Contents

Preface

PDF Version

Linear Algebra

Scalar, Vectors, Matrix and Tensor

- A scalar is a single number $s \in \mathbb{R}$. It can be regarded as a vector with the unit length or dimension.
- A vector is an array of numbers $v \in \mathbb{R}^d$. The d is called dimension or length. A vector can be regarded as a $m \times 1$ matrix.
- A matrix is a two-dimensional array $M \in \mathbb{R}^{m \times n}$. A matrix can be regarded as a special tensor.
- A tensor is an n-dimensional array $T \in \mathbb{R}^{m_1 \times m_2 \times \cdots}$. Tensor is the most generic way to denote data.

Operations

Products

We can define **inner product**, or known as **dot product** on vectors.

Norms

Determinant

- $\det(A^T) = \det(A)$.
- $\det(AB) = \det(A)\det(B)$.

Inversion

A matrix $A \in \mathbb{R}^{n\times n}$ is called invertible if there is $A^{-1} \in \mathbb{R}^{n\times n}$ such that $AA^{-1} = A^{-1}A = I$.

- A is invertible if and only if rows of A are linearly independent.
- A is invertible if and only if $\det(A) = |A| \neq 0$.
- If A is invertible, then $Ax = b$ has solution $x = A^{-1}b$.

Properties of Matrix

- **Symmetric Matrix**: A matrix is called symmetric if $A = A^T$.
- **Diagonal Matrix**: A matrix is called diagonal if $A_{i,j} = 0, i \neq j$.
- **Identity Matrix**: The identity matrix I_n is the $n \times n$ diagonal matrix where $I_{i,i} = 1$.
- **Unitary Matrix**: A complex matrix is called unitary if $UU^* = I$ where U^* is the conjugate transpose of U.

More concepts

- A set of vectors $v_1, \cdots, v_n$ is said to be **linearly independent** if $\sum a_i v_i = 0 \iff a_i = 0$. In another word, $\nexists a_i \in \mathbb{R} - \{0\}$ such that $\sum a_i v_i = 0$.
- The span of $v_1, \cdots, v_n$ for a vector space V is the set of all vectors that can be expressed as a linear combination of them. Formally, we have span$\{v_1, v_2, \cdots, v_n\} = \{v \in V : \exists a_1, \cdots, a_n$ such that $a_1 v_1 + \cdots + a_n v_n = v\}$.

Eigenvalues and Eigenvectors

Definition: $v \in \mathbb{R}^n$ is an **eigenvector** of $A \in \mathbb{R}^{n \times n}$ with **eigenvalue** $\lambda \in \mathbb{R}$ if $Av = \lambda v$.

If $A \in \mathbb{R}^{n \times n}$ has eigenvalues $\lambda_1, \cdots, \lambda_n$, then the determinant of A is $\det(A) = |A| = \prod \lambda_i$.

Steps to compute the eigenvalues and eigenvectors

- Computer the determinant of $A - \lambda I_n$. With λ subtracted along the diagonal, this determinant is a polynomial of degree n, starts with $(-\lambda)^n$.
- Find the roots of this polynomial. The n roots are the eigenvalues of A.
- For each eigenvalue λ solve the equation $(A - \lambda I)x = 0$. The solution $x \neq 0$ is the eigenvector corresponding to λ.

Definite Matrix

A symmetric matrix A is:

- Positive semi-definite (PSD) if for all $x \in \mathbb{R}^n$, there is $x^T Ax \geq 0$.
- Positive definite (PD) if for all non-zero $x \in \mathbb{R}^n$, there is $x^T Ax > 0$.

We can induce this property by using the eigenvalues:

- If all eigenvalues are strictly positive, then matrix A is PD. Furthermore, since the determinant is non-zero, the matrix is invertible.
- Similarly, if all eigenvalues are nonnegative, then the matrix A is PSD.

We can similarly define negative semi-definite and negative definite.

Rayleigh Quotient

$$R(M, v) = \frac{v^T M v}{v^T v}$$

For PSD M,

- The max value of $R(M, v)$ is the largest eigenvalue of M.
- This is attained for v being the corresponding eigenvector.

Singular Value Decomposition

We can decompose a matrix X into $U \sum V^T$ where $X \in \mathbb{R}^{N \times D}$.

- $\sum \in \mathbb{R}^{N \times D}$ is diagonal with singular values $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_D \geq 0$.
- $V \in \mathbb{R}^{D \times D}$ is orthonormal matrix with right singular vectors.
- $U \in \mathbb{R}^{N \times N}$ is orthonormal matrix with left singular vectors.

Calculus

Finding Minimisation

Let a function $f : \mathbb{R}^d \to \mathbb{R}$, then there are two extrema defined:

- Local minimum: x is local minimum for f if $f(x) \leq f(y)$ for all y in some neighbourhood of x.
- Global minimum: x is global minimum for f if $f(x) \leq f(y)$ for all y.

In order to find the extrema, the general approach is by first and second order derivative tests.

Finding the maximum is the same as minimising $-f$, hence it is enough to focus on minimisation.

Functions of one variable $f : \mathbb{R} \to \mathbb{R}$

Continuous and Differentiable Functions

- f is differentiable at x_0 if $f'(x_0) = \frac{d}{dx} f(x_0) = \lim_{h \to 0} \frac{f(x_0+h)-f(x_0)}{h}$ exists.
- Differentiation rules:
  - $\frac{d}{dx} x^n = nx^{n-1}$.
  - $\frac{d}{dx} a^x = a^x ln(a)$.
  - $\frac{d}{dx} log_a(x) = \frac{1}{xln(a)}$.
  - $(f + g)' = f' + g'$.
  - $(f \cdot g)' = f' \cdot g + f \cdot g'$.
- Chain rule:
  - if $f = h(g)$, then $f' = h'(g) \cdot g'$.

First Derivative Test

$f'(x^\star) = 0$ means that $x^\star$ is a critical or stationary point for f. It can be a local minimum, a local maximum or a saddle (minmax) point.

Second Derivative Test

- $f'(x^\star) = 0$ and $f''(x^\star) > 0$ means that f has local minimum at $x^\star$.
- $f'(x^\star) = 0$ and $f''(x^\star) < 0$ means that f has local maximum at $x^\star$.

- $f'(x^\star) = f''(x^\star) = 0$ and $f'''(x^\star) \neq 0$ means that f has a saddle point at $x^\star$.
- Otherwise, higher order derivative tests are necessary.

Functions of multiple variable to a scalar $f : \mathbb{R}^m \to \mathbb{R}$

Partial Derivative and Gradient

Partial derivative of $f(x_1, \cdots, x_m)$ in direction x_i at $a = (a_1, \cdots, a_m)$:

$$\frac{\partial}{\partial x_i} f(a) = \lim_{h \to 0} \frac{f(a_1, \cdots, a_i + h, \cdots, a_m) - f(a_1, \cdots, a_i, \cdots, a_m)}{h}$$

Assume that f is differentiable everywhere, then the gradient is defined as:

$$\nabla_x f = [\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_m}]^T$$

This means that $[\nabla_x f]_i = \frac{\partial f}{\partial x_i}$, and we know $\nabla_x f$ points in direction of the steepest ascent. Hence, $-\nabla_x f$ points in direction of steepest descent.

$x^\star$ is a critical point if $\nabla_x f(x^\star) = 0$.

Hessian Matrix

The Hessian $\nabla_x^2 f$ is a matrix of second-order partial derivatives.

$$\nabla_x^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_m^2} \end{bmatrix}$$

which means $[\nabla_x^2 f]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$.

If the partial derivatives are continuous, the order of differentiation does not matter, the Hessian matrix is always symmetric.

Functions of multiple variable to a vector $f : \mathbb{R}^m \to \mathbb{R}^n$

f given as $f = (f_1, \cdots, f_n)$ with $f_i : \mathbb{R}^m \to \mathbb{R}$, then the *Jacobian* J of f is an $n \times m$ matrix:

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{f_n}{x_m} \end{bmatrix}$$

which means $[J_f]_{i,j} = \frac{\partial f_i}{\partial x_j}$.

