Imperial College
London

NOTES

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Mathematics For Machine Learning

Author:
W (CID: your college-id number)

Date: November 30, 2019

1 Linear Algebra

1.1 Vector Space

Given a set $\mathcal{G}$ and an operation $\otimes$ on $\mathcal{G}$, $(\mathcal{G}, \otimes)$ is called a **group** if the following holds:

- $\forall x, y \in \mathcal{G} : \; x \otimes y \in \mathcal{G}$
- $\forall x, y, z \in \mathcal{G} : \; (x \otimes y) \otimes z = x \otimes (y \otimes z)$
- $\exists e \in \mathcal{G} \; \forall x \in \mathcal{G} : \; x \otimes e = e \otimes x = x$
- $\forall x \in \mathcal{G} \; \exists y \in \mathcal{G} : \; x \otimes y = y \otimes x = e$

The group is an **Abelian group** if $\forall x, y \in \mathcal{G} : \; x \otimes y = y \otimes x$.

A real value vector space is a set $\mathcal{V}$ with two operations:

$$+ : \mathcal{V} \times \mathcal{V} \to \mathcal{V} \qquad \cdot : \mathbb{R} \times \mathcal{V} \to \mathcal{V}$$

Where $+$ is vector addition and $\cdot$ is multiplication by scalar. $(\mathcal{V}, +)$ is an Abelian group with neutral element $e = \mathbf{0}$.

1.2 Basis and Rank

Consider a vector space $(\mathcal{V}, +, \cdot)$ and a set of vectors $\mathcal{A}$. $\mathcal{A}$ is the generating set of $\mathcal{V}$ is every $v \in \mathcal{V}$ can be expressed as a linear combination of vectors in $\mathcal{A}$. The **basis** of $\mathcal{V}$ is the minimal generating set.

The **rank** of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, denoted as $rk(\boldsymbol{A})$, equals the number of linearly independent columns or rows in matrix $\boldsymbol{A}$. A matrix has **full rank** if $rk(\boldsymbol{A}) = min(m, n)$.

2 Analytic Geometry

2.1 Orthonormal Basis

Given the basis $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ of a vector space $\mathcal{V}$, the basis is called **orthogonal basis** if every basis vector is orthogonal to each other, i.e. their dot product is 0

$$\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = 0, \;\; i \neq j$$

Additionally, the basis is called **orthonormal basis** if it is an orthogonal basis and each basis vector has length/norm 1.

$$\langle \boldsymbol{b}_i, \boldsymbol{b}_i \rangle = 1$$

A matrix $\boldsymbol{A}$ is an **orthogonal matrix** if its rows/columns form an orthonormal basis. A property of orthogonal matrix is that

$$\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{I}$$

It follows that orthogonal matrices are always invertible and $\boldsymbol{A}^{-1} = \boldsymbol{A}^\top$

3 Matrix Decomposition

3.1 Eigenvalues and Eigenvectors

Eigenvalues and Eigenvectors are only defined for square matrices. Given a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, then $\lambda \in \mathbb{R}$ is an eigenvalue of $\boldsymbol{A}$ and $\boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$ is an eigenvector of $\boldsymbol{A}$ if $\boldsymbol{Ax} = \lambda \boldsymbol{x}$. This means the transformation of $\boldsymbol{A}$ does not change the direction of eigenvectors $\boldsymbol{x}$ but only their magnitude.

We can find the eigenvalues of a matrix $\boldsymbol{A}$ by finding the roots of the characteristic polynomial $p_{\boldsymbol{A}}(\lambda)$ of $\boldsymbol{A}$

$$p_{\boldsymbol{A}}(\lambda) \doteq det(\boldsymbol{A} - \lambda \boldsymbol{I})$$

Once we find the eigenvalues, we can obtain the eigenvectors by solving the set of linear equations. If $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$ associated with eigenvalue λ, then for any $c \in \mathbb{R} \setminus \{0\}$ $c\boldsymbol{x}$ is also an eigenvector of $\boldsymbol{A}$

$$\boldsymbol{A}(c\boldsymbol{x}) = c\boldsymbol{A}\boldsymbol{x} = c\lambda\boldsymbol{x} = \lambda(c\boldsymbol{x})$$

3.1.1 Spectral Theorem

Given a **symmetric** matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, then the eigenvalues of $\boldsymbol{A}$ are real and the eigenvectors of $\boldsymbol{A}$ forms an orthonormal basis

3.1.2 Determinants and Traces

The determinant of a matrix $\boldsymbol{A}$ is the product of its eigenvalues

$$det(\boldsymbol{A}) = \prod_{n=1}^{N} \lambda_n$$

The trace of a matrix $\boldsymbol{A}$ is the sum of its eigenvalues

$$tr(\boldsymbol{A}) = \sum_{n=1}^{N} \lambda_n$$

3.2 Eigendecomposition and Diagonalisation

A diagonal matrix $\boldsymbol{D}$ is a matrix with all values 0 except for the main diagonal. Diagonal matrices are useful because they allow fast computations for certain operations:

- The determinant $det(\boldsymbol{D})$ is the product of its diagonal entries
- The matrix power $\boldsymbol{D}^k$ is each diagonal element raised to power k
- The inverse $\boldsymbol{D}^{-1}$ is the reciprocal of each diagonal element

A squared matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is **diagonalisable** if there is an invertible matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ such that $\boldsymbol{D} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$. It turns out that the columns of $\boldsymbol{P}$ are the eigenvectors of $\boldsymbol{A}$.

We define $\boldsymbol{D}$ as the diagonal matrix where the diagonal values are the eigenvalues of $\boldsymbol{A}$, and we define $\boldsymbol{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$ where each $\boldsymbol{p}_i$ is an eigenvector of $\boldsymbol{A}$. Then

$$\boldsymbol{A}\boldsymbol{P} = \boldsymbol{A}\begin{bmatrix}\boldsymbol{p}_1 & \ldots & \boldsymbol{p}_n\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}\boldsymbol{p}_1 & \ldots & \boldsymbol{A}\boldsymbol{p}_n\end{bmatrix}$$

$$\boldsymbol{P}\boldsymbol{D} = \begin{bmatrix}\boldsymbol{p}_1 & \ldots & \boldsymbol{p}_n\end{bmatrix}\begin{bmatrix}\lambda_1 & \ldots & 0\\ \vdots & \ddots & \vdots\\ 0 & \ldots & \lambda_n\end{bmatrix} = \begin{bmatrix}\lambda_1\boldsymbol{p}_1 & \ldots & \lambda_n\boldsymbol{p}_n\end{bmatrix}$$

$$\because \boldsymbol{A}\boldsymbol{p}_1 = \lambda_1\boldsymbol{p}_1$$

$$\because \quad \vdots \quad = \quad \vdots$$

$$\because \boldsymbol{A}\boldsymbol{p}_n = \lambda_n\boldsymbol{p}_n$$

$$\therefore \boldsymbol{A}\boldsymbol{P} = \boldsymbol{P}\boldsymbol{D}$$

Eigendecomposition is factoring a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ into

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$$

In the case where $\boldsymbol{A}$ is symmetric, from the spectral theorem we know the eigenvectors of $\boldsymbol{A}$ forms an orthonormal basis, meaning $\boldsymbol{P}^{-1} = \boldsymbol{P}^{\top}$, hence

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{\top}$$

3.3 Singular Value Decomposition (SVD)

Singular Value Decomposition can be applied to all matrices, not just squared matrices, and it always exists. Given a rectangular matrix $\boldsymbol{A}$, SVD decomposes $\boldsymbol{A}$ into

$$\boldsymbol{A}_{m\times n} = \boldsymbol{U}_{m\times m}\boldsymbol{\Sigma}_{m\times n}\boldsymbol{V}^{\top}_{n\times n}$$

Where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices and Σ is a diagonal matrix.

Σ is the singular matrix and the diagonal entries are called the singular values. The singular matrix has the same shape as $\boldsymbol{A}$ hence needs additional zero padding.

$$\Sigma = \begin{bmatrix}\sigma_1 & \ldots & 0\\ \vdots & \ddots & \vdots\\ 0 & \ldots & \sigma_n\\ 0 & \ldots & 0\\ \vdots & \ddots & \vdots\\ 0 & \ldots & 0\end{bmatrix} \qquad \Sigma = \begin{bmatrix}\sigma_1 & \ldots & 0 & 0 & \ldots & 0\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ 0 & \ldots & \sigma_n & 0 & \ldots & 0\end{bmatrix}$$

From Eigendecomposition, we saw that a symmetric matrix $\boldsymbol{S}$ can be decomposed into $\boldsymbol{S} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{\top}$. Also, we can construct a symmetric, positive semidefinite matrix

from a rectangular matrix $\boldsymbol{A}$ by $\boldsymbol{A}\boldsymbol{A}^\top$ or $\boldsymbol{A}^\top\boldsymbol{A}$.

Therefore, constructing the SVD consists of find the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{A}^\top\boldsymbol{A}$. The eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$ make up the columns of $\boldsymbol{U}$, and the eigenvectors of $\boldsymbol{A}^\top\boldsymbol{A}$ make up the columns of $\boldsymbol{V}^\top$. The singular values in Σ are the square roots of eigenvalues from $\boldsymbol{A}\boldsymbol{A}^\top$ or $\boldsymbol{A}^\top\boldsymbol{A}$.

$$\boldsymbol{A}^\top\boldsymbol{A} = (\boldsymbol{U}\Sigma\boldsymbol{V}^\top)^\top(\boldsymbol{U}\Sigma\boldsymbol{V}^\top) = \boldsymbol{V}\Sigma^\top\boldsymbol{U}^\top\boldsymbol{U}\Sigma\boldsymbol{V}^\top = \boldsymbol{V}\Sigma^\top\Sigma\boldsymbol{V}^\top$$
$$\boldsymbol{A}\boldsymbol{A}^\top = (\boldsymbol{U}\Sigma\boldsymbol{V}^\top)(\boldsymbol{U}\Sigma\boldsymbol{V}^\top)^\top = \boldsymbol{U}\Sigma\boldsymbol{V}^\top\boldsymbol{V}\Sigma^\top\boldsymbol{U}^\top = \boldsymbol{U}\Sigma\Sigma^\top\boldsymbol{U}^\top$$

4 Vector Calculus

Many algorithms in machine learning optimise an objective function with respect to a set of desired model parameters, and so calculus plays an important role in machine learning.

