Machine Learning Notebook

Cong Bao

Contents

Chapter 1

Introduction

1.1 About this Notebook

1.2 Policy of Use

Chapter 2

Mathematics Basics

2.1 Probability

2.1.1 Basic Rules

Three Axioms of Probability Let Ω be a sample space. A probability assigns a real number $P(X)$ to each event $X \subseteq \Omega$ in such a way that

1. $P(X) \geq 0, \forall X$
2. If $X_1, X_2, \ldots$ are pairwise disjoint events ($X_1 \cap X_2 = \emptyset,\ i \neq j,\ i, j = 1, 2, \ldots$), then $P\left(\bigcup_{i=1}^{\infty} X_i\right) = \sum_{i=1}^{\infty} P(X_i)$. (This property is called countable additivity.)
3. $P(\Omega) = 1$

Joint Probability The probability both event A and B occur. $P(X, Y) = P(X \cap Y)$.

Marginalization The probability distribution of any variable in a joint distribution can be recovered by integrating (or summing) over the other variables.

1. For continuous r.v. $P(x) = \int P(x, y)\, dy$; $P(y) = \int P(x, y)\, dx$.
2. For discrete r.v. $P(x) = \sum_y P(x, y)$; $P(y) = \sum_x P(x, y)$.
3. For mixed r.v. $P(x, y) = \sum_w \int P(w, x, y, z)\, dz$, where w is discrete and z is continuous.

Conditional Probibility $P(X = x|Y = y)$ is the probability $X = x$ occurs given the knowledge $Y = y$ occurs. Conditional probability can be extracted from joint probability

that

$$P(x|y=y^*) = \frac{P(x, y=y^*)}{\int P(x, y=y^*)\,dx} = \frac{P(x, y=y^*)}{P(y=y^*)}$$

Usually, the formula is written as $P(x|y) = \frac{P(x,y)}{P(y)}$.

Product Rule The formula can be rearranged as $P(x, y) = P(x|y)\,P(y) = P(y|x)\,P(x)$. In case of multiple variables

$$\begin{aligned} P(w, x, y, z) &= P(w, x, y|z)\,P(z) \\ &= P(w, x|y, z)\,P(y|z)\,P(z) \\ &= P(w|x, y, z)\,P(x|y, z)\,P(y|z)\,P(z) \end{aligned}$$

Independence If two variables x and y are independent, then r.v. x tells nothing about r.v. y (and vice-versa)

$$\begin{aligned} P(x|y) &= P(x) \\ P(y|x) &= P(y) \\ P(x, y) &= P(x)\,P(y) \end{aligned}$$

Baye's Rule By rearranging formula in Product Rule, we have

$$\begin{aligned} P(y|x) &= \frac{P(x|y)\,P(y)}{P(x)} \\ &= \frac{P(x|y)\,P(y)}{\int P(x, y)\,dy} \\ &= \frac{P(x|y)\,P(y)}{\int P(x|y)\,P(y)\,dy} \end{aligned}$$

Expectation Expectation tells us the expected or average value of some function $f(x)$, taking into account the distribution of x.

$$\begin{aligned} \mathbf{E}[f(x)] &= \sum_x f(x)P(x) \\ \mathbf{E}[f(x)] &= \int f(x)P(x)\,dx \end{aligned}$$

Definition in two dimensions: $\mathbf{E}[f(x, y)] = \iint f(x, y)P(x, y)\,dx\;dy$

Besides, Expectation has the following four rules

Function $f(\bullet)$	Expectation
x^k	k^{th} moment about zero
$(x-\mu_x)^k$	k^{th} moment about the mean

Function $f(\bullet)$	Expectation
x	mean, μ_x
$(x-\mu_x)^2$	variance
$(x-\mu_x)^3$	skew
$(x-\mu_x)^4$	kurtosis
$(x-\mu_x)(x-\mu_y)$	covariance of x and y

1. Expected value of a constant is the constant $\mathbf{E}[\kappa] = \kappa$.
2. Expected value of constant times function is constant times excepted value of function $\mathbf{E}[kf(x)] = k\mathbf{E}[f(x)]$.
3. Expectation of sum of functions is sum of expectation of functions $\mathbf{E}[f(x) + g(y)] = \mathbf{E}[f(x)] + \mathbf{E}[g(x)]$.
4. Expectation of product of functions in variables x and y is product of expectations of functions if x and y are independent $\mathbf{E}[f(x)g(y)] = \mathbf{E}[f(x)]\mathbf{E}[g(y)], x \perp\!\!\!\perp y$.

2.1.2 Common Probability Distributions

Bernoulli Bernoulli distribution describes situation where only two possible outcomes $y = 0/y = 1$ or failure/success exist.

1. $P(x) = \mathbf{Bern}_x[\lambda] = \lambda^x(1-\lambda)^{1-x}$
2. $x \in \{0, 1\}$; $\lambda \in [0, 1]$
3. $\mathbf{E}[x] = \lambda$, $\mathbf{Var}[x] = \lambda(1-\lambda)$

Binomial Binomial distribution describes n independent Bernoulli trials.

1. $P(x) = \mathbf{Bin}_x[n, \lambda] = \binom{n}{x}\lambda^x(1-\lambda)^{1-x}$
2. $x \in \mathbb{N}$; $n \in \mathbb{N}$, $\lambda \in [0, 1]$
3. $\mathbf{E}[x] = n\lambda$, $\mathbf{Var}[x] = n\lambda(1-\lambda)$

Geometric Geometric distribution describes the number of independent Bernoulli trails until we record the first success.

1. $P(x) = \mathbf{Geom}_x[\lambda] = \lambda(1-\lambda)^{x-1}$
2. $x \in \mathbb{Z}_+$; $\lambda \in [0, 1]$
3. $\mathbf{E}[x] = \frac{1}{\lambda}$, $\mathbf{Var}[x] = \frac{1-\lambda}{\lambda^2}$

Negative Binomial Negative Binomial distribution describes the number of independent Bernoulli trails until we record the r^{th} success.

1. $P(x) = \mathbf{NegBin}_x[r, \lambda] = \binom{x+r-1}{x}\lambda^x(1-\lambda)^r$
2. $x \in \mathbb{N}$; $r \in \mathbb{Z}_+$, $\lambda \in [0, 1]$
3. $\mathbf{E}[x] = \frac{r\lambda}{1-\lambda}$, $\mathbf{Var}[x] = \frac{r\lambda}{(1-\lambda)^2}$

Beta Beta distribution is a common conjugate prior to Bernoulli, Binomial, Geometric, and Negative Binomial distributions.

1. $P(\lambda) = \mathbf{Beta}_\lambda[\alpha, \beta] = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\lambda^{\alpha-1}(1-\lambda)^{\beta-1}$
2. $\lambda \in \mathbb{R}$; $\alpha \in \mathbb{R}_+$, $\beta \in \mathbb{R}_+$
3. $\mathbf{E}[\lambda] = \frac{\alpha}{\alpha+\beta}$, $\mathbf{Var}[\lambda] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

Poisson Poisson distribution describes the rate μ an event takes place rarely on an interval or surface over time or space.

1. $P(x) = \mathbf{Poi}_x[\lambda] = \frac{\lambda^x}{x!}e^{-\lambda}$
2. $x \in \mathbb{N}$; $\lambda \in \mathbb{R}_+$
3. $\mathbf{E}[x] = \lambda$, $\mathbf{Var}[x] = \lambda$

Exponential Exponential distribution describes the continuous time between events in Poisson process.

1. $P(x) = \mathbf{Exp}_x[\lambda] = \lambda e^{-\lambda x}$
2. $x \in \mathbb{N}$; $\lambda \in \mathbb{Z}_+$
3. $\mathbf{E}[x] = \frac{1}{\lambda}$, $\mathbf{Var}[x] = \frac{1}{\lambda^2}$

Gamma Gamma distribution describes the continuous time until the α^{th} event in Poisson process takes place. It is also a common conjugate prior to Poisson and Exponential distributions.

1. $P(\lambda) = \mathbf{Gamma}_\lambda[\alpha, \beta] = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}$
2. $\lambda \in \mathbb{Z}_+$; $\alpha \in \mathbb{Z}_+$, $\beta \in \mathbb{Z}_+$
3. $\mathbf{E}[\lambda] = \frac{\alpha}{\beta}$, $\mathbf{Var}[\lambda] = \frac{\alpha}{\beta^2}$

	Distribution	
Random Variables	Discrete time	continuous time
No. of events	Binomial	Poisson
Time till first event	Geometric	Exponential
Time till r^{th} event	Negative Binomial	Gamma

Categorical Categorical distribution describes situation with K possible outcomes.

1. $P(x) = \mathbf{Cat}_x[\boldsymbol{\lambda}]$, $P(x = k) = \lambda_k$, $P(\boldsymbol{x} = \boldsymbol{e}_k) = \prod_{j=1}^{K} \lambda_j^{x_j} = \lambda_k$
2. $x \in \{1, 2, \ldots, K\}$; $\lambda_k \in [0, 1]$ where $\sum_k \lambda_k = 1$
3. $\mathbf{E}[x_i] = \lambda_i$, $\mathbf{Var}[x_i] = \lambda_i(1 - \lambda_i)$, $\mathbf{Cov}[x_i, x_j] = -\lambda_i\lambda_j$ $(i \neq j)$

Dirichlet Dirichlet distribution is a common conjugate prior to Categorical distribution.

1. $P(\boldsymbol{\lambda}) = \mathbf{Dir}_{\boldsymbol{\lambda}}[\boldsymbol{\alpha}] = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \lambda_k^{\alpha_k - 1}$
2. $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_K]^\top$, $\lambda_k \in [0, 1]$, $\sum_{k=1}^{K} \lambda_k = 1$; $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_K]$, $\alpha_k \in \mathbb{R}_+$
3. $\mathbf{E}[\lambda_i] = \frac{\alpha_i}{\sum_k \alpha_k}$, $\mathbf{Var}[\lambda_i] = \frac{\alpha_i(\sum_k \alpha_k - \alpha_i)}{(\sum_k \alpha_k)^2(\sum_k \alpha_k + 1)}$, $\mathbf{Cov}[\lambda_i, \lambda_j] = \frac{-\alpha_i\alpha_j}{(\sum_k \alpha_k)^2(\sum_k \alpha_k + 1)}$ $(i \neq j)$

Univariate Normal Univariate normal distribution describes single continuous variable.

1. $P(x) = \mathbf{Norm}_x[\mu, \sigma^2] = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
2. $x \in \mathbb{R}$; $\mu \in \mathbb{R}$, $\sigma^2 \in \mathbb{R}_+$
3. $\mathbf{E}[x] = \mu$, $\mathbf{Var}[x] = \sigma^2$

Normal Inverse Gamma Normal inverse gamma distribution is a common conjugate prior to Univariate Normal distribution.

1. $P(\mu, \sigma^2) = \mathbf{NormInvGam}_{\mu,\sigma^2}[\alpha, \beta, \gamma, \delta] = \frac{\sqrt{\gamma}}{\sqrt{2\pi\sigma^2}} \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} exp\left(-\frac{2\beta + \gamma(\delta-\mu)^2}{2\sigma^2}\right)$
2. $\mu \in \mathbb{R}$, $\sigma^2 \in \mathbb{R}_+$; $\alpha \in \mathbb{R}_+$, $\beta \in \mathbb{R}_+$, $\gamma \in \mathbb{R}_+$, $\delta \in \mathbb{R}$
3. $\mathbf{E}[\mu] = \delta$, $\mathbf{E}[\sigma^2] = \frac{\beta}{\alpha - 1}$ $(\alpha > 1)$, $\mathbf{Var}[\mu] = \frac{\beta}{(\alpha-1)\gamma}$ $(\alpha > 1)$, $\mathbf{Var}[\sigma^2] = \frac{\beta^2}{(\alpha-1)^2(\alpha-2)}$ $(\alpha > 2)$, $\mathbf{Cov}[\mu, \sigma^2] = 0$ $(\alpha > 1)$

Multivariate Normal Multivariate normal distribution describes multiple continuous variables. It takes two parameters: a vector containing mean position $\boldsymbol{\mu}$, and a symmetric positive definite covariance matrix $\boldsymbol{\Sigma}$.

1. $P(\boldsymbol{x}) = \mathbf{Norm}_{\boldsymbol{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}] = \frac{1}{\sqrt{det(2\pi\boldsymbol{\Sigma})}} exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$
2. $\boldsymbol{x} \in \mathbb{R}^K$; $\boldsymbol{\mu} \in \mathbb{R}^K$, $\boldsymbol{\Sigma} \in \mathbb{R}^{K\times K}$ (positive semi-definite matrix)
3. $\mathbf{E}[\boldsymbol{x}] = \boldsymbol{\mu}$, $\mathbf{Var}[\boldsymbol{x}] = \boldsymbol{\Sigma}$

Normal Inverse Wishart Normal inverse wishart distribution is a common conjugate prior to Multivariate Normal distribution.

1. $P(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbf{NormInvWis}_{\boldsymbol{\mu},\boldsymbol{\Sigma}}[\alpha, \boldsymbol{\Psi}, \gamma, \boldsymbol{\delta}]$
   $= \frac{\gamma^{D/2}|\boldsymbol{\Psi}|^{\alpha/2}|\boldsymbol{\Sigma}|^{-\frac{\alpha+D+2}{2}}}{(2\pi)^{D/2}2^{(\alpha\boldsymbol{\Sigma})/2}\Gamma_D(\alpha/2)} exp\left(-\frac{1}{2}(\mathrm{Tr}(\boldsymbol{\Psi}\boldsymbol{\Sigma}^{-1}) + \gamma(\boldsymbol{\mu}-\boldsymbol{\delta})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\delta})\right)$
2. $\boldsymbol{\mu} \in \mathbb{R}^K$, $\boldsymbol{\Sigma} \in \mathbb{R}^{K\times K}$; $\alpha \in \mathbb{R}_{>D-1}$, $\boldsymbol{\Psi} \in \mathbb{R}^{K\times K}$, $\gamma \in \mathbb{R}_+$, $\boldsymbol{\delta} \in \mathbb{R}^K$

2.2 Linear Algebra

2.2.1 Vectors

Vectors Addition

$$\boldsymbol{v} + \boldsymbol{w} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} v_1 + w_1 \\ v_2 + w_2 \\ \vdots \\ v_n + w_n \end{pmatrix}$$

Vectors Scaling

$$a\boldsymbol{v} = a\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} av_1 \\ av_2 \\ \vdots \\ av_n \end{pmatrix}$$

Rules for Vectors Addition and Scaling

1. $\boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w}$
2. $\boldsymbol{v} + \boldsymbol{w} = \boldsymbol{w} + \boldsymbol{v}$

3. There is a vector $\mathbf{0}$ such that $\mathbf{0} + \boldsymbol{v} = \boldsymbol{v}$ for all $\boldsymbol{v}$
4. For every vector $\boldsymbol{v}$ there is a vector $-\boldsymbol{v}$ so that $\boldsymbol{v} + (-\boldsymbol{v}) = \mathbf{0}$
5. $a\,(b\boldsymbol{v}) = (ab)\,\boldsymbol{v}$
6. $1\boldsymbol{v} = \boldsymbol{v}$
7. $a\,(\boldsymbol{v} + \boldsymbol{w}) = a\boldsymbol{v} + a\boldsymbol{w}$
8. $(a + b)\,\boldsymbol{v} = a\boldsymbol{v} + b\boldsymbol{v}$

Linear Combination & Span Linear combination (e.g. in 2D space):

$$a\boldsymbol{v} + b\boldsymbol{w}$$

The span of $\boldsymbol{v}$ and $\boldsymbol{w}$ is the set of all their linear combinations.