Useful Differentiation Rules

- $\nabla_x(c^T x) = c$
- $\nabla_x(x^T x) = 2x$
- $\nabla_x(Ax) = A^T$
- $\nabla_x(x^T Ax) = Ax + A^T x$
- $\nabla_x(f + g) = \nabla_x f + \nabla_x g$
- $\nabla_x(f \cdot g) = f\nabla_x g + g\nabla_x f$

Chain Rule in Higher Dimensions

Let $y = g(x)$ and $z = f(y)$ for $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$, then

$$\frac{\partial z}{\partial x_i} = \sum_{j \in [n]} \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

and $\nabla_x z = J_g \nabla_y z = \frac{\partial y}{\partial x} \nabla_y z$.

Optimization with conditions: Lagrange Multipliers

The constrained optimization problem: we want to maximize $f(x)$ subject to $g_i(x) = 0$ for all $i \in [n]$.

- The optimal points of f lie tangential to the g_i.
- For $n = 1$, optimum should fulfil $\nabla_x f = \lambda \nabla_x g$.
- Optimum of the original optimization problem will be critical points of the Lagrangian: $\wedge(x, \lambda) := f(x) - \lambda \cdot g(x)$.
- This can be generalised to any $n > 0$ and inequality constraints.

Probability Theory

Probability Space

The probability space consists of sample space S and a probability function $p : P(S) \to [0, 1]$ assigning a probability to every event.

It satisfies the axioms of probability:

- $P(\emptyset) = 0$ and $P(S) = 1$.
- Nonnegativity: $(P(A) \geq 0)$. (Any event cannot have negative probability)
- Normalization: $(P(\Omega) = 1)$. (The sum of the probability of all events equals to 1)
- Finite Additivity: If $A \cap B = \emptyset$ (These two events disjoint to each other), then $P(A \cup B) = P(A) + P(B)$.
- Countable Additivity Axiom: If $A_1, A_2, \cdots$ is an infinite sequence of disjoint events, then $P(A_1 \cup A_2 \cup \cdots) = P(A_1) + P(A_2) + \cdots$. (Probability of disjoint events equals to the sum of their probabilities)

With these axioms, we can infer the following properties:

- $P(A) \leq 1$.
- $P(\emptyset) = 0$.
- $P(A) + P(A^c) = 1$.
- For k disjoint events, $P(\{s_1, s_2, \cdots, s_k\}) = P(s_1) + \cdots + P(s_k)$.
- If $A \subset B$, then $P(A) \leq P(B)$.
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
- $P(A \cup B) \leq P(A) + P(B)$. This property is called the *Union Bound*.
- $P(A \cup B \cup C) = P(A) + P(A^c \cap B) + P(A^c \cap B^c \cap C)$.

Conditional Probability

Given events A, B with $P(B) > 0$, then the conditional probability of A given B is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

which indicates that $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$.

Total Probability Law

Given partition $A_1, \cdots, A_n$ of S with $P(A_i) > 0$, then

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Bayes' Rule

We have

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

in which we call $P(A|B)$ posterior, $P(B|A)$ the likelihood and $P(A)$ the prior.

Chain Rule of Conditional Probability

$$P(X_1, \cdots, X_n) = P(X_1) \prod_{i=2}^{n} P(X_i | X_1, \cdots, X_{i-1})$$

Random Variable

A random variable is understood as a measureable function defined on the sample space that maps from the sample space to some numeric domain (usually $\mathbb{R}$). We use $P(X = x)$ to denote the probability of event $\{s \in S : X(s) = x\}$. We write $X \sim P(x)$ to specify probability distribution of X.

Discrete Random Variables

- A random variable is discrete if there are countably many $a_1, a_2, \cdots$ such that $\sum_{aj} P(X = a_j) = 1$.
- The probability mass function (PMF) P_X giving distribution of X as $P_X(x) = P(X = x)$.
- The cumulative distribution function (CDF) $\Phi(x)$ maps x to $P(X \leq x)$.

Continuous Random Variables

- For continuous random variables, the probability density function (PDF) $P(x)$ is the derivative of CDF giving distribution of X. We have:

  $$\int_{-\infty}^{\infty} P(x)dx = 1$$

  and

  $$P(a \leq X \leq b) = \int_a^b p(x)dx$$

Joint Probability Distributions

- The joint probability distribution is a natural generalisation to vectors of random variables giving joint probability distributions. For example, $P(X = x, Y = y)$.
- The marginal probability distribution is defined as: given $P(X, Y)$, obtain $P(X)$ via:
  - For discrete r.v. $P(X = x) = \sum_y P(X = x, Y = y)$.
  - For continuous r.v. $P(X = x) = \int P(x, y)dy$.
- The conditional probabilities: assume that $P(X = x) > 0$, then

  $$P(Y = y | X = x) = \frac{P(Y = y, X = x)}{P(X = x)}$$

- Chain rule of conditional probability:

$$P(X_1, \cdots, X_n) = P(X_1) \cdot \prod_{i=2}^{n} P(X_i | X_1, \cdots, X_{i-1})$$

Expectation and Variance

Expectation

- The expectation for discrete random variables is defined as $\mathbb{E}[X] = \sum_{x \in \text{dom}(X)} x \cdot P(x)$.
- The expectation for continuous random variables is defined as $\mathbb{E}(X) = \int x \cdot P(x) dx$.
- Properties:
  - Linearity: $\mathbb{E}(\alpha X + \beta Y) = \alpha \mathbb{E}(X) + \beta \mathbb{E}(Y)$.

Variance

Variance captures how much values of probability distribution vary on average if randomly drawn. Formally, we have

$$\text{Var}(X) = \mathbb{E}[X - \mathbb{E}(X)^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

Some properties of variance includes:

- $\text{Var}(\alpha X + \beta) = \alpha^2 \text{Var}(X)$.
- If X and Y are independent, then $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y)$.

Standard Deviation

Standard deviation is the square root of the variance, i.e. $SD(X) = \sqrt{Var(X)}$

Covariance

Covariance generalises variance to two random variables, the definition is:

$$\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

Covariance matrix generalises covariance to multiple random variables, the definition is:

$$\sum_{ij} = \text{Cov}(X_i, X_j)$$

Linear Regression

Linear Models

Suppose the input is a vector $x \in \mathbb{R}^d$ and the output is $y \in \mathbb{R}$, and we have the datapoints $(x_1, y_1), \cdots, (x_n, y_n)$. Then we define a linear model as

$$y = w_0 + x_1 w_1 + \cdots + x_D w_D + \epsilon$$

where we call the w_0 as the bias/intercept and the ϵ as the noise/uncertainty.

Learning of Linear Models

Least Squares Objective Function

Assume we have $(x_1, y_1), \cdots, (x_N, y_N)$ where $x_i, y_i \in \mathbb{R}$, and we have the predicted output as $\hat{y} = w_0 + x \cdot w$ (without noise term). We define the least square objective function as

$$\mathcal{L}(w) = \mathcal{L}(w_0, w) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}(w_0 + x \cdot w - y_i)^2$$

This objective function is also called loss function, cost function, energy function, etc. It is known as the residual sum of squares or RSS. The estimated parameters (w_0, w) is known as the least squares etimate.

One Dimensional case

With the objective function defined, we have

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{N}\sum_{i=1}^{N}(w_0 + w x_i - y_i) \frac{\partial \mathcal{L}}{\partial w} = \frac{1}{N}\sum_{i=1}^{N}(w_0 + w x_i - y_i) x_i$$

We obtain the solution for w_0, w by settingthe partial derivatives to 0 and solving the resulting system.

$$w_0 + w \cdot \frac{\sum_i x_i}{N} = \frac{\sum_i y_i}{N} w_0 \cdot \frac{\sum_i x_i}{N} + w\frac{\sum_i x_i^2}{N} = \frac{\sum_i x_i y_i}{N}$$

Since we know these four values:

- $\overline{x} = \frac{\sum_i x_i}{N}$.
- $\overline{y} = \frac{\sum_i y_i}{N}$.
- $\hat{Var}(x) = \frac{\sum_i x_i^2}{N} - \overline{x}^2$.

- $\hat{Cov}(x, y) = \frac{\sum_i x_i y_i}{N} - \overline{xy}$.

