{t-1})$;

2. for fixed $\tilde{P}^t$, find $\vec{\theta}^t$ that maximizes $F(\tilde{P}^t, \vec{\theta})$.

Proof. (1) According to Lemma 11.1, we can get

$$\tilde{P}^t(Z) = P(Z|X, \vec{\theta}^{t-1})$$

(2) According above, we can get

$$\begin{aligned}F(\tilde{P}^t, \vec{\theta}) &= \mathbb{E}_{\tilde{P}^t}[\log P(X,Z|\theta)] + H(\tilde{P}^t) \\ &= \sum_Z P(Z|X, \vec{\theta}^{t-1}) \log P(X,Z|\theta) + H(\tilde{P}^t) \\ &= Q(\vec{\theta}, \vec{\theta}^{t-1}) + H(\tilde{P}^t)\end{aligned}$$

Then

$$\vec{\theta}^t = \arg\max_\theta F(\tilde{P}^t, \vec{\theta}) = \arg\max_\theta Q(\vec{\theta}, \vec{\theta}^{t-1})$$

11.4.11.2 The Generalized EM Algorithm(GEM)

In the formulation of the EM algorithm described above, $\vec{\theta}^t$ was chosen as the value of $\vec{\theta}$ for which $Q(\vec{\theta}, \vec{\theta}^{t-1})$ was maximized. While this ensures the greatest increase in $\ell(\vec{\theta})$, it is however possible to relax the requirement of maximization to one of simply increasing $Q(\vec{\theta}, \vec{\theta}^{t-1})$ so that $Q(\vec{\theta}^t, \vec{\theta}^{t-1}) \geq Q(\vec{\theta}^{t-1}, \vec{\theta}^{t-1})$. This approach, to simply increase and not necessarily maximize $Q(\vec{\theta}^t, \vec{\theta}^{t-1})$ is known as the Generalized Expectation Maximization (GEM) algorithm and is often useful in cases where the maximization is difficult. The convergence of the GEM algorithm is similar to the EM algorithm.

11.4.12 Online EM

11.4.13 Other EM variants *

11.5 Model selection for latent variable models

When using LVMs, we must specify the number of latent variables, which controls the model complexity. In particular, in the case of mixture models, we must specify K, the number of clusters. Choosing these parameters is an example of model selection. We discuss some approaches below.

11.5.1 Model selection for probabilistic models

The optimal Bayesian approach, discussed in Section 5.3, is to pick the model with the largest marginal likelihood, $K^* = \arg\max_k p(\mathcal{D}|k)$.

There are two problems with this. First, evaluating the marginal likelihood for LVMs is quite difficult. In practice, simple approximations, such as BIC, can be used (see e.g., (Fraley and Raftery 2002)). Alternatively, we can use the cross-validated likelihood as a performance measure, although this can be slow, since it requires fitting each model F times, where Fis the number of CV folds.

The second issue is the need to search over a potentially large number of models. The usual approach is to perform exhaustive search over all candidate values ofK. However, sometimes we can set the model to its maximal size, and then rely on the power of the Bayesian Occam's razor to "kill off" unwanted components. An example of this will be shown in Section TODO 21.6.1.6, when we discuss variational Bayes.

An alternative approach is to perform stochastic sampling in the space of models. Traditional approaches, such as (Green 1998, 2003; Lunn et al. 2009), are based on reversible jump MCMC, and use birth moves to propose new centers, and death moves to kill off old centers. However, this can be slow and difficult to implement. A simpler approach is to use a Dirichlet process mixture model, which can be fit using Gibbs sampling, but still allows for an unbounded number of mixture components; see Section TODO 25.2 for details.

Perhaps surprisingly, these sampling-based methods can be faster than the simple approach of evaluating the quality of eachKseparately. The reason is that fitting the model for each K is often slow. By contrast, the sampling methods can often quickly determine that a certain value of K is poor, and thus they need not waste time in that part of the posterior.

11.5.2 Model selection for non-probabilistic methods

What if we are not using a probabilistic model? For example, how do we choose K for the K-means algorithm? Since this does not correspond to a probability model, there is no likelihood, so none of the methods described above can be used.

An obvious proxy for the likelihood is the **reconstruction error**. Define the squared reconstruction error of a data set $\mathcal{D}$, using model complexity K, as follows:

$$E(\mathcal{D}, K) \triangleq \frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} \|\vec{x}_i - \hat{\vec{x}}_i\|^2 \tag{11.38}$$

In the case of K-means, the reconstruction is given by $\hat{\vec{x}}_i = \vec{\mu}_{z_i}$, where $z_i = \arg\min_k \|\vec{x}_i - \hat{\vec{\mu}}_k\|^2$, as explained in Section 11.4.2.6 TODO.

In supervised learning, we can always use cross validation to select between non-probabilistic models of different complexity, but this is not the case with unsupervised learning. The most common approach is to plot the reconstruction error on the training set versus K, and to try to identify a **knee** or **kink** in the curve.

11.6 Fitting models with missing data

Suppose we want to fit a joint density model by maximum likelihood, but we have "holes" in our data matrix, due to missing data (usually represented by NaNs). More formally, let $O_{ij} = 1$ if component j of data case i is observed, and let $O_{ij} = 0$ otherwise. Let $\vec{X}_v = \{x_{ij} : \vec{O}_{ij} = 1\}$ be the visible data, and $X_h = \{x_{ij} : \vec{O}_{ij} = 0\}$ be the missing or hidden data. Our goal is to compute

$$\hat{\vec{\theta}} = \arg\max_{\vec{\theta}} p(\vec{X}_v|\vec{\theta}, \vec{O}) \tag{11.39}$$

Under the missing at random assumption (see Section 8.6.2), we have

TODO

11.6.1 EM for the MLE of an MVN with missing data

TODO

Chapter 12
Latent linear models

12.1 Factor analysis

One problem with mixture models is that they only use a single latent variable to generate the observations. In particular, each observation can only come from one of K prototypes. One can think of a mixture model as using K hidden binary variables, representing a one-hot encoding of the cluster identity. But because these variables are mutually exclusive, the model is still limited in its representational power.

An alternative is to use a vector of real-valued latent variables,$\vec{z}_i \in \mathbb{R}^L$. The simplest prior to use is a Gaussian (we will consider other choices later):

$$p(\vec{z}_i) = \mathcal{N}(\vec{z}_i|\vec{\mu}_0, \vec{\Sigma}_0) \tag{12.1}$$

If the observations are also continuous, so $\vec{x}_i \in \mathbb{R}^D$, we may use a Gaussian for the likelihood. Just as in linear regression, we will assume the mean is a linear function of the (hidden) inputs, thus yielding

$$p(\vec{x}_i|\vec{z}_i, \vec{\theta}) = \mathcal{N}(\vec{x}_i|\vec{W}\vec{z}_i + \vec{\mu}, \vec{\Psi}) \tag{12.2}$$

where $\vec{W}$ is a $D \times L$ matrix, known as the **factor loading matrix**, and $\vec{\Psi}$ is a $D \times D$ covariance matrix. We take $\vec{\Psi}$ to be diagonal, since the whole point of the model is to "force" $\vec{z}_i$ to explain the correlation, rather than "baking it in" to the observation's covariance. This overall model is called **factor analysis** or **FA**. The special case in which $\vec{\Psi} = \sigma^2\vec{I}$ is called **probabilistic principal components analysis** or **PPCA**. The reason for this name will become apparent later.

12.1.1 FA is a low rank parameterization of an MVN

FA can be thought of as a way of specifying a joint density model on $\vec{x}$ using a small number of parameters. To see this, note that from Equation 4.39, the induced marginal distribution $p(\vec{x}_i|\vec{\theta})$ is a Gaussian:

$$\begin{aligned} p(\vec{x}_i|\vec{\theta}) &= \int \mathcal{N}(\vec{x}_i|\vec{W}\vec{z}_i + \vec{\mu}, \vec{\Psi})\mathcal{N}(\vec{z}_i|\vec{\mu}_0, \vec{\Sigma}_0)\mathrm{d}\vec{z}_i \\ &= \mathcal{N}(\vec{x}_i|\vec{W}\vec{\mu}_0 + \vec{\mu}, \vec{\Psi} + \vec{W}\vec{\Sigma}_0\vec{W}) \end{aligned} \tag{12.3}$$

From this, we see that we can set $\vec{\mu}_0 = 0$ without loss of generality, since we can always absorb $\vec{W}\vec{\mu}_0$ into $\vec{\mu}$. Similarly, we can set $\vec{\Sigma}_0 = \vec{I}$ without loss of generality, because we can always "emulate" a correlated prior by using defining a new weight matrix, $\tilde{\vec{W}} = \vec{W}\vec{\Sigma}_0^{\frac{1}{2}}$. So we can rewrite Equation 12.6 and 12.2 as:

$$p(\vec{z}_i) = \mathcal{N}(\vec{z}_i|\vec{0}, \vec{I}) \tag{12.4}$$

$$p(\vec{x}_i|\vec{z}_i, \vec{\theta}) = \mathcal{N}(\vec{x}_i|\vec{W}\vec{z}_i + \vec{\mu}, \vec{\Psi}) \tag{12.5}$$

We thus see that FA approximates the covariance matrix of the visible vector using a low-rank decomposition:

$$\vec{C} \triangleq \mathrm{cov}[\vec{x}] = \vec{W}\vec{W}^T + \vec{\Psi} \tag{12.6}$$

This only uses $O(LD)$ parameters, which allows a flexible compromise between a full covariance Gaussian, with $O(D^2)$ parameters, and a diagonal covariance, with $O(D)$ parameters. Note that if we did not restrict $\vec{\Psi}$ to be diagonal, we could trivially set $\vec{\Psi}$ to a full covariance matrix; then we could set $\vec{W} = 0$, in which case the latent factors would not be required.

12.1.2 Inference of the latent factors

$$p(\vec{z}_i|\vec{x}_i, \vec{\theta}) = \mathcal{N}(\vec{z}_i|\vec{\mu}_i, \vec{\Sigma}_i) \tag{12.7}$$

$$\vec{\Sigma}_i \triangleq (\vec{\Sigma}_0^{-1} + \vec{W}^T\vec{\Psi}^{-1}\vec{W})^{-1} \tag{12.8}$$

$$= (\vec{I} + \vec{W}^T\vec{\Psi}^{-1}\vec{W})^{-1} \tag{12.9}$$

$$\vec{\mu}_i \triangleq \vec{\Sigma}_i[\vec{W}^T\vec{\Psi}^{-1}(\vec{x}_i - \vec{\mu}) + \vec{\Sigma}_0^{-1}\vec{\mu}_0] \tag{12.10}$$

$$= \vec{\Sigma}_i\vec{W}^T\vec{\Psi}^{-1}(\vec{x}_i - \vec{\mu}) \tag{12.11}$$

Note that in the FA model, $\vec{\Sigma}_i$ is actually independent of i, so we can denote it by $\vec{\Sigma}$. Computing this matrix takes $O(L^3 + L^2D)$ time, and computing each $\vec{\mu}_i = \mathbb{E}[\vec{z}_i|\vec{x}_i, \vec{\theta}]$ takes $O(L^2 + LD)$ time. The $\vec{\mu}_i$ are sometimes called the **latent scores**, or **latent factors**.

12.1.3 Unidentifiability

Just like with mixture models, FA is also unidentifiable. To see this, suppose $\vec{R}$ is an arbitrary orthogonal rotation matrix, satisfying $\vec{R}\vec{R}^T = \vec{I}$. Let us define $\tilde{\vec{W}} = \vec{W}\vec{R}$, then the likelihood function of this modified matrix is the same as for the unmodified matrix, since $\vec{W}\vec{R}\vec{R}^T\vec{W}^T + \vec{\Psi} = \vec{W}\vec{W}^T + \vec{\Psi}$. Geometrically, multiplying $\vec{W}$ by an orthogonal matrix is like rotating $\vec{z}$ before generating $\vec{x}$.

