METHODS of MATHEMATICAL MODELLING 3
Part I: Optimization
Part II: Approximation Theory

Martin Lotz
and
Marie-Therese Wolfram
Mathematics Institute
The University of Warwick

September 2023

Contents

Part I

Optimization

1

Introduction to Optimization

> "[N]othing at all takes place in the universe in which some rule of maximum or minimum does not appear."
>
> — Leonhard Euler

Mathematical optimization, traditionally also known as mathematical programming, is the theory of optimal decision making. Optimization problems arise in a large variety of contexts, including scheduling and logistics, finance, economics, signal processing and machine learning. The underlying mathematical problem always amounts to finding parameters that minimize (if we are dealing with cost) or maximize (if we are dealing with utility) an objective function in the presence or absence of constraints.

What is an optimization problem?

An optimization problem is of the form

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{x} \in \Omega \end{aligned} \tag{1.1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is a real-valued objective function and $\Omega \subseteq \mathbb{R}^n$ is a set of constraints. If $\Omega = \mathbb{R}^n$, then the problem is an unconstrained optimization problem. The constraint set Ω often consists of those $\boldsymbol{x} \in \mathbb{R}^n$ that satisfy certain equations and inequalities:

$$f_1(\boldsymbol{x}) \leq 0, \ldots, f_m(\boldsymbol{x}) \leq 0,\ g_1(\boldsymbol{x}) = 0, \ldots, g_p(\boldsymbol{x}) = 0.$$

A vector $\boldsymbol{x}^*$ satisfying the constraints is called an optimum, a solution, or a global minimizer of the problem (1.1), if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all other $\boldsymbol{x}$ that satisfy the constraints. Note that replacing f by $-f$, we could equivalently state the problem as a maximization problem. Besides global minimizers, there are also other types of minimizers. A point $\boldsymbol{x}^* \in \Omega$ is a local minimizer, if there is an open neighbourhood U of $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in U \cap \Omega$. A local minimizer is called strict, if $f(\boldsymbol{x}^*) < f(\boldsymbol{x})$ for all $\boldsymbol{x} \in U \cap \Omega$, and isolated if it is the only local minimizer in a small enough neighbourhood. In this module we only study continuous optimization problems, where the function f and the functions defining the constraints are continuos. We will not deal with discrete optimization problems, where some of the coordinates are restricted to a countable set, or combinatorial optimization problems, where the optimization is over discrete structures such as graphs.[1]

[1] An example of a combinatorial optimization problem is the famous Travelling Salesman Problem, where given a list of cities and their pairwise distances, the task is to find the shortest possible route that visits each city exactly once and returns to the starting city. Such problems are notoriously difficult to solve.

Examples

The following examples illustrate the range of problems that can be cast as (continuous) optimization problems. You are not expected to understand these examples in detail at this stage, and it is recommended to revisit these examples at the end of the module.

Example 1.1. (Linear Regression) Suppose we want to understand the relationship of a quantity y (for example, sales data) to a series of predictors $x_1, \dots, x_p$ (for example, advertising budget in different media). We can often assume the relationship to be approximately linear,

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon,$$

with random noise ε. The goal is to determine the model parameters $\boldsymbol{\beta} = (\beta_0, \dots, \beta_p)^\intercal$ from observed data. To determine these, we can collect $n \geq p$ sample realizations (from observations or experiments),

$$\{(x_{i1}, \dots, x_{ip}, y_i)\}_{i=1}^n.$$

and find parameters β_i that minimize the squared difference between the labels y_i and the predictions $\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$, called the mean square loss (MSL)

$$f(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2. \tag{1.2}$$

Taking the square instead of just the absolute value of the differences has both theoretical and practical reasons. For the theoretical justification, we refer to statistics modules. The practical reason is that it is easy to minimize quadratic functions (1.2), as we will see. Collecting the data in matrices and vectors,

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} x_{10} & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix},$$

where $x_{i0} = 1$ for $i \in \{1, \dots, n\}$, we can write the MSL concisely as

$$f(\boldsymbol{\beta}) = \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2,$$

where $\|\boldsymbol{x}\|^2 = \sum_{i=1}^d x_i^2 = \boldsymbol{x}^\intercal \boldsymbol{x}$ is the square of the 2-norm of a vector. The optimization problem consists of finding a vector $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ that solves the unconstrained optimization problem of minimizing the quadratic function $f(\boldsymbol{\beta})$ over $\Omega = \mathbb{R}^{p+1}$. Expanding the objective function, we get

$$\begin{aligned} f(\boldsymbol{\beta}) = \frac{1}{n} \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|^2 &\overset{(1)}{=} \frac{1}{n} (\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})^\intercal (\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) \\ &\overset{(2)}{=} \frac{1}{n} \left(\boldsymbol{\beta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X} \boldsymbol{\beta} - 2\boldsymbol{y}^\intercal \boldsymbol{X} \boldsymbol{\beta} + \boldsymbol{y}^\intercal \boldsymbol{y}\right) \\ &\overset{(3)}{=} \frac{1}{n} \left(\sum_{i=1}^{n} \sum_{j=0}^{p} \sum_{k=0}^{p} x_{ij} x_{ik} \beta_j \beta_k - 2 \sum_{i=1}^{n} \sum_{j=0}^{p} y_i x_{ij} \beta_j + \sum_{i=1}^{n} y_i^2 \right), \end{aligned} \tag{1.3}$$

where $\boldsymbol{X}^\intercal$ is the matrix transpose. Here, (1) follows from the definition of the 2-norm, (2) from multiplying out the expression and using that $(\boldsymbol{A} + \boldsymbol{B})^\intercal = \boldsymbol{A}^\intercal + \boldsymbol{B}^\intercal$ and $(\boldsymbol{A}\boldsymbol{B})^\intercal = \boldsymbol{B}^\intercal \boldsymbol{A}^\intercal$, and (3) from writing out

the matrix-vector products. This problem is called quadratic, because it involves at most quadratic terms in the variables $\beta_0, \ldots, \beta_p$ (that is, we have products of at most two of these). One can verify that this is the same expression we get by multiplying out (1.2).

If the columns of $\boldsymbol{X}$ are linearly independent (which, by the way, requires there to be more data than the number of parameters p), this simple optimization problem has a unique closed form solution,

$$\boldsymbol{\beta}^* = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}. \tag{1.4}$$

In practice one would not compute $\boldsymbol{\beta}^*$ by evaluating (1.4). There are more efficient methods available, such as gradient descent or the conjugate gradient method. It is important to note that even in this simple example, solving the optimization problem can be difficult if the number of samples is large.

To illustrate the least squares setting using a concrete example, assume that we have data relating the basal metabolic rate (energy expenditure per time unit) in mammals to their mass.[2] Using data for

573 mammals from the PanTHERIA database[3], we see that the relationship is approximately linear (see Figure 1.1), and can be modelled as

$$Y = \beta_0 + \beta_1 X + \varepsilon.$$

From the data we can assemble the vector $\boldsymbol{y}$ and the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times (p+1)}$ in order to compute $\boldsymbol{\beta} = (\beta_0, \beta_1)^\intercal$ (here, $p = 1$ and $n = 573$). We can find β_0 and β_1 by solving an optimization problem as described above (or solving the problem in closed form). Solving the problem, we get the values $\beta_0 = 1.362$ and $\beta_1 = 0.702$, and we can plot the line and see how it fits the data.

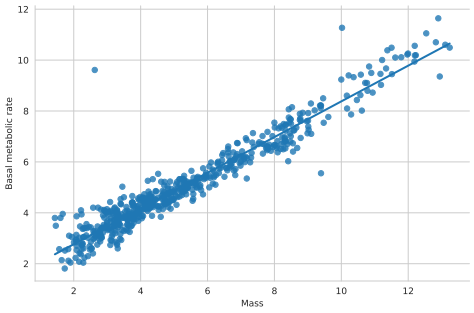


Figure 1.1: Fitting a regression line

Example 1.2. (Linear programming) Suppose an airplane has two cargo compartments with weight capacities $C_1 = 35$ and $C_2 = 40$ tonnes, and volumes (space capacities) $V_1 = 250$ and $V_2 = 400$ cubic meters. Assume we have three types of cargo to transport, specified as follows.

[2] This example is from the episode "Size Matters" of the BBC series Wonders of Life.

[3] http://esapubs.org/archive/ecol/E090/184/#data

	Volume (m^3 per tonne)	Weight (tonnes)	Profit (£/ tonne)
Cargo 1	8	25	£300
Cargo 2	10	32	£350
Cargo 3	7	28	£270

The problem is to decide how much of each cargo to take on board, and how to distribute it in an optimal way among the two compartments.

1. The decision variables x_{ij} specify the amount, in tonnes, of cargo i to go into compartment j. We collect them in a vector $\boldsymbol{x}$.

2. The objective function is the total profit,

$$f(\boldsymbol{x}) = 300 \cdot (x_{11} + x_{12}) + 350 \cdot (x_{21} + x_{22}) + 270 \cdot (x_{31} + x_{32}).$$

3. The constraints are given by the space and weight limitations of the compartments, and the amount of cargo available.

$$\begin{aligned}
x_{11} + x_{12} &\leq 25 && \text{(total amount of cargo 1)} \\
x_{21} + x_{22} &\leq 32 && \text{(total amount of cargo 2)} \\
x_{31} + x_{32} &\leq 28 && \text{(total amount of cargo 3)} \\
x_{11} + x_{21} + x_{31} &\leq 35 && \text{(weight constraint on compartment 1)} \\
x_{12} + x_{22} + x_{32} &\leq 40 && \text{(weight constraint on compartment 2)} \\
8x_{11} + 10x_{21} + 7x_{31} &\leq 250 && \text{(volume constraint on compartment 1)} \\
8x_{12} + 10x_{22} + 7x_{32} &\leq 400 && \text{(volume constraint on compartment 2)} \\
(x_{11} + x_{21} + x_{31})/35 - (x_{12} + x_{22} + x_{32})/40 &= 0 && \text{(maintain balance of weight ratio)} \\
x_{ij} &\geq 0 && \text{(cargo can't have negative weight)}
\end{aligned}$$

It is customary to write the objective function as a scalar product, $f(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle := \boldsymbol{c}^\intercal \boldsymbol{x}$, and to express the constraints as systems of linear equations and inequalities using matrix-vector products,

$$\begin{aligned}
\text{maximize} \quad & \langle \boldsymbol{c}, \boldsymbol{x} \rangle \\
\text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\
& B\boldsymbol{x} = \boldsymbol{d} \\
& \boldsymbol{x} \geq 0
\end{aligned}$$

where the inequalities $\geq$ and $\leq$ are to be understood component wise.

The solution found using a Python package is

$$x_{11} = 6.7500,\ x_{12} = 7.7143,\ x_{21} = 0,\ x_{22} = 32,\ x_{31} = 28,\ x_{32} = 0.$$

We made some simplifying assumptions, for example that the cargo can be split up into arbitrary fractions. Additional work is required to resolve these issues. Problems of this kind are known as linear programming, because the objective function and the constraints are given by linear functions. Such problems can be solved efficiently using the simplex algorithm or interior point methods. The highly developed theory of linear programming acts as a template for the more general theory of convex optimization.

Example 1.3. (Portfolio optimization) Suppose we would like to invest a certain amount of money among n assets (for example, stocks), with x_i denoting the proportion that we put in asset i. The return of asset prices is the relative change in price from one period to the next,

$$r = \frac{p_{\text{new}} - p_{\text{old}}}{p_{\text{old}}},$$

where p_{new} and p_{old} denote the old an new prices. If asset i has return r_i, then the return of the whole portfolio is $r = x_1 r_1 + \cdots + x_n r_n$. As we do not know the true future returns r_i, we have to work with statistical estimates μ_i (for example, the sample mean (average) of previous data). The total expected return of the portfolio is then

$$\mu = x_1\mu_1 + \cdots x_n\mu_n = \boldsymbol{\mu}^\intercal \boldsymbol{x},$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^\intercal$ and $\boldsymbol{x} = (x_1, \ldots, x_n)^\intercal$.

The goal is to select the proportions $\boldsymbol{x}$ in order to achieve a target expected return, while keeping the risk as small as possible. The risk can be measured in various ways, one being through a correlation matrix $\boldsymbol{\Sigma}$ estimated from factors and past data. The risk of taking the positions $\boldsymbol{x}$ can then be modelled as the quadratic function

$$\boldsymbol{x}^\intercal \boldsymbol{\Sigma} \boldsymbol{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \Sigma_{ij},$$

where Σ_{ij} are the entries off the matrix $\boldsymbol{\Sigma}$. The optimization problem we arrive at is

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^\intercal \boldsymbol{\Sigma} \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{\mu}^\intercal \boldsymbol{x} = \mu \\ & \sum_{i=1}^{n} x_i = 1 \\ & \boldsymbol{x} \geq 0. \end{aligned}$$

Like Example 1.1, this problem involves minimizing a quadratic function, and like Example 1.2, it has linear equalities and inequalities as constraints. If we take away the inequality $\boldsymbol{x} \geq 0$ (in financial terms, this corresponds to allowing short-selling), then we can get a closed form solution using the method of Lagrange multipliers. This example will be revisited once we developed the necessary theory.

Example 1.4. (Image inpainting) An image can be viewed as an $m \times n$ matrix $\boldsymbol{U}$, with each entry u_{ij} corresponding to a light intensity (for greyscale images), or a colour vector, represented by a triple of red, green and blue intensities (usually with values between 0 and 255 each). For simplicity the following discussion assumes a greyscale image. For computational purposes, the matrix of an image is often viewed as an mn-dimensional vector $\boldsymbol{u}$, with the columns of the matrix stacked on top of each other.

In the image inpainting problem, one aims to learn the true value of missing or corrupted entries of an image. There are different approaches to this problem. A conceptually simple approach is to replace the image with the closest image among a set of images satisfying typical properties. But what are typical properties of a typical image? Some properties that come to mind are:

- Images tend to have large homogeneous areas in which the colour does not change much;
- Images have approximately low rank, when interpreted as matrices.

Total variation image analysis takes advantage of the first property. The total variation or TV-norm is the sum of the norm of the horizontal and vertical differences,

$$\|\boldsymbol{U}\|_{\mathrm{TV}} = \sum_{i=1}^{m}\sum_{j=1}^{n}\sqrt{(u_{i+1,j}-u_{i,j})^2 + (u_{i,j+1}-u_{i,j})^2},$$

where we set entries with out-of-bounds indices to 0. The TV-norm naturally increases with increased variation or sharp edges in an image. Consider for example the two following matrices (imagine that they represent a 3×3 pixel block taken from an image).

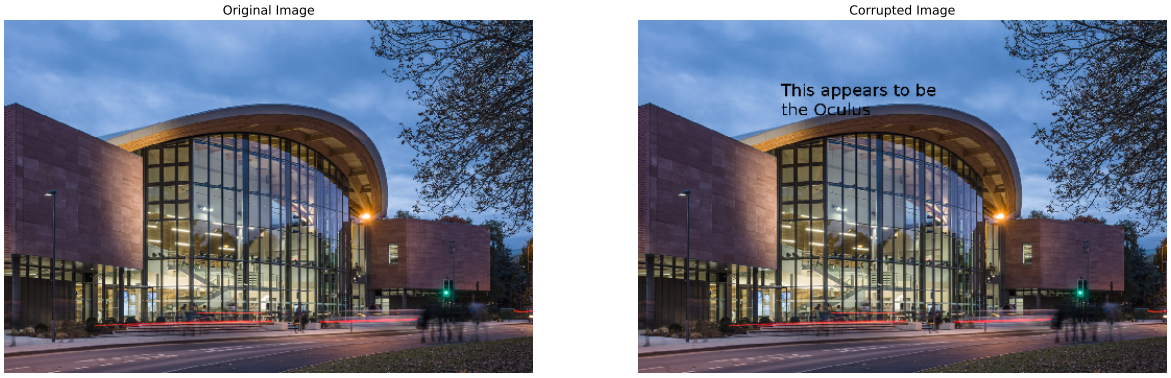
$$\boldsymbol{U}_1 = \begin{pmatrix} 0 & 17 & 3 \\ 7 & 32 & 0 \\ 2 & 9 & 27 \end{pmatrix}, \quad \boldsymbol{U}_2 = \begin{pmatrix} 1 & 1 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

The left matrix has TV-norm $\|\boldsymbol{U}_1\|_{\mathrm{TV}} = 200.637$, while the right one has TV-norm $\|\boldsymbol{U}_2\|_{\mathrm{TV}} = 14.721$ (verify this!). Intuitively, we would expect a natural image with artifacts added to it to have a higher TV norm.

Now let $\boldsymbol{U}$ be an image with entries u_{ij}, and let $\Omega \subset [m] \times [n] = \{(i,j) \mid 1 \le i \le m, 1 \le j \le n\}$ be the set of indices where the original image and the corrupted image coincide (all the other entries are missing). One could attempt to find the image with the smallest TV-norm that coincides with the known pixels u_{ij} for $(i,j) \in \Omega$. This is an optimization problem of the form

$$\text{minimize } \|\boldsymbol{x}\|_{\mathrm{TV}} \quad \text{subject to} \quad x_{ij} = u_{ij} \text{ for } (i,j) \in \Omega. \tag{1.5}$$

The TV-norm is an example of a convex function and the constraints are linear conditions which define a convex set. This is an example of a convex optimization problem and can be solved efficiently by a range of algorithms.



In our first example, we consider an image that has been corrupted. We first determine which entries of the corrupted image are known (the complement of the corrupted pixels), and these will specify the constraints in (1.5). As the problem is rather large (more than a million variables), it is important to choose a good solver that will solve the problem to sufficient accuracy in an acceptable amount of time.

Another typical structure of images is that the singular values of the image, considered as matrix, decay quickly. The singular value decomposition (SVD) of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the matrix product

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}},$$

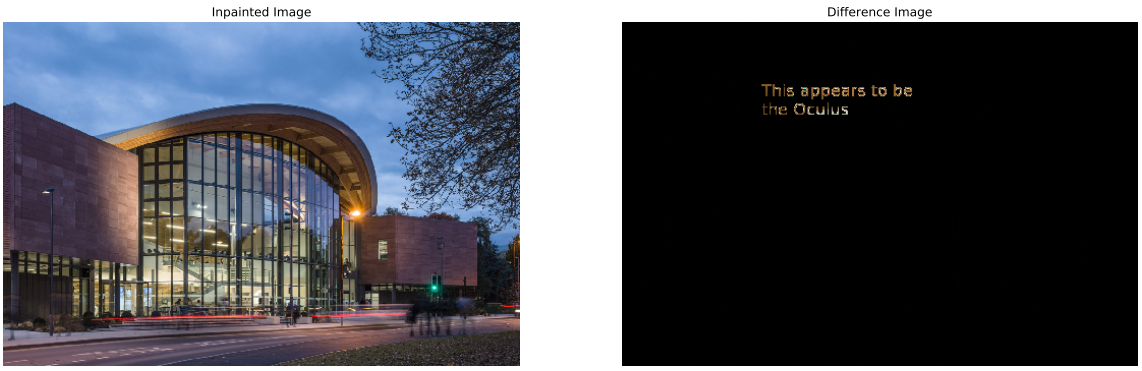


Figure 1.2: Removing writing from the Oculus

where $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix with entries $\sigma_1, \ldots, \sigma_{\min\{m,n\}}$ on the diagonal. Instead of minimizing the TV-norm of an image $\boldsymbol{X}$, one may instead try to minimize the Schatten 1-norm, defined as the sum of the singular values, $\|\boldsymbol{U}\|_{S_1} = \sigma_1 + \cdots + \sigma_{\min\{m,n\}}$. The problem is then

$$\text{minimize } \|\boldsymbol{x}\|_{S_1} \quad \text{subject to} \quad x_{ij} = u_{ij} \text{ for } (i,j) \in \Omega.$$

This is an instance of a type of convex optimization problem known as semidefinite programming. Alternatively, one may also use the 1-norm of the image applied to a discrete cosine transform (DCT) or a discrete wavelet transform (DWT). As this examples (and many more to come) shows: there is no unique choice of objective function to solve a particular problem, and there may be different ways of transforming a practical problem into an optimization problem. These choices depend on model assumptions and require some knowledge of the problem one is trying to solve.

We conclude by using total variation inpainting to liberate a caged parrot.

2

Review of Linear Algebra and Analysis

Without further assumptions, finding a global or local minimizer of a function f is a hopeless task: we cannot simply examine the function at all points in $\mathbb{R}^n$. The situation becomes more tractable if we assume some smoothness conditions. Consider for example a univariate function $f\colon \mathbb{R} \to \mathbb{R}$. If we know that f is continuously differentiable, we know that a local minimum x^* satisfies $f'(x^*) = 0$, i.e., x^* is a stationary point. Even though this condition is not sufficient (x^* could also be a saddle point or maximizer), finding values x with $f'(x) = 0$ narrows down the set of candidates for minimizers considerable, and with some further assumptions on f such as convexity, we can indeed conclude that stationary points are minima. The study of continuous optimization in a multivariate setting requires tools from linear algebra and multivariate calculus.

Linear Algebra

We restrict to linear algebra over the field of real numbers $\mathbb{R}$, as this is the setting that is of most interest in optimization. A vector in $\mathbb{R}^n$ and its transpose are written as

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = (x_1, \ldots, x_n)^\mathsf{T}, \quad \boldsymbol{x}^\mathsf{T} = (x_1, \ldots, x_n),$$

with coordinates $x_i \in \mathbb{R}$ for $1 \leq i \leq n$. The zero vector is denoted by $\mathbf{0}$, while $\boldsymbol{e}$ is the vector with every coordinate equal to 1. If $\lambda_1, \lambda_2 \in \mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, then $\lambda_1 \boldsymbol{x} + \lambda_2 \boldsymbol{y}$ is the vector with coordinates $\lambda_1 x_i + \lambda_2 y_i$ for $1 \leq i \leq n$.

In $\mathbb{R}^n$ we have the Euclidean (or standard) inner product (or scalar product),

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^\mathsf{T} \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i.$$

The Euclidean inner product is just one example of an inner product. In general, an inner product is a function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ that satisfies $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$, with equality if and only if $\boldsymbol{x} = \mathbf{0}$, is symmetric ($\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$) and bilinear: for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$ and $\alpha, \beta \in \mathbb{R}$,

$$\langle \alpha \boldsymbol{x}_1 + \beta \boldsymbol{x}_2, \boldsymbol{y} \rangle = \alpha \langle \boldsymbol{x}_1, \boldsymbol{y} \rangle + \beta \langle \boldsymbol{x}_2, \boldsymbol{y} \rangle.$$

Unless stated otherwise, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ refers to the Euclidean inner product and we will use the notation $\boldsymbol{x}^\mathsf{T} \boldsymbol{y}$ and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ interchangeably, though we prefer the former when referring to properties that apply to any

inner product. Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are called orthogonal, if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. A linear subspace is a subset $V \subseteq \mathbb{R}^n$ such that for any $\boldsymbol{x}, \boldsymbol{y} \in V$ and for all $\alpha, \beta \in \mathbb{R}$, $\alpha \boldsymbol{x} + \beta \boldsymbol{y} \in V$. In particular, the sets $\{\boldsymbol{0}\}$ and $\mathbb{R}^n$ are linear subspaces. A linear combination of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$ is an expression of the form $\boldsymbol{x} = \sum_{i=1}^k \lambda_i \boldsymbol{x}_i$, where $\lambda_i \in \mathbb{R}$ for $1 \leq i \leq k$. The set of linear combinations

$$V = \operatorname{span} \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\} := \left\{ \sum_{i=1}^k \lambda_i \boldsymbol{x}_i \mid \lambda_i \in \mathbb{R} \right\}$$

forms a linear subspace of $\mathbb{R}^n$. It is the intersection of all linear subspaces that contain $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$. The vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are linearly independent if $\sum_{i=1}^k \lambda_i \boldsymbol{x}_i = 0$ implies $\lambda_1 = \cdots = \lambda_k = 0$. A minimal set of vectors that span a linear subspace V is called a basis of this subspace, and the number of elements in a basis is the dimension of the linear subspace. The elements of a basis are always linearly independent, and a maximal linearly independent set in a vector subspace V is a basis. If $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}$ is a basis of a subspace V, then every $\boldsymbol{x} \in V$ has a unique representation $\boldsymbol{x} = \sum_{i=1}^k \lambda_i \boldsymbol{b}_i$. A basis is orthogonal if $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = 0$ for $i \neq j$, and orthonormal if in addition $\langle \boldsymbol{b}_i, \boldsymbol{b}_i \rangle = 1$ for $1 \leq i \leq k$. The unique expression of $\boldsymbol{x} \in V$ as linear combination of an orthonormal basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}$ of V is given by

$$\boldsymbol{x} = \sum_{i=1}^k \langle \boldsymbol{x}, \boldsymbol{b}_i \rangle \boldsymbol{b}_i.$$

The standard basis of $\mathbb{R}^n$ is the orthonormal basis $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, where $\boldsymbol{e}_i$ has a 1 in the i-th coordinate and 0 elsewhere.

An $m \times n$ matrix

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix},$$

represents a linear map from $\mathbb{R}^n$ to $\mathbb{R}^m$ by means of

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \quad y_i = \sum_{j=1}^n a_{ij} x_j.$$

The columns of a matrix are vectors, and we sometimes write

$$\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$$

for the matrix whose columns are given by the vectors $\boldsymbol{a}_i$.

The $n \times n$ matrix $\boldsymbol{1}$ is the matrix with 1 on the diagonal and 0 elsewhere, while $\boldsymbol{0}$ is the matrix consisting of only zeros. The transpose $\boldsymbol{A}^\intercal$ is the matrix with entries $a'_{ij} := a_{ji}$. It is the matrix $\boldsymbol{A}$ mirrored on the diagonal from top left to bottom right. A matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is called symmetric if $\boldsymbol{A}^\intercal = \boldsymbol{A}$. The set of symmetric matrices in $\mathbb{R}^{n \times n}$ is denoted by $\mathcal{S}^n$.

The product of an $m \times p$ matrix $\boldsymbol{A}$ with a $p \times n$ matrix $\boldsymbol{B}$,

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B},$$

is the $m \times n$ matrix $\boldsymbol{C}$ whose (i, j)-th entry is given by

$$c_{ij} = \sum_{k=1}^p a_{ik} b_{kj}.$$

It represents a composition of maps $\mathbb{R}^n \to \mathbb{R}^p \to \mathbb{R}^m$. The number of columns of $\boldsymbol{A}$ has to equal the number of rows of $\boldsymbol{B}$ for this definition to make sense. The matrix $\boldsymbol{1}$ satisfies $\boldsymbol{1}\boldsymbol{A} = \boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{1} = \boldsymbol{A}$, whenever the dimensions are such that this is defined. In general, even if $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$, $\boldsymbol{A}\boldsymbol{B} \neq \boldsymbol{B}\boldsymbol{A}$.

Example 2.1. Let

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 \\ 1 & 4 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix}.$$

Then

$$\boldsymbol{AB} = \begin{pmatrix} 8 & 7 \\ 14 & 11 \end{pmatrix}, \quad \boldsymbol{BA} = \begin{pmatrix} 5 & 16 \\ 5 & 14 \end{pmatrix}.$$

If we consider a vector $\boldsymbol{x} \in \mathbb{R}^n$ as an $n \times 1$ matrix and the transpose as an $1 \times n$ matrix, then for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \cong \mathbb{R}^{n\times 1}$ the Euclidean inner product is

$$\boldsymbol{x}^\intercal \boldsymbol{y} = \boldsymbol{y}^\intercal \boldsymbol{x}.$$

The transpose of a product satisfies $(\boldsymbol{AB})^\intercal = \boldsymbol{B}^\intercal \boldsymbol{A}^\intercal$. From this it follows that for any matrix, $\boldsymbol{A}^\intercal \boldsymbol{A}$ is symmetric. For any matrix $\boldsymbol{A}$ we have

$$\langle \boldsymbol{x}, \boldsymbol{Ax} \rangle = \boldsymbol{x}^\intercal \boldsymbol{Ax} = (\boldsymbol{A}^\intercal \boldsymbol{x})^\intercal \boldsymbol{x} = \langle \boldsymbol{A}^\intercal \boldsymbol{x}, \boldsymbol{x} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j.$$

It follows from this that if a matrix is symmetric, then it is also self-adjoint, meaning $\langle \boldsymbol{x}, \boldsymbol{Ax} \rangle = \langle \boldsymbol{Ax}, \boldsymbol{x} \rangle$.

The rank of a matrix $\boldsymbol{A}$, $\operatorname{rk}(\boldsymbol{A})$, is the maximum number of linearly independent rows or columns of $\boldsymbol{A}$. The kernel and image of $\boldsymbol{A}$ are the linear subspaces

$$\ker \boldsymbol{A} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{Ax} = 0\}, \quad \operatorname{im} \boldsymbol{A} = \{\boldsymbol{Ax} \mid \boldsymbol{x} \in \mathbb{R}^n\}.$$

The dimensions are given by $\dim \ker \boldsymbol{A} = n - \operatorname{rk}(\boldsymbol{A})$ and $\dim \operatorname{im} \boldsymbol{A} = \operatorname{rk}(\boldsymbol{A})$. While $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ represents a linear map from $\mathbb{R}^n$ to $\mathbb{R}^m$, the transpose $\boldsymbol{A}^\intercal$ represents a map in the other direction, and the image of $\boldsymbol{A}^\intercal$ coincides with the orthogonal complement of the kernel of $\boldsymbol{A}$, $(\ker \boldsymbol{A})^\perp = \operatorname{im} \boldsymbol{A}^\intercal$.

A system of linear equations

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ \vdots \qquad\qquad \vdots \qquad &\vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

is written as a matrix vector product

$$\boldsymbol{Ax} = \boldsymbol{b}, \tag{2.1}$$

where the $m \times n$ matrix $\boldsymbol{A}$ is defined as above, and $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$. If the columns of $\boldsymbol{A}$ are linearly independent, then the system of equations can have at most one solution, and otherwise it has infinitely many solutions (this is the case if $n > m$). If $n = m$, then the system (2.1) has a unique solution if and only if the matrix $\boldsymbol{A}$ is invertible or non-singular. This is the case if the rows of $\boldsymbol{A}$ (or equivalently, the columns of $\boldsymbol{A}$) are linearly independent. If $\boldsymbol{A}$ is not invertible, it is called singular.

If $\boldsymbol{A}$ is invertible, there exists a matrix $\boldsymbol{A}^{-1}$ (the inverse) such that

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \mathbf{1}.$$

The solution of (2.1) is then given by $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. The following conditions on a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ are equivalent:

1. $\boldsymbol{A}$ is invertible,

2. $\operatorname{rk}(\boldsymbol{A}) = n$,
3. $\ker \boldsymbol{A} = \{\boldsymbol{0}\}$,
4. $\operatorname{im} \boldsymbol{A} = \mathbb{R}^n$,
5. the rows of $\boldsymbol{A}$ are linearly independent,
6. the columns of $\boldsymbol{A}$ are linearly independent.

A vector $\boldsymbol{u} \neq \boldsymbol{0}$ is an eigenvector of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, if there exists a $\lambda \in \mathbb{C}$ such that

$$\boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}.$$

Such a number λ is called an eigenvalue of $\boldsymbol{A}$. Note that the eigenvectors are only defined up to scaling: if $\boldsymbol{u}$ is an eigenvector, then so is $\lambda\boldsymbol{u}$ for any non-zero $\lambda \in \mathbb{R}$.

The eigenvalues can be complex numbers, and appear in complex conjugate pairs. If the matrix $\boldsymbol{A}$ is symmetric, then the eigenvalues are all real numbers. A matrix has a zero eigenvalue if and only if it is singular. Eigenvalues may occur with multiplicity.

A norm in $\mathbb{R}^n$ is a function $\|\cdot\|$ that satisfies the following three properties

1. $\|\boldsymbol{x}\| \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{x} = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$;
2. $\|\lambda\boldsymbol{x}\| = |\lambda|\|\boldsymbol{x}\|$ for $\lambda \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^n$;
3. $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

Three important examples of norms are the following:

1. The 1-norm: $\|\boldsymbol{x}\|_1 = \sum_{i=1}^n |x_i|$;
2. The 2-norm: $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$;
3. The ∞-norm: $\|\boldsymbol{x}\|_\infty = \max_{1\leq i\leq n} |x_i|$.

Example 2.2. Let $\boldsymbol{x} = (2, -3, 4)^\intercal$. The $\|\boldsymbol{x}\|_1 = 9$, $\|\boldsymbol{x}\|_2 = \sqrt{29}$, and $\|\boldsymbol{x}\|_\infty = 4$.

Note that the 2-norm, also called Euclidean norm, can be defined as

$$\|\boldsymbol{x}\|_2^2 = \boldsymbol{x}^\intercal\boldsymbol{x} = \langle \boldsymbol{x}, \boldsymbol{x}\rangle,$$

that is, it is the norm induced by the Euclidean inner product.

The 1-, 2- and ∞-norms are equivalent, in the sense that they can be bounded in terms of each other. In particular,

$$\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{n}\|\boldsymbol{x}\|_\infty, \quad \|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_1 \leq n\|\boldsymbol{x}\|_\infty. \tag{2.2}$$

The inner product and the 2-norm are related the Cauchy-Schwarz inequality,

$$|\langle \boldsymbol{x}, \boldsymbol{y}\rangle| \leq \|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2,$$

with equality if and only if $\boldsymbol{x}$ and $\boldsymbol{y}$ are linearly dependent. As a consequence of the Cauchy-Schwarz inequality we get

$$-1 \leq \frac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2} \leq 1.$$

The angle between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is the number $\theta \in [0, 2\pi)$ such that

$$\cos(\theta) = \frac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2}.$$

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal, then $\cos(\theta) = 0$ and $\theta \in \{\pi/2, 3\pi/2\}$.

Calculus

We write $C([a,b]) = C^0([a,b])$ for the set of continuous functions on an interval $[a,b]$, and for $k \geq 1$ we write $C^k([a,b])$ for the set of functions continuous on $[a,b]$, and whose first k derivatives $f', \ldots, f^{(k)}$ exist and are continuous on (a,b). In the above definition we allow $a = -\infty$ or $b = \infty$. If $a, b \in \mathbb{R}$, then any function $f \in C([a,b])$ is bounded. The infimum (largest lower bound) and supremum (smallest upper bound) of a function f on an interval $[a,b]$ are defined as

$$\inf_{x\in[a,b]} f(x) = \max\{y \in \mathbb{R} \mid \forall x \in [a,b], f(x) \geq y\},$$
$$\sup_{x\in[a,b]} f(x) = \min\{y \in \mathbb{R} \mid \forall x \in [a,b], f(x) \leq y\}.$$

Again, we allow the "values" $-\infty$ and ∞. If the infimum is attained (i.e., there exists x^* such that $f(x^*) = \inf_{x\in[a,b]} f(x)$), then we write $\min_{x\in[a,b]} f(x)$, and similarly $\max$ if the supremum is attained. Any $f \in C([a,b])$ for $a, b \in \mathbb{R}$ attains its infimum and supremum on $[a,b]$.

Three important results are the Intermediate Value Theorem, the Mean Value Theorem, and the Taylor expansion.

Theorem 2.3 (Intermediate Value Theorem). If $f \in C([a,b])$ and if y satisfies

$$\inf_{x\in[a,b]} f(x) \leq y \leq \sup_{x\in[a,b]} f(x),$$

then there exists $\xi \in [a,b]$ such that $f(\xi) = y$. In particular, the infimum and supremum are attained.

Theorem 2.4 (Mean Value Theorem). Let $f \in C^1([a,b])$ and let $x, x_0 \in (a,b)$ with $x \neq x_0$. Then there exists a number $\xi \in (x_0, x)$ (or (x, x_0) if $x < x_0$) such that

$$f(x) = f(x_0) + f'(\xi)(x - x_0).$$

This can also be written as

$$f'(\xi) = \frac{f(x) - f(x_0)}{x - x_0}.$$

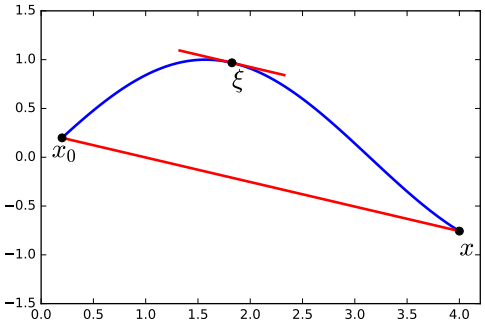


Figure 2.1: MVT: there exists a point at which the slope (derivative) is the same as that of the secant connecting $(x_0, f(x_0))$ and $(x, f(x))$.

The Mean Value Theorem is a special case of the Taylor expansion.

Theorem 2.5 (Taylor expansion). Let $f \in C^{(n)}([a,b])$ and let $x, x_0 \in (a,b)$ with $x \neq x_0$. Then there exists $\xi \in (x, x_0)$ (or (x, x_0) if $x < x_0$) such that

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$

The first $(n+1)$ terms of the above sum can be seen as an approximation to the function f that becomes more accurate as n increases. The last term is known as the truncation error in numerical approximation.

As an example, consider the Taylor expansion of the sine function at $x_0 = 0$,

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

A function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is called (Fréchet) differentiable at $\boldsymbol{x}_0 \in \mathbb{R}^n$ if there exists a linear map $\boldsymbol{J}f(\boldsymbol{x}_0)\colon \mathbb{R}^n \to \mathbb{R}^m$, such that

$$\lim_{\boldsymbol{h}\to\boldsymbol{0}} \frac{\|f(\boldsymbol{x}_0 + \boldsymbol{h}) - f(\boldsymbol{x}_0) - \boldsymbol{J}f(\boldsymbol{x}_0)\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = 0.$$

A function is differentiable on an open subset $U \subseteq \mathbb{R}^n$ if it is differentiable at every $\boldsymbol{x} \in U$. If $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^\intercal$ is differentiable, then all the partial derivatives exist, and $\boldsymbol{J}f(\boldsymbol{x}_0)$ is represented by the Jacobian matrix

$$\mathrm{D}f(\boldsymbol{x}_0) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix},$$

where the partial derivatives are evaluated at $\boldsymbol{x}_0$. If all the partial derivatives exist and are continuous in a neighbourhood of $\boldsymbol{x}_0$ (called continuously differentiable), then f is differentiable at $\boldsymbol{x}_0$.

If $m = 1$, then $\mathrm{D}(\boldsymbol{x}_0)$ is the transpose of the gradient $\nabla f(\boldsymbol{x}_0)$ of f at $\boldsymbol{x}_0$,

$$\nabla f(\boldsymbol{x}_0) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)^\top.$$

The gradient points in the direction in which f increases the most. As in the univariate case, we denote by $C^k(U)$ the set of functions that have continuous partial derivatives up to order k on some set U. The following generalizes the condition that a local minimizer is a stationary point in one dimension.

Theorem 2.6. Let $\boldsymbol{x}^*$ be a local minimizer of f and assume that $f \in C^1(U)$ for a neighbourhood of U of $\boldsymbol{x}^*$. Then $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$.

There are simple examples that show that this is not a sufficient condition: maxima and saddle points will also have a vanishing gradient. Theorem 2.6 is called a necessary condition for a local minumum. If we have access to second-order information, in form of the second derivative, or Hessian, of f, then we can say more.

A convenient way to visualise a function $f\colon \mathbb{R}^2 \to \mathbb{R}$ is through level sets $\{\boldsymbol{x} \in \mathbb{R}^2 \mid f(\boldsymbol{x}) = c\}$. For each $c \in \mathbb{R}$, such a level set defines a curve in $\mathbb{R}^2$, the curve on which the function value does not change.

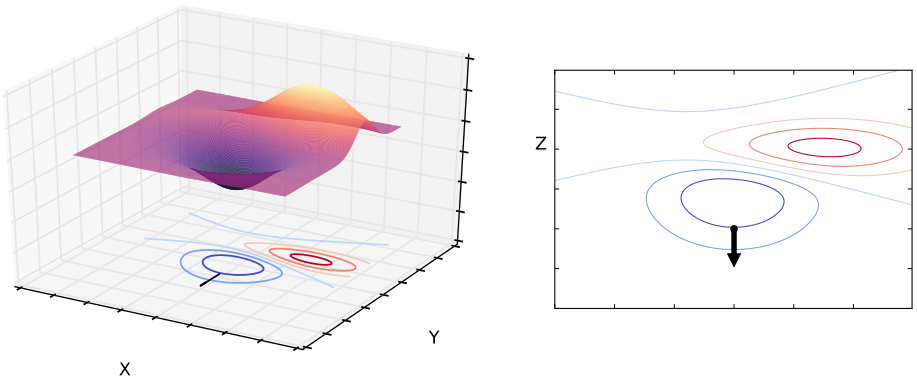


Figure 2.2: A surface, level sets, and the gradient

The gradient is always orthogonal to the level set, pointing in the direction in which f increases the most (see Figure 2.2).

The directional derivative $\mathrm{D}_{\boldsymbol{x}_0} f(\boldsymbol{x}_0)$ of a function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ in direction $\boldsymbol{v} \in \mathbb{R}^n$ at $\boldsymbol{x}_0$ is defined as

$$\mathrm{D}_{\boldsymbol{v}} f(\boldsymbol{x}_0) = \lim_{h \to 0} \frac{f(\boldsymbol{x}_0 + h\boldsymbol{v}) - f(\boldsymbol{x}_0)}{h}.$$

In the special case where $\boldsymbol{v} = \boldsymbol{e}_i$, we obtain the partial derivative with respect to x_i,

$$\frac{\partial f}{\partial x_i}(\boldsymbol{x}_0) = D_{\boldsymbol{e}_i} f(\boldsymbol{x}_0).$$

If $f\colon \mathbb{R}^n \to \mathbb{R}$ is differentiable with continuous derivative near $\boldsymbol{x}_0$, then

$$\mathrm{D}_{\boldsymbol{v}} f(\boldsymbol{x}_0) = \nabla f(\boldsymbol{x}_0)^\intercal \boldsymbol{v} = \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{v} \rangle.$$

If $f\colon \mathbb{R}^n \to \mathbb{R}^m$ and $g\colon \mathbb{R}^m \to \mathbb{R}^p$ are differentiable in a neighbourhood of $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $f(\boldsymbol{x}_0) \in \mathbb{R}^m$, respectively, then the composition $h = g \circ f\colon \mathbb{R}^n \to \mathbb{R}^p$ is continuously differentiable in a neighbourhood of $\boldsymbol{x}_0$, and the Jacobian matrix is defined by the chain rule:

$$\mathrm{D}h(\boldsymbol{x}_0) = \mathrm{D}g(f(\boldsymbol{x}_0))\mathrm{D}f(\boldsymbol{x}_0).$$

If $n = 1$, then $f\colon \mathbb{R} \to \mathbb{R}^n$ is called a curve, and we write

$$\mathrm{D}f = \frac{\mathrm{d}f}{\mathrm{d}t} = \dot{f} = (\dot{f}_1, \ldots, \dot{f}_n)^\intercal \in \mathbb{R}^n$$

for the derivative of the curve. If $\dot{f}(t_0) = \boldsymbol{v} \in \mathbb{R}^n$ and $g\colon \mathbb{R}^n \to \mathbb{R}$, then by the chain rule, the derivative of $g \circ f\colon \mathbb{R} \to \mathbb{R}$ is the directional derivative of g in the direction $\boldsymbol{v}$,

$$\frac{\mathrm{d}g \circ f(t_0)}{\mathrm{d}t} = \langle \nabla g(f(t_0)), \boldsymbol{v} \rangle.$$

The Mean Value Theorem has a generalization to higher dimensions.