Then we end up with

$$w = \frac{\hat{Cov}(x, y)}{\hat{Var}(x)} w_0 = \overline{y} - w\overline{x}$$

General Case

In general case, we express everything in matrix notation. We have $\hat{y} = Xw$, in which $\hat{y} \in \mathbb{R}^{N\times 1}$, $X \in \mathbb{R}^{N\times(D+1)}$ and $w \in \mathbb{R}^{(D+1)\times 1}$. Then we define the loss function as

$$\mathcal{L}(w) = \frac{1}{2N}\sum_{i=1}^{N}(x_i^T w - y_i)^2 = \frac{1}{2N}(Xw - y)^T(Xw - y)$$

Then we can calculate the gradient as

$$\nabla_w \mathcal{L} = \frac{1}{N}((X^T X)w - X^T y)$$

By letting the gradient $\nabla_w \mathcal{L} = 0$, we get $(X^T X)w = X^T y$ and hence $w = (X^T X)^{-1} X^T y$.

The predictions made by this model on the data X is given by

$$\hat{y} = Xw = X(X^T X)^{-1} X^T y$$

where $X(X^T X)^{-1} X^T$ is called the hat matrix.

Rank of the Matrix $X^T X$

Above induction requires $X^T X$ to be invertible. Formally, we have the rank defined as

$$\text{rank}(X^T X) = \text{rank}(X) \leq \min\{D + 1, N\}$$

Then $X^T X$ is invertible if $\text{rank}(X) = D + 1$.

If we use one-hot encoding, where we have $\sum X_i = 1$, we introduce some linear-denpendency in the columns of X and reduce the rank. In this case, we need to drop some features to adjust rank.

Complexity Analysis

There are several steps in calculating the weight matrix:

- Calculate $X^T X$, since $X \in \mathbb{R}^{(D+1)\times N}$, this step takes $O(D^2 N)$.
- Calculate the inverse $(X^T X)^{-1}$, which takes $O(D^3)$.
- Calculate $X^T Y$, which takes $O(DN)$ because $Y \in \mathbb{R}^{N\times 1}$.
- Calculate $(X^T X)^{-1} X^T Y$, which takes $O(D^2)$.

Hence the overall complexity is the sum of all these steps, i.e. $O(D^2N + D^3 + DN + D^2) = O(D^2N + D^3)$.

Joint of Several Relations

If X is defined by a join of several relations, then the number of rows N may be exponential in the number of relations, i.e. $N = O(M^{\text{number of relations}})$.

If X is sparse, then it can be represented in $O(M)$ space losslessly for acyclic joins, which are in common in practice.

In this case, W can still be calculated in $O(D^2M + D^3)$.

Expansion and Kernel Methods

We can perform expansions in higher dimensions. For example, we can expand the simple linear models $\psi(x) = [1, x_1, x_2]^T$ into quadratic form $\psi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2]^T$. By doing so, we are still fitting linear models, but with more, higher-dimensional features.

Using degree d polynomials in k dimensions results in $O(k^d)$ features.

We can use **kernels** as the features. For some expansions, a kernel computes the dot product, i.e. $\kappa(x', x) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \kappa(x', x) = \phi(x') \cdot \phi(x)$.

Some examples of kernels:

- Polynomial kernel: $\kappa_{\text{poly}}(x', x) = (x \cdot x' + \theta)^d$.
- Radial Basis Function (RBF) kernel: $\kappa_{\text{RBF}}(x', x) = \exp(-\gamma||x - x'||^2)$.

Polynomial Kernel

The polynomial kernel is defined as

$$\kappa_{\text{poly}}(x', x) = (x \cdot x' + \theta)^d$$

Since we have the binomial expansion as

$$(z_1 + \cdots + z_k)^d = \sum_{n_i \geq 0, \sum_i n_i = d} \frac{d!}{n_1! \cdots n_k!} z_1^{n_1} \cdots z_k^{n_k}$$

Let C represents the number of ways to distribute d balls into k bins, where the j-th bin holds $n_j \geq 0$ balls. Assume that $z_i = x_i x_i'$ in the above binomial expansion. Then we have

$$(xx')^d = \sum_{n_i \geq 0, \sum_i n_i = d} \sqrt{C(d; n_1, \cdots, n_k)} x_1^{n_1} \cdots x_k^{n_k} \sqrt{C(d; n_1, \cdots, n_k)} (x_1')^{n_1} \cdots (x_k')^{n_k}$$

where $\sum_{n_i \geq 0, \sum_i n_i = d}$ is the dimension of ϕ_{poly}, $\sqrt{C(d; n_1, \cdots, n_k)} x_1^{n_1} \cdots x_k^{n_k}$ is one row in $\phi_{\text{poly}}(x)$ and $\sqrt{C(d; n_1, \cdots, n_k)} (x'_1)^{n_1} \cdots (x'_k)^{n_k}$ is one row in $\phi_{\text{poly}}(x')$.

The dimension of the vector $\phi_{\text{poly}}(x)$ and $\phi_{\text{poly}}(x')$ is $O(k^d)$.

The complexity for computing $\kappa(x', x)$ is $O(k^d)$ using ϕ_{poly} vs. $O(k \log d)$ using $(xx')^d$.

RBF Kernel

A radial basis function (RBF) kernel is defined as

$$\kappa_{\text{RBF}}(x', x) = \exp(-\gamma ||x - x'||^2) = \frac{1}{e^{\gamma ||x - x'||^2}}$$

The hyperparameters in RBF kernel are the width γ and the centres x'.

RBF acts as a similarity measurement between x and x': $\kappa(x', x) \in (0, 1]$.

- If $\gamma ||x - x'|| \to \infty$, then $\kappa_{\text{RBF}}(x', x) \to 0$. This happens when x and x' are far apart (large distance) or γ is very large.
- If $\gamma ||x - x'|| \to 0$, then $\kappa_{\text{RBF}}(x', x) \to 1$. This happens when x and x' are close (small distance) or γ is very small.

Since we have

$$||x - x'||^2 = < x - x', x - x' > = < x, x > + < x', x' > - 2 < x, x' > \ = ||x||^2 + ||x'||^2 - 2xx'$$

and

$$\exp(x, x') = \sum_{k=0}^{\infty} \frac{1}{k!} (xx')^k$$

and the taylor expansion:

$$\exp(z) = \sum_{k=0}^{\infty} \frac{z^k}{k!}$$

Without loss of generality, assume $\gamma = 1$, then we have

$$\exp(-\gamma ||x - x'||^2) = \sum_{k=0}^{\infty} (\sqrt{\frac{1}{k!}} \exp(-||x||^2) \phi_{\text{poly}}(x)) (\sqrt{\frac{1}{k!}} \exp(-||x'||^2) \phi_{\text{poly}}(x'))$$

in which $(\sqrt{\frac{1}{k!}} \exp(-||x||^2) \phi_{\text{poly}}(x))$ is the k-th row in $\phi_{\text{RBF}}(x)$ and $(\sqrt{\frac{1}{k!}} \exp(-||x'||^2) \phi_{\text{poly}}(x'))$ is the k-th row in $\phi_{\text{RBF}}(x')$.

Note that $\phi_{\text{RBF}} : \mathbb{R}^d \to \mathbb{R}^\infty$ projects vectors into an infinite dimensional space, and hence it is not feasible to compute $\kappa_{\text{RBF}}(x', x)$ using ϕ_{RBF}.

Linear Model with RBF Kernel

First we choose centres $\mu_1, \cdots, \mu_M$, and then the feature maps become

$$\psi_{\mathsf{RBF}}(x) = [1, \kappa_{\mathsf{RBF}}(\mu_1, x), \cdots, \kappa_{\mathsf{RBF}}(\mu_M, x)]$$

where we have

$$\kappa_{\mathsf{RBF}}(x', x) = \mathsf{exp}(-\gamma||x - x'||^2)$$

Then the output from a linear model is

$$y = w_0 + \sum_{i=1}^{M} w_i \kappa_{\mathsf{RBF}}(\mu_1, x) + \epsilon = w\psi(x) + \epsilon$$

When choosing hyperparameters:

- The data points themselves can be centres.
- For the width parameter, overfitting or underfitting may occur.
  - If the kernel is too narrow (γ is too large), then overfitting occurs.
  - If the kernel is too wide (γ is too small), then underfitting occurs.

The choice for width parameter is similar with the choice for degree in polynomial basis expansion.

Generalization of Machine Learning

The fundamental goal of machine learning is to generalize beyond the examples in the training set. Hence the test data is highly important, but the data along is not enough.