To ensure a unique solution, we need to remove $L(L-1)/2$ degrees of freedom, since that is the number of orthonormal matrices of size $L \times L$.[22] In total, the FA model has

[22] To see this, note that there are $L-1$ free parameters in $\vec{R}$ in the first column (since the column vector must be normalized to unit length),

$D+LD-L(L-1)/2$ free parameters (excluding the mean), where the first term arises from $\vec{\Psi}$. Obviously we require this to be less than or equal to $D(D+1)/2$, which is the number of parameters in an unconstrained (but symmetric) covariance matrix. This gives us an upper bound on L, as follows:

$$L_{\max} = \lfloor D + 0.5(1 - \sqrt{1+8D}) \rfloor \tag{12.12}$$

For example, $D = 6$ implies $L \leq 3$. But we usually never choose this upper bound, since it would result in overfitting (see discussion in Section 12.3 on how to choose L).

Unfortunately, even if we set $L < L_{\max}$, we still cannot uniquely identify the parameters, since the rotational ambiguity still exists. Non-identifiability does not affect the predictive performance of the model. However, it does affect the loading matrix, and hence the interpretation of the latent factors. Since factor analysis is often used to uncover structure in the data, this problem needs to be addressed. Here are some commonly used solutions:

- **Forcing $\vec{W}$ to be orthonormal** Perhaps the cleanest solution to the identifiability problem is to force $\vec{W}$ to be orthonormal, and to order the columns by decreasing variance of the corresponding latent factors. This is the approach adopted by PCA, which we will discuss in Section 12.2. The result is not necessarily more interpretable, but at least it is unique.
- **Forcing $\vec{W}$ to be lower triangular** One way to achieve identifiability, which is popular in the Bayesian community (e.g., (Lopes and West 2004)), is to ensure that the first visible feature is only generated by the first latent factor, the second visible feature is only generated by the first two latent factors, and so on. For example, if $L = 3$ and $D = 4$, the correspond factor loading matrix is given by

$$\vec{W} = \begin{pmatrix} w_{11} & 0 & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{32} & w_{43} \end{pmatrix}$$

  We also require that $w_{jj} > 0$ for $j = 1 : L$. The total number of parameters in this constrained matrix is $D + DL - L(L-1)/2$, which is equal to the number of uniquely identifiable parameters. The disadvantage of this method is that the first L visible variables, known as the **founder variables**, affect the interpretation of the latent factors, and so must be chosen carefully.
- **Sparsity promoting priors on the weights** Instead of pre-specifying which entries in $\vec{W}$ are zero, we can encourage the entries to be zero, using ℓ_1 regularization (Zou et al. 2006), ARD (Bishop 1999; Archambeau and Bach 2008), or spike-and-slab priors (Rattray et al. 2009). This is called sparse factor analysis. This does not necessarily ensure a unique MAP estimate, but it does encourage interpretable solutions. See Section 13.8 TODO.
- **Choosing an informative rotation matrix** There are a variety of heuristic methods that try to find rotation matrices $\vec{R}$ which can be used to modify $\vec{W}$(and hence the latent factors) so as to try to increase the interpretability, typically by encouraging them to be (approximately) sparse. One popular method is known as **varimax**(Kaiser 1958).
- **Use of non-Gaussian priors for the latent factors** In Section 12.6, we will dicuss how replacing $p(\vec{z}_i)$ with a non-Gaussian distribution can enable us to sometimes uniquely identify $\vec{W}$ as well as the latent factors. This technique is known as ICA.

there are $L-2$ free parameters in the second column (which must be orthogonal to the first), and so on.

12.1.4 Mixtures of factor analysers

The FA model assumes that the data lives on a low dimensional linear manifold. In reality, most data is better modeled by some form of low dimensional *curved* manifold. We can approximate a curved manifold by a piecewise linear manifold. This suggests the following model: let the k'th linear subspace of dimensionality L_k be represented by $\vec{W}_k$, for $k = 1 : K$. Suppose we have a latent indicator $qi \in \{1, \cdots, K\}$ specifying which subspace we should use to generate the data. We then sample $\vec{z}_i$ from a Gaussian prior and pass it through the $\vec{W}_k$ matrix (where $k = q_i$), and add noise. More precisely, the model is as follows:

$$p(q_i|\vec{\theta}) = \text{Cat}(q_i\vec{\pi}) \tag{12.13}$$

$$p(\vec{z}_i|\vec{\theta}) = \mathcal{N}(\vec{z}_i|\vec{0},\vec{I}) \tag{12.14}$$

$$p(\vec{x}_i|q_i = k, \vec{z}_i, \vec{\theta}) = \mathcal{N}(\vec{x}_i|\vec{W}\vec{z}_i + \vec{\mu}_k, \vec{\Psi}) \tag{12.15}$$

This is called a **mixture of factor analysers**(MFA) (Hinton et al. 1997).

Another way to think about this model is as a low-rank version of a mixture of Gaussians. In particular, this model needs $O(KLD)$ parameters instead of the $O(KD^2)$ parameters needed for a mixture of full covariance Gaussians. This can reduce overfitting. In fact, MFA is a good generic density model for high-dimensional real-valued data.

12.1.5 EM for factor analysis models

Below we state the results without proof. The derivation can be found in (Ghahramani and Hinton 1996a). To obtain the results for a single factor analyser, just set $r_{ic} = 1$ and $c = 1$ in the equations below. In Section 12.2.4 we will see a further simplification of these equations that arises when fitting a PPCA model, where the results will turn out to have a particularly simple and elegant interpretation.

In the E-step, we compute the posterior responsibility of cluster k for data point i using

$$r_{ik} \triangleq p(q_i = k|\vec{x}_i, \vec{\theta}) \propto \pi_k \mathcal{N}(\vec{x}_i|\vec{\mu}_k, \vec{W}_k\vec{W}_k^T\vec{\Psi}_k) \tag{12.16}$$

The conditional posterior for $\vec{z}_i$ is given by

$$p(\vec{z}_i|\vec{x}_i, q_i = k, \vec{\theta}) = \mathcal{N}(\vec{z}_i|\vec{\mu}_{ik}, \vec{\Sigma}_{ik}) \tag{12.17}$$

$$\vec{\Sigma}_{ik} \triangleq (\vec{I} + \vec{W}_k^T \vec{\Psi}_k^{-1} \vec{W})_k^{-1} \tag{12.18}$$

$$\vec{\mu}_{ik} \triangleq \vec{\Sigma}_{ik} \vec{W}_k^T \vec{\Psi}_k^{-1} (\vec{x}_i - \vec{\mu}_k) \tag{12.19}$$

In the M step, it is easiest to estimate $\vec{\mu}_k$ and $\vec{W}_k$ at the same time, by defining $\tilde{\vec{W}}_k = (\vec{W}_k, \vec{\mu}_k)$, $\tilde{\vec{z}} = (\vec{z}, 1)$, also, define

$$\tilde{\vec{W}}_k = (\vec{W}_k, \vec{\mu}_k) \tag{12.20}$$

$$\tilde{\vec{z}} = (\vec{z}, 1) \tag{12.21}$$

$$\vec{b}_{ik} \triangleq \mathbb{E}[\tilde{\vec{z}}|\vec{x}_i, q_i = k] = \mathbb{E}[(\vec{\mu}_{ik}; 1)] \tag{12.22}$$

$$\vec{C}_{ik} \triangleq \mathbb{E}[\tilde{\vec{z}}\tilde{\vec{z}}^T|\vec{x}_i, q_i = k] \tag{12.23}$$

$$= \begin{pmatrix} \mathbb{E}[\vec{z}\vec{z}^T|\vec{x}_i, q_i = k] & \mathbb{E}[\vec{z}|\vec{x}_i, q_i = k] \\ \mathbb{E}[\vec{z}|\vec{x}_i, q_i = k]^T & 1 \end{pmatrix} \tag{12.24}$$

Then the M step is as follows:

$$\hat{\pi}_k = \frac{1}{N} \sum_{i=1}^N r_{ik} \tag{12.25}$$

$$\hat{\tilde{\vec{W}}}_k = \left(\sum_{i=1}^N r_{ik} \vec{x}_i \vec{b}_{ik}^T \right) \left(\sum_{i=1}^N r_{ik} \vec{C}_{ik}^T \right)^{-1} \tag{12.26}$$

$$\hat{\vec{\Psi}} = \frac{1}{N} \text{diag} \left[\sum_{i=1}^N r_{ik} (\vec{x}_i - \hat{\tilde{\vec{W}}}_{ik} \vec{b}_{ik}) \vec{x}_i^T \right] \tag{12.27}$$

Note that these updates are for "vanilla" EM. A much faster version of this algorithm, based on ECM, is described in (Zhao and Yu 2008).

12.1.6 Fitting FA models with missing data

In many applications, such as collaborative filtering, we have missing data. One virtue of the EM approach to fitting an FA/PPCA model is that it is easy to extend to this case. However, overfitting can be a problem if there is a lot of missing data. Consequently it is important to perform MAP estimation or to use Bayesian inference. See e.g., (Ilin and Raiko 2010) for details.

12.2 Principal components analysis (PCA)

Consider the FA model where we constrain $\vec{\Psi} = \sigma^2 \vec{I}$, and $\vec{W}$ to be orthonormal. It can be shown (Tipping and Bishop 1999) that, as $\sigma^2 \to 0$, this model reduces to classical (nonprobabilistic) **principal components analysis**(PCA), also known as the Karhunen Loeve transform. The version where $\sigma^2 > 0$ is known as **probabilistic PCA**(**PPCA**) (Tipping and Bishop 1999), orsensible PCA(Roweis 1997).

12.2.1 Classical PCA

12.2.1.1 Statement of the theorem

The synthesis viewof classical PCA is summarized in the forllowing theorem.

Theorem 12.1. *Suppose we want to find an orthogonal set of L linear basis vectors $\vec{w}_j \in \mathbb{R}^D$, and the corresponding scores $\vec{z}_i \in \mathbb{R}^L$, such that we minimize the average* ***reconstruction error***

$$J(\vec{W}, \vec{Z}) = \frac{1}{N} \sum_{i=1}^N \|\vec{x}_i - \hat{\vec{x}}_i\|^2 \tag{12.28}$$

where $\hat{\vec{x}}_i = \vec{W}\vec{z}_i$, subject to the constraint that $\vec{W}$ is orthonormal. Equivalently, we can write this objective as follows

$$J(\vec{W}, \vec{Z}) = \frac{1}{N} \|\vec{X} - \vec{W}\vec{Z}^T\|^2 \tag{12.29}$$

where $\vec{Z}$ is an $N \times L$ matrix with the $\vec{z}_i$ in its rows, and $\|A\|_F$ is the ***Frobenius norm*** *of matrix $\vec{A}$, defined by*

$$\|A\|_F \triangleq \sqrt{\sum_{i=1}^M \sum_{j=1}^N a_{ij}^2} = \sqrt{\text{tr}(\vec{A}^T \vec{A})} \tag{12.30}$$

The optimal solution is obtained by setting $\hat{\vec{W}} = \vec{V}_L$, where $\vec{V}_L$ contains the L eigenvectors with largest eigenvalues of the empirical covariance matrix, $\hat{\vec{\Sigma}} = \frac{1}{N} \sum_{i=1}^N \vec{x}_i \vec{x}_i^T$. (We assume the $\vec{x}_i$ have zero mean, for notational simplicity.) Furthermore, the optimal low-dimensional encoding of the data is given by $\hat{\vec{z}}_i = \vec{W}^T \vec{x}_i$, which is an orthogonal projection of the data onto the column space spanned by the eigenvectors.

An example of this is shown in Figure 12.1(a) for $D = 2$ and $L = 1$. The diagonal line is the vector $\vec{w}_1$; this is called the first principal component or principal direction. The data points $\vec{x}_i \in \mathbb{R}^2$ are orthogonally projected onto this line to get $\vec{z}_i \in \mathbb{R}$. This is the best 1-dimensional approximation to the data. (We will discuss Figure 12.1(b) later.)

The principal directions are the ones along which the data shows maximal variance. This means that PCA can be "misled" by directions in which the variance is high merely because of the measurement scale. It is therefore standard practice to standardize the data first, or equivalently, to work with correlation matrices instead of covariance matrices.

12.2.1.2 Proof *

See Section 12.2.2 of MLAPP.