Representation of Basis In Euclidean:

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = v_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + v_2 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + v_n \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

We can write this as

$$\boldsymbol{v} = v_1\boldsymbol{e}^1 + v_2\boldsymbol{e}^2 + \cdots + v_n\boldsymbol{e}^n$$

In different basis, we choose other basis vector and then write the same vector

$$\boldsymbol{v} = w_1\boldsymbol{b}^1 + w_2\boldsymbol{b}^2 + \cdots + w_n\boldsymbol{b}^n$$

If these basis vectors are orthonormal, $w_i = \boldsymbol{v}^\top \boldsymbol{b}^i$

Linear Dependence

1. Linearly dependent: A set of vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n$ is linearly dependent if there exists a vector $\boldsymbol{v}^j$ that can be expressed as a linear combination of the other vectors. (The vector $\boldsymbol{v}^j$ is already located in the span of other vectors)
2. Linearly Independent: Each vector really does add another dimension to the span. And the only solution to $\sum_{i=1}^{n} a_i \boldsymbol{v}^i = \mathbf{0}$ is for all $a_i = 0, i = 1, \ldots, n$.

Dot Products

$$\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{n} v_i w_i = \boldsymbol{v}^\top \boldsymbol{w}$$

The length of a vector is denoted as $||\boldsymbol{v}||$, the squared length is given by

$$||\boldsymbol{v}||^2 = \boldsymbol{v}^\top \boldsymbol{v} = \boldsymbol{v}^2 = v_1^2 + v_2^2 + \cdots + v_n^2$$

A natural geometric interpretation of dot products is

$$\boldsymbol{v} \cdot \boldsymbol{w} = ||\boldsymbol{v}||\,||\boldsymbol{w}|| \cos\theta$$

where θ is the angle between two vectors.

2.2.2 Matrices (Square Matrices)

Linear Transformation

$$\boldsymbol{v}' = (\mathbf{M}_n \ldots \mathbf{M}_2 \mathbf{M}_1)\, \boldsymbol{v}$$

Linear transformation always maps linear subspaces onto linear subspaces (possibly of a lower dimension). Intuitively, linear transformation keeps the grid lines stay parallel and evenly spaced, and so that the origin remains fixed.

Basic Formulae

1. $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$
2. $(\mathbf{A} + \mathbf{B})^\top = \mathbf{A}^\top + \mathbf{B}^\top$
3. $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$

   if individual inverses exist:

4. $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
5. $(\mathbf{A}^{-1})^\top = (\mathbf{A}^\top)^{-1}$

Trace Trace is the sum of diagonal elements of matrix $\mathbf{M}$,

$$\mathrm{Tr}(\mathbf{M}) = \mathrm{Tr}(\mathbf{M}^\top) = \sum_{i=1}^{n} \mathbf{M}_{ii} = \sum \text{eigenvalues of } \mathbf{M}$$

Cyclic permutations in trace,

$$\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{CAB}) = \text{Tr}(\mathbf{BCA})$$

Determinants The determinant of a matrix $\mathbf{M}$, denoted as $|\mathbf{M}|$, is a function mapping matrices to scalars, measuring how much multiplication by the matrix expands or contracts space. If the determinant is 0, then space is contracted completely along at least one dimension, causing it to lose all of its volume. If the determinant is 1, then the transformation preserves volume.

Some properties of determinant:

1. $|\mathbf{M}| = \prod$ eigenvalues of $\mathbf{M}$
2. $|\mathbf{AB}| = |\mathbf{A}|\,|\mathbf{B}|$
3. $|a| = a$
4. $|a\mathbf{M}| = a^n\,|\mathbf{M}|$
5. $|\mathbf{M}^{-1}| = |\mathbf{M}|^{-1}$

Symmetric Matrix

$$\mathbf{M} = \mathbf{M}^\top$$

Orthogonal Matrix

$$\mathbf{M}^\top\mathbf{M} = \mathbf{M}\mathbf{M}^\top = \mathbf{I}$$
$$\mathbf{M}^{-1} = \mathbf{M}^\top$$

Eigendecomposition If a matrix $\mathbf{M}$ satisfies

$$\mathbf{M}\boldsymbol{v} = \lambda\boldsymbol{v}$$

then $\boldsymbol{v}$ is the ***eigenvector*** of $\mathbf{M}$, and λ is the ***eigenvalue*** of $\mathbf{M}$ (The vector $\boldsymbol{v}$ is just scaled by value λ after a linear transformation $\mathbf{M}$). To solve the equation, we can rewrite it as

$$(\mathbf{M} - \lambda\mathbf{I})\boldsymbol{v} = \mathbf{0}$$

When $|\mathbf{M} - \lambda\mathbf{I}| = 0$, it means at least one dimension is contracted by the linear transformation $\mathbf{M} - \lambda\mathbf{I}$ so that there exists at least one non-zero vector to be transformed to zero, which gives us a solution to λ.

Suppose matrix $\mathbf{M}$ has n linearly independent eigenvectors, $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}\}$, with corresponding eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$. We may concatenate all of the eigenvectors to form a matrix $\mathbf{V}$ with one eigenvector per column: $\mathbf{V} = [\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}]$. Likewise, we can concatenate the eigenvalues to form a vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]$. The eigendecomposition of $\mathbf{M}$ is then given by

$$\mathbf{M} = \mathbf{V}\,\mathrm{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1}$$

Specifically, every real symmetric matrix can be decomposed into an expression using only real-valued eigenvectors and eigenvalues:

$$\mathbf{M} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\top}$$

where $\mathbf{Q}$ is an orthogonal matrix composed of eigenvectors of $\mathbf{M}$, and $\boldsymbol{\Lambda}$ is a diagonal matrix.

A matrix is ***singular*** if and only if any of the eigenvalues are zero ($|\mathbf{M}| = 0$). A matrix whose eigenvalues are all positive is called ***positive definite***. A matrix whose eigenvalues are all positive or zero-valued (non-negative) is called ***positive semi-definite***. Likewise, if all eigenvalues are negative, the matrix is ***negative definite***, and if all eigenvalues are negative or zero-valued (non-positive), it is ***negative semi-definite***. Positive semi-definite matrices guarantee that $\forall \boldsymbol{v}, \boldsymbol{v}^{\top}\mathbf{M}\boldsymbol{v} \geq 0$. Positive definite matrices additionally guarantee that $\boldsymbol{v}^{\top}\mathbf{M}\boldsymbol{v} = 0 \Rightarrow \boldsymbol{v} = \mathbf{0}$.

Singular Value Decomposition (SVD) The singular value decomposition of a $m \times n$ matrix $\mathbf{M}$ can be formed as

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\top}$$

where $\mathbf{U}$ is a $m \times m$ orthogonal matrix, $\boldsymbol{\Sigma}$ is a $m \times n$ diagonal matrix, and $\mathbf{V}$ is a $n \times n$ orthogonal matrix. The elements along the diagonal of $\boldsymbol{\Sigma}$ are known as the ***singular values*** of the matrix $\mathbf{M}$. The columns of $\mathbf{U}$ are known as the ***left-singular vectors***. The columns of $\mathbf{V}$ are known as as the ***right-singular vectors***.

There are relations between SVD and eigendecomposition. According to the definition

of SVD, we have

$$\mathbf{M}^\top\mathbf{M} = \mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma})\mathbf{V}^\top$$
$$\mathbf{M}\mathbf{M}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top = \mathbf{U}(\mathbf{\Sigma}\mathbf{\Sigma}^\top)\mathbf{U}^\top$$

Hence, we can conclude that

1. The right-singular vectors of $\mathbf{M}$ are the eigenvectors of $\mathbf{M}^\top\mathbf{M}$.
2. The left-singular vectors of $\mathbf{M}$ are the eigenvectors of $\mathbf{M}\mathbf{M}^\top$.
3. The non-zero singular values of $\mathbf{M}$ are the square roots of the eigenvalues of $\mathbf{M}^\top\mathbf{M}$ or $\mathbf{M}\mathbf{M}^\top$.

Moore-Penrose Pseudoinverse SVD can be applied to compute the pseudoinverse of a matrix $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, given by

$$\mathbf{M}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^\top$$

where $\mathbf{\Sigma}^+$ is the pseudoinverse of $\mathbf{\Sigma}$, which is formed by replacing every non-zero diagonal entry by its reciprocal and transposing the resulting matrix.

2.3 Calculus

2.3.1 Differentiation

Basic Formulae Here, $f(x)$ and $g(x)$ are differentiable functions (the derivative exists), c and n are any real numbers.

1. $(cf)' = cf'(x)$
2. $(f \pm g)' = f'(x) \pm g'(x)$
3. $(fg)' = f'g + fg'$ (Product Rule)
4. $(\frac{f}{g})' = \frac{f'g - fg'}{g^2}$ (Quotient Rule)
5. $\frac{d}{dx}(c) = 0$
6. $\frac{d}{dx}(x^n) = nx^{n-1}$ (Power Rule)
7. $\frac{d}{dx}(f(g(x))) = f'(g(x))g'(x)$ (Chain Rule)

Mean Value Theorem If $f(x)$ is continuous on the closed interval $[a, b]$ and differentiable on the open interval (a, b) then there is a number $a < c < b$ such that $f'(c) = \frac{f(b)-f(a)}{b-a}$.

Newton's Method If x_n is the n^{th} guess for the root/solution of $f(x) = 0$ then $(n+1)^{\text{st}}$ guess is $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ provided $f'(x_n)$ exists.

Taylor Series The Taylor series of a real or complex-valued function $f(x)$ that is infinitely differentiable at a real or complex number a is the power series

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n$$

where $n!$ denotes the factorial of n and $f^n(a)$ denotes the n^{th} derivative of f evaluated at the point a. Sometimes, we can represent it as

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \cdots + \frac{h^{r-1}}{(r-1)!}f^{(r-1)}(x) + O\left(h^r\right)$$

If we set the function as $f(\boldsymbol{x} + h\boldsymbol{v})$ for some vector $\boldsymbol{v}$, then

$$f(\boldsymbol{x} + h\boldsymbol{v}) = f(\boldsymbol{x}) + h\nabla f^\top \boldsymbol{v} + \frac{h^2}{2}\boldsymbol{v}^\top \mathbf{H}_f \boldsymbol{v} + O\left(h^3\right)$$

where ∇f is the gradient vector and H_f is the Hessian matrix.

2.3.2 Integration

Properties

1. $\int f(x) \pm g(x)\ dx = \int f(x)\ dx \pm \int g(x)\ dx$
2. $\int_a^b f(x) \pm g(x)\ dx = \int_a^b f(x)\ dx \pm \int_a^b g(x)\ dx$
3. $\int_a^a f(x)\ dx = 0$
4. $\int_a^b = -\int_b^a f(x)\ dx$
5. $\int_a^b f(x)\ dx = \int_a^c f(x)\ dx + \int_c^b f(x)\ dx$
6. $\int kf(x)\ dx = k\int f(x)\ dx$
7. $\int_a^b kf(x)\ dx = k\int_a^b f(x)\ dx$
8. $\int_a^b k\ dx = k(b-a)$

9. $\left|\int_a^b f(x)\ dx\right| \leq \int_a^b |f(x)|\ dx$
10. If $f(x) \geq g(x)$ on $a \leq x \leq b$ then $\int_a^b f(x)\ dx \geq \int_a^b g(x)\ dx$
11. If $f(x) \geq 0$ on $a \leq x \leq b$ then $\int_a^b f(x)\ dx \geq 0$
12. If $m \leq f(x) \leq M$ on $a \leq x \leq b$ then $m(b-a) \leq \int_a^b f(x)\ dx \leq M(b-a)$

Average Function Value The average value of $f(x)$ on $a \leq x \leq b$ is

$$f_{avg} = \frac{1}{b-a}\int_a^b f(x)\ dx$$

Jensen's Inequality If φ is a convex function,

$$\varphi\left(\frac{1}{b-a}\int_a^b f(x)\ dx\right) \leq \frac{1}{b-a}\int_a^b \varphi\left(f(x)\right)\ dx$$

Additionally, if X is an integrable real-valued random variable, then

$$\varphi\left(\mathbf{E}\left[X\right]\right) \leq \mathbf{E}\left[\varphi(X)\right]$$

2.3.3 Multivariate Calculus

Partial Derivatives Partial derivatives are simply holding all other variables constant (and act like constants for the derivative) and only taking the derivative with respect to a given variable.

$$f_x(x,y) = f_x = \frac{\partial f}{\partial x} = \frac{\partial}{\partial x}f(x,y) = D_x f$$

Clairaut's Theorem If the function f_{xy} and f_{yx} are both continuous, then $f_{xy}(a,b) = f_{yx}(a,b)$.

Chain Rule If $z = f(x,y)$, $x = g(t)$, and $y = h(t)$, then

$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\frac{dx}{dt} + \frac{\partial z}{\partial y}\frac{dy}{dt}$$

Gradient

$$\nabla f(x,y) = \left[f_x(x,y)\ f_y(x,y)\right]^\top$$

Directional Derivative

$$D_{\boldsymbol{u}} f(x, y) = \nabla f(x, y) \cdot \boldsymbol{u}$$

where $\boldsymbol{u}$ is an unit vector. The maximum value of the directional derivative $D_{\boldsymbol{u}} f(x, y)$ is $|\nabla f(x, y)|$, and it occurs when $\boldsymbol{u}$ has the same direction as the gradient vector $\nabla f(x, y)$.