- Every learner must embody knowledge/assumptions beyond the data.
- No free lunch: no learner can beat random guessing over all possible functions to be learned.
- Hope: Functions to learn in the real world are not drawn uniformly from the set of all mathematically possible functions.
- Reasonable assumptions: similar examples have similar classes; limited dependencies; limited complexity.

Bias-Variance Tradeoff

Having high bias means that we are underfitting, while having high variance means we are overfitting.

To combat overfitting, or high variance, there are several approaches:

- **Cross Validation**.
- **Regularisation** term to the evaluation function. It penalise models with more structure and favors smaller models with less room to overfit.
- **Statistical Significance Test**: like chi-square before adding new structure. It decide whether the prediction is different with or without this structure.

Avoiding overfitting may lead to high bias, a.k.a underfitting.

Regularisation and Validation

Ridge Regression

Suppose we have data $X \in \mathbb{R}^{N \times D}$, where $D \gg N$. One idea to avoid overfitting is to add a penalty term for weights. With a penalty term, our **least squares estimate objective** defined as

$$\mathcal{L}(w) = (Xw - y)^T (Xw - y)$$

changed to the **Ridge Regression Objective** defined as

$$\mathcal{L}_{\text{ridge}}(w) = (Xw - y)^T (Xw - y) + \lambda \sum_{i=1}^{D} w_i^2$$

By doing so, we add a penalty term for all weights but w_0 to control model complexity. The effect of varying w_0 is output translation and not on model complexity, hence we do not control w_0.

In ridge regression, in addition to the residual sum of squares, we penalise the sum of squares of weights. This is also called ℓ_2-regularization or weight-decay. Penalizing weights encourages fitting signals rather than just noise.

Weight from Ridge Regression

We first standardise the input data to make the original data becomes values drawn from standard normal distribution i.e. $\mathcal{N}(0, 1)$. If we center the outputs, i.e. their mean is 0, then w_0 becomes 0 as well. Now we only need to find w that minimise the ridge regression objective, i.e.

$$\mathcal{L}_{\text{ridge}}(w) = (Xw - y)^T (Xw - y) + \lambda w^T w$$

The gradient of the objective with respect to w is

$$\nabla_w \mathcal{L}_{\text{ridge}} = 2(X^T X)w - 2X^T y + 2\lambda w = 2((X^T X + \lambda I_D)w - X^T y)$$

If y is not centred, the gradient of the objective with respect to the leading weight, or b, is

$$\nabla_b \mathcal{L}_{\text{ridge}} = 2Nb - 2\sum_{i=1}^{N} y$$

Set the gradient to 0 and solve for w, we get

$$(X^T X + \lambda I_D)w = X^T y$$

hence,

$$w_{\text{ridge}} = (X^T X + \lambda I_D)^{-1} X^T y$$

$$b_{\text{ridge}} = \frac{1}{N} \sum y_i$$

An alternative formulation of ridge regression is by considering it as a constriained optimisation problem:

$$\text{Minimise} (Xw - y)^T (Xw - y) \text{ subject to } w^T w \leq R$$

The former construction is the Lagrangian formulation of this approach.

Relationship between λ and R

With λ decreasing, the magnitudes of weights start increasing. The larger λ, the smaller R.

LASSO Regression

LASSO represents **L**east **A**bsolute **S**hrinkage and **S**election **O**perator. The objective function of LASSO regression is defined as

$$\mathcal{L}_{\text{Lasso}}(w) = (Xw - y)^T (Xw - y) + \lambda \sum_{i=1}^{D} |w_i|$$

- As with ridge regression, there is a penalty on the weights.
- The absolute value function does not allow for a simple closed-form expression. It is also called ℓ_1-regularization.
- Similar to Ridge regression, there is an alternative formulation as below:

$$\text{Minimise } (Xw - y)^T (Xw - y) \text{ subject to } \sum_{i=1}^{D} |w_i| \leq R$$

- As λ decreasing, the R is increasing and the magnitudes of weights start increasing.

Hyper-Parameter Choice

For ridge regression or lasso or other regularizers, we need to choose λ. To do so, we divide the data into training and validation set. Then we pick the value of λ that minimises the validation error.

If we perform basis expansion,

- For kernels, we need to pick the width parameter γ.
- For polynomials, we need to pick degree d.

Grid search is the main and trivial approach for selecting hyper-parameters. There are generally four steps:

- Assume a small domain D_i for each hyper-parameter λ_i.
- Iterate over all possible combinations of hyper-parameter values $D_1 \times \cdots \times D_k$.
- Perform cross-validation for each combination.
- Pick the combination with the lowest validation error.

K-Fold Cross Validation

When the data is scarce, we can divide the data into K folds (parts), instead of splitting as training and validation. Then we use $K-1$ folds for training and 1 folds for validation.

- We commonly set $K=5$ or $K=10$.
- When K is the number of datapoints, it is called LOOCV (Leave one out cross-validation).
- After the cross-validation, the validation error for fixed hyper-parameter values is the average over all runs.

Logistic Regression

Discriminative and Classification method

Discriminative models the conditional distribution over the output y given the input x and the parameters w.

$$P(y|w,x)$$

In classification, the output y is categorical.

Logistic Regression

Logistic Regression builds up on a linear model composed with sigmoid function, formally, we have

$$P(y|w,x) = \text{Bernoulli}(\text{sigmoid}(w \cdot x))$$

in which $w\log x_0 = 1$, so we do not need to handle the bias term w_0 separately. The sigmoid function σ is defined as

$$\sigma(t) = \frac{1}{1+e^{-t}}, \sigma : \mathbb{R} \to (0,1)$$

in which $t \geq 0$ and hence $\sigma(t) \geq \frac{1}{2}$.

Suppose we have estimated the model parameters $w \in \mathbb{R}^d$. For a new data points x_{new}, the model gives us the probability

$$P(y_{new} = 1|x_{new}, w) = \sigma(w \cdot x_{new}) = \frac{1}{1 + \exp(-x_{new} \cdot w)}$$

In order to make a prediction, we can simply use a threshold at $\frac{1}{2}$.

- The contour line for $P(y = 1|x, w) = P_0$ is given by $xw = \log\frac{P_0}{1-P_0}$.
- The decision boundary is given by $P(y = 1|x, w) = P(y = 0|x, w) = \frac{1}{2}$, hence $xw = 0$.

Multi-Class Logistic Regression

Consider now there are $C > 2$ classes and $y \in \{1, \cdots, C\}$. Then

- There are parameters $w_c \in \mathbb{R}^D$ for every class c.
- The parameters form a matrix $W = [w_1, \cdots, w_c] \in \mathbb{R}^{D \times C}$.
- The multi-class logistic model is given by

$$P(y = c|x, W) = \frac{\exp(w_c^T x)}{\sum_{c'=1}^{C} \exp(w_{c'}^T)x}$$

- Parameter estimation: NLL convex, convex optimisation.
- It can be alternatively expressed using softmax:

$$P(y|x, W) = \text{softmax}([w_1^T x, \cdots, w_C^T x]^T)$$

- Two-class logistic regression is a special case where $W = [w_0, w_1]$:

$$\text{softmax}([w_1^T x, w_0^T x]^T) = \frac{\exp(w_1^T x)}{\exp(w_1^T x) + \exp(w_2^T x)} = \sigma((w_1 - w_0)^T x)$$

Summary

- Logistic Regression is a binary discriminative classification model.
- It can be extended to multiclass by replacing sigmoid with softmax.
- Can derive maximum likelihood estimates using convex optimization.
- Basis expansion and regularisation can be applied to logistic regression as well.

- Regularisation may be necessary if data is linearly separable.
- If the classification boundaries are non-linear
  - Polynomial or kernel-based basis expansion.
  - ℓ_1 or ℓ_2 regularisation if risk of overfitting.

Feature Selection

In the feature selection phase, our goal is select features that are relevant to the model to learn, given a set of possible features.

Premise in feature selection

- Data contains redundant or irrelevant features.
- Removing them does not incur loss of information
- Relevant features maybe redundant in the presence of another relevant feature. Hence two features maybe strongly correlated.

Reasons for Feature Selection

- Enahnced generalisation by reducing overfitting.
- Avoid the curse of dimensionality.
- Shorter training times.
- Simplified models that are easier to interpret by users.