Theorem 2.7 (Multivariate Mean Value Theorem). Let $f \in C^1(U)$ for an open set U with $\boldsymbol{x}_0, \boldsymbol{x} \in U$, $\boldsymbol{x} \neq \boldsymbol{x}_0$. Then there exists $t \in (0, 1)$ such that

$$f(\boldsymbol{x}) - f(\boldsymbol{x}_0) = \langle \nabla f(t\boldsymbol{x} + (1-t)\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0 \rangle.$$

Note that $t\boldsymbol{x} + (1-t)\boldsymbol{x}_0$ parametrises the line segment connecting $\boldsymbol{x}$ and $\boldsymbol{x}_0$.

The Hessian of f at $\boldsymbol{x}$, $\nabla^2 f(\boldsymbol{x})$, is the $d \times d$ symmetric matrix given by the second derivatives,

$$\nabla^2 f(\boldsymbol{x}) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)_{1 \le i,j \le d}.$$

In the one-variable case we know that if w^* is a local minimizer of $f \in C^2([a,b])$, then $f'(w^*) = 0$ and $f''(w^*) \ge 0$. Moreover, the conditions $f'(w^*) = 0$ and $f''(w^*) > 0$ guarantee that we have a local minimizer. These conditions generalise to higher dimension, but first we need to know what $f''(w) > 0$ when we have more than one variable.

A matrix $\boldsymbol{A}$ is positive semidefinite, written $\boldsymbol{A} \succeq \boldsymbol{0}$, if for every $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} \ge 0$, and positive definite, written $\boldsymbol{A} \succ \boldsymbol{0}$, if $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} > 0$. The property that the Hessian matrix is positive semidefinite is a multivariate generalization of the property that the second derivative is nonnegative. The known conditions for a minimizer involving the second derivative generalize accordingly.

Theorem 2.8. Let $f \in C^2(U)$ for some open set U and $\boldsymbol{x}^* \in U$. If $\boldsymbol{x}^*$ is a local minimizer, then $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite. Conversely, if $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, then $\boldsymbol{x}^*$ is a strict local minimizer.

The above theorem gives sufficient conditions to characterise local minima. However, the above criteria are not able to identify global minimizers, as differentiability is a local property. For convex functions, however, local optimality implies global optimality.

Example 2.9. Consider a quadratic function of the form

$$f(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^\intercal \boldsymbol{A}\boldsymbol{v} + \boldsymbol{b}^\intercal \boldsymbol{v} + c,$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric. Writing out the product, we get

$$\begin{aligned}
\boldsymbol{v}^\intercal \boldsymbol{A}\boldsymbol{v} &= \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \\
&= \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} \begin{pmatrix} a_{11}v_1 + \cdots + a_{1n}v_n \\ \vdots \\ a_{n1}v_1 + \cdots + a_{nn}v_n \end{pmatrix} \\
&= \sum_{i=1}^n \sum_{j=1}^n a_{ij} v_i v_j.
\end{aligned}$$

Because $\boldsymbol{A}$ is symmetric, we have $a_{ij} = a_{ji}$, and the above product simplifies to

$$\boldsymbol{v}^\intercal \boldsymbol{A}\boldsymbol{v} = \sum_{i=1}^n a_{ii} v_i^2 + 2 \sum_{1 \le i < j \le n} a_{ij} v_i v_j.$$

This is a quadratic function, because it involves products of the v_i. The gradient and the Hessian of $f(\boldsymbol{v})$ are found by computing the partial derivatives of f:

$$\frac{\partial f}{\partial v_i} = \sum_{j=1}^n a_{ij} v_j + b_i, \quad \frac{\partial^2 f}{\partial v_i \partial v_j} = a_{ij}.$$

In summary, we have

$$\nabla f(\boldsymbol{v}) = \boldsymbol{A}\boldsymbol{v} + \boldsymbol{b}, \quad \nabla^2 f(\boldsymbol{v}) = \boldsymbol{A}.$$

Using the previous theorem, we see that f is convex if and only if $\boldsymbol{A}$ is positive semidefinite. A typical example for such a function is

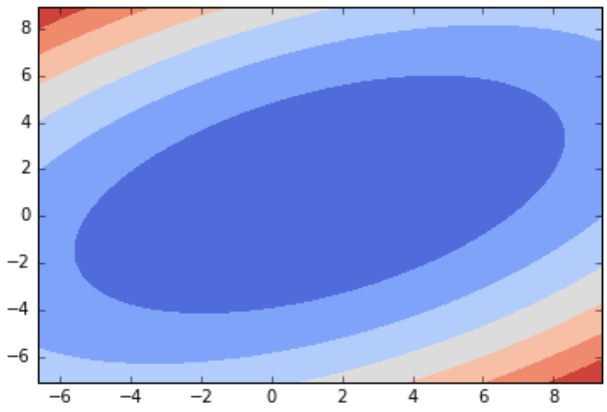
$$f(\boldsymbol{v}) = \|\boldsymbol{A}\boldsymbol{v} - \boldsymbol{b}\|^2 = (\boldsymbol{A}\boldsymbol{v} - \boldsymbol{b})^\intercal(\boldsymbol{A}\boldsymbol{v} - \boldsymbol{b}) = \boldsymbol{v}^\intercal\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{v} - 2\boldsymbol{b}^\intercal\boldsymbol{A}\boldsymbol{v} + \boldsymbol{b}^\intercal\boldsymbol{b}.$$

The matrix $\boldsymbol{A}^\intercal\boldsymbol{A}$ is always symmetric and positive semidefinite (why?) so that the function f is convex.

A convenient way to visualise a function $f\colon \mathbb{R}^2 \to \mathbb{R}$ is through contour plots. A level set of the function f is a set of the form

$$\{\boldsymbol{v} \mid f(\boldsymbol{v}) = c\},$$

where c is the level. Each such level set is a curve in $\mathbb{R}^2$, and a contour plot is a plot of a collection of such curves for various c. If one colours the areas between adjacent curves, one gets a plot as in the following figure. A convex function is has the property that there is only one sink in the contour plot.



3

Convexity 1

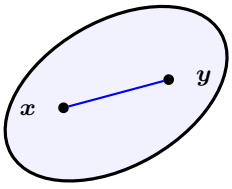
> "[A]lmost every "convex" idea can be explained by a two-dimensional picture."
>
> — Alexander Barvinok

Convexity is a central theme in optimization. The reason is that convex optimization problems have many favourable properties, such the existence of unique global minima. In addition, convex optimization problems can be solved efficiently, leading to the often uttered statement that a problem can be considered solved if it can be cast as a convex optimization problem. Despite being a seemingly special property, convex optimization problems arise surprisingly often (for example, all the problems discussed in the introduction are convex optimization problems), and many difficult optimization problems can be approximated by convex problems.

Convex sets

Convex sets are sets in $\mathbb{R}^n$ such that for any two points in that set, the line segment joining them is also in that set.

Definition 3.1. A set $C \subseteq \mathbb{R}^n$ is a convex set, if for all $\boldsymbol{x}, \boldsymbol{y} \in C$ and $\lambda \in [0,1]$, $\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in C$. A compact (closed and bounded) convex set is called a convex body.



We will denote by $\mathcal{C}(\mathbb{R}^n)$ the collection of convex sets and by $\mathcal{K}(\mathbb{R}^n)$ the collection of convex bodies. The following Lemma is left as an exercise.

Lemma 3.2. Let $C, D \in \mathcal{C}(\mathbb{R}^n)$ be convex sets. Then the following are also convex.

- $C \cap D$;
- $C + D = \{\boldsymbol{x} + \boldsymbol{y} \mid \boldsymbol{x} \in C, \boldsymbol{y} \in D\}$;

- $\boldsymbol{A}C = \{\boldsymbol{A}\boldsymbol{x} \mid \boldsymbol{x} \in C\}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

The convex hull $\operatorname{conv} S$ of a set S is the intersection of all convex sets containing S. Clearly, if S is convex, then $S = \operatorname{conv} S$.

Example 3.3. Let $S = \{(1,1)^\intercal, (1,-1)^\intercal, (-1,1)^\intercal, (-1,-1)^\intercal, (0,0)^\intercal\}$. The convex hull of this set is the square.

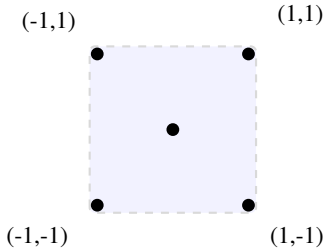


Figure 3.1: A convex hull of five points.

A convex combination of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ is a linear combination

$$\sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i$$

such that $\lambda_i \geq 0$ and $\sum_{i=1}^{k} \lambda_i = 1$. It can be shown inductively that convex sets are closed under convex combinations: any convex combination of points in $C \in \mathcal{C}(\mathbb{R}^n)$ is still in C. In fact, the set of all convex combinations of points in a set S is the convex hull of S.

Lemma 3.4. Let S be a set. Then

$$\operatorname{conv} S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \sum_{i=1}^{k} \lambda_i \boldsymbol{x}_k,\ \boldsymbol{x}_i \in S,\ \sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \geq 0\}.$$

Example 3.5. A hyperplane, defined as the solution set of one linear equation,

$$H = \{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\},$$

is a convex set. Define the halfspaces H_+ and H_- as the two sides that H divides $\mathbb{R}^n$ into:

$$H_- = \{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b\}, \quad H_+ = \{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x} \rangle \geq b\}$$

These are also convex sets.

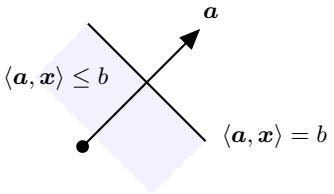


Figure 3.2: Hyperplane and halfspace

Example 3.6. Euclidean balls and ellipsoids are common examples of convex sets. Let $\boldsymbol{P}$ be a positive semidefinite symmetric matrix. Then an ellipsoid with center $\boldsymbol{x}_0$ is a set of the form

$$\mathcal{E} = \{\boldsymbol{x} \mid (\boldsymbol{x} - \boldsymbol{x}_0)^\intercal \boldsymbol{P}^{-1}(\boldsymbol{x} - \boldsymbol{x}_0) \leq 1\}.$$

A Euclidean unit ball is the special case $\boldsymbol{P} = \boldsymbol{I}$.

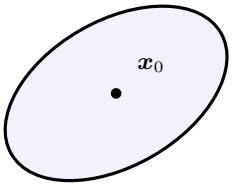


Figure 3.3: An ellipse

Example 3.7. A convex cone is a set C such that for all $\boldsymbol{x}, \boldsymbol{y}$ and $\lambda \geq 0, \mu \geq 0$, $\lambda\boldsymbol{x} + \mu\boldsymbol{y} \in C$. It is easily verified that such a set is convex. Two important cones are the non-negative orthant $\mathbb{R}^n_+ = \{\boldsymbol{x} \in \mathbb{R}^n \mid x_i \geq 0, 1 \leq i \leq n\}$ and the ice-cream (or Lorentz) cone

$$C_\alpha = \{\boldsymbol{x} \mid \sum_{i=1}^{n-1} x_i^2 \leq x_n^2\}.$$

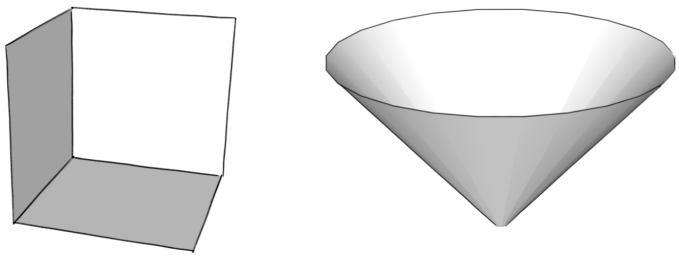


Figure 3.4: The orthant and the second order cone

Possibly the most important result in convex geometry is the hyperplane separation theorem. The following preparatory result is intuitive when thinking of it in terms of a two-dimensional picture.

Lemma 3.8. Let C be a non-empty convex set and $\boldsymbol{x} \notin C$. Then there exists a point $\boldsymbol{y} \in C$ that minimizes the distance $\|\boldsymbol{x} - \boldsymbol{y}\|$. Moreover, for all $\boldsymbol{z} \in C$ we have

$$\langle \boldsymbol{z} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle \leq 0.$$

In words, the vectors $\boldsymbol{z} - \boldsymbol{y}$ and $\boldsymbol{x} - \boldsymbol{y}$ form an obtuse angle.

The proof relies on the following basic fact from analysis.

Fact 3.9. A continuous function on a closed and bounded set attains a maximum and a minimum on that set.

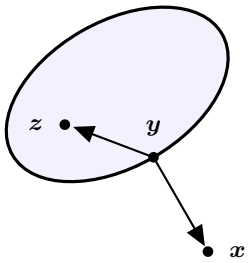


Figure 3.5: Internal and external directions

Proof of Lemma 3.8. Since $C \neq \emptyset$, there exists $r > 0$ such that the ball $B(\boldsymbol{x}, r) := \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{x}\| \leq \varepsilon\}$ intersected with C is not empty. Since $K := C \cap B(\boldsymbol{x}, r)$ is compact (closed and bounded) and the function $\|\boldsymbol{y} - \boldsymbol{x}\|$ is continuous on K, it has a minimizer $\boldsymbol{y} \in K$. For the second claim, note that since C is convex, for every $\lambda \in [0, 1]$,

$$\boldsymbol{w} = \lambda \boldsymbol{z} + (1 - \lambda)\boldsymbol{y} \in C.$$

For the distance between $\boldsymbol{z}$ and $\boldsymbol{x}$ we then get

$$\begin{aligned} \|\boldsymbol{w} - \boldsymbol{x}\|^2 &= \|\lambda \boldsymbol{z} + (1 - \lambda)\boldsymbol{y} - \boldsymbol{x}\|^2 = \|\lambda(\boldsymbol{z} - \boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})\|^2 \\ &= \lambda^2 \|\boldsymbol{z} - \boldsymbol{y}\|^2 - 2\lambda \langle \boldsymbol{z} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle + \|\boldsymbol{x} - \boldsymbol{y}\|^2. \end{aligned}$$

We now prove the claim by contradition. Assume $\langle \boldsymbol{z} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle > 0$. Then we can choose λ such that

$$0 < \lambda < \min \left\{ \frac{2\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{z} - \boldsymbol{y} \rangle}{\|\boldsymbol{z} - \boldsymbol{y}\|^2}, 1 \right\}.$$

With such a λ we get

$$\|\boldsymbol{w} - \boldsymbol{x}\|^2 = \lambda^2 \|\boldsymbol{z} - \boldsymbol{y}\|^2 - 2\lambda \langle \boldsymbol{z} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle + \|\boldsymbol{x} - \boldsymbol{y}\|^2 < \|\boldsymbol{x} - \boldsymbol{y}\|^2.$$

This inequality, however, contradicts the assumption that $\boldsymbol{y}$ is a closest point, so that $\langle \boldsymbol{z} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle \leq 0$ has to hold. □

In what follows write $\mathrm{int} S$ for the interior of a set S.

Theorem 3.10. Let C be a closed convex set and $\boldsymbol{x} \notin C$. Then there exists a hyperplane H such that $C \subset \mathrm{int} H_-$ and $\boldsymbol{x} \in \mathrm{int} H_+$.

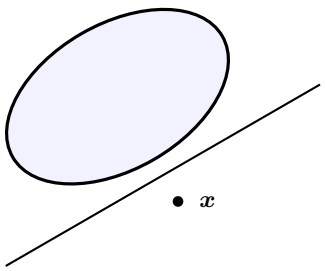


Figure 3.6: A separating hyperplane

Proof. Let $\boldsymbol{y} \in C$ be a nearest point to $\boldsymbol{x}$ in C, i.e., a point such that for all other $\boldsymbol{z} \in C$, $\|\boldsymbol{x} - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{z}\|$. Define

$$\boldsymbol{a} = \boldsymbol{x} - \boldsymbol{y}, \quad b = (\|\boldsymbol{x}\|^2 - \|\boldsymbol{y}\|^2)/2.$$

We aim to show that $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b$ defines a separating hyperplane.

For this we have to show that

1. $\langle \boldsymbol{a}, \boldsymbol{x} \rangle > b$;
2. For all $\boldsymbol{z} \in C$, $\langle \boldsymbol{a}, \boldsymbol{z} \rangle < b$.

For (1), note that

$$\langle \boldsymbol{a}, \boldsymbol{x} \rangle = \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} \rangle > \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} \rangle - \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 = \frac{1}{2}(\|\boldsymbol{x}\|^2 - \|\boldsymbol{y}\|^2) = b.$$

To prove (2), assume on the contrary that there exists a $\boldsymbol{z} \in C$ such that $\langle \boldsymbol{a}, \boldsymbol{z} \rangle \geq b$. We know that the point $\boldsymbol{y} \in C$ satisfies the inequality (2), since

$$\langle \boldsymbol{a}, \boldsymbol{y} \rangle < \langle \boldsymbol{a}, \boldsymbol{y} \rangle + \frac{1}{2}\|\boldsymbol{a}\|^2 = \langle \boldsymbol{a}, \boldsymbol{y} \rangle + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 = b.$$

Therefore,

$$\langle \boldsymbol{a}, \boldsymbol{z} - \boldsymbol{y} \rangle = \langle \boldsymbol{a}, \boldsymbol{z} \rangle - \langle \boldsymbol{a}, \boldsymbol{y} \rangle > b - b = 0,$$

but this contradicts Lemma 3.8. We therefore conclude $\langle \boldsymbol{a}, \boldsymbol{z} \rangle < b$. The separating hyperplane H is thus defined by the equation $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b$. □

4

Convexity 2

Besides convex sets, another important concept is that of a convex function.

Convex functions

Informally, convex functions are functions whose graph between two points lies below the line segment joining these points.

Definition 4.1. Let $S \subseteq \mathbb{R}^n$. A function $f\colon S \to \mathbb{R}$ is called convex if S is convex and for all $\boldsymbol{v}, \boldsymbol{w} \in S$ and $\lambda \in [0,1]$,

$$f(\lambda \boldsymbol{v} + (1-\lambda)\boldsymbol{w}) \leq \lambda f(\boldsymbol{v}) + (1-\lambda) f(\boldsymbol{w}).$$

The function f is called strictly convex if

$$f(\lambda \boldsymbol{v} + (1-\lambda)\boldsymbol{w}) < \lambda f(\boldsymbol{v}) + (1-\lambda) f(\boldsymbol{w}).$$

A function f is called concave, if $-f$ is convex.

Figure 4.1 illustrates what a convex function of one variable looks like. A function f has a domain $\mathrm{dom}(f)$, which is where the function is defined. For example, if $f(x) = \log(x)$, then $\mathrm{dom}(f) = \mathbb{R}_+$, the positive integers. The definition of a convex function thus states that the domain of f is a convex set S. We can also restrict a function on a smaller domain, even though the function could be defined more generally. For example, $f(x) = x^3$ is a convex function if restricted to the domain $\mathbb{R}_{\geq 0}$, but is not convex on $\mathbb{R}$.

A convex optimization problem is an optimization problem in which the set of constraints Ω and the function f are convex. While most general optimization problems are practically intractable, convex optimization problems can be solved efficiently, and still cover a surprisingly large range of applications!

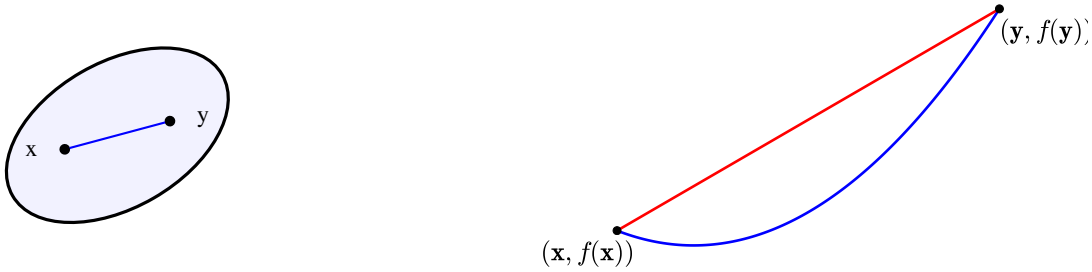


Figure 4.1: A convex set and a convex function

Convex function have pleasant properties, while at the same time covering many of the functions that arise in applications. Perhaps the most important property is that local minima are global minima.

Theorem 4.2. Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a convex function. Then any local minimizer of f is a global minimizer.

Proof. Let $\boldsymbol{v}^*$ be a local minimizer and assume that it is not a global minimizer. Then there exists a vector $\boldsymbol{w} \in \mathbb{R}^d$ such that $f(\boldsymbol{w}) < f(\boldsymbol{v}^*)$. Since f is convex, for any $\lambda \in [0,1]$ and $\boldsymbol{v} = \lambda \boldsymbol{w} + (1-\lambda)\boldsymbol{v}^*$ we have

$$f(\boldsymbol{v}) \leq \lambda f(\boldsymbol{w}) + (1-\lambda) f(\boldsymbol{v}^*) < \lambda f(\boldsymbol{v}^*) + (1-\lambda) f(\boldsymbol{v}^*) = f(\boldsymbol{v}^*).$$

This holds for all $\boldsymbol{v}$ on the line segment connecting $\boldsymbol{w}$ and $\boldsymbol{v}^*$. Since every open neighbourhood U of $\boldsymbol{v}^*$ contains a bit of this line segment, this means that every open neighbourhood U of $\boldsymbol{v}^*$ contains an $\boldsymbol{v} \neq \boldsymbol{v}^*$ such that $f(\boldsymbol{v}) \leq f(\boldsymbol{v}^*)$, in contradiction to the assumption that $\boldsymbol{v}^*$ is a local minimizer. It follows that $\boldsymbol{v}^*$ has to be a global minimizer. □

Remark 4.3. Note that in the above theorem we made no assumptions about the differentiability of the function f! In fact, while a convex function is always continuous on the interior of its domain, it need not be differentiable. The function $f(x) = |x|$ is a typical example: it is convex, but not differentiable at $x = 0$.

Example 4.4. Affine functions $f(\boldsymbol{v}) = \langle \boldsymbol{v}, \boldsymbol{a} \rangle + b$ and the exponential function e^x are examples of convex functions.

Example 4.5. In optimization we will often work with functions of matrices, where an $m \times n$ matrix is considered as a vector in $\mathbb{R}^{m \times n} \cong \mathbb{R}^{mn}$. If the matrix is symmetric, that is, if $\boldsymbol{A}^\intercal = \boldsymbol{A}$, then we only care about the upper diagonal entries, and we consider the space $\mathcal{S}^n$ of symmetric matrices as a vector space of dimension $d = n(n+1)/2$ (the number of entries on and above the main diagonal). Important functions on symmetric matrices that are convex are the operator norm $\|\boldsymbol{A}\|_2$, defined as

$$\|\boldsymbol{A}\|_2 := \max_{\boldsymbol{v}\colon \|\boldsymbol{v}\|=1} \frac{\|\boldsymbol{A}\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2},$$

or the function $\log\det(\boldsymbol{X})$, defined on the set of positive semidefinite symmetric matrices $\mathcal{S}^d_+$.

If $\boldsymbol{A} \in \mathcal{S}^n$ is a symmetric matrix and $\boldsymbol{u} \in \mathbb{R}^n$ an eigenvector with $\|u\|_2 = 1$ and corresponding eigenvalue λ, then $\boldsymbol{u}^\intercal \boldsymbol{A}\boldsymbol{u} = \lambda \boldsymbol{u}^\intercal \boldsymbol{u} = \lambda$. In particular, the largest and smallest values of an eigenvalue are given by

$$\lambda_1 = \max_{\boldsymbol{u}\colon \|\boldsymbol{u}\|_2=1} \boldsymbol{u}^\intercal \boldsymbol{A}\boldsymbol{u}, \quad \lambda_n = \min_{\boldsymbol{u}\colon \|\boldsymbol{u}\|_2=1} \boldsymbol{u}^\intercal \boldsymbol{A}\boldsymbol{u}.$$

A symmetric matrix $\boldsymbol{A}$ is called positive semidefinite, written $\boldsymbol{A} \succeq \boldsymbol{0}$, if for all non-zero $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} \geq 0$, and positive definite, written $\boldsymbol{A} \succ \boldsymbol{0}$, if $\boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$. Equivalently, a symmetric matrix is positive semidefinite if all its eigenvalues are non-negative, and positive definite if they are all positive. The set of positive semidefinite symmetric matrices in $\mathbb{R}^{n \times n}$ is denoted by $\mathcal{S}^n_+$, while the set of positive definite matrices is $\mathcal{S}^n_{++}$.

There are useful ways of characterising convexity using differentiability.

Theorem 4.6.

1. Let $f \in C^1(\mathbb{R}^d)$. Then f is convex if and only if for all $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$,

$$f(\boldsymbol{w}) \geq f(\boldsymbol{v}) + \nabla f(\boldsymbol{v})^\intercal (\boldsymbol{w} - \boldsymbol{v}).$$

2. Let $f \in C^2(\mathbb{R}^n)$. Then f is convex if and only if $\nabla^2 f(\boldsymbol{v})$ is positive semidefinite for all $\boldsymbol{v}$. If $\nabla^2 f(\boldsymbol{v})$ is positive definite for all $\boldsymbol{v}$, then f is strictly convex.

Example 4.7. Consider the function

$$f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2,$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times m}$. Writing everything out, we get

$$f(\boldsymbol{x}) = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ij}a_{ik}x_j x_k - 2\sum_{i=1}^{n}\sum_{j=1}^{m} b_i a_{ij} x_j + \sum_{i=1}^{n} b_i^2\right).$$

Computing the partial derivatives, we get (verify this!)

$$\begin{aligned}\frac{\partial f}{\partial x_j} &= \sum_{j=1}^{m}\left(\sum_{i=1}^{n} a_{ij}a_{ij}\right)x_k - \sum_{i=1}^{n} a_{ij}b_i \\ \frac{\partial^2 f}{\partial x_j x_k} &= \ldots\end{aligned}$$

5

Iterative algorithms 1

Most modern optimization methods are iterative: they generate a sequence of points $\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots$ in $\mathbb{R}^d$ in the hope that this sequences converges to a local or global minimizer $\boldsymbol{w}^*$ of a function $f(\boldsymbol{w})$. A typical rule for generating such a sequence is to start with a vector $\boldsymbol{w}_0$, chosen by an educated guess, and then for $k \geq 0$, move from step k to $k+1$ by

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

in a way that ensures that $f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k)$. The parameter α_k is called the step length (or learning rate in machine learning), while $\boldsymbol{p}_k$ is the search direction. The are many ways in which the direction $\boldsymbol{p}_k$ and the step length α_k can be chosen. If we take

$$\boldsymbol{p}_k = -\nabla f(\boldsymbol{w}_k), \tag{5.1}$$

then we take a step in the direction of steepest descent and the resulting method is (unsurprisingly) called gradient descent. If there is second-order information available, then we can take steps of the form

$$\boldsymbol{p}_k = -\nabla^2 f(\boldsymbol{w}_k)^{-1} \nabla f(\boldsymbol{w}_k). \tag{5.2}$$

The resulting method is called Newton's Method. If applicable, Newton's method converges faster to a solution, but the computation at each step is more expensive.

Gradient descent

In the method of gradient descent, the search direction is chosen as

$$\boldsymbol{p}_k = -\nabla f(\boldsymbol{w}_k).$$

To see why this makes sense, let $\boldsymbol{p}$ be a direction and consider the Taylor expansion

$$f(\boldsymbol{w}_k + \alpha \boldsymbol{p}) = f(\boldsymbol{w}_k) + \alpha \boldsymbol{p}^\intercal \nabla f(\boldsymbol{w}_k) + O(\alpha^2).$$

Considering this as a function of α, the rate of change in direction $\boldsymbol{p}$ at $\boldsymbol{w}_k$ is the derivative of this function at $\alpha = 0$,

$$\frac{\mathrm{d} f(\boldsymbol{w}_k + \alpha \boldsymbol{p})}{\mathrm{d}\alpha}|_{\alpha=0} = \langle \boldsymbol{p}, \nabla f(\boldsymbol{w}_k) \rangle,$$

also known as the directional derivative of f at $\boldsymbol{w}_k$ in the direction $\boldsymbol{p}$. This formula indicates that the rate of change is negative, and we have a descent direction, if $\langle \boldsymbol{p}, \nabla f(\boldsymbol{w}_k) \rangle < 0$.

The Cauchy-Schwarz inequality gives the bounds

$$-\|\boldsymbol{p}\|_2\|\nabla f(\boldsymbol{w}_k)\|_2 \leq \langle \boldsymbol{p}, \nabla f(\boldsymbol{w}_k)\rangle \leq \|\boldsymbol{p}\|_2\|\nabla f(\boldsymbol{w}_k)\|_2.$$

We see that the rate of change is the smallest when the first inequality is an equality, which happens if

$$\boldsymbol{p} = -\alpha\nabla f(\boldsymbol{w}_k)$$

for some $\alpha > 0$.

For a visual interpretation of what it means to be a descent direction, note that the angle θ between a vector $\boldsymbol{p}$ and the gradient $\nabla f(\boldsymbol{w})$ at a point $\boldsymbol{w}$ is given by (see Preliminaries, Page 9)

$$\langle \boldsymbol{p}, \nabla f(\boldsymbol{w})\rangle = \|\boldsymbol{p}\|_2\|\nabla f(\boldsymbol{w})\|_2 \cos(\theta).$$

This is negative if the vector $\boldsymbol{p}$ forms and angle greater than $\pi/2$ with the gradient. Recall that the gradient points in the direction of steepest ascent, and is orthogonal to the level sets. If you are standing on the slope of a mountain, walking along the level set lines will not change your elevation, the gradient points to the steepest upward direction, and the negative gradient to the steepest descent.

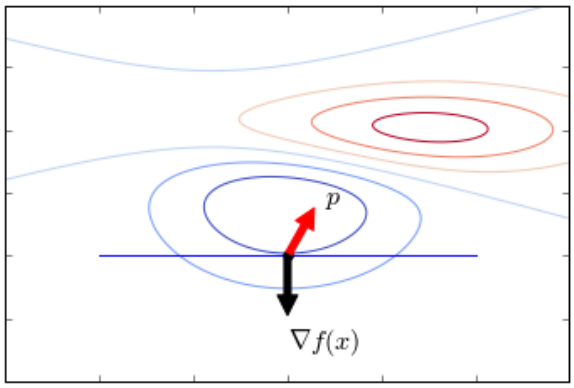


Figure 5.1: A descent direction

Any multiple $\alpha\nabla f(\boldsymbol{w}_k)$ points in the direction of steepest descent, but we have to choose a sensible parameter α to ensure that we make sufficient progress, but at the same time don't overshoot. Ideally, we would choose the value α_k that minimizes $f(\boldsymbol{w}_k - \alpha_k\nabla f(\boldsymbol{w}_k))$. While finding such a minimizer is in general not easy (see Section Lecture 4 for alternatives), for quadratic functions in can be given in closed form.

Linear least squares

Consider a function of the form

$$f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}\|_2^2.$$

The Hessian is symmetric and positive semidefinite, with the gradient given by

$$\nabla f(\boldsymbol{w}) = \boldsymbol{A}^\intercal(\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}).$$

The method of gradient descent proceeds as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{w}_k - \boldsymbol{b}).$$

To find the best α_k, we compute the minimum of the function

$$\alpha \mapsto \varphi(\alpha) = f(\boldsymbol{w}_k - \alpha \boldsymbol{A}^\intercal (\boldsymbol{A}\boldsymbol{w}_k - \boldsymbol{b})). \tag{5.3}$$

If we set $\boldsymbol{r}_k := \boldsymbol{A}^\intercal(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{w}_k) = -\nabla f(\boldsymbol{w}_k)$ and compute the minimum of (5.3) by setting the derivative to zero,

$$\begin{aligned}
\varphi'(\alpha) &= \frac{d}{d\alpha} f(\boldsymbol{w}_k + \alpha \boldsymbol{r}_k) = \langle \nabla f(\boldsymbol{w}_k + \alpha \boldsymbol{r}_k), \boldsymbol{r}_k \rangle \\
&= \langle \boldsymbol{A}^\intercal (\boldsymbol{A}(\boldsymbol{w}_k + \alpha \boldsymbol{r}_k) - \boldsymbol{b}), \boldsymbol{r}_k \rangle \\
&= \langle \boldsymbol{A}^\intercal (\boldsymbol{A}\boldsymbol{w}_k - \boldsymbol{b}), \boldsymbol{r}_k \rangle + \alpha^2 \langle \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{r}_k, \boldsymbol{r}_k \rangle \\
&= -\boldsymbol{r}_k^\intercal \boldsymbol{r}_k + \alpha \boldsymbol{r}_k^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{r}_k = 0,
\end{aligned}$$

we get the step length

$$\alpha_k = \frac{\boldsymbol{r}_k^\intercal \boldsymbol{r}_k}{\boldsymbol{r}_k^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{r}_k} = \frac{\|\boldsymbol{r}_k\|_2^2}{\|\boldsymbol{A}\boldsymbol{r}_k\|_2^2}.$$

Note also that when we have $\boldsymbol{r}_k$ and α_k, we can compute the next $\boldsymbol{r}_k$ as

$$\begin{aligned}
r_{k+1} &= \boldsymbol{A}^\intercal (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{w}_{k+1}) \\
&= \boldsymbol{A}^\intercal (\boldsymbol{b} - \boldsymbol{A}(\boldsymbol{w}_k + \alpha_k \boldsymbol{r}_k)) \\
&= \boldsymbol{A}^\intercal (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{w}_k - \alpha_k \boldsymbol{A}\boldsymbol{r}_k) = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{r}_k.
\end{aligned}$$

The gradient descent algorithm for the linear least squares problem proceeds by first computing $\boldsymbol{r}_0 = \boldsymbol{A}^\intercal(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{w}_0)$, and then at each step

$$\begin{aligned}
\alpha_k &= \frac{\boldsymbol{r}_k^\intercal \boldsymbol{r}_k}{\boldsymbol{r}_k^\intercal \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{r}_k} \\
\boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + \alpha_k \boldsymbol{r}_k \\
\boldsymbol{r}_{k+1} &= \boldsymbol{r}_k - \alpha_k \boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{r}_k.
\end{aligned}$$

Does this work? How do we know when to stop? It is worth noting that the residual satisfies $\boldsymbol{r} = 0$ if and only if $\boldsymbol{w}$ is a stationary point, in our case, a minimizer. One criteria for stopping could then be to check whether $\|\boldsymbol{r}_k\|_2 \leq \varepsilon$ for some given tolerance $\varepsilon > 0$. One potential problem with this criterion is that the function can become flat long before reaching a minimum, so an alternative stopping method would be to stop when the difference between two successive points, $\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\|_2$, becomes smaller than some $\varepsilon > 0$.

Example 5.1. We plot the trajectory of gradient descent with the data

$$\boldsymbol{A} = \begin{pmatrix} 2 & 0 \\ 1 & 3 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}.$$

As can be seen from the plot, we always move in the direction orthogonal to a level set, and stop at a point where we are tangent to a level set.

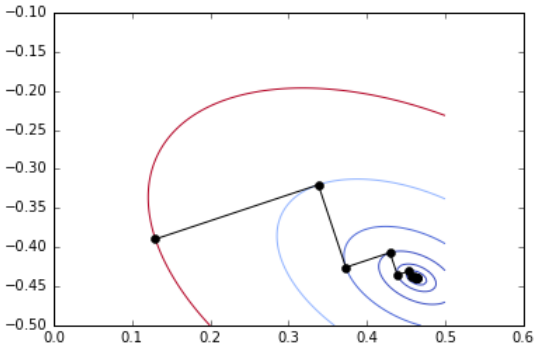


Figure 5.2: Trajectory of gradient descent

6

Iterative algorithms 2

Iterative algorithms for solving a problem of the form

$$\text{minimize } f(\boldsymbol{x}) \tag{6.1}$$

on $\mathbb{R}^n$ generate a sequence of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ in the hope that this sequence converges to a (local or global) minimizer $\boldsymbol{x}^*$ of (6.1). In this lecture we first study step length selection procedures and then study what it means for a sequence to converge, and how to quantify the speed of convergence.

Step length selection

Recall that a descent direction of a function $f\mathbb{R}^n \to \mathbb{R}$ at $\boldsymbol{x} \in \mathbb{R}^n$ is a vector $\boldsymbol{p}$ such that $\langle \boldsymbol{p}, \nabla f(\boldsymbol{x}) \rangle < 0$. When moving from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ along a descent direction $\boldsymbol{p}_k$,

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

we can only guarantee that the function value will decrease if the step length α_k is small enough. If we move too far in a descent direction, we might even land at a point where f is larger than where we started! It is therefore important to choose a step length that

- is not too small (so that the algorithm does not take too long);
- is not too large (we might end up at a point with larger function value);
- is easy to compute.

An optimal step α for a descent method would be the minimizer of the function

$$\alpha \mapsto \varphi(\alpha) := f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k).$$

To visualize the function $\varphi(\alpha)$, we think of taking a two-dimensional "slice" of the graph of f in the direction of $\boldsymbol{p}$ (see Figure 6.1). Recall from Lecture 3 or Page 19 in the Preliminaries document that the derivative of $\varphi(\alpha)$ at $\alpha = 0$ is the same as the directional derivative of f in the direction of $\boldsymbol{p}_k$,

$$\varphi'(0) = \frac{\mathrm{d} f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)}{\mathrm{d}\alpha}|_{\alpha=0} = \langle \boldsymbol{p}_k, \nabla f(\boldsymbol{x}_k) \rangle,$$

so that $\varphi'(0) < 0$ is equivalent to having a descent direction.

In practice, minimizing this function is not always the most efficient thing to do (or even possible). One would rather choose a step length that satisfies some criteria that ensure that the sequence $\boldsymbol{x}_k$ converges to a minimizer $\boldsymbol{x}^*$ under suitable conditions on f. One such condition is a sufficient decrease condition,

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) \leq f(\boldsymbol{x}_k) + c\alpha \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{p}_k \rangle =: \ell(\alpha), \tag{6.2}$$

with $c \in (0, 1)$. Note that $\varphi'(0) = \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{p}_k \rangle < 0$, because $\boldsymbol{p}_k$ is a descent direction. The function $\ell(\alpha)$ is therefore a line through $f(\boldsymbol{x}_k)$ with a slope $c\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{p}_k \rangle = c\varphi'(0) > \varphi'(0)$ (remember that $\varphi'(0) < 0$, therefore multiplying with $c \in (0, 1)$ increases this value!). To see why this condition is necessary, consider the function $f(x) = x^2 - 1$ and the sequence $x_k = \sqrt{1 + 1/k}$ for $k \geq 1$. Clearly, the sequence $f(\boldsymbol{x}_k) = 1/k$ decreases, but fails to converge to the minimizer $f(0) = -1$.

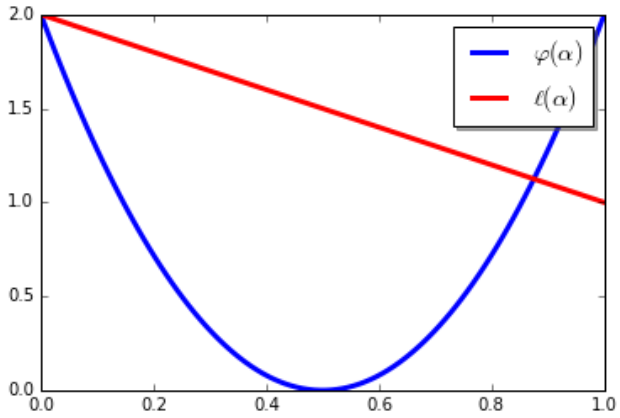


Figure 6.1: The sufficient decrease condition

The sufficient decrease condition (also called Armijo condition) can always be satisfied if α is chosen small enough, but the algorithm may become very slow. It is therefore common to supplement the sufficient descent condition with other criteria that guarantee that sufficient progress is made. Two of the commonly used criteria are:

- the Wolfe conditions, which add a curvature condition

  $$\varphi'(\alpha) \geq \tilde{c}\varphi'(0)$$

  for some $\tilde{c} \in (c, 1)$, which gives a lower bound on the slope of the new point;
- the Armijo-Goldstein conditions, which state that a step length α_k should satisfy the bound

  $$f(\boldsymbol{x}_k) + (1 - c)\alpha_k \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{p}_k \rangle \leq f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k), \tag{6.3}$$

  in addition to (6.2), which gives a lower bound on the step size.

Another common approach is backtracking: in this method one uses a high initial value of α (for example, $\alpha = 1$), and then decreases it until the sufficient descent condition is satisfied.

Example 6.1. Consider the function $f\colon \mathbb{R}^2 \to \mathbb{R}$ defined by $f(\boldsymbol{x}) = x_1^2 + x_2^2$. The gradient is $\nabla f(\boldsymbol{x}) = 2\boldsymbol{x}$, and the φ function at $\boldsymbol{x}_k = (1, 1)^\intercal$

$$\varphi(\alpha) = f(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)) = 2(1 - 2\alpha)^2, \quad \varphi'(\alpha) = -8(1 - 2\alpha).$$

The optimal step length in this case would be $\alpha = 0.5$, found by setting $\varphi'(\alpha) = 0$ and noting that the second derivative is positive. The Armijo-Goldstein conditions (6.2) and (6.3) then state that we can choose α such that

$$\varphi(0) + (1-c)\alpha\varphi'(0) \leq f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) \leq \varphi(0) + c\alpha\varphi'(0)$$

for some $c \in (0,1)$, where we used that $f(\boldsymbol{x}_k) = \varphi(0)$ and $\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{p}_k \rangle = \varphi'(0)$. In our case, this translates to the condition

$$2 - 8(1-c)\alpha \leq 2(1-2\alpha)^2 \leq 2 - 8c\alpha.$$

for some $c \in (0,1)$. If we choose $c = 0.5$, then the left and right bounds are the same, namely $2 - 4\alpha$. In this case, there is only one choice of α, namely the solution to the equation $2(1-2\alpha)^2 = 2(1-2\alpha)$. This implies $\alpha = 0$ (no movement) or $\alpha = 0.5$, so that the choice $c = 0.5$ leads to the optimal step length $\alpha = 0.5$. Other choices of c may lead to other, suboptimal step lengths α.