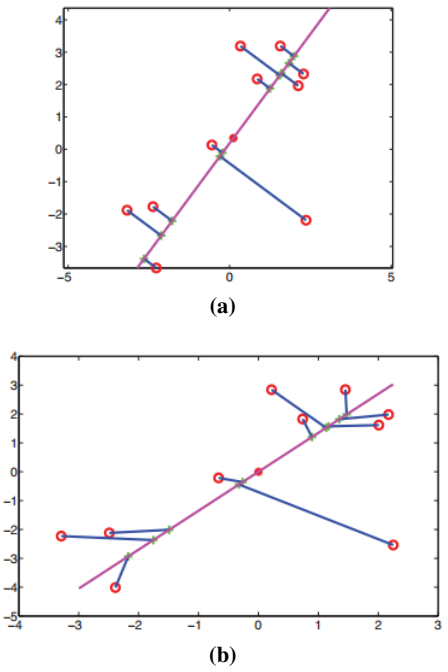


Fig. 12.1: An illustration of PCA and PPCA where $D = 2$ and $L = 1$. Circles are the original data points, crosses are the reconstructions. The red star is the data mean. (a) PCA. The points are orthogonally projected onto the line. (b) PPCA. The projection is no longer orthogonal: the reconstructions are shrunk towards the data mean (red star).

12.2.2 Singular value decomposition (SVD)

We have defined the solution to PCA in terms of eigenvectors of the covariance matrix. However, there is another way to obtain the solution, based on the **singular value decomposition**, or **SVD**. This basically generalizes the notion of eigenvectors from square matrices to any kind of matrix.

Theorem 12.2. *(SVD). Any matrix can be decomposed as follows*

$$\underbrace{\vec{X}}_{N\times D} = \underbrace{\vec{U}}_{N\times N}\underbrace{\vec{\Sigma}}_{N\times D}\underbrace{\vec{V}^T}_{D\times D} \tag{12.31}$$

where $\vec{U}$ is an $N \times N$ matrix whose columns are orthornormal(so $\vec{U}^T\vec{U} = \vec{I}$), $\vec{V}$ is $D \times D$ matrix whose rows and columns are orthonormal (so $\vec{V}^T\vec{V} = \vec{V}\vec{V}^T = \vec{I}_D$), and $\vec{\Sigma}$ is a $N \times D$ matrix containing the $r = \min(N, D)$ singular values $\sigma_i \geq 0$ on the main diagonal, with 0s filling the rest of the matrix.

This shows how to decompose the matrix X into the product of three matrices: $\vec{V}$ describes an orthonormal basis in the domain, and $\vec{U}$ describes an orthonormal basis in the co-domain, and $\vec{\Sigma}$ describes how much the vectors in $\vec{V}$ are stretched to give the vectors in $\vec{U}$.

Since there are at most D singular values (assuming $N > D$), the last $N - D$ columns of $\vec{U}$ are irrelevant, since they will be multiplied by 0. The **economy sized SVD**, or **thin SVD**, avoids computing these unnecessary elements. Let us denote this decomposition by $\hat{\vec{U}}\hat{\vec{\Sigma}}\hat{\vec{V}}^T$.If $N > D$, we have

$$\underbrace{\vec{X}}_{N\times D} = \underbrace{\hat{\vec{U}}}_{N\times D}\underbrace{\hat{\vec{\Sigma}}}_{D\times D}\underbrace{\hat{\vec{V}}^T}_{D\times D} \tag{12.32}$$

as in Figure 12.2(a). If $N < D$, we have

$$\underbrace{\vec{X}}_{N\times D} = \underbrace{\hat{\vec{U}}}_{N\times N}\underbrace{\hat{\vec{\Sigma}}}_{N\times N}\underbrace{\hat{\vec{V}}^T}_{N\times D} \tag{12.33}$$

Computing the economy-sized SVD takes $O(ND\min(N, D))$ time (Golub and van Loan 1996, p254).

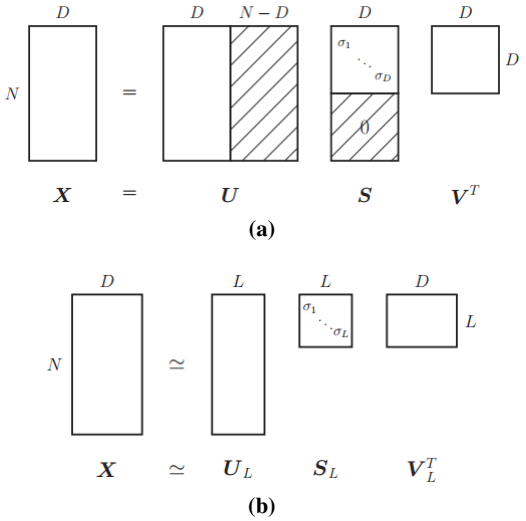


Fig. 12.2: (a) SVD decomposition of non-square matrices $\vec{X} = \vec{U}\vec{\Sigma}\vec{V}^T$. The shaded parts of $\vec{\Sigma}$, and all the off-diagonal terms, are zero. The shaded entries in $\vec{U}$ and $\vec{\Sigma}$ are not computed in the economy-sized version, since they are not needed. (b) Truncated SVD approximation of rank L.

The connection between eigenvectors and singular vectors is the following:

$$\vec{U} = \text{evec}(\vec{X}\vec{X}^T) \tag{12.34}$$

$$\vec{V} = \text{evec}(\vec{X}^T\vec{X}) \tag{12.35}$$

$$\vec{\Sigma}^2 = \text{eval}(\vec{X}\vec{X}^T) = \text{eval}(\vec{X}^T\vec{X}) \tag{12.36}$$

For the proof please read Section 12.2.3 of MLAPP.

Since the eigenvectors are unaffected by linear scaling of a matrix, we see that the right singular vectors of $\vec{X}$ are equal to the eigenvectors of the empirical covariance $\hat{\vec{\Sigma}}$. Furthermore, the eigenvalues of $\hat{\vec{\Sigma}}$ are a scaled version of the squared singular values.

However, the connection between PCA and SVD goes deeper. From Equation 12.31, we can represent a rank r matrix as follows:

$$\vec{X} = \sigma_1 \begin{pmatrix} | \\ \vec{u}_1 \\ | \end{pmatrix} \begin{pmatrix} - & \vec{v}_1 & - \end{pmatrix} + \cdots + \sigma_r \begin{pmatrix} | \\ \vec{u}_r \\ | \end{pmatrix} \begin{pmatrix} - & \vec{v}_r^T & - \end{pmatrix}$$

If the singular values die off quickly, we can produce a rank L approximation to the matrix as follows:

$$\vec{X} \approx \sigma_1 \begin{pmatrix} | \\ \vec{u}_1 \\ | \end{pmatrix} \begin{pmatrix} - & \vec{v}_1 & - \end{pmatrix} + \cdots + \sigma_r \begin{pmatrix} | \\ \vec{u}_L \\ | \end{pmatrix} \begin{pmatrix} - & \vec{v}_L^T & - \end{pmatrix}$$
$$= \vec{U}_{:,1:L} \vec{\Sigma}_{1:L,1:L} \vec{V}_{:,1:L}^T \tag{12.37}$$

This is called a **truncated SVD** (see Figure 12.2(b)).

One can show that the error in this approximation is given by

$$\|\vec{X} - \vec{X}_L\|_F \approx \sigma_L \tag{12.38}$$

Furthermore, one can show that the SVD offers the best rank L approximation to a matrix (best in the sense of minimizing the above Frobenius norm).

Let us connect this back to PCA. Let $\vec{X} = \vec{U}\vec{\Sigma}\vec{V}^T$ be a truncated SVD of $\vec{X}$. We know that $\hat{\vec{W}} = \vec{V}$, and that $\hat{\vec{Z}} = \vec{X}\hat{\vec{W}}$, so

$$\hat{\vec{Z}} = \vec{U}\vec{\Sigma}\vec{V}^T\vec{V} = \vec{U}\vec{\Sigma} \tag{12.39}$$

Furthermore, the optimal reconstruction is given by $\hat{\vec{X}} = \vec{Z}\hat{\vec{W}}$,so we find

$$\hat{\vec{X}} = \vec{U}\vec{\Sigma}\vec{V}^T \tag{12.40}$$

This is precisely the same as a truncated SVD approximation! This is another illustration of the fact that PCA is the best low rank approximation to the data.

12.2.3 Probabilistic PCA

Theorem 12.3. *((**Tipping and Bishop 1999**)). Consider a factor analysis model in which $\vec{\Psi} = \sigma^2\vec{I}$ and $\vec{W}$ is orthogonal. The observed data log likelihood is given by*

$$\log p(\vec{X}|\vec{W}, \sigma^2\vec{I}) = -\frac{N}{2}\ln|\vec{C}| - \frac{1}{2}\sum_{i=1}^{N} \vec{x}_i^T \vec{C}^{-1}\vec{x}_i$$
$$= -\frac{N}{2}\ln|\vec{C}| + \mathrm{tr}(\vec{C}^{-1}\vec{\Sigma}) \tag{12.41}$$

where $\vec{C} = \vec{W}\vec{W}^T + \sigma^2\vec{I}$ and $\vec{\Sigma} = \frac{1}{N}\sum_{i=1}^N \vec{x}_i\vec{x}_i^T = \frac{1}{N}\vec{X}\vec{X}^T$. (We are assuming centred data, for notational simplicity.) The maxima of the log-likelihood are given by

$$\hat{\vec{W}} = \vec{V}(\vec{\Lambda} - \sigma^2\vec{I})^{\frac{1}{2}}\vec{R} \tag{12.42}$$

where $\vec{R}$ is an arbitrary $L \times L$ orthogonal matrix, $\vec{V}$ is the $D \times L$ matrix whose columns are the first L eigenvectors of $\vec{\Sigma}$, and $\vec{\Lambda}$ is the corresponding diagonal matrix of eigenvalues. Without loss of generality, we can set $\vec{R} = \vec{I}$. Furthermore, the MLE of the noise variance is given by

$$\hat{\sigma}^2 = \frac{1}{D-L}\sum_{j=L+1}^{D} \lambda_j \tag{12.43}$$

which is the average variance associated with the discarded dimensions.

Thus, as $\sigma^2 \to 0$, we have $\hat{\vec{W}} \to \vec{V}$, as in classical PCA. What about $\hat{\vec{Z}}$? It is easy to see that the posterior over the latent factors is given by

$$p(\vec{z}_i|\vec{x}_i, \hat{\vec{\theta}}) = \mathcal{N}(\vec{z}_i|\hat{\vec{F}}^{-1}\hat{\vec{W}}^T\vec{x}_i, \sigma^2\hat{\vec{F}}^{-1}) \tag{12.44}$$
$$\hat{\vec{F}} \triangleq \hat{\vec{W}}^T\hat{\vec{W}} + \sigma^2\vec{I} \tag{12.45}$$

(Do not confuse $\vec{F} = \vec{W}^T\vec{W} + \sigma^2\vec{I}$ with $\vec{C} = \vec{W}\vec{W}^T + \sigma^2\vec{I}$.) Hence, as $\sigma^2 \to 0$, we find $\hat{\vec{W}} \to \vec{V}$, $\hat{\vec{F}} \to \vec{I}$ and $\vec{z}_i \to \vec{V}^T\vec{x}_i$. Thus the posterior mean is obtained by an orthogonal projection of the data onto the column space of $\vec{V}$, as in classical PCA.

Note, however, that if $\sigma^2 \to 0$, the posterior mean is not an orthogonal projection, since it is shrunk somewhat towards the prior mean, as illustrated in Figure 12.1(b). This sounds like an undesirable property, but it means that the reconstructions will be closer to the overall data mean, $\hat{\vec{\mu}} = \bar{\vec{x}}$.

12.2.4 EM algorithm for PCA

Although the usual way to fit a PCA model uses eigenvector methods, or the SVD, we can also use EM, which will turn out to have some advantages that we discuss below. EM for PCA relies on the probabilistic formulation of PCA. However the algorithm continues to work in the zero noise limit, $\sigma^2 = 0$, as shown by (Roweis 1997).