Feature Selection Methods

- We need search technique for subsets of a given set of features.
- We need evaluation measurement to score the different feature subsets.
  - **Wrapper methods**: train a new model for each featuresubset. Score is the count of the number of mistakes on the hold-out set. This method is expensive, yet usually provides the best performing feature set.
  - **Filter methods**: use proxy measurement as score instead of the actual test error. It exposes relationships between features. Typical scores include mutual information, Pearson correlation coefficient. It is fast to compute, yet resulting feature set not tuned to a specific type of models.
  - **Embeded methods**: Perform feature selection whiel constructing the model (e.g. regularizer). Typical examples include LASSO (non-zero parameters for corresponding features) and elastic net regularisation (combines ℓ_1 and ℓ_2 regularisations).

Forward Stepwise Selection

For n features, we ask the following sequence of n questions:

- What is the best 1-feature model? Let the chosen feature be f_1.
- What is the best i-feature model that also ahs the previously selected features $f_1, \cdots, f_{i-1}$? Let the newly chosen feature be f_i.

The output is the best seen k-feature model for any $1 \leq k \leq n$.

Analysis: At each step i, it trains and tests $n - i + 1$ new models, and hence $O(n^2)$ models to train and test in total. For linear regression, it is the same complexity as building one model.

Feature Selection via Mutual Information

The mutual information for two random variables X and Y is defined as

$$I(X, Y) = \sum_x \sum_y P(X = x, Y = y) \log \frac{P(X = x, Y = y)}{P(X = x)P(Y = y)}$$

Approach

- Compute mutual information for each feature and the label/target.
- Only keep the features that provide information about output
  - Ranking of features instead of finding the best subset.
  - Cut-off point using cross-validation.

Computational Considerations

The probabilities $P(X = x), P(Y = y)$ and $P(X = x, Y = y)$ can be empirically obtained from the training set.

- $P(X = x)$ is the fraction of the number of samples with $X = x$ over the number of all samples.
- Variables with continuous domains: first discretise their domains.

Covariance vs. Correlation

Covariance for random variables X and Y measures how the random variables change jointly. It is defined as

$$\text{cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

The (pearson) correlation coefficient normalises the covariance to give a value between -1 and $+1$. It is defined as

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sqrt{\text{cov(X,X)} \cdot \text{cov(Y,Y)}}}$$

Note: Independent variables are uncorrelated, but the converse is not always true.

Support Vector Machines

SVM is a popular discriminative model for classification.

Maximum Margin Principle

Background

Data is linearly separable if there exists a linear separator that classifies all points correctly.

For a data point, its distance to the separating hyperplane is called the margin. When picking the separating boundary, we wish to pick one that **maximises the smallest marging** (the least distance between data and the boundary). That is, we want to maximise the distance of the closest point from the decision boundary. The points that are closet to the decision boundary are called **support vectors**.

Given a hyperplane $H := w \cdot a + w_0 = 0$ and a point $x \in \mathbb{R}^d$, the distance between x and H can be computed by

$$d = \frac{w \cdot x + w_0}{||w||_2}$$

- Positive halfspace: $wx + w_0 > 0$
- Negative halfspace: $wx + w_0 < 0$

Formulation as Optimisation Problem

Our goal as stated before can be formulated as

Find distance $||x - x^\star||_2$ between point $x^\star$ and the hyperplane $w \cdot x + w_0 = 0$ where the distance is given by the point x on the hyperplane that is closest to $x^\star$.

Equivalently, we can formulate it as an optimisation problem. Formally, we want to find x that optimises the following problem

$$\text{minimise } ||x - x^\star||_2^2 \text{ subject to } w \cdot x + w_0 = 0$$

The lagrangian can be formulated as

$$\wedge(x, \lambda) = ||x - x^\star||_2^2 - 2\lambda(w \cdot x + w_0)$$

$$= ||x||_2^2 - 2(x^\star + \lambda w) \cdot x - 2\lambda w_0 + ||x^\star||_2^2$$

By setting the gradient of the lagrangian to 0, we can find its critial point, i.e.

$$\nabla_x \wedge (x, \lambda) = 2x - 2x^\star - 2\lambda w = 0$$

Then we have

$$x = x^\star + \lambda w$$

Then by substituting x into the hyperplane equation, we have

$$w(x^\star + \lambda w) + w_0 = 0$$

hence

$$\lambda = -\frac{wx^\star + w_0}{w \cdot w} = -\frac{wx^\star + w_0}{||w||_2^2}$$

Linearly Separable Case

When the dataset $D = (x_i, y_i)$ is linearly separable, we have

$$y_i(wx_i + w_0) > 0$$

or

$$y_i(\frac{w}{\epsilon}x_i + \frac{w_0}{\epsilon}) \geq 1$$

Margin Maximisation

The margin of data point x_i to hyperplane is defined as

$$\frac{|wx_i + w_0|}{||w||_2} = \frac{y_i(wx_i + w_0)}{||w||_2} \geq \frac{1}{||w||_2}$$

We want to pick a boundary that maximises the margin

$$\text{maximise } \frac{1}{||w||_2} \text{ subject to } y_i(wx_i + w_0) \geq 1$$

Equivalently, we can minimise the squared ℓ_2 norm of w subject to the constraints.

$$\text{minimise } ||w||_2 \text{ subject to } y_i(wx_i + w_0) \geq 1$$

- The objective is a convex quadratic function
- The constraints are convex linear functions

- The feasible set as defined by the constraints is convex as well.

Hence, our optimisation problem is convex quadratic and it solvable using generic convex optimisation methods.

Relaxation of the SVM Formulation

Sometimes, the data is not linearly-separable. In this case, we want to find a separator that makes the least mistakes on the training error. That is, we need to satisfy as many of the N constraints as possible. Alternatively, we allow all constraints to be satisified with some slack variable ξ_i. Then the optimisation problem becomes

$$\text{minimise} ||w||_2^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{subject to } y_i(wx_i + w_0) \geq 1 - \xi_i,\ \ \xi_i \geq 0$$

With ξ_i, a feasible solution always exists because we can let ξ_i very large. The larger the ξ_i, the constraints can be violated as much as necessary.

- Imagine $\xi_i \ll 0$ and x_i are correctly classified by a huge margin.
- The constraints will compensate the penalty incurred on misclassified points.
- $\xi \geq 0$ ensures no bonus for correct classification by a huge margin.

With relaxation, the optimal solution must satisfy

- either $\xi_i = 0$. No slack is needed to classify x_i.
- $\xi_i = 1 - y_i(w \cdot x_i + w_0) \geq 0$: Minimal ξ_i that satisfies the constraint.

Hinge Loss

The hinge loss is defined as

$$\ell_{\text{hinge}} := \max\{0, 1 - y_i(wx_i + w_0)\}$$

Our SVM formulation becomes equivalent to minimising the objective function

$$\ell_{\text{SVM}}(w, w_0|X, y) = C\sum_{i=1}^{N} \ell_{\text{hinge}}(w, w_0; x_i, y_i) + ||w||_2^2$$

Dual SVM Formulation

The lagrange function of SVM can be formulated as

$$\wedge(w, w_0; \alpha) = ||w||_2^2 - \sum_{i=1}^{N} \alpha_i(y_i(wx_i + w_0) - 1)$$

in which w are the primal variables, α_i are the dual variables and $(y_i(wx_i+w_0)-1)$ is the constraints.

The gradients w.r.t the primal variables are

$$\frac{\partial \wedge}{\partial w_0} = -\sum_{i=1}^{N} \alpha_i y_i$$

$$\nabla_w \wedge = w - \sum_{i=1}^{N} \alpha_i y_i x_i$$

Set the gradients to zero, we get

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \;\; w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

For w_0, we can calculate from the constraints, and we will get

$$y_i^2((\sum_{j=1}^{N} \alpha_j y_j x_j)x_i + w_0) = y_i$$

hence

$$w_0 = y_i - \sum_{j=1}^{N} \alpha_j y_j (x_j x_i)$$

It is numerically more stable to have w_0 the average over all possible values

$$w_0 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \sum_{j=1}^{N} \alpha_j y_j (x_j x_i))$$

For α, we can plug the optimal w and constraint $\sum_{i=1}^{N} \alpha_i y_j = 0$ into Lagrangian,

$$g(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j (x_i x_j)$$

To find the critical points of $\wedge$, it is sufficient to find the critical points of g that satisfy the constraints

$$\alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

Compared with Primal form, the dual form

- is a Quadratic concave problem
- there are N variables

- it is a very simple box constraints + one zero-sum constraint.