Convergence of iterative methods

To quantify the convergence of iterative methods, we need a way to measure distances between vectors. Distances between vectors $\boldsymbol{x}, \boldsymbol{y}$ are measured by a norm of the difference $\boldsymbol{x} - \boldsymbol{y}$, $\|\boldsymbol{x} - \boldsymbol{y}\|$ (see Page 8 in the Preliminaries document for a discussion of norms). A sequence of vectors $\{\boldsymbol{x}_k\}$ in $\mathbb{R}^n$, $k \geq 0$, converges to a vector $\boldsymbol{x}^*$ with respect to a norm $\|\cdot\|$ as $k \to \infty$, written $\boldsymbol{x}_k \to \boldsymbol{x}$, if the sequence of numbers $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$ converges to zero. More formally, if for every $\varepsilon > 0$ there exists an index N such that for all $n \geq N$,

$$\|\boldsymbol{x}_n - \boldsymbol{x}^*\| < \varepsilon.$$

Iterative algorithms will rarely find the exact solution to a problem like (6.1), so we will usually be happy to find a solution that differs from the true one by at most some specified accuracy. In fact, computers are not capable to tell very small numbers (like 2^{-53} in double precision arithmetic) from 0, so finding a numerically exact solution is in general not necessary.

Definition 6.2. Assume that a sequence of vectors $\{\boldsymbol{x}_k\}$, $k \geq 0$, converges to $\boldsymbol{x}^*$. Then the sequence $\{\boldsymbol{x}_k\}$, $k \geq 0$, is said to converge

(a) linearly (or Q-linear, Q for quotient), if there exist an $r \in (0,1)$ such that for sufficiently large k,

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq r\|\boldsymbol{x}_k - \boldsymbol{x}^*\|.$$

(b) superlinearly, if

$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} = 0,$$

(c) with order p, if there exists a constant $M > 0$, such that for sufficiently large k,

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq M\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^p.$$

The case $p = 2$ is called quadratic convergence.

Of course, as mentioned earlier, these definitions depend on the choice of a norm. It can be shown that quadratic convergence implies superlinear convergence, and superlinear convergence implies linear convergence.

Example 6.3. Consider the sequence of numbers $x_k = 1/2^{r^k}$ for some $r > 1$. Clearly, $x_k \to x^* = 0$ as $k \to \infty$. Moreover,

$$x_{k+1} = \frac{1}{2^{r^{k+1}}} = \frac{1}{2^{r^k r}} = \left(\frac{1}{2^{r^k}}\right)^r = x_k^r,$$

which shows that the sequence has rate of convergence r.

Convergence of gradient descent

In this section, a norm $\|\cdot\|$ will refere to the 2-norm, unless otherwise stated. We now study the convergence of gradient descent for the least squares problem

$$\text{minimize } f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2, \tag{6.4}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$ of full rank. As we have seen in Lecture 3, the gradient descent method is the procedure

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

where the step length and the residual are given by

$$\alpha_k = \frac{\|\boldsymbol{r}_k\|^2}{\|\boldsymbol{A}\boldsymbol{r}_k\|^2}, \quad \boldsymbol{r}_k = \boldsymbol{A}^\intercal(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k) = -\nabla f(\boldsymbol{x}_k).$$

At the minimizer, the residual is

$$\boldsymbol{r} = -\nabla f(\boldsymbol{x}^*) = \boldsymbol{A}^\intercal(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*) = 0, \tag{6.5}$$

and as the sequence $\boldsymbol{x}_k$ converges to $\boldsymbol{x}^*$, the norms of the residuals converge to 0. Conversely, the residual is related to the difference $\boldsymbol{x}_k - \boldsymbol{x}^*$ by

$$\boldsymbol{r}_k = \boldsymbol{A}^\intercal(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k) = \boldsymbol{A}^\intercal(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k - (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*)) = \boldsymbol{A}^\intercal\boldsymbol{A}(\boldsymbol{x_k} - \boldsymbol{x}^*), \tag{6.6}$$

where we used the "intelligent zero" (6.5). Therefore

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\| = \|(\boldsymbol{A}^\intercal\boldsymbol{A})^{-1}\boldsymbol{r}\| \leq \|(\boldsymbol{A}^\intercal\boldsymbol{A})^{-1}\|\|\boldsymbol{r}_k\|,$$

where $\|\boldsymbol{B}\| = \max_{\boldsymbol{x}\neq\boldsymbol{0}} \|\boldsymbol{B}\boldsymbol{x}\|/\|\boldsymbol{x}\|$ is the operator norm of a matrix $\boldsymbol{B}$ with respect to the 2-norm. Consequently, if the sequence $\|\boldsymbol{r}_k\|$ converges to zero, so does the sequence $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$. A reasonable criterium is to stop the algorithm is therefore when the residual norm $\|\boldsymbol{r}_k\|$ is below a predefined tolerance ε.

The following theorem (whose proof we omit) shows that the gradient descent method for linear least squares converges linearly with respect to the $\boldsymbol{A}$ norm. The statement involves the condition number of $\boldsymbol{A}$[1]. This quantity is defined as

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^\dagger\|,$$

where $\boldsymbol{A}^\dagger$ is the pseudoinverse of $\boldsymbol{A}$. If $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m > n$ and linearly independent columns, it is defined as $\boldsymbol{A}^\dagger = (\boldsymbol{A}^\intercal\boldsymbol{A})^{-1}\boldsymbol{A}^\intercal$.

Theorem 6.4. The error in the $k+1$-th iterate is bounded by

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq \left(\frac{\kappa^2(\boldsymbol{A}) - 1}{\kappa^2(\boldsymbol{A}) + 1}\right)\|\boldsymbol{x}_k - \boldsymbol{x}^*\|.$$

In particular, the gradient descent algorithm converges linearly.

[1]The concept of condition number, introduced by Alan Turing while in Manchester, is one of the most important ideas in numerical analysis, as it is indispensable in studying the performance of numerical algorithms.

7

Newton's method

We saw that gradient descent, applied to the linear least squares problem, has linear convergence. In this lecture we introduce Newton's method, an algorithm that takes advantage of the second derivative and has quadratic convergence under certain circumstances. Throughout this lecture, $\|\cdot\|$ will refer to the 2-norm $\|\cdot\|_2$.

Newton's Method

Let $f \in C^2(\mathbb{R}^n)$ and let's look again at the unconstrained problem

$$\text{minimize } f(\boldsymbol{x}).$$

Newton's method starts with a guess $\boldsymbol{x}_0$ and then proceeds to compute a sequence of points $\{\boldsymbol{x}_k\}_{k\geq 0}$ in $\mathbb{R}^n$ by the rule

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \nabla^2 f(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k), \quad k \geq 0. \tag{7.1}$$

The algorithm stops when $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \varepsilon$ for some predefined tolerance $\varepsilon > 0$. In the context of the general scheme $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$, the step length is $\alpha_k = 1$, and the search direction is the inverse of the Hessian multiplied with the negative gradient.

Recall that the inner product $\langle \boldsymbol{p}, \nabla f(\boldsymbol{x})\rangle$ is the directional derivative of f, and that a descent direction is a direction in which the rate of change (slope) is negative. The following gives a criterion for the search direction in Newton's method to be a descent direction.

Lemma 7.1. Let $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ be an invertible, positive definite symmetric matrix and $f \in C^1(\boldsymbol{R}^n)$. Then $\boldsymbol{p} = -\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x})$ is a descent direction of f at $\boldsymbol{x}$.

Proof. If $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, then $\boldsymbol{B}^{-1}$ is also positive definite, since for all $\boldsymbol{v} \in \mathbb{R}^n$,

$$\boldsymbol{v}^\intercal \boldsymbol{B}^{-1}\boldsymbol{v} = (\boldsymbol{B}\boldsymbol{B}^{-1}\boldsymbol{v})^\intercal \boldsymbol{B}^{-1}\boldsymbol{v} = (\boldsymbol{B}^{-1}\boldsymbol{v})^\intercal \boldsymbol{B}^\intercal (\boldsymbol{B}^{-1}\boldsymbol{v}) = (\boldsymbol{B}^{-1}\boldsymbol{v})^\intercal \boldsymbol{B}(\boldsymbol{B}^{-1}\boldsymbol{v}) > 0.$$

(This can also be seen by noting that the eigenvalues of $\boldsymbol{B}^{-1}$ are the inverses of the eigenvalues of $\boldsymbol{B}$.) For $\boldsymbol{p} = -\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x})$ we then get

$$\langle \boldsymbol{p}, \nabla f(\boldsymbol{x})\rangle = -\langle \boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}), \nabla f(\boldsymbol{x})\rangle = -\nabla f(\boldsymbol{x})^\intercal \boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}) < 0,$$

which shows that $\boldsymbol{p}$ is a descent direction. □

To better understand Newton's method, we first look at the one dimensional case.

Example 7.2. Let $f \in C^2(\mathbb{R})$. In this case Newton's method is described as

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}, \quad k \geq 0. \tag{7.2}$$

Newton's method looks for a local minimizer in the form of a point x^* such that $f'(x^*) = 0$ and $f''(x^*) > 0$. Setting $g(x) := f'(x)$, we are looking for a root x^*,

$$g(x^*) = 0.$$

One approach to find such a root is to approximate the function $g(x)$ at a point x_k by its tangent line,

$$g(x) \approx g(x_k) + g'(x_k)(x - x_k),$$

and then identify the next iterate x_{k+1} as the root of this linear approximation:

$$g(x_k) + g'(x_k)(x_{k+1} - x_k) = 0 \iff x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}$$

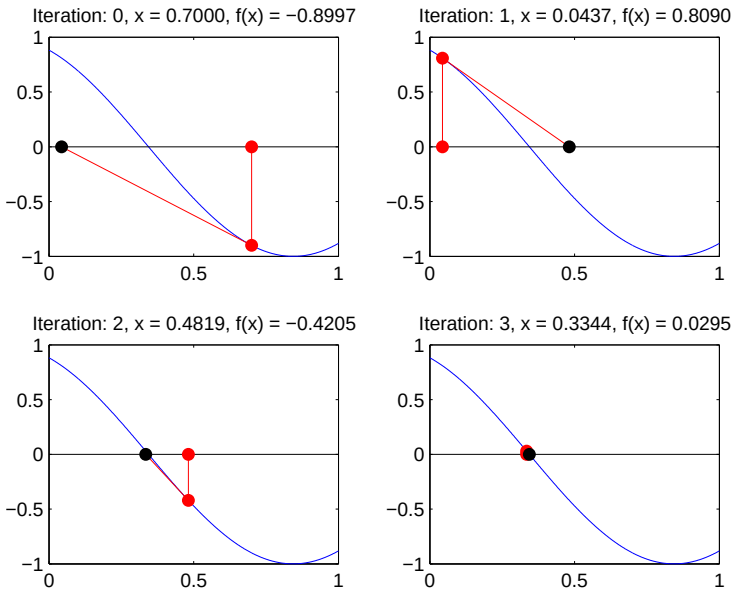


Figure 7.1: Newton's method

Geometrically this corresponds to taking the tangent to g at x_k and setting x_{k+1} to be the intersection of this tangent line with the x-axis, as shown in Figure 7.1. Replacing $g(x) = f'(x)$ gives precisely Newton's method (7.2).

Another way to understand Newton's method is to view it in contrast with gradient descent. While gradient descent corresponds to working with a linear approximation

$$f(\boldsymbol{x}_{k+1}) \approx f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle,$$

Newton's method is based on the quadratic approximation,

$$f(\boldsymbol{x}_{k+1}) \approx f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{1}{2} \langle \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle.$$

Example 7.3. Consider the function f on $\mathbb{R}^2$,

$$f(\boldsymbol{x}) = \frac{1}{2}(x_1^2 + 10x_2^2).$$

Starting with $\boldsymbol{x}_0 = (10, 1)^\intercal$, gradient descent takes 84 iterations to reach accuracy 10^{-6}, while Newton's method, unsurpringly, takes only one.

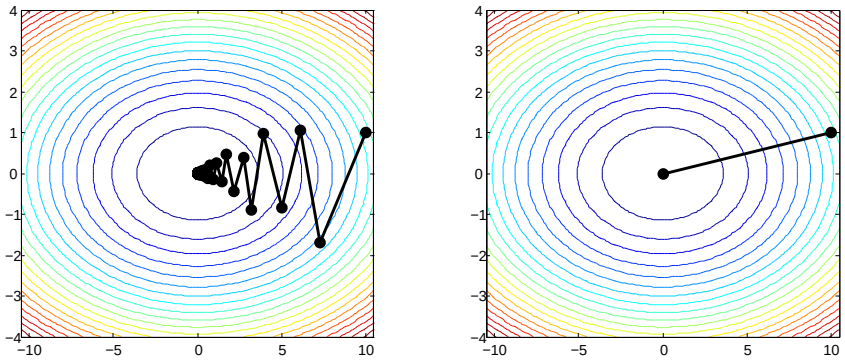


Figure 7.2: Gradient descent vs. Newton's method on a quadratic function.

In practice, when implementing Newton's method one does not explicitly compute the inverse of the Hessian. The reason is that one does not need the inverse itself, but only the product $\nabla^2 f(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k)$, which is the solution of a system of equations $\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{y} = \nabla f(\boldsymbol{x}_k)$ that can be solved much more efficiently. One therefore replaces the update step (7.1) with the following two steps:

$$\begin{aligned} \text{Find } \boldsymbol{y} \text{ such that} \quad & \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{y} = -\nabla f(\boldsymbol{x}_k), \\ & \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{y}. \end{aligned}$$

There is a lot that can go wrong with Newton's method. In particular, the matrix $\nabla^2 f(\boldsymbol{x})$ has to be non-singular, or invertible, at every step. If, however, we start at a point $\boldsymbol{x}_0$ that is not too far from a local minimizer $\boldsymbol{x}^*$ with $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\nabla^2 f(\boldsymbol{x}^*)$ positive definite, then we are on the safe side.

Lemma 7.4. Let $\boldsymbol{x}^* \in \mathbb{R}^n$ be such that $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then there exists an open neighbourhood U of $\boldsymbol{x}^*$ such that for all $\boldsymbol{x} \in U$, $\nabla^2 f(\boldsymbol{x})$ is positive definite.

For the main result of this lecture, namely the quadratic convergence of Newton's method, we make the additional assumption that the Hessian $\nabla^2 f(\boldsymbol{x})$ is Lipschitz continuous as a function of $\boldsymbol{x}$.

Definition 7.5. A function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz continuous on a domain $\Omega \subseteq \mathbb{R}^n$ with respect to a pair of norms on $\mathbb{R}^n$ and $\mathbb{R}^m$ if there is a constant $L > 0$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in \Omega$,

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|.$$

The constant L is called the Lipschitz constant of the map.

In particular, the Hessian of a function $f \in C^2(\mathbb{R}^n)$, considered as a map from $\mathbb{R}^n$ to $\mathbb{R}^{n\times n}$, is Lipschitz continuous with respect to a norm on $\mathbb{R}^n$ and the corresponding operator norm on $\mathbb{R}^{n\times n}$, if for any $\boldsymbol{x}, \boldsymbol{y}$ we have

$$\|\nabla^2 f(\boldsymbol{x}) - \nabla^2 f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|.$$

Theorem 7.6. Let $f \in C^2(\mathbb{R}^n)$ with Lipschitz continuous Hessian, and let $\boldsymbol{x}^* \in \mathbb{R}^n$ be a local minimizer with $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}^*) > 0$. Then for $\boldsymbol{x}_0$ sufficiently close to $\boldsymbol{x}^*$, Newton's method has quadratic convergence.

Proof. (Optional) Assume that $\nabla^2 f(\boldsymbol{x}_k)$ is positive definite. Consider the difference

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| &= \|\boldsymbol{x}_k - \boldsymbol{x}^* - \nabla^2 f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)\| \\ &\overset{(1)}{=} \|\boldsymbol{x}_k - \boldsymbol{x}^* - \nabla^2 f(\boldsymbol{x}_k)^{-1} (\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}^*))\| \\ &= \|\nabla^2 f(\boldsymbol{x}_k)^{-1} (\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}^*) - (\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}^*))\| \end{aligned} \tag{7.3}$$

where (1) follows from $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$. The Fundamental Theorem of Calculus tells us

$$\begin{aligned} \nabla f(\boldsymbol{x}^*) - \nabla f(\boldsymbol{x}_k) &= \int_0^1 \frac{\mathrm{d}}{\mathrm{d}t} \nabla f(\boldsymbol{x}_k + t(\boldsymbol{x}^* - \boldsymbol{x}_k)) \, \mathrm{d}t \\ &= \int_0^1 \nabla^2 f(\boldsymbol{x}_k + t(\boldsymbol{x}^* - \boldsymbol{x}_k))(\boldsymbol{x}^* - \boldsymbol{x}_k) \, \mathrm{d}t. \end{aligned}$$

Continuing from (7.3), by inserting this identity,

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| &= \left\| \nabla^2 f(\boldsymbol{x}_k)^{-1} \int_0^1 \left[\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}_k + t(\boldsymbol{x}^* - \boldsymbol{x}_k)) \right] (\boldsymbol{x}_k - \boldsymbol{x}^*) \, \mathrm{d}t \right\| \\ &\leq \|\nabla^2 f(\boldsymbol{x}_k)^{-1}\| \cdot \\ &\quad \int_0^1 \|\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}_k + t(\boldsymbol{x}^* - \boldsymbol{x}_k))\| \, \mathrm{d}t \cdot \|(\boldsymbol{x}_k - \boldsymbol{x}^*)\|. \end{aligned}$$

Applying the Lipschitz bound to the term inside the integral gives

$$\|\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}_k + t(\boldsymbol{x}^* - \boldsymbol{x}_k))\| \leq Lt \|\boldsymbol{x}_k - \boldsymbol{x}^*\|.$$

Integrating this out, we end up with the bound

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq \frac{L}{2} \|\nabla^2 f(\boldsymbol{x}_k)^{-1}\| \cdot \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2.$$

The only remaining issue is that the "constant" on the right-hand side is not a constant. However, the Lipschitz continuity implies that $\nabla^2 f(\boldsymbol{x}_k)$ converges to $\nabla^2 f(\boldsymbol{x}^*)$ if $\boldsymbol{x}_k$ converges to $\boldsymbol{x}^*$. Since the inversion of a matrix is a continuous operation, also $\nabla^2 f(\boldsymbol{x}_k)^{-1}$ converges to $\nabla^2 f(\boldsymbol{x}^*)^{-1}$. In particular, if $\boldsymbol{x}_k$ is sufficiently close to $\boldsymbol{x}^*$, we have $\|\nabla^2 f(\boldsymbol{x}_k)^{-1}\| \leq 2\|\nabla^2 f(\boldsymbol{x}^*)^{-1}\|$ for k sufficiently large. Setting $M := L\|\nabla^2 f(\boldsymbol{x}^*)^{-1}\|$, we end up with the bound

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leq M \cdot \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2.$$

By Lemma 7.4 there exists an open neighbourhood around $\boldsymbol{x}^*$ in which $\nabla^2 f(\boldsymbol{x})$ is positive definite, and within this neighbourhood there is an $\boldsymbol{x}_0$ such that $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 < 1/M$, which ensures that all following iterates remain in U. This shows quadratic convergence. □

Note that the conditions for quadratic convergence in an open neighbourhood U of $\boldsymbol{x}^*$ are precisely that f is convex on U.

Quasinewton methods

One drawback of Newton's method is that it requires the computation of the Hessian matrix, which can be expensive. Quasinewton methods use an approximation of the Hessian, $\boldsymbol{B}_k \approx \nabla f(\boldsymbol{x}_k)$, at each step of the algorithm. These are constructed in a way that $\boldsymbol{B}_{k+1}$ can easily be computed from $\boldsymbol{B}_k$. A popular method, that is used often in practical applications because of its efficiency, is the Broyden-Fletcher-Shanno-Goldfarb (BFGS) method. The BFGS method may be described as follows.

- Start with $\boldsymbol{x}_0$, $\boldsymbol{B}_0$.
- For $k \geq 0$, compute

$$\begin{aligned}
\boldsymbol{p}_k &= \boldsymbol{B}_k^{-1} \nabla f(\boldsymbol{x}_k) \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \text{ for a suitable step length } \alpha_k \\
\boldsymbol{s}_k &= \alpha_k \boldsymbol{p}_k \\
\boldsymbol{y}_k &= \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k) \\
\boldsymbol{B}_{k+1} &= \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^\intercal}{\boldsymbol{y}_k^\intercal \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^\intercal \boldsymbol{B}_k}{\boldsymbol{s}_k^\intercal \boldsymbol{B}_k \boldsymbol{s}_k}.
\end{aligned}$$

- Stop if $\|\nabla f(\boldsymbol{x}_k)\| < \varepsilon$ for some tolerance ε, or if $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \varepsilon$.

8

Machine Learning

In this lecture we will embark on one of the most important modern applications of convex optimization: machine learning. The goal of machine learning is to develop methods to automatically learn a function

$$h\colon \mathcal{X} \to \mathcal{Y},$$

where $\mathcal{X}$ is a space of inputs or features, and $\mathcal{Y}$ consists of outputs or responses. The input space $\mathcal{X}$ is usually an $\mathbb{R}^n$, though the inputs could represent anything such as images, texts, emails, genome sequences, or networks, financial time series, or demographic data. The output could be either quantitative values, such as a temperature or the amount of some substance in the body, or qualitative or categorical, such as YES, NO or $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (for recognising spam or handwritten digits, respectively). The first type of problem is usually called regression, while the latter is called classification. Machine learning techniques underlie much of modern technology, from car electronics to online product recommendation systems and search engines.

Supervised Learning

In supervised learning, we have at our disposal a collection of input-output pairs

$$(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y},\ 1 \leq i \leq m,$$

and the goal is to learn a function $h\colon \mathcal{X} \to \mathcal{Y}$ from this data. The given pairs $\{(\boldsymbol{x}_i, y_i)\}_{1 \leq i \leq m}$ are called the training set.

Example 8.1. (Handwriting recognition) Given a dataset of pixel matrices, each representing a grey-scale image, with associated labels telling us for each image the number it represents, the task is to use this to train a computer program to recognise new numbers. Such classification tasks are often carried out using deep neural networks.

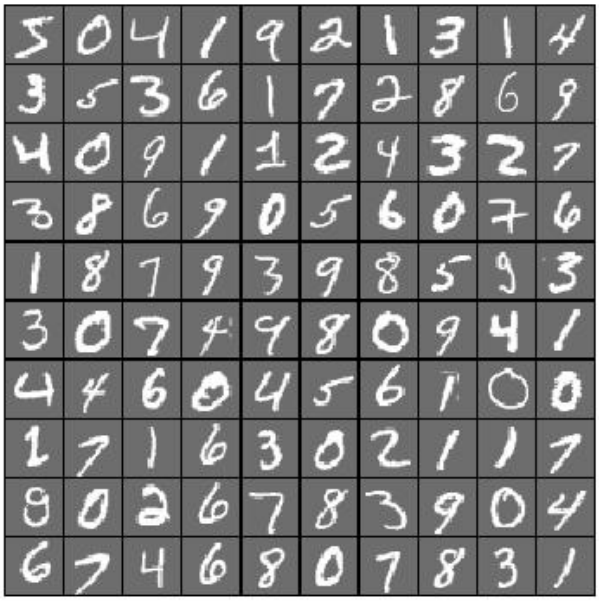
Example 8.2. (Linear regression) Recall from Lecture 1 the problem of finding a function of the form

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

Given a set of input-output pairs $(\boldsymbol{x}_i, y_i)$, arranged in a matrix $\boldsymbol{X}$ and a vector $\boldsymbol{y}$, we saw that we could guess the correct $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\intercal$ by solving the least-squares optimization problem

$$\underset{\boldsymbol{\beta}}{\text{minimize}}\ \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2.$$

By now, we know how to solve this problem using gradient descent.



Example 8.3. In text classification, the task is to decide to which of a given set of categories a given text belongs. The training data consists of a bag of words: this is a large sparse matrix, whose columns represent words and the rows represent articles, with the (i, j)-th entry containing the number of times word j is contained in text i.

	Rooney	Boris
Article 1	5	0
Article 2	1	7

For example, in the above set we would classify the first article as "Sports" and the second one as "Politics". One such training dataset is the Reuters Corpus Volume I (RCV1), an archive of over 800,000 categorised newswire stories.

A typical binary classifier for such a problem would be a linear classifier of the form

$$h(\boldsymbol{x}) = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - \tau,$$

with $\boldsymbol{w} \in \mathbb{R}^n$ and $\tau \in \mathbb{R}$. Given a text, represented as a row of the dataset $\boldsymbol{x}$, it is classified into one of two classes $\{+1, -1\}$, depending on whether $h(\boldsymbol{x}) > 0$ or $h(\boldsymbol{x}) < 0$.

Suppose we have the training data $\{\boldsymbol{x}_i, y_i\}_{1 \le i \le m}$ and we found a candidate function $h\colon \mathcal{X} \to \mathcal{Y}$. How do we assess if this is a good fit? One usually assigns to the problem a loss function $\ell\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ that measures the mismatch of a prediction. One would then aim to find a function h among a set of candidates that minimizes the loss when applying the function to the training data:

$$\underset{h}{\text{minimize}}\ \frac{1}{m}\sum_{i=1}^{m} \ell(h(\boldsymbol{x}_i, y_i)).$$

The form of the loss function depends on the problem at hand, but two typical candidates are the square error for regression problems,

$$\ell(h(\boldsymbol{x}), y) = (h(\boldsymbol{x}) - y)^2,$$

and the indicator loss function

$$\mathbf{1}\{h(\boldsymbol{x}) \neq y\},$$

where $\mathbf{1}\{A\}$ is the indication function, which takes the value 1 if A is true, and 0 else. As this function is not continuous, in practice one often encounters the log loss function,

$$\ell(h(\boldsymbol{x}), y) = \log\left(1 + e^{-h(\boldsymbol{x})y}\right).$$

Note that if $h(\boldsymbol{x})$ and $\boldsymbol{y}$ have the same sign (corresponding to a match), then the value of this function will be close to zero.

Suppose we have a binary classification task at hand, with $\mathcal{Y} = \{-1, 1\}$. We could learn the following function from our data:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i, \\ 1 & \text{otherwise.} \end{cases}$$

The empirical misclassification risk in for this problem is then 0,

$$R(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\{h(\boldsymbol{x}_i \neq y_i)\} = 0.$$

Nevertheless, this is not a good classifier: it will not perform very well outside of the training set. This is an example of overfitting: when the function is adapted too closely to the seen data. To remedy this, one often (randomly) splits the available data into a training set and a test set. The training set is used to find the function h, while the test set is for testing how good it is (in practise, one often adds a validation set, used to tune some parameters of the problem).

Example 8.4. In the linear regression problem with the quadratic loss function, we end up with the minimization problem

$$\underset{\boldsymbol{\beta}}{\text{minimize}}\, \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^\intercal\boldsymbol{\beta} - y_i)^2 = \frac{1}{n}\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2.$$

This is precisely the problem encountered in Lecture 1.

Example 8.5. In the example of classifying a text into two classes using a linear classifier, we can look at the problem

$$\underset{\boldsymbol{w},\tau}{\text{minimize}}\, \frac{1}{n}\sum_{i=1}^{m} \mathbf{1}\{\boldsymbol{w}^\intercal\boldsymbol{x}_i - \tau \neq y_i\}.$$

Since this function is not smooth or even continuous, we can work with the log-loss function described above,

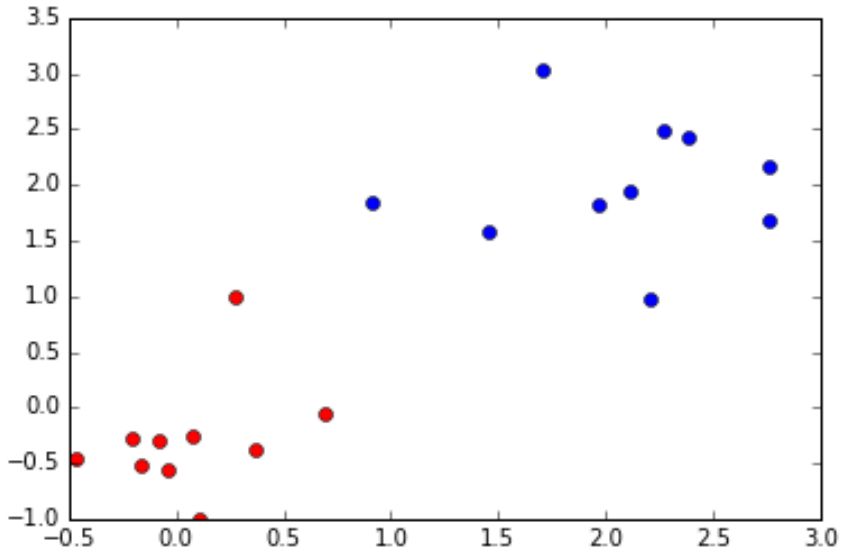
$$\underset{\boldsymbol{w},\tau}{\text{minimize}}\, \frac{1}{n}\sum_{i=1}^{m} \ell(\boldsymbol{w}^\intercal\boldsymbol{x}_i - \tau, y_i).$$

This function can, in principle, be minimized using gradient descent or Newton's method.

In the following example, we use this classifier to separate an artificial data set of 20 points. 10 points are generated as normal (Gaussian) distributed vectors in $\mathbb{R}^2$, with mean $(2, 2)^\intercal$ and variance 0.25, while the other 10 points are generated with mean $(0, 0)^\intercal$ and the same variance.

In [1]:

```
X = np.zeros((2,20))
for i in range(10):
    X[:,i] = np.array([2,2]) + 0.5*rnd.randn(2)
for i in range(10):
    X[:,10+i] = np.array([0,0]) + 0.5*rnd.randn(2)

% matplotlib inline
plt.plot(X[0,0:10], X[1,0:10], 'o')
plt.plot(X[0,10:], X[1,10:], 'o', color='red')
plt.show()
```



We next minimize the log-loss function,

$$F(\boldsymbol{w}) = \sum_{i=1}^{20} \log(1 + e^{-y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i - \tau)}),$$

where the $\boldsymbol{x}_i \in \mathbb{R}^2$ are the points, and the $y_i \in \{-1, 1\}$ classify these as either red or blue. As minimization algorithm we choose gradient descent with backtracking, though any other reasonable algorithm would do.

In [2]:

```
def graddesc_bt(f, df, x0, tol, maxiter=100, rho=0.5, c=0.1):
    """
    Gradient descent with backtracking
    """
    x = np.vstack((x0+2*tol*np.ones(x0.shape),x0)).transpose()
    i = 1
    while ( la.norm(x[:,i]-x[:,i-1]) > tol ) and ( i < maxiter ):
        p = -df(x[:,i])
        # Start backtracking
        alpha = 1
        xnew = x[:,i] + alpha*p
        while (f(xnew) >= f(x[:,i]) + alpha*c*np.dot(p, df(x[:,i]))):
            alpha = alpha*rho
            xnew = x[:,i] + alpha*p
        x = np.concatenate((x,xnew.reshape((len(x0),1))), axis=1)
        i += 1
    return x[:,1:]
```

In [3]:

```
y = np.concatenate((-np.ones(10),np.ones(10)))
X = np.concatenate((X, np.ones((1,20))), axis=0)

def f(w):
    return np.sum(np.log(1+np.exp(-y*(np.dot(w,X)))))

def df(w):
    return np.array([-np.sum(y*X[0,:]*np.exp(-y*np.dot(w,X))/
                             (1+np.exp(-y*np.dot(w,X)))),
                     -np.sum(y*X[1,:]*np.exp(-y*np.dot(w,X))/
                     (1+np.exp(-y*np.dot(w,X)))),
                     -np.sum(y*X[2,:]*np.exp(-y*np.dot(w,X))/
                     (1+np.exp(-y*np.dot(w,X))))])

W = graddesc_bt(f, df, np.array([1.,1.,1.]), 1.e-8)
```

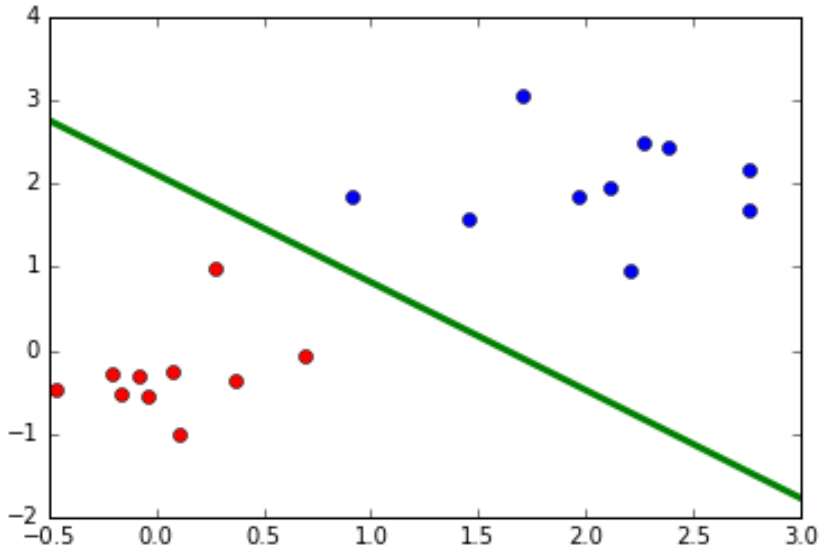
The algorithm gives as output a vector $\boldsymbol{w} = (w_1, w_2, w_3) \in \mathbb{R}^3$, so that the line separating the two sets of points is given by

$$w_1 x_1 + w_2 x_2 + w_3 = 0.$$

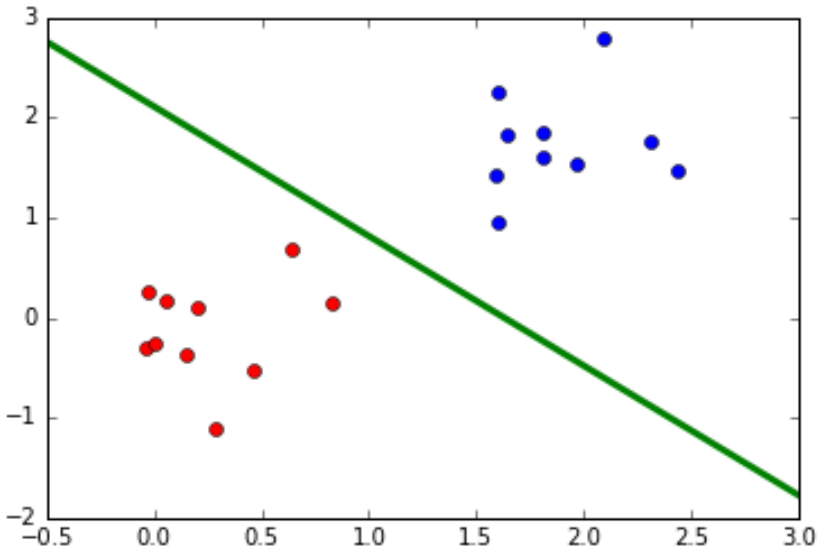
Rearranging this as a function $\ell(x_1) = x_2 = -(w_1 x_1 + w_3)/w_2$ and plotting the line into the pointcloud, we get the following graph.

In [4]:

```
def l(x):
    return (-W[0,-1]*x-W[2,-1])/W[1,-1]

xx = np.linspace(-0.5,3,100)
yy = l(xx)
plt.plot(X[0,0:10], X[1,0:10], 'o')
plt.plot(X[0,10:], X[1,10:], 'o', color='red')
plt.plot(xx,yy,linewidth=3)
plt.show()
```



Of course, we would like to know how our classifier does with a test set generated according to the same distribution. The result is seen the following plot.



The given example is unproblematic, since the two sets of points are generated in a way that makes it unlikely that they can't be separated using a linear classifier. With real data, the situation may not be as favourable.

It is also interesting to see how fast the objective function (the empirical risk) approaches the minimum. While the value of the objective function decreases very quickly at the beginning, the decrease slows down dramatically and the algorithm does not reach the desired accuracy within the limit we set for the number of iterations. Now, however, that for the classification task we are interested, we don't need the best possible precision! In fact, the noise in the data makes too much precision meaningless, and we can safely use the result obtained after 100 iterations. Other algorithms and approaches may perform better for this problem.

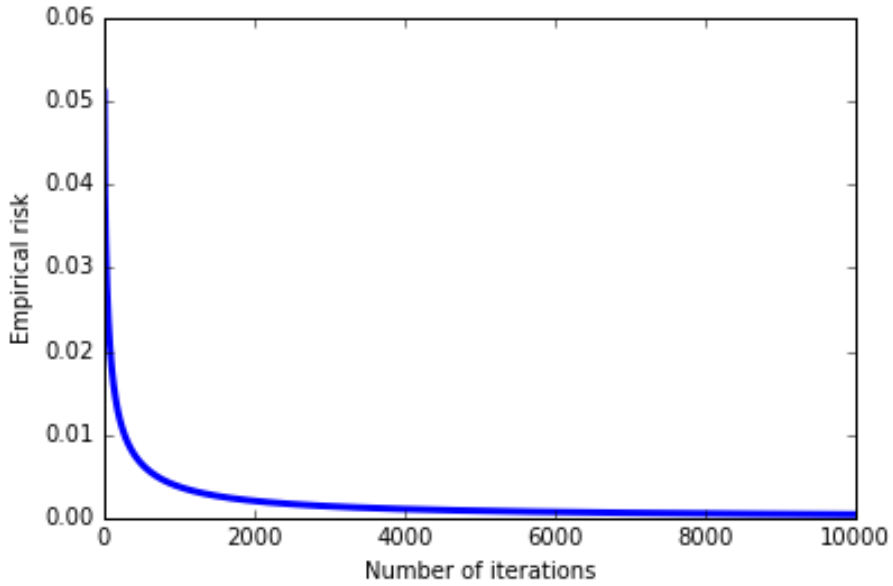
In [5]:
```
risk = np.zeros(W.shape[1])
for i in range(W.shape[1]):
    risk[i] = f(W[:,i])

plt.plot(range(W.shape[1]), risk, linewidth=3)
plt.xlabel('Number of iterations')
plt.ylabel('Empirical risk')
plt.show()
```

We have seen how a machine learning problem can be turned into an optimization problem. Unfortunately, when the amount of data is large, computing gradients and Hessians of the resulting functions can be very expensive. Suppose the function h depends on parameters $\boldsymbol{w}$ which we want to optimize over, $h = h(\boldsymbol{x}; \boldsymbol{w})$, and denote the composition of the loss function ℓ with h on the data $\boldsymbol{x}_i$ by

$$f_i(\boldsymbol{w}) := \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), y_i).$$

We can then write our optimization problem as

$$\underset{\boldsymbol{w}}{\text{minimize}}\, F(\boldsymbol{w}), \quad F(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} f_i(\boldsymbol{w}).$$



The gradient is then

$$\nabla F(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{w}).$$

For large datasets (for example, $m > 10^6$ is not uncommon), computing all the gradients in each step is highly ineffective.

An old algorithm that has recently been revived in light of big data is stochastic gradient descent. The idea is to compute, at each step, only the gradient of one (or sometimes a few) of the summands of the loss function:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k),$$

where the index i_k is chosen at random from $\{1, \ldots, m\}$ at each step.

Each step of this algorithm is clearly faster than gradient descent, but does it even work? It turns out that this algorithm converges quickly and that it is surprisingly effective on large datasets.

9

Linear Programming 1

Linear programming is about problems of the form

$$\begin{aligned} \text{maximize} \quad & \langle \boldsymbol{c}, \boldsymbol{x} \rangle \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, and $\boldsymbol{b} \in \mathbb{R}^m$, and the inequality sign means inequality in each row. The feasible set is the set of all possible candidates,

$$\mathcal{F} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}.$$

This set can be empty (example: $x \leq 1,\ -x \leq -2$), unbounded (example: $x \leq 1$) or bounded (example: $x \leq 1, -x \leq 0$). In any case, it is a convex set. The set of feasible solutions is the set over which we will search for solutions, and it is therefore important to understand the geometry of this set. The set of the feasible sets of linear programming is called a polyhedron.

Linear Programming Duality: a first glance

Suppose we are faced with a linear programming problem. A first important task is to find out if there exist any feasible solutions at all. That is, we would like to know if the feasible set $\mathcal{F}$ is empty or not, i.e., if $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ has a solution. If it is not empty, we can certify that by producing a vector from $\mathcal{F}$. To verify that the feasible set is empty is more tricky: we are asked to show that no vector lives in $\mathcal{F}$. What we can try to do, however, is to show that the assumption of a solution to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ would lead to a contradiction. Denote by $\boldsymbol{a}_i^\intercal$ the rows of $\boldsymbol{A}$, so that the entries of $\boldsymbol{A}\boldsymbol{x}$ are given by $\boldsymbol{a}_i^\intercal \boldsymbol{x}$ for $1 \leq i \leq m$. Assuming $\boldsymbol{x} \in \mathcal{F}$, then given a vector $\boldsymbol{0} \neq \boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^\intercal$ with $\lambda_i \geq 0$, the linear combination satisfies

$$\langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} \rangle = \sum_{i=1}^m \lambda_i \boldsymbol{a}_i^\intercal \boldsymbol{x} \leq \sum_{i=1}^m \lambda_i b_i = \langle \boldsymbol{\lambda}, \boldsymbol{b} \rangle. \tag{9.1}$$

If we can find parameters $\boldsymbol{\lambda}$ such that the left-hand side of (9.1) is identically 0 and the right-hand side is strictly negative, then we have found a contradition and can conclude that no $\boldsymbol{x}$ satisfies $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$. A condition that ensures this is

$$\sum_{i=1}^m \lambda_i \boldsymbol{a}_i = \boldsymbol{0}, \quad \langle \boldsymbol{\lambda}, \boldsymbol{b} \rangle < 0. \tag{9.2}$$

In matrix form,

$$\exists \boldsymbol{\lambda} \geq \boldsymbol{0},\ \boldsymbol{A}^\intercal \boldsymbol{\lambda} = \boldsymbol{0},\ \langle \boldsymbol{\lambda}, \boldsymbol{b} \rangle < 0.$$

This condition will still be satisfied if we normalise the vector $\boldsymbol{\lambda}$ such that $\sum_{i=1}^{m} \lambda_i = 1$, so the statement says that $\mathbf{0}$ is a convex combination of the vectors defining the equations.

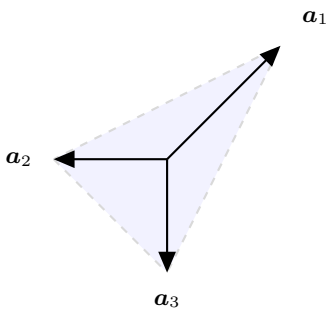
Example 9.1. Consider the system

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ -x_1 &\leq -1 \\ -x_2 &\leq -1.5. \end{aligned} \tag{9.3}$$

The transpose matrix $\boldsymbol{A}^\intercal$ and the vector $\boldsymbol{b}$ are

$$\boldsymbol{A}^\intercal = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 2 \\ -1 \\ -1.5. \end{pmatrix}$$

Drawing the columns of $\boldsymbol{A}^\intercal$ we get



We can get the origin as a convex combination of the vectors $\boldsymbol{a}_i$ (drawing a rope around them encloses the origin), and such a combination is given by $\boldsymbol{\lambda} = (1/3, 1/3, 1/3)$. Taking the scalar product with the vector $\boldsymbol{b}$ we get

$$\langle \boldsymbol{\lambda}, \boldsymbol{b} \rangle = \frac{1}{3}(2 - 1 - 1.5) = -\frac{1}{6} < 0.$$

This shows that the system (9.3) does not have a solution (a fact that in this simple example can also be seen by drawing a picture).