Lagrange Multipliers To find the maximum and minimum values of $f(x, y, z)$ subject to the constraint $g(x, y, z) = k$,

1. Find all values of x, y, z and λ such that

$$\begin{aligned} \nabla f(x, y, z) &= \lambda \nabla g(x, y, z) \\ g(x, y, z) &= k \end{aligned}$$

2. Evaluate f at all of these points. The largest is the maximum value, and the smallest is the minimum value of f subject to the constraint g.

2.3.4 Matrix Calculus

Vector by Scalar The derivative of a vector $\boldsymbol{y} = [y_1 \ y_2 \ \cdots \ y_n]^\top$, by a scalar x:

$$\frac{\partial \boldsymbol{y}}{\partial x} = \begin{bmatrix} \dfrac{\partial y_1}{\partial x} & \dfrac{\partial y_2}{\partial x} & \cdots & \dfrac{\partial y_n}{\partial x} \end{bmatrix}^\top$$

Scalar by Vector (Gradient) The derivative of a scalar y, by a vector $\boldsymbol{x} = [x_1 \ x_2 \ \cdots \ x_n]^\top$:

$$\frac{\partial y}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial y}{\partial x_1} & \dfrac{\partial y}{\partial x_2} & \cdots & \dfrac{\partial y}{\partial x_n} \end{bmatrix}^\top$$

Vector by Vector (Jacobian Matrix) The derivative of a vector $\boldsymbol{y} = [y_1 \ y_2 \ \cdots \ y_m]^\top$, by another vector $\boldsymbol{x} = [x_1 \ x_2 \ \cdots \ x_n]^\top$:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Matrix by Scalar The derivative of a matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$, by a scalar x:

$$\frac{\partial \mathbf{Y}}{\partial x} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x} & \frac{\partial y_{12}}{\partial x} & \cdots & \frac{\partial y_{1n}}{\partial x} \\ \frac{\partial y_{21}}{\partial x} & \frac{\partial y_{22}}{\partial x} & \cdots & \frac{\partial y_{2n}}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{m1}}{\partial x} & \frac{\partial y_{m2}}{\partial x} & \cdots & \frac{\partial y_{mn}}{\partial x} \end{bmatrix}$$

Scalar by Matrix (Gradient Matrix) The derivative of a scalar y, by a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$:

$$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{21}} & \cdots & \frac{\partial y}{\partial x_{m1}} \\ \frac{\partial y}{\partial x_{12}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{m2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{1n}} & \frac{\partial y}{\partial x_{2n}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix}$$

2.3.5 Backpropagation Modules

Backpropagation An application of chain rule. For a simple nested function: $y = f(g(x))$:

$$\frac{dy}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

For multivariate function $y = f\left(g^{(1)}(x), \ldots, g^{(n)}(x)\right)$:

$$\frac{\partial y}{\partial x} = \sum_{i=1}^{n} \frac{\partial f}{\partial g^{(i)}} \frac{\partial g^{(i)}}{\partial x}$$

Linear Module

1. Forward pass: $\boldsymbol{y} = \mathbf{W}\boldsymbol{x} + \boldsymbol{b}$
2. Backward pass: $\partial L/\partial \boldsymbol{x} = (\partial L/\partial \boldsymbol{y})\,\mathbf{W}$
3. Parameter gradient: $\partial L/\partial \mathbf{W} = (\partial L/\partial \boldsymbol{y})^{\top}\boldsymbol{x}^{\top}$, $\partial L/\partial \boldsymbol{b} = \partial L/\partial \boldsymbol{y}$

ReLU Module

1. Forward pass: $\boldsymbol{y} = relu(\boldsymbol{x})$, $y_i = max(0, x_i)$
2. backward pass: $\partial L/\partial x_i = (y_i > 0)\,(\partial L/\partial y_i)$

Softmax Module

1. Forward pass: $\boldsymbol{y} = softmax(\boldsymbol{x})$, $y_n = e^{x_n} / \sum_m e^{x_m}$
2. Backward pass: $\partial L / \partial \boldsymbol{x} = \boldsymbol{s} - \boldsymbol{y} \sum_i s_i$, where $s_i = (\partial L / \partial y_i)\, y_i$

Cross-entropy Module

1. Forward pass: $y = L = crossentropy(\boldsymbol{p}, \boldsymbol{x})$, $y = -\sum_i^C p_i \log x_i$
2. Backward pass: $\partial L / \partial x_i = -p_i / x_i$

Cross-entropy with Logits Module

1. Forward pass: $y = L = crossentropy(\boldsymbol{p}, softmax(\boldsymbol{x}))$, $y = -\sum_i^C p_i \log \left(e^{x_i} / \sum_m e^{x_m}\right)$
2. Backward pass: $\partial L / \partial \boldsymbol{x} = \boldsymbol{y} - \boldsymbol{p}$

2.4 Information Theory

2.4.1 Self-information

Properties of information quantification:

1. Likely events should have low information content, and in the extreme case, events that are guaranteed to happen should have no information content whatsoever.
2. Less likely events should have higher information content.
3. Independent events should have additive information.

In order to satisfy all three of these properties, we define the self-information of an event $\mathtt{x} = x$ to be $I(x) = -\log P(x)$.

2.4.2 Entropy

Self-information deals only with a single outcome. We can quantify the amount of uncertainty in an entire probability distribution using the Shannon entropy:

$$H(x) = \mathbf{E}_{x \sim P}[I(x)] = -\mathbf{E}_{x \sim P}[\log P(x)] = -\sum_i p_i \log p_i$$

2.4.3 Kullback-Leibler (KL) Divergence

If we have two separate probability distributions $P(x)$ and $Q(x)$ over the same random variable x, we can measure how different these two distribution are using the Kullback-Leibler (KL) divergence:

$$KL(P\|Q) = \mathbf{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbf{E}_{x\sim P}\left[\log P(x) - \log Q(x)\right]$$

KL divergence is non-negative, and it is not symmetric for some P and Q: $KL(P\|Q) \neq KL(Q\|P)$. The KL divergence is 0 if and only if P and Q are the same distribution in the case of discrete variables, or equal "almost everywhere" in the case of continuous variables.

2.4.4 Cross-entropy

A quantity that is closely related to the KL divergence is the cross-entropy, which is similar to the KL divergence but lacking the term on the left:

$$H(P,Q) = H(P) + KL(P\|Q) = -\mathbf{E}_{x\sim P}\left[\log Q(x)\right]$$

2.5 Optimization

2.5.1 One-dimensional Minimization

Sufficient Conditions for a Minimum For x^* to be a local minimum of a univariate function $f(x)$, we require that $f(x^*) \leq f(x)$ for all x which are sufficiently close to x^*. More precisely, for x^* to be a local minimum, there must exist a positive constant Δ such that the inequality holds for all x satisfying $|x - x^*| < \Delta$. Further, x^* is a global minimum if the inequality holds for all x, i.e. one can choose $\Delta = \infty$.

Now assume that $f(x)$ can be differentiated three times at some point x^*. Then we have the Taylor expansion:

$$f(x^* + h) = f(x^*) + f'(x^*)h + \frac{f''(x^*)}{2}h^2 + O\left(h^3\right)$$

Choosing a small positive ϵ and setting $h = -f'(x^*)\epsilon$, we get $f(x^* + h) \approx f(x^*) - f'(x^*)^2\epsilon$ and thus $f(x^* + h) < f(x^*)$ unless $f'(x^*) = 0$. Consequently x^* can only be a local minimum if $f'(x^*) = 0$. Let us assume that $f'(x^*) = 0$ and that further $f''(x^*)$ is greater than zero.

Then for small values of h the Taylor expansion yields,

$$f(x^* + h) \approx f(x^*) + \frac{f''(x^*)}{2}h^2$$

and thus $f(x^* + h) > f(x^*)$. So in this case x^* is a local minimum of f. In case that also $f''(x^*) = 0$, one has to take into account the behavior of the higher order term $O(h^3)$. To summarize, $f'(x^*) = 0$, $f''(x^*) > 0$ is a sufficient condition for x^* to be a local minimum.

Rate of Convergence In the sequel we shall consider algorithms which compute a sequence of improving approximation x_k to x^*. So, numerical considerations aside, $\lim_{k\to\infty} x_k = x^*$. One way of comparing different algorithms, is to consider the rate of convergence p of the sequence x_k. The rate of convergence p is defined to be the largest number for which it is possible to find a bount C_p so that the following inequality holds for all k:

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} \leq C_p$$

For example, the sequence $x_k = 2^{-k}$ has rate of convergence $p = 1$. Obviously $x^* = 0$ and

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} = \frac{2^{-k-1}}{2^{-kp}} = 2^{-k-1+kp} = 2^{-1}2^{k(p-1)}$$

So for $p = 1$ we can use $C_1 = 2^{-1}$ whereas $2^{k(p-1)}$ diverges for $p > 1$.

Brackets A bracket for minimizing a function f, is a triple $\{a, b, c\}$ of real numbers, such that $a < b < c$ and $f(a) \geq f(b) \leq f(c)$. Since the value of f on the interior point b is smaller or equal to the value on the boundary points a and c, one will expect that there is a local minimum somewhere inside the interval $[a, c]$. Once we have bracket, it is straightforward to obtain a better approximation, by picking a point $x \neq b$ in the interval (a, c) and comparing $f(x)$ to $f(b)$.

Assume we pick x such that $a < x < b$, then either $f(x) \geq f(b)$, so $\{x, b, c\}$ is a new bracket; or $f(x) < f(b)$, so $\{a, x, b\}$ is a new bracket. In the first case, it is obvious that $\{x, b, c\}$ is a bracket. For the second case, $f(x) < f(b)$, note that $f(a) \geq f(b)$, since $\{a, b, c\}$ is a bracket. So $f(a) > f(x)$ and $\{a, x, b\}$ is indeed a bracket. In the case that we choose to pick x such that $b < x < c$, it is either $f(x) \geq f(b)$, so $\{a, b, x\}$ is a new bracket; or $f(x) < f(b)$, so $\{b, x, c\}$ is a new bracket. So, by choosing an interior point x in the current bracket and obtaining a new bracket, we arrive at better and better approximations to the location of the minimum.

2.5.2 Gradient Descent

Exact Line Search Condition The iterative search technique that we proceed towards the minimum $\boldsymbol{x}^*$ by a sequence of steps. On the k^{th} step we take a step of length α_k in the direction $\boldsymbol{p}_k$,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$$

The length of the step can either be chosen using prior knowledge, or by carrying out a line search in the direction $\boldsymbol{p}_k$. At the k^{th} step, we chose α_k to minimize $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)$. So setting $F(\lambda) = f(\boldsymbol{x}_k + \lambda \boldsymbol{p}_k)$, at this step we solve the one-dimensional minimization problem for $F(\lambda)$. Thus our choice of $\alpha_k = \lambda^*$ will satisfy $F'(\alpha_k) = 0$. We can calculate that $F'(\alpha_k) = \nabla f^\top(\boldsymbol{x}_{k+1})\boldsymbol{p}_k$. So $F'(\alpha_k) = 0$ means the directional derivative in the search direction must vanish at the new point and this gives the exact line search condition: $\nabla f^\top(\boldsymbol{x}_{k+1})\boldsymbol{p}_k = 0$.

For a quadratic function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \mathbf{A}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x} + c$, with symmetric $\mathbf{A}$, we can use he condition to analytically calculate α_k. Since $\nabla f(\boldsymbol{x}_{k+1}) = \mathbf{A}\boldsymbol{x}_k + \alpha_k \mathbf{A}\boldsymbol{p}_k - \boldsymbol{b} = \nabla f(\boldsymbol{x}_k) + \alpha_k \mathbf{A}\boldsymbol{p}_k$, we find $\alpha_k = -\frac{\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k)}{\boldsymbol{p}_k^\top \mathbf{A}\boldsymbol{p}_k}$.

Gradient Descent The simplest choice for $\boldsymbol{p}_k$ is to set it equal to $-\nabla f(\boldsymbol{x}_k)$. If we find that $\nabla f(\boldsymbol{x}_k) = \mathbf{0}$ we can stop. To see this, expand f around $\boldsymbol{x}_k$ using Taylor's theorem:

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \approx f(\boldsymbol{x}_k) + \alpha_k \nabla f^\top(\boldsymbol{x}_k)\boldsymbol{p}_k$$

With $\boldsymbol{p}_k = -\nabla f(\boldsymbol{x}_k)$ and for small positive α_k, we see a guaranteed reduction:

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \approx f(\boldsymbol{x}_k) - \alpha_k \left|\left|\nabla f(\boldsymbol{x}_k)\right|\right|^2$$

2.5.3 Quadratic Functions

Minimizing Quadratic Functions Using Line Search Consider a function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \mathbf{A}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x} + c$ where $\mathbf{A}$ is positive definite and symmetric. One approach to finding minima is to search along a particular direction $\boldsymbol{p}$, and find a minimum along this direction. We can then search for a deeper minima by looking in different directions. That is, we can search along a line $\boldsymbol{x}^* = \boldsymbol{x}^0 + \lambda \boldsymbol{p}$ such that the function attains a minimum. Differentiate the original function and set to zero, we have the solution

$$\lambda = \frac{(\boldsymbol{b} - \mathbf{A}\boldsymbol{x}^0)\boldsymbol{p}}{\boldsymbol{p}^\top \mathbf{A}\boldsymbol{p}} = \frac{-\nabla f(\boldsymbol{x}^0)\boldsymbol{p}}{\boldsymbol{p}^\top \mathbf{A}\boldsymbol{p}}$$

It would seem sensible to choose successive line search directions $\boldsymbol{p}$ according to $\boldsymbol{p}^{new} = -\nabla f(\boldsymbol{x}^*)$, so that each time we minimize the function along the line of steepest descent. However, this is far from the optimal choice in the case of minimizing quadratic functions.