A function f relates two quantities to each other. In machine learning, the input is a vector $\boldsymbol{x} \in \mathbb{R}^D$ with D features. Here, $\mathbb{R}^D$ is the **domain** of f and the values $f(\boldsymbol{x})$ are **image** of f.

4.1 Differentiate of Univariate Functions

Given a variable $x \in \mathbb{R}$ and a function $f : \mathbb{R} \to \mathbb{R}$. The **derivative** for f with respect to x, for $h > 0$

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

4.2 Partial Differentiations

In the following, we consider the general case of f with input $\boldsymbol{x} \in \mathbb{R}^D$.

Tthe gradient of f with respect to $\boldsymbol{x}$ is a collection of partial derivatives. Each partial derivative is obtained differentiating one variable at a time and keeping the others constant.

The gradient of f is a row vector $\mathbb{R}^{1\times D}$ is called the **Jacobian**.

$$\frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_D} \end{bmatrix} \in \mathbb{R}^{1\times D}$$

4.3 Vector-Valued Functions

In the following, we consider the general case where $\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^E$ with input $\boldsymbol{x} \in \mathbb{R}^D$.

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_E(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^E$$

The derivative of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$ is a $E \times D$ matrix called the **Jacobian**.

$$\frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_D} \\ \vdots & & \vdots \\ \frac{\partial f_E(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_E(\boldsymbol{x})}{\partial x_D} \end{bmatrix} \in \mathbb{R}^{E \times D}$$

Example

Find the derivatives $d\boldsymbol{f}/d\boldsymbol{x}$ and $d\boldsymbol{f}/d\boldsymbol{A}$.

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \qquad \boldsymbol{f} \in \mathbb{R}^M \qquad \boldsymbol{A} \in \mathbb{R}^{M \times N} \qquad \boldsymbol{x} \in \mathbb{R}^N$$

Writing down each function $f_i(\boldsymbol{x})$ makes it easier to find the gradients.

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{N} A_{ij} x_j$$

Since function $\boldsymbol{f}$ has dimensions M and the target $\boldsymbol{x}$ has dimensions N. The derivative $d\boldsymbol{f}/d\boldsymbol{x}$. has the shape $M \times N$. Because

$$\frac{\partial f_i}{\partial x_j} = A_{ij} \Rightarrow \frac{d\boldsymbol{f}}{d\boldsymbol{x}} = \boldsymbol{A}$$

Therefore the derivative $d\boldsymbol{f}/d\boldsymbol{x} = \boldsymbol{A}$.

Since function $\boldsymbol{f}$ has dimensions M and the target $\boldsymbol{A}$ has dimensions $M \times N$. The derivative $d\boldsymbol{f}/d\boldsymbol{A}$ has the shape $M \times M \times N$.

$$\frac{\partial f_i}{\partial A_{ij}} = x_j \Rightarrow \frac{\partial f_i}{\partial A_k} = \begin{cases} \boldsymbol{x}^T, \ k = i \\ \boldsymbol{0}^T, \ k \neq i \end{cases}$$

Therefore the derivative $d\boldsymbol{f}/d\boldsymbol{A}$ is

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \boldsymbol{A}} \end{bmatrix}$$

Each of the partial derivatives $\frac{\partial f_i}{\partial \boldsymbol{A}}$ is a $M \times N$ matrix where the k^{th} is $\boldsymbol{x}^T$ if $k = i$, and $\boldsymbol{0}^T$ otherwise.

4.4 Higher Order Derivatives

Some methods of optimisation requires second derivatives. Given a function $f : \mathbb{R}^D \to \mathbb{R}$, the second derivative is represented by a $D \times D$ matrix called the **Hessian**. The Hessian matrix is a collection of second partial derivatives.

The second partial derivative means f is first differentiate with respect to x then with respect to y. Notice the order of differentiation does not matter.

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial f}{\partial x}\frac{\partial f}{\partial y} = \frac{\partial^2 f}{\partial y \partial x}$$

The point is a minimum point if the Hessian matrix is **positive definite**.

Given a function $f(x, y)$ that is twice differentiable. The Hessian matrix is given by

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

5 Probability and Distributions

In machine learning and statistics, there are two major interpretations of probabilities: the Bayesian and Frequentist interpretations. The Bayesian interpretation uses probability to specify the degree of uncertainty a user has about an event. The Frequentist interpretation considers the relative frequencies of events of interests to the total number of events.

5.1 Random Variables

The **sample space** Ω is the set of all possible outcomes of the experiment. The **event space** $\mathcal{A}$ is obtained by considering the collection of subsets of Ω. Each event $A \in \mathcal{A}$ is associated with a **probability** $P(A)$ that measures the probability the event will occur.

In machine learning, we often avoid explicitly referring to the probability space, but instead refer to probabilities on quantity of interest. This quality of interest is in the **target space** $\mathcal{T}$. A **random variable** $X : \Omega \to \mathcal{T}$ is a function that maps elements of Ω to a quantity of interest $x \in \mathcal{T}$.

It is possible to equate the probability of output of X and the probability of the samples in Ω. Given a set $\mathcal{S} \subseteq \mathcal{T}$, the probability of $\mathcal{S}$

$$P_X(\mathcal{S}) = P(X^{-1}(\mathcal{S})) = P(\{\omega \in \Omega : \ X(\omega) \in \mathcal{S}\})$$

Example

In the case of tossing two coins, the sample space $\Omega = \{HH, \ HT, \ TH, \ TT\}$. The quality of interest is the number of heads in each outcome. The random variable X maps each element in the sample space to the target space $\mathcal{T} = \{0, \ 1, \ 2\}$.

$$X(HH) = 2 \qquad X(HT) = 1 \qquad X(TH) = 1 \qquad X(TT) = 0$$

Say we want to find the probability of obtaining one head, $\mathcal{S} = \{1\}$. Then

$$P_X(\mathcal{S}) = P(X^{-1}(\mathcal{S})) = P(\{HT, \ TH\})$$

5.2 Discrete and Continuous Probabilities

Depending on whether the target space $\mathcal{T}$ is discrete or continuous, the natural way to refer to distributions is very different.

Continuous probabilities uses a **Probability Density Function (pdf)** $f : \mathbb{R}^D \to \mathbb{R}$ that must satisfy two conditions:

$$1.\ \forall \boldsymbol{x} \in \mathbb{R}^D : f(\boldsymbol{x}) \geq 0$$
$$2.\ \int f(\boldsymbol{x}) d\boldsymbol{x} = 1$$

Discrete probabilities uses **Probability Mass Function (pmf)** that is the same as pdf but the integral is replace with sum in the second condition.

Continuous probabilities also uses a **Cumulative Distribution Function (cdf)** defined as

$$F_X(\boldsymbol{x}) = P(X_1 \leq x_1, \ldots, X_D \leq x_D)$$

5.3 Sum Rule, Product Rule, and Bayes Theorem

The **joint probability** $p(\boldsymbol{x}, \boldsymbol{y})$ is the probability of intersection of both events.

The **sum rule** is used to find corresponding **marginal distributions** $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$.

$$p(\boldsymbol{x}) = \int_{\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \qquad p(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y})$$

The **product rule** relates the joint distribution to conditional distribution

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{y}) p(\boldsymbol{y})$$

Bayes' Theorem is used to draw conclusions about $\boldsymbol{x}$ given observed values in $\boldsymbol{y}$.

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x})}{p(\boldsymbol{y})}$$

The prior $p(\boldsymbol{x})$ encapsulates subjective prior knowledge $\boldsymbol{x}$. The likelihood $p(\boldsymbol{y} \mid \boldsymbol{x})$ describes how $\boldsymbol{x}$ and $\boldsymbol{y}$ are related. The evidence $p(\boldsymbol{y})$ can be seen as a normalising constant.

5.4 Summary Statistics

Mean and (co)variance are often useful to describe properties of probability distributions.

The **Expected Value** of a function $g : \mathbb{R} \to \mathbb{R}$ of a univariate random variable X

$$\mathbb{E}_X[g(x)] = \int_{\mathcal{X}} g(x)p(x)dx \qquad \mathbb{E}_X[g(x)] = \sum_{x\in\mathcal{X}} g(x)p(x)$$

For multivariate random variables, the expected value is defined element wise

$$\mathbb{E}_X[g(\boldsymbol{x})] = \begin{bmatrix} \mathbb{E}_{X_1}[g(x_1)] \\ \dots \\ \mathbb{E}_{X_D}[g(x_D)] \end{bmatrix}$$

The **Mean** is a special case of expected value where the function g is an identity function.

The **Covariance** between two univariate random variable X, Y is the expected product of their deviations from their respective mean

$$\begin{aligned} Cov_{X,Y}[x,y] &= \mathbb{E}_{X,Y}[(x - \mathbb{E}_X[x])(y - \mathbb{E}_Y[y])] \\ &= \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

For two multivariate random variables X, Y, the covariance is defined as

$$Cov[\boldsymbol{x},\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{y}]^T = Cov[\boldsymbol{y},\boldsymbol{x}]^T$$

The **Variance** of a random variable X is the covariance of the variable with itself

$$\begin{aligned} \mathbb{V}_X[\boldsymbol{x}] &= Cov_X[\boldsymbol{x},\boldsymbol{x}] \\ &= \mathbb{E}_X[\boldsymbol{x}\boldsymbol{x}^T] - \mathbb{E}_X[\boldsymbol{x}]\mathbb{E}_X[\boldsymbol{x}]^T \end{aligned}$$

The covariance matrix in this case is symmetric and positive semidefinite.