It turns out that Condition (D) is not only sufficient but also necessary, and the separating hyperplane theorem is an essential part of this. We first make a detour in order to better understand the feasible sets, the polyhedra.

Polyhedra

Definition 9.2. A polyhedron (plural: polyhedra) is a set defined as the solution of linear equalities and inequalities,

$$P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}, \tag{9.4}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$.

More classically, we can write out the equations.

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &\leq b_1, \\ &\cdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n &\leq b_m. \end{aligned} \tag{9.5}$$

We now introduce some useful terminology and concepts associated to polyhedra, and illustrate them with a few examples. A supporting hyperplane H of a polyhedron P is a hyperplane such that $P \subseteq H_-$, where H_- is a halfspace associated to H. If H is a supporting hyperplane, then a set of the form $F = H \cap P$ is called a face of P. In particular, the polyhedron P is a face. Each of the inequalities $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leq b_i$ in (9.4) defines a supporting hyperplane, and therefore a face. The dimension of a face F, $\dim F$, is the smallest dimension of an affine space containing F. Faces of dimension $\dim F = \dim P - 1$ are called facets, faces of dimension 0 are vertices, and of dimension 1 edges. A vertex can equivalently be characterised as a point $\boldsymbol{x} \in P$ that can not be written as a convex combination of two other points in P.

Example 9.3. Polyhedra in one dimension are the sets $[a, b]$, $[a, \infty)$, $(-\infty, b]$, $\mathbb{R}$ or $\emptyset$, where $a \leq b$. Each of them is clearly convex.

Example 9.4. The set

$$P = \{\boldsymbol{x} \in \mathbb{R}^2 \mid x_1 + x_2 \leq 1,\ x_1 \geq 0,\ x_2 \geq 0\}.$$

is the polyhedron shown in Figure 9.1. We can write the defining inequalities in standard form $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ by setting

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

This polyhedron has one face of dimension 2 (itself), three facets of dimension 1 (the sides, corresponding

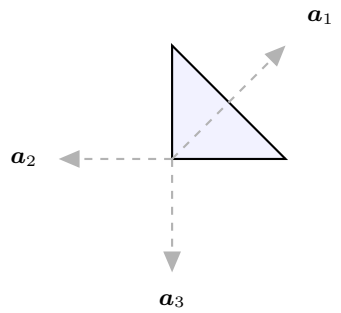


Figure 9.1: A two-dimensional polyhedron and defining equations.

to the three equations), and three vertices of dimension 0 (the corners, corresponding any two of the defining equations).

Example 9.5. The polyhedron in Albrecht Dürer's famous "Melencolia I", aka the "truncated triangular trapezohedron", can be described using eight inequalities, see Figure 9.2.

We now move to a different characterization of bounded polyhedra. The main result of this lecture is that bounded polytopes can be described completely from knowing their vertices. A polyhedron P is called bounded, if there exists a ball $B(\mathbf{0}, r)$ with $r > 0$ such that $P \subset B(\mathbf{0}, r)$. For example, halfspaces are not bounded, but the polytope from Examples 9.4 and 9.5 are.

We first observe the nontrivial fact that a polyhedron has only fin

Definition 9.6. A polytope is the convex hull of finitely many points,

$$P = \operatorname{conv}(\{x_1, \ldots, x_k\}) = \{\sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i,\ \lambda_i \geq 0, \sum_i \lambda_i = 1\}.$$

$$
\begin{aligned}
0.7071x_1 - 0.4082x_2 + 0.3773x_3 &\leq 1\\
-0.7071x_1 + 0.4082x_2 - 0.3773x_3 &\leq 1\\
0.7071x_1 + 0.4082x_2 - 0.3773x_3 &\leq 1\\
-0.7071x_1 - 0.4082x_2 + 0.3773x_3 &\leq 1\\
0.8165x_2 + 0.3773x_3 &\leq 1\\
-0.8165x_2 - 0.3773x_3 &\leq 1\\
0.6313x_3 &\leq 1\\
-0.6313x_3 &\leq 1
\end{aligned}
$$

Figure 9.2: Dürer's Melencolia and equations defining the polyhedron

Theorem 9.7. A bounded polyhedron P is the convex hull of its vertices.

Example 9.8. The triangle in Example 9.4 is the convex hull of the points $(0,0)^\intercal$, $(0,1)^\intercal$, and $(1,0)^\intercal$.

Example 9.9. The Dürer polytope is the convex hull of the following 12 vertices:

$$
\begin{aligned}
\boldsymbol{v}_1 &= \begin{pmatrix} -1.4142 \\ -0.8165 \\ -0.8835 \end{pmatrix}, \boldsymbol{v}_2 = \begin{pmatrix} -1.4142 \\ 0.8165 \\ 0.8835 \end{pmatrix}, \boldsymbol{v}_3 = \begin{pmatrix} -0.8536 \\ -0.4928 \\ -1.5840 \end{pmatrix}, \boldsymbol{v}_4 = \begin{pmatrix} -0.8536 \\ 0.4928 \\ 1.5840 \end{pmatrix},\\
\boldsymbol{v}_5 &= \begin{pmatrix} -0.0000 \\ -1.6330 \\ 0.8835 \end{pmatrix}, \boldsymbol{v}_6 = \begin{pmatrix} 0.0000 \\ -0.9856 \\ 1.5840 \end{pmatrix}, \boldsymbol{v}_7 = \begin{pmatrix} -0.0000 \\ 0.9856 \\ -1.5840 \end{pmatrix}, \boldsymbol{v}_8 = \begin{pmatrix} 0.0000 \\ 1.6330 \\ -0.8835 \end{pmatrix},\\
\boldsymbol{v}_9 &= \begin{pmatrix} 0.8536 \\ -0.4928 \\ -1.5840 \end{pmatrix}, \boldsymbol{v}_{10} = \begin{pmatrix} 0.8536 \\ 0.4928 \\ 1.5840 \end{pmatrix}, \boldsymbol{v}_{11} = \begin{pmatrix} 1.4142 \\ -0.8165 \\ -0.8835 \end{pmatrix}, \boldsymbol{v}_{12} = \begin{pmatrix} 1.4142 \\ 0.8165 \\ 0.8835 \end{pmatrix}.
\end{aligned}
$$

The converse of Theorem 9.7 is also true.

Theorem 9.10. A polytope is a bounded polyhedron.

The equivalence between polytopes and bounded polyhedra gives a first glimpse into linear programming duality theory, a topic of central importance in both modeling and algorithm design.

10

Linear Programming 2

Linear programming duality associates to a linear programming problem a dual problem, with the property that the optimal values of the original and of the dual problem coincide. Duality is an important tool in applications and in the design of algorithms. Linear programming duality rests upon an important family of results in convex geometry, known collectively as Farkas' Lemma.

Farkas' Lemma

Recall the definition of a convex cone. This is a set C such that for all $\boldsymbol{x}, \boldsymbol{y} \in C$ and $\lambda_1, \lambda_2 \geq 0$ we have $\lambda_1 \boldsymbol{x} + \lambda_2 \boldsymbol{y} \in C$.

Lemma 10.1. (Hyperplane separation for cones) Let $C \neq \mathbb{R}^n$ be a closed convex cone and $\boldsymbol{z} \notin C$. Then there exists a linear hyperplane such that $C \subseteq H_-$ and $\boldsymbol{z} \in \text{int}H_+$.

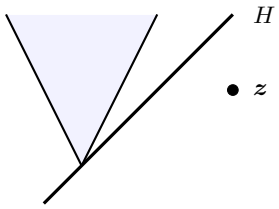


Figure 10.1: Separating hyperplane for a cone

Proof. From Lecture 7 we know that there exists an affine hyperplane H^a separating C and $\boldsymbol{z}$. Let this affine hyperplane be given by $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b$. We would like to show that $H = \{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x} \rangle = 0\}$ is a linear hyperplane separating C and $\boldsymbol{z}$. From $\langle \boldsymbol{a}, \boldsymbol{z} \rangle > b$ we clearly get $\langle \boldsymbol{a}, \boldsymbol{z} \rangle > 0$. Also, $\mathbf{0} \in H_-$, since $\langle \boldsymbol{a}, \mathbf{0} \rangle = 0$. Assume now that there exists a point $\boldsymbol{x} \in C$ such that $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = c > 0$. Since C is a cone, for all $\lambda > 0$ we have that $\lambda \boldsymbol{x} \in C$. Choosing λ so that $\lambda > b/c$ we get

$$\langle \boldsymbol{a}, \lambda \boldsymbol{x} \rangle = \lambda c > b,$$

in contradiction to $C \subset H_-^a$. This shows that H is a linear separating hyperplane. □

Theorem 10.2. (Farkas' Lemma) Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, there exists a vector $\boldsymbol{x}$ such that

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq 0$$

if and only if there is not $\boldsymbol{y} \in \mathbb{R}^m$ such that

$$\boldsymbol{A}^\intercal \boldsymbol{y} \geq \boldsymbol{0}, \quad \langle \boldsymbol{y}, \boldsymbol{b} \rangle < 0.$$

Proof. Assume $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has a solution $\boldsymbol{x} \geq 0$. Then for any $\boldsymbol{y} \neq \boldsymbol{0}$ such that $\boldsymbol{A}^\intercal \boldsymbol{y} \geq \boldsymbol{0}$,

$$0 \leq \langle \boldsymbol{A}^\intercal \boldsymbol{y}, \boldsymbol{x} \rangle = \langle \boldsymbol{y}, \boldsymbol{A}\boldsymbol{x} \rangle = \langle \boldsymbol{y}, \boldsymbol{b} \rangle,$$

which shows that $\boldsymbol{A}^\intercal \boldsymbol{y} \geq \boldsymbol{0}$ and $\langle \boldsymbol{y}, \boldsymbol{b} \rangle < 0$ are not simultaneously possible.

Assume now that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has no solution that satisfies $\boldsymbol{x} \geq 0$. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ be the columns of $\boldsymbol{A}$. The set of $\boldsymbol{A}\boldsymbol{x}$ for $\boldsymbol{x} \geq \boldsymbol{0}$ is the set of all nonnegative linear combinations

$$C = \{\boldsymbol{z} \in \mathbb{R}^m \mid \boldsymbol{z} = x_1 \boldsymbol{a}_1 + \cdots + x_n \boldsymbol{a}_n,\ x_i \geq 0\},$$

and this set is a convex cone. The assumption that there is no nonnegative $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ means that $\boldsymbol{b} \notin C$. By Lemma 10.1, there exists a linear hyperplane $H = \{\boldsymbol{x} \mid \langle \boldsymbol{y}, \boldsymbol{x} \rangle = 0\}$ such that $C \in H_-$ and $\boldsymbol{b} \in \operatorname{int} H_+$. Formulated differently, there exists a $\boldsymbol{y} \in \mathbb{R}^m$ such that

$$\forall \boldsymbol{z} \in C \colon \langle \boldsymbol{z}, \boldsymbol{y} \rangle \geq 0, \quad \langle \boldsymbol{b}, \boldsymbol{y} \rangle < 0.$$

Since every $\boldsymbol{z} \in C$ has the form $\boldsymbol{z} = \sum_{i=1}^n x_i \boldsymbol{a}_i$ with $x_i \geq 0$, the relation

$$\langle \boldsymbol{z}, \boldsymbol{y} \rangle = \sum_{i=1}^n x_i \langle \boldsymbol{a}_i, \boldsymbol{y} \rangle \geq 0$$

for all $\boldsymbol{x} \geq 0$ is equivalent to the condition that $\langle \boldsymbol{a}_i, \boldsymbol{y} \rangle \geq 0$ for $1 \leq i \leq n$, which again is equivalent to $\boldsymbol{A}^\intercal \boldsymbol{y} \geq \boldsymbol{0}$. This concludes the proof. □

The following consequence is perhaps a more familiar form of Farkas' Lemma.

Corollary 10.3. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, there exists a vector $\boldsymbol{x} \neq \boldsymbol{0}$ such that

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

if and only there is no $\boldsymbol{y}$ such that

$$\boldsymbol{y} \geq \boldsymbol{0}, \quad \boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{0}, \quad \langle \boldsymbol{y}, \boldsymbol{b} \rangle < 0.$$

Proof. Consider the matrix

$$\boldsymbol{A}' := \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A} & \boldsymbol{I} \end{pmatrix},$$

where $\boldsymbol{I}$ is the $m \times m$ identity matrix. A nonnegative solution of $\boldsymbol{A}'\boldsymbol{x}' = \boldsymbol{b}$ has the form $\boldsymbol{x}' = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)^\intercal$, and implies $\boldsymbol{A}(\boldsymbol{x}_1 - \boldsymbol{x}_2) + \boldsymbol{x}_3 = \boldsymbol{b}$. Therefore, such a solution $\boldsymbol{x}'$ exists if and only if the system

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

has a solution. Applying Theorem 10.2, the complementary condition is

$$\boldsymbol{A}'^\intercal \boldsymbol{y} \geq \boldsymbol{0}, \quad \langle \boldsymbol{b}, \boldsymbol{y} \rangle < 0,$$

which in terms of $\boldsymbol{A}$ translates to

$$\boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{0}, \quad \boldsymbol{y} \geq 0, \quad \langle \boldsymbol{b}, \boldsymbol{y} \rangle < 0.$$

This concludes the proof. □

One more important corollary will be given without proof.

Corollary 10.4. Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. Then for $\delta > 0$ and every vector $\boldsymbol{x} \in \mathbb{R}^n$ with $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$, $\langle \boldsymbol{c}, \boldsymbol{x} \rangle \leq \delta$ holds if and only if there exists $\boldsymbol{y} \in \mathbb{R}^m$ such that

$$\boldsymbol{y} \geq \boldsymbol{0}, \quad \boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{c}, \quad \langle \boldsymbol{y}, \boldsymbol{b} \rangle \leq \delta.$$

Linear programming duality

After studying the feasible sets of linear programming, the polyhedra, we now return to linear programming itself, in the form

$$\text{maximize} \langle \boldsymbol{c}, \boldsymbol{x} \rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \tag{10.1}$$

Geometrically this amounts to moving the hyperplane orthogonal to $\boldsymbol{c}$ to the highest level along $\boldsymbol{c}$, under the condition that it still intersects $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$.

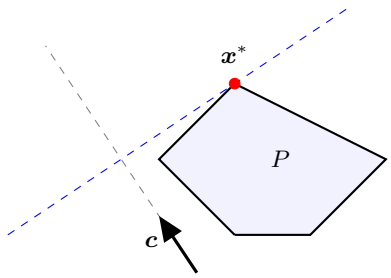


Figure 10.2: Geometry of linear programming

The famous duality theorem for linear programming states that if the maximum of (10.1) exists, then it coincides with the solution of a dual linear programming problem.

Theorem 10.5. Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$. Then the optimal value of

$$\text{maximize} \langle \boldsymbol{c}, \boldsymbol{x} \rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{P}$$

coincides with the optimal value of

$$\text{minimize} \langle \boldsymbol{b}, \boldsymbol{y} \rangle \quad \text{subject to} \quad \boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{c},\ \boldsymbol{y} \geq \boldsymbol{0}, \tag{D}$$

provided both (P) and (D) have a finite solution.

The problem (P) is called the primal problem, and (D) the dual problem.

Example 10.6. Consider the simple problem

$$\text{maximize}\, x_1 \quad \text{subject to} \quad x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0.$$

The dual problem is

$$\text{minimize}\, y_1 \quad \text{subject to} \quad y_1 - y_2 = 1, y_1 - y_3 = 0, y_1 \geq 0, y_2 \geq 0, y_3 \geq 0.$$

For the proof we need the following observation.

Lemma 10.7. Let $P \subset \mathbb{R}^n$ be a polyhedron and $\boldsymbol{c} \in \mathbb{R}^n$ such that $\sup_{\boldsymbol{x} \in P} \langle \boldsymbol{c}, \boldsymbol{x} \rangle$ is finite. Then the supremum is attained, that is, it is a maximum.

Proof of Theorem 10.5. Let $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \in \boldsymbol{b}\}$ and $D = \{\boldsymbol{y} \in \mathbb{R}^m \mid \boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq \boldsymbol{0}\}$. If $\boldsymbol{x} \in P$ and $\boldsymbol{y} \in Q$, then

$$\langle \boldsymbol{c}, \boldsymbol{x} \rangle = \langle \boldsymbol{A}^\intercal \boldsymbol{y}, \boldsymbol{x} \rangle = \langle \boldsymbol{y}, \boldsymbol{A}\boldsymbol{x} \rangle \leq \langle \boldsymbol{y}, \boldsymbol{b} \rangle,$$

so that in particular

$$\max_{\boldsymbol{x} \in P} \langle \boldsymbol{c}, \boldsymbol{x} \rangle \leq \min_{\boldsymbol{y} \in Q} \langle \boldsymbol{b}, \boldsymbol{y} \rangle,$$

which shows one inequality. To show the other inequality, set $\delta = \max_{\boldsymbol{x}\in P}\langle \boldsymbol{c}, \boldsymbol{x}\rangle$. By definition, if $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$, then $\langle \boldsymbol{c}, \boldsymbol{x}\rangle \leq \delta$. By Corollary 10.4, there exists a vector $\boldsymbol{y} \in \mathbb{R}^m$ such that

$$\boldsymbol{y} \geq \boldsymbol{0}, \quad \boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{c}, \quad \langle \boldsymbol{b}, \boldsymbol{y}\rangle \leq \delta.$$

In particular,

$$\min_{\boldsymbol{y}\in Q}\langle \boldsymbol{b}, \boldsymbol{y}\rangle \leq \delta = \max_{\boldsymbol{x}\in P}\langle \boldsymbol{c}, \boldsymbol{x}\rangle.$$

This finishes the proof. □

So far we have seen aspects of the geometry of linear programming, discussed the feasible sets (polyhedra) and presented a useful duality theorem: the optimal value of

$$\text{maximize}\langle \boldsymbol{c}, \boldsymbol{x}\rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{P}$$

coincides with the optimal value of

$$\text{minimize}\langle \boldsymbol{b}, \boldsymbol{y}\rangle \quad \text{subject to} \quad \boldsymbol{A}^\intercal \boldsymbol{y} = \boldsymbol{c},\ \boldsymbol{y} \geq \boldsymbol{0}, \tag{D}$$

provided both (P) and (D) have a finite solution.

We now turn attention to algorithms for linear programming. We present the main idea behind the classical Simplex Algorithm and then discuss the more modern class of Interior Point Algorithms in more detail. The latter can be shown to be efficient in a very precise sense (polynomial time) and generalize to non-linear convex optimization.

A first algorithm for linear programming

For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ let $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ be the polyhedron of feasible points for (P). Recall that a vertex of P is a zero-dimensional face of P, or equivalently, a point that cannot be written as convex combination of two distinct points of P.

If the optimization problem (P) has a solution, then it can be shown that there exists a vertex $\boldsymbol{x}^*$ such that

$$\max_{\boldsymbol{x}\in P}\langle \boldsymbol{c}, \boldsymbol{x}\rangle = \langle \boldsymbol{c}, \boldsymbol{x}^*\rangle.$$

Intuitively this is clear by looking at a picture, as in Figure 10.3: Move a hyperplane orthogonal to the objective $\boldsymbol{c}$ along this direction to the highest level that still intersects a polyhedron P, and try to imagine that this intersection does not contain a vertex.

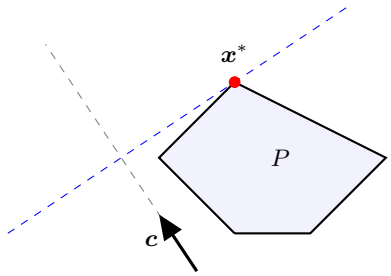


Figure 10.3: Geometry of linear programming

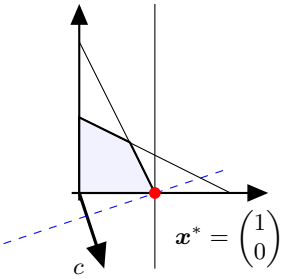
This crucial observation turns linear programming into a finite problem: we only have to test the objective function on the finite set of vertices to determine the maximizer. The vertices can be identified as the solutions of the systems of equations

$$\boldsymbol{A}_I \boldsymbol{x} = \boldsymbol{b}_I,$$

for which $\boldsymbol{A}_I$ has full rank and $\boldsymbol{x} \in P$, where $I \subset \{1, \dots, m\}$ is a subset of indices with $|I| = d$, and $\boldsymbol{A}_I, \boldsymbol{b}_I$ are the matrix and vector arising from $\boldsymbol{A}$ and $\boldsymbol{b}$ by taking the rows indexed by I. This gives a first primitive algorithm for solving the optimization problem (P).

Example 10.8. Consider the linear programming problem

$$\begin{aligned} \text{maximize} & x_1 - 3x_2 \\ \text{subject to} \quad & 0.5x_1 + x_2 \leq 1 \\ & x_1 + 0.5x_2 \leq 1 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_1 \leq 1 \end{aligned}$$



The matrix $\boldsymbol{A}$ and vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ are

$$\boldsymbol{A} = \begin{pmatrix} 0.5 & 1 \\ 1 & 0.5 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} 1 \\ -3 \end{pmatrix}.$$

We see that the minor $\boldsymbol{A}_{\{3,5\}} = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}$, corresponding to the two parallel hyperplanes $x_1 = 0$ and $x_1 = 1$, is singular and does not define a vertex, while all other minors are invertible. For the minor $I = \{1, 2\}$, $\boldsymbol{A}_I \boldsymbol{x} = \boldsymbol{b}_I$ has the form

$$\begin{aligned} 0.5x_1 + x_2 &= 1 \\ x_1 + 0.5x_2 &= 1, \end{aligned}$$

which has the solution $\boldsymbol{x} = (2/3, 2/3)^\intercal$. Since $\boldsymbol{x}$ also satisfies all the other inequalities, it is a vertex. In a similar fashion we can find all the vertices as the points

$$\boldsymbol{x}_1 = \begin{pmatrix} 2/3 \\ 2/3 \end{pmatrix}, \ \boldsymbol{x}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \boldsymbol{x}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ \boldsymbol{x}_4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The values of the objective function at these vertices are

$$\langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle = -\frac{4}{3}, \ \langle \boldsymbol{c}, \boldsymbol{x}_2 \rangle = 1, \ \langle \boldsymbol{c}, \boldsymbol{x}_3 \rangle = -3, \ \langle \boldsymbol{c}, \boldsymbol{x}_4 \rangle = 0.$$

It follows that the optimization problem has the optimal value 1 at $\boldsymbol{x}^* = (1, 0)^\intercal$.

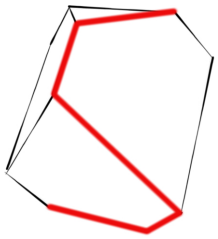
Unfortunately, this method requires solving up to $\binom{m}{n}$ systems of linear equations, which can be exponential in n in the worst case, making this method not practical.

Example 10.9. Consider the hypercube

$$\begin{aligned} -1 \leq x_1 \leq 1 \\ \dots \\ -1 \leq x_n \leq 1. \end{aligned}$$

It has 2^n vertices that need to be checked. Even though there are so many vertices, one can travel along edge between any two vertices in at most n steps.

The simplex algorithm is a way to search through the vertices in a more clever way than just listing all of them. The idea is to start with a vertex and see if there is a vertex connected to it by an edge that has a bigger objective value. If not, we are done. If there is a neighbouring vertex, we move there and continue the process. We keep walking along the vertices of the feasible polytope until we find an optimal one. The



simplex algorithm is one of the most successful algorithms around and usually very fast, even though there examples that show that its worst case running time can be exponential.

11

Portfolio Optimization

So far we have only dealt with constrained optimization problems where the objective and the constraints are linear. We now turn attention to general problems of the form

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}, \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{f} = (f_1, \ldots, f_m)^\intercal$, $\boldsymbol{h} = (h_1, \ldots, h_p)$, and the inequalities are componentwise. The problem (P) is convex, if f and the g_i are convex, and the h_j are linear. We also denote by $\mathrm{dom}(f)$ the domain of f, which is the set of points $\boldsymbol{x}$ where f takes a finite value. The feasible set

$$\mathcal{F} = \{\boldsymbol{x} \mid f_i(\boldsymbol{x}) \leq 0,\ h_j(\boldsymbol{x}) = 0,\ 1 \leq i \leq m,\ 1 \leq j \leq \ell\}$$

for a convex constrained problem is a convex set.

Quadratic Programming and Portfolio Optimization

The simplest case of non-linear, constrained convex optimization is quadratic programming. Quadratic programming problems are of the form

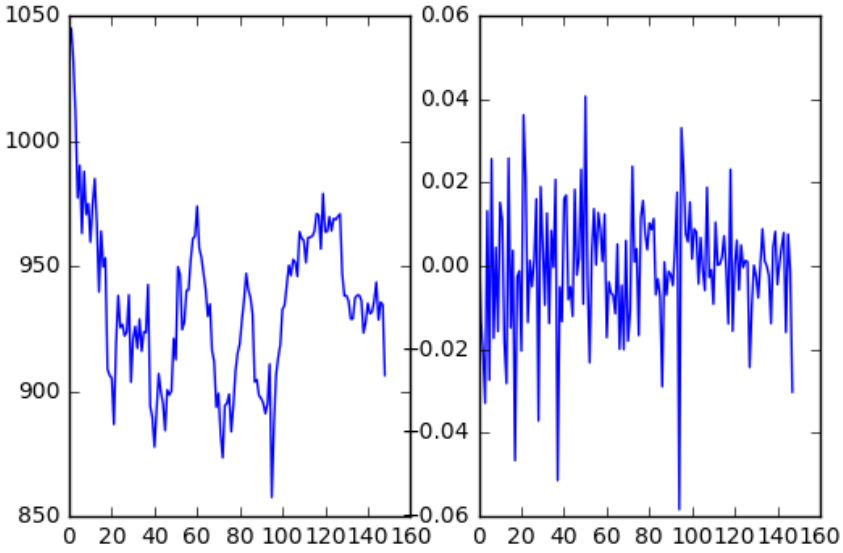
$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{x}^\intercal \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\intercal \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{aligned}$$

with $\boldsymbol{Q}$ symmetric and positive semidefinite. Note that this problem combines two problems we studied in some detail before: minimizing a quadratic function, and linear constraints. Just as we did for unconstrained minimization and for linear programming, we will derive optimality conditions for such problems. Before studying the theory, we first present an important application: portfolio optimization.

Mean-variance portfolio theory

If we invest an amount x^0 into a product at period 0, and at period 1 (for example, one day) the value is x^1, then the relative return is defined as

$$r = \frac{x^1 - x^0}{x^0}.$$



The following figure shows the price movement and the returns of a stock from January to November 2016.

As we can't predict the future, we usually work with the expected return $\boldsymbol{E}[r] = \mu$, which is a statistical estimate of the future return. One naive method of estimating the future return is by taking the average of past returns, but other more sophisticated methods are possible.

In portfolio optimization, we have a proportion x_i of our available funds that we want to invest in a stock i (in particular, as x_i measures a proportion, we have $\sum_{i=1}^{n} x_i = 1$). We may or may not allow $x_i < 0$, which would correspond to short-selling or borrowing. Given this allocation, the overall return is $\boldsymbol{x}^\intercal \boldsymbol{r}$, where $\boldsymbol{x} = (x_1, \ldots, x_n)^\intercal$ and $\boldsymbol{r} = (r_1, \ldots, r_n)^\intercal$ is the vector of (relative) returns. If $\boldsymbol{\mu} = \boldsymbol{E}[\boldsymbol{r}]$ denotes the vector of expected returns, then the total expected return is

$$\mu = \boldsymbol{x}^\intercal \boldsymbol{\mu} = \sum_{i=1}^{n} x_i \mu_i.$$

The risk in an investment is measured in terms of the variance of the returns $r = \boldsymbol{x}^\intercal \boldsymbol{r}$. Let $\boldsymbol{\Sigma}$ denote the covariance matrix, where the (i, j)-th entry is $\mathrm{Cov}(r_i, r_j) = \boldsymbol{E}[(r_i - \mu_i)(r_j - \mu_j)]$. The (i, j)-th entry measures how much products i and j are correlated. The variance of the returns r is then the quadratic function

$$\boldsymbol{x}^\intercal \Sigma \boldsymbol{x}.$$

A portfolio optimization problem either seeks to maximize the return while bounding the risk,

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{x}^\intercal \boldsymbol{\mu} \\ \text{subject to} \quad & \boldsymbol{x}^\intercal \boldsymbol{\Sigma} \boldsymbol{x} \leq \sigma, \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0, \end{aligned}$$

or minimize the risk given a certain target return μ,

$$\begin{aligned} \text{minimize} & \boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x}^\mathsf{T}\boldsymbol{\mu} = \mu \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0. \end{aligned}$$

Both of these problems are convex optimization problems. The constraints $x_i \geq 0$ mean that we are not allowed to short-sell; when dealing with futures or options, or if we are a large institutional investor, we may drop these constraints. As we will see later, dropping the inequality constraints from the second formulation allows the problem to be solved in closed form.

A third form of the portfolio optimization problem is to combine the expected return and the risk into a single objective function, as follows:

$$\begin{aligned} \text{maximize} & \boldsymbol{\mu}^\mathsf{T}\boldsymbol{x} - \gamma\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0. \end{aligned}$$

Note that this is a convex quadratic problem: we can transform it into a minimization problem with the matrix $\boldsymbol{\Sigma}$ by changing the sign. The parameter γ is called a risk aversion parameter; it adjusts the level of risk we are willing to take. If $\gamma = 0$, then the quadratic term does not feature in the objective and we just aim to maximize the expected return. If γ is big, then the risk term weighs heavily on the objective function, and the optimal value will like be one where $\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{x}$ is small. By varying the value of γ, we get different expected return / risk (variance) trade-offs. The following code computes this trade-off curve for a portfolio of 10 stock from the FT100 index. The mean and covariance where computing by averaging over the past 60 trading days (3 months). The x-axis is the standard deviation, which is given as the square root of the variance (risk).

In [6]:
```
from datetime import datetime, date
import pandas as pd
from pandas_datareader import data, wb
import numpy as np
import matplotlib.pyplot as plt
from cvxpy import *
```

We first use the functionality of the Pandas module to load the price data from the Yahoo Finance web site.

In [7]:
```
START = datetime(2016,1,1)
END = date.today()
TICKER = ['ADN', 'AZN', 'EZJ', 'GSK', 'ITV', 'LSE', 'TSCO', 'PSON', 'PRU', 'DGE']
mydata = data.DataReader('TSCO', "yahoo", START, END)
dates = mydata.index
df = pd.DataFrame(index=dates, columns=TICKER)
for x in TICKER:
    mydata = data.DataReader(x, "yahoo", START, END)
    df.loc[:,x] = mydata['Adj Close']
df = df.dropna()
df.head(2)
```

The following shows a small sample of the data loaded.

Date	ADN	AZN	EZJ	GSK	ITV	LSE	TSCO
2016-01-04	2.41197	31.926706	84.900002	37.813187	0.4	0.002	82.800593
2016-01-05	2.41197	32.404650	86.620003	38.028194	0.4	0.001	82.741275

We next compute the returns and the mean and covariance.

In [8]:
```
price_matrix = df.values
returns = (price_matrix[1:]-price_matrix[0:-1])/price_matrix[0:-1]
mu = np.mean(returns[-60:,:], axis=0)
Sigma = np.cov(returns[-60:,:].T)
n = 10
```

In the next step, we start the CVXPY engine and compute the mean-variance trade-off curve. We include the constraint $\boldsymbol{x} \geq 0$ to disallow for short-selling.

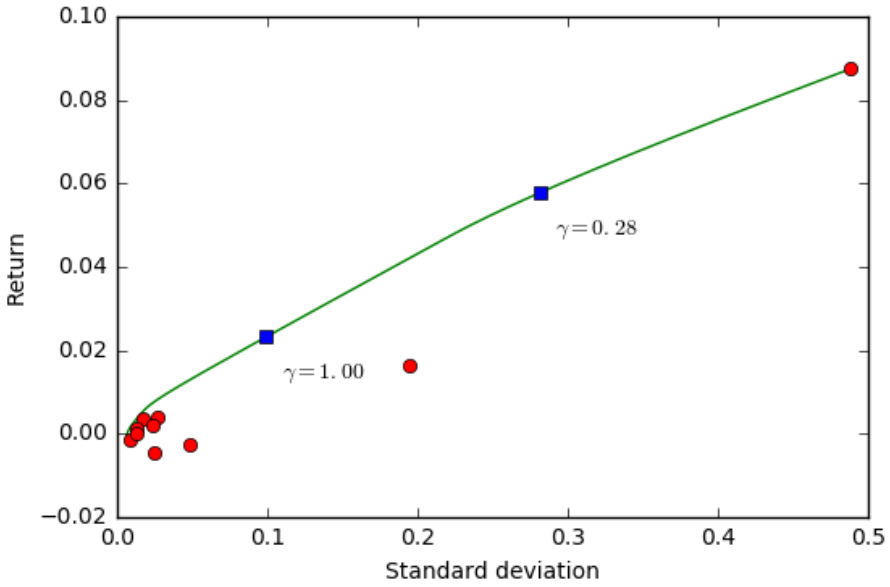
In [9]:
```
x = Variable(n)
gamma = Parameter(sign='positive')
ret = mu.T*x
risk = quad_form(x, Sigma)
prob = Problem(Maximize(ret - gamma*risk),
               [sum_entries(x) == 1,
                x >= 0])
```

In [10]:
```
SAMPLES = 1000
risk_data = np.zeros(SAMPLES)
ret_data = np.zeros(SAMPLES)
gamma_vals = np.logspace(-2, 3, num=SAMPLES)
for i in range(SAMPLES):
    gamma.value = gamma_vals[i]
    prob.solve()
    risk_data[i] = sqrt(risk).value
    ret_data[i] = ret.value
```

In [11]:
```
markers_on = [290, 400]
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(risk_data, ret_data, 'g-')
for marker in markers_on:
    plt.plot(risk_data[marker], ret_data[marker], 'bs')
    ax.annotate(r"$\gamma = %.2f$"
    % gamma_vals[marker], xy=(risk_data[marker]+0.01, ret_data[marker]-0.01))
for i in range(n):
    plt.plot(sqrt(Sigma[i,i]).value, mu[i], 'ro')
plt.xlabel('Standard deviation')
plt.ylabel('Return')
plt.show()
```

The red dots represent the standard deviation and expected return of the individual stocks. The two blue dots indicate the standard deviation and expected return of the portfolio for two values of γ. Which value of γ we go for depends on how much risk we are willing to take: if we are risk-averse, we may prefer the $\gamma = 1$ value to the $\gamma = 0.028$ value, at the expense of smaller expected returns.

In practical applications there are a lot of other factors to be considered, such as whether the estimation procedure for the mean and covariance makes sense. In addition, it is often common to add terms that account for transaction costs into the objective funtion.


0.10
0.08
0.06
0.04
0.02
0.00
−0.02
Return
0.0
0.1
0.2
0.3
0.4
0.5
Standard deviation
$\gamma = 0.28$
$\gamma = 1.00$

12

Lagrange Duality

In this chapter we study optimality conditions for convex problems of the form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & f(\boldsymbol{w}) \\ \text{subject to} \quad & \boldsymbol{f}(\boldsymbol{w}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{w}) = \boldsymbol{0}, \end{aligned} \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$, $\boldsymbol{f} = (f_1, \ldots, f_m)^\mathsf{T}$, $\boldsymbol{h} = (h_1, \ldots, h_p)^\mathsf{T}$, and the inequalities are componentwise. We assume that f and the f_i are convex, and the h_j are linear. It is also customary to write the conditions $\boldsymbol{h}(\boldsymbol{w}) = \boldsymbol{0}$ as $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}$, with $h_j(\boldsymbol{w}) = \boldsymbol{a}_j^\mathsf{T}\boldsymbol{w} - b_j$, $\boldsymbol{a}_j^\mathsf{T}$ being the j-th row of $\boldsymbol{A}$. The feasible set of (P) is the set

$$\mathcal{F} = \{\boldsymbol{w} \mid f_i(\boldsymbol{w}) \leq 0 \ \text{ for } i \in [m], \ \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}\}$$

It is easy to see that $\mathcal{F}$ is convex if the f_i are convex. If the objective f and the f_i are also linear, then (P) is called a linear programming problem, and $\mathcal{F}$ is a polyhedron. Such problems have been studied extensively and can be solved with efficient algorithms such as the simplex method.

A first-order optimality condition

We first generalize the standard first-order optimality conditions for differentiable functions to the setting of constrained convex optimization.

Theorem 12.1. Let $f \in C^1(\mathbb{R}^d)$ be a convex, differentiable function, and consider a convex optimization problem of the form (P). Then $\boldsymbol{w}^*$ is a minimizer of the optimization problem

$$\text{minimize } f(\boldsymbol{w}) \quad \text{ subject to } \quad \boldsymbol{w} \in \mathcal{F}$$

if and only if for all $\boldsymbol{w} \in \mathcal{F}$,

$$\nabla f(\boldsymbol{w}^*)^\mathsf{T}(\boldsymbol{w} - \boldsymbol{w}^*) \geq 0, \tag{12.1}$$

where $\mathcal{F}$ is the feasible set of the problem.

Proof. Suppose $\boldsymbol{w}^*$ is such that (12.1) holds. Then, since f is a convex function, for all $\boldsymbol{w} \in \mathcal{F}$ we have,

$$f(\boldsymbol{w}) \geq f(\boldsymbol{w}^*) + \langle \nabla f(\boldsymbol{w}^*), (\boldsymbol{w} - \boldsymbol{w}^*) \rangle \geq f(\boldsymbol{w}^*),$$

which shows that $\boldsymbol{w}^*$ is a minimizer in $\mathcal{F}$. To show the opposite direction, assume that $\boldsymbol{w}^*$ is a minimizer but that (12.1) does not hold. This means that there exists a $\boldsymbol{w} \in \mathcal{F}$ such that $\langle \nabla f(\boldsymbol{w}^*), (\boldsymbol{w} - \boldsymbol{w}^*) \rangle < 0$. Since both $\boldsymbol{w}^*$ and $\boldsymbol{w}$ are in $\mathcal{F}$ and $\mathcal{F}$ is convex, any point $\boldsymbol{v}(\lambda) = (1-\lambda)\boldsymbol{w}^* + \lambda \boldsymbol{w}$ with $\lambda \in [0,1]$ is also in $\mathcal{F}$. At $\lambda = 0$ we have

$$\frac{df}{d\lambda} f(\boldsymbol{v}(\lambda))|_{\lambda=0} = \langle \nabla f(\boldsymbol{w}^*), (\boldsymbol{w} - \boldsymbol{w}^*) \rangle < 0.$$

Since the derivative at $\lambda = 0$ is negative, the function $f(\boldsymbol{v}(\lambda))$ is decreasing at $\lambda = 0$, and therefore, for small $\lambda > 0$, $f(\boldsymbol{v}(\lambda)) < f(\boldsymbol{v}(0)) = f(\boldsymbol{w}^*)$, in contradiction to the assumption that $\boldsymbol{w}^*$ is a minimizer. □

Example 12.2. In the absence of constraints, $\mathcal{F} = \mathbb{R}^d$, and the statement says that

$$\forall \boldsymbol{w} \in \mathbb{R}^n \colon \langle \nabla f(\boldsymbol{w}^*), (\boldsymbol{w} - \boldsymbol{w}^*) \rangle \geq 0.$$

Given $\boldsymbol{w}$ such that $\langle \nabla f(\boldsymbol{w}^*), (\boldsymbol{w} - \boldsymbol{w}^*) \rangle \geq 0$, then replacing $\boldsymbol{w}$ by $2\boldsymbol{w}^* - \boldsymbol{w}$ we also have the converse inequality, and therefore the optimality condition is equivalent to saying that $\nabla f(\boldsymbol{w}^*) = \boldsymbol{0}$. We therefore recover the well-known first order optimality condition.

Geometrically, the first order optimality condition means that the set

$$\{\boldsymbol{w} \mid \langle \nabla f(\boldsymbol{w}^*), \boldsymbol{w} \rangle = \langle \nabla f(\boldsymbol{w}^*), \boldsymbol{w}^* \rangle\}$$

defines a supporting hyperplane to the set $\mathcal{F}$.

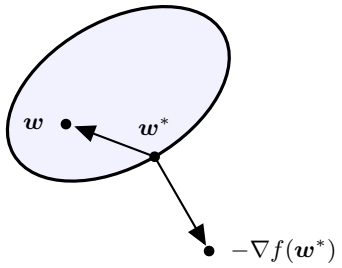


Figure 12.1: Optimality condition

Lagrangian duality

Recall the method of Lagrange multipliers. Given two functions $f(x, y)$ and $h(x, y)$, if the problem

$$\text{minimize } f(x,y) \qquad \text{subject to} \quad h(x,y) = 0$$

has a solution (x^*, y^*), then there exists a parameter λ, the Lagrange multiplier, such that

$$\nabla f(x^*, y^*) = \lambda \nabla h(x^*, y^*). \tag{12.2}$$

In other words, if we define the Lagrangian as

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda h(x, y),$$

then (12.2) says that $\nabla \mathcal{L}(x^*, y^*, \lambda) = 0$ for some λ. The intuition is as follows. The set

$$M = \{(x, y) \in \mathbb{R}^2 \mid h(x, y) = 0\}$$

is a curve in $\mathbb{R}^2$, and the gradient $\nabla h(x, y)$ is perpendicular to M at every point $(x, y) \in M$. For someone living inside M, a vector that is perpendicular to M is not visible, it is zero. Therefore the gradient $\nabla f(x, y)$ is zero as viewed from within M if it is perpendicular to M, or equivalently, a multiple of $\nabla h(x, y)$.

Alternatively, we can view the graph of $f(x, y)$ in three dimensions. A maximum or minimum of $f(x, y)$ along the curve defined by $h(x, y) = 0$ will be a point at which the direction of steepest ascent $\nabla f(x, y)$ is perpendicular to the curve $h(x, y) = 0$.

Example 12.3. Consider the function $f(x, y) = x^2 y$ with the constraint $h(x, y) = x^2 + y^2 - 3$ (a circle of radius $\sqrt{3}$). The Lagrangian is the function

$$\mathcal{L}(x, y, \lambda) = x^2 y - \lambda(x^2 + y^2 - 3).$$

Computing the partial derivatives gives the three equations

$$\begin{aligned}
\frac{\partial}{\partial x}\mathcal{L} &= 2xy - 2\lambda x = 0 \\
\frac{\partial}{\partial y}\mathcal{L} &= x^2 - 2\lambda y = 0 \\
\frac{\partial}{\partial \lambda}\mathcal{L} &= x^2 + y^2 - 3 = 0.
\end{aligned}$$

From the second equation we get $\lambda = \frac{x^2}{2y}$, and the first and third equations become

$$\begin{aligned}
2xy - \frac{x^3}{y} &= 0 \\
x^2 + y^2 - 3 &= 0.
\end{aligned}$$

Solving this system, we get six critical point $(\pm\sqrt{2}, \pm 1)$, $(0, \pm\sqrt{3})$. To find out which one of these is the minimizers, we just evaluate the function f on each of these.