Conjugate Vectors The vectors $\boldsymbol{p}_i, i = 1, \ldots, k$ are called conjugate to the matrix $\mathbf{A}$, iff for $i, j = 1, \ldots, k$ and $i \neq j$:

$$\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_j = 0 \quad \text{and} \quad \boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_i > 0$$

The two conditions guarantee that conjugate vectors are linearly independent: assume that

$$\mathbf{0} = \sum_{j=1}^{k} \alpha_j \boldsymbol{p}_j = \sum_{j=1}^{i-1} \alpha_j \boldsymbol{p}_j + \alpha_i \boldsymbol{p}_i + \sum_{j=i+1}^{k} \alpha_j \boldsymbol{p}_j$$

Now multiplying from the left with $\boldsymbol{p}_i^\top \mathbf{A}$ yields $0 = \alpha_i \boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_i$. So α_i is zero since we know that $\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_i > 0$. As we can make this argument for any $i = 1, \ldots, k$, all of the α_i must be zero. That conjugate vectors are linearly independent, means that the matrix $\mathbf{P} = [\boldsymbol{p}_1 \ \boldsymbol{p}_2 \ \cdots \ \boldsymbol{p}_k]$ has full rank. In particular, if $k = n$, the matrix $\mathbf{P}$ will be invertible. If the symmetric $n \times n$ matrix $\mathbf{A}$ has n conjugate directions, it is positive definite.

Gram-Schmidt Procedure We now turn to constructing conjugate vectors using a procedure which is analogous to Gram-Schmidt orthogonalization. Assume we already have k conjugate vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$ and let $\boldsymbol{v}$ be a vector which is linearly independent of $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$. We then set

$$\boldsymbol{p}_{k+1} = \boldsymbol{v} - \sum_{j=1}^{k} \frac{\boldsymbol{p}_j^\top \mathbf{A} \boldsymbol{v}}{\boldsymbol{p}_j^\top \mathbf{A} \boldsymbol{p}_j} \boldsymbol{p}_j$$

and claim that now the vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k+1}$ are conjugate if $\mathbf{A}$ is positive definite: since $\boldsymbol{v}$ is linearly independent of $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$, the new vector $\boldsymbol{p}_{k+1}$ cannot be zero, and thus for positive definite $\mathbf{A}$, $\boldsymbol{p}_{k+1}^\top \mathbf{A} \boldsymbol{p}_{k+1} > 0$. Also, $\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_{k+1} = 0$ that

$$\begin{aligned}
\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_{k+1} &= \boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{v} - \sum_{k=1}^{k} \frac{\boldsymbol{p}_j^\top \mathbf{A} \boldsymbol{v}}{\boldsymbol{p}_j^\top \mathbf{A} \boldsymbol{p}_j} \boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_j \\
&= \boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{v} - \frac{\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{v}}{\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_i} \boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_i \\
&= 0
\end{aligned}$$

An important property of the Gram-Schmidt procedure is that the vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k+1}$ span the same subspace as the vectors $\boldsymbol{v}$ and $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$: any linear combination of $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k+1}$ can be rewritten as a linear combination of $\boldsymbol{v}$ and $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$ just by replacing $\boldsymbol{p}_{k+1}$ with the RHS of Gram-Schmidt equation. So the space spanned by $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k+1}$ is contained in the one spanned by $\boldsymbol{v}$ and $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$. But solving the Gram-Schmidt equation for $\boldsymbol{v}$ allows us to turn the argument around and so the two subspaces are the same.

Using the Gram-Schmidt procedure, we can construct n conjugate vectors for a positive definite matrix in the following way. We start with n linearly independent vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$, e.g. we might choose $\boldsymbol{u}_i = \boldsymbol{e}_i$, the unit vector in the i^{th} direction. We then set $\boldsymbol{p}_1 = \boldsymbol{u}_1$ and use the equation to compute $\boldsymbol{p}_2$ from $\boldsymbol{p}_1$ and $\boldsymbol{v} = \boldsymbol{u}_2$. Next we set $\boldsymbol{v} = \boldsymbol{u}_3$ and compute $\boldsymbol{p}_3$ from $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{v}$. Continuing in this manner we obtain n conjugate vectors. Note that at each stage of the procedure the vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ span the same subspace as the vector $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_j$.

The Conjugate Vectors Algorithm Assume that when minimizing $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \mathbf{A}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x} + c$ we first construct n vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ conjugate to $\mathbf{A}$ which we use as our search directions. So

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_{\boldsymbol{k}}$$

At each step we choose α_k by an exact line search, thus

$$\alpha_k = -\frac{\boldsymbol{p}_k^\top \nabla f(\boldsymbol{x}_k)}{\boldsymbol{p}_k^\top \mathbf{A}\boldsymbol{p}_k}$$

We call this procedure the conjugate vectors algorithm. According to Expanding Subspace Theorem, the directional derivative $D_{\boldsymbol{p}_i} f(\boldsymbol{x}_{k+1}) = \nabla f^\top(\boldsymbol{x}_{k+1})\boldsymbol{p}_i = 0,\ i = 1, \ldots, k$, which means it is not only zero at the new point along the direction $\boldsymbol{p}_k$, but also zero along all the previous search direction $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$.

The Conjugate Gradients Algorithm The conjugate gradients algorithm is a special case of the conjugate vectors algorithm, where we choose vector $\boldsymbol{v} = -\nabla f(\boldsymbol{x}_{k+1})$. According to subspace theorem the gradient at the new point $\boldsymbol{x}_{k+1}$ is orthogonal to $\boldsymbol{p}_i,\ i = 1, \ldots, k$ so it is linearly independent of $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$ and a valid choice for $\boldsymbol{v}$. The equation for the new search direction given by the Gram-Schmidt procedure is

$$\boldsymbol{p}_{k+1} = -\nabla f(\boldsymbol{x}_{k+1}) + \sum_{i=1}^{k} \frac{\boldsymbol{p}_i^\top \mathbf{A} \nabla f(\boldsymbol{x}_{k+1})}{\boldsymbol{p}_i^\top \mathbf{A}\boldsymbol{p}_i} \boldsymbol{p}_i$$

Since $\nabla f(\boldsymbol{x}_{k+1})$ is orthogonal to $\boldsymbol{p}_i,\ i = 1, \ldots, k$, by the subspace theorem we have $\boldsymbol{p}_{k+1}^\top \nabla f(\boldsymbol{x}_{k+1}) = -\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})$. So α_{k+1} can be written as

$$\alpha_{k+1} = \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})}{\boldsymbol{p}_{k+1}^\top \mathbf{A}\boldsymbol{p}_{k+1}}$$

We can show that

$$\begin{aligned}\nabla f(\boldsymbol{x}_{i+1}) - \nabla f(\boldsymbol{x}_i) = \mathbf{A}\boldsymbol{x}_{i+1} - \boldsymbol{b} - (\mathbf{A}\boldsymbol{x}_i - b) = \mathbf{A}(\boldsymbol{x}_{i+1} - \boldsymbol{x_i}) = \alpha_i \mathbf{A}\boldsymbol{p}_i \\ \mathbf{A}\boldsymbol{p}_i = (\nabla f(\boldsymbol{x}_{i+1}) - \nabla f(\boldsymbol{x}_i))/\alpha_i \quad \text{since } \alpha_i \neq 0\end{aligned}$$

Further,

$$\begin{aligned}\boldsymbol{p}_i^\top \mathbf{A}\nabla f(\boldsymbol{x}_{k+1}) &= \nabla f^\top(\boldsymbol{x}_{k+1})\mathbf{A}\boldsymbol{p}_i = \nabla f^\top(\boldsymbol{x}_{k+1})(\nabla f(\boldsymbol{x}_{i+1}) - \nabla f(\boldsymbol{x}_i))/\alpha_i \\ &= \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{i+1}) - \nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_i)}{\alpha_i} \\ &= \begin{cases} 0 & \text{if } 1 \leq i < k \\ \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})}{\alpha_k} & \text{if } i = k \end{cases}\end{aligned}$$

Hence we can simplify the equations as

$$\begin{aligned}\boldsymbol{p}_{k+1} &= -\nabla f(\boldsymbol{x}_{k+1}) + \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})/\alpha_k}{\boldsymbol{p}_k^\top \mathbf{A}\boldsymbol{p}_k}\boldsymbol{p}_k \\ &= -\nabla f(\boldsymbol{x}_{k+1}) + \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})}{\boldsymbol{p}_k^\top \mathbf{A}\boldsymbol{p}_k}\frac{\boldsymbol{p}_k^\top \mathbf{A}\boldsymbol{p}_k}{\nabla f^\top(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)}\boldsymbol{p}_k \\ &= -\nabla f(\boldsymbol{x}_{k+1}) + \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})}{\nabla f^\top(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)}\boldsymbol{p}_k \\ &= -\nabla f(\boldsymbol{x}_{k+1}) + \beta_k \boldsymbol{p}_k \quad \text{where } \beta_k = \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1})}{\nabla f^\top(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)}\end{aligned}$$

We can conclude the conjugate gradients algorithm as

Algorithm 1 The Conjugate Gradients Algorithm

$k = 1$
input $\boldsymbol{x}_1$
$\boldsymbol{p}_1 = -\nabla f(\boldsymbol{x}_1)$
while $\nabla f(\boldsymbol{x}_k) \neq 0$ **do**
  $\alpha_k := -\nabla f^\top(\boldsymbol{x}_k)\boldsymbol{p}_k / \left(\boldsymbol{p}_k^\top \mathbf{A} \boldsymbol{p}_k\right)$
  $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
  $\beta_k := \nabla f^\top(\boldsymbol{x}_{k+1})\nabla f(\boldsymbol{x}_{k+1}) / \left(\nabla f^\top(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)\right)$
  $\boldsymbol{p}_{k+1} := -\nabla f(\boldsymbol{x}_{k+1}) + \beta_k \boldsymbol{p}_k$
  $k = k + 1$
end while

An common alternative is the Polak-Ribiere formula:

$$\beta_k = \frac{\nabla f^\top(\boldsymbol{x}_{k+1})\left(\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)\right)}{\nabla f^\top(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)}$$

Newton Methods Consider the Taylor expansion of a target function $f(\boldsymbol{x})$:

$$f(\boldsymbol{x} + \epsilon \boldsymbol{v}) = f(\boldsymbol{x}) + \epsilon \nabla f^\top \boldsymbol{v} + \frac{\epsilon^2}{2} \boldsymbol{v}^\top \mathbf{H}_f \boldsymbol{v} + O\left(\epsilon^3\right)$$

Differentiating the RHS with respect to $\boldsymbol{v}$, we find the RHS has its lowest value when

$$\nabla f = -\epsilon \mathbf{H}_f \boldsymbol{v} \Rightarrow \boldsymbol{v} = -\epsilon^{-1} \mathbf{H}_f^{-1} \nabla f$$

Hence, an optimization routine to minimize f is given by the Newton update

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \mathbf{H}_f^{-1} \nabla f$$

Although Newton's method can help to find the minima in one step, for large scale problems the Hessian matrix is computationally demanding, especially if it is singular or nearly so.

Quasi-Newton Methods An alternative is to set up the iteration

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \mathbf{S}_k \nabla f(\boldsymbol{x}_k)$$

This is a very general form: if $\mathbf{S}_k = \mathbf{H}_f^{-1}$ then we have Newton's method, while if $\mathbf{S}_k = \mathbf{I}$ we have steepest (gradient) descent. The idea behind most quasi-Newton methods is to try to construct an approximate inverse Hessian $\mathbf{H}_k$ using information gathered as the descent progresses, and to set $\mathbf{S}_k = \mathbf{H}_k$. As we have seen, for a quadratic optimization problem we have the relationship $\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k) = \mathbf{A}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$. Defining $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$

and $\boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$, we see that the original equation becomes $\boldsymbol{y}_k = \mathbf{A}\boldsymbol{s}_k$. It is reasonable to demand that $\mathbf{H}_{k+1}\boldsymbol{y}_i = \boldsymbol{s}_i, 1 \leq i \leq k$. After n linearly independent steps we would then have $\mathbf{H}_{n+1} = \mathbf{A}^{-1}$. For $k < n$ there are an infinity of solutions for $\mathbf{H}_{k+1}$ satisfying the equation. We shall focus on one of the most popular, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update. This is given by

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \left(1 + \frac{\boldsymbol{y}_k^\top \mathbf{H}_k \boldsymbol{y}_k}{\boldsymbol{y}_k^\top \boldsymbol{s}_k}\right) \frac{\boldsymbol{s}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{y}_k} - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^\top \mathbf{H}_k + \mathbf{H}_k \boldsymbol{y}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{y}_k}$$

This is a rand-2 correction to $\mathbf{H}_k$ constructed from the vectors $\boldsymbol{s}_k$ and $\mathbf{H}_k\boldsymbol{y}_k$. The quasi-Newton algorithm is

Algorithm 2 The Quasi-Newton Algorithm

input $\boldsymbol{x}_1$
set $\mathbf{H}_i = \mathbf{I}$
for $k = 1, \ldots, n$ **do**
  $\boldsymbol{p}_k = -\mathbf{H}_k \nabla f(\boldsymbol{x}_k)$
  compute α_k to minimize $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)$
  $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, $\boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$, and update $\mathbf{H}_{k+1}$
  $k = k + 1$
end for

Properties of the BFGS update:

1. All the $\mathbf{H}$'s are symmetric.
2. All the $\mathbf{H}$'s are positive definite.
3. The direction vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$ produced by the algorithm obey $\boldsymbol{p}_i^\top \mathbf{A} \boldsymbol{p}_j = 0, 1 \leq i < j \leq k$ and $\mathbf{H}_{k+1}\mathbf{A}\boldsymbol{p}_i = \boldsymbol{p}_i, 1 \leq i \leq k$.

2.5.4 General Functions

The basic strategy for the minimization of general (i.e. non-quadratic) functions is to apply the same algorithms as in the quadratic case, except that exact line-minimizations (the calculation of the α_k's) have to be replaced with inexact line-searches.