5.4.1 Empirical Mean and Covariance

The definitions above are **population mean** and **population covariance**. In machine learning, we usually have to learn form empirical observations of data. Given $\boldsymbol{x} \in \mathbb{R}^D$:

The **empirical mean** vector is the arithmetic average of the observations for each variable.

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

The **empirical covariance** is a $D \times D$ matrix

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T$$

If we focus on a single random variable X, the **variance** can be defined as

$$\mathbb{V}_X[x] = \mathbb{E}_X[(x-\mu)^2], \text{ where } \mu = \mathbb{E}_X[x]$$

5.5 Statistical Independence

Two random variables X and Y are **statistically independent** if and only if

$$p(\boldsymbol{x},\boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y})$$

X and Y are **conditionally independent** given Z if and only if

$$\forall \boldsymbol{z} \in Z : p(\boldsymbol{x},\boldsymbol{y} \mid \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{y} \mid \boldsymbol{z})$$

From the product rule

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{y})} \Rightarrow p(\boldsymbol{x} \mid \boldsymbol{y},\boldsymbol{z}) = \frac{p(\boldsymbol{x},\boldsymbol{y} \mid \boldsymbol{z})}{p(\boldsymbol{y} \mid \boldsymbol{z})}$$

If X and Y are conditionally independent given Z, then

$$\begin{aligned} p(\boldsymbol{x} \mid \boldsymbol{y},\boldsymbol{z}) &= \frac{p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{y} \mid \boldsymbol{z})}{p(\boldsymbol{y} \mid \boldsymbol{z})} \\ &= p(\boldsymbol{x} \mid \boldsymbol{z}) \end{aligned}$$

5.6 Probability Distributions

5.6.1 Bernoulli Distribution

The Bernoulli distribution is a distribution for a single binary random variable X. It has a single parameter $\mu \in [0,1]$ that represents the probability $P(X = 1)$. The Bernoulli distribution $Ber(\mu)$ is defined as

$$p(x \mid \mu) = \mu^x(1-\mu)^{1-x},\ x \in \{0,\ 1\}$$

The mean of Bernoulli $\mathbb{E}[x] = \mu$ and the variance $\mathbb{V}[x] = \mu(1-\mu)$

5.6.2 Binomial Distribution

The Binomial distribution can be used to describe the probability of observing m occurrences of $X = 1$ in a set of N samples from a Bernoulli distribution $Ber(\mu)$. The Binomial distribution $Bin(N,\ \mu)$ is defined as

$$p(m \mid N,\ \mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m}$$

The mean of Binomial $\mathbb{E}[m] = N\mu$ and the variance $\mathbb{V}[m] = N\mu(1-\mu)$

5.6.3 Beta Distribution

The Beta distribution represents a probability distribution of probabilities, it is often used to represent the probability of μ in Bernoulli distribution. The Beta distribution $Beta(\alpha, \beta)$ is defined as

$$p(\mu \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\mu^{\alpha-1}(1-\mu)^{\beta-1}$$

The two parameters $\alpha > 0$, $\beta > 0$ governs the distribution. Intuitively, α moves the probability mass towards 1 and β moves the probability mass towards 0.

An interesting case is when $\alpha = \beta = 1$, in this case we obtain Uniform distribution.

5.7 Gaussian Distribution

The Gaussian Distribution, also known as normal distribution, is extensively used in machine learning.

Given a multivariate data $\boldsymbol{x} \in \mathbb{R}^D$. The **multivariate Gaussian distribution** is defined by two parameters: the mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ.

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) = (2\pi)^{-\frac{D}{2}}|\Sigma|^{-\frac{1}{2}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

The Gaussian with $\boldsymbol{\mu} = \boldsymbol{0}$ and $\Sigma = \boldsymbol{I}$ is called the **standard normal distribution**.

5.7.1 Marginal and Conditional of Gaussians

Let X and Y be multivariate random variables. Their joint probability is a Gaussian distribution defined as

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right)$$

Where $\Sigma_{xx} = Cov[\boldsymbol{x}, \boldsymbol{x}]$ and $\Sigma_{yy} = Cov[\boldsymbol{y}, \boldsymbol{y}]$. Σ_{xy} and Σ_{yx} are transpose of each other.

The **marginal distribution** $p(\boldsymbol{x})$ is a Gaussian distribution

$$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_x, \Sigma_{xx}\right)$$

The **conditional probability** $p(\boldsymbol{x} \mid \boldsymbol{y})$ is a Gaussian distribution

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{\mu}_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_y),\ \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)$$

Notice that $\boldsymbol{y}$ here is not a random variable but a value.

5.7.2 Product of Gaussian

The product of two Gaussians $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{b}, \boldsymbol{B})$ is a Gaussian $z\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{c}, \boldsymbol{C})$. Where

$$\begin{aligned}\boldsymbol{C} &= (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1} \\ \boldsymbol{c} &= \boldsymbol{C}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}) \\ z &= (2\pi)^{-\frac{D}{2}}|\boldsymbol{A} + \boldsymbol{B}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{A}+\boldsymbol{B})^{-1}(\boldsymbol{a}-\boldsymbol{b})} \\ &= \mathcal{N}(\boldsymbol{a} \mid \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B}) = \mathcal{N}(\boldsymbol{b} \mid \boldsymbol{a}, \boldsymbol{A} + \boldsymbol{B})\end{aligned}$$

The scaling constant z can be written in the functional form of a Gaussian density. Meaning in $\mathcal{N}(\boldsymbol{a} \mid \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B})$ and $\mathcal{N}(\boldsymbol{a} \mid \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B})$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are not random variables but actual values.

5.7.3 Linear Transformation

Consider a Gaussian distributed random sample $X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Let Y be a random variable that is a transformed version of $\boldsymbol{x}$ by applying matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$.

$$\begin{aligned}\mathbb{E}[\boldsymbol{y}] &= \mathbb{E}[\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}] = \boldsymbol{A}\mathbb{E}[\boldsymbol{x}] + \boldsymbol{b} = \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b} \\ \mathbb{V}[\boldsymbol{y}] &= \mathbb{V}[\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}] = \boldsymbol{A}\mathbb{V}[\boldsymbol{x}]\boldsymbol{A}^\top = \boldsymbol{A}\Sigma\boldsymbol{A}^\top\end{aligned}$$

Therefore Y is a Gaussian distribution defined as

$$p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b},\ \boldsymbol{A}\Sigma\boldsymbol{A}^\top\right)$$

Consider the reverse transformation. Given $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ and we know the probability distribution of $p(\boldsymbol{y})$. We want to find the probability distribution of $p(\boldsymbol{x})$.

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{A}\boldsymbol{x},\ \Sigma)$$

If $\boldsymbol{A}$ is **invertible**. Then we can write $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$ and apply the transformation as shown above.

$$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x},\ \boldsymbol{A}^{-1}\Sigma(\boldsymbol{A}^{-1})^T\right)$$

If $\boldsymbol{A}$ is **not invertible**. We multiply both sides with $\boldsymbol{A}^T$ then invert $\boldsymbol{A}^T\boldsymbol{A}$. Then apply the transformation a shown above.

$$\begin{aligned}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y} &= \boldsymbol{x} \\ p(\boldsymbol{x}) &= \mathcal{N}\left(\boldsymbol{x} \mid (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x},\ (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\Sigma\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\right)\end{aligned}$$

5.8 Conjugacy

From Bayes' Theorem, the posterior is proportional to the product of the prior and likelihood. A prior is **conjugate** for the likelihood function if the posterior is the same as the prior.

Conjugacy is particularly convenient because we can algebraically calculate our posterior distribution by updating the parameters of the prior distribution.

6 Continuous Optimisation

Training a machine learning model means finding a good set of parameters that minimise a predefined objective function. In this chapter we will look at optimisation using gradient descent.

6.1 Gradient Descent

Gradient descent is an optimisation algorithm. It finds the minimum of a function by iteratively taking a step proportional to the negative of the gradient of the function at the current point.

Given a multivariate function $f(\boldsymbol{x})$ starting at location $\boldsymbol{x}_0$. In each step we update the location by

$$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i - \alpha_i \left((\nabla f)(\boldsymbol{x}_i)\right)^\top, \ \alpha_i \in (0, 1]$$

where α_i is the learning rate, or step-size.

6.1.1 Learning Rate

In standard gradient descent, the step-size α is constant throughout training. Choosing an appropriate step-size is important,

- A step-size that is too small takes a long time to converge
- A step-size that is too large may lead to overshooting or even diverge.

Ideally we want to have large step-size initially to make big steps towards the goal, but a small step-size for fine tuning when it is near the optimum. We can use **adaptive gradient methods** that rescales the step-size at each iteration.

6.1.2 Momentum

Gradient descent struggles in pathological curvatures (often described as valleys or trenches). Since the surface at the ridge curves is much steeper, the parameters will zigzag back and forth across the valley, making convergence very slow.

Momentum takes advantage of the knowledge from previous gradient to determine the direction to go.

$$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i - \alpha_i \left((\nabla f)(\boldsymbol{x}_i)\right)^\top - \beta\alpha_{i-1} \left((\nabla f)(\boldsymbol{x}_{i-1})\right)^\top, \ \beta \in (0, 1]$$

6.1.3 Stochastic Gradient Descent (SGD)

Computing the gradient can be time consuming, stochastic gradient descent approximates the gradient of an objective function that is written as a sum of differentiable functions.

Given a dataset with N data, the objective function $L(\boldsymbol{\theta})$ usually sum of losses $L_n(\boldsymbol{\theta})$ of each individual data.