Let $\tilde{\vec{Z}}$ be a $L \times N$ matrix storing the posterior means (low-dimensional representations) along its columns. Similarly, let $\tilde{\vec{X}} = \vec{X}^T$ store the original data along its columns. From Equation 12.44, when $\sigma^2 = 0$, we have

$$\tilde{\vec{Z}} = (\vec{W}^T\vec{W})^{-1}\vec{W}^T\tilde{\vec{X}} \tag{12.46}$$

This constitutes the E step. Notice that this is just an orthogonal projection of the data.

From Equation 12.26, the M step is given by

$$\hat{\vec{W}} = \left(\sum_{i=1}^{N} \vec{x}_i\mathbb{E}[\vec{z}_i]^T\right)\left(\sum_{i=1}^{N} \mathbb{E}[\vec{z}_i]\mathbb{E}[\vec{z}_i]^T\right)^{-1} \tag{12.47}$$

where we exploited the fact that $\vec{\Sigma} = \mathrm{cov}[\vec{z}_i|\vec{x}_i, \vec{\theta}] = \vec{0}$ when $\sigma^2 = 0$.

(Tipping and Bishop 1999) showed that the only stable fixed point of the EM algorithm is the globally optimal solution. That is, the EM algorithm converges to a solution where $\vec{W}$ spans the same linear subspace as that defined by the first L eigenvectors. However, if we want $\vec{W}$ to be orthogonal, and to contain the eigenvectors in descending order of eigenvalue, we have to orthogonalize the resulting matrix (which can be done quite cheaply). Alternatively, we can modify EM to give the principal basis directly (Ahn and Oh 2003).

This algorithm has a simple physical analogy in the case $D = 2$ and $L = 1$(Roweis 1997). Consider some points in $\mathbb{R}^2$ attached by springs to a rigid rod, whose orientation is defined by a vector $\vec{w}$. Let $\vec{z}_i$ be the location where the i'th spring attaches to the rod. See Figure 12.11 of MLAPP for an illustration.

Apart from this pleasing intuitive interpretation, EM for PCA has the following advantages over eigenvector methods:

- EM can be faster. In particular, assuming $N, D \gg L$, the dominant cost of EM is the projection operation in the E step, so the overall time is $O(TLND)$, where T is the number of iterations. This is much faster than the $O(\min(ND^2, DN^2))$ time required by straightforward eigenvector methods, although more sophisticated eigenvector methods, such as the Lanczos algorithm, have running times comparable to EM.
- EM can be implemented in an online fashion, i.e., we can update our estimate of $\vec{W}$ as the data streams in.
- EM can handle missing data in a simple way (see Section 12.1.6).
- EM can be extended to handle mixtures of PPCA/ FA models.
- EM can be modified to variational EM or to variational Bayes EM to fit more complex models.

12.3 Choosing the number of latent dimensions

In Section 11.5, we discussed how to choose the number of components K in a mixture model. In this section, we discuss how to choose the number of latent dimensions L in a FA/PCA model.

12.3.1 Model selection for FA/PPCA

TODO

12.3.2 Model selection for PCA

TODO

12.4 PCA for categorical data

In this section, we consider extending the factor analysis model to the case where the observed data is categorical rather than real-valued. That is, the data has the form $y_{ij} \in \{1, ..., C\}$, where $j = 1 : R$ is the number of observed response variables. We assume each y_{ij} is generated from a latent variable $\vec{z}_i \in \mathbb{R}^L$, with a Gaussian prior, which is passed through the softmax function as follows:

$$p(\vec{z}_i) = \mathcal{N}(\vec{z}_i | \vec{0}, \vec{I}) \tag{12.48}$$

$$p(\vec{y}_i | \vec{z}_i, \vec{\theta}) = \prod_{j=1}^{R} \text{Cat}(y_{ir} | \mathcal{S}(\vec{W}_r^T \vec{z}_i + \vec{w}_{0r})) \tag{12.49}$$

where $\vec{W}_r \in \mathbb{R}^L$ is the factor loading matrix for response j, and $\vec{W}_{0r} \in \mathbb{R}^M$ is the offset term for response r, and $\vec{\theta} = (\vec{W}_r, \vec{W}_{0r})_{r=1}^R$. (We need an explicit offset term, since clamping one element of $\vec{z}_i$ to 1 can cause problems when computing the posterior covariance.) As in factor analysis, we have defined the prior mean to be $\vec{\mu}_0 = \vec{0}$ and the prior covariance $\vec{V}_0 = \vec{I}$, since we can capture non-zero mean by changing $\vec{w}_{0j}$ and non-identity covariance by changing $\vec{W}_r$. We will call this categorical PCA. See Chapter 27 TODO for a discussion of related models.

In (Khan et al. 2010), we show that this model outperforms finite mixture models on the task of imputing missing entries in design matrices consisting of real and categorical data. This is useful for analysing social science survey data, which often has missing data and variables of mixed type.

12.5 PCA for paired and multi-view data

12.5.1 Supervised PCA (latent factor regression)

12.5.2 Discriminative supervised PCA

12.5.3 Canonical correlation analysis

12.6 Independent Component Analysis (ICA)

Let $\vec{x}_t \in \mathbb{R}^D$ be the observed signal at the sensors at "time" t, and $\vec{z}_t \in \mathbb{R}^L$ be the vector of source signals. We assume that

$$\vec{x}_t = \vec{W}\vec{z}_t + \vec{\varepsilon}_t \tag{12.50}$$

where $\vec{W}$ is an $D \times L$ matrix, and $\vec{\varepsilon}_t \sim \mathcal{N}(\vec{0}, \vec{\Psi})$. In this section, we treat each time point as an independent observation, i.e., we do not model temporal correlation (so we could replace the t index with i, but we stick with t to be consistent with much of the ICA literature). The goal is to infer the source signals, $p(\vec{z}_t | \vec{x}_t, \vec{\theta})$. In this context, $\vec{W}$ is called the **mixing matrix**. If $L = D$ (number of sources = number of sensors), it will be a square matrix. Often we will assume the noise level, $|\vec{\Psi}|$, is zero, for simplicity.

So far, the model is identical to factor analysis. However, we will use a different prior for $p(\vec{z}_t)$. In PCA, we assume each source is independent, and has a Gaussian distribution. We will now relax this Gaussian assumption and let the source distributions be any *non-Gaussian* distribution

$$p(\vec{z}_t) = \prod_{j=1}^{L} p_j(z_{tj}) \tag{12.51}$$

Without loss of generality, we can constrain the variance of the source distributions to be $\vec{1}$, because any other variance can be modelled by scaling the rows of $\vec{W}$ appropriately. The resulting model is known as **independent component analysis** or **ICA**.

The reason the Gaussian distribution is disallowed as a source prior in ICA is that it does not permit unique recovery of the sources. This is because the PCA likelihood is invariant to any orthogonal transformation of the sources $\vec{z}_t$ and mixing matrix $\vec{W}$. PCA can recover the best linear subspace in which the signals lie, but cannot uniquely recover the signals themselves.

ICA requires that $\vec{W}$ is square and hence invertible. In the non-square case (e.g., where we have more sources than sensors), we cannot uniquely recover the true signal, but we can compute the posterior $p(\vec{z}_t|\vec{x}_t, \hat{\vec{W}})$, which represents our beliefs about the source. In both cases, we need to estimate Was well as the source distributions p_j. We discuss how to do this below.

12.6.1 Maximum likelihood estimation

In this section, we discuss ways to estimate square mixing matrices $\vec{W}$ for the noise-free ICA model. As usual, we will assume that the observations have been centered; hence we can also assume $\vec{z}$ is zero-mean. In addition, we assume the observations have been whitened, which can be done with PCA.

If the data is centered and whitened, we have $\mathbb{E}[\vec{x}\vec{x}^T] = \vec{I}$. But in the noise free case, we also have

$$\text{cov}[\vec{x}] = \mathbb{E}[\vec{x}\vec{x}^T] = \vec{W}\mathbb{E}[\vec{z}\vec{z}^T]\vec{W}^T \tag{12.52}$$

Hence we see that $\vec{W}$ must be orthogonal. This reduces the number of parameters we have to estimate from D^2 to $D(D-1)/2$. It will also simplify the math and the algorithms.

Let $\vec{V} = \vec{W}^{-1}$; these are often called the recognition weights, as opposed to $\vec{W}$, which are the generative weights.

Since $\vec{x} = \vec{W}\vec{z}$, we have, from Equation 2.46,

$$\begin{aligned} p_x(\vec{W}\vec{z}_t) &= p_z(\vec{z}_t)|\det(\vec{W}^{-1})| \\ &= p_z(\vec{V}\vec{x}_t)|\det(\vec{V})| \end{aligned} \tag{12.53}$$

Hence we can write the log-likelihood, assuming T iid samples, as follows:

$$\frac{1}{T}\log p(\mathcal{D}|\vec{V}) = \log|\det(\vec{V})| + \frac{1}{T}\sum_{j=1}^{L}\sum_{t=1}^{T}\log p_j(\vec{v}_j^T\vec{x}_t)$$

where $\vec{v}_j$ is the j'th row of $\vec{V}$. Since we are constraining $\vec{V}$ to be orthogonal, the first term is a constant, so we can drop it. We can also replace the average over the data with an expectation operator to get the following objective

$$\text{NLL}(\vec{V}) = \sum_{j=1}^{L}\mathbb{E}[G_j(\vec{z}_j)] \tag{12.54}$$

where $\vec{z}_j = \vec{v}_j^T\vec{x}$ and $G_j(\vec{z}) \triangleq -\log p_j(\vec{z})$. We want to minimize this subject to the constraint that the rows of $\vec{V}$ are orthogonal. We also want them to be unit norm, since this ensures that the variance of the factors is unity (since, with whitened data, $\mathbb{E}[\vec{v}_j^T\vec{x}] = \|\vec{v}_j\|^2$, which is necessary to fix the scale of the weights. In otherwords, $\vec{V}$ should be an orthonormal matrix.

It is straightforward to derive a gradient descent algorithm to fit this model; however, it is rather slow. One can also derive a faster algorithm that follows the natural gradient; see e.g., (MacKay 2003, ch 34) for details. A popular alternative is to use an approximate Newton method, which we discuss in Section 12.6.2. Another approach is to use EM, which we discuss in Section 12.6.3.

12.6.2 The FastICA algorithm

12.6.3 Using EM

12.6.4 Other estimation principles *

Chapter 13
Sparse linear models

Chapter 14
Kernels

14.1 Introduction

So far in this book, we have been assuming that each object that we wish to classify or cluster or process in anyway can be represented as a fixed-size feature vector, typically of the form $\vec{x}_i \in \mathbb{R}^D$. However, for certain kinds of objects, it is not clear how to best represent them as fixed-sized feature vectors. For example, how do we represent a text document or protein sequence, which can be of variable length? or a molecular structure, which has complex 3d geometry? or an evolutionary tree, which has variable size and shape?

One approach to such problems is to define a generative model for the data, and use the inferred latent representation and/or the parameters of the model as features, and then to plug these features in to standard methods. For example, in Chapter 28 TODO, we discuss deep learning, which is essentially an unsupervised way to learn good feature representations.

Another approach is to assume that we have some way of measuring the similarity between objects, that doesn't require preprocessing them into feature vector format. For example, when comparing strings, we can compute the edit distance between them. Let $\kappa(\vec{x},\vec{x}') \geq 0$ be some measure of similarity between objects $\kappa(\vec{x},\vec{x}') \in \mathcal{X}$, we will call κ a **kernel function**. Note that the word "kernel" has several meanings; we will discuss a different interpretation in Section 14.7.1 TODO.

In this chapter, we will discuss several kinds of kernel functions. We then describe some algorithms that can be written purely in terms of kernel function computations. Such methods can be used when we don't have access to (or choose not to look at) the "inside" of the objects $\vec{x}$ that we are processing.

14.2 Kernel functions

Definition 14.1. A **kernel function**[23] is a real-valued function of two arguments, $\kappa(\vec{x},\vec{x}') \in \mathbb{R}$. Typically the function is symmetric (i.e., $\kappa(\vec{x},\vec{x}') = \kappa(\vec{x}',\vec{x})$, and non-negative (i.e., $\kappa(\vec{x},\vec{x}') \geq 0$).

We give several examples below.