We now turn to convex problems of the more general form form

$$\begin{aligned}
\underset{\boldsymbol{w}}{\text{minimize}} \quad & f(\boldsymbol{w}) \\
\text{subject to} \quad & \boldsymbol{f}(\boldsymbol{w}) \leq \mathbf{0} \\
& \boldsymbol{h}(\boldsymbol{w}) = \mathbf{0},
\end{aligned} \tag{12.3}$$

Denote by $\mathcal{D}$ the domain of all the functions f, f_i, h_j, i.e.,

$$\mathcal{D} = \operatorname{dom}(f) \cap \operatorname{dom}(f_1) \cap \cdots \cap \operatorname{dom}(f_m) \cap \operatorname{dom}(h_1) \cap \cdots \cap \operatorname{dom}(h_p).$$

Assume that $\mathcal{D}$ is not empty and let p^* be the optimal value of (12.3).

The Lagrangian of the system is defined as

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{w}) + \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{w}) + \boldsymbol{\mu}^\mathsf{T} h(\boldsymbol{w}) = f(\boldsymbol{w}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{w}) + \sum_{i=1}^{p} \mu_i h_i(\boldsymbol{w}).$$

The vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are called the dual variables or Lagrange multipliers of the system. The domain of $\mathcal{L}$ is $\mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$.

Definition 12.4. The Lagrange dual of the problem (12.3) is the function

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{w} \in \mathcal{D}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}).$$

If the Lagrangian $\mathcal{L}$ is unbounded from below, then the value is $-\infty$.

The Lagrangian $\mathcal{L}$ is linear in the λ_i and μ_j variables. The infimum of a family of linear functions is concave, so that the Lagrange dual is a concave function. Therefore the negative $-g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is a convex function.

Lemma 12.5. Let p^* be the optimal value of the problem 12.3. Then for any $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\lambda} \geq \mathbf{0}$ we have

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq p^*.$$

Proof. Let $\boldsymbol{w}^*$ be a feasible point for (12.3), that is,

$$f_i(\boldsymbol{w}^*) \leq 0, \quad h_j(\boldsymbol{w}^*) = 0, \ 1 \leq i \leq m, \ 1 \leq j \leq p.$$

Then for $\boldsymbol{\lambda} \geq \mathbf{0}$ and any $\boldsymbol{\mu}$, since each $h_j(\boldsymbol{w}^*) = 0$ and $\lambda_j f_j(\boldsymbol{w}^*) \leq 0$,

$$\mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{w}^*) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{w}^*) + \sum_{j=1}^{p} \mu_j h_j(\boldsymbol{w}^*) \leq f(\boldsymbol{w}^*).$$

In particular,

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f(\boldsymbol{w}^*).$$

Since this holds for all feasible $\boldsymbol{w}^*$, it holds in particular for the $\boldsymbol{w}^*$ that minimizes (12.3), for which $f(\boldsymbol{w}^*) = p^*$. □

The Lagrange dual problem of the optimization problem (12.3) is the problem

$$\text{maximize } g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \qquad \text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}. \tag{12.4}$$

If d^* is the optimal value of (12.4), then $d^* \leq p^*$. In the special case of linear programming we actually have $d^* = p^*$. We will see that under certain conditions, we have $d^* = p^*$ for more general problems, but this is not always the case.

13

KKT Optimality Conditions

For convex problems of the form

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{w}) \\ \text{subject to} & \boldsymbol{f}(\boldsymbol{w}) \leq \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}, \end{array} \tag{P}$$

we introduced the Lagrangian $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ and defined the Lagrange dual as

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{w} \in \mathcal{D}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}).$$

We saw that $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is a lower bound on the optimal value of (P). Note that we wrote the linear equality conditions $h_j(\boldsymbol{w}) = 0$ as system of linear equations $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}$. We will derive conditions under which the lower bound provided by the Lagrange dual matches the upper bound, and derive a system of equations and inequalities that certify optimality, the Karush-Kuhn-Tucker (KKT) conditions. These conditions can be seen as generalizations of the first-order optimality conditions to the setting when equality and inequality constraints are present.

Constraint qualification

Let p^* and d^* denote the primal and dual optimal values, so that

$$d^* = \sup_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\mu}} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq \inf_{\boldsymbol{w} \in \mathcal{D}} \{f(\boldsymbol{w}) \mid f_i(\boldsymbol{w}) \leq 0,\ \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}\} = p^*.$$

Once certain conditions, called constraint qualifications, hold, we can ensure that strong duality holds, which means $d^* = p^*$. One particular such constraint qualification is Slater's Theorem.

Theorem 13.1. (Slater conditions) Assume that the interior of the domain $\mathcal{D}$ of (P) is non-empty, that the problem (P) is convex, and that there exists a point $\boldsymbol{w} \in \mathcal{D}$ such that

$$f_i(\boldsymbol{w}) < 0,\ 1 \leq i \leq m, \quad \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b},\ 1 \leq j \leq p.$$

Assume that $\boldsymbol{A}$ has maximal rank. Then $d^* = p^*$.

The proof makes use of a fundamental result on convex sets, the separating hyperplane theorem. For an affine hyperplane $H = \{\boldsymbol{w} \colon \boldsymbol{a}^\intercal \boldsymbol{w} + b = 0\}$, we denote by $H_+ = \{\boldsymbol{w} \colon \boldsymbol{a}^\intercal \boldsymbol{w} + b \geq 0\}$ one of the two closed halfspaces defined by the hyperplane, and by H_- the other.

Theorem 13.2 (Separating Hyperplane Theorem). Let C and D be disjoint, nonempty convex subsets of $\mathbb{R}^d$. Then there exists an affine hyperplane H such that $C \subset H_+$ and $D \subset H_-$.

A hyperplane with the properties of Theorem 13.2 is called a separating hyperplane. The hyperplane separates the two sets strictly, if each of the sets is contained in the interior of the respective halfspaces. It can be shown that for compact convex sets, there exists a hyperplane separating them strictly.

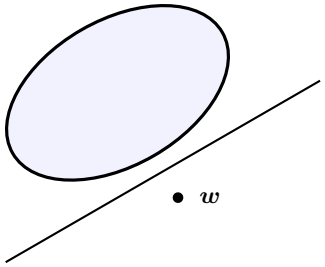


Figure 13.1: A hyperplane separating a convex sets and a point

Proof of Theorem 13.1. Assume $\boldsymbol{A}$ has rank p (the number of rows). Assume moreover that p^* is finite, since if $p^* = -\infty$, then by weak duality we already have $d^* = p^*$. Define the convex set

$$\mathcal{A} = \{(\boldsymbol{u}, \boldsymbol{v}, t) \in \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R} \mid \forall \boldsymbol{w} \in \mathcal{D}, f_i(\boldsymbol{w}) \leq u_i, \boldsymbol{A}\boldsymbol{w} - \boldsymbol{b} = \boldsymbol{v}, f(\boldsymbol{w}) \leq t\}.$$

Then the optimal value of (P) is

$$p^* = \inf\{t \mid (\boldsymbol{0}, \boldsymbol{0}, t) \in \mathcal{A}\}.$$

Define the convex set $\mathcal{B}$ as

$$\mathcal{B} = \{(\boldsymbol{0}, \boldsymbol{0}, s) \in \mathbb{R}^m \times \mathbb{R}^p \times \mathbb{R} \mid s < p^*\}.$$

The sets $\mathcal{A}$ and $\mathcal{B}$ are disjoint. To see this, assume to the contrary that there is a point $\boldsymbol{z} \in \mathcal{A} \cap \mathcal{B}$. Then since $\boldsymbol{z} \in \mathcal{B}$, $\boldsymbol{z} = (\boldsymbol{0}, \boldsymbol{0}, s)$ with $s < p^*$, but also since $\boldsymbol{z} \in \mathcal{A}$, ther exists an $\boldsymbol{w} \in \mathcal{D}$ with $f_i(\boldsymbol{w}) \leq 0$, $\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b} = \boldsymbol{0}$, and $f(\boldsymbol{w}) \leq s < p^*$, in contradiction to the optimality of p^*.

By the separating hyperplane theorem, there exists a hyperplane separating $\mathcal{A}$ and $\mathcal{B}$ (but not necessarily strictly!), defined by a vector $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}, \nu) \neq \boldsymbol{0}$ and $\alpha \neq 0$ with the property that

$$(\boldsymbol{u}, \boldsymbol{v}, t) \in \mathcal{A} \Longrightarrow \tilde{\boldsymbol{\lambda}}^\intercal \boldsymbol{u} + \tilde{\boldsymbol{\mu}}^\intercal \boldsymbol{v} + \nu t \geq \alpha \tag{13.1}$$

and

$$(\boldsymbol{u}, \boldsymbol{v}, t) \in \mathcal{B} \Longrightarrow \tilde{\boldsymbol{\lambda}}^\intercal \boldsymbol{u} + \tilde{\boldsymbol{\mu}}^\intercal \boldsymbol{v} + \nu t \leq \alpha. \tag{13.2}$$

If $\tilde{\boldsymbol{\lambda}} < \boldsymbol{0}$ or $\nu < 0$, we could find values of $\boldsymbol{u}$ and t for which $(\boldsymbol{u}, \boldsymbol{v}, t) \in \mathcal{A}$, but that make the right-hand side of (13.1) arbitrary small, contradicting the bound by α. It follows that $\tilde{\boldsymbol{\lambda}} \geq \boldsymbol{0}$ and $\nu \geq 0$. Condition (13.2) simply means that $\nu t \leq \alpha$ for all $t < p^*$, so that $\nu p^* \leq \alpha$. Combining this bound with (13.1), we get the two inequalities, valid for any $\boldsymbol{w} \in \mathcal{D}$,

$$\sum_{i=1}^{m} \tilde{\lambda}_i f_i(\boldsymbol{w}) + \tilde{\boldsymbol{\mu}}^\intercal(\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}) + \nu f(\boldsymbol{w}) \geq \alpha \geq \nu p^*. \tag{13.3}$$

Note that the left-hand side has the form of a Lagrangian function scaled by ν. If $\nu > 0$ we can divide by ν and set $\lambda_i = \tilde{\lambda}_i/\nu$, $\mu_i = \tilde{\mu}_i/\nu$, to obtain

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \geq p^*.$$

By weak duality we have $p^* \geq g(\boldsymbol{\lambda}, \boldsymbol{\mu})$, so that we get strong duality if $\nu > 0$. If $\nu = 0$, then (13.3) implies

$$\sum_{i=1}^{m} \tilde{\lambda}_i f_i(\boldsymbol{w}) + \tilde{\boldsymbol{\mu}}^\top (\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}) \geq 0, \tag{13.4}$$

and for a point $\tilde{\boldsymbol{w}}$ satisfying the conditions $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}$ and $f_i(\boldsymbol{w}) < 0$ for $1 \leq i \leq m$, this means that $\tilde{\boldsymbol{\lambda}} = \boldsymbol{0}$. As $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}, \nu) \neq \boldsymbol{0}$, we have $\boldsymbol{\mu} \neq \boldsymbol{0}$. Since $\tilde{\boldsymbol{\mu}}^\top (\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}) \geq 0$ and $= 0$ for some $\tilde{\boldsymbol{w}}$ in the interior of $\mathcal{D}$, there must be a $\boldsymbol{w}$ in the interior such that $\tilde{\boldsymbol{\mu}}^\top (\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}) < 0$, in contradiction to (13.4) (unless $\tilde{\boldsymbol{\mu}}^\top \boldsymbol{A} = \boldsymbol{0}$, which would contradict the condition that $\boldsymbol{A}$ has maximal rank p). □

Example 13.3. Consider a linear programming problem of the form

$$\begin{aligned} \text{minimize} \quad & \langle \boldsymbol{c}, \boldsymbol{w} \rangle \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b} \\ & \boldsymbol{w} \geq \boldsymbol{0}. \end{aligned}$$

The inequality constraints are $-w_i \leq 0$, while the equality constraints are $\boldsymbol{a}_i^\top \boldsymbol{w} - b_i$. The Lagrangian has the form

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \langle \boldsymbol{c}, \boldsymbol{w} \rangle - \sum_{i=1}^{m} \lambda_i w_i + \sum_{j=1}^{p} \mu_j (\boldsymbol{a}_j^\top \boldsymbol{w} - b_j) \\ &= (\boldsymbol{c} - \boldsymbol{\lambda} + \boldsymbol{A}^\top \boldsymbol{\mu})^\top \boldsymbol{w} - \boldsymbol{b}^\top \boldsymbol{\mu}. \end{aligned}$$

The infimum over $\boldsymbol{w}$ of this function is $-\infty$ unless $\boldsymbol{c} - \boldsymbol{\lambda} + \boldsymbol{A}^\top \boldsymbol{\mu} = \boldsymbol{0}$. The Lagrange dual is therefore

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{cases} -\boldsymbol{\mu}^\top \boldsymbol{b} & \text{if } \boldsymbol{c} - \boldsymbol{\lambda} + \boldsymbol{A}^\top \boldsymbol{\mu} = \boldsymbol{0} \\ -\infty & \text{else.} \end{cases}$$

From the previous chapter we conclude that

$$\max\{-\boldsymbol{b}^\top \boldsymbol{\mu} \mid \boldsymbol{c} - \boldsymbol{\lambda} + \boldsymbol{A}^\top \boldsymbol{\mu} = \boldsymbol{0},\ \boldsymbol{\lambda} \geq \boldsymbol{0}\} \leq \min\{\boldsymbol{c}^\top \boldsymbol{w} \mid \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b},\ \boldsymbol{w} \geq \boldsymbol{0}\}.$$

Note that if we write $\boldsymbol{v} = -\boldsymbol{\mu}$ and $\boldsymbol{s} = \boldsymbol{\lambda}$, then we get the dual version of the linear programming problem we started out with, and in this case it is known that

$$\max_{\boldsymbol{\lambda} \geq \boldsymbol{0}, \boldsymbol{\mu}} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = p^*.$$

A more common way to express linear programming duality is as follows: the optimal value of

$$\text{maximize} \quad \langle \boldsymbol{b}, \boldsymbol{v} \rangle \quad \text{subject to} \quad \boldsymbol{A}^\top \boldsymbol{v} \leq \boldsymbol{c} \tag{D}$$

coincides with the optimal value of

$$\text{minimize} \quad \langle \boldsymbol{c}, \boldsymbol{w} \rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b},\ \boldsymbol{w} \geq \boldsymbol{0}, \tag{P}$$

provided both (D) and (P) have a finite solution (here, we set $\boldsymbol{v} = -\boldsymbol{\mu}$).

Example 13.4. Consider the problem

$$\text{minimize } \|\boldsymbol{w}\|^2 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}.$$

Note that $\|\boldsymbol{w}\|^2 = \boldsymbol{w}^\mathsf{T}\boldsymbol{w}$. The Lagrangian is $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\mu}) = \|\boldsymbol{w}\|^2 + \boldsymbol{\mu}^\mathsf{T}(\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b})$. For any $\boldsymbol{\mu}$, we can find the infimum

$$g(\boldsymbol{\mu}) = \inf_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\mu})$$

by setting the derivative of the Lagrangian to $\boldsymbol{w}$ to zero:

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\mu}) = 2\boldsymbol{w} + \boldsymbol{A}^\mathsf{T}\boldsymbol{\mu} = \boldsymbol{0},$$

which gives the solution

$$\boldsymbol{w} = -\frac{1}{2}\boldsymbol{A}^\mathsf{T}\boldsymbol{\mu}.$$

The dual function is therefore

$$g(\boldsymbol{\mu}) = -\frac{1}{4}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\mu}.$$

As the negative of a positive semidefinite quadratic function, it is concave. Moreover, we get the lower bound

$$-\frac{1}{4}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\mu} \leq \inf\{\|\boldsymbol{w}\|^2 \mid \boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}\}.$$

The problem we started out with is convex, and if we assume that there exists a feasible primal point, then the above inequality is in fact an equality by Slater's conditions.

Karush-Kuhn-Tucker optimality conditions

Consider now a not necessarily convex problem of the form

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{w}) \\ \text{subject to} \quad & \boldsymbol{f}(\boldsymbol{w}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{w}) = \boldsymbol{0}, \end{aligned} \tag{13.5}$$

If p^* is the optimal solution of (13.5) and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ dual variables, then we have seen that (this holds even in the non-convex case)

$$p^* \geq g(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

From this is follows that for any primal feasible point $\boldsymbol{w}$,

$$f(\boldsymbol{w}) - p^* \leq f(\boldsymbol{w}) - g(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

The difference $f(\boldsymbol{w}) - g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ between the primal objective function at a primal feasible point and the dual objective function at a dual feasible point is called the duality gap at $\boldsymbol{w}$ and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$. For any such points we know that

$$p^*, q^* \in [g(\boldsymbol{\lambda}, \boldsymbol{\mu}), f(\boldsymbol{w})],$$

and if the gap is small we have a good approximation of the primal and dual optimal values. The duality gap can be used in iterative algorithms to define stopping criteria: if the algorithm generates a sequence of primal-dual variables $(\boldsymbol{w}^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)$, then we can stop if the duality gap is less than, say, a predefined tolerance ε.

Now suppose that $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is such that the duality gap is zero. Then

$$\begin{aligned} f(\boldsymbol{w}^*) = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) &= \inf_{\boldsymbol{w}} \left(f(\boldsymbol{w}) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{w}) + \sum_{j=1}^{p} \mu_j^* h_j(\boldsymbol{w}) \right) \\ &\leq f(\boldsymbol{w}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{w}^*) + \sum_{j=1}^{p} \mu_j^* h_j(\boldsymbol{w}^*) \\ &\leq f(\boldsymbol{w}^*), \end{aligned}$$

where the last inequality follows from the fact that $h_j(\boldsymbol{w}^*) = 0$ and $\lambda_i^* f_i(\boldsymbol{w}^*) \leq 0$ for $1 \leq j \leq p$ and $1 \leq i \leq m$. It follows that the inequalities are in fact equalities. From the identity

$$f(\boldsymbol{w}^*) = f(\boldsymbol{w}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{w}^*)$$

and $\lambda_i^* \geq 0$ and $f_i(\boldsymbol{w}^*) \leq 0$ we also conclude that at such optimal points,

$$\lambda_i^* f_i(\boldsymbol{w}^*) = 0, \quad 1 \leq i \leq m.$$

This condition is known as complementary slackness. From the above we also see that $\boldsymbol{w}^*$ minimizes the Lagrangian $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, so that

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \mathbf{0}.$$

Collecting these conditions (primal and dual feasibility, complementary slackness, vanishing gradient), we arrive at a set of optimality conditions known as the Karush-Kuhn-Tucker (KKT) conditions.

Theorem 13.5. (KKT conditions) Let $\boldsymbol{w}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be primal and dual optimal solutions of (13.5) with zero duality gap. The the following conditions are satisfied:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{w}^*) &\leq \mathbf{0} \\ \boldsymbol{h}(\boldsymbol{w}^*) &= \mathbf{0} \\ \boldsymbol{\lambda}^* &\geq \mathbf{0} \\ \lambda_i^* f_i(\boldsymbol{w}^*) &= 0, \ 1 \leq i \leq m \\ \nabla_{\boldsymbol{w}} f(\boldsymbol{w}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla_{\boldsymbol{w}} f_i(\boldsymbol{w}^*) + \sum_{j=1}^{p} \mu_j^* \nabla_{\boldsymbol{w}} h_j(\boldsymbol{w}^*) &= \mathbf{0}. \end{aligned}$$

Moreover, if the problem is convex and the Slater Conditions (Theorem 13.1) are satisfied, then any points satisfying the KKT conditions have zero duality gap.

14

Support Vector Machines

In this lecture we return to the task of classification, and introduce a popular method for classification, Support Vector Machines (SVMs), from the point of view of convex optimization.

Linear Support Vector Machines

In the simplest case there is a set of labels $\mathcal{Y} = \{-1, 1\}$ and the set of training points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ is linearly separable: this means that there exists an affine hyperplane $h(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} + b$ such that $h(\boldsymbol{x}_i) > 0$ if $y_i = 1$ and $h(\boldsymbol{x}_j) < 0$ if $y_j = -1$. We call the points for which $y_i = 1$ positive, and the ones for which $y_j = -1$ negative. The problem of finding such a hyperplane can be posed as a linear programming feasibility problem as follows: we look for a vector of weights $\boldsymbol{w}$ and a bias term b (together a $(p+1)$-dimensional vector) such that

$$\boldsymbol{w}^\intercal \boldsymbol{x}_i + b \geq 1, \text{ for } y_i = 1, \quad \boldsymbol{w}^\intercal \boldsymbol{x}_j + b \leq -1, \text{ for } y_j = -1.$$

Note that we can replace the $+1$ and -1 with any other positve or negative quantity by rescaling the $\boldsymbol{w}$ and $\boldsymbol{b}$, so this is just convention. We can also describe the two inequalities concisely as

$$y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) - 1 \geq 0. \tag{14.1}$$

A hyperplane separating the two point sets will in general not be unique. As we want to use the linear classifier on new, yet unknown data, we want to find a separating hyperplane with best possible margin. Let δ_+ and δ_- denote the distance of a separating hyperplane to the closest positive and closest negative point, respectively. The quantity $\delta = \delta_+ + \delta_-$ is then called the margin of the classifier, and we want to find a hyperplane with largest possible margin.

We next show that the margin for a separating hyperplane that satisfies (14.1) is $\delta = 2/\|\boldsymbol{w}\|_2$. Given a hyperplane H described in (14.1) and a point $\boldsymbol{x}$ such that we have the equality $\boldsymbol{w}^\intercal \boldsymbol{x} + b = 1$ (the point is as close as possible to the hyperplane, also called a support vector), the distance of that point to the hyperplane can be computed by first taking the difference of $\boldsymbol{x}$ with a point $\boldsymbol{p}$ on H (an anchor), and then computing the dot product of $\boldsymbol{x} - \boldsymbol{p}$ with the unit vector $\boldsymbol{w}/\|\boldsymbol{w}\|$ normal to H.

As anchor point $\boldsymbol{p}$ we can just choose a multiple $c\boldsymbol{w}$ that is on the plane, i.e., that satisfies $\langle \boldsymbol{w}, c\boldsymbol{w} \rangle + b = 0$. This implies that $c = -b/\|\boldsymbol{w}\|^2$, and consequently $\boldsymbol{p} = -(b/\|\boldsymbol{w}\|^2)\boldsymbol{w}$. The distance is then

$$\begin{aligned}\delta_+ &= \langle \boldsymbol{x} + \frac{b}{\|\boldsymbol{w}\|^2}\boldsymbol{w}, \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \rangle = \frac{\langle \boldsymbol{x}, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|} + \frac{b}{\|\boldsymbol{w}\|^2}\langle \boldsymbol{w}, \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \rangle \\ &= \frac{1-b}{\|\boldsymbol{w}\|} + \frac{b}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}.\end{aligned}$$

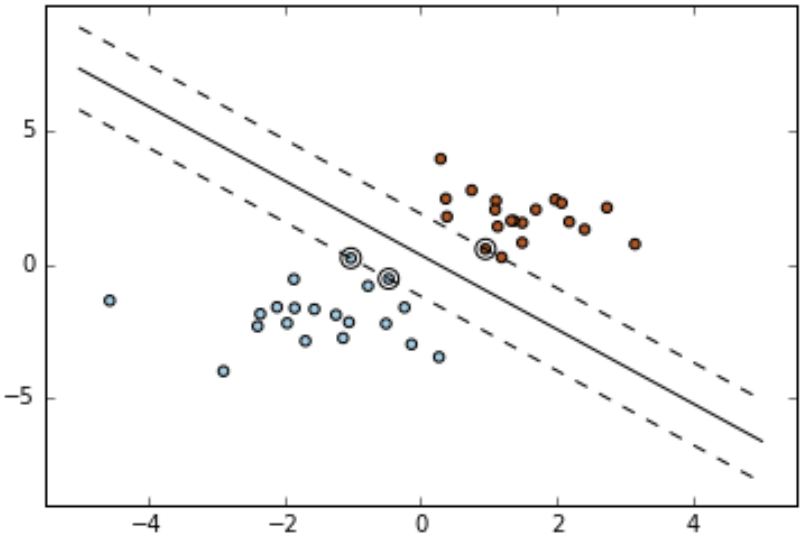


Figure 14.1: A hyperplane separating two sets of points with margin and support vectors.

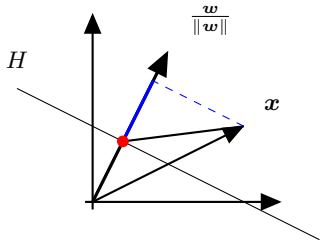


Figure 14.2: Computing the distance to the hyperplane

Similarly, we get $\delta_- = 1/\|\boldsymbol{w}\|$. The margin of this particular separating hyperplane is thus $\delta = 2/\|\boldsymbol{w}\|$. If we want to find a hyperplane with largest margin, we thus have to solve the quadratic optimization problem

$$\begin{aligned} \underset{\boldsymbol{w},b}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) - 1 \geq 0, \quad 1 \leq i \leq n. \end{aligned}$$

Note that b is also an unknown variable in this problem! The factor $1/2$ in the objective function is just to make the gradient look nicer. The Lagrangian of this problem is

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \lambda_i y_i \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i - \lambda_i y_i b + \lambda_i \\ &= \frac{1}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w} - \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{X}\boldsymbol{w} - b\boldsymbol{\lambda}^\mathsf{T}\boldsymbol{y} + \sum_{i=1}^{m} \lambda_i, \end{aligned}$$

where we denote by $\boldsymbol{X}$ the matrix with the $y_i \boldsymbol{x}_i^\intercal$ as rows. We can then write the conditions on the gradient with respect to $\boldsymbol{w}$ and b of the Lagrangian as

$$\begin{aligned} \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) &= \boldsymbol{w} - \boldsymbol{X}^\intercal \boldsymbol{\lambda} = \boldsymbol{0} \\ \frac{\partial \mathcal{L}}{\partial b}(\boldsymbol{w}, b, \boldsymbol{\lambda}) &= \boldsymbol{y}^\intercal \boldsymbol{\lambda} = 0. \end{aligned} \tag{14.2}$$

If $\boldsymbol{y}^\intercal \boldsymbol{\lambda} \neq 0$, then the conditions (14.2) cannot be satisfied and the problem is unbounded from below. If $\boldsymbol{y}^\intercal \boldsymbol{\lambda} = 0$, then the first condition in (14.2) is necessary and sufficient for a minimizer. Replacing $\boldsymbol{w}$ by $\boldsymbol{X}^\intercal \boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^\intercal \boldsymbol{y}$ by 0 in the Lagrangian function then gives the expression for the Lagrange dual $g(\boldsymbol{\lambda})$,

$$g(\boldsymbol{\lambda}) = \begin{cases} -\frac{1}{2} \boldsymbol{\lambda}^\intercal \boldsymbol{X} \boldsymbol{X}^\intercal \boldsymbol{\lambda} + \sum_{i=1}^m \lambda_i & \text{if } \boldsymbol{y}^\intercal \boldsymbol{\lambda} = 0 \\ -\infty & \text{else.} \end{cases}$$

Finally, maximizing this function and changing the sign, so that the maximum becomes a minimum, we can formulate the Lagrange dual optimization problem as

$$\underset{\boldsymbol{\lambda}}{\text{minimize}} \quad \frac{1}{2} \boldsymbol{\lambda}^\intercal \boldsymbol{X} \boldsymbol{X}^\intercal \boldsymbol{\lambda} - \boldsymbol{\lambda}^\intercal \boldsymbol{e} \quad \text{subject to} \quad \boldsymbol{\lambda} \geq \boldsymbol{0}, \tag{14.3}$$

where $\boldsymbol{e}$ is the vector of all ones.

Assuming that the data is linearly separable, the primal and dual optimization problems for finding the parameters of a separating hyperplane of maximal margin can therefore be formulated as

$$\text{minimize} \, \frac{1}{2} \|\boldsymbol{w}\|^2 \quad \text{subject to} \quad \boldsymbol{e} - b\boldsymbol{y} - \boldsymbol{X}^\intercal \boldsymbol{x} \leq \boldsymbol{0}, \tag{P}$$

where $\boldsymbol{e}$ is the vector of all ones. The Lagrange dual optimization problem is

$$\text{minimize} \, \frac{1}{2} \boldsymbol{\lambda}^\intercal \boldsymbol{X} \boldsymbol{X}^\intercal \boldsymbol{\lambda} - \boldsymbol{\lambda}^\intercal \boldsymbol{e} \quad \text{subject to} \quad \boldsymbol{\lambda} \geq \boldsymbol{0}. \tag{D}$$

Note that there is one dual variable λ_i per data point $\boldsymbol{x}_i$. We can find the optimal value by solving the dual problem (D), but that does not give us automatically the weights $\boldsymbol{w}$ and the bias b. We can find the weights by $\boldsymbol{w} = \boldsymbol{X}^\intercal \boldsymbol{\lambda}$. As for b, this is best determined from the KKT conditions of the problem. These can be written by combining the constraints of the primal problem with the conditions on the gradient of the Lagrangian, the condition $\boldsymbol{\lambda} \geq \boldsymbol{0}$, and complementary slackness as

$$\begin{aligned} \boldsymbol{X}\boldsymbol{w} + b\boldsymbol{y} - \boldsymbol{e} &\geq \boldsymbol{0} \\ \boldsymbol{\lambda} &\geq \boldsymbol{0} \\ \lambda_i (1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)) &= 0 \text{ for } 1 \leq i \leq n \\ \boldsymbol{w} - \boldsymbol{X}^\intercal \boldsymbol{\lambda} &= \boldsymbol{0} \\ \boldsymbol{y}^\intercal \boldsymbol{\lambda} &= 0. \end{aligned}$$

To get b, we can choose one of the equations in which $\lambda_i \neq 0$, and then find b by setting $b = y_i(1 - y_i \boldsymbol{w}^\intercal \boldsymbol{x}_i)$. With the KKT conditions written down, we can go about solving the problem of finding a maximum margin linear classifier using standard optimization methods.

Extensions

So far we looked at the particularly simple case where (a) the data falls into two classes, (b) the points can actually be well separated, and (c) they can be separated by an affine hyperplane. In reality, these three assumptions may not hold. We briefly discuss extensions of the basic model to account for the three situations just mentioned.

Non-exact separation

What happens when the data can not be separated by a hyperplane? In this case the constraints can not be satisfied: there is no feasible solution to the problem. We can still modify the problem to allow for misclassification: we want to find a hyperplane that separates the two point sets as good as possible, but we allow for some mistakes.

One approach is to add an additional set of n slack variables $s_1, \ldots, s_n$, and modify the constraints to

$$\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b \geq 1 - s_i, \text{ for } y_i = 1, \quad \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_j + b \leq -1 + s_j, \text{ for } y_j = -1, \quad s_i \geq 0.$$

The i-th data point can land on the wrong side of the hyperplane if $s_i > 1$, and consequently the sum $\sum_{i=1}^n s_i$ is an upper bound on the number of errors possible. If we want to minimize the number of misclassified points, we may want to minimize this upper bound, so a sensible choice for objective function would be to add a multiple of this sum. The new problem thus becomes

$$\begin{aligned} \text{minimize} & \frac{1}{2}\|\boldsymbol{w}\|^2 + \mu \sum_{j=1}^{n} s_j \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) - 1 + s_i \geq 0, \quad 1 \leq i \leq n \\ & s_i \geq 0, \quad 1 \leq i \leq n, \end{aligned}$$

for some parameter μ. The Lagrangian of this problem and the KKT conditions can be derived in a similar way as in the separable case and are left as an exercise.

Non-linear separation and kernels

The key to extending SVMs from linear to non-linear separation is the observation that the dual form of the optimization problem (D) depends only on the dot products $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ of the data points. In fact, the (i, j)-th entry of the matrix $\boldsymbol{X}\boldsymbol{X}^\mathsf{T}$ is precisely $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$!

If we map our data into a higher (possibly infinite) dimensional space $\mathcal{H}$,

$$\varphi \colon \mathbb{R}^d \to \mathcal{H},$$

and consider the data points $\varphi(\boldsymbol{x}_i)$, $1 \leq i \leq n$, then applying the support vector machine to these higher dimensional vectors will only depend on the dot products

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{x}_j) \rangle.$$

The function K is called a kernel function. A typical example, often used in practice, is the Gaussian radial basis function (RBF),

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/2\sigma^2}.$$

Note that we don't need to know how the function φ looks like! In the equation for the hyperplane we simply replace $\boldsymbol{w}^\mathsf{T}\boldsymbol{x}$ with $K(\boldsymbol{w}, \boldsymbol{x})$. The only difference now is that the function ceases to be linear in $\boldsymbol{x}$: we get a non-linear decision boundary.

Multiple classes

One is often interested in classifying data into more than two classes. There are two simple ways in which support vector machines can be extended for such problems: one-vs-one and one-vs-rest. In the one-vs-one case, given k classes, we train one classifier for each pair of classes in the training data,

obtaining a total of $k(k-1)/2$ classifiers. When it comes to prediction, we apply each of the classifiers to our test data and choose the class that was chosen the most among all the classifiers. In the one-vs-rest approach, each train k binary classifiers: in each one, one class corresponds to a chosen class, and the second class corresponds to the rest. By associating confidence scores to each classifier, we choose the one with the highest confidence score.

Example 14.1. An example that uses all three extensions mentioned is handwritten digit recognition. Suppose we have a series of pixels, each representing a number, and associated labels $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. We would like to train a support vector machine to recognize new digits. Given the knowledge we have, we can implement this task using standard optimization software such as CVXPY. Luckily, there are packages that have this functionality already implemented, such as the Scikit-learn package for Python. We illustrate its functioning below. The code also illustrates some standard procedures when tackling a machine learning problem:

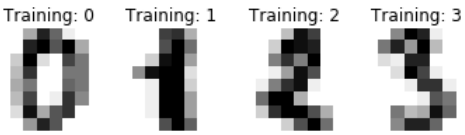
- Separate the data set randomly into training data and test data;
- Create a support vector classifier with optional parameters;
- Train (using fit) the classifier with the training data;
- Predict the response using the test data and compare with the true response;
- Report the results.

An important aspect to keep in mind is that when testing the performance using the test data, we should compare the classification accuracy to a naive baseline: if, for example, 80% of the test data is classified as $+1$, then making a prediction of $+1$ for all the data will give us an accuracy of 80%; in this case, we would want our classifier to perform considerably better than getting the right answer 80% of the time!

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
% matplotlib inline
from sklearn import svm, datasets, metrics
from sklearn.model_selection import train_test_split
```

In [13]:
```
digits = datasets.load_digits()

# Display images and labels
images_and_labels = list(zip(digits.images, digits.target))
for index, (image, label) in enumerate(images_and_labels[:4]):
    plt.subplot(2, 4, index + 1)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    plt.title('Training: %i' % label)

# Turn images into 1-D arrays
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create classifier
svc = svm.SVC(gamma=0.001)

# Randomly split data into train and test set
X_train, X_test, y_train, y_test = train_test_split(data,
 digits.target, test_size = 0.4, random_state=0)
svc.fit(X_train, y_train)
```

Out [2]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.001,
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```



Now apply prediction to test set and report performance.

In [3]:
```
predicted = svc.predict(X_test)
print("Classification report for classifier %s:\n%s\n"
      % (svc, metrics.classification_report(y_test, predicted)))
```

Out [3]:
```
Classification report for classifier SVC(C=1.0, cache_size=200,
 class_weight=None, coef0=0.0,vdecision_function_shape=None,
 degree=3, gamma=0.001, kernel='rbf', max_iter=-1, probability=False,
 random_state=None, shrinking=True, tol=0.001, verbose=False):
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        60
          1       0.97      1.00      0.99        73
          2       1.00      0.97      0.99        71
          3       1.00      1.00      1.00        70
          4       1.00      1.00      1.00        63
          5       1.00      0.98      0.99        89
          6       0.99      1.00      0.99        76
          7       0.98      1.00      0.99        65
          8       1.00      0.99      0.99        78
          9       0.99      1.00      0.99        74

avg / total       0.99      0.99      0.99       719
```

In [4]:
```
import skimage
from skimage import data
from skimage.transform import resize
from skimage import io
import os
```

Now try this out on some original data!

In [5]:

```
mydigit1 = io.imread('images/digit9.png')
mydigit2 = io.imread('images/digit4.png')
plt.figure(figsize=(8, 4))
plt.subplot(1,2,1)
plt.imshow(mydigit1, cmap=plt.cm.gray_r, interpolation='nearest')
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(mydigit2, cmap=plt.cm.gray_r, interpolation='nearest')
plt.axis('off')
plt.show()
```

In [6]:

```
smalldigit1 = resize(mydigit1, (8,8))
smalldigit2 = resize(mydigit2, (8,8))
mydigits = np.concatenate((np.round(15*(np.ones((8,8))-
                        smalldigit1[:,:,0])).reshape((64,1)).T,
            np.round(15*(np.ones((8,8))-
            smalldigit2[:,:,0])).reshape((64,1)).T),axis=0)
# After some preprocessing, make prediction
guess = svc.predict(mydigits)
print guess
```