Predictions with SVM Dual

Prediction on a new point x_{new} requires inner products with the support vectors

$$wx_{\text{new}} = \sum_{i=1}^{N} \alpha_i y_i (x_i x_{\text{new}})$$

We can as well use blackbox access to a function κ that maps two inputs (x, x') to their inner product (xx'). This is called a kernel function. — title: Naive Bayes —

Naive Bayes

Abstractly, naive Bayes is a conditional model: given a problem instance to be classified, represented by a vector $x = (x_1, \cdots, x_n)$ representing some *n* features (independent variables), it assigns to this instance probabilities.

$$P(C_k|x_1, \cdots, x_n)$$

for each of K possible outcomes or classes C_k.

The problem with the above formulation is that if the number of features *n* is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)}$$

Note:

- $P(x)$ should be computed by total probability theorem.
- We assume that $x_1, \cdots, x_n$ are conditionally independent.

Maximum Likelihood

Model and Loss Function Choice

Optimization view of Machine Learning

- Pick a model that we expect may fit the data well enough.
- Pick a measure of performance that makes sense and can be optimised.
- Run optimization algorithm to obtain model parameters.

Probabilistic View of Machine Learning

- Pick a model for data and explicitly formulate the deviation (or uncertainty) from the model using probability.
- Use notions from probability to define suitability of various models.
- Find the parameters or make predictions on unseen data using these suitability criteria.

From a probability perspective, we can approach linear regression as well.

Maximum Likelihood

Given observations $x_1, \cdots, x_N$ indepdently and identically drawn from the same distribution P where P has a parameter θ.

The likelihood of observing $x_1, \cdots, x_N$ is defined as the probability of making these observations assuming they are generated according to P. Formally, the likelihood is defined as $P(x_1, \cdots, x_N|\theta)$: the joint probability distribution of the observations given θ.

Maximum Likelihood Principle

We pick the parameter θ that maximises the likelihood. Formally, we choose

$$\theta^\star = \text{argmax}_\theta P(x_1, \cdots, x_N|\theta)$$

Since we know the observations are i.i.d, we have

$$p(x_1, \cdots, x_N|\theta) = \prod_{i=1}^{N} P(x_i|\theta)$$

MLE in Linear Regression

Recall that our linear model is $y = wx + \epsilon$. Given x, w, the model y can be regarded as a random variable with mean $w^T x$.

We specifically assume that given x, w, then y is normal with mean $w^T x$ and variance σ^2. Formally we have

$$P(y|w, x) = \mathcal{N}(w^T x, \sigma^2) = w^T x + \mathcal{N}(0, \sigma^2)$$

Alternatively, we can view this model as $\epsilon \sim \mathcal{N}(0, \sigma^2)$. This is called Gaussian noise.

This is a **discriminative framework** in which we do not model a distribution over X as the input is fixed.

Discriminative/Generative framework

- Discriminative modeling studies the $P(y|X)$ or the direct maps between the given unobserved variable (target) X and a class label y depended on the observed variables (training samples).
- Generative modelling, which studies from the joint probability $P(X, y)$.

Likelihood of Linear Regression

With the observed data (X, y) made up of $(x_1, y_1), \cdots, (x_N, y_N)$, we want to know the likelihood of observing the data for model parameters w and σ^2. To do so, there are two estimators available.

- MLE Estimator: Find parameters which maximise the likelihood.
- Least Square Estimator: Find parameters which minimise the sum of squares of the residuals.

MLE Estimator

With the observed data $(x_1, y_1), \cdots, (x_N, y_N)$, the likelihood of observing the data for the parameters w, σ is given by

$$P(y|X, w, \sigma) = P(y_1, \cdots, y_N|x_1, \cdots, x_N, w, \sigma) = \prod_{i=1}^{N} P(y_i|x_i, w, \sigma)$$

Since we assume the model is normal, we have $y_i \sim \mathcal{N}(w^T x_i, \sigma^2)$, we have

$$P(y_1, \cdots, y_N|x_1, \cdots, x_N, w, \sigma) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \mathsf{exp}(-\frac{(y_i - w^T x_i)^2}{2\sigma^2})$$

$$= (\frac{1}{2\pi\sigma^2})^{N/2} \mathsf{exp}(-\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - w^T x_i)^2)$$

$$= (\frac{1}{2\pi\sigma^2})^{N/2} \mathsf{exp}(-\frac{1}{2\sigma^2} \sum_{i=1}^{N} (Xw - y)^T (Xw - y))$$

Then the negative log-likelihood will become

$$\mathsf{NLL}(y|X, w, \sigma) = -\mathsf{LL}(y|X, w, \sigma) = \frac{N}{2} \mathsf{log}(2\pi\sigma^2) + \frac{1}{2\sigma^2} (Xw - y)^T (Xw - y)$$

In order to maximise the likelihood, it is equivalent to minimise the negative log-likelihood. Similar to

least square estimator, we end up with

$$\hat{w}_{\mathsf{ML}} = (X^T X)^{-1} X^T y$$

This result is the same as the least square estimator.

The for σ^2, we have

$$\sigma^2_{\mathsf{ML}} = \frac{1}{N}(X w_{\mathsf{ML}} - y)^T (X w_{\mathsf{ML}} - y)$$

To make predictions, we have

$$\hat{y} = w_{\mathsf{ML}} x$$

and we have the confidence interval as

$$y \sim \hat{y} + \mathcal{N}(0, \sigma^2)$$

Optimization

Most machine learning methods can be cast as optimisation problems. Besides of closed-form solutions, there are generally two approaches to solving the problems beyond closed-form solutions:

- Frame the objective of the machine learning problem as a mathematical problem, and use existing blackbox solver for such problems. This is possible when objectives can be formulated as convex optimisation problems.
- Gradient-based optimisation methods.

Convex

Convex Sets

A set $C \subseteq \mathbb{R}^D$ is convex if for any $x, y \in C$ and $\lambda \in [0, 1]$, we have

$$\lambda x + (1 - \lambda) y \in C$$

Some examples of convex sets include

- Set $\mathbb{R}^d$.
- Intersections of convex sets.
- Norm balls. For any norm $||\cdot||$, the set $B = \{x \in \mathbb{R}^d : ||x|| \leq 1\}$ is convex.
- Polyhedra. Given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, the polyhedron $\{x \in \mathbb{R}^n : Ax \leq b\}$ is convex.
- Positive semidefinite cone: The set of positive semi-definite (PSD) matrices.

Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}^n$ defined on a convex domain is convex if

$$\forall x, y \in \mathbb{R}^D, 0 \leq \lambda \leq 1$$

we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

Some examples of convex functions include

- Affine functions: $f(x) = b^T x + c$.
- Quadratic functions: $f(x) = x^T Ax + b^T x + c$ where A is symmetric positive semi-definite.
- Nonnegative weighted sums of convex functions.
- Norms: $|| \cdot ||_p$ except $p = 0$.

Convex Optimisation

Given convex function $f(x), g_1(x), \cdots, g_m(x)$ and affine functions $h_1(x), \cdots, h_n(x)$. A convex optimisation problem is of the form:

$$\text{minimize} f(x) \text{ subject to } g_i(x) \leq 0 \text{ and } h_j(x) = 0$$

where $i \in \{1, \cdots, m\}$ and $j \in \{1, \cdots, n\}$.

The goal is find an optimal value of a convex optimisation problem:

$$V^\star = \text{min}\{f(x) : g_i(x) \leq 0, i \in \{1, \cdots, m\}, h_j(x) = 0, j \in \{1, \cdots, n\}\}$$

Whenever $f(x^\star) = V^\star$, then $x^\star$ is a (not necessarily) unique optimal point.

- $V^\star := \infty$ for infeasible instances. (Feasible means fulfils all constraints g_i and h_j).
- $V^\star := -\infty$ for unbounded instances. (Unbounded means the set of feasible instances has no infimum).