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} L_n(\boldsymbol{\theta})$$

In standard gradient descent, we calculate the gradient using the entire dataset in each iteration. However when the training set is enormous, evaluating the sum of gradients become expensive.

$$\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i - \alpha_i \sum_{n=1}^{N} (\nabla L_n(\boldsymbol{\theta}_i))^\top$$

Instead of using all L_n from $n = 1 \dots N$, we can randomly select a subset for batch gradient descent. In the extreme case, we can select a single L_n to estimate the gradient.

Taking only a subset of a data to approximate the gradient still works because for gradient descent to converge, we only require the gradient to be an unbiased estimate of the true gradient.

7 When Models Meet Data

7.1 Directed Graph Models

Directed Graph Models are used for representing conditional dependencies in a probabilistic model. In a graphical model, **nodes** are random variables. **Directed link** from a to b indicates conditional probability $p(b \mid a)$.

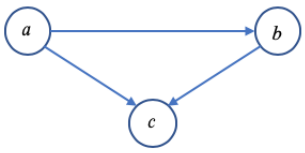
7.1.1 Constructing a Graph

Given a joint distribution, we can construct a graph in two steps:

1. Create a node for all random variables
2. For each conditional probability, add a directed link between the nodes

Example

The joint distribution $p(a, b, c) = p(c \mid a, b)p(b \mid a)p(a)$ is represented by the graph



7.1.2 Extracting Joint Probability

Given a graph, we can extract the joint probability in two steps

1. The joint distribution $p(x_1, x_2, ..., x_K)$ should include all random variable appeared in the graph
2. Each conditional depends only on the parents of a node.

In general, the joint distribution is given by

$$p(x_1, x_2, .., x_K) = \prod_{k=1}^{K} p(x_k \mid Parents(x_k))$$

7.1.3 Conditional Independence

Recall the definition of conditional independence.

$$\begin{aligned} \boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \mid \boldsymbol{z} &\doteq p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{y} \mid \boldsymbol{z}) \\ &\doteq p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}) \end{aligned}$$

Directed graph models allow us to find conditional independence using a concept called **d-separation**. Consider a directed graph with $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ which are nonintersecting sets of nodes.

We consider all paths (ignoring arrow direction) from any node in $\mathcal{A}$ to any node in $\mathcal{B}$. The path is blocked if

- Links meet **head to tail** or **tail to tail** at a node in $\mathcal{C}$, or
- Links meet **head to head** at a node. The node and all its descendants are not in $\mathcal{C}$.

If all paths are blocked, then we say $\mathcal{A}$ is d-separated from $\mathcal{B}$ by $\mathcal{C}$, and $\mathcal{A} \perp\!\!\!\perp \mathcal{B} \mid \mathcal{C}$.

7.2 Model Selection

In machine learning, the choice of model influence the number and type of free parameters in the model and thereby also the flexibility and expressivity of the model. In this section, we look at mechanisms for assessing how good a model generalises to unseen test data.

7.2.1 Cross Validation

K-fold cross validation partitions the entire data set into K chunks. In each iteration k, it uses $K-1$ chunks as the training set $\mathcal{R}^{(k)}$ to produce a predictor $f^{(k)}$, which is then applied to the validation set $\mathcal{V}^{(k)}$ to compute the loss $L(f^{(k)}, \mathcal{V}^{(k)})$. The process is

repeated K times using a different validation set in each iteration. The performance of the model is then averaged from all iterations:

$$\mathbb{E}_{\mathcal{V}}[L(f,\mathcal{V})] = \frac{1}{K}\sum_{k=1}^{K} L(f^{(k)},\mathcal{V}^{(k)})$$

We repeat this procedure for all models and choose the model that performs best. Once the model is chosen, we can evaluate the final performance on the test set.

7.2.2 Bayesian Model Selection

Simpler models are less expressive but are also less prone to overfitting. Bayesian model selection aims to find the simplest model that explains the data reasonably well.

Given a finite number of models $M = \{M_1, M_2, \ldots, M_K\}$ where each model M_k possesses parameters $\boldsymbol{\theta}_k$. Bayesian model selection places a prior on the set of model to create a generative process to generate data $\mathcal{D}$.

$$M_k \sim p(M) \qquad \boldsymbol{\theta}_k \sim p(\boldsymbol{\theta} \mid M_k) \qquad \mathcal{D} \sim p(\mathcal{D} \mid \boldsymbol{\theta}_k)$$

We can compute the posterior distribution of the model using Bayes' Theorem. Where $p(\mathcal{D} \mid M_k)$ is the marginal likelihood. The model no longer depends on the parameters because they are integrated out.

$$p(M_k \mid \mathcal{D}) \propto p(\mathcal{D} \mid M_k)p(M_k)$$
$$p(\mathcal{D} \mid M_k) = \int p(\mathcal{D} \mid \boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k \mid M_k)\, d\boldsymbol{\theta}_k$$

The best model M^* is the one that maximises the posterior distribution. If the prior $p(M)$ is uniformly distributed, then maximising the posterior is same as maximising the marginal likelihood.

$$M^* = \operatorname*{argmax}_{M_k} p(M_k \mid \mathcal{D})$$

8 Linear Regression

Regression task aims to find a function f that maps inputs $\boldsymbol{x} \in \mathbb{R}^D$ to corresponding noisy observed values $y \in \mathbb{R}$. The relationship between $\boldsymbol{x}$ and y is given as

$$y = f(\boldsymbol{x}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

In linear regression, we only consider the special case where the parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ appears linearly in the model. The aim is to seek for parameters $\boldsymbol{\theta}$.

$$y = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

This chapter will discuss maximum likelihood and maximum a posteriori (MAP) estimation to find optimal model parameters.

8.1 Maximum Likelihood Estimation

A widely used approach to finding the desired parameters $\boldsymbol{\theta}_{ML}$ is to find one that maximise the likelihood, which in this case is the joint probability.

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta})$$

Where $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$ and $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$.

Given a set of training data $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$ where all data points are independent and identically distributed, the joint probability is the product of all individual probability. Each individual probability is a Gaussian distribution because the noise ϵ is Gaussian distributed.

$$p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right)$$

In practice, instead of maximising the likelihood function, we maximise the log of likelihood.

$$\begin{aligned}
\boldsymbol{\theta}_{ML} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\log\left(p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta})\right)\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\log\left(\prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right)\right)\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2}}\right)\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\sum_{n=1}^{N} -\frac{(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2} + const\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-\sum_{n=1}^{N} (y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2\right)
\end{aligned}$$

We can write the above equation in vectors. Notice that maximising the likelihood is same as minimising the sum of square errors.

$$\begin{aligned}
\boldsymbol{\theta}_{ML} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})\right)
\end{aligned}$$

Where $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^\top$ is a $N \times D$ matrix and $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]$ is a column vector containing all the targets.

We can solve for $\boldsymbol{\theta}_{ML}$ by finding the derivative and setting it to 0.

$$\frac{d}{d\boldsymbol{\theta}} \left(-(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})\right) = \mathbf{0}^\top$$

$$
\begin{aligned}
2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}_{ML})^\top \boldsymbol{X} &= \boldsymbol{0}^\top \\
\boldsymbol{\theta}_{ML}^\top \boldsymbol{X}^\top \boldsymbol{X} &= \boldsymbol{y}^\top \boldsymbol{X} \\
\boldsymbol{\theta}_{ML} &= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}
\end{aligned}
$$

Notice that we require $\boldsymbol{X}^\top \boldsymbol{X}$ to be invertible, which is when all row in $\boldsymbol{X}$ are linearly independent.

8.1.1 Maximum Likelihood with Features

The previous section only allows us to fit straight lines to data using maximum likelihood estimation. To overcome this problem, we can apply a nonlinear transformation $\boldsymbol{\phi}(\boldsymbol{x})$ to the inputs $\boldsymbol{x} \in \mathbb{R}^D$. Therefore the relation between $\boldsymbol{x}$ and observed value y becomes

$$
y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)
$$

Where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^K$ is a nonlinear transformation to the input $\boldsymbol{x}$ make up of $\phi_k : \mathbb{R}^D \to \mathbb{R}$.

$$
\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{K-1}(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^K
$$

We can group all the transformed inputs into one single matrix $\boldsymbol{\Phi}$.

$$
\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}^\top(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\phi}^\top(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \dots & \phi_{K+1}(\boldsymbol{x}_1) \\ \vdots & & \vdots \\ \phi_0(\boldsymbol{x}_N) & \dots & \phi_{K+1}(\boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times K}
$$

Then find the parameters using maximum likelihood as above

$$
\boldsymbol{\theta}_{ML} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}
$$

Similarly, we require $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ to be invertible, meaning the rank of $\boldsymbol{\Phi} = K$.

8.1.2 Overfitting

A problem with maximum likelihood estimation is that it can lead to severe overfitting if complex models are trained using data sets of limited size. This is because it is possible to increase the likelihood beyond any bound.

We can evaluate the quality of a model by computing the error incurred. In linear regression we often use the **Root Mean Squared Error (RMSE)**.

$$
\sqrt{\frac{1}{N} \| \boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta} \|^2}
$$

For model selection, we can use RMSE to determine the best degree of the polynomial by finding the polynomial degree M that minimises the loss function. Given a

dataset of size N, we can fit a polynomial of degree $0 \leq M \leq N-1$.

We will notice that polynomials of low degree fit the data poorly. As the degree increase RMSE decreases. Until we reach even higher degrees the function passes through every data point and can not be used to generalise the data, this is when overfitting occurs.

8.2 Maximum a Posteriori Estimation

Maximum likelihood estimation is prone to overfitting. To mitigate the effect of large parameter values, we can place a prior distribution $p(\boldsymbol{\theta})$ on the parameters. Instead of maximising the likelihood, we find the parameters that maximise the posterior distribution.

$$p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{Y} \mid \mathcal{X})}$$

We can set the prior to a Gaussian distribution to explicitly encode what parameter values are plausible.