```
[9 4]
```

15

Neural Networks

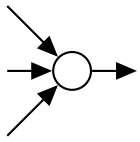
Originally introduced as simplified models of neurons in the brain, (artificial) neural networks are now one of the most widely used models in machine learning and data science. Their popularity owes to their ability to combine generality with computational tractability: while a neural network can approximate virtually any reasonable function to arbitrary accuracy, its structure is still simple enough so that it can be trained efficiently by gradient descent.

Connectivity and Activation

We begin by revisiting the problem of binary classification using a linear function. Given a vector of weights $\boldsymbol{w} \in \mathbb{R}^d$ and a bias term $b \in \mathbb{R}$, define the binary classifier

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = \mathbf{1}\{\boldsymbol{w}^\intercal \boldsymbol{x} + b > 0\} = \begin{cases} 1 & \boldsymbol{w}^\intercal \boldsymbol{x} + b > 0 \\ 0 & \boldsymbol{w}^\intercal \boldsymbol{x} + b \leq 0 \end{cases}$$

We already encountered this problem when studying linear support vector machines. Visually, we can represent this classifier by a node that takes d inputs $(x_1, \ldots, x_d)$, and outputs 0 or 1:



Such a unit is called a perceptron. The node represents a neuron that fires if a certain linear combination of inputs, $\boldsymbol{w}^\intercal \boldsymbol{x}$, exceeds a threshold $-b$. In order to determine the weights and bias term from data using optimization, it is useful to approximate the indicator with a smooth function such as the sigmoid

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}}.$$

We can then replace the function $h_{\boldsymbol{w},b}$ with the smooth function $g(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + b)$. A convenient property of the sigmoid function is that the derivative has the form

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)),$$

so that the gradient of $\boldsymbol{g}$ can be computed as

$$\nabla g(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + b)(1 - \sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + b)) \cdot \boldsymbol{w}.$$

If we drop the requirement that the activator function should approximate the indicator function, then other popular activation functions are the hyperbolic tangent, $\tanh(x)$ (which maps into $[-1, 1]$), and the rectifiable linear unit (ReLU), $\max\{x, 0\}$ (which maps into $[0, \infty)$ and is not differentiable at 0). Figure 15.1 illustrates these different activations functions.

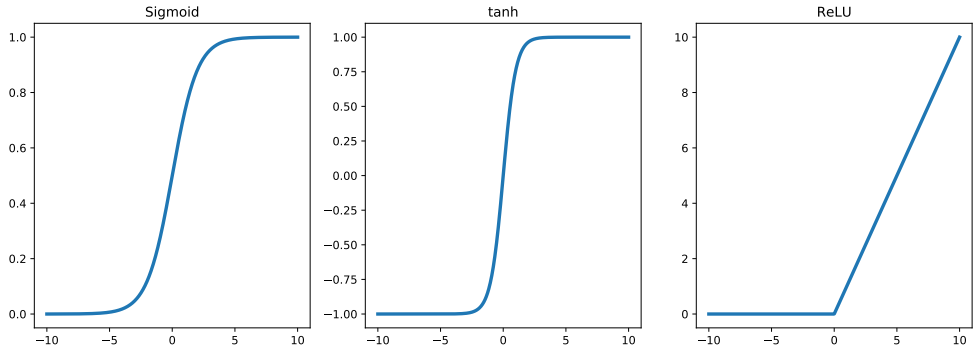


Figure 15.1: Activation functions

A feedforward neural network arises by combining various perceptrons by feeding the outputs of a series of perceptrons into new perceptrons, see Figure 15.2.

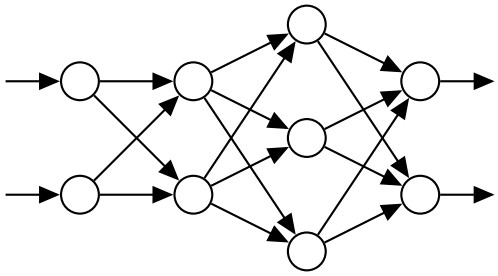


Figure 15.2: A fully connected neural network.

We interpret a neural network as consisting of different layers. To the k-th layer we associated a linear map $\boldsymbol{W}^k \colon \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ and a bias vector $\boldsymbol{b}^k \in \mathbb{R}^{d_k}$. Applying the activation function componentwise, we obtain a map

$$\sigma(\boldsymbol{W}^k \boldsymbol{x} + \boldsymbol{b}^k).$$

The first layer is the input layer, while the last layer is the output layer. Hence, a neural network with ℓ layers is a function of the form $F^\ell(\boldsymbol{x})$, where the F^k are recursively defined as

$$\begin{aligned} F^1(\boldsymbol{x}) &= \sigma(\boldsymbol{W}^1 \boldsymbol{x} + \boldsymbol{b}^1) \\ F^{k+1}(\boldsymbol{x}) &= \sigma(\boldsymbol{W}^{(k+1)} F^k(\boldsymbol{x}) + \boldsymbol{b}^{k+1}), \quad 1 \leq k < \ell, \end{aligned}$$

and $\boldsymbol{W}^k \in \mathbb{R}^{d_k \times d_{k-1}}$, $\boldsymbol{b}^k \in \mathbb{R}^{d_k}$ for $1 \leq k \leq \ell$ (with $d_0 = d$). The layers between the input and output layer are called the hidden layers. If we fix the format, that is, the vector $\boldsymbol{d} = (d_0, d_1, \ldots, d_\ell)$, where d_i represents the number of nodes in each layer, we get a hypothesis class

$$\mathcal{H}_{\boldsymbol{d}} = \{F \colon \mathbb{R}^d \to \mathbb{R}^{d_\ell} \colon F \text{ is a NN with format } \boldsymbol{d}\}.$$

A neural network $F \in \mathcal{H}_{\boldsymbol{d}}$ can be used to approximate a function $\mathbb{R}^d \to \mathbb{R}^{d_\ell}$, or it can be used for classification tasks, for example by assigning an input $\boldsymbol{x}$ to the class represented by the index of the largest

coordinate of the output, $\arg\max_i F_i(\boldsymbol{x})$. Sometimes, the output is processed through an additional function.

If every node connects to a node in the next layer, then the network is called fully connected. We can place further restrictions on the form of individual linear maps at each layer (for example, by requiring them to perform a convolution on subsets of the input) and use different activation functions in each layer. Such a set of specifications defines an architecture.

Example 15.1. An elementary function that can be realized by a neural network is the exclusive or, or XOR function, which can be interpreted as addition in $\mathbb{Z}/2\mathbb{Z}$:

$$\mathrm{XOR}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = (1,0) \text{ or } \boldsymbol{x} = (0,1), \\ 0 & \text{if } \boldsymbol{x} = (0,0) \text{ or } \boldsymbol{x} = (1,1) \end{cases}$$

Thinking of the possible inputs as the corners of the unit square $[0,1]^2$ in $\mathbb{R}^2$, one easily verifies that the XOR function cannot be realized by a linear separator: there is no linear function h such that $h(\boldsymbol{x}) > 0$ if $\mathrm{XOR}(\boldsymbol{x}) = 1$ and $h(\boldsymbol{x}) < 0$ if $\mathrm{XOR}(\boldsymbol{x}) = 0$, for $\boldsymbol{x} \in \{0,1\}^2$. One can, however, implement XOR using a fully-connected neural network as

$$\mathrm{XOR}(\boldsymbol{x}) := \boldsymbol{w}^\intercal([\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}]_+),$$

where

$$\boldsymbol{U} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \qquad \boldsymbol{w} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}.$$

and $[x]_+ = \max\{x, 0\}$ is the ReLU activation function. One easily verifies that $\mathrm{XOR}(\boldsymbol{x}) = 1$ if exactly one of the coordinates of $\boldsymbol{x}$ is 1, and $\mathrm{XOR}(\boldsymbol{x}) = 0$ else.

We can train a neural network on data $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, $i \in \{1, \ldots, n\}$, by minimizing the empirical risk with respect to our favourite (or most suitable) loss function L. A common loss function is the square loss,

$$L(F(\boldsymbol{x}), \boldsymbol{y}) = \frac{1}{2}\|F(\boldsymbol{x}) - \boldsymbol{y}\|^2,$$

where we use the ℓ_2-norm (or Euclidean norm). If the neural network has only one layer and no activation function, then training it with respect to the square loss amounts to solving a least squares problem. Another common option that is used for classification problems with k classes is the cross-entropy, or negative log-likelihood. We assume that the training data is given by input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y} \in \mathbb{R}^k$ is a vector with $y_j = 1$ if $\boldsymbol{x}$ belongs to class j, and 0 else. The output of a neural network for this classification problem is then often a probability distribution, $F(\boldsymbol{x}) = (p_1, \ldots, p_k)$, where p_j is the probability that $\boldsymbol{x}$ belongs to class j. The cross-entropy is defined as

$$L(F(\boldsymbol{x}), y) = -\sum_k \left(y_k \log(p_k) + (1 - y_k)\log(1 - p_k)\right).$$

If the output consists of a vector $F(\boldsymbol{x}) = (v_1, \ldots, v_k)$ instead of a probability distribution, it is often transformed into a probability distribution by setting

$$p_i = \frac{\mathrm{e}^{v_i}}{\sum_j \mathrm{e}^{v_j}}.$$

For the rest of this chapter we will not make use of the specific form of the loss function, and we will get back to it when discussing information theory and applications.

Denote by $\boldsymbol{W}$ and $\boldsymbol{b}$ the concatenation of all the weight matrices and bias vectors. We denote by w_{ij}^k the (i,j)-th entry of the k-th matrix, and by b_i^k the i-th entry of the k-th bias vector. The task is then to minimize the empirical risk

$$f(\boldsymbol{W}, \boldsymbol{b}) := \frac{1}{n} \sum_{i=1}^{n} L(F^\ell(\boldsymbol{x}_i), \boldsymbol{y}_i).$$

If L is differentiable almost everywhere, then the method of choice is gradient descent, using a computational implementation of the chain rule known as backpropagation.

Backpropagation

Gradient descent, or stochastic gradient descent, requires evaluating the gradient of a function of the form

$$f_i(\boldsymbol{W}, \boldsymbol{b}) := L(F^\ell(\boldsymbol{x}_i), \boldsymbol{y}_i).$$

In what follows, set $\boldsymbol{x} = \boldsymbol{x}_i$ and $\boldsymbol{y} = \boldsymbol{y}_i$. Also write $\boldsymbol{a}^0 := \boldsymbol{x}$, and for $k \in \{1, \dots, \ell\}$,

$$\boldsymbol{z}^k = \boldsymbol{W}^k \boldsymbol{a}^{k-1} + \boldsymbol{b}^k, \quad \boldsymbol{a}^k = \sigma(\boldsymbol{z}^k). \tag{15.1}$$

In particular, $\boldsymbol{a}^\ell = F^\ell(\boldsymbol{x})$ is the output of the neural network on input $\boldsymbol{x}$. Moreover, set $C = C(\boldsymbol{W}, \boldsymbol{b}) = L(\boldsymbol{a}^\ell, \boldsymbol{y})$ for the loss function.

For every layer k and coordinate $j \in \{1, \dots, d_k\}$, define the sensitivities

$$\delta_j^k := \frac{\partial C}{\partial z_j^k},$$

where z_j^k is the j-th coordinate of $\boldsymbol{z}^k$. Thus δ_j^k measures the sensitivity of the loss function to the input at the j-th node of the k-th layer. Denote by $\boldsymbol{\delta}^k \in \mathbb{R}^{d_k}$ the vector of δ_j^k for $j \in \{1, \dots, d_k\}$. The partial derivatives of C can be computed in terms of these quantities. In what follows, we denote by $\boldsymbol{x} \circ \boldsymbol{y}$ the componentwise product, that is, the vector with entries $x_i y_i$.

Proposition 15.2. For a neural network with ℓ layers and $k \in \{1, \dots, \ell\}$, we have

$$\frac{\partial C}{\partial w_{ij}^k} = \delta_i^k a_j^{k-1}, \qquad \frac{\partial C}{\partial b_i^k} = \delta_i^k \tag{15.2}$$

for $i, j \in \{1, \dots, d_k\}$. Moreover, the sensitivities δ_i^k can be computed as follows:

$$\boldsymbol{\delta}^\ell = \sigma'(\boldsymbol{z}^\ell) \circ \nabla_{\boldsymbol{a}^\ell} L(\boldsymbol{a}^\ell, \boldsymbol{y}), \qquad \boldsymbol{\delta}^k = \sigma'(\boldsymbol{z}^k) \circ \left(\boldsymbol{W}^{k+1}\right)^\intercal \boldsymbol{\delta}^{k+1} \tag{15.3}$$

for $k \in \{1, \dots, \ell - 1\}$.

Proof. We begin by showing (15.3). For $\boldsymbol{\delta}^\ell$, note that by the chain rule, we have

$$\delta_i^\ell = \frac{\partial L(\boldsymbol{a}^\ell, \boldsymbol{y})}{\partial z_i^\ell} = \sum_{j=1}^{d_\ell} \frac{\partial L(\boldsymbol{a}^\ell, \boldsymbol{y})}{\partial a_j^\ell} \frac{\partial a_j^\ell}{\partial z_i^\ell} = \frac{\partial L(\boldsymbol{a}^\ell, \boldsymbol{y})}{\partial a_i^\ell} \cdot \sigma'(z_i^\ell).$$

For $k < \ell$, we compute δ_i^k in terms of the δ_j^{k+1} as follows:

$$\delta_i^k = \frac{\partial C}{\partial z_i^k} = \sum_{j=1}^{d_{k+1}} \frac{\partial C}{\partial z_j^{k+1}} \frac{\partial z_j^{k+1}}{\partial z_i^k} = \sum_{j=1}^{d_{k+1}} \delta_j^{k+1} \cdot \frac{\partial z_j^{k+1}}{\partial z_i^k}.$$

For the summands in the last expression we use

$$z_j^{k+1} = \sum_{s=1}^{d_k} w_{js}^{k+1} \sigma(z_s^k) + b_j^{k+1},$$

so that the derivatives evaluate to

$$\frac{\partial z_j^{k+1}}{\partial z_i^k} = w_{ji}^{k+1} \cdot \sigma'(z_i^k).$$

Putting everything together, we arrive at

$$\delta_i^k = \sum_{j=1}^{d_{k+1}} \delta_j^{k+1} \cdot w_{ji}^{k+1} \cdot \sigma'(z_i^k) = \sigma'(z_i^k) \cdot \left((\boldsymbol{W}^{k+1})^\intercal \boldsymbol{\delta}^{k+1} \right)_i .$$

The expressions for the partial derivatives of C with respect to the weights $\boldsymbol{W}$ and the bias $\boldsymbol{b}$ are computed in a straight-forward way using the chain rule. More precisely, at the k-th layer write

$$z_i^k = \sum_{j=1}^{d_{k-1}} w_{ij}^k a_j^{k-1} + b_i^k.$$

The claimed expressions for the derivatives then follow by applying the chain rule,

$$\frac{\partial C}{\partial w_{ij}^k} = \frac{\partial C}{\partial z_i^k} \frac{\partial z_i^k}{\partial w_{ij}^k} = \delta_i^k \cdot a_j^{k-1},$$

and similarly for the derivative with respect to b_i^k. □

For the common quadratic loss function

$$L(\boldsymbol{a}^\ell, \boldsymbol{y}) = \frac{1}{2} \|\boldsymbol{a}^\ell - \boldsymbol{y}\|^2,$$

we get

$$\nabla_{\boldsymbol{a}^\ell} L(\boldsymbol{a}^\ell, \boldsymbol{y}) = \boldsymbol{a}^\ell - \boldsymbol{y},$$

which can be computed easily from the function value $F^\ell(\boldsymbol{x})$ and $\boldsymbol{y}$. Other differentiable loss functions may lead to different terms. Given initial weights $\boldsymbol{W}$ and bias terms $\boldsymbol{b}$, we can compute the values $\boldsymbol{a}^k$ and $\boldsymbol{z}^k$ using a forward pass, that is, by applying (15.1) recursively. We can then compute the sensitivities δ_j^k using (15.3), and the partial derivatives of the loss function using (15.2), starting at layer ℓ. This way of computing the gradients is called backpropagation. We note that choosing the sigmoid σ as activation function, the computation of the derivative $\sigma'(x)$ is easy. The whole process of computing the gradient of a neural network thus reduces to a simple sequence of matrix-vector product operations and sigmoids.

Example 15.3. Consider the following setting with $n = 10$ points. We train a neural network with four layers and format $\boldsymbol{d} = (d_0, d_1, d_2, d_3) = (2, 2, 3, 2)$ using stochastic gradient descent. The neural network outputs a vector in $\mathbb{R}^2$ and classifies an input point according to whether the first coordinate is greater or smaller than the second coordinate. Figure 15.3 shows the decision boundary by the neural network based on 10 training points, and the display on the right shows the error per iteration of stochastic gradient descent.

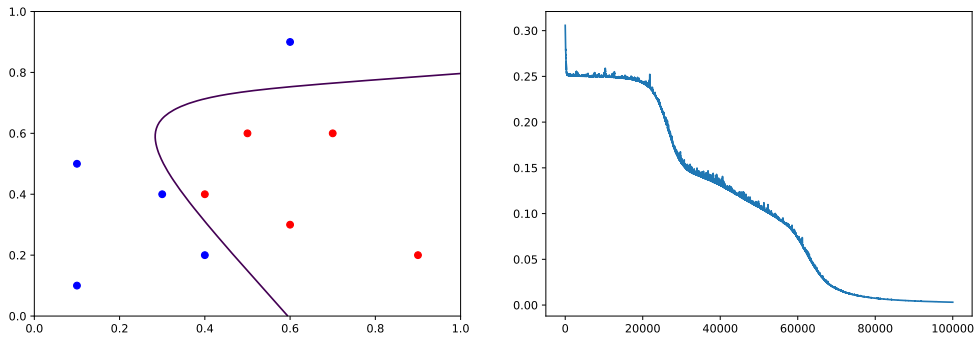


Figure 15.3: Training a neural network for classification and the error of stochastic gradient descent.

Part II

Approximation Theory

16

Intro to Approximation Theory

> "All exact science is dominated by the idea of approximation."
>
> —Bertrand Russell

Approximation theory of functions, a fundamental branch of mathematical analysis, traces its origins back to the pioneering work of Chebyshev in the 19th century. In 1859, Chebyshev introduced the concept of polynomial approximation, aiming to find the best possible polynomial approximation to a given function on a specific interval. Building upon Chebyshev's ideas, Weierstrass further advanced the theory in the late 19th century by establishing the famous Weierstrass approximation theorem. This theorem demonstrated that every continuous function on a closed interval can be uniformly approximated by polynomials, thereby providing a significant result in the mathematical understanding of approximation.

In addition to its historical significance, the development of approximation theory by Chebyshev, Weierstrass, and Bernstein laid the foundations for a multitude of modern applications and techniques. One notable advancement that emerged from these early contributions is the Fast Fourier Transform (FFT). The FFT, developed by Cooley and Tukey in the mid-20th century, utilises polynomial approximation techniques to efficiently compute the discrete Fourier transform, revolutionising signal processing, data compression, and various areas of scientific computing.

Moreover, approximation theory plays a crucial role in the field of wavelet analysis. Wavelets, which are functions that can capture localised features of a signal, rely on the approximation of signals using wavelet bases. This mathematical framework, rooted in approximation theory, enables applications such as image processing, data compression, and denoising.

The second part of this module will provide an introduction to approximation theory. Starting with the classical results by Chebyshev and Weierstrass, then continuing with more recent applications, for example the Fast Fourier Transform (FFT) or wavelets.

Although ideas of approximation theory can be traced back to Euler, Chebyshev was the first one to study the problem thoroughly. In particular Chebyshev (Russian mathematician, 1821-1894) posed the following problem in 1853:

Given a continuous function f defined on a closed interval $[a, b]$ and a positive integer n, can we approximate f by a polynomial $p(x) = \sum_{k=0}^{n} a_k x^k$, in such a way that the error at any point x in $[a, b]$ is

controlled? In particular, is it possible to construct p such that the error $\max_{a\leq x\leq b}|f(x)-p(x)|$ is minimised?

This problem raises several questions, in particular

1. Does such a polynomial p exist?
2. If it exists, is it unique and how can we construct it?
3. What happens, if we change the measure of error? For example if we consider $\int_a^b |f(x)-p(x)|^2\,\mathrm{d}x$ instead.

Chebyshev ignored the first question, but made significant contributions to advance on the second question. Before taking a closer look at Chebyshev's problem, we discuss some examples of approximation problems in the data sciences.

Approximation problems arise in various areas of data science, serving as the foundation for numerous techniques and algorithms. Here is a not comprehensive list of common approximation problems encountered in data sciences:

- Linear regression: We discussed linear regression already in the Chapter ??. We recall that we wish to understand the relation of y and a series of x_i, $i=1,\dots n$. In linear regression one assumes that the relation between these quantities linear, that is

$$y=\beta_0+\beta_1x_1+\beta_2x_2+\dots\beta_nx_n+\varepsilon \tag{16.1}$$

  where $\varepsilon>0$ is the noise level. In the context of approximation theory, this means that we choose a linear polynomial to approximate an unknown function.
  In nonlinear regression the relation between y and the x_i, $i=1\dots n$ is assumed to be nonlinear. For further information see
- Fourier Transform: The Fourier transform approximates f by a sum of trigonometric functions. This approach is widely used in signal processing, image analysis, ... and is particularly useful if the unknown function f is periodic. We will discuss the Fourier Transform in Chapter 20
- Wavelet Transform: The wavelet transform approximates a signal using so called wavelets. Wavelet functions capture features of signals on different levels, making it an important tool in image compression, denoising, and feature extraction. We discuss the wavelet transform in Chapter 21.

Interpolation vs Approximation theory: We conclude by discussing the difference between interpolation theory and approximation theory.interpolation is primarily concerned with finding a function that passes exactly through a given set of data points. In other words, it aims to find a polynomial or another mathematical function that matches the exact values of the function at specific points. The primary goal is to achieve a perfect fit to the data. Approximation theory, on the other hand, is more concerned with finding a function that closely approximates another function over a specified range or interval, rather than fitting specific data points exactly. It's often used when an exact match isn't possible or practical.
In this module we will focus on approximation theory, but many of the presented methods can be used for interpolation problems as well.

16.1 Normed Function Spaces

Before delving into the content of this course, it is essential to establish some necessary definitions and introduce the concept of normed vector spaces. We recall that a vector space is a collection of objects

with two binary operations that satisfy eight axioms (as discussed in MA149 Linear Algebra). Examples of vector spaces, which we will encounter in this module, are:

1. $\mathbf{R}^n = \{(a_1, \dots a_n) \colon a_1, \dots a_n \in \mathbb{R}\}$
2. The space of continuous function on $[a, b]$, that is $C[a, b] = \{f \colon [a, b] \Rightarrow \mathbb{R} \colon f \text{ is continuous}\}$.
3. The space of polynomials of degree n, that is $\mathcal{P}_n = \{\sum_{k=0}^{n} a_k x^k \colon a_0, \dots a_n \in \mathbb{R}\}$
4. The space of trigonometric polynomials of degree n, defined as $\mathcal{T}_n = \{\sum_{k=1}^{n} a_k \sin k\pi x + b_k \cos k\pi x \colon a_1, \dots, a_n, b_1, \dots b_n\}$.

A normed vector space is a vector space (over the real or complex numbers), on which a norm is defined. Any norm on X induces a metric or distance function by setting $d(x, y) = \|x - y\|$, making the normed vector space a metric space. We say that $(X, \|\cdot\|)$ is complete, that is every Cauchy sequence in X as a limit in X, then $(X, \|\cdot\|)$ is a Banach space.

Example 16.1. Examples of normed vector spaces

- $X = \mathbb{R}^d$ with the ℓ_2-norm of $\mathbf{x} = (x_1, \dots x_d)$ given by $\|\mathbf{x}\|_2 = \left(\sum_{k=1}^{d} |x_k|^2\right)^{\frac{1}{2}}$. We can define many other norms on $\mathbb{R}$, of particular interest are the ℓ_p norms:

$$\|\mathbf{x}\|_p = \Big(\sum_{k=1}^{d} |x_k|^p\Big)^{\frac{1}{p}} \quad \text{for } 1 \le p < \infty$$
$$\|\mathbf{x}\|_\infty = \max_{1 \le i \le d} |x_k|.$$

  We will see (in the examples) that the $\|\cdot\|_p$ are indeed norms.

- Let $I = [a, b] \subset \mathbb{R}$ be a compact interval. The vector space of continuous functions

$$C(I) = \{f \colon [a, b] \to \mathbb{R}, f \text{continuous}\}$$

  with the ∞-norm (also known as the minmax or Chebyshev norm)

$$\|f\|_\infty = \max_{a \le x \le b} |f(x)| \tag{16.1}$$

  is a normed linear vector space.

- Let w be a real-valued function, that is continuous, positive and integrable on the interval (a, b). Then the vector space of continuous functions on $[a, b]$, that is $C[a, b]$ equipped with the 2-norm

$$\|f\|_2 = \left(\int_a^b w(x) \|f(x)\|^2 \, \mathrm{d}x\right)^{\frac{1}{2}} \tag{16.2}$$

  is a normed linear vector space. The function w is called a weight function. This norm is thought as the analogue to the ℓ_2 norm for vectors.

Lemma 16.2. Let $w \colon (a, b) \to \mathbb{R}$ be a continuous, positive and integrable function. Then for any function $f \in C[a, b]$

$$\|f\|_2 \le C \|f\|_\infty \quad \text{where} \quad C = \left[\int_a^b w(x) \, \mathrm{d}x\right]^{\frac{1}{2}}.$$

Proof. Trivial. □

We will see that the best approximation in the 2-norm is closely related to orthogonality and the concept of an inner product, see Chapter 18.2. A linear space with an inner product is called an inner product space. An orthogonal basis for an inner product space V is a basis for V whose vectors are mutually orthonormal. If the vectors of the orthonormal basis are normalised, the basis is call an orthonormal basis.

Example 16.3. Example of inner products

- Let $\mathbf{x}$ and $\mathbf{y}$ be d-dimensional vectors. Then $\mathbb{R}^d$ is an inner product space with

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{k=1}^{d} x_k y_k \quad \text{for } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

- Let $C[a, b]$ be the set of continuous real valued functions. It is an inner product space with

$$\langle f, g \rangle = \int_a^b w(x) f(x) g(x) \, \mathrm{d}x, \tag{16.3}$$

  where w is a positive, continuous and integrable weight function.

We know that every scalar product induces a norm

$$\|f\| = \langle f, f \rangle^{\frac{1}{2}}.$$

With this norm every inner product space becomes a normed vector space. For example (16.3) induces (16.2).

17

Best Approximation in Finite Dimensional Vector Spaces

We start by discussing the existence and uniqueness of a best approximation in general finite dimensional sub-spaces. Note that the following results do not tell us how to actually construct these approximations. The results in the subsequent chapters will focus on the specific construction for different choices of finite dimensional vector spaces and distance functions, such as the polynomial or rational approximation wrt to the ∞ or 2-norm.

While existence follows if the finite dimensional subspace is a closed subspace, uniqueness is more of an issue as the following example illustrates.

Example 17.1 (Uniqueness). Let $X = \mathbb{R}^2$, the subspace $Y = \{(y, 0) \colon y \in \mathbb{R}\}$, and to the ∞ and the 2-norm, respectively. It is easy to see that the point $(0, 1) \in \mathbb{R}^2$ has infinitely many nearest points in Y when considering the ∞ norm, but a unique nearest point wrt the 2-norm.

We will see that the convexity of the distance function as well as the convexity of the set of approxim-

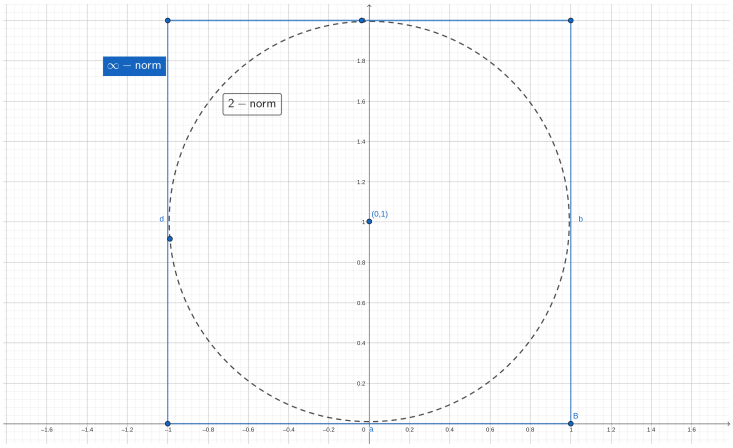


Figure 17.1: Example 17.1: The blue circle illustrates the sphere of radius 1 wrt the ∞-norm; the dashed line wrt to the 2-norm. The point $(0, 1)$ has infinitely many nearest points, in particular the interval $[-1, 1]$, when considering the ∞-norm, while there is only one point on the x axis which is closest wrt to the 2 norm.

ations plays are essential to obtain a unique solution in the following.

But let's start with existence first. We recall that every finite dimensional normed space is complete (hence every Cauchy sequence converges). In particular if Y is a finite dimensional subspace of a normed linear space X, then Y is a closed subset of X.

Corollary 17.2. Let Y be a finite dimensional normed space, let $\mathbf{x} \in Y$ and let $M > 0$. Then, any closed ball of radius M, that is $\mathcal{B}_M(y) = \{\mathbf{y} \in Y \colon \|\mathbf{x} - \mathbf{y}\| \leq M\}$, is compact.

Proof. First we note that it sufficient to show that the set $\{\mathbf{y} \in Y \colon \|\mathbf{y}\| \leq M\}$, that is the ball centred at 0, is compact.
Let $\mathbf{e}_1, \dots \mathbf{e}_n$ denote the basis of Y. We recall that because of Lemma ?? there exists a constant $C > 0$ such that

$$C \sum_{k=1}^{n} |v_k| \leq \| \sum_{k=1}^{n} v_k \mathbf{e}_k \|$$

for all $\mathbf{v} = \sum_{k=1}^{n} v_k \mathbf{e}_k \in Y$. In particular

$$C|v_k| \leq \| \sum_{k} v_k \mathbf{e}_k \| \leq M \Rightarrow v_k \leq \frac{M}{C} \text{ for } k = 1, \dots n.$$

Therefore $\mathcal{B}_M$ is a closed subset of the compact subset

$$\{v = \sum_{k=1}^{n} v_k \mathbf{e}_k \colon |v_k| \leq \frac{M}{C}, k = 1, \dots n\} = [-\frac{M}{C}, \frac{M}{C}]^n.$$

□

Next we will show that an approximation to $x \in X$ by elements of a finite dimensional subspace Y does indeed exist (although it may not be unique).

Theorem 17.3. Let Y be a finite-dimensional subspace of a normed linear space X, and let $x \in X$. Then there a exists (a not necessarily unique) vector $\mathbf{y}^* \in Y$ such that

$$\|x - \mathbf{y}^*\| = \min_{\mathbf{y} \in Y} \|x - \mathbf{y}\|$$

for all $\mathbf{y} \in Y$.

Proof. Since $\mathbf{0}$ is an element of Y, the nearest point $\mathbf{y}^* \in Y$ will satisfy $\|x - \mathbf{y}^*\| \leq \|x\|$.Therefore, it is sufficient to find $\mathbf{y}^*$ in the compact set

$$K = \{\mathbf{y} \in Y \colon \|x - \mathbf{y}\| \leq \|x\|\}.$$

Next we note that the function $q(\mathbf{y}) \colon \mathbf{y} \to \|x - \mathbf{y}\|$ is continuous:

$$|q(\mathbf{y}) - q(z)| = |\|x - \mathbf{y}\| - \|x - \mathbf{z}\|| \leq \|\mathbf{y} - \mathbf{z}\|$$

Hence q attains a minimum at some point $\mathbf{y}^* \in K$. □

Consider the space of polynomials of degree n, defined as

$$\mathcal{P}_n = \{\sum_{k=0}^{n} a_k x^k \colon a_0, \dots a_n \in \mathbb{R}\}. \tag{17.1}$$

Then the previous theorem ensures that for every function $f \in C[a,b]$ and positive integer $n \in \mathbb{N}$ there exists a not necessarily unique polynomial $p_n^* \in \mathcal{P}_n$ such that

$$\|f - p_n^*\| = \min_{p \in \mathcal{P}_n} \|f - p\|.$$

Note that the polynomial does not need to have exactly the degree n, it is a polynomial of at most degree n. For example, the best approximation to $f(x) = x$ is of course the polynomial $p(x) = x$ (also when considering the problem in the subspace of polynomials of degree at most 3).

After haven shown that we have indeed a solution, we address the question of uniqueness.

Lemma 17.4. Let Y be a finite dimensional subspace of a normed linear space X, and suppose that each $x \in X$ has a unique nearest point $\mathbf{y}_x \in Y$. Then the nearest map $x \to \mathbf{y}_x$ is continuous.

Proof. We denote by $P(x) = \mathbf{y}_x$ the nearest point map and assume that $x_n \to x$ in X. We want to show that $P(x_n) \to P(x)$, however, it is sufficient to show that there exists a converging sub-sequence of $P(x_n)$ (why?).
Since x_n is bounded, we have that $\|x_n\| \leq C$ for all $n \in \mathbb{N}$ and therefore

$$\|P(x_n)\| \leq \|P(x_n) - x_n\| + \|x_n\| \leq 2\|x_n\| \leq 2C.$$

Therefore $(P(x_n))$ is a bounded sequence in Y, and therefore there exists a sub-sequence which converges to some element $P_0 \in Y$.
Next we show that $P_0 = P(x)$. We know that

$$\|P(x_n) - x_n\| \leq \|P(x) - x_n\| \text{ for any } n.$$

Letting $n \to \infty$ we obtain

$$\|P_0 - x\| \leq \|P(x) - x\|.$$

Since nearest points in Y are unique, we conclude that $P_0 = P(x)$. □

Note that the nearest point map is, in general, nonlinear and therefore hard to work with. However, the set of best approximations has a nice underlying structure.

Theorem 17.5. Let Y be a subspace of a normed linear space X, and let $x \in X$. Then the set of all best approximations to x in Y, denoted by Y_x, is bounded and convex.

Proof. We recall that a subset K of a vector space V is said to be convex if K contains the line segment joining every pair of points. Specifically, K is convex if

$$x, y \in K, 0 \leq \lambda \leq \Rightarrow \lambda x + (1-\lambda)y \in K.$$

Therefore for any y_1 and y_2 in Y_x and $0 \leq \lambda \leq 1$ we have to show that the vector $y_x = \lambda y_1 + (1-\lambda)y_2 \in Y_x$:

$$\|x - y_1\| = \|x - y_2\| = \min\|x - y\|.$$

Then

$$\begin{aligned}\|x - y_x\| &= \|x - \lambda y_1 - (1-\lambda)y_2\| \\ &= \|\lambda(x - y_1) + (1-\lambda)(x - y_2)\| \\ &\leq \lambda\|x - y_1\| + (1-\lambda)\|x - y_2\| \\ &= \min_{y \in Y}\|x - y\|.\end{aligned}$$

Therefore $\|x - y^*\| = \min_{y\in Y}\|x - y\|$, hence $y^* \in Y_x$. □

Note that if we assume that Y is finite dimensional then Y_x is a compact convex set.

The previous theorem implies that Y_x is either empty (no solution), contains one point (unique best approximate) or an entire line segment (then we have infinitely many solutions). This provides a sufficient condition for the uniqueness of nearest points; if the normed space X contains no line segments on any sphere $\{x \in X \colon \|x\| = r\}$, then the best approximate (in a convex subset) will be unique.

We recall that a norm on a vector space is called strictly convex if for any $x, y \in X$, $x \neq y$ with $\|x\| = \|y\| = r$ we have that

$$\|\lambda x + (1-\lambda)y\| < r \quad \text{for all } 0 < \lambda < 1.$$

Hence the open line segment between any points on the sphere of radius r, lies entirely within the open ball of radius r. We often call a space X strictly convex, when referring to the strict convexity of the norm in X.

Corollary 17.6. Consider X with a strictly convex norm. Then for any subspace Y of X and any $x \in X$, the set Y_x is either empty or consists of one element.

The following lemma allows us to check whether a space X has a strictly convex norm.

Lemma 17.7. A normed space X has a strictly convex norm, if and only if the triangle inequality is strict for non-parallel vectors; that is

$$x \neq \alpha x, y \neq \alpha x, \text{ for all } \alpha \in \mathbb{R} \Rightarrow \|x + y\| < \|x\| + \|y\|.$$

Proof. $'\Rightarrow'$: Let x and y be non-parallel vectors and X strictly convex. Then $\frac{x}{\|x\|}$ and $\frac{y}{\|y\|}$ are different and therefore

$$\left\|\frac{\|x\|}{\|x\| + \|y\|}\frac{x}{\|x\|} + \frac{\|y\|}{\|x\| + \|y\|}\frac{y}{\|y\|}\right\| < 1$$

Therefore $\|x + y\| < \|x\| + \|y\|$.
$'\Leftarrow'$: assume that the triangle inequality is strict on non-parallel vectors and let $x \neq y \in Y$ with $\|x\| = r = \|y\|$. If x and y would be parallel, then $y = -x$ and then

$$\|\lambda x + (1-\lambda)y\| = |\lambda - 1|\|x\| < r$$

since $-1 < 2\lambda - 1 < 1$ for $0 < \lambda < 1$. Otherwise x and y are not parallel. Therefore, for any $0 < \lambda < 1$ the vectors λx and $(1-\lambda)y$ are non-parallel and

$$\|\lambda x + (1-\lambda)y\| < \lambda\|x\| + (1-\lambda)\|y\| = r.$$

□

18

Polynomial Approximation

In polynomial approximation we wish to find a polynomial function that closely matches the behaviour of the target function within a specified interval or region. This approach relies on expressing the function as a finite sum of polynomial terms, where the coefficients of the polynomial are determined to minimise the discrepancy between the approximation and the original function.

We start by defining the spaces we wish our approximation to belong to. In this section we consider the space of polynomials of degree n, defined as

$$\mathcal{P}_n(I) = \{f = \sum_{k=1}^{n} a_k x^k \colon a_0, \dots a_n \in \mathbb{R}\}. \tag{18.1}$$

We can now rephrase Chebyshev's problem in a more mathematical way:

Given a closed subspace Y of a normed linear space X and a point $x \in X$, is there an element $y \in Y$ that is nearest to x? That is, can we find a vector $y \in Y$ such that $\|x - y\| = \min_{z \in Y} \|x - z\|$? And if there is such a best approximation to x from y, is it unique?

It makes sense to choose a subspace Y that is a closed set of X (otherwise points in $\bar{Y} \backslash Y$ do not have a nearest point). In this section we will set $Y = \mathcal{P}_n$.

It is clear that the choice of the distance measure/norm has a strong influence on the outcome (and the mathematical tools to use). Despite the fundamental differences in the norms there is one common underlying result that is independent from the choice of norms: if no limitation on the degree of the approximation polynomial is imposed, then the approximation error $f - p$ can be made arbitrarily small in both norms. This central result in approximation theory of polynomial approximation is formulated in the next theorem.

Theorem 18.1 (Weierstrass Approximation Theorem). Suppose that f is a real valued function, defined and continuous on a bounded closed interval $[a, b]$ of the real line. Then given any $\varepsilon > 0$ there exists a polynomial p such that

$$\|f - p\|_\infty \leq \varepsilon.$$

The same result holds in the $2-$ norm if the weight function w is real valued, positive, continuous and integrable on (a, b).

Several proofs for Weierstrass' Theorem can be found in the literature. The first proof was given by Bernstein, and uses the so-called Bernstein polynomials. As the proof is non-constructive we will omit its details in this lecture.

It should not be surprising that the mathematical tools to analyse the best approximation problem in the $\|\cdot\|_\infty$-norm are quite different to the ones for the $\|\cdot\|_2$-norm. We therefore discuss the two problems separately in the next two sections.

The rest of this chapter follows the book by E. Süli and D. F Mayers, see [?].

18.1 Best Approximation in the ∞-Norm

Weierstrass's theorem ensures that any function in $C[a,b]$ can be approximated arbitrarily well from the set of all polynomials. However, if we restrict ourselves to the set of polynomials of degree n or less, with n fixed, then the theorem does not guarantee the existence of such a polynomial.

Consider a polynomial $q_n \in \mathcal{P}_n$ of the form

$$q_n(x) = c_0 + c_1 x + \ldots + c_n x^n.$$

and define the function $E : (c_0, \ldots c_n) \mapsto E(c_0, c_1, \ldots c_n)$ defined by

$$\begin{aligned} E(c_0, c_1, \ldots c_n) &= \|f - q\|_\infty \\ &= \max_{x\in[a,b]} |f(x) - c_0 - c_1 x - \ldots c_n x^n| \end{aligned}$$

Since the best approximation is found by minimising the maximum absolute value of the error $f - q$, the problem is often referred to as the minimax polynomial.
The following theorem will establish the existence of a polynomial of best approximation wrt to the ∞ norm.

Theorem 18.2. Let $f \in C[a,b]$. Then there exists a polynomial $p_n \in \mathcal{P}_n$ such that

$$\|f - p\| = \min_{q\in\mathcal{P}_n} \|f - q\|_\infty.$$

Proof. Consider the function

$$(c_0, \ldots, c_n) \in \mathbb{R}^{n+1} \mapsto E(c_0, \ldots, c_n)$$

defined by

$$E(c_0, \ldots c_n) = \|f - q_n\|_\infty \text{ where } q_n = c_0 + c_1 x + \ldots + c_n x^n.$$

We first show that E is continuous. Therefore E will attain its min/max on any bounded closed set in $\mathbb{R}^{n+1}$. Hence, we construct a non-empty bounded closed set $\mathcal{S} \subset \mathbb{R}^{n+1}$ such that the lower bound of E on $\mathcal{S}$ is the same as the lower bound of E on the whole of $\mathbb{R}^{n+1}$.