For the conjugate gradient (CG) method, the exact formula used to compute β_k may be important in this respect. For example, the two possible formulae are

$$\beta_k = \frac{\nabla f^\top(\boldsymbol{x}_{k+1}) \nabla f(\boldsymbol{x}_{k+1})}{\nabla f^\top(\boldsymbol{x}_k) \nabla f(\boldsymbol{x}_k)} \qquad \text{Fletcher-Reeves}$$

$$\beta_k = \frac{\nabla f^\top(\boldsymbol{x}_{k+1}) \left(\nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)\right)}{\nabla f^\top(\boldsymbol{x}_k) \nabla f(\boldsymbol{x}_k)} \qquad \text{Polak-Ribiere}$$

These formulae are equivalent for quadratic functions, however, they can have quite different effects in the non-quadratic case. If the algorithm is not making progress then we can expect $\nabla f(\boldsymbol{x}_{k+1}) \simeq \nabla f(\boldsymbol{x}_k)$, and hence that β_k computed by the Polak-Ribiere formula will be near 0, and hence that $\boldsymbol{p}_{k+1} \simeq -\nabla f(\boldsymbol{x}_{k+1})$. On the other hand, under the same circumstances the Fletcher-Reeves update will set $\beta_k \simeq 1$.

For general functions it can be shown that the BFGS quasi-Newton (QN) method:

1. preserves positive definite matrices $\mathbf{H}_k$, and hence that the descent property holds;
2. requires $O(n^2)$ multiplications per iteration;
3. has superlinear order of convergence;
4. has global convergence for strictly convex functions (with exact line searches).

A comparison of the CG and QN algorithms shows that CG methods require only $O(n)$ storage, as opposed to $O(n^2)$ for QN methods. Typically this means that QN methods are preferred for problems of relatively small size, while CG methods are more suitable for very large problems. QN methods appear to be more tolerant of moderate precision line-searches compared to CG methods, and CG methods are less efficient and less robust than QN methods on problems to which they can both be applied.

2.5.5 Optimization with Constraints

Constraint Types There are two types of constraints:

1. Equality (type A). $c(\boldsymbol{x}) = 0$. These reduce the dimension of the search space.
2. Inequality (type B). $c(\boldsymbol{x}) \geq 0$ or $c(\boldsymbol{x}) > 0$. These reduce the volume of the search space, but do not affect the dimension.

Transformation Methods For some type A constraints, it is possible to solve for one variable x_i in terms of the other coordinates of $\boldsymbol{x}$. If this can be done, then x_i can be eliminated from f, and the whole problem reduced to one of lower dimension with fewer constraints.

Some type B constraints can be removed by simple variable substitutions so that they are automatically satisfied. For example, $x_i \geq 0$ can be removed by substituting $x_i = y_i^2$ or $x_i = e^{y_i}$, while $a < x_i < b$ can be removed by substituting $x_i = (a+b)/2 + ((a-b)/2)\tanh y_i$. Another useful substitution is the softmax function. Minimizing $f(\boldsymbol{x})$ subject to the constraint $\sum_{i=1}^n x_i = 1$, where $0 \leq x_i \leq 1$, for $i = 1, \ldots, n$ can be transformed to minimizing f with $x_i = e^{y_i} / \sum_{j=1}^n e^{y_j}$.

Lagrange Multipliers (Single Constraint) Consider the problem of minimizing $f(\boldsymbol{x})$ subject to a single constraint $c(\boldsymbol{x}) = 0$. Assume we already identified an $\boldsymbol{x}$ that satisfies the constraint, what we are allowed is to search for lower function values around this $\boldsymbol{x}$ in directions which are consistent with te constraint. That is, $c(\boldsymbol{x} + \boldsymbol{\delta}) = 0$. We know that $c(\boldsymbol{x}+\boldsymbol{\delta}) \approx c(\boldsymbol{x})+\boldsymbol{\delta}\cdot\nabla c(\boldsymbol{x})$. Hence, in order that the constraint remains satisfies, we can only search in a direction such that $\boldsymbol{\delta} \cdot \nabla c(\boldsymbol{x}) = 0$, which means in the direction $\boldsymbol{\delta}$ is orthogonal to $\nabla c(\boldsymbol{x})$.

The change in f along a direction $\boldsymbol{n}$ where $\boldsymbol{n}\cdot\nabla c(\boldsymbol{x}) = 0$ is $f(\boldsymbol{x}+\epsilon\boldsymbol{n}) \approx f(\boldsymbol{x})+\epsilon\nabla f(\boldsymbol{x})\cdot\boldsymbol{n}$. Since we are looking for a point $\boldsymbol{x}$ that minimizes the function f, we require $\boldsymbol{x}$ to be a stationary point, $\nabla f(\boldsymbol{x})\cdot\boldsymbol{n} = 0$. That is, $\nabla f(\boldsymbol{x})$ must lie parallel to $\nabla c(\boldsymbol{x})$, so that $\nabla f(\boldsymbol{x}) = \lambda\nabla c(\boldsymbol{x})$ for some $\lambda \in \mathbb{R}$.

To solve the optimization problem therefore, we look for a point $\boldsymbol{x}$ such that $\nabla f(\boldsymbol{x}) = \lambda\nabla c(\boldsymbol{x})$, for some λ, and for which $c(\boldsymbol{x}) = 0$. An alternative formulation of this dual requirement is to look for $\boldsymbol{x}$ and λ that jointly minimize the ***Lagrangian***:

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda c(\boldsymbol{x})$$

Differentiating with respect to $\boldsymbol{x}$, we get the requirement $\nabla f(\boldsymbol{x}) = \lambda\nabla c(\boldsymbol{x})$, and differentiating with respect to λ, we get that $c(\boldsymbol{x}) = 0$.

Lagrange Multipliers (Multiple Constraints) Consider the problem of optimizing $f(\boldsymbol{x})$ subject to the constraints $c_i(\boldsymbol{x}) = 0, i = 1, \ldots, r < n$, where n is the dimensionality of the space. We can apply the ***Lagrangian function*** $\mathcal{L} : \mathbb{R}^{n+r} \to \mathbb{R}$,

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_i \lambda_i c_i(\boldsymbol{x})$$

Computational Approaches to Constrained Optimization A computational approach to the non-linear constrained optimization problem is the penalty and barrier function methods. Suppose we are minimizing a function $f(\boldsymbol{x})$ subject to constraints $c_i(\boldsymbol{x}) = 0, i = 1, \ldots, r$. We can construct a quadratic penalty function

$$P_Q(\boldsymbol{x}, \rho) = f(\boldsymbol{x}) + \frac{\rho}{2}\boldsymbol{c}(\boldsymbol{x})^\top \boldsymbol{c}(\boldsymbol{x})$$

where $\boldsymbol{c}(\boldsymbol{x})$ is the vector of constraints $[c_1(\boldsymbol{x}) \ \cdots \ c_r(\boldsymbol{x})]^\top$ and ρ is the penalty parameter. Note that the penalty term is continuously differentiable. Under mild technical conditions, if $\boldsymbol{x}^*(\rho)$ denotes an unconstrained munimum of P_Q then it can be shown that $\lim_{\rho\to\infty} \boldsymbol{x}^*(\rho) = \boldsymbol{x}^*$ where $\boldsymbol{x}^*$ is the constrained minimum of f.

Chapter 3

Machine Learning Basics

3.1 Regularization

3.1.1 Under-fitting & Over-fitting

Under-fitting If $N > D$ (e.g. 30 data points, 2 dimensions) we have more equations than unknowns: over-determined system. Input-output relations can only hold approximately.

Over-fitting If $N < D$ (e.g. 30points, 15265 dimensions) we have more unknowns than equations: under-determined system. Input-output equations hold exactly, but we are simply memorizing data.

3.1.2 Bias & Variance

High Bias & Low Variance A rigid model's (low complexity) performance is more predictable in the test set but the model may not be good even on the training set.

Low Bias & High Variance A flexible model (high complexity) approximates the target function well in the training set but can "overtrain" and have poor performance on the test set.

3.1.3 Vector Norm

L1, ("Manhattan") norm $||\boldsymbol{w}||_1 = \sum_{d=1}^{D} |w_d|$

L2, ("Euclidean") norm $||\boldsymbol{w}||_2 = \sqrt{\sum_{d=1}^{D} w_d^2} = \sqrt{\boldsymbol{w}^\top \boldsymbol{w}}$

Lp norm, p>1 $||\boldsymbol{w}||_p = \left(\sum_{d=1}^{D} w_d^p\right)^{1/p}$

3.1.4 Penalize Complexity

In linear regression, the residual vector is $\boldsymbol{\epsilon} = \boldsymbol{y} - \boldsymbol{\Psi}\boldsymbol{w}$. The loss function is $L(\boldsymbol{w}) = \boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}$. We add a complexity term $R(\boldsymbol{w}) = ||\boldsymbol{w}||_2 = \boldsymbol{w}^\top \boldsymbol{w}$ to the loss function. Hence, the original loss function becomes $L(\boldsymbol{w}) = \boldsymbol{\epsilon}^\top \boldsymbol{\epsilon} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$.

Without regularization, the loss function is $L(\boldsymbol{w}) = \boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}$. Let $\nabla L(\boldsymbol{w}^*) = 0$, we have $\boldsymbol{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}$.

With L2-regularization, the loss function is $L(\boldsymbol{w}) = \boldsymbol{\epsilon}^\top \boldsymbol{\epsilon} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$. Let $\nabla L(\boldsymbol{w}^*) = 0$, we have $\boldsymbol{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \boldsymbol{y}$. The additional $\lambda \mathbf{I}$ makes the data matrix more robust to calculate inversion.

3.2 Cross-Validation

We can select hyperparameters with (cross-)validation. Cross-validation excludes part of the training data from parameter estimation, and use them only to predict the test error.

K-fold cross validation: split data set into K folds and each time train on (K-1) folds and valid on the remaining fold until all folds have been used as validation fold. The cross-validation error is the average of K validation errors. We pick hyperparameters that minimize cross-validation error.

3.3 Bayesian Learning

3.3.1 Bayes' Rule Terminology

Bayes' Rule:

$$P(y|x) = \frac{P(x|y)\,P(y)}{\int P(x|y)\,P(y)\,dy}$$

Prior $P(y)$ what we know about y before seeing x. In parameters learning we choose prior that is conjugate to likelihood.

Likelihood $P(x|y)$ propensity for observing a certain value of x given a certain value of y.

Posterior $P(y|x)$ what we know about y after seeing x. Posterior must have same form as conjugate prior distribution.

Evidence $\int P(x|y)\,P(y)\,dy$ a constant to ensure that the LHS is a valid distribution. Posterior must be a distribution which implies that evidence equals to a constant κ from conjugate relation.

3.3.2 Maximum Likelihood

Fitting As the name suggests we find the parameters under which the data $\boldsymbol{x}_{1...I}$ are most likely. Here, we have assumed that data was independent.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} P\left(\boldsymbol{x}_{1...I}|\boldsymbol{\theta}\right) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{I} P\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)\end{aligned}$$

Predictive Density Evaluate new data point $\boldsymbol{x}^*$ under probability distribution $P\left(\boldsymbol{x}^*|\hat{\boldsymbol{\theta}}\right)$ with best parameters.

3.3.3 Maximum a Posterior (MAP)

Fitting As the name suggests we find the parameters which maximize the posterior probability $P\left(\boldsymbol{\theta}|\boldsymbol{x}_{1...I}\right)$. Again we have assumed that data was independent.

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} P\left(\boldsymbol{\theta}|\boldsymbol{x}_{1...I}\right) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{P\left(\boldsymbol{x}_{1...I}|\boldsymbol{\theta}\right) P\left(\boldsymbol{\theta}\right)}{P\left(\boldsymbol{x}_{1...I}\right)} \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{\prod_{i=1}^{I} P\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)\ P\left(\boldsymbol{\theta}\right)}{P\left(\boldsymbol{x}_{1...I}\right)}\end{aligned}$$

Since the denominator does not depend on the parameters we can instead maximize

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{I} P\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)\ P\left(\boldsymbol{\theta}\right)$$

Predictive Density Evaluate new data point $\boldsymbol{x}^*$ under probability distribution with MAP parameters $P\left(\boldsymbol{x}^*|\hat{\boldsymbol{\theta}}\right)$

3.3.4 Bayesian Approach

Fitting Compute the posterior distribution over possible parameter values using Bayes' rule. Principle: There are many values that could have explained the data. Instead of picking one set of parameters, try to capture all of the possibilities.

$$P\left(\boldsymbol{\theta}|\boldsymbol{x}_{1...I}\right) = \frac{\prod_{i=1}^{I} P\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)\ P\left(\boldsymbol{\theta}\right)}{P\left(\boldsymbol{x}_{1...I}\right)}$$

Predictive Density (a) Each possible parameter value makes a prediction. (b) Some parameters more probable than others.

$$P\left(\boldsymbol{x}^{*}|\boldsymbol{x}_{1...I}\right)=\int P\left(\boldsymbol{x}^{*}|\boldsymbol{\theta}\right)P\left(\boldsymbol{\theta}|\boldsymbol{x}_{1...I}\right)d\boldsymbol{\theta}$$

Make a prediction that is an infinite weighted sum (integral) of the predictions for each parameter value ($P\left(\boldsymbol{x}^{*}|\boldsymbol{\theta}\right)$), where weights are the probabilities ($P\left(\boldsymbol{\theta}|\boldsymbol{x}_{1...I}\right)$).