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{0}, b^2 \boldsymbol{I}\right)$$

To find the MAP estimate, we follow the steps similar to maximum likelihood estimation.

$$\begin{aligned}
\boldsymbol{\theta}_{MAP} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}(\log(p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y}))) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\log \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2}} + \log \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi b^2}} e^{-\frac{\boldsymbol{\theta}^\top \boldsymbol{\theta}}{2b^2}} \right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 - \frac{1}{2b^2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) - \frac{\sigma^2}{b^2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \right)
\end{aligned}$$

Maximising the posterior is same as minimising the sum of square errors with L2 regularisation. We use the parameter λ to control the degree of regularisation.

$$\begin{aligned}
\boldsymbol{\theta}_{MAP} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(-(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) - \frac{\sigma^2}{b^2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta} \right)
\end{aligned}$$

We can solve for $\boldsymbol{\theta}_{MAP}$ by finding the derivative and setting it to 0.

$$\frac{d}{d\boldsymbol{\theta}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \frac{\sigma^2}{b^2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \right) = \mathbf{0}^\top$$

$$\frac{\sigma^2}{b^2}\boldsymbol{\theta}_{MAP}^\top = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}_{MAP})^\top \boldsymbol{X}$$
$$\frac{\sigma^2}{b^2}\boldsymbol{\theta}_{MAP}^\top = \boldsymbol{y}^\top \boldsymbol{X} - \boldsymbol{\theta}_{MAP}^\top \boldsymbol{X}^\top \boldsymbol{X}$$
$$\boldsymbol{\theta}_{MAP} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

Comparing this with $\boldsymbol{\theta}_{ML}$, the only different is the additional term $\frac{\sigma^2}{b^2}\boldsymbol{I}$. If we set b^2 to be infinitely large, the term approaches 0 and we get the same result as maximum likelihood estimation. Intuitively, setting the variance b^2 of the prior to a large number removes the constraint on the parameters $\boldsymbol{\theta}$.

8.3 Bayesian Linear Regression

Unlike Maximum likelihood estimation and maximising a posteriori that aims to find a single best value of model parameter, Bayesian linear regression aims to determine the posterior distribution of the model parameters.

Not only is the response generated from a probability distribution, but the model parameters are assumed to come from a distribution as well. This means we do not fit any parameters, but compute the mean over all plausible parameters settings.

8.3.1 Prior Estimation

Given a prior distribution of the parameters $\boldsymbol{\theta}$, we can make a prediction of the noise corrupted target y_* at input $\boldsymbol{x}$ by averaging all plausible parameter settings.

$$p(y_* \mid \boldsymbol{x}) = \int p(y_* \mid \boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta}}[p(y_* \mid \boldsymbol{x})]$$

Since the likelihood is a Gaussian distribution, we choose a conjugate prior on $\boldsymbol{\theta}$ so that the above integration can be computed in close form.

$$p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\left(y \mid \phi^\top(\boldsymbol{x})\boldsymbol{\theta}, \sigma^2\right)$$
$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0\right)$$

We can apply the marginalisation property of Gaussian to find $p(y_* \mid \boldsymbol{x})$.

$$\mathbb{E}[y_*] = \mathbb{E}[\phi^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon] = \phi^\top(\boldsymbol{x})\boldsymbol{m}_0$$
$$\mathbb{V}[\boldsymbol{y}_*] = \mathbb{V}[\phi^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon] = \phi^\top(\boldsymbol{x})\boldsymbol{S}_0\phi(\boldsymbol{x}) + \sigma^2$$
$$p(y_* \mid \boldsymbol{x}) = \mathcal{N}\left(y_* \mid \phi^\top(\boldsymbol{x})\boldsymbol{m}_0,\ \phi^\top(\boldsymbol{x})\boldsymbol{S}_0\phi(\boldsymbol{x}) + \sigma^2\right)$$

The noise free target y is simply removing the noise variance σ^2 in the predictive variance.

$$p(y \mid \boldsymbol{x}) = \mathcal{N}\left(\phi^\top(\boldsymbol{x})\boldsymbol{m}_0,\ \phi^\top(\boldsymbol{x})\boldsymbol{S}_0\phi(\boldsymbol{x})\right)$$

8.3.2 Posterior Distribution

Given the distribution of the parameters $\boldsymbol{\theta}$, we can update the distribution using the training data to obtain a better set of plausible parameters. Given training set inputs $\mathcal{X}$ and the corresponding observations $\mathcal{Y}$. We can compute the posterior distribution using Bayes' Theorem, which is a better representation of the parameter distribution.

$$p(\boldsymbol{\theta} \mid \mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{Y} \mid \mathcal{X})}$$

The likelihood and prior are Gaussian distributions.

$$\begin{aligned} p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) &= \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{\theta},\ \sigma^2 \boldsymbol{I}\right) \\ p(\boldsymbol{\theta}) &= \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0\right) \end{aligned}$$

The posterior distribution is in the form $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{m}_N, \boldsymbol{S}_N)$. We can solve for $\boldsymbol{m}_N$ and $\boldsymbol{S}_N$ by finding the log posterior, log likelihood and log prior and comparing like terms.

The log posterior is

$$\begin{aligned} \log \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{m}_N, \boldsymbol{S}_N) &= -\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{m}_N)^\top \boldsymbol{S}_N^{-1}(\boldsymbol{\theta} - \boldsymbol{m}_N) \\ &= -\frac{1}{2}\left(\boldsymbol{\theta}^\top \boldsymbol{S}_N^{-1}\boldsymbol{\theta} - 2\boldsymbol{m}_N^\top \boldsymbol{S}_N^{-1}\boldsymbol{\theta}\right) + const \end{aligned}$$

The sum of log likelihood is log prior is

$$\begin{aligned} \log\left(\frac{p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta} p(\boldsymbol{\theta})}{p(\mathcal{Y} \mid \mathcal{X})}\right) &= \log \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{\theta},\ \sigma^2 \boldsymbol{I}\right) + \log \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0) + const \\ &= -\frac{1}{2}\left(\sigma^{-2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + (\boldsymbol{\theta} - \boldsymbol{m}_0)^\top \boldsymbol{S}_0^{-1}(\boldsymbol{\theta} - \boldsymbol{m}_0)\right) + const \\ &= -\frac{1}{2}\left(\boldsymbol{\theta}^\top (\sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{S}_0^{-1})\boldsymbol{\theta} - 2(\sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)^\top \boldsymbol{\theta}\right) + const \end{aligned}$$

By comparing like terms

$$\begin{aligned} \boldsymbol{S}_N &= \left(\sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{S}_0^{-1}\right)^{-1} \\ \boldsymbol{m}_N &= \boldsymbol{S}_N\left(\sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0\right) \end{aligned}$$

After we find the posterior distribution of the parameters, we can use this distribution as the prior in prior estimation to predict samples.

8.3.3 Computing Marginal Likelihood

In the following section, we compute the marginal likelihood for Bayesian linear regression with a conjugate Gaussian prior on the parameters.

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0) \qquad \text{prior}$$

$$p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{\theta}, \sigma^2 \boldsymbol{I}\right) \qquad \text{likelihood}$$

The marginal likelihood is given by

$$\begin{aligned} p(\mathcal{Y} \mid \mathcal{X}) &= \int p(\mathcal{Y} \mid \mathcal{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \\ &= \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{\theta}, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0\right) \, d\boldsymbol{\theta} \end{aligned}$$

Recall from Section 2.7

1. The product of two Gaussians $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{a}, \boldsymbol{A}) \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{b}, \boldsymbol{B})$ is a Gaussian in the form $z\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{c}, \boldsymbol{C})$, where z is the functional form of a Gaussian density $\mathcal{N}(\boldsymbol{a} \mid \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B})$
2. We can apply a linear transformation to bring $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0)$ into the form $\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}, \Sigma)$ because $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}$.

$$\begin{aligned} p(\mathcal{Y} \mid \mathcal{X}) &= \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{\theta}, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{m}_0, \boldsymbol{S}_0\right) \, d\boldsymbol{\theta} \\ &= \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{\theta}, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{m}_0, \boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top\right) \, d\boldsymbol{\theta} \\ &= z \int \mathcal{N}\left(\boldsymbol{c}, \boldsymbol{C}\right) \, d\boldsymbol{\theta} \\ &= \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{m}_0, \, \boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2 \boldsymbol{I}\right) \end{aligned}$$

9 Dimensionality Reduction with PCA

Storing high-dimensional data vectors is expensive, dimensionality reduction exploits structures and correlations and allow us to work with more compact representation of data. This chapter looks at an algorithm for linear dimensionality reduction called **Principle Component Analysis (PCA)**.

Consider a dataset $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ where $\boldsymbol{x}_n \in \mathbb{R}^D$, with mean $\boldsymbol{0}$ and covariance

$$\boldsymbol{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

Furthermore, we assume there is a low-dimensional compressed representation of $\boldsymbol{x}_n$ using the projection matrix $\boldsymbol{B} \in \mathbb{R}^M$, where $M < D$.

$$\begin{aligned} \boldsymbol{B} &= \begin{bmatrix} \boldsymbol{b}_1 & \ldots & \boldsymbol{b}_M \end{bmatrix} \in \mathbb{R}^{D \times M} \\ \boldsymbol{z}_n &= \boldsymbol{B}^\top \boldsymbol{x}_n \in \mathbb{R}^M \end{aligned}$$

The aim is to find matrix $\boldsymbol{B}$ that retains as much information as possible when compressing the data. Retaining most information after data compression is equivalent to capturing the largest amount of variance in the low-dimensional code.