[23] http://en.wikipedia.org/wiki/Kernel_function

14.2.1 RBF kernels

The **Gaussian kernel** or **squared exponential kernel**(SE kernel) is defined by

$$\kappa(\vec{x},\vec{x}') = \exp\left(-\frac{1}{2}(\vec{x}-\vec{x}')^T\vec{\Sigma}^{-1}(\vec{x}-\vec{x}')\right) \tag{14.1}$$

If $\vec{\Sigma}$ is diagonal, this can be written as

$$\kappa(\vec{x},\vec{x}') = \exp\left(-\frac{1}{2}\sum_{j=1}^{D}\frac{1}{\sigma_j^2}(x_j - x'_j)^2\right) \tag{14.2}$$

We can interpret the σ_j as defining the **characteristic length scale** of dimension j.If $\sigma_j = \infty$, the corresponding dimension is ignored; hence this is known as the **ARD kernel**. If $\vec{\Sigma}$ is spherical, we get the isotropic kernel

$$\kappa(\vec{x},\vec{x}') = \exp\left(-\frac{\|\vec{x}-\vec{x}'\|^2}{2\sigma^2}\right) \tag{14.3}$$

Here σ^2 is known as the **bandwidth**. Equation 14.4 is an example of a **radial basis function** or **RBF** kernel, since it is only a function of $\|\vec{x}-\vec{x}'\|^2$.

14.2.2 TF-IDF kernels

$$\kappa(\vec{x},\vec{x}') = \frac{\phi(\vec{x})^T\phi(\vec{x}')}{\|\phi(vecx)\|_2\|\phi(\vec{x}')\|_2} \tag{14.4}$$

where $\phi(\vec{x}) = \text{tf-idf}(\vec{x})$.

14.2.3 Mercer (positive definite) kernels

If the kernel function satisfies the requirement that the **Gram matrix**, defined by

$$\vec{K} \triangleq \begin{pmatrix} \kappa(\vec{x}_1,\vec{x}_2) & \cdots & \kappa(\vec{x}_1,\vec{x}_N) \\ \vdots & \vdots & \vdots \\ \kappa(\vec{x}_N,\vec{x}_1) & \cdots & \kappa(\vec{x}_N,\vec{x}_N) \end{pmatrix} \tag{14.5}$$

be positive definite for any set of inputs $\{\vec{x}_i\}_{i=1}^N$. We call such a kernel a **Mercer kernel**,or **positive definite kernel**.

If the Gram matrix is positive definite, we can compute an eigenvector decomposition of it as follows

$$\vec{K} = \vec{U}^T\vec{\Lambda}\vec{U} \tag{14.6}$$

where $\vec{\Lambda}$ is a diagonal matrix of eigenvalues $\lambda_i > 0$. Now consider an element of $\vec{K}$:

$$k_{ij} = (\vec{\Lambda}^{\frac{1}{2}}\vec{U}_{:,i})^T(\vec{\Lambda}^{\frac{1}{2}}\vec{U}_{:,j}) \tag{14.7}$$

Let us define $\phi(\vec{x}_i) = \vec{\Lambda}^{\frac{1}{2}}\vec{U}_{:,i}$, then we can write

$$k_{ij} = \phi(\vec{x}_i)^T\phi(\vec{x}_j) \tag{14.8}$$

Thus we see that the entries in the kernel matrix can be computed by performing an inner product of some feature vectors that are implicitly defined by the eigenvectors $\vec{U}$. In general, if the kernel is Mercer, then there exists a function ϕ mapping $\vec{x} \in \mathcal{X}$ to $\mathbb{R}^D$ such that

$$\kappa(\vec{x},\vec{x}') = \phi(\vec{x})^T\phi(\vec{x}') \tag{14.9}$$

where ϕ depends on the eigen functions of κ(so D is a potentially infinite dimensional space).

For example, consider the (non-stationary) **polynomial kernel** $\kappa(\vec{x},\vec{x}') = (\gamma\vec{x}\vec{x}' + r)^M$, where $r > 0$. One can show that the corresponding feature vector $\phi(\vec{x})$ will contain all terms up to degree M. For example, if $M = 2, \gamma = r = 1$ and $\vec{x},\vec{x}' \in \mathbb{R}^2$, we have

$$\begin{aligned}(\vec{x}\vec{x}' + 1)^2 &= (1 + x_1x_1' + x_2 + x_2')^2 \\ &= 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 x_1x_1'x_2x_2' \\ &= \phi(\vec{x})^T\phi(\vec{x}') \\ \text{where } \phi(\vec{x}) &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)\end{aligned}$$

In the case of a Gaussian kernel, the feature map lives in an infinite dimensional space. In such a case, it is clearly infeasible to explicitly represent the feature vectors.

In general, establishing that a kernel is a Mercer kernel is difficult, and requires techniques from functional analysis. However, one can show that it is possible to build up new Mercer kernels from simpler ones using a set of standard rules. For example, if κ_1 and κ_2 are both Mercer, so is $\kappa(\vec{x},\vec{x}') = \kappa_1(\vec{x},\vec{x}') + \kappa_2(\vec{x},\vec{x}') =$. See e.g., (Schoelkopf and Smola 2002) for details.

14.2.4 Linear kernels

$$\kappa(\vec{x},\vec{x}') = \vec{x}^T\vec{x}' \tag{14.10}$$

14.2.5 Matern kernels

The **Matern kernel**, which is commonly used in Gaussian process regression (see Section 15.2), has the following form

$$\kappa(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{\ell}\right)^{\nu} K_\nu\frac{\sqrt{2\nu}r}{\ell} \tag{14.11}$$

where $r = \|\vec{x} - \vec{x}'\|$, $\nu > 0$, $\ell > 0$, and K_ν is a modified Bessel function. As $\nu \to \infty$, this approaches the SE kernel. If $\nu = \frac{1}{2}$, the kernel simplifies to

$$\kappa(r) = \exp(-r/\ell) \tag{14.12}$$

If $D = 1$, and we use this kernel to define a Gaussian process (see Chapter 15 TODO), we get the **Ornstein-Uhlenbeck process**, which describes the velocity of a particle undergoing Brownian motion (the corresponding function is continuous but not differentiable, and hence is very "jagged").

14.2.6 String kernels

Now let $\phi_s(\vec{x})$ denote the number of times that substrings appears in string $\vec{x}$. We define the kernel between two strings $\vec{x}$ and $\vec{x}'$ as

$$\kappa(\vec{x},\vec{x}') = \sum_{s\in\mathcal{A}^*} w_s\phi_s(\vec{x})\phi_s(\vec{x}') \tag{14.13}$$

where $w_s \geq 0$ and $\mathcal{A}^*$ is the set of all strings (of any length) from the alphabet $\mathcal{A}$(this is known as the Kleene star operator). This is a Mercer kernel, and be computed in $O(|\vec{x}| + |\vec{x}'|)$ time (for certain settings of the weights $\{ws\}$) using suffix trees (Leslie et al. 2003; Vishwanathan and Smola 2003; Shawe-Taylor and Cristianini 2004).

There are various cases of interest. If we set $w_s = 0$for $|s| > 1$ we get a bag-of-characters kernel. This defines $\phi(\vec{x})$ to be the number of times each character in $\mathcal{A}$ occurs in $\vec{x}$.If we require s to be bordered by white-space, we get a bag-of-words kernel, where $\phi(\vec{x})$ counts how many times each possible word occurs. Note that this is a very sparse vector, since most words will not be present. If we only consider strings of a fixed lengthk, we get the **k-spectrum** kernel. This has been used to classify proteins into SCOP superfamilies (Leslie et al. 2003).

14.2.7 Pyramid match kernels

14.2.8 Kernels derived from probabilistic generative models

Suppose we have a probabilistic generative model of feature vectors, $p(\vec{x}|\vec{\theta})$. Then there are several ways we can use this model to define kernel functions, and thereby make the model suitable for discriminative tasks. We sketch two approaches below.

14.2.8.1 Probability product kernels

$$\kappa(\vec{x}_i,\vec{x}_j) = \int p(\vec{x}|\vec{x}_i)^\rho p(\vec{x}|\vec{x}_j)^\rho \mathrm{d}\vec{x} \tag{14.14}$$

where $\rho > 0$, and $p(\vec{x}|\vec{x}_i)$ is often approximated by $p(\vec{x}|\hat{\vec{\theta}}(\vec{x}_i))$,where $\hat{\vec{\theta}}(\vec{x}_i)$ is a parameter estimate computed

using a single data vector. This is called a **probability product kernel**(Jebara et al. 2004).

Although it seems strange to fit a model to a single data point, it is important to bear in mind that the fitted model is only being used to see how similar two objects are. In particular, if we fit the model to $\vec{x}_i$ and then the model thinks $\vec{x}_j$ is likely, this means that $\vec{x}_i$ and $\vec{x}_j$ are similar. For example, suppose $p(\vec{x}|\vec{\theta}) \sim \mathcal{N}(\vec{\mu}, \sigma^2\vec{I})$, where σ^2 is fixed. If $\rho = 1$, and we use $\hat{\vec{\mu}}(\vec{x}_i) = \vec{x}_i$ and $\hat{\vec{\mu}}(\vec{x}_j) = \vec{x}_j$, we find (Jebara et al. 2004, p825) that

$$\kappa(\vec{x}_i, \vec{x}_j) = \frac{1}{(4\pi\sigma^2)^{D/2}} \exp\left(-\frac{1}{4\sigma^2}\|\vec{x}_i - \vec{x}_j\|^2\right) \quad (14.15)$$

which is (up to a constant factor) the RBF kernel.

It turns out that one can compute Equation 14.14 for a variety of generative models, including ones with latent variables, such as HMMs. This provides one way to define kernels on variable length sequences. Furthermore, this technique works even if the sequences are of real-valued vectors, unlike the string kernel in Section 14.2.6. See (Jebara et al. 2004) for further details

14.2.8.2 Fisher kernels

A more efficient way to use generative models to define kernels is to use a **Fisher kernel** (Jaakkola and Haussler 1998) which is defined as follows:

$$\kappa(\vec{x}_i, \vec{x}_j) = g(\vec{x}_i)^T \vec{F}^{-1} g(\vec{x}_j) \quad (14.16)$$

where g is the gradient of the log likelihood, or **score vector**, evaluated at the MLE $\hat{\vec{\theta}}$

$$g(\vec{x}) \triangleq \frac{\mathrm{d}}{\mathrm{d}\vec{\theta}} \log p(\vec{x}|\vec{\theta})|_{\hat{\vec{\theta}}} \quad (14.17)$$

and $\vec{F}$ is the Fisher information matrix, which is essentially the Hessian:

$$\vec{F} \triangleq \left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \log p(\vec{x}|\vec{\theta})\right]|_{\hat{\vec{\theta}}} \quad (14.18)$$

Note that $\hat{\vec{\theta}}$ is a function of all the data, so the similarity of $\vec{x}_i$ and $\vec{x}_j$ is computed in the context of all the data as well. Also, note that we only have to fit one model.

14.3 Using kernels inside GLMs

14.3.1 Kernel machines

We define a **kernel machine** to be a GLM where the input feature vector has the form

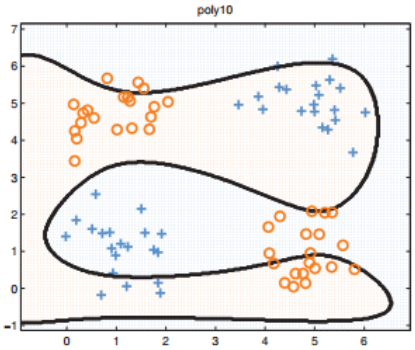
$$\phi(\vec{x}) = (\kappa(\vec{x}, \vec{\mu}_1), \cdots, \kappa(\vec{x}, \vec{\mu}_K)) \quad (14.19)$$

where $\vec{\mu}_k \in \mathcal{X}$ are a set of K centroids. If κ is an RBF kernel, this is called an **RBF network**. We discuss ways to choose the $\vec{\mu}_k$ parameters below. We will call Equation 14.19 a **kernelised feature vector**. Note that in this approach, the kernel need not be a Mercer kernel.