Step 1: Continuity of E

To show that E is continuous at every $(c_0, c_1, \dots c_n) \in \mathbb{R}^{n+1}$, we consider $(\delta_0, \dots \delta_n) \in \mathbb{R}^{n+1}$ and define the corresponding polynomial $\eta_n(x) = \delta_0 + \delta_1 x + \ldots + \delta_n x^n$. We compute

$$\begin{aligned} E(c_0 + \delta_0, c_1 + \delta_1, \dots c_n + \delta_n) &= \|f - (q_n + \eta_n)\|_\infty \\ &\leq \|f - q_n\|_\infty + \|\eta_n\|_\infty \\ &= E(c_0, \dots c_n) + \|\eta_n\|_\infty. \end{aligned}$$

For every $\varepsilon > 0$ we choose $\delta = \frac{\varepsilon}{(1+\ldots+K^n)}$, where $K = \max\{|a|, |b|\}$. Consider any $(\delta_0, \dots \delta_n) \in \mathbb{R}^{n+1}$ such that $|\delta_i| \leq \delta$ for all $i = 0, 1, \dots n$. Then

$$\begin{aligned} E(c_0 + \delta_0, \dots c_n + \delta_n) - E(c_0, \dots c_n) &\leq \|\eta_n\|_\infty \\ &\leq \max_{x \in [a,b]} (|\delta_0| + |\delta_1||x| + \ldots + |\delta_n||x|^n) \\ &\leq \delta(1 + \ldots + K^n) \\ &= \varepsilon. \end{aligned}$$

Using the same arguments we see that

$$\begin{aligned} E(c_0, \dots c_n) = \|f - q\|_\infty = \|f - (q_n + \eta_n) - \eta_n\|_\infty \\ \leq \|f - (q_n - \eta_n)\|_\infty + \|\eta_n\|_\infty \\ \leq E(c_0 + \delta_0, \dots c_n + \delta_n) + \varepsilon. \end{aligned}$$

Therefore $E(c_0, \dots, c_n) - E(c_0 + \delta_0, \dots c_n + \delta_n) \leq \varepsilon$. These two estimates imply that

$$|E(c_0 + \delta_0, \dots, c_n + \delta_n) - E(c_0, \dots c_n)| \leq \varepsilon.$$

And therefore E is continuous at any $(c_0, \dots c_n) \in \mathbb{R}^{n+1}$.

Step 2: Construction of $\mathcal{S}$: Define the set $\mathcal{S}$ as the set of all points $(c_0, \dots c_n) \in \mathbb{R}^{n+1}$ such that

$$E(c_0, \dots c_n) \leq \|f\|_\infty + 1.$$

First, we note that $\mathcal{S}$ is non-empty (since $E(0, 0, 0 \dots 0) = \|f\|_\infty$ and therefore $(0, \dots 0) \in \mathcal{S}$). Hence the continuous function E attains its lower bound on the set $\mathcal{S}$ and we will call this point $(c_0^*, \dots c_n^*)$.
Since $(0, \dots 0) \in \mathcal{S}$, we have that

$$d = \min_{(c_0, c_1, \dots c_n) \in \mathcal{S}} E(c_0, \dots c_n) \leq E(0, 0 \dots 0) = \|f\|_\infty.$$

From the definition of $\mathcal{S}$ we know that all elements outside $\mathcal{S}$ satisfy

$$E(c_0, \dots c_n) > \|f\|_\infty + 1 \qquad \forall (c_0, \dots c_n) \in \mathbb{R}^{n+1} \backslash \mathcal{S}.$$

And therefore $(c_0, \dots c_n) \notin \mathcal{S}$ we have that $E(c_0, \dots c_n) > d + 1 > d$. Thus the lower bound d of E on the set $\mathcal{S}$ is the same as the lower bound over all values $(c_0, \dots c_n) \in \mathbb{R}^{n+1}$. We know that it is attained at the point $(c_0^*, \dots c_n^*) \in \mathcal{S}$. Therefore $p_n^* = c_0^* + c_1^* x + \dots c_n^* x^n$ is the best approximation of degree n to f in the ∞ norm.

□

The proof of the previous theorem is non-constructive and does not give us an idea how we can actually find the minimax polynomial. In the following we discuss a simple example, which will give us a better idea about the characteristics of the minimax polynomial.

Example 18.3. Let $f \in C[0,1]$ be strictly monotonically increasing function on $[0,1]$. We wish to find the minimax polynomial of degree zero.
The minimax polynomial of degree zero is of the form $p_0(x) \equiv 0$, so have to determines the coefficient $c_0 \in \mathbb{R}$ by finding a minimum of

$$\|f - p_0\|_\infty = \max_{x\in[0,1]} |f(x) - c_0|.$$

Since f is monotonic increasing, the function $f(x) - c_0$ attains its minimum at $x = 0$ and its max at $x = 1$. We see that $|f - c_0|$ attains its maximum at one of the two endpoints, and therefore

$$E(c_0) = \min_{x\in[0,1]} |f(x) - c_0| = \max\left(|f(0) - c_0|, |f(1) - c_0|\right).$$

Since

$$E(c_0) = \begin{cases} f(1) - c_0 & \text{if } c_0 < \frac{1}{2}(f(0) + f(1)) \\ c_0 - f(0) & \text{if } c_0 \geq (f(0) + f(1)). \end{cases}$$

The minimum of the graph $c_0 \mapsto E(c_0)$ is attained at $c_0^* = \frac{1}{2}(f(0) + f(1))$ and therefore the minimum zero order polynomial is given by

$$p_0(x) = \frac{1}{2}(f(0) + f(1)).$$

More generally, if we drop the monotonicity assumption on f and assume that ξ and ζ are two points in $[a,b]$ where f attains its minimum and maximum, then the minimax polynomial of degree 0 is given by

$$p_0(x) = \frac{1}{2}(f(\xi) + f(\zeta)) \qquad x \in [a,b].$$

This example shows that the minimax polynomial p_0 has the property that the approximation error attains its extrema at two points, that is $x = \xi$ and $x = \zeta$, with the error

$$f(x) - p_0(x) = \frac{1}{2}(f(x) - f(\xi)) + \frac{1}{2}(f(x) - f(\zeta))$$

being negative at $x = \xi$ and positive at $x = \zeta$. One can show that this property holds in general. The corresponding statement about the oscillating nature of the approximation error is referred to as the Oscillation Theorem. We only state the result, but do not prove it in the following.

Theorem 18.4 (The Oscillation Theorem). Let $f \in C[a,b]$. A polynomial $p_n \in \mathcal{P}_n$ is a minimax polynomial for f on $[a,b]$, if and only if, there exists a sequence of $n+1$ points x_i, $i = 0, 1, \ldots n+1$ such that $0 \leq x_0 < \ldots < x_{n+1} \leq b$ with

$$|f(x_i) - p_n(x_i)| = \|f - r\|_\infty \quad i = 0, 1, \ldots n+1$$

and

$$f(x_i) - p_n(x_i) = -|f(x_{i+1} - p_n(x_{i+1})|), \quad i = 0, \ldots n.$$

In other words, the theorem says that $f - p_n$ attains its maximum absolute value with alternating signs at the points x_i. The points x_i are often referred to as the critical points.

We will use the oscillation theorem to construct a minimax polynomial of order one for a function $f \in C[a, b]$ with strictly monotonically increasing f'. We seek $p_1 \in \mathcal{P}_1$ of the form

$$p_1(x) = c_0 + c_1 x.$$

The difference $f - (c_0 + c_1 x)$ attains its extrema either at the endpoints of $[a, b]$ or at points where $f'(x) - c_1$ is zero. Since f' is strictly monotonically increasing it can only take the value c_1 at a single point only, and therefore the endpoints a and b are critical points. Let d denote the third critical point inside the interval $[a, b]$ and whose location remains to be determined. At $x = d$ we have that

$$(f(x) - (c_1 x + c_0))' \,|_{x=d} = 0.$$

By the oscillator theorem with $x_0 = a$, $x_1 = d$ and $x_2 = b$ we have that

$$f(a) - (c_1 a + c_0) = A \tag{18.1a}$$
$$f(d) - (c_1 d + c_0) = -A \tag{18.1b}$$
$$f(b) - (c_1 b + c_0) = A \tag{18.1c}$$

where A either takes the value $A = L$ or $A = -L$ with $L = \max_{x\in[a,b]} |f(x) - p_1(x)|$. Since

$$f'(d) = c_1 \tag{18.2}$$

this gives us enough equations to compute d, c_0, c_1 and A.
By subtracting the first from the third equation in (18.1) we deduce that

$$c_1 = \frac{f(b) - f(a)}{b - a}.$$

Then the value of d is uniquely determined by (18.2) (since f' is assumed to be continuous and strictly monotonically increasing). Then we can compute c_0 by adding the second to the first equation in (18.1), and finally A.

Chebyshev polynomials

It is in general very difficult to write down simple closed forms for minimax polynomials. The Chebyshev polynomials are one of the few cases.

Definition 18.5. The Chebyshev polynomials of degree n are defined as follows:

$$T_n(x) = \cos(n \cos^{-1} x) \qquad \text{for } n = 0, 1, 2, \ldots \text{ and } x \in [-1, 1]. \tag{18.3}$$

Although it is not immediately clear, T_n are in fact polynomials. For example, the first three Chebyshev polynomials on $[-1, 1]$ are:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$

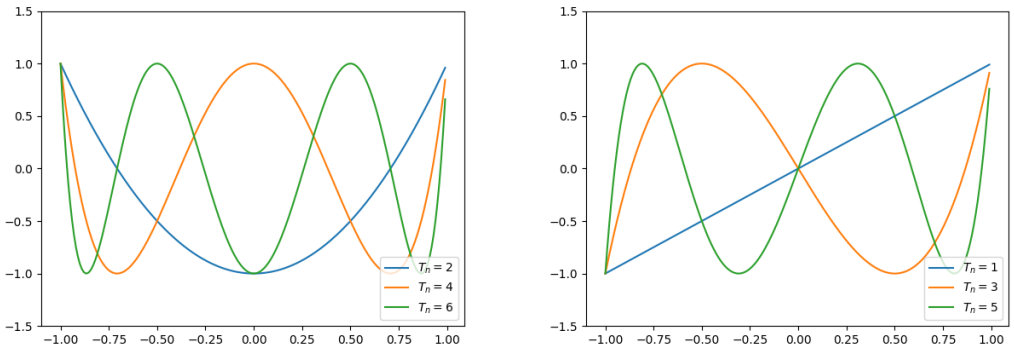


Figure 18.1: Left: first three even Chebyshev polynomials on $[-1, 1]$; Right: first three odd Chebyshev polynomials on $[-1, 1]$.

The first three even and odd Chebyshev polynomials are depicted in Figure 18.1.

The Chebyshev polynomials can be calculated from the following trigonometric identity

$$\cos((n+1)\varphi) + \cos((n-1)\varphi) = 2\cos\varphi\cos(n\varphi)$$

which yields the recurrence relation

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \qquad n = 1, 2, 3\ldots \text{ and } x \in [-1, 1].$$

Since T_0 and T_1 are polynomials on $[-1, 1]$, T_n $n = 2, 3\ldots$ are polynomials of degree n (follows by induction). The Chebyshev polynomials have the following properties

Lemma 18.6. Consider the Chebyshev polynomials T_n as defined in (18.3). Then T_n satisfy

1. $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$, $x \in [-1, 1]$ and $n = 1, 2, 3\ldots$
2. T_n is a polynomial of order n with leading coefficient $2^{n-1}x^n$ for $n \geq 1$.
3. T_n is even on $[-1, 1]$ is n is even, and odd if n is odd.
4. The real and distinct zeros of T_n are given by

$$x_j = \cos\frac{(2j-1)\pi}{2n} \quad j = 1, \ldots n.$$

5. $l|T_n(x)| \leq 1$ for all $x \in [-1, 1]$
6. T_n alternates between 1 and -1 at the points $x_k = \cos(\frac{k\pi}{n})$, $k = 0, 1, \ldots$.

Theorem 18.7. Let $n \geq 0$ and consider the polynomial $p_n \in \mathcal{P}_n$ given by

$$p_n(x) = x^{n+1} + 2^{-n}T_{n+1}(x) \qquad x \in [-1, 1]$$

Then p_n is the minimax approximation of degree n to the function $x \mapsto x^{n+1}$ on the interval $[-1, 1]$.

Proof. Lemma 18.6 we have that $p_n \in \mathcal{P}_n$. Since

$$x^{n+1} - p_n(x) = 2^{-n} T_{n+1}(x)$$

and using Lemma 18.6 we know that the difference $x^{n+1} - p_n(x)$ does not exceed the value 2^{0n} on $[-1, 1]$ and attains this value with alternating signs at $n+2$ points $x_k = \cos\left(\frac{k\pi}{n+1}\right)$ for $k = 0, 1, \ldots n+1$.- Then, by the oscillation theorem, we can conclude that p_n is the (unique) minimax polynomial approximation from $\mathcal{P}_n$ to the function $x \mapsto x^{n+1}$ over $[-1, 1]$. □

18.2 Best Approximation in the 2-Norm

Consider the best approximation problem in the 2 norm, that is

Given a function $f \in L^2_w(a, b)$, find a polynomial $p \in \mathcal{P}_n$ such that

$$\|f - p_n\|_2 = \inf_{q \in \mathcal{P}_n} \|f - q\|_2.$$

The function p_n is called the best approximation of degree n in the 2-norm.
Before discussing the existence and uniqueness of such a best approximation we consider some simple illustrative examples first.

Example 18.8. Let $\varepsilon > 0$ and let $f(x) = 1 - e^{-\frac{x}{\varepsilon}}$ with $x \in [0, 1]$, see Figure 18.2. We wish to find the lowest order approximation in the 2-norm with weight function $w \equiv 1$ on $[0, 1]$ and compare it to the minmax polynomial of degree 0.
To find the minimal polynomial we have to minimise the function

$$\|f - c\|_2 = \int_0^1 (f(x) - c)^2 \, \mathrm{d}x = \int_0^1 |f(x)|^2 \, \mathrm{d}x - 2c \int_0^1 f(x) \, \mathrm{d}x + c^2.$$

We minimise over $c \in \mathbb{R}$, and the previous expression is a quadratic function in c. Its minimum is achieved at

$$c = \int_0^1 f(x) \, \mathrm{d}x = 1 - \varepsilon + \varepsilon e^{-\frac{1}{\varepsilon}}.$$

Hence the best approximating polynomial of order 0 in the 2-norm is given by

$$p_0^{\mathbf{2}}(x) = 1 - \varepsilon + \varepsilon e^{-\frac{1}{\varepsilon}}.$$

Note that the super-index is NOT a power, it only refers to the best approximation in the 2 norm.
The minmax approximation of degree 0 is the arithmetic mean of $f(0)$ and $f(1)$:

$$p_0^{\boldsymbol{\infty}}(x) = \frac{1}{2}\left(1 - e^{-\frac{1}{\varepsilon}}\right).$$

We see that for $0 < \varepsilon \ll 1$, $p_0^{\boldsymbol{\infty}} \approx \frac{1}{2}$, while $p_0^{\mathbf{2}}(x) \approx 1$. Quite a discrepancy .....

Next we wish to construct $p_n \in \mathcal{P}_n$, with $n \geq 0$ for a weight function $w \equiv 1$. We can write the polynomial as

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots c_n x^n$$

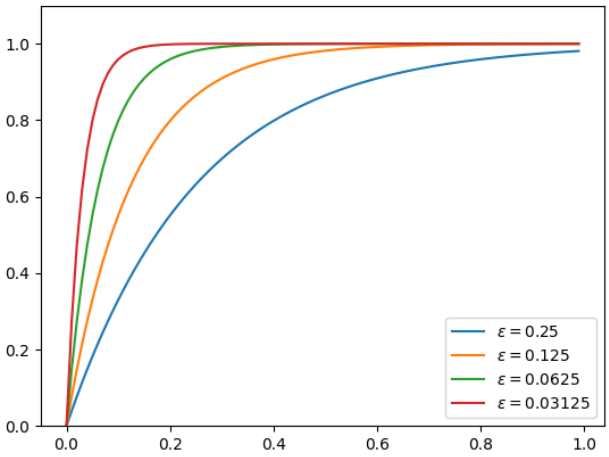


Figure 18.2: Graph of the function $x \mapsto 1 - e^{\frac{x}{\varepsilon}}$ for different values of ε.

and wish to determine the best coefficients c_j, $j = 0, \dots n$ minimising the error function

$$\|e_n\|_2 = \|f - p_n\|_2 = \left(\int_0^1 |f(x) - p_n(x)|^2 \, \mathrm{d}x \right)^{\frac{1}{2}}.$$

Since the function $\xi \to \sqrt{\xi}$ is monotonically increasing, the problem is equivalent to minimising the square of the norm. Therefore we consider the following

$$\begin{aligned} E(c_0, c_1, \dots c_n) &= \int_0^1 (f(x) - p_n(x))^2 \, \mathrm{d}x \\ &= \int_0^1 f(x)^2 \, \mathrm{d}x - 2 \sum_{j=0}^{n} c_j \int_0^1 f(x) x^j \, \mathrm{d}x + \sum_{j=0}^{n} \sum_{k=0}^{n} c_j c_k \int_0^1 x^{k+j} \, \mathrm{d}x. \end{aligned}$$

To find a minimum we set the partial derivatives wrt c_j, $j = 0, \dots n$ to zero and obtain a system of $(n+1)$ linear equations for the coefficients c_j:

$$\sum_{k=0}^{n} a_{jk} c_k = b_j \quad j = 0, \dots n$$

with

$$a_{jk} = \int_0^1 x^{k+j} \, \mathrm{d}x = \frac{1}{k+j+1} \text{ and } b_j = \int_0^1 f(x) x^j \, \mathrm{d}x.$$

We can also use the inner product to define the entries of the matrix $\mathbf{A} = (a_{jk})$ and the vector $\mathbf{b} = (b)_j$, since $a_{jk} = \langle x^k, x^j \rangle$ and $b_j = \langle f, x^j \rangle$. A modified calculation can be used for positive continuous and integrable weight functions on the interval (a, b).

The matrix $\mathbf{A}$ is known as the Hilbert matrix, and we will show in the example classes that it becomes singular as the dimension/the polynomial degree n increases. The polynomials x^j, $j = 0, \dots n$ form a basis of the linear space $\mathcal{P}_n$. However, we have seen that this results in what we call an ill-conditioned

problem, which is why we wish to use different basis functions. In particular we wish to find a basis such that the system of linear equations is a diagonal matrix (since solving the system will be a trivial exercise then). Let $\varphi(x)_j$, $j = 0, \dots n$ form a basis of $\mathcal{P}_n$, $n \geq 0$ and consider polynomials of the form

$$p_n(x) = \gamma_0\varphi_0(x) + \gamma_1\varphi_1(x) + \ldots + \gamma_n\varphi_n,$$

with $\gamma_0, \dots \gamma_n \in \mathbb{R}$ to be determined. The same calculation as above yields

$$\sum_{k=0}^{n} m_{jk}\gamma_k = \beta_j \quad j = 0, \dots n$$

with $m_{jk} = \langle \varphi_j, \varphi_k \rangle$ and $\beta_j = \langle f, \varphi_j \rangle$ with the inner product defined as

$$\langle g, h \rangle = \int_a^b w(x)g(x)h(x)\,\mathrm{d}x.$$

For $\mathbf{M} = (m_{jk})$ to be a diagonal matrix, the functions φ have to be chosen such that $\langle \varphi_j, \varphi_k \rangle = 0$ for $j \neq k$. Hence φ_k has to be orthogonal to φ_j for $j \neq k$. Having a diagonal matrix makes the approximation much more efficient, which is why orthogonal polynomials play such a central role in approximation theory. In the following we will discuss different orthogonal polynomials.

Orthogonal polynomials

We start with the proper definition of orthonormal polynomials.

Definition 18.9. Let w be an integrable, positive and continuous weight function on (a, b). Consider a sequence of polynomials φ_j, $j = 1, \ldots$. If

$$\int_a^b w(x)\varphi_i(x)\varphi_j(x)\,\mathrm{d}x = \begin{cases} 0 & \text{for all } i \neq j \\ 1 & \text{if } i = j, \end{cases}$$

we call the sequence φ_i, $i = 1, \dots$ a system of orthogonal polynomials.

We continue by discussing that such a sequence exists (for weight functions w which satisfy the above conditions).
Let $\varphi_0(x) \equiv 1$ and assume that φ_j have been constructed for $j = 1, \dots n$. Since these polynomials are orthogonal, we have that

$$\int_a^b w(x)\varphi_k(x)\varphi_j(x)\,\mathrm{d}x = 0 \quad j = 1, \dots n \backslash \{k\}.$$

Next we define the polynomial

$$q(x) = x^{n+1} - a_0\varphi_0(x) - a_1\varphi_1(x) \dots - a_n\varphi_n(x).$$

with

$$a_j = \frac{\int_a^b w(x)x^{n+1}\varphi_j(x)\,\mathrm{d}x}{\int_a^b w(x)\left(\varphi(x)\right)^2\,\mathrm{d}x}$$

for $j = 0, \ldots, n$.
It follows

$$\begin{aligned}\int_a^b w(x)q(x)\varphi_j(x)\,\mathrm{d}x &= \int_a^b w(x)x^{n+1}\varphi_j(x)\,\mathrm{d}x - a_j\int_a^b w(x)\left(\varphi_j(x)\right)^2\,\mathrm{d}x \\ &= 0. \qquad \text{for all } 0 \le j \le n.\end{aligned}$$

Therefore the polynomial q is orthogonal to all previous members of the sequence and φ_{n+1} can be defined as a nonzero-constant multiple of q. The above procedure is known as Gram Schmidt orthogonalisation.

Example 18.10. We wish to construct the sequence $\{\varphi_0, \varphi_1, \varphi_2\}$ on the interval (a, b) if $w(x) \equiv 1$.
We set $\varphi_0(x) \equiv 1$ and determine $\varphi_1 = x - c_0\varphi_0$, such that $\langle \varphi_0, \varphi_1\rangle = 1$. Then

$$\langle x, \varphi_0\rangle - c_0\langle \varphi_0, \varphi_0\rangle = 0.$$

and therefore

$$c_0 = \frac{\langle x, \varphi_0\rangle}{\langle \varphi_0, \varphi_0\rangle} = \frac{1}{2}.$$

Hence $\varphi_1 = x - \frac{1}{2}$.
To calculate φ_2 we have to find

$$\varphi_2(x) = x^2 - (d_1\varphi_1(x) + d_2\varphi_2(x))$$

such that $\langle \varphi_2(x), \varphi_1(x)\rangle = \langle \varphi_2(x), \varphi_0(x)\rangle = 0$. This implies that

$$\begin{aligned}\langle x^2, \varphi_1\rangle - d_1\langle \varphi_1, \varphi_1\rangle - d_0\langle \varphi_1, \varphi_1\rangle = 0 \\ \langle x^2, \varphi_0\rangle - d_1\langle \varphi_1, \varphi_0\rangle - d_0\langle \varphi_1, \varphi_0\rangle = 0\end{aligned}$$

Since $\langle \varphi_1, \varphi_0\rangle = 0$ and $\langle \varphi_0, \varphi_1\rangle = 0$ we have that

$$\begin{aligned}d_1 &= \frac{\langle x^2, \varphi_1\rangle}{\langle \varphi_1, \varphi_1\rangle} = 1 \\ d_0 &= \frac{\langle x^2, \varphi_0\rangle}{\langle \varphi_0, \varphi_0\rangle} = \frac{1}{3}.\end{aligned}$$

and therefore $\varphi_2(x) = x^2 - x + \frac{1}{6}$. One can easily check that φ_i, $i = 1, 2, 3$ are orthogonal on the interval $(0, 1)$ for a weight function $w \equiv 1$.

If we continue the above procedure we can construct a polynomial sequence $\{\varphi_1, \ldots \varphi_n\}$ wrt the weight function $w \equiv 1$ on the interval $(0, 1)$. Note that we can obtain a sequence of orthonormal polynomials on the interval (a, b) via the linear transformation

$$x \to (b - a)x + a$$

Consider for example the interval $(-1, 1)$ then the mapping $x \to 2x - 1$ gives the so-called Legendre polynomials on $(-1, 1)$.

Example 18.11. Example of orthogonal polynomials include

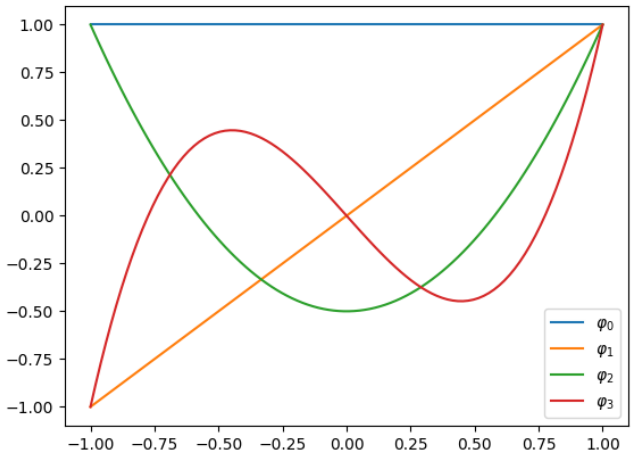


Figure 18.3: The first four Legendre polynomials on the interval $[-1, 1]$

- Legendre polynomials: Show that the first three Legendre polynomials, shown in Figure 18.3, and given by

$$\varphi_0(x) = 1, \varphi_1(x) = x, \varphi_2(x) = \frac{3}{2}x^2 - \frac{1}{2}, \varphi_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x.$$

  are orthogonal polynomials on $(-1, 1)$ for the weight function $w(x) \equiv 1$.

- Chebyshev polynomials: the polynomials

$$T_n \colon x \to \cos(n \cos^{-1} x)$$

  $n = 0, 1, \ldots$ already discussed in Section 18.1 are an orthogonal sequence on the interval $(-1, 1)$ for the weight function $w(x) = \left(1 - x^2\right)^{-\frac{1}{2}}$.

We will now show that the best approximation in the 2 norm exists and is unique.

Theorem 18.12. Let $f \in L^2_w(a, b)$, then there exists a unique polynomial $p_n \in \mathcal{P}_n$ such that

$$\|f - p_n\|_2 = \min_{q \in \mathcal{P}_n} \|f - q\|_2.$$

Proof. We start by normalising the orthogonal polynomials φ_j, $j = 1, \ldots n$ (wrt to the weight w on (a, b)). We define

$$\psi_j(x) = \frac{\varphi_j(x)}{\|\varphi_j\|^2} \qquad j = 1, \ldots n$$

which gives a so-called orthonormal system of polynomials. Then

$$\langle \psi_i, \psi_j \rangle = \begin{cases} 1 & i = j \\ 0 & \text{otherwise.} \end{cases}$$

The polynomials ψ_j, $j = 0, 1, \ldots n$ are linear independent, and form a basis of $\mathcal{P}_n$. Therefore each element $q \in \mathcal{P}_n$ can be written as

$$q(x) = \beta_0\psi_0(x) + \beta_1\psi_1(x) \ldots \beta_n\psi_n(x).$$

To find the best approximate in the 2 norm, we consider the function $E : (\beta_0, \beta_1, \ldots \beta_n) \to E(\beta_0, \ldots \beta_n)$ with $E(\beta_0, \beta_1, \ldots \beta_n) = \|f - q\|$ where $q = \beta_0\psi_0 + \ldots \beta_n\psi_n$. Then

$$\begin{aligned} E(\beta_0, \beta_1, \ldots, \beta_n) &= \langle f - q, f - q\rangle \\ &= \langle f, f\rangle - 2\langle f, q\rangle + \langle q, q\rangle \\ &= \|f\|_2^2 - 2\sum_{j=0}^{n} \beta_j\langle f, \psi_j\rangle + \sum_{j=0}^{n}\sum_{k=0}^{n} \beta_j\beta_k\langle\psi_k, \psi_j\rangle \\ &= \|f\|_2^2 - 2\sum_{j=0}^{n} \beta_j\langle f, \psi_j\rangle + \sum_{j=0}^{n} \beta_j^2 \\ &= \sum_{j=0}^{n} [\beta_j - \langle f, \psi_j\rangle]^2 + \|f\|_2^2 - \sum_{j=0}^{n} |\langle f, \psi_j\rangle|^2. \end{aligned}$$

Since we minimise over $(\beta_0, \beta_1, \ldots \beta_n)$ we see that the function achieves its minimum at

$$\beta_j^* = \langle f, \psi_j\rangle, \qquad j = 0, \ldots n.$$

Its minimum is unique (since it's quadratic in the coefficients β_i, $i = 0, \ldots n$). □

From the proof we can deduce another useful inequality; the so-called Bessel inequality. Since $E(\beta_0^*, \ldots \beta_n^*) = \|f - p_n\|_2^2 \geq 0$, we have that

$$\sum_{j=0}^{n} \|\langle f, \psi_j\rangle|^2 \leq \|f\|_2^2$$

for all $f \in L_w^2(a, b)$ and orthonormal polynomials in $L_w^2(a, b)$.

We conclude by stating a useful theorem to construct orthonormal polynomials.

Theorem 18.13. Let $f \in L_w^2(a, b)$. A polynomial $p_n \in \mathcal{P}_n$ is the polynomial of best approximation in the 2 norm, if and only if

$$\langle f - p_n, q\rangle = 0 \qquad \text{for all } q \in \mathcal{P}_n \tag{18.1}$$

That is, the difference $f - p_n$ is orthogonal to every element in $\mathcal{P}_n$.

Proof. $'\Rightarrow'$: Suppose (18.1) holds, then

$$\langle f - p_n, p_n - q\rangle = 0 \quad \text{for every } q \in \mathcal{P}_n$$

since $p_n - q \in \mathcal{P}_n$ for each $q \in \mathcal{P}_n$. Then

$$\begin{aligned} \|f - p_n\|_2^2 &= \langle f - p_n, f - p_n\rangle \\ &= \langle f - p_n, f - q\rangle + \langle f - p_n, q - p_n\rangle \\ &= \langle f - p_n, f - q\rangle. \end{aligned}$$

Cauchy Schwarz implies that

$$\|f - p_n\|_2^2 \leq \|f - p_n\|_2 \|f - q\|_2 \quad \forall q \in \mathcal{P}_n.$$

Therefore

$$\|f - p_n\|_2 \leq \|f - q\|_2 \quad \forall q \in \mathcal{P}_n.$$

By choosing $q = p_n$ equality will hold and therefore

$$\|f - p_n\|_2 = \min_{q \in \mathcal{P}_n} \|f - q\|_2.$$

$' \Leftarrow '$: Assume that p_n is the best approximating polynomial. We have seen in the proof of Theorem 18.12 that we can write p_n in terms of orthonormal polynomials ψ_k, $k = 0, \ldots n$:

$$p_n(x) = \beta_0^* \psi_0 + \beta_1^* \psi_1 + \ldots \beta_n^* \psi_n(x).$$

where $\beta_\ell^* = \langle f, \psi_\ell \rangle$, $\ell = 0, \ldots n$.
We calculate

$$\begin{aligned}
\langle f - p_n, \psi_j \rangle &= \langle f, \psi_j \rangle - \sum_{k=0}^{n} \beta_k^* \langle \psi_k, \psi_j \rangle \\
&= \langle f, \psi_j \rangle - \sum_{k=0}^{n} \beta_k^* \delta_{kj} \\
&= \langle f, \psi_j \rangle - \beta_j^* \\
&= 0.
\end{aligned}$$

Since $\mathcal{P}_n = \text{span}\{\psi_0, \ldots \psi_n\}$ the statement follows. □

19

Approximation by Rational Functions

In the previous chapter we discussed how we can approximate functions using algebraic polynomials. This has clear advantages, for example

- Any continuous function on a closed interval can be approximated within an arbitrary tolerance (Weierstrass).
- Polynomials can be easily evaluated at arbitrary points
- The derivatives and integrals of polynomials exist and can be calculated quite easily.

However, there are also disadvantages. For example their tendency to oscillate, thereby causing errors. Therefore methods that spread the approximation error more evenly have been developed. They are often based on rational functions of the form

$$r(x) = \frac{p(x)}{q(x)}$$

where $p(x)$ and $q(x)$ are polynomials whose degree sums to N. Every algebraic polynomial is a rational function, hence we expect that they give at least as good results as algebraic polynomials. However, rational functions whose nominator and denominator have the same or a very similar degree often produce much better results than algebraic approximation. They also have the advantage to give better approximations to functions that have discontinuities near, but outside the interval of approximation.

Rational approximation was systematically studied by Henri Pade (1873-1953) in his PhD thesis in 1892. However, the idea was not completely new - already Daniel Bernoulli (1700-1782) and James Stirling (1692-1770) used similar methods in the context of ???.

Consider a rational polynomial of degree $n = N + M$ of the form

$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 + p_1 x + \ldots + p_n x^n}{q_0 + q_1 x + \ldots + q_m x^m}. \tag{19.1}$$

To ensure that $r(x)$ is defined at $x = 0$ we require that $q_0 \neq 0$, and we therefore assume that $q_0 = 1$ from now on. Therefore we have to determine $N + 1$ parameters $q_1, q_2, \ldots q_m, p_0, p_1, \ldots p_n$.

Pade approximation is an extension of Taylor approximation for rational functions; in particular it selects the p_i and q_i

$$\begin{aligned} f(x) - r(x) = f(x) - \frac{p(x)}{q(x)} &= \frac{f(x)q(x) - p(x)}{q(x)} \\ &= \frac{\sum_{i=0}^{\infty} a_i x^i \sum_{i=0}^{m} q_i x^i - \sum_{i=0}^{n} p_i x^i}{q(x)} \end{aligned} \tag{19.2}$$

where we assumed that f has an Maclaurin expansion of the form $f \sim \sum_{i=0}^{\infty} a_i x^i$.
Next we choose the coefficients $q_1, \dots q_m$ and $p_0, \dots p_n$ such that

$$f^{(k)}(0) - r^{(k)}(0) = 0 \text{ for each } k = 0, 1, \dots N.$$

This implies that f agrees with r up to the degree N and therefore we choose the coefficients such that the nominator in (19.2)

$$(a_0 + a_1 x + \dots)(1 + q_1 x + \dots q_m x^m) - (p_0 + p_1 + \dots p_n x^n)$$

has no degrees less then order N. To simplify the notation we set $p_{n+1} = p_{n+2} = \dots = p_N = 0$ and $q_{m+1} = \dots q_N = 0$ to obtain the more compact statement

$$\left(\sum_{i=0}^{k} a_i q_{k-i} \right) - p_k = 0$$

and therefore we obtain $N + 1$ linear equations

$$\sum_{i=0}^{k} a_i q_{k-i} = p_k \quad k = 0, 1, \dots N$$

for the $N + 1$ unknowns $p_0, p_1, \dots p_n, q_1, q_2, \dots q_m$.

Example: Find the Pade approximation of degree 5, then the corresponding linear equations are

$$\begin{array}{c|r} x^0 & a_0 - p_0 = 0 \\ x^1 & a_0 q_1 + a_1 - p_1 = 0 \\ x^2 & a_0 q_2 + a_1 q_1 + a_2 = 0 \\ x^3 & a_0 q_3 + a_1 q_2 + a_2 q_1 + a_3 = 0 \\ x^4 & a_0 q_4 + a_1 q_3 + a_2 q_2 + a_3 q_1 + a_4 = 0 \\ x^5 & a_0 q_5 + a_1 q_4 + a_2 q_3 + a_3 q_2 + a_4 q_1 + a_5 = 0 \end{array} \tag{19.3}$$

We see that $p_0 = a_0$ and obtain a linear system for the remaining parameters

$$\begin{pmatrix} a_0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix} - \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix} = - \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5. \end{pmatrix}$$

If we want that $n = 3$ and $m = 2$ one can reduce the system to

$$\begin{pmatrix} a_0 & 0 & -1 & 0 & 0 \\ a_1 & a_0 & 0 & -1 & 0 \\ a_2 & a_1 & 0 & 0 & -1 \\ a_3 & a_2 & 0 & 0 & 0 \\ a_4 & a_3 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} = - \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5. \end{pmatrix}$$

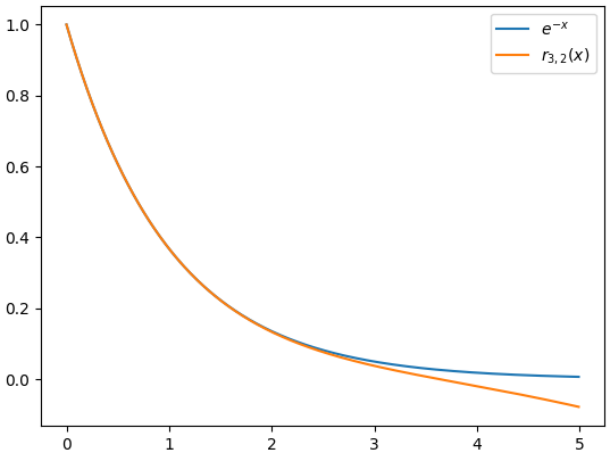


Figure 19.1: Pade approximation of $f(x) = e^{-x}$ by $r_{3,2}(x)$ around x_0

If we now consider the function $f(x) = e^{-x}$ which has the MaxLaurin series $f(x) \approx \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} x^k$ we obtain that $\{a_0, a_1, a_2, a_3, a_4, a_5\} = \{1, -1, \frac{1}{2}, \frac{-1}{6}, \frac{1}{24}, -\frac{1}{120}\}$, which gives

$$\{q_1, q_2, p_1, p_2, p_3\} = \{\frac{2}{5}, \frac{1}{20}, -\frac{3}{5}, \frac{3}{20}, -\frac{1}{60}\}$$

or written as a rational polynomial

$$r_{3,2}(x) = \frac{1 - \frac{3}{5}x + \frac{3}{20}x^2 - \frac{1}{60}x^3}{1 + \frac{2}{5}x + \frac{1}{20}x^2}$$

The Pade approximation is only a local approximation, we see in Figure 19.1 that the approximation gets worse away from $x_0 = 0$.
To overcome the locality one can consider a rational approximation using Chebyshev polynomials, that is

$$r_{n,m}(x) = \frac{\sum_{k=0}^{n} p_k T_k(x)}{\sum_{k=0}^{m} q_k T_k(x)}$$

where $N = n + m$ and $q_0 = 1$. One needs to expand $f(x)$ as a series of Chebyshev polynomials $f(x) = \sum_{k=0}^{\infty} a_k T_k(x)$. The construction works the same as before, but the details of this go beyond the scope of this lecture. Further information can be found in ???.

20

Trigonometric Approximation and the Fourier Series

Trigonometric approximation involves approximating a given function using a combination of trigonometric functions, such as sine and cosine. This approach is particularly useful for periodic functions or functions with periodic components.
Trigonometric approximation is closely tied to Fourier series and the Discrete Fourier Transform (DFT). Fourier series is a representation of a periodic function as an infinite sum of sine and cosine functions. The Discrete Fourier Transform (DFT) is the discrete analogue - it is a numerical algorithm that computes the Fourier coefficients of a discrete sequence of data points. It is widely used to transform a time-domain signal into its frequency-domain representation.

In this chapter we will start by discussing trigonometric approximation in general, and then continue with Fourier series and DFT.

20.1 Trigonometric Approximation

A trigonometric polynomial of degree n is a function of the form

$$a_0 + \sum_{k=1}^{n} (a_k \cos kx + b_k \sin kx) \tag{20.1}$$

with coefficients $a_0, a_1, \ldots, a_n, b_1, \ldots b_n$ in $\mathbb{R}$. The degree of the polynomial corresponds to the highest frequency appearing. Note that one can actually rewrite $\sin nx$ and $\cos nx$ as polynomials of degree n in $\cos x$ (hence (20.1) is indeed a polynomial of degree n in $\cos x$).
We define the space of all trigonometric polynomials as

$$\mathcal{T}_n = \{a_0 + \sum_{k=1}^{n} a_k \cos kx + b_k \sin kx \colon a_0, \ldots a_n, b_0, \ldots b_n \in \mathbb{R}\} \tag{20.2}$$

Note that we will work in the space of 2π periodic continuous function for the space containing $\mathcal{T}_n$, which we will denote by $C^{2\pi}$ in the following. The space $C^{2\pi}$ is a subspace of $C(\mathbb{R})$, and also a subspace of $C[0, 2\pi]$ by considering $f \in C[0, 2\pi]$ with $f(0) = f(2\pi)$.
We will consider $C^{2\pi}$ with the sup-norm:

$$\|f\|_\infty = \sup_{0 \le x \le 2\pi} |f(x)| = \sup_{x \in \mathbb{R}} |f(x)|.$$

or the 2-norm

$$\|f\|_2 = \int_{-\pi}^{\pi} f(x)^2 \, \mathrm{d}x. \tag{20.3}$$

We note that the functions

$$\{1, \cos x, \cos 2x, \ldots, \cos nx, \sin x, \ldots \sin nx\} \tag{20.4}$$

are linearly independent, orthogonal and form a basis of $\mathcal{T}_n$. To show this we define the inner product on $C^{2\pi}$ (which induces the norm (20.3))

$$\langle f, g \rangle = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) g(x) \, \mathrm{d}x.$$

Since

$$\int_{-\pi}^{\pi} \cos mx \cos nx \, \mathrm{d}x = \int_{-\pi}^{\pi} \sin mx \sin nx \, \mathrm{d}x = \int_{-\pi}^{\pi} \cos mx \sin nx \, \mathrm{d}x = 0.$$

and

$$\int_{-\pi}^{\pi} \cos^2 mx \, \mathrm{d}x = \int_{-\pi}^{\pi} \sin^2 mx \, \mathrm{d}x = \pi,$$

we see that (20.4) do indeed form an orthogonal basis.

Complex trigonometric polynomials: We can use Euler's identity

$$e^{i\theta} = \cos\theta + i \sin\theta \tag{20.5}$$

to rewrite (20.4) as

$$\sum_{k=-n}^{n} c_k e^{ikx} \tag{20.6}$$

where $c_k \in \mathbb{C}$. This is called the complex form of the Fourier series (while (20.4) is called the real form. This is actually a polynomial of $\mathbb{C}$ since

$$\sum_{k=-n}^{n} c_k e^{ikx} = \sum_{k=-n}^{n} c_k z_k = \sum_{k=0}^{n} c_k z^k + \sum_{k=1}^{n} c_{-k} \bar{z}^k,$$

where $z = e^{ix}$ and $\bar{z} = e^{-ix}$. On the other hand, every sum of the form (20.6) can be written as (20.4) with complex a_k and b_k.

Weierstrass's second theorem states that any function in $C^{2\pi}$ can be approximated by a trigonometric polynomial. In particular

Theorem 20.1 (Weierstrass's Second Theorem). Let $f \in C^{2\pi}$. Then there exists a trigonometric polynomial $p_n \in \mathcal{T}_n$ for every $\varepsilon > 0$, such that

$$\|f - p_n\|_\infty < \varepsilon.$$

We will omit the proof, and refer to [?] for more details (if interested).
Next we focus on the Fourier series, which was first introduced by Fourier in 1922. The truncated Fourier series can be used to approximate periodic functions and we will see that it is indeed the best trigonometric approximation with respect to the 2-norm (20.3).

20.2 The Fourier Series

In 1822, Fourier introduced the concept of Fourier series, which revolutionised the understanding of periodic functions and their representation. One notable application of Fourier series was Fourier's solution to the heat equation. By applying Fourier series to the heat equation, Fourier was able to derive a solution that provided insights into the behaviour of heat distribution over time. His approach involved expressing the initial temperature distribution as a sum of sine and cosine functions and determining the coefficients that best fit the given conditions. This solution, known as the Fourier transform, enabled the understanding and prediction of heat flow and diffusion in various physical systems. For a very good introduction to Fourier analysis we refer to the book of Stein and Shakarchi, see [?].

We will see that the Fourier series is a solution to an approximation problem. The Fourier series of a 2π periodic, bounded and integrable function is given by

$$F(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx) \tag{20.1}$$

where the coefficients are defined by

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kx \,\mathrm{d}x \text{ and } b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kx \,\mathrm{d}x.$$

This representation is well defined if f is Riemann integrable on $[-\pi, \pi]$; then all integrals are well defined and finite since

$$|a_k| \leq \frac{1}{\pi}\int_{-\pi}^{\pi} |f(x)| \,\mathrm{d}x \tag{20.2}$$

and therefore $|a_k| \leq 2\|f\|$ for all $C^{2\pi}$. Using the Euler identity (20.5) we can rewrite (20.1) as

$$F(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} \tag{20.3}$$

with Fourier coefficients

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-ikx} \,\mathrm{d}x \quad -\infty < k < \infty. \tag{20.4}$$

It is called the complex form of the Fourier series, while the first one (20.1) is known as the real form. Obviously we have the relation

$$a_0 = 2c_0, \quad a_k = 2\mathrm{Re}(c_k) \text{ and } b_k = \mathrm{Im}(c_k)$$

Note that if f is real valued, then the corresponding Fourier coefficients are real.
The Fourier transform simplifies considerably if f is an even or an odd function. In particular,

- if f is even then (20.1) reduces to a cosine series, that is $b_k = 0$ for all $k = 1, 2, \ldots$.
- if f is odd then (20.1) reduces to a sine series, that is $a_k = 0$ for all $k = 0, 1, \ldots$.

We denote the partial sum of (20.1) by

$$F_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n} (a_k \cos kx + b_k \sin kx) \tag{20.5}$$

Note that it is not clear that F_n converges. To discuss convergence of Fourier series we need to fix a space for the function we wish to approximate. We will work in $L^2[a, b]$, which is the space of all real-valued functions on $[a, b]$ for which

$$\int_a^b |f(x)|^2 \mathrm{d}x < \infty. \tag{20.6}$$

We call such a function square integrable. Note that we are extremely sloppy here, since the integral in (20.6) is the Lebesgue integral and not the Riemann integral. Lebesgue integrals will be discussed in measure theory, for the time we will only consider Riemann integrable functions f (which are also Lebesgue integrable; note that the opposite does not hold).

Convergence in function spaces is a delicate issue, and there are several ways to answer this question (depending on the error measure of the partial sum)). We will briefly mention two possible measures

1. Point-wise convergence: define the error

$$E_n(x) = f(x) - F_n(x) = f(x) - \sum_{k=0}^{n} c_k e^{ikx}.$$

   For a fixed number n, the function $E_n(x)$ is a function of x. If for every fixed $x \in [a, b]$ we have $\lim_{n\to\infty} E_n(x) = 0$, we say that the Fourier series converges point-wise to f on the interval $[a, b]$.

2. Uniform convergence: the uniform error e_n is the maximum point-wise error over the interval $[a, b]$, that is

$$e_n = \max_{a\le x\le b} \|f(x) - F_n(x)\|.$$

   We say that the Fourier series converges uniformly to $f(x)$ on $[a, b]$ if

$$\lim_{n\to\infty} e_n = 0.$$

   Thus for every given tolerance $\varepsilon > 0$ there exists a number n such that for all $N > n$

$$\max_{a\le x\le b} |f(x) - F_n(x)| < \varepsilon.$$

   Uniform convergence is a much stronger notion of convergence than point-wise. It implies point-wise convergence (but not the other way around).

There has been a lot of research which conditions the function f has to satisfy to make sure that the Fourier series converges to f (in whatever notion of convergence). It is beyond the scope of this module to study them in detail, but we will illustrate the problems by means of examples.