9.1 Direction with Maximum Variance

We will start by finding a single vector $\boldsymbol{b}_1 \in \mathbb{R}^D$ that maximises the variance of projected data, i.e. the first coordinate z_1 of $\boldsymbol{z}$.

$$\begin{aligned}
\max \frac{1}{N}\sum_{n=1}^{N} z_{1n}^2 &= \underset{\boldsymbol{b}_1}{\operatorname{argmax}} \frac{1}{N}\sum_{n=1}^{N} \left(\boldsymbol{b}_1^\top \boldsymbol{x}_n\right)^2 \\
&= \underset{\boldsymbol{b}_1}{\operatorname{argmax}} \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{b}_1^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_1 \\
&= \underset{\boldsymbol{b}_1}{\operatorname{argmax}} \boldsymbol{b}_1^\top \left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top\right) \boldsymbol{b}_1 \\
&= \underset{\boldsymbol{b}_1}{\operatorname{argmax}} \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1
\end{aligned}$$

One problem is that $\boldsymbol{b}_1$ can become infinitely large to maximise the variance. Therefore we add a restriction $\| \boldsymbol{b}_1 \| = 1$ and solved for the constrained optimisation problem using **Lagrangian**.

$$\mathcal{L}(\boldsymbol{b}_1, \lambda_1) = \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 + \lambda_1 (1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1)$$

We can solve for the partial derivatives

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_1} = 2\boldsymbol{S}\boldsymbol{b}_1 - 2\lambda_1 \boldsymbol{b}_1 \qquad \frac{\partial \mathcal{L}}{\partial \lambda_1} = 1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1$$

Setting the derivatives to zero, we can obtain

$$\boldsymbol{S}\boldsymbol{b}_1 = \lambda_1 \boldsymbol{b}_1 \qquad \boldsymbol{b}_1^\top \boldsymbol{b}_1 = 1$$

Comparing this with the definition of Eigendecomposition, we see that $\boldsymbol{b}_1$ is an eigenvector of the covariance matrix $\boldsymbol{S}$, and λ_1 corresponds to the eigenvalue. Therefore, to maximise the variance of the low-dimensional code, we choose the basis vector associated with the largest eigenvalue of the data covariance matrix.

$$\max \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 = \max \lambda_1 \boldsymbol{b}_1^\top \boldsymbol{b}_1 = \max \lambda_1$$

9.2 M-dimensional Maximum Variance

Now we consider the case of M dimensions, that is

$$\boldsymbol{z}_n = \begin{bmatrix} z_{1n} \\ \vdots \\ z_{Mn} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1^\top \boldsymbol{x}_n \\ \vdots \\ \boldsymbol{b}_M^\top \boldsymbol{x}_n \end{bmatrix} = \boldsymbol{B}^\top \boldsymbol{x}_n$$

Similar to above, we want to maximise the variance

$$\max \sum_{m=1}^{M}\sum_{n=1}^{N} z_{mn}^2 = \underset{\boldsymbol{B}}{\operatorname{argmax}} \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} \left(\boldsymbol{b}_m^\top \boldsymbol{x}_n\right)^2$$

$$
\begin{aligned}
&= \underset{\boldsymbol{B}}{\operatorname{argmax}} \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} \boldsymbol{b}_m^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_m \\
&= \underset{\boldsymbol{B}}{\operatorname{argmax}} \sum_{m=1}^{M} \boldsymbol{b}_m^\top \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top \right) \boldsymbol{b}_m \\
&= \underset{\boldsymbol{B}}{\operatorname{argmax}} \sum_{m=1}^{M} \boldsymbol{b}_m^\top \boldsymbol{S} \boldsymbol{b}_m \\
&= \underset{\boldsymbol{B}}{\operatorname{argmax}}\, tr\left(\boldsymbol{B}^\top \boldsymbol{S} \boldsymbol{B} \right)
\end{aligned}
$$

Similar to above, we add a constraint such that $\boldsymbol{B}$ is an orthogonal matrix. Hence, the Lagrangian we obtain becomes

$$\mathcal{L}(\boldsymbol{B}, \boldsymbol{\Lambda}) = tr\left(\boldsymbol{B}^\top \boldsymbol{S} \boldsymbol{B} \right) + tr\left(\boldsymbol{\Lambda} (\boldsymbol{I} - \boldsymbol{B}^\top \boldsymbol{B}) \right)$$

Finding the derivative and setting them to $\mathbf{0}$, we obtain

$$\boldsymbol{S}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{\Lambda} \Rightarrow \boldsymbol{S}\boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m$$

This means the columns of $\boldsymbol{B}$ corresponds to the eigenvectors of $\boldsymbol{S}$. To maximise the variance, it means finding the eigenvectors with the largest eigenvalues.

9.3 Computing PCA

In this section, we will look at how to compute PCA for **small sample size problems**, where the the number of features D is much larger than the number of samples N.

Lemma Let $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^\top$ and $\boldsymbol{C} = \boldsymbol{X}^\top \boldsymbol{X}$. Then

1. $\boldsymbol{B}$ and $\boldsymbol{C}$ have the same eigenvalues $\boldsymbol{\Lambda}$
2. The eigenvectors of $\boldsymbol{B}$, $\boldsymbol{U}$, and the eigenvectors of $\boldsymbol{C}$, $\boldsymbol{V}$, are related by $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{\Lambda}^{-\frac{1}{2}}$

At the previous section, we saw that computing PCA requires computing the eigenvectors of $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top$, which has shape $D \times D$. When $D >> N$, this is very inefficient. Using the Lemma, we can instead compute eigenvectors of $\boldsymbol{S}$ from $\boldsymbol{X}^\top \boldsymbol{X}$, which is more efficient.

1. Centralised $\boldsymbol{X}$ to mean $\mathbf{0}$. $\boldsymbol{X} = \{\boldsymbol{x}_1 - \boldsymbol{\mu}, \ldots, \boldsymbol{x}_N - \boldsymbol{\mu}\}$
2. Compute dot product matrix $\boldsymbol{X}^\top \boldsymbol{X}$ and perform eigenanalysis $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$
3. Compute eigenvectors of $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top$ (We dropped $\frac{1}{N}$ for convenience) using Lemma, $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{\Lambda}^{-\frac{1}{2}}$
4. Compute features $\boldsymbol{Y} = \boldsymbol{U}^\top \boldsymbol{X}$

9.4 Kernel PCA (KPCA)

PCA is a linear method. That is it can only be applied to datasets which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable.

Given a dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$. We can assume there is a non-linear mapping function that maps $\boldsymbol{x}_i \in \mathbb{R}^D$ to another feature space $\boldsymbol{X}_\Phi = [\phi(\boldsymbol{x}_i), \ldots, \phi(\boldsymbol{x}_N)]$

$$\phi : \mathbb{R}^D \to \mathbb{R}^E$$

Usually this mapping function ϕ is unknown or is extremely expensive to compute. Instead, we only know the kernel functions which computes the dot product in feature space.

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$$

We define the centralised feature space as

$$\overline{\boldsymbol{X}}_\Phi = [\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\Phi, \ldots, \boldsymbol{x}_N - \boldsymbol{\mu}_\Phi] = \boldsymbol{X}_\phi(\boldsymbol{I} - \boldsymbol{E}) \qquad \boldsymbol{\mu}_\Phi = \frac{1}{N}\sum_{i=1}^{N}\phi(\boldsymbol{x}_i) \qquad \boldsymbol{E} = \frac{1}{N}\boldsymbol{1}\boldsymbol{1}^\top$$

We define the kernel as

$$\boldsymbol{K} = [\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)] = \boldsymbol{X}_\Phi^\top \boldsymbol{X}_\Phi$$

We also define a centralised kernel

$$\begin{aligned}\overline{\boldsymbol{K}} &= \left[\left(\phi(x_i) - \boldsymbol{\mu}_\Phi\right)\left(\phi(x_i) - \boldsymbol{\mu}_\Phi\right)^\top\right] \\ &= (\boldsymbol{I} - \boldsymbol{E})\boldsymbol{X}_\Phi^\top \boldsymbol{X}_\Phi(\boldsymbol{I} - \boldsymbol{E}) \\ &= \overline{\boldsymbol{X}}_\Phi^\top \overline{\boldsymbol{X}}_\Phi\end{aligned}$$

We can now perform PCA on the feature space, solving the optimisation problem

$$\underset{\boldsymbol{B}}{\operatorname{argmax}}\, tr\left(\boldsymbol{B}^\top \boldsymbol{S}_\Phi \boldsymbol{B}\right) \qquad\qquad Subject\ to\ \boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}$$

We cannot find $\boldsymbol{S}_\Phi$ since we do not know the mapping function ϕ. However, we can use the Lemma defined in the previous section to compute $\boldsymbol{B}$

$$\begin{aligned}\overline{\boldsymbol{K}} &= \overline{\boldsymbol{X}}_\Phi^\top \overline{\boldsymbol{X}}_\Phi = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top \\ \boldsymbol{S}_\Phi &= \overline{\boldsymbol{X}}_\Phi \overline{\boldsymbol{X}}_\Phi^\top = \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^\top \\ \boldsymbol{B} &= \overline{\boldsymbol{X}}_\Phi \boldsymbol{V}\boldsymbol{\Lambda}^{-\frac{1}{2}}\end{aligned}$$

We can use this $\boldsymbol{B}$ to calculate the latent features of data in the feature space

$$\boldsymbol{z}_i = \boldsymbol{B}^\top\left(\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\Phi\right)$$

$$
\begin{aligned}
&= \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{V}^\top \overline{\boldsymbol{X}}_\Phi^\top \left(\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\Phi \right) \\
&= \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{V}^\top (\boldsymbol{I} - \boldsymbol{E}) \boldsymbol{X}_\Phi^\top \left(\phi(\boldsymbol{x}_i) - \frac{1}{N} \boldsymbol{X}_\Phi \mathbf{1} \right) \\
&= \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{V}^\top (\boldsymbol{I} - \boldsymbol{E}) \left(\boldsymbol{X}_\Phi^\top \phi(\boldsymbol{x}_i) - \frac{1}{N} \boldsymbol{K} \mathbf{1} \right) \\
&= \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{V}^\top (\boldsymbol{I} - \boldsymbol{E}) \left(\begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_i) \\ \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_i) \end{bmatrix} - \frac{1}{N} \boldsymbol{K} \mathbf{1} \right)
\end{aligned}
$$

10 Dimensionality Reduction with LDA

Both LDA and PCA are linear transformation techniques, while PCA is used for unsupervised approach, **Linear Discriminant Analysis (LDA)** is used in supervised classification problems.