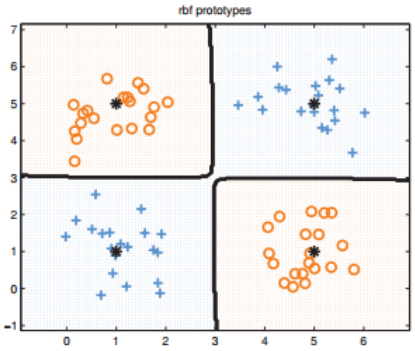
We can use the kernelized feature vector for logistic regression by defining $p(y|\vec{x}, \vec{\theta}) = \mathrm{Ber}(y|\vec{w}^T\phi(\vec{x}))$. This provides a simple way to define a non-linear decision boundary. For example, see Figure 14.1.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)



(c)

Fig. 14.1: (a) xor truth table. (b) Fitting a linear logistic regression classifier using degree 10 polynomial expansion. (c) Same model, but using an RBF kernel with centroids specified by the 4 black crosses.

We can also use the kernelized feature vector inside a linear regression model by defining $p(y|\vec{x}, \vec{\theta}) = \mathcal{N}(y|\vec{w}^T\phi(\vec{x}), \sigma^2)$.

14.3.2 L1VMs, RVMs, and other sparse vector machines

The main issue with kernel machines is: how do we choose the centroids $\vec{\mu}_k$? We can use **sparse vector machine**, **L1VM**, **L2VM**, **RVM**, **SVM**.

14.4 The kernel trick

Rather than defining our feature vector in terms of kernels, $\phi(\vec{x}) = (\kappa(\vec{x},\vec{x}_1),\cdots,\kappa(\vec{x},\vec{x}_N))$, we can instead work with the original feature vectors $\vec{x}$, but modify the algorithm so that it replaces all inner products of the form $<\vec{x}_i,\vec{x}_j>$ with a call to the kernel function, $\kappa(\vec{x}_i,\vec{x}_j)$. This is called the **kernel trick**. It turns out that many algorithms can be kernelized in this way. We give some examples below. Note that we require that the kernel be a Mercer kernel for this trick to work.

14.4.1 Kernelized KNN

The Euclidean distance can be unrolled as

$$\|\vec{x}_i - \vec{x}_j\| = <\vec{x}_i,\vec{x}_i> + <\vec{x}_j,\vec{x}_j> -2<\vec{x}_i,\vec{x}_j> \quad (14.20)$$

then by replacing all $<\vec{x}_i,\vec{x}_j>$ with $\kappa(\vec{x}_i,\vec{x}_j)$ we get Kernelized KNN.

14.4.2 Kernelized K-medoids clustering

K-medoids algorothm is similar to K-means(see Section 11.4.4), but instead of representing each cluster's centroid by the mean of all data vectors assigned to this cluster, we make each centroid be one of the data vectors themselves. Thus we always deal with integer indexes, rather than data objects.

This algorithm can be kernelized by using Equation 14.20 to replace the distance computation.

14.4.3 Kernelized ridge regression

Applying the kernel trick to distance-based methods was straightforward. It is not so obvious how to apply it to parametric models such as ridge regression. However, it can be done, as we now explain. This will serve as a good "warm up" for studying SVMs.

14.4.3.1 The primal problem

we rewrite Equation 7.22 as the following

$$J(\vec{w}) = (\vec{y} - \vec{X}\vec{w})^T(\vec{y} - \vec{X}\vec{w}) + \lambda\|\vec{w}\|^2 \quad (14.21)$$

and its solution is given by Equation 7.23.

14.4.3.2 The dual problem

Equation 14.21 is not yet in the form of inner products. However, using the matrix inversion lemma (Equation 4.107 TODO) we rewrite the ridge estimate as follows

$$\vec{w} = \vec{X}^T(\vec{X}\vec{X}^T + \lambda\vec{I}_N)^{-1}\vec{y} \quad (14.22)$$

which takes $O(N^3 + N^2D)$ time to compute. This can be advantageous if D is large. Furthermore, we see that we can partially kernelize this, by replacing $\vec{X}\vec{X}^T$ with the Gram matrix $\vec{K}$. But what about the leading $\vec{X}^T$ term?

Let us define the following **dual variables**:

$$\vec{\alpha} = (\vec{K} + \lambda\vec{I}_N)^{-1}\vec{y} \quad (14.23)$$

Then we can rewrite the **primal variables** as follows

$$\vec{w} = \vec{X}^T\vec{\alpha} = \sum_{i=1}^{N}\alpha_i\vec{x}_i \quad (14.24)$$

This tells us that the solution vector is just a linear sum of the N training vectors. When we plug this in at test time to compute the predictive mean, we get

$$y = f(\vec{x}) = \sum_{i=1}^{N}\alpha_i\vec{x}_i^T\vec{x} = \sum_{i=1}^{N}\alpha_i\kappa(\vec{x}_i,\vec{x}) \quad (14.25)$$

So we have succesfully kernelized ridge regression by changing from primal to dual variables. This technique can be applied to many other linear models, such as logistic regression.

14.4.3.3 Computational cost

The cost of computing the dual variables $\vec{\alpha}$ is $O(N^3)$, whereas the cost of computing the primal variables $\vec{w}$ is $O(D^3)$. Hence the kernel method can be useful in high dimensional settings, even if we only use a linear kernel (c.f., the SVD trick in Equation 7.24). However, prediction using the dual variables takes $O(ND)$ time, while prediction using the primal variables only takes $O(D)$ time. We can speedup prediction by making $\vec{\alpha}$ sparse, as we discuss in Section 14.5.

14.4.4 Kernel PCA

TODO

14.5 Support vector machines (SVMs)

In Section 14.3.2, we saw one way to derive a sparse kernel machine, namely by using a GLM with kernel basis functions, plus a sparsity-promoting prior such as ℓ_1 or ARD. An alternative approach is to change the objective function from negative log likelihood to some other loss function, as we discussed in Section 6.4.5. In particular, consider the ℓ_2 regularized empirical risk function

$$J(\vec{w},\lambda) = \sum i = 1^N L(y_i,\hat{y_i}) + \lambda \|\vec{w}\|^2 \tag{14.26}$$

where $\hat{y_i} = \vec{w}^T \vec{x}_i + w_0$.

If L is quadratic loss, this is equivalent to ridge regression, and if L is the log-loss defined in Equation 6.3, this is equivalent to logistic regression.

However, if we replace the loss function with some other loss function, to be explained below, we can ensure that the solution is sparse, so that predictions only depend on a subset of the training data, known as **support vectors**. This combination of the kernel trick plus a modified loss function is known as a **support vector machine** or **SVM**.

Note that SVMs are very unnatural from a probabilistic point of view.

- First, they encode sparsity in the loss function rather than the prior.
- Second, they encode kernels by using an algorithmic trick, rather than being an explicit part of the model.
- Finally, SVMs do not result in probabilistic outputs, which causes various difficulties, especially in the multi-class classification setting (see Section 14.5.2.4 TODO for details).

It is possible to obtain sparse, probabilistic, multi-class kernel-based classifiers, which work as well or better than SVMs, using techniques such as the L1VM or RVM, discussed in Section 14.3.2. However, we include a discussion of SVMs, despite their non-probabilistic nature, for two main reasons.

- First, they are very popular and widely used, so all students of machine learning should know about them.
- Second, they have some computational advantages over probabilistic methods in the structured output case; see Section 19.7 TODO.

14.5.1 SVMs for classification

14.5.1.1 Primal form

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b) \tag{14.27}$$

Evaluation

$$\min_{\vec{w},b} \quad \frac{1}{2}\|\vec{w}\|^2 \tag{14.28}$$

$$s.t. \quad y_i(\vec{w}\vec{x}_i + b) \geqslant 1, i = 1,2,\ldots,N \tag{14.29}$$

14.5.1.2 Dual form

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b\right) \tag{14.30}$$

Evaluation

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) - \sum_{i=1}^{N} \alpha_i \tag{14.31}$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{14.32}$$

$$\alpha_i \geqslant 0, i = 1,2,\ldots,N \tag{14.33}$$

14.5.1.3 Primal form with slack variables

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b) \tag{14.34}$$

Evaluation

$$\min_{\vec{w},b} \quad C\sum_{i=1}^{N} \xi_i + \frac{1}{2}\|\vec{w}\|^2 \tag{14.35}$$

$$s.t. \quad y_i(\vec{w}\vec{x}_i + b) \geqslant 1 - \xi_i \tag{14.36}$$

$$\xi_i \geqslant 0, \quad i = 1,2,\ldots,N \tag{14.37}$$

14.5.1.4 Dual form with slack variables

Representation

$$\mathcal{H} : y = f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b\right) \tag{14.38}$$

Evaluation

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) - \sum_{i=1}^{N} \alpha_i \tag{14.39}$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{14.40}$$

$$0 \leqslant \alpha_i \leqslant C, i = 1,2,\ldots,N \tag{14.41}$$

$$\alpha_i = 0 \Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) \geqslant 1 \tag{14.42}$$

$$\alpha_i = C \Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) \leqslant 1 \tag{14.43}$$

$$0 < \alpha_i < C \Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) = 1 \tag{14.44}$$

14.5.1.5 Hinge Loss

Linear support vector machines can also be interpreted as hinge loss minimization:

$$\min_{\vec{w},b} \sum_{i=1}^{N} L(y_i, f(\vec{x}_i)) + \lambda \|\vec{w}\|^2 \tag{14.45}$$

where $L(y, f(\vec{x}))$ is a **hinge loss function**:

$$L(y, f(\vec{x})) = \begin{cases} 1 - yf(x), & 1 - yf(x) > 0 \\ 0, & 1 - yf(x) \leqslant 0 \end{cases} \tag{14.46}$$

Proof. We can write equation 14.45 as equations 14.35 $\sim$ 14.37.

Define slack variables

$$\xi_i \triangleq 1 - y_i(\vec{w} \cdot \vec{x}_i + b), \xi_i \geqslant 0 \tag{14.47}$$

Then $\vec{w}, b, \xi_i$ satisfy the constraints 14.35 and 14.36. And objective function 14.37 can be written as

$$\min_{\vec{w},b} \sum_{i=1}^{N} \xi_i + \lambda \|\vec{w}\|^2$$

If $\lambda = \frac{1}{2C}$, then

$$\min_{\vec{w},b} \frac{1}{C} \left(C \sum_{i=1}^{N} \xi_i + \frac{1}{2} \|\vec{w}\|^2 \right) \tag{14.48}$$

It is equivalent to equation 14.35.

14.5.1.6 Optimization

QP, SMO

14.5.2 SVMs for regression

14.5.2.1 Representation

$$\mathcal{H} : y = f(\vec{x}) = \vec{w}^T \vec{x} + b \tag{14.49}$$

14.5.2.2 Evaluation

$$J(\vec{w}) = C \sum_{i=1}^{N} L(y_i, f(\vec{x}_i)) + + \frac{1}{2} \|\vec{w}\|^2 \tag{14.50}$$

where $L(y, f(\vec{x}))$ is a **epsilon insensitive loss function**:

$$L(y, f(\vec{x})) = \begin{cases} 0 & , |y - f(\vec{x})| < \varepsilon \\ |y - f(\vec{x})| - \varepsilon & , \text{otherwise} \end{cases} \tag{14.51}$$

and $C = 1/\lambda$ is a regularization constant.

This objective is convex and unconstrained, but not differentiable, because of the absolute value function in the loss term. As in Section 13.4 TODO, where we discussed the lasso problem, there are several possible algorithms we could use. One popular approach is to formulate the problem as a constrained optimization problem. In particular, we introduce **slack variables** to represent the degree to which each point lies outside the tube:

$$y_i \leq f(\vec{x}_i) + \varepsilon + \xi_i^+$$
$$y_i \geq f(\vec{x}_i) - \varepsilon - \xi_i^-$$

Given this, we can rewrite the objective as follows:

$$J(\vec{w}) = C \sum_{i=1}^{N} (\xi_i^+ + \xi_i^-) + + \frac{1}{2} \|\vec{w}\|^2 \tag{14.52}$$

This is a standard quadratic problem in $2N + D + 1$ variables.