3.3.5 Example: Univariate Normal Distribution

Maximum Likelihood

Likelihood given by normal distribution pdf:

$$P\left(x|\mu,\sigma^{2}\right)=\mathbf{Norm}_{x}[\mu,\sigma^{2}]=\frac{1}{\sqrt{2\pi\sigma^{2}}}exp\left(-\frac{(x-\mu)^{2}}{2\sigma^{2}}\right)$$

Apply maximum likelihood:

$$\begin{aligned}\hat{\mu},\hat{\sigma}^{2}&=\underset{\mu,\sigma^{2}}{\operatorname{argmax}}\,P\left(x_{1...I}|\mu,\sigma^{2}\right)\\&=\underset{\mu,\sigma^{2}}{\operatorname{argmax}}\prod_{i=1}^{I}P\left(x_{i}|\mu,\sigma^{2}\right)\\&=\underset{\mu,\sigma^{2}}{\operatorname{argmax}}\prod_{i=1}^{I}\mathbf{Norm}_{x_{i}}[\mu,\sigma^{2}]\\&=\underset{\mu,\sigma^{2}}{\operatorname{argmax}}\sum_{i=1}^{I}\log\mathbf{Norm}_{x_{i}}[\mu,\sigma^{2}]\\&=\underset{\mu,\sigma^{2}}{\operatorname{argmax}}\left(-\frac{I}{2}\log 2\pi-\frac{I}{2}\log\sigma^{2}-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_{i}-\mu)^{2}}{\sigma^{2}}\right)\end{aligned}$$

Let $\nabla L(\hat{\mu},\hat{\sigma}^{2})=0$, we have the solution:

$$\begin{aligned}\hat{\mu}&=\frac{\sum_{i=1}^{I}x_{i}}{I}\\\hat{\sigma}^{2}&=\sum_{i=1}^{I}\frac{(x_{i}-\hat{\mu})^{2}}{I}\end{aligned}$$

Maximum a Posterior

Likelihood given by normal distribution pdf:

$$P\left(x|\mu,\sigma^2\right) = \mathbf{Norm}_x[\mu,\sigma^2] = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Prior given by normal inverse gamma distribution pdf:

$$P\left(\mu,\sigma^2\right) = \mathbf{NormInvGam}_{\mu,\sigma^2}[\alpha,\beta,\gamma,\delta] = \frac{\sqrt{\gamma}}{\sqrt{2\pi\sigma^2}}\frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{\sigma^2}\right)^{\alpha+1} exp\left(-\frac{2\beta+\gamma(\delta-\mu)^2}{2\sigma^2}\right)$$

Apply maximum a posterior:

$$\begin{aligned}
\hat{\mu},\hat{\sigma}^2 &= \underset{\mu,\sigma^2}{\text{argmax}} \prod_{i=1}^{I} P\left(x_i|\mu,\sigma^2\right)\ P\left(\mu,\sigma^2\right) \\
&= \underset{\mu,\sigma^2}{\text{argmax}} \prod_{i=1}^{I} \mathbf{Norm}_{x_i}[\mu,\sigma^2]\ \mathbf{NormInvGam}_{\mu,\sigma^2}[\alpha,\beta,\gamma,\delta] \\
&= \underset{\mu,\sigma^2}{\text{argmax}} \left(\sum_{i=1}^{I} \log \mathbf{Norm}_{x_i}[\mu,\sigma^2] + \log \mathbf{NormInvGam}_{\mu,\sigma^2}[\alpha,\beta,\gamma,\delta]\right)
\end{aligned}$$

Let $\nabla L(\hat{\mu},\hat{\sigma}^2) = 0$, we have the solution:

$$\begin{aligned}
\hat{\mu} &= \frac{\sum_{i=1}^{I} x_i + \gamma\delta}{I+\gamma} \\
\hat{\sigma}^2 &= \frac{\sum_{i=1}^{I}(x_i-\mu)^2 + 2\beta + \gamma(\delta-\mu)^2}{I+3+2\alpha}
\end{aligned}$$

Bayesian Approach

Compute the posterior distribution using Bayes' rule:

$$\begin{aligned}
P\left(\mu,\sigma^2|x_{1\ldots I}\right) &= \frac{\prod_{i=1}^{I} P\left(x_i|\mu,\sigma^2\right)\ P\left(\mu,\sigma^2\right)}{P\left(x_{1\ldots I}\right)} \\
&= \frac{\prod_{i=1}^{I} \mathbf{Norm}_{x_i}[\mu,\sigma^2]\ \mathbf{NormInvGam}_{\mu,\sigma^2}[\alpha,\beta,\gamma,\delta]}{P\left(x_{1\ldots I}\right)} \\
&= \frac{\kappa(\alpha,\beta,\gamma,\delta,x_{1\ldots I})\mathbf{NormInvGam}_{\mu,\sigma^2}[\tilde{\alpha},\tilde{\beta},\tilde{\gamma},\tilde{\delta}]}{P\left(x_{1\ldots I}\right)} \\
&= \mathbf{NormInvGam}_{\mu,\sigma^2}[\tilde{\alpha},\tilde{\beta},\tilde{\gamma},\tilde{\delta}]
\end{aligned}$$

where

$$\begin{aligned}
\tilde{\alpha} &= \alpha + \frac{I}{2} \\
\tilde{\beta} &= \frac{\sum_i x_i^2}{2} + \beta + \frac{\gamma\delta^2}{2} - \frac{(\gamma\delta + \sum_i x_i)^2}{2(\gamma + I)} \\
\tilde{\gamma} &= \gamma + I \\
\tilde{\delta} &= \frac{\gamma\delta + \sum_i x_i}{\gamma + I}
\end{aligned}$$

Take weighted sum of predictions from different parameter values:

$$\begin{aligned}
P\left(x^*|x_{1\ldots I}\right) &= \iint P\left(x^*|\mu,\sigma^2\right) P\left(\mu,\sigma^2|x_{1\ldots I}\right) d\mu d\sigma \\
&= \iint \mathbf{Norm}_{x^*}[\mu,\sigma^2]\mathbf{NormInvGam}_{\mu,\sigma^2}[\tilde{\alpha},\tilde{\beta},\tilde{\gamma},\tilde{\delta}]d\mu d\sigma \\
&= \iint \kappa(\alpha,\beta,\gamma,\delta,x_{1\ldots I})\mathbf{NormInvGam}_{\mu,\sigma^2}[\breve{\alpha},\breve{\beta},\breve{\gamma},\breve{\delta}]d\mu d\sigma \\
&= \kappa(\alpha,\beta,\gamma,\delta,x_{1\ldots I}) \iint \mathbf{NormInvGam}_{\mu,\sigma^2}[\breve{\alpha},\breve{\beta},\breve{\gamma},\breve{\delta}]d\mu d\sigma \\
&= \kappa(\alpha,\beta,\gamma,\delta,x_{1\ldots I}) \\
&= \frac{1}{\sqrt{2\pi}}\frac{\sqrt{\tilde{\gamma}}\tilde{\beta}^{\tilde{\alpha}}}{\sqrt{\breve{\gamma}}\breve{\beta}^{\breve{\alpha}}}\frac{\Gamma(\breve{\alpha})}{\Gamma(\tilde{\alpha})}
\end{aligned}$$

where

$$\begin{aligned}
\breve{\alpha} &= \tilde{\alpha} + \frac{1}{2} \\
\breve{\beta} &= \frac{x^{*2}}{2} + \tilde{\beta} + \frac{\tilde{\gamma}\tilde{\delta}^2}{2} - \frac{(\tilde{\gamma}\tilde{\delta} + x^*)^2}{2(\tilde{\gamma} + 1)} \\
\breve{\gamma} &= \tilde{\gamma} + 1
\end{aligned}$$

3.3.6 Example: Categorical Distribution

Maximum Likelihood

Likelihood given by categorical distribution pdf:

$$P\left(x|\boldsymbol{\lambda}\right) = \mathbf{Cat}_x[\boldsymbol{\lambda}] = \prod_{j=1}^{K} \lambda_j^{x_j} = \lambda_k$$

Apply maximum likelihood:

$$\begin{aligned}
\hat{\boldsymbol{\lambda}} &= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{i=1}^{I} P\left(x_i|\boldsymbol{\lambda}\right) && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{i=1}^{I} \mathbf{Cat}_{x_i}[\boldsymbol{\lambda}] && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{k=1}^{K} \lambda_k^{N_k} && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \sum_{k=1}^{K} N_k \log \lambda_k && s.t. \sum_k \lambda_k = 1
\end{aligned}$$

Here, N_k represents the number of times the data is classified in class k. As before, we will instead optimize log probability. Since there is a constraint $s.t. \sum_k \lambda_k = 1$, we use Lagrange multiplier to reconstruct the loss function.

$$L(\boldsymbol{\lambda}) = \sum_{k=1}^{K} N_k \log \lambda_k + v\left(\sum_{k=1}^{K} \lambda_k - 1\right)$$

Let $\nabla L(\boldsymbol{\lambda}, v) = 0$, we have the solution:

$$\hat{\lambda}_k = \frac{N_k}{\sum_{m=1}^{K} N_m}$$

Maximum a Posterior

Likelihood given by categorical distribution pdf:

$$P\left(x|\boldsymbol{\lambda}\right) = \mathbf{Cat}_x[\boldsymbol{\lambda}] = \prod_{j=1}^{K} \lambda_j^{x_j} = \lambda_k$$

Prior given by Dirichlet distribution pdf:

$$P\left(\boldsymbol{\lambda}\right) = \mathbf{Dir}_{\boldsymbol{\lambda}}[\boldsymbol{\alpha}] = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \lambda_k^{\alpha_k - 1}$$

Apply maximum a posterior:

$$\begin{aligned}
\hat{\boldsymbol{\lambda}} &= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{i=1}^{I} P\left(x_i|\boldsymbol{\lambda}\right)\ P\left(\boldsymbol{\lambda}\right) && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\, \mathbf{Cat}_{x_i}[\boldsymbol{\lambda}]\ \mathbf{Dir}_{\boldsymbol{\lambda}}[\boldsymbol{\alpha}] && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{k=1}^{K} \lambda_k^{N_k} \prod_{k=1}^{K} \lambda_k^{\alpha_k - 1} && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \prod_{k=1}^{K} \lambda_k^{N_k + \alpha_k - 1} && s.t. \sum_k \lambda_k = 1 \\
&= \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \sum_{k=1}^{K} (N_k + \alpha_k - 1) \log \lambda_k && s.t. \sum_k \lambda_k = 1
\end{aligned}$$

The loss function is very similar to maximum likelihood (same when the prior is uniform, i.e. $\alpha_{1\ldots k} = 1$). Take derivative with Lagrange multiplier, we have the solution:

$$\hat{\lambda}_k = \frac{N_k + \alpha_k - 1}{\sum_{m=1}^{K} (N_m + \alpha_m - 1)}$$

Bayesian Approach

Compute the posterior distribution using Bayes' rule:

$$\begin{aligned}
P\left(\boldsymbol{\lambda}|x_{1\ldots I}\right) &= \frac{\prod_{i=1}^{I} P\left(x_i|\boldsymbol{\lambda}\right)\ P\left(\boldsymbol{\lambda}\right)}{P\left(x_{1\ldots I}\right)} \\
&= \frac{\prod_{i=1}^{I} \mathbf{Cat}_{x_i}[\boldsymbol{\lambda}]\ \mathbf{Dir}_{\boldsymbol{\lambda}}[\boldsymbol{\alpha}]}{P\left(x_{1\ldots I}\right)} \\
&= \frac{\kappa(\boldsymbol{\alpha}, x_{1\ldots I}) \mathbf{Dir}_{\boldsymbol{\lambda}}[\tilde{\boldsymbol{\alpha}}]}{P\left(x_{1\ldots I}\right)} \\
&= \mathbf{Dir}_{\boldsymbol{\lambda}}[\tilde{\boldsymbol{\alpha}}]
\end{aligned}$$

Compute predictive distribution:

$$\begin{aligned} P\left(x^{*}|x_{1\ldots I}\right) &= \int P\left(x^{*}|\boldsymbol{\lambda}\right) P\left(\boldsymbol{\lambda}|x_{1\ldots I}\right) d\boldsymbol{\lambda} \\ &= \int \mathbf{Cat}_{x^{*}}[\boldsymbol{\lambda}]\mathbf{Dir}_{\boldsymbol{\lambda}}[\tilde{\boldsymbol{\alpha}}]d\boldsymbol{\lambda} \\ &= \int \kappa(x^{*},\tilde{\boldsymbol{\alpha}})\mathbf{Dir}_{\boldsymbol{\lambda}}[\breve{\boldsymbol{\alpha}}]d\boldsymbol{\lambda} \\ &= \kappa(x^{*},\boldsymbol{\alpha}) \end{aligned}$$

3.4 Machine Learning Models

3.4.1 Learning and Inference

In real world problems, we usually have two tasks:

1. Observe measured data, $\mathbf{x}$
2. Draw inferences from it about world, $\mathbf{w}$

and

1. When the world state $\mathbf{w}$ is *continuous*, we'll call this *regression*.
2. When the world state $\mathbf{w}$ is *discrete*, we'll call this *classification*.

We want take observations $\mathbf{x}$, and return probability distribution $P\left(\mathbf{w}|\mathbf{x}\right)$ over possible worlds compatible with data. To solve this, we need

1. A *model* that mathematically relates the visual data $\mathbf{x}$ to the world state $\mathbf{w}$. Model specifies family of relationships, particular relationship depends on parameter θ.
2. A *learning algorithm* fits parameters θ from paired training examples $\mathbf{x_i}$, $\mathbf{w_i}$.
3. An *inference algorithm* uses model to return $P\left(\mathbf{w}|\mathbf{x}\right)$ given new observation data $\mathbf{x}$.

3.4.2 Three Types of Model

We have three types of model:

1. Model contingency of the world on the data $P\left(w|x\right)$. (Discriminative Model)
2. Model joint occurrence of world and data $P\left(x,w\right)$. (Generative Model)

3. Model contingency of data on world $P(x|w)$. (Generative Model)

Within the three models, type 1 is called *Discriminative Model*. Type 2 and 3 are called *Generative Model*.

Model $P(w|x)$ - Discriminative

1. $P(w|x,\theta) = \textbf{Distrib}_w[f(x,\theta)]$
2. How to model: (a) Choose an appropriate form for $P(w)$. (b) Make parameters a function of x. (c) Function takes parameters θ that define its shape.
3. Learning algorithm: Learn parameters θ from training data x, w.
4. Inference algorithm: Just evaluate $P(w|x)$

Model $P(x,w)$ - Generative

1. $P(z|\theta) = \textbf{Distrib}_z[\theta]$
2. How to model: (a) Concatenate x and w to make $z = [x^\top, w^\top]^\top$. (b) Model the pdf of z. (c) pdf takes parameters θ that define its shape.
3. Learning algorithm: Learn parameters θ from training data x, w.
4. Inference algorithm: Compute $P(w|x)$ using Bayes' rule $P(w|x) = \frac{P(x,w)}{P(x)} = \frac{P(x,w)}{\int P(x,w)dw}$.

Model $P(x|w)$ - Generative

1. $P(x|w,\theta) = \textbf{Distrib}_x[f(w,\theta)]$
2. How to model: (a) Choose an appropriate form for $P(x)$. (b) Make parameters a function of w. (c) Function takes parameters θ that define its shape.
3. Learning algorithm: Learn parameters θ from training data x, w.
4. Define prior $P(w)$ and then compute $P(w|x)$ using Bayes' rule $P(w|x) = \frac{P(x|w)P(w)}{\int P(x|w)P(w)dw}$.