Local Optima and Global Optima

x is locally optimal if

- x is feasible.
- There is $B > 0$ such that $f(x) \leq f(y)$ for all feasble y with $||x - y||_2 \leq B$.

x is globally optimal if

- x is feasible.

- $f(x) \leq f(y)$ for all feasible y.

Theorem: For a convex optimisation problem, all locally optimal points are globally optimal.

Convex Optimisation Problems

Linear Programming

$$\text{minimize } c^T x + d \text{ subject to } Ax \leq e \text{ and } Bx = f$$

There is no closed-form solution for this class of problems. However, efficient algorithms exist, both in theory and practice (for tens of thousands of variables).

Linear Model with Absolute Loss

Suppose we have data (X, y) and that we want to minimise the objective

$$\mathcal{L}(w) = \sum_{i=1}^{N} |x_i^T w - y_i|$$

We want to transform this optimisation problem into a linear program. We introduce a ξ_i for each data point.

The linear program in the $D + N$ variables $w_1, \cdots, w_D, \xi_1, \cdots, \xi_N$. Then our linear program is

$$\text{minimize } \sum_{i=1}^{N} \xi_i$$

$$\text{subject to } w^T x_i - y_i \leq \xi_i \text{ and} y_i - w^T x_i \leq \xi_i$$

where $i \in \{1, \cdots, N\}$.

The solution to this linear program gives w that minimises the objective $\mathcal{L}$.

Maximise Likelihood

Calculating the maximum likelihood is a convex quadratic optimisation problem with no constraints.

Mimising the Lasso Objective

For the Lasso objective, i.e. linear model with ℓ_1-regularisation, we have

$$\mathcal{L}(w) = \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \sum_{i=1}^{D} |w_i|$$

$$= w^T x^T x w - 2y^T X w + y^T y + \lambda \sum_{i=1}^{D} |w_i|$$

- Quadratic part of the loss function cannot be framed as linear program.
- Lasso regularisation does not allow for closed-form solutions.
- It can be rephrased as quadratic programming problem.
- Alternatively, it can be resorted to general optimisation methods.

Quadratically constrained Quadratic Programming

$$\text{minimize } \frac{1}{2} x^T B x + c^T x + d$$

$$\text{subject to } \frac{1}{2} x^T Q_i x + r_i^T x + s_i \leq 0 \text{ and } Ax = b$$

where $i \in \{1, \cdots, m\}$.

Semidefinite Programming

$$\text{minimize } tr(CX) \text{ subject to } tr(A_i X) = b_i$$

where X is positive semidefinite and $i \in \{1, \cdots, m\}$.

Gradient Based Optimisation

Theorem of Gradient

If a function f is differentiable, the gradient of f at that point is either zero or perpendicular to the contour line of f at that point.

Background

Gradient Points in the Direction of Steepest Increase

Each component of the gradient says how fast the function changes w.r.t the standard basis

$$\frac{\partial f}{\partial x}(a) = \lim_{h \to 0} \frac{f(a + h\mathbf{i}) - f(a)}{h}$$

where **i** is the unit vector in the direction of x, i.e. it captures information in which direction we move.

We can also change w.r.t the direction of some arbitrary vector **v**. The derivative in direction of **v** is called **directional derivative**.

$$\nabla_v f(a) = \lim_{h \to 0} \frac{f(a + h\mathbf{v}) - f(a)}{h} = \nabla f(a)\mathbf{v}$$

From an geometric perspective, we multiply $||\nabla f(a)||$ by the scalar projection of **v**. Hence $\nabla f(a)\mathbf{v} = ||\nabla f(a)|| \cdot ||v|| \cos\theta$ where $\cos\theta$ is the angle between $\nabla_f(a)$ and **v**. The maximal value is for $\cos\theta = 1$, or $\theta = 0$, so $\nabla f(a)$ and **v** have the same direction.

Hessian Matrix

Hessian matrix H is the matrix of all second-order partial derivatives of f.

- It is symmetric as long as all second derivatives exist.
- It captures the curvature of the surface.

There are several cases regarding the eigenvalues of H:

- H has positive eigenvalues, then the point x is local minimum.
- H has negative eigenvalues, then the point x is local maximum.
- H has mixed eigenvalues, then the pooint x is saddle point.
- Degenerate case: If the eigenvalue is 0, then there is no inverse and the gradient is unchanging.

Gradient Descent

Gradient descent is one of the simplest, but very general optimisation algorithm for finding a local minimum of a differentiable function. It is iterative and produces a new vector w_{t+1} at each iteration t:

$$w_{t+1} = w_t - \eta_t g_t = w_t - \eta_t \nabla f(w_t)$$

where $\eta_t > 0$ is the learning rate or step size.

At each iteration, it moves in the direction of the steepest descent.

Gradient Descent for Least Squares Regression

Recall the objective function for least squares regression

$$\mathcal{L}(w) = (Xw - y)^T(Xw - y) = \sum_{i=1}^{N}(x_i^T w - y_i)^2$$

We can compute the gradient of $\mathcal{L}$ with respect to w as

$$\nabla_w \mathcal{L} = 2(X^T X w - X^T y)$$

The complexity for gradient descent vs. closed-form solution for very large and wide datasets:

- Computational Complexity of inverting matrices in closed-form solution.
- Each gradient calculation is linear in N and D.

Learning Rate (Step Size) Choice

Choosing a good learning rate is key and we may want a time-varying step size

- If the step size is too large, algorithms may nerver converge.
- If the step size is too small, convergence may be very slow.

When choosing the learning rate, we have the following options:

- Constant step size: $\eta_t = c$.
- Decaying step size: $\eta_t = c/t$. Different rates of decay common, e.g. $\frac{1}{\sqrt{t}}$.
- Backtracking line search.
  - Start with c/t, usually a large value.
  - Check for a decrease: Is $f(w_t - \eta_t \nabla f(w)) < f(w_t)$?
  - If decrease condition not met, multiply η_t by a decaying factor, e.g. 0.5.
  - Repeat until the decrease condition is met.

Test for Convergence

- Fixed number of iterations: Terminate if $t \geq T$.
- Small increase: Terminate if $f(w_{t+1}) - f(w_t) \leq \epsilon$.
- Small change: Terminate if $||w_{t+1} - w_t|| \leq \epsilon$.

Stochastic Gradient Descent (SGD)

We minimise the objective function over data points $(x_1, y_1), \cdots, (x_N, y_N)$, and our objective function is defined as

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} \ell(w; x_i, y_i) + \lambda \mathcal{R}(w)$$

Where the $\lambda \mathcal{R}(w)$ is the regularisation term.

The gradient of the objective function is

$$\nabla_w \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \nabla_w \ell(w; x_i, y_i) + \lambda \nabla_w \mathcal{R}(w)$$

For example, with Ridge Regression, we have

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda w^T w$$

$$\nabla_w \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} 2(w^T x_i - y_i) x_i + 2\lambda w$$

As part of the learning algorithm, we calculate the gradient. Suppose we pick a random datapoint (x_i, y_i) and evaluate $g_i = \nabla_w \ell(w; x_i, y_i)$. Then we have

$$\mathbb{E}[g_i] = \frac{1}{N} \sum_{i=1}^{N} \nabla_w \ell(w; x_i, y_i)$$

In the expectation of g_i, it points in the same direction as the entire gradient (except for the regularisation term).

In SGD, we compute the gradient at one data point instead of at all data points.

In practice, **mini-batch gradient descent** significantly improves the performance, and reduces the variance in the gradients and hence, it is more stable than SGD.

Sub-Gradient Descent

Mimising the Lasso Objective

Recall that in the Lasso objective, i.e. linear model with ℓ_1-regularisation, we have

$$\mathcal{L}(w) = \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \sum_{i=1}^{D} |w_i|$$

$$= w^T x^T x w - 2 y^T X w + y^T y + \lambda \sum_{i=1}^{D} |w_i|$$

- Quadratic part of the loss function can't be framed as linear programming.
- Lasso regularisation does not allow for closed-form solutions.
- It must be resorted to general optimisation methods.

- The objective function is not differentiable.

The sub-gradient descent focuses on the case when f is convex:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \forall x, y, \forall \alpha \in [0,1]$$

At x_0, any g satisfying the following inequality is called a sub-gradient at x_0:

- $f(x) \geq f(x_0) + g(x - x_0)$, where g is a sub-derivative.
- $f(x) \geq f(x_0) + g^T(x - x_0)$, where g is a sub-gradient.