Consider a dataset $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ each with a corresponding label l_i, where $l_i \in 1, \ldots, C$ and C is the number of classes. The aim of LDA is the project $\boldsymbol{x}_i$ to a low-dimensional space $\boldsymbol{y}_i$ such that

1. The distance between mean of classes is maximised
2. The variance within each class is minimised

10.1 LDA with Two Classes

This section looks at LDA when there are only two classes, i.e. $C = 2$. We want to project $\boldsymbol{x}_i$ to a latent space of $\boldsymbol{y}_i$ such that the two classes have means $\mu_y(c_1)$, $\mu_y(c_2)$ and variances $\sigma_y^2(c_1)$, $\sigma_y^2(c_2)$. The goal is to

1. Maximise the distance between mean of classes $(\mu_y(c_1) - \mu_y(c_2))^2$
2. Minimise the variance within each class $\sigma_y^2(c_1) + \sigma_y^2(c_2)$

$$
\begin{aligned}
\sigma_y^2(c_1) + \sigma_y^2(c_2) &= \frac{1}{N_{c_1}} \sum_{x_i \in c_1} \left(\boldsymbol{y}_i - \mu_y(c_1) \right)^2 + \frac{1}{N_{c_2}} \sum_{x_i \in c_2} \left(\boldsymbol{y}_i - \mu_y(c_2) \right)^2 \\
&= \frac{1}{N_{c_1}} \sum_{x_i \in c_1} \left(\boldsymbol{w}^\top (\boldsymbol{x}_i - \mu_x(c_1)) \right)^2 + \frac{1}{N_{c_2}} \sum_{x_i \in c_2} \left(\boldsymbol{w}^\top (\boldsymbol{y}_i - \mu_x(c_2)) \right)^2 \\
&= \boldsymbol{w}^\top \frac{1}{N_{c_1}} \sum_{x_i \in c_1} \left((\boldsymbol{x}_i - \mu_x(c_1))(\boldsymbol{x}_i - \mu_x(c_1))^\top \right) \boldsymbol{w} + \\
&\quad \boldsymbol{w}^\top \frac{1}{N_{c_2}} \sum_{x_i \in c_2} \left((\boldsymbol{x}_i - \mu_x(c_2))(\boldsymbol{x}_i - \mu_x(c_2))^\top \right) \boldsymbol{w} \\
&= \boldsymbol{w}^\top \boldsymbol{S}_1 \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{S}_2 \boldsymbol{w}
\end{aligned}
$$

$$= \boldsymbol{w}^\top (\boldsymbol{S}_1 + \boldsymbol{S}_2)\boldsymbol{w}$$
$$= \boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w}$$

We denote the within class variance $\boldsymbol{S}_1 + \boldsymbol{S}_2$ as $\boldsymbol{S}_w$

$$(\mu_y(c_1) - \mu_y(c_2))^2 = \boldsymbol{w}^\top (\mu_x(c_1) - \mu_x(c_2))(\mu_x(c_1) - \mu_x(c_2))^\top \boldsymbol{w} \qquad = \boldsymbol{w}^\top \boldsymbol{S}_b \boldsymbol{w}$$

We denote the between class variance $(\mu_x(c_1) - \mu_x(c_2))(\mu_x(c_1) - \mu_x(c_2))^\top$ as $\boldsymbol{S}_b$.

The quantity we want to maximise becomes

$$\frac{\boldsymbol{w}^\top \boldsymbol{S}_b \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w}}$$

This is equivalent to solving for the constrained optimisation problem using Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \lambda) = \boldsymbol{w}^\top \boldsymbol{S}_b \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^\top \boldsymbol{S}_w \boldsymbol{w})$$

Solving for the partial derivatives

$$\frac{\partial}{\partial \boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \lambda) = 2\boldsymbol{S}_b \boldsymbol{w} - 2\lambda \boldsymbol{S}_w \boldsymbol{w}$$

Setting the derivatives to 0, we can obtain

$$\boldsymbol{S}_w^{-1} \boldsymbol{S}_b \boldsymbol{w} = \lambda \boldsymbol{w}$$

Therefore, $\boldsymbol{w}$ is the eigenvector that corresponds to the largest eigenvalue of $\boldsymbol{S}_w^{-1} \boldsymbol{S}_b$

10.2 LDA with Multi-class

We can now extend LDA to classify multiple classes, $C > 2$. The within class variance matrix is defined as

$$\boldsymbol{S}_w = \sum_{j=1}^{C} \frac{1}{N_{c_j}} \sum_{x_i \in c_j} \Big(\boldsymbol{x}_i - \mu_x(c_j)\Big)\Big(\boldsymbol{x}_i - \mu_x(c_j)\Big)^\top = \sum_{j=1}^{C} \boldsymbol{S}_j$$

The between class matrix is defined as

$$\boldsymbol{S}_b = \sum_{j=1}^{C} N_{c_j} \Big(\mu_x(c_j) - \boldsymbol{\mu}\Big)\Big(\mu_x(c_j) - \boldsymbol{\mu}\Big)^\top$$

Where $\boldsymbol{\mu}$ is the mean of all the classes center.

The goal is to find the projection matrix $\boldsymbol{W}$ by solving the Lagrangian

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{\Lambda}) = tr\Big(\boldsymbol{W}^\top \boldsymbol{S}_b \boldsymbol{W}\Big) + tr\Big(\boldsymbol{\Lambda}(\boldsymbol{I} - \boldsymbol{W}^\top \boldsymbol{S}_w \boldsymbol{W})\Big)$$

Finding the derivatives and setting it to zero, we obtain

$$\boldsymbol{S}_w^{-1} \boldsymbol{S}_b \boldsymbol{W} = \boldsymbol{W} \boldsymbol{\Lambda}$$

Therefore the columns of $\boldsymbol{W}$ are the eigenvectors that corresponds to the largest eigenvalues of $\boldsymbol{S}_w^{-1} \boldsymbol{S}_b$.

10.3 Computing LDA

In this section, we will look at how to compute LDA for **small sample size problems**, where the number of features D is much larger than the number of samples N, and without assuming S_w is invertible.

10.3.1 Scatter Matrix

The scatter matrix is defined as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{E}_1 & 0 & \dots & 0 \\ 0 & \boldsymbol{E}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{E}_C \end{bmatrix} \qquad \boldsymbol{E}_i = \begin{bmatrix} \frac{1}{N_{c_i}} & \cdots & \frac{1}{N_{c_i}} \\ \vdots & \ddots & \vdots \\ \frac{1}{N_{c_i}} & \cdots & \frac{1}{N_{c_i}} \end{bmatrix}_{N_{c_i} \times N_{c_i}} = \frac{1}{N_{c_i}} \mathbf{1}\mathbf{1}^\top$$

For example, given a dataset with 3 samples from class 1 and 2 samples from class 2, the scatter matrix is defined as

$$\boldsymbol{M} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

One interesting property of scatter matrix is that it is **idempotent**, meaning $\boldsymbol{M}\boldsymbol{M} = \boldsymbol{M}$.

We can express S_b and S_w in terms of the scatter matrix.

$$\begin{aligned}
S_b &= \sum_{j=1}^{C} N_{c_j} \mu_x(c_j) \mu_x(c_j)^\top && \text{Assume the center points are centered} \\
&= \boldsymbol{X}\boldsymbol{M}\boldsymbol{M}\boldsymbol{X}^\top \\
S_w &= \sum_{j=1}^{C} \sum_{x_i \in c_j} \Big(\boldsymbol{x}_i - \boldsymbol{\mu}_x(c_j)\Big)\Big(\boldsymbol{x}_i - \boldsymbol{\mu}_x(c_j)\Big)^\top && \text{Drop } \frac{1}{N_{c_j}} \text{ for convenience} \\
&= \sum_{j=1}^{C} \sum_{x_i \in c_j} \Big(\boldsymbol{x}_i \boldsymbol{x}_i^\top - \boldsymbol{\mu}_x(c_j)\boldsymbol{x}_i^\top - \boldsymbol{x}_i \boldsymbol{\mu}_x(c_j)^\top + \boldsymbol{\mu}_x(c_j)\boldsymbol{\mu}_x(c_j)^\top\Big) \\
&= \sum_{j=1}^{C} \left(\sum_{x_i \in c_j} \Big(\boldsymbol{x}_i \boldsymbol{x}_i^\top\Big) - N_{c_j} \boldsymbol{\mu}_x(c_j) \boldsymbol{\mu}(c_j)^\top \right) \\
&= \sum_{j=1}^{C} \sum_{x_i \in c_j} \boldsymbol{x}_i \boldsymbol{x}_i^\top - \sum_{j=1}^{C} N_{c_j} \boldsymbol{\mu}_x(c_j) \boldsymbol{\mu}_x(c_j)^\top \\
&= \boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{M}\boldsymbol{X}^\top \\
&= \boldsymbol{X}(\boldsymbol{I} - \boldsymbol{M})(\boldsymbol{I} - \boldsymbol{M})\boldsymbol{X}^\top && (\boldsymbol{I} - \boldsymbol{M}) \text{ is also idempotent}
\end{aligned}$$

10.3.2 Simultaneous Diagonalisation

Recall in the original LDA problem we want to optimise

$$\underset{W}{\operatorname{argmax}}\, tr\left(W^\top S_b W\right) \qquad Subject\ to\ W^\top S_w W = I$$

We can now rewrite the optimisation problem expressing S_b and S_w in terms of M.

$$\underset{W}{\operatorname{argmax}}\, tr\left(W^\top XMMX^\top W\right) \qquad Subject\ to\ W^\top X(I-M)(I-M)X^\top W = I$$

Assume that we can express $W = UQ$. The above optimisation problem becomes

$$\underset{Q}{\operatorname{argmax}}\, tr\left(Q^\top U^\top XMMX^\top UQ\right) \quad Subject\ to\ Q^\top U^\top X(I-M)(I-M)X^\top UQ = I$$

The next step is to find a U such that $U^\top X(I-M)(I-M)X^\top U = I$. The purpose of this step is so that we can rewrite the above optimisation into

$$\underset{Q}{\operatorname{argmax}}\, tr\left(Q^\top U^\top XMMX^\top UQ\right) \qquad Subject\ to\ Q^\top Q = I$$

This optimisation problem has the same form as Principle Component Analysis, where the columns of Q are the eigenvectors that corresponds to the largest eigenvalues of $U^\top XMMX^\top U$.