14.5.3 Choosing C

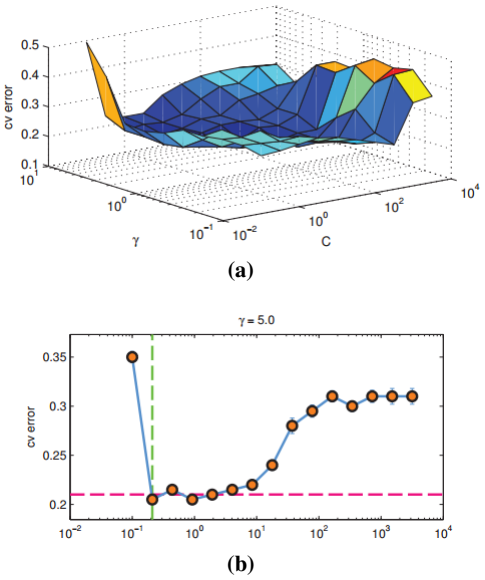
SVMs for both classification and regression require that you specify the kernel function and the parameter C. Typically C is chosen by cross-validation. Note, however, that C interacts quite strongly with the kernel parameters. For example, suppose we are using an RBF kernel with precision $\gamma = \frac{1}{2\sigma^2}$. If $\gamma = 5$, corresponding to narrow kernels, we need heavy regularization, and hence small C(so $\lambda = 1/C$is big). If $\gamma = 1$, a larger value of Cshould be used. So we see that γ and C are tightly coupled. This is illustrated in Figure 14.2, which shows the CV estimate of the 0-1 risk as a function of C and γ.

The authors of libsvm recommend (Hsu et al. 2009) using CV over a 2d grid with values $C \in \{2^{-5}, 2^{-3}, \cdots, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, \cdots, 2^3\}$. In addition, it is important to standardize the data first, for a spherical Gaussian kernel to make sense.

To choose C efficiently, one can develop a path following algorithm in the spirit of lars (Section 13.3.4 TODO). The basic idea is to start with λ large, so that the margin $1/\|\vec{w}(\lambda)\|$ is wide, and hence all points are inside of it and have $\alpha_i = 1$. By slowly decreasing λ, a small set of points will move from inside the margin to outside, and their α_i values will change from 1 to 0, as they cease to be support vectors. When λ is maximal, the function is completely smoothed, and no support vectors remain. See (Hastie et al. 2004) for the details.

14.5.4 A probabilistic interpretation of SVMs

TODO see MLAPP Section 14.5.5



(a)

(b)

Fig. 14.2: (a) A cross validation estimate of the 0-1 error for an SVM classifier with RBF kernel with different precisions $\gamma = 1/(2\sigma^2)$ and different regularizer $\gamma = 1/C$, applied to a synthetic data set drawn from a mixture of 2 Gaussians. (b) A slice through this surface for $\gamma = 5$ The red dotted line is the Bayes optimal error, computed using Bayes rule applied to the model used to generate the data. Based on Figure 12.6 of (Hastie et al. 2009).

14.5.5 Summary of key points

Summarizing the above discussion, we recognize that SVM classifiers involve three key ingredients: the kernel trick, sparsity, and the large margin principle. The kernel trick is necessary to prevent underfitting, i.e., to ensure that the feature vector is sufficiently rich that a linear classifier can separate the data. (Recall from Section 14.2.3 that any Mercer kernel can be viewed as implicitly defining a potentially high dimensional feature vector.) If the original features are already high dimensional (as in many gene expression and text classification problems), it suffices to use a linear kernel, $\kappa(\vec{x},\vec{x}') = \vec{x}^T\vec{x}'$, which is equivalent to working with the original features.

The sparsity and large margin principles are necessary to prevent overfitting, i.e., to ensure that we do not use all the basis functions. These two ideas are closely related to each other, and both arise (in this case) from the use of the hinge loss function. However, there are other methods of achieving sparsity (such as ℓ_1), and also other methods of maximizing the margin(such as boosting). A deeper discussion of this point takes us outside of the scope of this book. See e.g., (Hastie et al. 2009) for more information.

14.6 Comparison of discriminative kernel methods

We have mentioned several different methods for classification and regression based on kernels, which we summarize in Table 14.1. (GP stands for "Gaussian process", which we discuss in Chapter 15 TODO.) The columns have the following meaning:

- Optimize $\vec{w}$: a key question is whether the objective $J(\vec{w} = -\log p(\mathcal{D}|\vec{w}) - \log p(\vec{w}))$ is convex or not. L2VM, L1VM and SVMs have convex objectives. RVMs do not. GPs are Bayesian methods that do not perform parameter estimation.
- Optimize kernel: all the methods require that one "tune" the kernel parameters, such as the bandwidth of the RBF kernel, as well as the level of regularization. For methods based on Gaussians, including L2VM, RVMs and GPs, we can use efficient gradient based optimizers to maximize the marginal likelihood. For SVMs, and L1VM, we must use cross validation, which is slower (see Section 14.5.3).
- Sparse: L1VM, RVMs and SVMs are sparse kernel methods, in that they only use a subset of the training examples. GPs and L2VM are not sparse: they use all the training examples. The principle advantage of sparsity is that prediction at test time is usually faster. In addition, one can sometimes get improved accuracy.
- Probabilistic: All the methods except for SVMs produce probabilistic output of the form $p(y|\vec{x})$. SVMs produce a "confidence" value that can be converted to a probability, but such probabilities are usually very poorly calibrated (see Section 14.5.2.3 TODO).
- Multiclass: All the methods except for SVMs naturally work in the multiclass setting, by using a multinoulli output instead of Bernoulli. The SVM can be made into a multiclass classifier, but there are various difficulties with this approach, as discussed in Section 14.5.2.4 TODO.
- Mercer kernel: SVMs and GPs require that the kernel is positive definite; the other techniques do not.

Method	Opt. $\vec{w}$	Opt.	Sparse	Prob.	Multiclass	Non-Mercer	Section
L2VM	Convex	EB	No	Yes	Yes	Yes	14.3.2
L1VM	Convex	CV	Yes	Yes	Yes	Yes	14.3.2
RVM	Not convex	EB	Yes	Yes	Yes	Yes	14.3.2
SVM	Convex	CV	Yes	No	Indirectly	No	14.5
GP	N/A	EB	No	Yes	Yes	No	15

Table 14.1: Comparison of various kernel based classifiers. EB = empirical Bayes, CV = cross validation. See text for details

14.7 Kernels for building generative models

TODO

Chapter 15
Gaussian processes

15.1 Introduction

In supervised learning, we observe some inputs $\vec{x}_i$ and some outputs y_i. We assume that $y_i = f(\vec{x}_i)$, for some unknown function f, possibly corrupted by noise. The optimal approach is to infer a *distribution over functions* given the data, $p(f|\mathcal{D})$, and then to use this to make predictions given new inputs, i.e., to compute

$$p(y|\vec{x},\mathcal{D}) = \int p(y|f,\vec{x})p(f|\mathcal{D})\mathrm{d}f \tag{15.1}$$

Up until now, we have focussed on parametric representations for the function f, so that instead of inferring $p(f|\mathcal{D})$, we infer $p(\vec{\theta}|\mathcal{D})$. In this chapter, we discuss a way to perform Bayesian inference over functions themselves.

Our approach will be based on **Gaussian processes** or **GP**s. A GP defines a prior over functions, which can be converted into a posterior over functions once we have seen some data.

It turns out that, in the regression setting, all these computations can be done in closed form, in $O(N^3)$ time. (We discuss faster approximations in Section 15.6.) In the classification setting, we must use approximations, such as the Gaussian approximation, since the posterior is no longer exactly Gaussian.

GPs can be thought of as a Bayesian alternative to the kernel methods we discussed in Chapter 14, including L1VM, RVM and SVM.

15.2 GPs for regression

Let the prior on the regression function be a GP, denoted by

$$f(\vec{x}) \sim GP(m(\vec{x}), \kappa(\vec{x},\vec{x}')) \tag{15.2}$$

where $m(\vec{x}$ is the mean function and $\kappa(\vec{x},\vec{x}')$ is the kernel or covariance function, i.e.,

$$m(\vec{x} = \mathbb{E}[f(\vec{x})] \tag{15.3}$$

$$\kappa(\vec{x},\vec{x}') = \mathbb{E}[(f(\vec{x}) - m(\vec{x}))(f(\vec{x}) - m(\vec{x}))^T] \tag{15.4}$$

where κ is a positive definite kernel.

15.3 GPs meet GLMs

15.4 Connection with other methods

15.5 GP latent variable model

15.6 Approximation methods for large datasets

Chapter 16
Adaptive basis function models

16.1 AdaBoost

16.1.1 Representation

$$y = \text{sign}(f(\vec{x})) = \text{sign}\left(\sum_{i=1}^{m} \alpha_m G_m(\vec{x})\right) \tag{16.1}$$

where $G_m(\vec{x})$ are sub classifiers.

16.1.2 Evaluation

$L(y, f(\vec{x})) = \exp[-yf(\vec{x})]$ i.e., exponential loss function

$$(\alpha_m, G_m(x)) = \arg\min_{\alpha,G} \sum_{i=1}^{N} \exp\left[-y_i(f_{m-1}(\vec{x}_i) + \alpha G(\vec{x}_i))\right] \tag{16.2}$$

Define $\bar{w}_{mi} = \exp\left[-y_i(f_{m-1}(\vec{x}_i)\right]$, which is constant w.r.t. α, G

$$(\alpha_m, G_m(x)) = \arg\min_{\alpha,G} \sum_{i=1}^{N} \bar{w}_{mi} \exp\left(-y_i \alpha G(x_i)\right) \tag{16.3}$$

16.1.3 Optimization

16.1.3.1 Input

$$\mathcal{D} = \{(\vec{x}_1, y_1), (\vec{x}_2, y2), \ldots, (\vec{x}_N, y_N)\}$$
$$\text{where } \vec{x}_i \in \mathbb{R}^D,\ y_i \in \{-1, +1\}$$
$$\text{Weak classifiers } \{G_1, G_2, \ldots, G_m\}$$

16.1.3.2 Output

Final classifier: $G(x)$

16.1.3.3 Algorithm

1. Initialize the weights' distribution of training data(when $m = 1$)

$$\mathcal{D}_0 = (w_{11}, w_{12}, \cdots, w_{1n}) = (\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N})$$

2. Iterate over $m = 1, 2, \ldots, M$

   (a) Use training data with current weights' distribution $\mathcal{D}_m$ to get a classifier $G_m(\vec{x})$

   (b) Compute the error rate of $G_m(\vec{x})$ over the training data

$$e_m = P(G_m(\vec{x}_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} \mathbb{I}(G_m(\vec{x}_i) \neq y_i) \tag{16.4}$$

   (c) Compute the coefficient of classifier $G_m(x)$

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} \tag{16.5}$$

   (d) Update the weights' distribution of training data

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(\vec{x}_i)) \tag{16.6}$$

   where Z_m is the normalizing constant

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i G_m(\vec{x}_i)) \tag{16.7}$$

3. Ensemble M weak classifiers

$$G(x) = \text{sign} f(\vec{x}) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(\vec{x})\right] \tag{16.8}$$

16.1.4 The upper bound of the training error of AdaBoost

Theorem 16.1. *The upper bound of the training error of AdaBoost is*

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(G(\vec{x}_i) \neq y_i) \leq \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(\vec{x}_i)) = \prod_{m=1}^{M} Z_m \tag{16.9}$$

Note: the following equation would help proof this theorem

$$w_{mi} \exp(-\alpha_m y_i G_m(\vec{x}_i)) = Z_m w_{m+1,i} \tag{16.10}$$

Chapter 17
Hidden markov Model

17.1 Introduction

17.2 Markov models

Chapter 18
State space models

Chapter 19
Undirected graphical models (Markov random fields)

Chapter 20
Exact inference for graphical models

Chapter 21
Variational inference

Chapter 22
More variational inference

Chapter 23
Monte Carlo inference

Chapter 24
Markov chain Monte Carlo (MCMC)inference

24.1 Introduction

In Chapter 23, we introduced some simple Monte Carlo methods, including rejection sampling and importance sampling. The trouble with these methods is that they do not work well in high dimensional spaces. The most popular method for sampling from high-dimensional distributions is **Markov chain Monte Carlo** or **MCMC**.