3.4.3 Example: Regression

Consider a simple case:

1. We make a univariate continuous measurement x.
2. Use this to predict a univariate continuous state w.

Model $P(w|x)$ - Discriminative

1. $P(w|x, \boldsymbol{\theta}) = \mathbf{Norm}_w[\phi_0 + \phi_1 x, \sigma^2], \boldsymbol{\theta} = \{\phi_0, \phi_1, \sigma^2\}$
2. How to model: (a) Choose normal distribution over w. (b) Make mean μ linear function of x (variance constant). (c) Parameters are ϕ_0 (y-offset), ϕ_1 (slope), σ^2 (variance). This model is called *linear regression*.
3. Learning algorithm: Learn $\boldsymbol{\theta}$ from training data x, w. e.g. MAP:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} P(\boldsymbol{\theta}|w_{1...I}, x_{1...I}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} P(w_{1...I}|x_{1...I}, \boldsymbol{\theta}) P(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{I} P(w_i|x_i, \boldsymbol{\theta}) P(\boldsymbol{\theta})\end{aligned}$$

4. Inference algorithm:Just evaluate $P(w|x)$ for new data x.

Model $P(x, w)$ - Generative

1. $P(x, w|\boldsymbol{\theta}) = \mathbf{Norm}_{x,w}[\boldsymbol{\mu}, \boldsymbol{\Sigma}], \boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$
2. How to model: (a) Concatenate x and w to make $z = [x^\top, w^\top]^\top$. (b) Model the pdf of z as normal distribution. (c) pdf makes parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ that define its shape.
3. Learning algorithm: Learn parameters θ from training data x, w.
4. Inference algorithm: Compute $P(w|x)$ using Bayes' rule $P(w|x) = \frac{P(x,w)}{P(x)} = \frac{P(x,w)}{\int P(x,w)dw}$.

Model $P(x|w)$ - Generative

1. $P(x|w, \boldsymbol{\theta}) = \mathbf{Norm}_x[\phi_0 + \phi_1 w, \sigma^2], \boldsymbol{\theta} = \{\phi_0, \phi_1, \sigma^2\}$
2. How to model: (a) Choose normal distribution over x. (b) Make mean μ linear function of w (variance constant). (c) Parameters are ϕ_0, ϕ_1, σ^2.
3. Learning algorithm: Learn $\boldsymbol{\theta}$ from training data x, w. e.g. MAP
4. Inference algorithm: Compute $P(w|x)$ using Bayes' rule $P(w|x) = \frac{P(x,w)}{P(x)} = \frac{P(x,w)}{\int P(x,w)dw}$.

3.4.4 Example: Classification

Consider a simple case:

1. We make a univariate continuous measurement x.
2. Use this to predict a discrete binary world $w \in \{0, 1\}$.

Model $P(w|x)$ - Discriminative

1. $P(w|x, \boldsymbol{\theta}) = \mathbf{Bern}_w[\sigma(\phi_0 + \phi_1 x)], \boldsymbol{\theta} = \phi_0, \phi_1$
2. How to model: (a) Choose Bernoulli distribution for $P(w)$. (b) Make parameters a sigmoid-activated function of x. (c) Function takes parameters ϕ_0 and ϕ_1. This model is called *logistic regression*.
3. Learning algorithm: Learning by standard methods, e.g. ML, MAP, Bayesian Approach.
4. Inference algorithm: Just evaluate $P(w|x)$.

Model $P(x, w)$ - Generative

Can't build this mode very easily:

1. Concatenate continuous vector x and discrete w to make z.
2. No obvious probability distribution to model joint probability of discrete and continuous.

Model $P(x|w)$ - Generative

1. $P(x|w, \boldsymbol{\theta}) = \mathbf{Norm}_x[\mu_w, \sigma_w^2], \boldsymbol{\theta} = \{\mu_0, \mu_1, \sigma_0^2, \sigma_1^2\}$
2. How to model: (a) Choose a Normal distribution for $P(x)$. (b) Make parameters a function of discrete binary w. (c) Function takes parameters μ_0, μ_1, σ_0^2, σ_1^2 that define its shape.
3. Learning algorithm: Learning by standard methods, e.g. ML, MAP, Bayesian Approach.
4. Define prior $P(w)$ and then compute $P(w|x)$ using Bayes' rule $P(w|x) = \frac{P(x|w)P(w)}{\int P(x|w)P(w)dw}$.

3.5 Overview of Common Algorithms

Properties of common machine learning methods:

Method[1]	Problem[2]	Model[3]	Learning[4]	Loss Function	Algorithm[5]
Perceptron	BC	D			SGD
K-NN	MC, R	D			
Naive Bayes	MC	G	ML, MAP	$-\log P(w\|x)$	Bayes, EM
Decision Tree	MC, R	D	NML	$-\log P(w\|x)$	
LR	MC	D	ML, NML	$\log(1+exp(-wf(x)))$	SGD, QN
SVM	BC	D		$[1-wf(x)]_+$	SMO
Boosting	BC	D		$exp(-wf(x))$	
EM			ML, MAP	$-\log P(w\|x)$	Iteration
HMM	T	G	ML, MAP	$-\log P(w\|x)$	Bayes, EM
CRF	T	D	ML, NML	$-\log P(w\|x)$	SGD, QN

[1] K-NN=K-Nearest Neighbors, LR=Logistic Regression, SVM=Support Vector Machine, HMM=Hidden Markov Model, CRF=Conditional Random Field

[2] BC=Binary Classification, MC=Multi-class Classification, R=Regression, T=Tagging

[3] D=Discriminative Model, G=Generative Model

[4] ML=Maximum Likelihood, NML=Normalized ML MAP=Maximum a Posterior

[5] SGD=Stochastic Gradient Descent, QN=Quasi-Newton

Chapter 4

Matrix Factorization

4.1 SVD

4.2 PCA

4.3 (N)NMF

Chapter 5

K-Nearest Neighbors

5.1 Simple K-NN

5.2 Fast K-NN Computation

Chapter 6

Linear Regression

6.1 Basic Model

1. Discriminative, Regression
2. $P\left(w_i|\mathbf{X}_i, \boldsymbol{\theta}\right) = \mathbf{Norm}_{w_i}[\phi_0 + \boldsymbol{\phi}^\top \mathbf{X}_i, \sigma^2]$
3. (Neater Notation) $\mathbf{X}_i \leftarrow [1 \quad \mathbf{X}_i^\top]^\top$, $\boldsymbol{\phi} \leftarrow [\phi_0 \quad \boldsymbol{\phi}^\top]^\top$
   $P\left(w_i|\mathbf{X}_i, \boldsymbol{\theta}\right) = \mathbf{Norm}_{w_i}[\boldsymbol{\phi}^\top \mathbf{X}_i, \sigma^2]$
4. (Combining Equations) $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_I]$
   $P\left(\boldsymbol{w}|\mathbf{X}, \boldsymbol{\theta}\right) = \mathbf{Norm}_{\boldsymbol{w}}[\mathbf{X}^\top \boldsymbol{\phi}, \sigma^2 \mathbf{I}]$
5. Learning with Maximum Likelihood: $\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}}\, P\left(\boldsymbol{w}|\mathbf{X}, \boldsymbol{\theta}\right) = \text{argmax}_{\boldsymbol{\theta}} \log P\left(\boldsymbol{w}|\mathbf{X}, \boldsymbol{\theta}\right)$, result:

$$\hat{\boldsymbol{\phi}} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\boldsymbol{w}$$
$$\hat{\sigma}^2 = \frac{(\boldsymbol{w} - \mathbf{X}^\top\boldsymbol{\phi})^\top(\boldsymbol{w} - \mathbf{X}^\top\boldsymbol{\phi})}{\mathbf{I}}$$

6.2 Bayesian Regression

Parameter $\boldsymbol{\phi}$

Likelihood $P\left(\boldsymbol{w}|\mathbf{X}, \boldsymbol{\theta}\right) = \mathbf{Norm}_{\boldsymbol{w}}[\mathbf{X}^\top \boldsymbol{\phi}, \sigma^2 \mathbf{I}]$

Prior $P\left(\boldsymbol{\phi}\right) = \mathbf{Norm}_{\boldsymbol{\phi}}[\mathbf{0}, \sigma_p^2 \mathbf{I}]$

Posterior

$$\begin{aligned} P\left(\boldsymbol{\phi}|\mathbf{X}, \boldsymbol{w}\right) &= \frac{P\left(\boldsymbol{w}|\mathbf{X}, \boldsymbol{\phi}\right) P\left(\boldsymbol{\phi}|\mathbf{X}\right)}{P\left(\boldsymbol{w}|\mathbf{X}\right)} \\ &= \mathbf{Norm}_{\phi}\left[\frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\boldsymbol{w}, \mathbf{A}^{-1}\right] \end{aligned}$$

where

$$\begin{aligned} \mathbf{A} &= \frac{1}{\sigma^2}\mathbf{X}\mathbf{X}^{\top} + \frac{1}{\sigma_p^2}\mathbf{I} \\ \mathbf{A}^{-1} &= \sigma_p^2\mathbf{I}_D - \sigma_p^2\mathbf{X}\left(\mathbf{X}^{\top}\mathbf{X} + \frac{\sigma^2}{\sigma_p^2}\mathbf{I}_I\right)^{-1}\mathbf{X}^{\top} \end{aligned}$$

Inference (Bayesian Approach)

$$\begin{aligned} P\left(w^*|\boldsymbol{x}^*, \mathbf{X}, \boldsymbol{w}\right) &= \int P\left(w^*|\boldsymbol{x}^*, \boldsymbol{\phi}\right) P\left(\boldsymbol{\phi}|\mathbf{X}, \boldsymbol{w}\right) d\boldsymbol{\phi} \\ &= \int \mathbf{Norm}_{w^*}[\boldsymbol{\phi}^{\top}\boldsymbol{x}^*, \sigma^2]\mathbf{Norm}_{\boldsymbol{\phi}}\left[\frac{1}{\sigma^2}\mathbf{A}^{-1}\mathbf{X}\boldsymbol{w}, \mathbf{A}^{-1}\right] d\boldsymbol{\phi} \\ &= \mathbf{Norm}_{w^*}\left[\frac{1}{\sigma^2}\boldsymbol{x}^{*\top}\mathbf{A}^{-1}\mathbf{X}\boldsymbol{w}, \boldsymbol{x}^{*\top}\mathbf{A}^{-1}\boldsymbol{x}^* + \sigma^2\right] \end{aligned}$$

Parameter σ^2

$$\begin{aligned} P\left(\boldsymbol{w}|\mathbf{X}, \sigma^2\right) &= \int P\left(\boldsymbol{w}|\mathbf{X}, \boldsymbol{\phi}, \sigma^2\right) P\left(\boldsymbol{\phi}\right) d\boldsymbol{\phi} \\ &= \int \mathbf{Norm}_{\boldsymbol{w}}[\mathbf{X}^{\top}\boldsymbol{\phi}, \sigma^2\mathbf{I}]\mathbf{Norm}_{\boldsymbol{\phi}}[\mathbf{0}, \sigma_p^2\mathbf{I}]d\boldsymbol{\phi} \\ &= \mathbf{Norm}_{\boldsymbol{w}}[\mathbf{0}, \sigma_p^2\mathbf{X}^{\top}\mathbf{X} + \sigma^2\mathbf{I}] \end{aligned}$$

6.3 Non-linear Regression

6.4 Kernel Trick & Gaussian Processes

6.5 Sparse Linear Regression

6.6 Dual Linear Regression

6.7 Relevance Vector Regression

Chapter 7

Logistic Regression

7.1 Logistic Regression

7.2 Non-linear Logistic Regression

7.3 Kernel Trick & Gaussian Process Classification

7.4 Multi-class Classification

Chapter 8

Support Vector Machines

8.1 Geometric Margins

8.2 Primal & Dual Problems

8.3 Support Vectors

8.4 Slack Variables

8.5 Hinge Loss

8.6 Non-linear SVMs

8.7 Kernel Trick

Chapter 9

EM Algorithm

9.1 Expectation Maximization

9.2 Example: Mixture of Gaussians

9.3 Example: t-distributions

9.4 Example: Factor Analysis

Chapter 10

Bagging & Boosting

10.1 Ensemble Methods

10.2 Bagging

10.3 CART

10.4 ID3

10.5 C4.5

10.6 Random Forest

10.7 Boosting

10.8 Adaboost

Chapter 11

Clustering

11.1 K-Means

11.2 Spectral Clustering

Chapter 12

Graphical Models & Markov Network

12.1 Graph Definitions

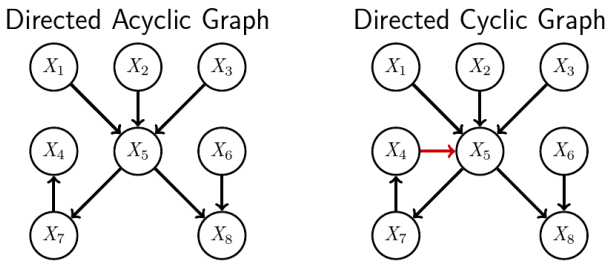
12.1.1 Graph

Graph A graph consists of nodes (vertices) and undirected or directed links (edges) between nodes.

Path A path from X_i to X_j is a sequence of connected nodes starting at X_i and ending at X_j.

12.1.2 Directed Graph

Directed Graphs Graphs that all the edges are directed.



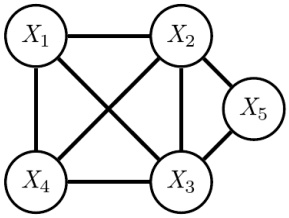
Directed Acyclic Graph (DAG) Graph in which by following the direction of the arrows a node will never be visited more than once.

Parents and Children X_i is a parent of X_j if there is a link from X_i to X_j. X_i is a child of X_j if there is a link from X_j to X_i.

Ancestors and Descendants The ancestors of a node X_i are the nodes with a directed path ending at X_i. The descendants of X_i are the nodes with a directed path beginning at X_i.

12.1.3 Undirected Graph

Undirected Graph Graph that all the edges are undirected.



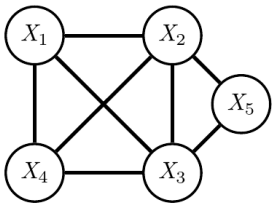
Clique A clique is a fully connected subset of nodes. (X_1, X_2, X_4) forms a (non-maximal) clique.

Maximal Clique Clique which is not a subset of a larger clique. (X_1, X_2, X_3, X_4) and (X_2, X_3, X_5) are both maximal cliques.