To solve for U, we perform eigenanalysis and whitening to $X(I-M)(I-M)X^\top$. We will use the Lemma in PCA section to compute eigenvectors more efficiently

1. Perform eigenanalysis $(I-M)X^\top X(I-M) = V\Lambda V^\top$
2. Use Lemma $U = X(I-M)V\Lambda^{-\frac{1}{2}}$
3. Perform whitening so all eigenvalues are normalised to 1, this is to create identity matrix. $U = U\Lambda^{-\frac{1}{2}} = X(I-M)V\Lambda^{-1}$

Obtaining U, we can solve for Q by performing eigenanalysis on $U^\top XMMX^\top U$.

We can now solve for $W = UQ$

11 Support Vector Machine

Support Vector Machines (SVM) is a supervised machine learning algorithm which is mostly used in classification problems. In this algorithm, each data is a vector $x_i \in \mathbb{R}^D$ that belongs to class $y_i \in \{-1, 1\}$. The goal of SVM is to find a separating hyperplane that separates the two classes.

More formally, SVM aims to find parameters w and b such that the distance between the lines $w^\top x + b = -1$ and $w^\top x + b = 1$ is maximised.

Let $\tilde{\boldsymbol{x}}$ be a point on $\boldsymbol{w}^\top \boldsymbol{x} + b = -1$. We know $\boldsymbol{w}$ is a vector perpendicular to the line. Starting from $\tilde{\boldsymbol{x}}$, we move along direction $\boldsymbol{w}$ until we reach the line $\boldsymbol{w}^\top \boldsymbol{x} + b = 1$. This gives us the equations

$$\begin{cases} \boldsymbol{w}^\top(\tilde{\boldsymbol{x}} + t\boldsymbol{w}) + b = 1 \\ \boldsymbol{w}^\top \tilde{\boldsymbol{x}} + b = -1 \end{cases} \Rightarrow \begin{cases} \boldsymbol{w}^\top \tilde{\boldsymbol{x}} + b + t\boldsymbol{w}^\top \boldsymbol{w} = 1 \\ \boldsymbol{w}^\top \tilde{\boldsymbol{x}} + b = -1 \end{cases} \Rightarrow t\boldsymbol{w}^\top \boldsymbol{w} = 2$$

This goal is to maximise the distance between the two lines

$$\begin{aligned} \max_{w,b} \|t\boldsymbol{w}\| &= \max_{w,b} \frac{2}{\boldsymbol{w}^\top \boldsymbol{w}} \|\boldsymbol{w}\| \\ &= \max_{w,b} \frac{2}{\|\boldsymbol{w}\|} \\ &= \min_{w,b} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} \end{aligned}$$

We want to introduce additional constraints such that data in class 1 must be one the right of $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$, whereas data in class 2 must be on the left.

$$\begin{aligned} \boldsymbol{w}^\top \boldsymbol{x}_i + b &\geq 1 && \text{if } y_i = 1 \\ \boldsymbol{w}^\top \boldsymbol{x}_i + b &\leq -1 && \text{if } y_i = -1 \end{aligned}$$

These two constraints can be combined and written as a single constaint

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1, \quad i \in 1 \dots N$$

Hence, the problem can be written as a constrained optimisation problem

$$\begin{gathered} \min_{w,b} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} \\ \text{Subject to } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1, \quad i \in 1 \dots N \end{gathered}$$

11.1 Lagrangian Duality

We can use the method of Lagrange Multiplier to solve the constrained optimisation problem. Given an optimisation of the form

$$\begin{gathered} \min_{\boldsymbol{w}} f(\boldsymbol{w}) \\ \text{Subject to } g(\boldsymbol{w}) \leq 0 \end{gathered}$$

We first define the Lagrangian to be the original objective function added to a weighted combination of the constraints.

$$\mathcal{L}(\boldsymbol{w}, a) = f(\boldsymbol{w}) + ag(\boldsymbol{w})$$

Then rewrite the original optimisation problem in the following form where $\boldsymbol{p}$ is the **primal solution**.

$$\boldsymbol{p} = \min_{\boldsymbol{w}} \max_{a \geq 0} \mathcal{L}(\boldsymbol{w}, a)$$

Optimisation problems can be viewed from a different perspective as a dual problem, $\boldsymbol{d}$ is the **dual solution**.

$$\boldsymbol{d} = \max_{a \geq 0} \min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, a)$$

The solution to the dual problem provides a lower bound to the solution of the primal problem. The two solutions are equal if the following conditions are met

- $f(\boldsymbol{w})$ is convex
- $g(\boldsymbol{w})$ is affine

For SVM problems, these conditions hold and therefore we can solve for the optimisation problem using the dual.

11.1.1 Karush-Kuhn-Tucker Conditions

The method of Lagrange Multipliers is used to find the solution for optimisation problems constrained to one or more equalities. When our constraints also have inequalities, we need to extend the method to the KKT conditions:

- $g(\boldsymbol{w}) \leq 0$ and $a \geq 0$
- $ag(\boldsymbol{w}) = 0$

The KKT conditions hold at the optimal solution.

11.2 Solving SVM

Going back to the original problem

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w}$$
$$\text{Subject to } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1, \;\; i \in 1 \dots N$$

We can now define the Lagrangian as

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \sum_{i=1}^{N} a_i \left(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1 \right)$$

And solve for the dual problem

$$\max_{a_i \geq 0} \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a})$$

To solve for the dual, we first minimise $\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a})$ with respect to $\boldsymbol{w}$ and b for a fixed value $\boldsymbol{a}$.

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = \boldsymbol{w} - \sum_{i=1}^{N} a_i y_i \boldsymbol{x}_i \qquad \nabla_b \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = -\sum_{i=1}^{N} a_i y_i$$

$$\nabla_w \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N} a_i y_i \boldsymbol{x}_i \qquad \nabla_b \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = 0 \Rightarrow \sum_{i=1}^{N} a_i y_i = 0$$

Putting this back into the original equation

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) &= \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \sum_{i=1}^{N} a_i \left(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1 \right) \\
\mathcal{L}(\boldsymbol{a}) &= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j - \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j - b \sum_{i=1}^{N} a_i y_i + \sum_{i=1}^{N} a_i \\
&= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j - b(0) + \sum_{i=1}^{N} a_i \\
&= \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j
\end{aligned}$$

Putting this together with the constraints $a_i \geq 0$ and $\sum a_i y_i = 0$. We obtain the following optimisation problem

$$\begin{aligned}
&\max_{\boldsymbol{a}} \mathbf{1}^\top \boldsymbol{a} - \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K} \boldsymbol{a} \qquad\qquad \boldsymbol{K} = y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \\
&\text{Subject to } a_i \geq 0, \ \ i \in 1 \ldots N \\
&\text{Subject to } \boldsymbol{a}^\top \boldsymbol{y} = 0
\end{aligned}$$

11.3 Slack Variables

Sometimes the training data is not linearly separable, we can introduce slack variables, ξ, to each data to allow misclassification. We define ξ for each data to be

$$\xi = \begin{cases} 0, \text{ Point is on the correct side of the margin} \\ 0 < \xi \leq 1, \text{ Point is inside margin but correct side of hyperplane} \\ > 1, \text{ Point is at wrong side of hyperplane} \end{cases}$$

Introducing slack variables, the optimisation problem becomes

$$\begin{aligned}
&\min_{\boldsymbol{w}, b} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{N} \xi_i \\
&\text{Subject to } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \ \ i \in 1 \ldots N \\
&\text{Subject to } \xi_i \geq 0, \ \ i \in 1 \ldots N
\end{aligned}$$

Where C is a hyperparameter called the regularisation term. A smaller C leads to larger margin and is able tolerate more mistakes, whereas a larger C will make constraints hard to be ignored. The Lagrangian is define as

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{r}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} a_i \left(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1 + \xi_i \right) - \sum_{i=1}^{N} r_i \xi_i$$

Same as the previous section, we first minimise $\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{r})$ with respect to $\boldsymbol{w}$, b and $\boldsymbol{\xi}$.

$$\nabla_{\xi_i} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{r}) = C - a_i - r_i$$
$$\nabla_{\xi_i} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{r}) = 0 \Rightarrow C - a_i - r_i = 0$$

Putting this back into the original equation

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{r}) &= \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} a_i \left(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{N} r_i \xi_i \\
\mathcal{L}(\boldsymbol{a}) &= \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} a_i \xi_i - \sum_{i=1}^{N} r_i \xi_i \\
&= \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j + \sum_{i=1}^{N} \xi_i (C - a_i - r_i) \\
&= \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j
\end{aligned}$$

Putting this together with the new constraint $C - a_i - r_i = 0 \Rightarrow a_i \leq C$. We obtain the following optimisation problem

$$\max_{\boldsymbol{a}} \mathbf{1}^\top \boldsymbol{a} - \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K} \boldsymbol{a} \qquad \boldsymbol{K} = y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$$
$$\text{Subject to } 0 \leq a_i \leq C, \quad i \in 1 \ldots N$$
$$\text{Subject to } \boldsymbol{a}^\top \boldsymbol{y} = 0$$