The basic idea behind MCMC is to construct a Markov chain (Section 17.2) on the state space $\mathcal{X}$ whose stationary distribution is the target density $p^*(\vec{x})$ of interest (this may be a prior or a posterior). That is, we perform a random walk on the state space, in such a way that the fraction of time we spend in each state $\vec{x}$ is proportional to $p^*(\vec{x})$. By drawing (correlated!) samples $\vec{x}_0, \vec{x}_1, \vec{x}_2, \cdots$ from the chain, we can perform Monte Carlo integration wrt p^*.

24.2 Metropolis Hastings algorithm

24.3 Gibbs sampling

24.4 Speed and accuracy of MCMC

24.5 Auxiliary variable MCMC *

Chapter 25
Clustering

https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/

Chapter 26
Graphical model structure learning

Chapter 27
Latent variable models for discrete data

27.1 Introduction

In this chapter, we are concerned with latent variable models for discrete data, such as bit vectors, sequences of categorical variables, count vectors, graph structures, relational data, etc. These models can be used to analyse voting records, text and document collections, low-intensity images, movie ratings, etc. However, we will mostly focus on text analysis, and this will be reflected in our terminology.

Since we will be dealing with so many different kinds of data, we need some precise notation to keep things clear. When modeling variable-length sequences of categorical variables (i.e., symbols or **tokens**), such as words in a document, we will let $y_{il} \in \{1, \cdots, V\}$ represent the identity of the l'th word in document i,where V is the number of possible words in the vocabulary. We assume $l = 1 : L_i$, where L_i is the (known) length of document i, and $i = 1 : N$, where N is the number of documents.

We will often ignore the word order, resulting in a **bag of words**. This can be reduced to a fixed length vector of counts (a histogram). We will use $n_{iv} \in \{0, 1, \cdots, Li\}$ to denote the number of times word v occurs in document i, for $v = 1 : V$. Note that the $N \times V$ count matrix $\vec{N}$ is often large but sparse, since we typically have many documents, but most words do not occur in any given document.

In some cases, we might have multiple different bags of words, e.g., bags of text words and bags of visual words. These correspond to different "channels" or types of features. We will denote these by y_{irl}, for $r = 1 : R$(the number of responses) and $l = 1 : L_{ir}$. If $L_{ir} = 1$,it means we have a single token (a bag of length 1); in this case, we just write $y_{ir} \in \{1, \cdots, V_r\}$ for brevity. If every channel is just a single token, we write the fixed-size response vector as $y_{i,1:R}$; in this case, the $N \times R$ design matrix $\vec{Y}$ will not be sparse. For example, in social science surveys, y_{ir} could be the response of personito the r'th multi-choice question.

Out goal is to build joint probability models of $p(\vec{y}_i)$ or $p(\vec{n}_i)$ using latent variables to capture the correlations. We will then try to interpret the latent variables, which provide a compressed representation of the data. We provide an overview of some approaches in Section 27.2 TODO, before going into more detail in later sections.

27.2 Distributed state LVMs for discrete data

Chapter 28
Deep learning

Appendix A
Optimization methods

A.1 Convexity

Definition A.1. (**Convex set**) We say a set $\mathcal{S}$ is convex if for any $\vec{x}_1, \vec{x}_2 \in \mathcal{S}$, we have

$$\lambda \vec{x}_1 + (1-\lambda)\vec{x}_2 \in \mathcal{S}, \forall \lambda \in [0,1] \tag{A.1}$$

Definition A.2. (**Convex function**) A function $f(\vec{x})$ is called convex if its **epigraph**(the set of points above the function) defines a convex set. Equivalently, a function $f(\vec{x})$ is called convex if it is defined on a convex set and if, for any $\vec{x}_1, \vec{x}_2 \in \mathcal{S}$, and any $\lambda \in [0,1]$, we have

$$f(\lambda \vec{x}_1 + (1-\lambda)\vec{x}_2) \leq \lambda f(\vec{x}_1) + (1-\lambda) f(\vec{x}_2) \tag{A.2}$$

Definition A.3. A function $f(\vec{x})$ is said to be **strictly convex** if the inequality is strict

$$f(\lambda \vec{x}_1 + (1-\lambda)\vec{x}_2) < \lambda f(\vec{x}_1) + (1-\lambda) f(\vec{x}_2) \tag{A.3}$$

Definition A.4. A function $f(\vec{x})$ is said to be (strictly) **concave** if $-f(\vec{x})$ is (strictly) convex.

Theorem A.1. *If $f(x)$ is twice differentiable on $[a,b]$ and $f''(x) \geq 0$ on $[a,b]$ then $f(x)$ is convex on $[a,b]$.*

Proposition A.1. $\log(x)$ *is strictly convex on* $(0,\infty)$.

Intuitively, a (strictly) convex function has a "bowl shape", and hence has a unique global minimum x^* corresponding to the bottom of the bowl. Hence its second derivative must be positive everywhere, $\frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) > 0$. A twice-continuously differentiable, multivariate function f is convex iff its Hessian is positive definite for all $\vec{x}$. In the machine learning context, the function f often corresponds to the NLL.

Models where the NLL is convex are desirable, since this means we can always find the globally optimal MLE. We will see many examples of this later in the book. However, many models of interest will not have concave likelihoods. In such cases, we will discuss ways to derive locally optimal parameter estimates.

input : Training data $\mathcal{D} = \{(\vec{x}_i, y_i) | i = 1:N\}$
output: A linear model: $y_i = \vec{\theta}^T \vec{x}$
$\vec{w} \leftarrow 0$; $b \leftarrow 0$; $k \leftarrow 0$;
while *no mistakes made within the for loop* **do**
  for $i \leftarrow 1$ **to** N **do**
    if $y_i(\vec{w} \cdot \vec{x}_i + b) \leq 0$ **then**
      $\vec{w} \leftarrow \vec{w} + \eta y_i \vec{x}_i$;
      $b \leftarrow b + \eta y_i$;
      $k \leftarrow k + 1$;
    end
  end
end

Algorithm 5: Stochastic gradient descent

A.2 Gradient descent

A.2.1 Stochastic gradient descent

A.2.2 Batch gradient descent

A.2.3 Line search

The **line search**[1] approach first finds a descent direction along which the objective function f will be reduced and then computes a step size that determines how far $\vec{x}$ should move along that direction. The descent direction can be computed by various methods, such as gradient descent(Section A.2), Newton's method(Section A.4) and Quasi-Newton method(Section A.5). The step size can be determined either exactly or inexactly.

A.2.4 Momentum term

A.3 Lagrange duality

A.3.1 Primal form

Consider the following, which we'll call the **primal** optimization problem:

$$xyz \tag{A.4}$$

[1] http://en.wikipedia.org/wiki/Line_search

A.3.2 Dual form

A.4 Newton's method

$$f(\vec{x}) \approx f(\vec{x}_k) + \vec{g}_k^T(\vec{x}-\vec{x}_k) + \frac{1}{2}(\vec{x}-\vec{x}_k)^T \vec{H}_k(\vec{x}-\vec{x}_k)$$

where $\vec{g}_k \triangleq \vec{g}(\vec{x}_k) = f'(\vec{x}_k), \vec{H}_k \triangleq \vec{H}(\vec{x}_k)$,

$$\vec{H}(\vec{x}) \triangleq \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{D\times D} \quad \text{(Hessian matrix)}$$

$$f'(\vec{x}) = \vec{g}_k + \vec{H}_k(\vec{x}-\vec{x}_k) = 0 \Rightarrow \tag{A.5}$$

$$\vec{x}_{k+1} = \vec{x}_k - \vec{H}_k^{-1}\vec{g}_k \tag{A.6}$$

Initialize $\vec{x}_0$
while *(!convergency)* **do**
  Evaluate $\vec{g}_k = \nabla f(\vec{x}_k)$
  Evaluate $\vec{H}_k = \nabla^2 f(\vec{x}_k)$
  $\vec{d}_k = -\vec{H}_k^{-1}\vec{g}_k$
  Use line search to find step size η_k along $\vec{d}_k$
  $\vec{x}_{k+1} = \vec{x}_k + \eta_k \vec{d}_k$
end

Algorithm 6: Newton's method for minimizing a strictly convex function

A.5 Quasi-Newton method

From Equation A.5 we can infer out the **quasi-Newton condition** as follows:

$$\begin{aligned}
&f'(\vec{x}) - \vec{g}_k = \vec{H}_k(\vec{x}-\vec{x}_k)\\
&\vec{g}_{k-1} - \vec{g}_k = \vec{H}_k(\vec{x}_{k-1}-\vec{x}_k) \Rightarrow\\
&\vec{g}_k - \vec{g}_{k-1} = \vec{H}_k(\vec{x}_k - \vec{x}_{k-1})\\
&\vec{g}_{k+1} - \vec{g}_k = \vec{H}_{k+1}(\vec{x}_{k+1}-\vec{x}_k) \quad \text{(quasi-Newton condition)}
\end{aligned} \tag{A.7}$$

The idea is to replace $\vec{H}_k^{-1}$ with a approximation $\vec{B}_k$, which satisfies the following properties:

1. $\vec{B}_k$ must be symmetric
2. $\vec{B}_k$ must satisfies the quasi-Newton condition, i.e., $\vec{g}_{k+1} - \vec{g}_k = \vec{B}_{k+1}(\vec{x}_{k+1}-\vec{x}_k)$.
   Let $\vec{y}_k = \vec{g}_{k+1} - \vec{g}_k$, $\vec{\delta}_k = \vec{x}_{k+1} - \vec{x}_k$, then
   $$\vec{B}_{k+1}\vec{y}_k = \vec{\delta}_k \tag{A.8}$$
3. Subject to the above, $\vec{B}_k$ should be as close as possible to $\vec{B_{k-1}}$.

Note that we did not require that $\vec{B}_k$ be positive definite. That is because we can show that it must be positive definite if $\vec{B_{k-1}}$ is. Therefore, as long as the initial Hessian approximation $\vec{B}_0$ is positive definite, all $\vec{B}_k$ are, by induction.

A.5.1 DFP

Updating rule:

$$\vec{B}_{k+1} = \vec{B}_k + \vec{P}_k + \vec{Q}_k \tag{A.9}$$

From Equation A.8 we can get

$$\vec{B}_{k+1}\vec{y}_k = \vec{B}_k\vec{y}_k + \vec{P}_k\vec{y}_k + \vec{Q}_k\vec{y}_k = \vec{\delta}_k$$

To make the equation above establish, just let

$$\vec{P}_k\vec{y}_k = \vec{\delta}_k$$
$$\vec{Q}_k\vec{y}_k = -\vec{B}_k\vec{y}_k$$

In DFP algorithm, $\vec{P}_k$ and $\vec{Q}_k$ are

$$\vec{P}_k = \frac{\vec{\delta}_k\vec{\delta}_k^T}{\vec{\delta}_k^T\vec{y}_k} \tag{A.10}$$

$$\vec{Q}_k = -\frac{\vec{B}_k\vec{y}_k\vec{y}_k^T\vec{B}_k}{\vec{y}_k^T\vec{B}_k\vec{y}_k} \tag{A.11}$$

A.5.2 BFGS

Use $\vec{B}_k$ as a approximation to $\vec{H}_k$, then the quasi-Newton condition becomes

$$\vec{B}_{k+1}\vec{\delta}_k = \vec{y}_k \tag{A.12}$$

The updating rule is similar to DFP, but $\vec{P}_k$ and $\vec{Q}_k$ are different. Let

$$\vec{P}_k\vec{\delta}_k = \vec{y}_k$$
$$\vec{Q}_k\vec{\delta}_k = -\vec{B}_k\vec{\delta}_k$$

Then

$$\vec{P}_k = \frac{\vec{y}_k\vec{y}_k^T}{\vec{y}_k^T\vec{\delta}_k} \tag{A.13}$$

$$\vec{Q}_k = -\frac{\vec{B}_k\vec{\delta}_k\vec{\delta}_k^T\vec{B}_k}{\vec{\delta}_k^T\vec{B}_k\vec{\delta}_k} \tag{A.14}$$

A.5.3 Broyden

Broyden's algorithm is a linear combination of DFP and BFGS.

Glossary

feature vector A feature vector to represent one data.

loss function a function that maps an event onto a real number intuitively representing some ”cost” associated with the event.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.