Constrained Convex Optimisation

With constraint on the parameters, we extend the approach from gradient descnt to projected gradient descent.

Projected Gradient Descent

The goal of Projected Gradient Descent is to minimises $f(x)$ subject to additional constraints $w_C \in C$. Formally, we have

$$z_{t+1} = w_t - \eta_t \nabla_f(w_t)$$

$$w_{t+1} = \text{argmin}_{w_c \in C} ||z_{t+1} - w_c||$$

Gradient step is followed by a **projection step**.

Other methods

Second Order Methods

In calculus, our goal is find the roots of a differentiable function f, i.e. solutions to $f(x) = 0$. In optimisations, the goal is to find roots of f', i.e. solutions to $f'(x) = 0$. In this case,

- Function f needs to be twice-differentiable.
- The roots of f' are stationary points of f, i.e. minima/maxima/saddle points.

Newton Methods in One Dimension

- Construct a sequence of points $x_1, \cdots, x_n$ starting with an initial guess x_0.

- Sequence converges towards a minimiser $x^\star$ of f using the sequence of **second-order Taylor approximations of f around the iterates**:

$$f(x) \approx f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2}(x - x_k)^2 f''(x_k)$$

- $x_{k+1} = x^\star$ defined as the minimiser of this quadratic approximation.
- If f'' is positive, then the quadratic approximation is **convex**, and a minimiser is obtained by setting the derivative to zero:

$$0 = \frac{d}{dx}(f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2}(x - x_k)^2 f''(x_k)) = f'(x_k) + (x^\star - x_k)f''(x_k)$$

  Then we have

$$x_{k+1} = x^\star = x_k - f'(x_k)[f''(x_k)]^{-1}$$

Multiclass Classification

Classifiers

In practice, there are two common approaches

- One-vs-One:
  - Train $\binom{c}{2}$ different classifiers for all pairs of classes, i.e. $\frac{c(c-1)}{2}$ classifiers.
  - Each training procedure only uses on average $\frac{2}{k}$ of the training data.
  - For new input, choose the most common classification for the classifiers.
- One-vs-Rest
  - Train C different classifiers, one class vs. the rest $C - 1$.
  - It is typical for classifiers that yield class membership probability or scores.
  - For new input x, pick the class with the largest probability or score.

Measuring Performance

In Regression, we us the same loss function to test data as for training. In classification, we use the number of misclassified data points as classification error.

Confusion Matrix

For binary classification, we can use the confusion matrix.

Prediction	Actual Labels	Actual Labels
	Yes	No
Yes	True Positive	False Positive
No	False Negative	True Negative

- True positive rate, sensitivity, Recall: TPR $= \frac{TP}{TP+FN}$. TPR is the ratio of true positives to actual positives.
- False Positive Rate, Fall-out: FPR $= \frac{FP}{FP+TN}$. FPR is the ratio of false positives to actual negatives.
- True negative rate, Specificity: TNR $= 1 -$ FPR.
- Precision: P $= \frac{\text{TP}}{\text{TP+FP}}$. It is the ratio of true positives to predicted positives.
- Accuracy: Acc $= \frac{\text{TP+TN}}{\text{P+N}}$

The confusion matrix can be easily extended into multiclass classification.

Receiver Operating Characteristic (ROC)

ROC space is defined by $\frac{\text{TPR}}{\text{FPR}}$

Areas under the ROC Curve (AUC)

AUC makes it easy to compare ROC curves for diferent classsifers.

Precision-Recall Curves

Beyond ROC curves, replace FPR with Precision.

Principal Component Analysis (PCA)

Dimensionality Reduction

Real-world data can have redundant information and may have correlated variables. Dimensionality reduction project the data into a lower dimensional subspace while still capturing the essence of the original data. It is usually considered as a preprocessing step before applying other learning algorithms.

PCA is used for dimensionality reduction by identifying a small number of directions which explan most variation in data.

PCA

First we find the principal component as the line passing through the multidimensional mean and minimises the sum of squares of the error.

The second principal component is computed in the same way as the first principal component, after correlation with first principal component has been subtracted from the points. The second principal component is **orthogonal** to the first principal component (PC).

Two Views of PCA

Maximum Variance: we want to find k directions that maximise the variance in the data.

- The first PC v_1 is the direction of largest variance.
- The second PC v_2 is the direction of largest variance orthogonal to v_1.
- The ith PC v_i is the direction of largest variance orthogonal to $v_1, \cdots, v_{i-1}$.

Formally, our data matrix is $X = [x_1^T, \cdots, x_N^T]^T$, and our goal is find $v_1 \in \mathbb{R}^D$, $||v_1|| = 1$ that maximizes $||Xv_1||^2$.

The solution can be derived by:

- Let $z = Xv_1 = [x_1^T v_1, \cdots, x_N^T v_1]$, so $z_i = x_i^T v_1$.
- $||Xv_1||^2$ is the variance of the projections of data onto v_1, then we have

$$||Xv_1||^2 = \sum_{i=1}^{N} z_i^2 = \sum_{i=1}^{N} z^T z = v_1^T X^T X v_1$$

- This assumes that the data is centred $\sum_i x_i = 0$.
- Find $v_2, v_3, \cdots, v_k$ that are all successively orthogonal to previous directions and maximise variance.

$$X_j = X - \sum_{s=1}^{j-1} X v_s v_s^T$$

By applying the Rayleigh quotient, we can get

$$v_1 = \text{argmax}(v_1^T X^T X v_1) = \text{argmax}(\frac{v_1^T X^T X v_1}{v_1^T v_1})$$

Since $X^T X$ is PSD, the largest eigenvalue of $X^T X$ is the maximum value attained by $v_1^T X^T X v_1$ for $||v_1|| = 1$. At this moment, v_1 is the corresponding eigenvector.

Best Reconstruction: we want to find k dimensional subspace with least reconstruction error.

Formally, given i.i.d data matrix $X = [x_1^T, \cdots, x_N^T]^T$, our goal is to find a k-dimensional linear projection that best represents the data.

In case of one PC, the solution is:

- Let v_1 be the direction of projection
- The point x_i is mapped to $\tilde{x_i} = (v_1 x_i)v_1$ where $||v_1|| = 1$
- We minimise the reconstruction error $\sum_{i=1}^N ||x_i - \tilde{x_i}||^2$.

In case of k PCs, the solution is

- Suppose $V_k \in \mathbb{R}^{D\times k}$ is such that columns of V_k are orthogonal.
- Project data X onto subspace defined by V : $Z = XV_k$.
- Minimise the reconstruction error $\sum_{i=1}^N ||x_i - V_k V_k^T x_i||^2$.

Finding PCs with SVD

PCA can be associated with SVD of X, the first k PCs are the first k columns of V.

- PCA is the linear transformation $Z = XV$.
- Let V_k be the first k columns in V, then the dimensionality reduction via PCA: $Z_k = XV_k$.

Sidenotes

Ridge Regularizer

$$\mathcal{L}_{\text{ridge}}(w) = (Xw - y)^T(Xw - y) + \lambda \sum_{i=1}^D w_i^2$$

Lasso Regularizer

$$\mathcal{L}_{\text{Lasso}}(w) = (Xw - y)^T(Xw - y) + \lambda \sum_{i=1}^D |w_i|$$

Generative Models and Discriminative Models

Generative $P(X|Y = y)$: Naive Bayes; Linear Discriminative Analysis;

Discriminative $P(Y|X = x)$: logistic regression; linear regression; K-NN; SVM; Neural Networks;

Newton Methods

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Naive Bayes

$$P(Y = y|X_i = x_i) = \frac{P(x_i|y = c)P(y = c)}{\sum_{y_i} P(x_i|Y = y_i)P(Y = y_i)}$$

Example: exercise 4.4.

Linear Regression

$$\hat{y} = Xw = X(X^T X)^{-1} X^T y$$

If it has solution, then the matrix $X^T X$ must be full rank D, and the rank is bounded by min$\{D, N\}$, hence $D \leq N$.

Distance from point to hyperplane

Hyperplane: $w^T x + b = 0$, point x_0, then the distance is

$$\frac{|w^T x + b|}{||w||}$$

SVM in Dual Form