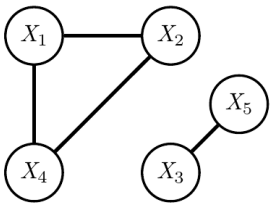
12.1.4 Connectivity

Connected Graph There is a path between every pair of vertices.

Connected Components In a non-connected graph, the connected components are the connected-subgraphs. (X_1, X_2, X_4) and (X_3, X_5) are the two connected components.



Connected Graph

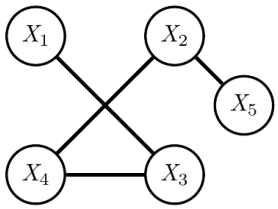


Connected Components

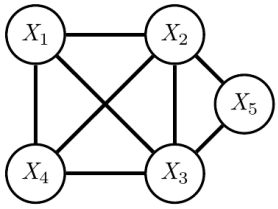
12.1.5 Connectedness

Singly-connected There is only one path from any node a to another node b.

Multiply-connected A graph is multiply-connected if it is not singly-connected.



Singly-connected

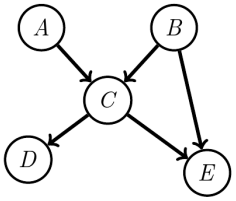


Multiply-connected

12.2 Belief Networks

12.2.1 Definition

A belief network is a directed acyclic graph in which each node is associated with the conditional probability of the node given its parents. The joint distribution is obtained by taking the product of the conditional probabilities.



$p(E|B, C)$

$$P(A, B, C, D, E) =$$
$$P(A)\, P(B)\, P(C|A, B)\, P(D|C)\, P(E|B, C)$$

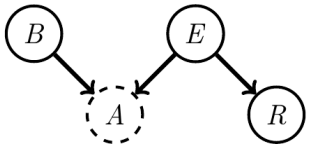
12.2.2 Uncertain Evidence

Definition In soft/uncertain evidence the variable is in more than one state, with the strength of out belief about each state being given by probabilities. For example, if y has the states $\text{dom}(y) = \{red, blue, green\}$, the vector (0.6, 0.1, 0.3) could represent the probabilities of the respective states.

Hard Evidence We are certain that a variable is in a particular state. In this state, all the probability mass is in one of the vector components, (0, 0, 1).

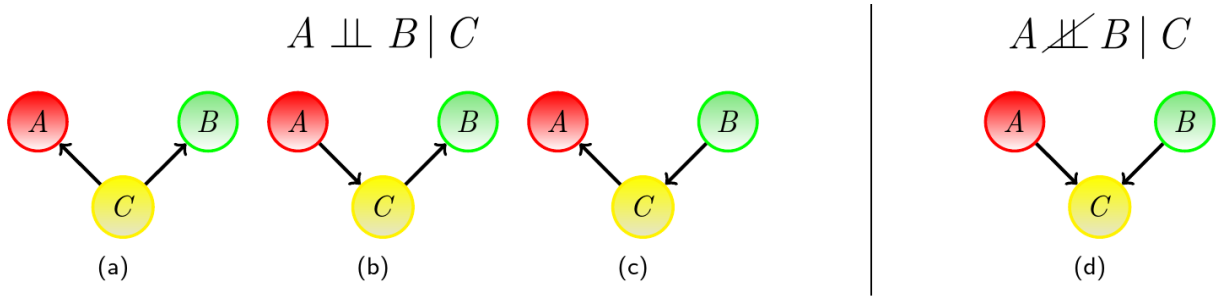
Inference Inference with soft-evidence can be achieved using Bayes' rule. Writing the soft-evidence as $\tilde{y}$, we have $P(x|\tilde{y}) = \sum_y P(x|y)\, P(y|\tilde{y})$, where $P(y = i|\tilde{y})$ represents the probability that y is in state i under the soft-evidence.

Jeffrey's Rule For variables x, y and $P_1(x, y)$, how do we form a joint distribution given soft-evidence $\tilde{y}$? (a) From the conditional we first define $P_1(x|y) = \frac{P_1(x,y)}{\sum_x P_1(x,y)}$. (b) Define the joint. The soft-evidence $P(y|\tilde{y})$ then defines a new joint distribution $P_2(x, y|\tilde{y}) = P_1(x|y)P_1(y|\tilde{y})$. One can therefore view soft-evidence as defining a new joint distribution. We use a dashed circle to represent a variable in an uncertain state.



12.2.3 Independence

Conditionally Independent



In (a), (b) and (c), A, B are conditionally independent given C.

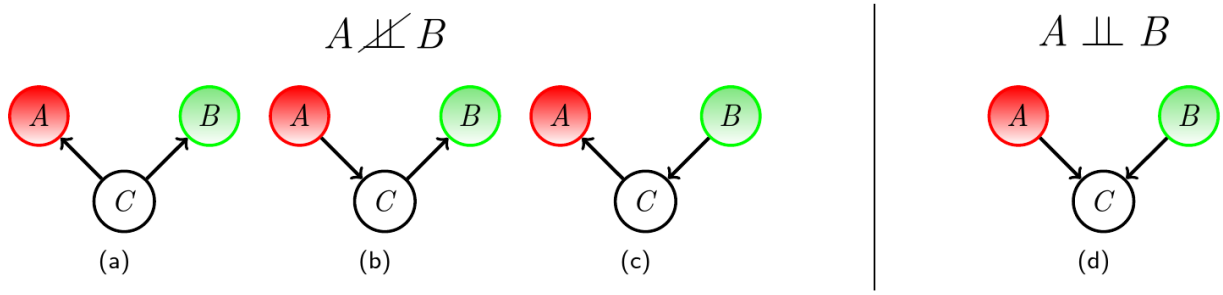
(a) $P(A,B|C) = \frac{P(A,B,C)}{P(C)} = \frac{P(A|C)P(B|C)P(C)}{P(C)} = P(A|C)\,P(B|C)$

(b) $P(A,B|C) = \frac{P(A,B,C)}{P(C)} = \frac{P(A)P(C|A)P(B|C)}{P(C)} = \frac{P(A,C)P(B|C)}{P(C)} = P(A|C)\,P(B|C)$

(c) $P(A,B|C) = \frac{P(A,B,C)}{P(C)} = \frac{P(A|C)P(C|B)P(B)}{P(C)} = \frac{P(A|C)P(B,C)}{P(C)} = P(A|C)\,P(B|C)$

In (d) the variables A, B are conditionally dependent given C, $P(A,B|C) \propto P(C|A,B)\,P(A)\,P(B)$.

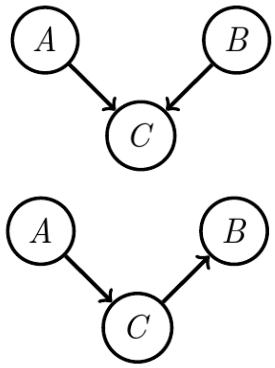
Marginally Dependent



In (a), (b) and (c), the variables A, B are marginally dependent. In (d) the variables A, B are marginally independent.

$$P(A,B) = \sum_C P(A,B,C) = \sum_C P(A)\,P(B)\,P(C|A,B) = P(A)\,P(B)$$

Colliders



If C has more than one incoming link, then $A \not\perp\!\!\!\perp B|C$. In this case C in called *collider*.

If C has at most one incoming link, then $A \perp\!\!\!\perp B|C$ and $A \not\perp\!\!\!\perp B$. In this case C is called *non-collider*.

12.2.4 General Rule for Independence in Belief Networks

Given three sets of nodes $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{C}$, if all paths from any element of $\mathcal{X}$ to any element of $\mathcal{Y}$ are blocked by $\mathcal{C}$, then $\mathcal{X}$ and $\mathcal{Y}$ are conditionally independent given $\mathcal{C}$. A path $\mathcal{P}$ is blocked by $\mathcal{C}$ if at least one of the following conditions is satisfied:

1. There is a collider in the path $\mathcal{P}$ such that neither the collider nor any of its descendants is in the conditioning set $\mathcal{C}$.

2. There is a non-collider in the path $\mathcal{P}$ that is in the conditioning set $\mathcal{C}$.

Independence of $\mathcal{X}$ and $\mathcal{Y}$

When the conditioning set is empty $\mathcal{C} = \emptyset$, then a path $\mathcal{P}$ from an element of $\mathcal{X}$ to an element of $\mathcal{Y}$ is blocked if there is a collider on the path. Hence $\mathcal{X}$ and $\mathcal{Y}$ are independent if every path from an element of $\mathcal{X}$ to any element of $\mathcal{Y}$ has a collider.

d-connected

We use the term that $\mathcal{X}$ and $\mathcal{Y}$ are "d-connected" by $\mathcal{Z}$ if there is any path from $\mathcal{X}$ to $\mathcal{Y}$ that is not blocked by $\mathcal{Z}$. If $\mathcal{Z}$ is the empty set then we just say that $\mathcal{X}$ and $\mathcal{Y}$ are d-connected.

Separation and Independence

Note first that d-separation and connection are properties of the graph (not of the distribution). d-separation implies that $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$, but d-connection does not necessarily imply conditional dependence. That is, for any distribution in which $\mathcal{X}$ and $\mathcal{Y}$ are "d-separated" by $\mathcal{Z}$, then no matter what the settings of the conditional tables are, then conditional independence holds, namely $\mathcal{X} \perp\!\!\!\perp \mathcal{Y}|\mathcal{Z}$.

12.2.5 Markov Equivalence

Skeleton

Formed from a graph by removing the arrows.

Immorality

An immorality in a DAG is a configuration of three nodes, A,B,C such that C is a child of both A and B, with A and B not directly connected.

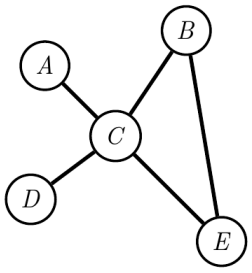
Markov Equivalence

Markov equivalence Two graphs represent the same set of independence assumptions if and only if they have the same skeleton and the same set of immoralities.

12.3 Markov Networks

12.3.1 Definition

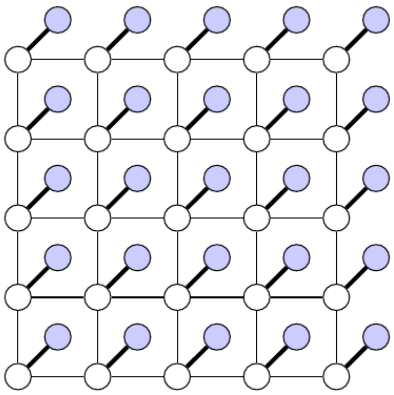
A Markov Network is an undirected graph in which there is a potential (non-negative function) ψ defined on each maximal clique.



$$P(A,B,C,D,E) = \frac{1}{Z}\psi(A,C)\psi(C,D)\psi(B,C,E)$$
$$Z = \sum_{A,B,C,D,E} \psi(A,C)\psi(C,D)\psi(B,C,E)$$

12.3.2 Examples

Binary Image



$X = \{X_i, i = 1,\ldots,D\}$ $X_i \in \{-1,1\}$: clean pixel

$Y = \{Y_i, i = 1,\ldots,D\}$ $Y_i \in \{-1,1\}$: corrupted pixel

$\phi(Y_i, X_i) = e^{\gamma X_i Y_i}$: encourage Y_i and X_i to be similar

$\psi(X_i, X_j) = e^{\beta X_i X_j}$: encourage the image to be smooth

$$P(X,Y) \propto \left[\prod_{i=1}^{D} \phi(Y_i, X_i)\right]\left[\prod_{i\sim j} \psi(X_i, X_j)\right]$$

Boltzmann Machine

The Ising Model

12.3.3 Independence

12.3.4 Expressiveness of Markov and Belief Networks

12.3.5 Factor Graphs

12.4 Markov Chains

12.5 Hidden Markov Models

Appendix A

Bayesian Statistics

A.1 Bayesian Inference

A.2 Prior Distributions

Appendix B

Statistical Assessment

B.1 Hypothesis Testing

B.1.1 Testing Basics

Null Hypothesis H_0 The hypothesis we would like to test.

Alternative Hypothesis H_1 An alternative result when H_0 is rejected. In most cases the alternative hypothesis is simply the negation of the null hypothesis.

P-value The p-value is the probability of observing a test statistic, X, as or more extreme than the value x seen in the data, under the assumption that the null hypothesis, H_0, is true. The p-value is most certainly not the probability of H_0 being true.

False Positives (Type I Error) Rejecting H_0 when it is true.

False Negatives (Type II Error) Not rejecting H_0 when it is false. (N.B. Not rejecting H_0 is not the same as accepting H_0)

Power The power of a hypothesis test is the probability of avoiding a false negative.

B.1.2 Testing Procedure

A common testing procedure includes the following steps:

1. Specify a null hypothesis (H_0) and alternative hypothesis (H_1).
   e.g.
   $H_0 : \theta = 0.5$ The proportion of males and females is identical.
   $H_1 : \theta < 0.5$ There is a smaller proportion of females than males.

2. Specify the level of the test.
   e.g. a common level=0.05
   - Bearing in mind the need to balance probabilities of Type I and Type II errors.
   - Reducing the level reduces the probability of a Type I error.
   - Increasing the level reduces the probability of a Type II error.

3. Specify a suitable test statistic.
   e.g. $X =$ The number of females $= 15$.

4. Determine the distribution of the test statistic under H_0.
   e.g. $X \sim \mathbf{Bin}(40, 0.5)$

5. Determine what it means to be "more extreme" by considering H_0 and H_1.
   e.g. $H_1 : \theta < 0.5$, so smaller values of X are more extreme.

6. Determine the corresponding p-value.
   e.g. $p = p\,(X \leq 15) = 0.077$

7. Reject H_0 if the p-value is less than the level of the test.
   e.g. $p > 0.05$, so we fail to reject H_0 in this instance. Conclude that the proportion of females and males is identical.

An alternative procedure is that rather than determining a p-value, we may determine a critical region for the test statistic – the set of all test statistic values which would cause us to reject H_0.
e.g. level $= 0.05$, $p\,(X \leq 15) = 0.077$, $p\,(X \leq 14) = 0.04 \Rightarrow CR = \{0, 1, 2, \ldots, 14\}$.

We may therefore simply compare our observed value to the critical region to judge whether to reject H_0.

B.1.3 Power Investigation

B.1.4 Useful Tests

B.2 Confidence Intervals

B.3 Bootstrap