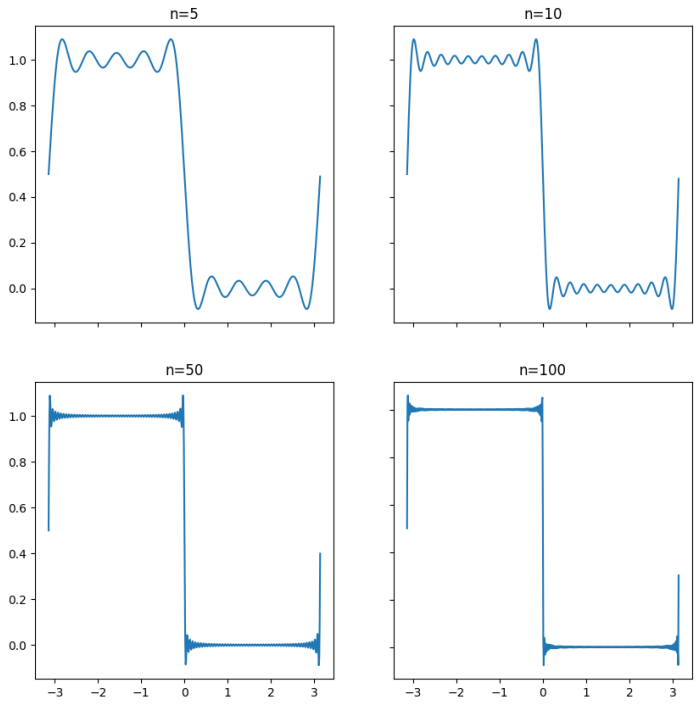


Figure 20.1: Gibbs phenomenon: Fourier approximation of the step function (20.7) for different values of n.

Example 20.2. Consider the step function

$$f(x) = \begin{cases} 1 & -\pi \le x < 0 \\ 0 & 0 \le x < \pi. \end{cases} \tag{20.7}$$

Then the Fourier coefficients are given by

$$a_0 = 1,\ a_k = 0 \text{ for } k \ge 1,\ b_k = -\frac{1}{\pi k}\left[1 - (-1)^k\right] \text{ for } k \ge 1.$$

Therefore the Fourier series is given by

$$F(x) = \frac{1}{2} - \frac{2}{\pi}\sum_{k=1}^{\infty} \frac{1}{2k-1}\sin(2k-1)x.$$

Figure 20.1 shows the partial sums F_n for $n = 5, 10, 50$ and $n = 100$. We observed the convergence of F_N for fixed values of $x \in (-\pi, 0) \cup (0, \pi)$. But F_N does obviously not converge at the discontinuity $x = 0$. As N increases the accuracy of F_N increases and the ripples move closer to the discontinuity. However, the size of these ripples does not decrease to zero. This phenomena is known as the Gibbs's phenomena - we have only point-wise convergence of F_n to f, but not uniform convergence.

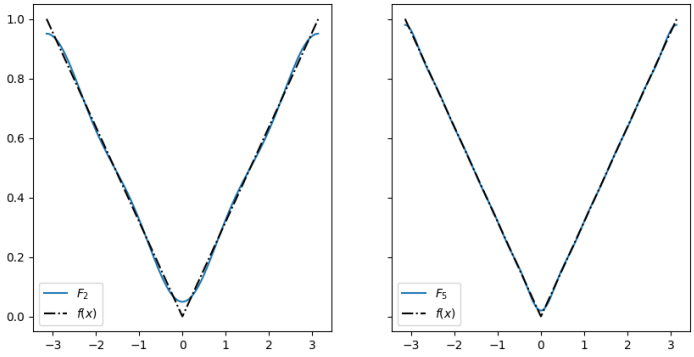


Figure 20.2: Fourier approximation of (20.8) for $N = 2$ and $N = 5$.

Example 20.3. As a second example we consider

$$f(x) = \frac{|x|}{\pi} \quad \pi \leq x \leq \pi. \tag{20.8}$$

This function is even, and therefore does not contain any sine terms. Its Fourier series is given by

$$F(x) = \frac{1}{2} - \frac{4}{\pi^2} \sum_{k=1}^{\infty} \frac{1}{(2k-1)^2} \cos((2k-1)x).$$

We observe that the truncated Fourier series approximates the function f much better, see Figure 20.2.

The convergence behaviour of the partial sum F_N does depend on the smoothness of the function f. We don't go into further details (as this is beyond the scope of this module) and refer to [?] and ??? for further details.

We conclude by discussing how the truncated Fourier series (20.1) relates to best approximation problems. Note that the coefficients $a_0, a_1, \ldots a_n$ and $b_1, \ldots b_n$ are computed by minimising

$$K(a_0, \ldots a_n, b_1, \ldots b_n) = \int_{-\pi}^{\pi} \left[f(x) - F_n(x)\right]^2 \, \mathrm{d}x.$$

Calculating the optimality conditions wrt to a_l, $k = 0, \ldots n$ and b_k, $k = 1, \ldots n$ we obtain:

$$\frac{\delta K}{\delta a_k} = \int_{-\pi}^{\pi} 2\left[f(x) - F_n\right] \cos kx \, \mathrm{d}x = 0$$
$$\frac{\delta K}{\delta b_k} = -\int_{-\pi}^{\pi} 2\left[f(x) - F_n\right] \sin kx \, \mathrm{d}x = 0.$$

The relations hold when

$$\int_{-\pi}^{\pi} f(x) \cos kx \, \mathrm{d}x = a_k \int_{-\pi}^{\pi} \cos^2 kx \, \mathrm{d}x$$
$$\int_{-\pi}^{\pi} f(x) \sin kx \, \mathrm{d}x = b_k \int_{-\pi}^{\pi} \sin^2 kx \, \mathrm{d}x.$$

We conclude by briefly discussing that the truncated Fourier series is the solution of the best approximation problem using the 2-norm.

Theorem 20.4. Let $f \in L^2[a,b]$ and let c_k denote the Fourier coefficients of the complex form (20.4) of the Fourier series $F_n = \sum_{k=0}^{n} c_k w^k$ with $w = e^{ix}$. Then for every other sequence d_k we have

$$\|f - \sum_{k=0}^{n} c_k w^k\|_2^2 \leq \|f - \sum_{k=0}^{n} d_k w^k\|_2^2$$

Proof. We compute

$$\begin{aligned}
\|f - \sum_{k=0}^{n} a_k w^k\|_2^2 &= \langle f - \sum_{j=0}^{n} a_k w^k, f - \sum_{k=0}^{n} a_k w^k \rangle \\
&= \langle f, f \rangle - 2\sum_{k=0}^{n} a_k \langle f, w^k \rangle + \sum_{j=0}^{n} a_k^2 + \sum_{k=0}^{n} c_k^2 - \sum_{k=0}^{n} c_k^2 \\
&= \langle f, f \rangle - 2\sum_{k=0}^{n} a_k c_k + \sum_{k=0}^{n} a_k^2 \\
&= \langle f, f \rangle - \sum_{k=0}^{n} c_k^2 + \sum_{k=0}^{n} (a_k - c_k)^2 \\
&= \|f - \sum_{k=0}^{n} c_k w^k\|_2^2 + \sum_{k=0}^{n} (a_k - c_k)^2.
\end{aligned}$$

Since the second term is non-negative, the statement holds. □

In order to obtain a Fourier representation for a function that is not periodic, it seems desirable to consider a function with period L and take the limit $L \to \infty$. Then one can derive the Fourier transform of a continuous function f

$$\mathcal{F}(\omega) = \int_{-\infty}^{\infty} e^{-i\omega x} f(x)\, \mathrm{d}x. \tag{20.9}$$

The Fourier transformation (20.9) is an important mathematical tool in differential equations, you can learn more about its applications in the module MA433 Fourier Analysis, as well as signal and image processing.

20.3 The Discrete Fourier Transform

In 1965 J. Cooley and J. Tukey published a paper presenting an efficient method to compute the Fourier transform for a discrete signal. Some people claim that their approach was already known to Gauss in the mid 1800s; since the basic idea of the algorithm was discussed in one of Gauss' unpublished works (which appeared post mortem in 1866). Their work made an important contribution to signal processing, since computers and digital processing systems work with finite sums only.
We recall the Fourier coefficients

$$c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx}\, \mathrm{d}x.$$

We now approximate the integral using the trapezoidal rule. Let n be a natural number, and define $x_j = jh$ where $h = \frac{2\pi}{n}$. Since f is 2π-periodic we obtain that

$$c_k \approx \frac{1}{n} \sum_{k=0}^{n-1} f(x_k) e^{-ikx_k}.$$

Then the discrete Fourier transform of a vector $\mathbf{f} = (f_0, \ldots, f_n)$ is given by

$$\hat{f}_k = \sum_{j=0}^{n-1} w^{-kj} f_j \quad 0 \leq k \leq n-1. \tag{20.1}$$

where $w_n = e^{\frac{2\pi i}{n}}$. We refer to n as the order of the discrete Fourier transform.
Equation (20.1) can also be written in matrix notation $\hat{\mathbf{f}} = \mathbf{W}_n \mathbf{f}$ with

$$\mathbf{f} = \begin{pmatrix} f(0) \\ \vdots \\ f(n-1) \end{pmatrix} \text{ and } \hat{\mathbf{f}} = \begin{pmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(n-1) \end{pmatrix}$$

with

$$\mathbf{W}_n = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & w_n^{-1} & w_n^{-2} & \ldots & w_n^{-(n-1)} \\ 1 & w_n^{-2} & w_n^{-3} & \ldots & w_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & w_n^{-(n-1)} & w_n^{-2(n-1)} & \ldots & w_n^{-(n-1)^2} \end{pmatrix}$$

The matrix $\mathbf{W}_n$ is called the n^{th} order DFT matrix. We state a few properties of the DFT

Theorem 20.5. The DFT has the following properties

- Linear Shift: $\hat{\mathbf{f}}(k-j) = \hat{\mathbf{f}}(k) e^{-\frac{2\pi i j}{n}}$
- Frequency Shift: $\widehat{\mathbf{f}(k) e^{\frac{2\pi i j}{n}}} = \hat{\mathbf{f}}(k-j)$.

Next we compute the inverse DFT, which is the discrete analogue to the inverse continuous Fourier transform:

$$f(n) = \sum_{k=0}^{n-1} \hat{f}(k) w^{jk}.$$

The Fast Fourier Transform

Even though the DFT provided important information about the frequency components, it was rarely used in practical applications until the 1960's. This changed, with the development of modern computers as well as the discovery of the Fast Fourier Transform, by Cooley and Tukey.

The DFT of a signal evaluated at n points, requires n^2 multiplications and $n(n-1)$ additions. The FFT algorithm reduces this number to something proportional to $n \log_2 n$ multiplications if $n = 2^k$.
To reduce the computational complexity one has to look at the even and odd entries separately. Furthermore we note that if $n = 2M$, $M \in \mathbb{N}$ then

$$w_n^2 = e^{-\frac{2\pi i}{n}} = e^{-\frac{2\pi i}{M}} = w - M.$$

Next we define a vector of even and odd entries, that is

$$\begin{aligned}\mathbf{a}(k) &= \mathbf{f}(2k) \qquad &\text{for } k = 0, 1, \dots M-1\\ \mathbf{b}(k) &= \mathbf{f}(2k+1) \qquad &\text{for } k = 0, 1, \dots M-1.\end{aligned}$$

Then $\mathbf{f}(k) = (\mathbf{a}(0), \mathbf{b}(0), \mathbf{a}(1), \mathbf{b}(1), \dots, \mathbf{a}(M-1), \mathbf{b}(M-1))$.
We compute

$$\begin{aligned}\hat{\mathbf{f}}(k) &= \sum_{j=0}^{n-1} \mathbf{f}(k) w_n^{jk}\\ &= \sum_{j=0}^{M-1} \mathbf{f}(2k) w_n^{2jk} + \sum_{k=0}^{M-1} \mathbf{f}(2k+1) w_n^{2j(k+1)}\\ &= \sum_{j=0}^{M-1} \mathbf{f}(2k) (w_n^2)^{jk} + \sum_{k=0}^{M-1} \mathbf{f}(2k+1) (w_n^2)^{j(k+1)}\\ &= \sum_{j=0}^{M-1} \mathbf{a}(k) w_n^{jk} + w_n^k \sum_{k=0}^{M-1} \mathbf{b}(k) w_n^{jk}\\ &= \hat{\mathbf{a}}(k) + w_n^k \hat{\mathbf{b}}(k),\end{aligned}$$

where $\hat{\mathbf{a}}(n)$ and $\hat{\mathbf{b}}(n)$ are the DFTs of $\mathbf{a}$ and $\mathbf{b}$.
If $0 \le k \le M-1$ then $\hat{\mathbf{f}}(k) = \hat{\mathbf{a}}(k) + w_n^k \hat{\mathbf{b}}(k)$. If $M \le k \le N-1$ then

$$\begin{aligned}\hat{\mathbf{f}}(k) &= \hat{\mathbf{a}}(k-M) + w_n^{(k-M)} \hat{\mathbf{b}}(k-M)\\ &= \mathbf{a}(k) + w_n^{-M} w_n^k b(k)\\ &= \mathbf{a}(k) - w_n^k b(k)\end{aligned}$$

where we used that $w_n^{-M} = \left(e^{-\frac{2\pi i}{n}}\right)^{-M} = e^{\frac{2\pi i M}{n}} = e^{2\pi i} = -1$. We see that we only have to compute $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ (each takes $\frac{n^2}{4}$ multiplications) and furthermore and additional $\frac{n}{2}$ multiplications to compute $\hat{\mathbf{f}}(n) w_n^k$. This cuts the computational cost by half. That's not a big improvement. A much more significant cut comes when choosing $n = 2^j$, then one can use the above argument recursively when computing $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$.

20.4 The Discrete Cosine Transform

The discrete cosine transform (DCT) was originally developed by Nasir Ahmed in the early 1970s. Nowadays it plays a crucial role in transforming spatial image data into frequency domain coefficients. In the realm of image compression, the DCT has become a cornerstone algorithm in popular image formats like JPEG, as it enables substantial data reduction without significant loss of visual quality. By dividing an image into smaller blocks and applying the DCT to each block, image compression techniques can store and transmit visual data more efficiently, making it a fundamental tool in the digital imaging landscape.

Different versions of the DCT can be found in the literature. We will discuss the so-called cosine-I transform, which is used in JPEG and MPEG video compression. The advantages of DCT compared to DFT are

- the basis functions and the series itself are all real-valued
- the basis functions have different symmetries (and images are usually not periodic).

The DCT is given by

$$\hat{\mathbf{f}}_C(w) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} f(k) \cos \frac{(2k+1)wk}{2n} \tag{20.1}$$

for $0 \leq w \leq n$ with coefficients $c(w) = 1$ if $w = 0$ and $c(w) = \sqrt{2}$ otherwise. Again we can write this in matrix vector form $\hat{\mathbf{f}}_C = \mathbf{C}_n \mathbf{f}$ with

$$\mathbf{C}_n = \sqrt{\frac{2}{n}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \dots & \frac{1}{\sqrt{2}} \\ \cos \frac{\pi}{2n} & \cos \frac{3\pi}{2n} & \dots & \frac{(2n-1)\pi}{2n} \\ \cos \frac{2N}{2n} & \cos \frac{6\pi}{2n} & \dots & \cos \frac{2(2n-1)\pi}{2n} \\ \vdots & \vdots & \dots & \vdots \\ \cos \frac{(n-1)\pi}{2n} & \cos \frac{3\pi(n-1)}{2n} & \dots & \cos \frac{(2n-1)(n-1)\pi}{2n} \end{pmatrix}$$

Note that the matrix $\mathbf{C}_n$ is orthogonal, therefore $\mathbf{C}_n^{-1} = \mathbf{C}_n^T$.
The DCT corresponds to the best approximation in the 2-norm $\|f - p_n\|_2$ among all cosine polynomials p_n from

$$p_n(x) = \frac{1}{\sqrt{n}} c_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{n-1} c_k \cos \frac{k(2x+1)\pi}{2n}$$

DCT is mostly used in image compressions. Images are 2D, hence we need to extend (20.1) to 2D. Given an input matrix $\mathbf{X}$, the DCT is first applied to the rows of $\mathbf{X}$, and then again to the rows of the resulting matrix, that is

$$\hat{\mathbf{F}}_c = \mathbf{C}_n(\mathbf{C}_n\mathbf{X})^T = \mathbf{C}_n\mathbf{X}\mathbf{C}_n^T.$$

Image quality is usually given in dots per inch (dpi). Given for example a 15 in^2 image with 400 dpi, we have $400 \times 400 \times 15 = 2,400,000$ pixels. Each pixel corresponds to 3 byte of data (encoding red, green and blue), therefore we would need $7,200,000$ bytes, about 7 MB, to store a single image. That's obviously too much. Several compression techniques have been developed to store images at a much smaller size.
Compression can be classified as lossless or lossy. In lossless compression (such as zip or tar files), one regains the original data without loss after compression. This is not the case in lossy compression, where some information is lost/discarded. The reconstructed images are therefore of lower quality than the original. However, much better compression rates can be obtained.
JPEG, one of the most used image formats, is based on the following compression

1. Level shifting
2. Apply DCT to 8×8 image blocks
3. Low pass filter
4. Apply IDCT to the transformed 8×8 matrix
5. Round the entries and shift the values again

Example: We will briefly discuss how these 8×8 blocks are compressed. We consider an 8×8 black and white picture. Each entry in the matrix corresponds to the respective value of the pixel. The values range from 0 (black) to 256 (white). In particular we consider

$$\mathbf{f} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

The (rounded) output of the DCT is

$$\hat{\mathbf{f}}_C = \begin{pmatrix} 461 & -168 & -15 & 30 & -31 & 9 & 1 & -3 \\ -194 & -2 & 38 & 5 & 7 & 4 & 6 & -6 \\ 32 & 44 & 11 & -22 & 15 & -11 & -6 & 4 \\ -3 & -28 & -1 & -0 & -3 & 7 & 4 & -3 \\ 2 & 11 & 1. & 1 & 4 & -5 & 1 & -3 \\ -1 & -4 & 2. & 3 & 1 & -3 & 0 & 3 \\ -10 & 6 & 4. & -9 & 8 & -1 & -6 & 8 \\ 8 & -2 & -3. & 1 & 3 & 4 & -2 & -1 \end{pmatrix}$$

Next we use 'so-called' quantisation; thereby each value of the $\hat{\mathbf{f}}$ is divided by the respective value in the quantisation matrix $\mathbf{Q}$, in particular

$$\hat{\mathbf{f}}_c^Q[i,j] = \text{round}(\frac{\hat{\mathbf{f}}_c[i,j]}{\mathbf{Q}[i,j]})$$

Let $\mathbf{Q}$ be given by

$$\mathbf{Q} = \begin{pmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{pmatrix}$$

We multiply the rounded matrix by the values of $\mathbf{Q}$ again and obtain

$$\begin{pmatrix} 459 & -165 & -14 & 27 & -22 & 0 & 0 & 0 \\ -190 & 0 & 38 & 5 & 7 & 0 & 0 & 0 \\ 28 & 36 & 11 & -13 & 15 & 0 & 0 & 0 \\ 0 & -22 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

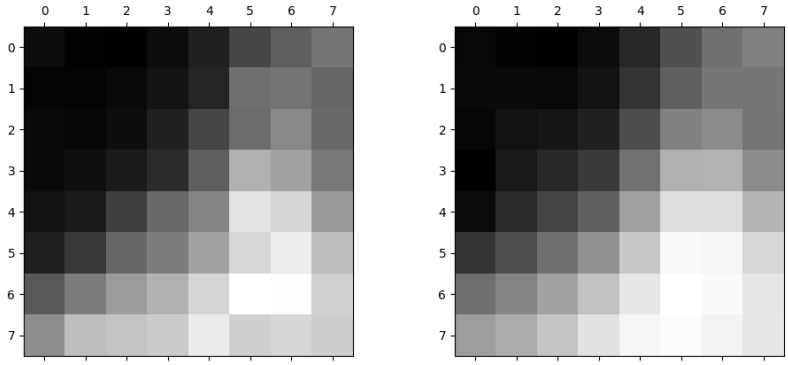


Figure 20.3: Original image (left) and reconstructed one (right)

Reconstructed image

$$\begin{pmatrix} 13 & 11 & 10 & 15 & 24 & 37 & 54 & 60 \\ 14 & 14 & 14 & 17 & 27 & 58 & 57 & 56 \\ 13 & 17 & 19 & 23 & 40 & 58 & 65 & 56 \\ 11 & 20 & 26 & 33 & 55 & 79 & 80 & 65 \\ 15 & 27 & 37 & 48 & 73 & 97 & 97 & 81 \\ 30 & 42 & 54 & 67 & 89 & 108 & 107 & 95 \\ 54 & 63 & 74 & 87 & 101 & 111 & 108 & 101 \\ 72 & 78 & 88 & 99 & 107 & 109 & 106 & 101 \end{pmatrix}$$

21

Wavelets

Fourier representations, including the Fourier series and the DFT, are a pillar of signal processing, but they have a fundamental limitation: Fourier coefficients capture global fluctuations of a signal, and hence do not provide information about events that are localised in time or space.

Wavelets are specifically designed to analyse signals at different scales and resolutions. Unlike Fourier analysis, wavelets use a windowed approach where they analyse small, localised portions of the signal at different scales. This enables wavelets to efficiently capture both high and low-frequency components in regions where they are most concentrated while ignoring the rest of the signal, making them ideal for analysing signals with localised characteristics.
The story of wavelets starts in the early 20th century with the works of the Hungarian mathematician Alfred Haar. In 1909, Haar introduced the concept of orthogonal wavelets, particularly the Haar wavelet, which served as a simple but effective wavelet basis. Haar wavelets are piece-wise constant functions, and their simplicity made them an excellent starting point for further developments in wavelet theory.
Fast forward to the late 20th century, the field of wavelets experienced a major breakthrough with the contributions of Ingrid Daubechies, a Belgian physicist and mathematician. In the mid-1980s, Daubechies introduced a family of wavelets, now known as Daubechies wavelets. These wavelets are compactly supported and have a specific number of vanishing moments, making them highly effective for signal compression and denoising. Daubechies' work laid the groundwork for the practical application of wavelets in various fields, including image and signal processing.

21.1 The Continuous and Discrete Wavelet Transform

We start by stating the formal definition of a wavelet

Definition 21.1 (Wavelet). A wavelet is a function $\psi \in L^2(\mathbb{R})$ which satisfies the so-called admissibility condition

$$\int_{-\infty}^{\infty} \frac{|\hat{\psi}(w)|^2}{|w|} \mathrm{d}w < \infty \tag{21.1}$$

where $\hat{\psi}$ is the Fourier transform of ψ.

Condition (21.1) ensures the existence of the inverse wavelet transform. Furthermore it implies that $\hat{\psi}(w) \to 0$ as $w \to \infty$ and that $\hat{\psi}(0) = 0$. The latter implication is equivalent to $\int_{-\infty}^{\infty} \psi(w)\,\mathrm{d}w = 0$ hence ψ must be an oscillatory function with mean zero. One could imagine wavelets to look like small

oscillatory decaying waves, which also motivates the name.
The function ψ is often called the mother wavelet (we will however not use this convention in this lecture). One wishes to localise the information - that is to zoom in to fine details, use coarser functions to approximate global structures, ... To do so we define

$$\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k) \tag{21.2}$$

for $j, k \in \mathbb{Z}$. Here the prefector 2^j corresponds to a dilation/compression in x, while the parameter k to a translation. Then given $\psi \in L^2(\mathbb{R})$ and $\psi_{j,k} \in L(\mathbb{R})$ of form (21.2), the continuous wavelet transform is defined as

$$\mathcal{W}[f] = \int_{-\infty}^{\infty} f(x)\bar{\psi}_{j,k}\,\mathrm{d}x. \tag{21.3}$$

The function $\bar{\psi}_{j,k}$ plays the same role as the e^{-iwx} in the Fourier transform (20.9). The wavelet transform is similar to the Fourier transform, however it is more localised (due to the translation allowing to resolve certain regions and the dilation to resolve fine features better).

21.2 Haar wavelets

The Haar wavelet is the simplest wavelet and the most useful to illustrate the main ideas of wavelet theory. It is defined as

$$\psi(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{21.1}$$

The Haar wavelet has compact support, that is supp $\psi(x) = [0, 1]$. This would not be a great advantage to Fourier series (you only have local support), however, wavelet theory works with scaled, dilated and translated copies of wavelet functions. For the Haar wavelet we consider

$$\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k),$$

for $j, k \in \mathbb{Z}$. The prefactor $2^{\frac{j}{2}}$ is the scaling factor, the term 2^j the dilation factor and k the translation.

1. Scaling: It stretches the graph for $j > 0$, and compresses it for $j < 0$ in the y/vertical direction.
2. Dilation: this prefactor stretches the graph in the horizontal/x-direction for $j < 0$, and compressing it for $j > 0$.
3. Translation: shifting the graph in the x-direction to the right for $k > 0$ or left for $k < 0$.

The set of Haar wavelets

$$\{\psi_{j,k} \colon j, k \in \mathbb{Z}\}.$$

form an orthonormal basis of $L^2(\mathbb{R})$. They are orthogonal, since

$$\langle \psi_{j,k}, \psi_{j',k'} \rangle = \int_{-\infty}^{\infty} \psi_{j,k}\psi_{j',k'}\,\mathrm{d}x = \delta_{j,j'}\delta_{k,k'} = \begin{cases} 1 & j = j', k = k' \\ 0 & \text{otherwise.} \end{cases}$$

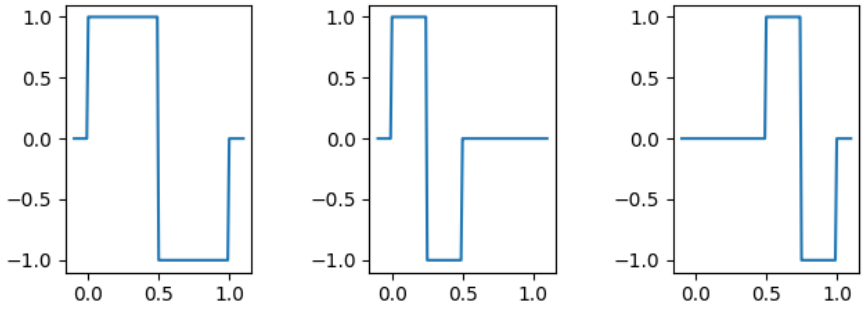


Figure 21.1: The Haar wavelet function ψ as well as dilated and shifted versions of it.

Since they are a basis every function $f \in L^2(\mathbb{R})$ has a Haar wavelet series expansion

$$f = \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} w_{j,k} \psi_{j,k}$$

with coefficients

$$w_{j,k} = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f \psi_{j,k} \, \mathrm{d}x.$$

The Haar wavelet gives a local basis.

21.3 Multiresolution Analysis

The concept of multiresolution analysis was introduced and developed by the French mathematicians Stephane Mallat and Yves Meyer in the 1980ties, providing a framework for the systematic creation of wavelets.

Definition 21.2. A multiresolution analysis (MRA) is a family of closed subspaces $V_j \in L^2(\mathbb{R})$ that satisfies the following properties

1. $V_j \subset V_{j+1}$ for all $j \in \mathbb{Z}$
2. $f(x) \in V_j \leftrightarrow f(2x) \in V_{j+1}$ for all $j \in \mathbb{Z}$.
3. $\bigcup_j V_j$ is dense in $L^2(\mathbb{R})$.
4. $\bigcap_j V_j = \{0\}$.
5. There exists a scaling function $\varphi \in V_0$, such that $\varphi(x-k)$ is a Riesz basis for V_0.

First we discuss what a Riesz basis is.

Definition 21.3. A linear independent set $\{\varphi_j\}_j$ is a Riesz basis of a Hilbert space H, if for all $f \in H$ with $f = \sum_j c_j \varphi_j$ we have

$$A\|f\|^2 \leq \sum_k |c_k|^2 \leq B\|f\|^2$$

with constants A, B not depending on f.

The Riesz basis ensures numerically stable approximations. Consider an approximation $\tilde{f}$ of f, with $\tilde{f} = \sum \tilde{c}_k \varphi_k$ and $f = \sum c_k \varphi_k$, then

$$A\|f - \tilde{f}\|^2 \leq \sum |c_k - \tilde{c}_k|^2 \leq B\|f - \tilde{f}\|^2.$$

We see that small errors in the approximation give small errors in the coefficients.

Let us return to the definition of an MRA. The first condition ensures that functions from V_{j+1} contain more information/details than functions from V_j. The second gives a connection between the different scales. The third condition ensures that any function in $L^2(\mathbb{R})$ can be approximated well, while the fourth condition states that only the zero function can be approximated on the coarse scale.

The MRA is completely determined by the scaling function. φ (but not the other way around). For a given φ we define V_0

$$V_0 = \{f(x) = \sum_{k\in\mathbb{Z}} c_k \varphi_{0,k} = \sum_{k\in\mathbb{Z}} c_k \varphi(x-n) \colon c_k \in \ell^2(\mathbb{Z}\}.$$

The function φ forms an orthonormal basis for V_0, the dilated, scaled and transformed the scaling function $\varphi_{j,k}$ defined as

$$\varphi_{j,k}(x) = 2^{\frac{j}{2}} \varphi(2^j x - k).$$

is a basis for V_j.
Note that the scaling factor $2^{\frac{j}{2}}$ ensures that all scaling functions have the same norm, that is $\|\varphi\| = \|\varphi_{j,k}\|$ for every $j, k \in \mathbb{Z}$. The functions $\varphi_{j,k}$ also form a Riesz basis for V_j (after scaling them), and therefore every $f_j \in V_j$ can be written as

$$f_j(x) = \sum_k c_{j,k} \varphi_{j,k}(x).$$

The MRA already poses quite strong assumptions on the scaling function. However, there are some other properties which are desirable. For example, that the scaling function should have local support and that is normalised, that is

$$\int_{-\infty}^{\infty} \varphi(x)\,\mathrm{d}x = 1.$$

Since the scaling function belongs to V_0, it also belongs to V_1. Furthermore $\varphi_{1,n} = \sqrt{2}\varphi(2x-n)$ is an orthonormal basis of V_1 and therefore

$$\varphi(x) = \sqrt{2} \sum_k c_k \varphi(2x-k) \tag{21.1}$$

for some coefficients $c_k = \langle \varphi, \varphi_{1,k} \rangle$ and $\sum |c_k|^2 = 1$. Equation (21.1) is known as the dilation equation. The MRA allows us to construct a simple orthonormal basis for $L^2(\mathbb{R})$. Let $\{V_j\}_j$ be an MRA. Since $V_j \subset V_{j+1}$ we define W_j as the orthogonal complement of V_j in V_{j+1} for every $j \in \mathbb{J}$. Therefore

$$\begin{aligned} V_{j+1} &= V_j \oplus W_j \\ &= [V_{j-1} \oplus W_{j-1}] \oplus W_j \\ &= \dots \\ &= V_0 \oplus W_0 \oplus W_1 \oplus \dots \oplus W_j \end{aligned}$$

and $V_n \perp W_m$ for $m \neq n$. Since $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$, one can take the limit and deduce that $V_0 + \bigcup_{j\in\mathbb{Z}} W_j = L^2(\mathbb{R})$. Furthermore

$$\begin{aligned} V_0 &= V_{-1} \oplus W_{-1} \\ &= [V_{-2} \oplus W_{-2}] \oplus W_{-1} \\ &= \dots \\ &= V_{-m} \oplus W_{-m} \oplus \cdots \oplus W_{-1}. \end{aligned}$$

Since $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$, we conclude $V_{-j} = \{0\}$. In the limit $j \to \infty$ we have that

$$\bigoplus_{j\in\mathbb{Z}} W_j = L^2(\mathbb{R}). \tag{21.2}$$

One can show that the MRA definition implies that the spaces W_j are also scaled versions of W_0. Moreover there exists a function ψ in W_0, such that $\{\psi(x-k)\colon k \in \mathbb{Z}\}$ is an orthonormal basis for W_0. Then

$$\{\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k)\colon k \in \mathbb{Z}\}$$

is an orthonormal basis for W_0. The $\psi_{j,k}$ are the orthonormal wavelet basis.

We already introduced the Haar wavelet (21.1) in the previous subsection. In this case the scaling function is given by

$$\varphi(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{21.3}$$

The scaling function satisfies the dilation equation

$$\varphi(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} c_k \psi(2x-k)$$

with coefficients

$$c_k = \sqrt{2}\int_{-\infty}^{\infty} \varphi(x)\varphi(2x-k)\,\mathrm{d}x$$

For $\varphi = \mathbb{1}_{[0,1)}$ we calculate $c_0 = c_1 = \frac{1}{\sqrt{2}}$ and $c_k = 0$ for $k \neq 0$ or 1. Therefore the dilation equation becomes

$$\varphi(x) = \varphi(2x) + \varphi(2x-1).$$

Which gives us the Haar wavelet (21.1).

The Haar wavelet is of course not the only wavelet. Different choices of scaling functions result in different wavelets, such as the Meyer wavelet, the Mexican hat wavelet and others. For more information on different wavelet basis see REF MISSING.

We conclude with an example, illustrating why the Wavelet Transform is better suited for signals that are not periodic. Consider the piece-wise constant function f

$$f(x) = \begin{cases} 9 & x \in [0, 0.25) \\ 1 & x \in [0.25, 0.5) \\ 2 & x \in [0.5, 0.75) \\ 0 & x \in [0.75, 1). \end{cases}$$

We wish to find an approximation in $V_1 = V_0 \oplus W_0$, which is spanned by the basis vectors Haar wavelet vectors $\varphi_0, \psi_{0,0}, \psi_{1,0}, \psi_{1,1}$. We see that

$$\begin{pmatrix} 9 \\ 1 \\ 2 \\ 0 \end{pmatrix} = 3 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + 2 \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} + 4 \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}$$

which we can write in the more compact form

$$\mathbf{y}_w = \mathbf{W}_4 c \text{ with } \mathbf{W}_4 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 - 1 & \end{pmatrix}$$

and $\mathbf{y}_w = (9, 1, 2, 0)^T$ and wavelet coefficients $\mathbf{c} = (3, 2, 4, 1)^T$.
If we compute the discrete Fourier transform (up to index 4) we obtain

$$\mathbf{y}_f = \mathbf{F}_4 a \text{ with } \mathbf{F}_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{pmatrix}$$

and $\mathbf{y}_f = (f(0), f(\frac{\pi}{2}), f(\pi), f(\frac{3\pi}{2}))$. The Fourier matrix $\mathbf{F}_4$ is full and complex, while the wavelet matrix is orthogonal, sparse and with real entries. To inverse of $\mathbf{W}_4$ is given by

$$\mathbf{W}_4^{-1} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 2 & -2 & 0 & 0 \\ 0 & 0 & -2 & -2 \end{pmatrix}$$

for the inverse Fourier matrix we have to transpose its complex conjugate

$$\mathbf{F}_4^{-1} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & (-i) & (-i)^2 & (-i)^3 \\ 1 & (-i)^2 & (-i)^4 & (-i)^6 \\ 1 & (-i)^3 & (-i)^6 & (-i)^9 \end{pmatrix}$$

The two inverse matrices have the same structure as the originals. If they can be transformed quickly, they can be inverted quickly. We have seen that we can reduce the complexity of calculating the Fourier coefficients to $\mathcal{O}(n \log n)$. The wavelet coefficients can be, however, calculated even more efficiently - in $\mathcal{O}(n)$. For more information on Fourier vs. Wavelets we refer to the article by Gilbert Strang, see [?]

22

Best-Fit Subspaces and Singular Value Decomposition

In the last chapter we want to discuss how to find the best low rank approximation of a given matrix $\mathbf{A}$. The matrix $\mathbf{A}$ corresponds to N data points in $\mathbb{R}^d$, and we wish to minimise the sum of squares of the perpendicular distance of the points to an n dimensional subspace. We will see later that the best-fitting n dimensional subspace can be found by n applications of the best fitting line algorithm. This procedure also gives the singular value decomposition of the matrix. This chapter follows [?].

We recall the singular value decomposition (SVD) of an $N \times d$ matrix $\mathbf{A}$ - which corresponds to the N points in d dimensional space - is a factorisation of $\mathbf{A}$ into a product of three matrices. It is given by

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{22.1}$$

where $\mathbf{D} \in \mathbb{R}^{N\times d}$ is a diagonal matrix, and $\mathbf{U} \in \mathbb{R}^{N\times N}$, $\mathbf{V} \in \mathbb{R}^{d\times d}$ are orthogonal matrices. The SVD is useful in many application - one obtains the rank, the singular values or can compute the inverse if $\mathbf{A}$ is square and invertible. The SVD also gives the best n-rank approximation to $\mathbf{A}$ for any n.

Let us start with finding the best fitting line through the origin given a set of points $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$. Consider the point $\mathbf{x}_k = (x_{k1}, x_{k2}, \ldots x_{kd})$. We wish to minimise the square distance $\mathbf{x}_k$ to the line through the origin (which can be represented by the vector $\mathbf{v}$). We wish to minimise

$$(\text{distance of the point to line})^2 = x_{k1}^2 + x_{k2}^2 + \ldots x_{kd}^2 - (\text{length of the projection})^2.$$

For N points one would minimise

$$\sum_{k=1}^{N} x_{k1}^2 + x_{k2}^2 + \ldots x_{kd}^2 - (\text{sum of projections})^2. \tag{22.2}$$

Since the point $\mathbf{x}_k$ is given (and constant), it does not change the solution of the minimisation problem. And actually we can maximise over the sum of projections.

But how does this relate to the SVD (22.1)? The columns of V, also known as the right singular vectors of $\mathbf{A}$, are the unit vectors defining the best fitting lines as we will see in the following.
To construct the singular vectors of the $N \times d$ matrix $\mathbf{A}$, we consider the best fit problem (22.2) again. Let $\mathbf{a}_k$ be the k-th row of the matrix $\mathbf{A}$. The length of the projection onto a line, parameterised by a unit

vector $\mathbf{v}$ is $|\mathbf{a}_k \cdot \mathbf{v}|$. Hence (22.2) corresponds to maximising $|\mathbf{A}\mathbf{v}|^2$.
The first singular vector is defined as

$$\mathbf{v}_1 = \arg\max_{|\mathbf{v}|=1} |\mathbf{A}\mathbf{v}|.$$

We note that $-\mathbf{v}_1$ is as good as $\mathbf{v}_1$. However, we will not worry about this and pick one. The value $\sigma_1(\mathbf{A}) = |\mathbf{A}\mathbf{v}|$ is the first singular value of $\mathbf{A}$. Note that σ_1^2 is the sum of squares of the projections of $\mathbf{x}_i$ to the line through the origin described by $\mathbf{v}_1$. If the data points $\mathbf{x}_i$ would be on or very close to a line, it should be given by $\mathbf{v}_1$. If the points are however lie close to a low dimensional subspace, for example a plane how can we find it.
The greedy approach takes the first singular vector as the basis for the 2-dimensional subspace and looks for another unit vector $\mathbf{v}_2$ perpendicular to $\mathbf{v}_1$ that maximises $|\mathbf{A}\mathbf{v}|^2$. In particular

$$\mathbf{v}_2 = \arg\max_{\mathbf{v}\perp\mathbf{v}_1, |\mathbf{v}|=1} |\mathbf{A}\mathbf{v}|.$$

This defines the second singular vector and $\sigma_2(\mathbf{A}) = |\mathbf{A}\mathbf{v}_2|$ is the second singular value of $\mathbf{A}$. This way we can construct a sequence of orthogonal vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$, which stops when we find a vector $\mathbf{v}_r$ such that

$$\arg\max_{\mathbf{v}\perp\mathbf{v}_1, \mathbf{v}_2 \ldots \mathbf{v}_r, |\mathbf{v}|=1} |\mathbf{A}\mathbf{v}| = 0.$$

The following theorem states that the greedy algorithm does indeed construct the best subspaces.

Theorem 22.1. Let $\mathbf{A} \in \mathbb{R}^{N\times d}$ and denote by $\mathbf{v}_1, \ldots \mathbf{v}_r$ the singular vectors of the matrix $\mathbf{A}$. Let V_k, $1 \le k \le r$ denote the subspace spanned by $\mathbf{v}_1, \ldots \mathbf{v}_r$. Then V_n, $1 \le n \le r$ is the best-fit n-dimensional subspace for $\mathbf{A}$.

Proof. We show the statement by induction.

- It is obviously true for $n = 1$.
- $n = 2$: Let W be the best fit 2-dimensional subspace for $\mathbf{A}$, with basis vectors $\mathbf{w}_1, \mathbf{w}_2$. Then $|\mathbf{A}\mathbf{w}_1|^2 + |\mathbf{A}\mathbf{w}_2|^2$ is the sum of the squared projections of the rows onto W. We choose the orthonormal basis such that $\mathbf{w}_2$ is orthogonal to $\mathbf{v}_1$.
  Since $\mathbf{v}_1$ maximises $|\mathbf{A}\mathbf{v}_1|^2$, we have that $|\mathbf{A}\mathbf{w}_1|^2 \le |\mathbf{A}\mathbf{v}_1|^2$. For the optimal $\mathbf{v}_2$ we have $|\mathbf{A}\mathbf{w}_2|^2 \le |\mathbf{A}\mathbf{v}_2|^2$. Therefore

  $$|\mathbf{A}\mathbf{w}_1|^2 + |\mathbf{A}\mathbf{w}_2|^2 \le |\mathbf{A}\mathbf{v}_1|^2 + |\mathbf{A}\mathbf{v}_2|^2.$$

  and we see that V_2 is at least as good as W, and therefore a best-fit 2-dimensional subspace.
- For general n, the induction hypothesis ensures that V_{n-1} is indeed the best fit $n-1$ dimensional subspace. Let W be the best fit n dimensional subspace with a basis $\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_n$, with $\mathbf{w}_n$ perpendicular to $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_{n-1}$. Then

  $$|\mathbf{A}\mathbf{w}_1|^2 + |\mathbf{A}\mathbf{w}_2|^2 + \ldots + |\mathbf{A}\mathbf{w}_n|^2 \le |\mathbf{A}\mathbf{v}_1|^2 + |\mathbf{A}\mathbf{v}_2|^2 + \ldots + |\mathbf{A}\mathbf{v}_{n-1}|^2 + |\mathbf{A}\mathbf{w}_n|^2$$

  since V_{n-1} is an optimal $n-1$ dimensional subspace. Since $\mathbf{w}_n$ is orthogonal to $\mathbf{v}_i$, $i = 1, 2 \ldots n-1$, we have $|\mathbf{A}\mathbf{w}_n|^2 \le |\mathbf{A}\mathbf{v}_n|^2$. Therefore

  $$|\mathbf{A}\mathbf{w}_1|^2 + |\mathbf{A}\mathbf{w}_2|^2 + \ldots + |\mathbf{A}\mathbf{w}_n|^2 \le |\mathbf{A}\mathbf{v}_1|^2 + |\mathbf{A}\mathbf{v}_2|^2 + \ldots + |\mathbf{A}\mathbf{v}_{n-1}|^2 + |\mathbf{A}\mathbf{v}_n|^2$$

  and consequently V_{n-1} is at least as good as W, hence optimal.

□

The n-vector $\mathbf{A}\mathbf{v}_i$ can interpreted as a list of lengths (with signs) of the projections of $\mathbf{A}$ onto $\mathbf{v}_i$. Consider a row $\mathbf{a}_i$ of the matrix $\mathbf{A}$. Since $\mathbf{v}_i$, $i = 1, \dots r$ span the space of all rows of $\mathbf{A}$, we have $\mathbf{a}_i \cdot \mathbf{v} = 0$ for all $\mathbf{v} \perp \mathbf{v}_i$, $i = 1, \dots r$. Then $\sum_{i=1}^{N} \mathbf{a}_j \cdot \mathbf{v}_i = |\mathbf{a}_j|^2$ and summing over all rows yields

$$\sum_{j=1}^{N} |\mathbf{a}_j|^2 = \sum_{j=1}^{N} \sum_{i=1}^{r} (\mathbf{a}_j \cdot \mathbf{v}_i)^2 = \sum_{i=1}^{r} |\mathbf{A}\mathbf{v}_i|^2 = \sum_{i=1}^{r} \sigma_i^2(A).$$

Since $\sum_{j=1}^{N} |\mathbf{a}_j|^2 = \sum_{j=1}^{N} \sum_{k=1}^{d} a_{jk}^2$, that is the sum of all squares of the entries of $\mathbf{A}$, we see that this equals the sum of all squared singular values (hence carrying quite a bit of information about the matrix $\mathbf{A}$). There is a norm associated to this quantity - the Frobenius norm, defined as

$$\|\mathbf{A}\|_F = \sqrt{\sum_{j,k} a_{jk}^2}. \tag{22.3}$$

Next we define the set of left singular vectors of $\mathbf{A}$ by

$$\mathbf{u}_i = \frac{1}{\sigma_i(\mathbf{A})} \mathbf{A}\mathbf{v}_i.$$

The right singular vector $\mathbf{v}_i$, $i = 1, \dots r$ are orthogonal by definition. But so are the left-singular vectors $\mathbf{u}_i$.

Lemma 22.2. Let $\mathbf{A} \in \mathbb{R}^{N \times d}$ be a matrix with positive entries and rank r. Then its left singular vectors $\mathbf{u}_1, \dots \mathbf{u}_r$ are orthogonal.

We will leave this proof as an exercise. The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ define the columns of $\mathbf{U}$ and $\mathbf{V}$ in (22.1). The entries of the diagonal matrix $\mathbf{D}$ are the singular values σ^2. In particular we can rewrite (22.1) as

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

The best n-rank approximation of the matrix $\mathbf{A}$ aims to find the best n-rank matrix of $\mathbf{A}$ with respect to a certain norm. We already introduced the Frobenius norm (22.3), but we will also discuss the 2-norm (also known as the spectral norm)

$$\|\mathbf{A}\|_2 = \max_{|v| \leq 1} |\mathbf{A}\mathbf{v}|. \tag{22.4}$$

We also define the n approximation for $n \in \{1, \dots r\}$ as

$$\mathbf{A}_n = \sum_{j=1}^{n} \sigma_j \mathbf{u}_j \mathbf{v}_j^T. \tag{22.5}$$

It is clear that $\mathbf{A}_n$ has rank n. We will show that $\mathbf{A}_n$ is the best n-rank approximation to $\mathbf{A}$ wrt the Frobenius norm (22.3).

Theorem 22.3. Let $\mathbf{A} \in \mathbb{R}^{N \times d}$ with real entries. Then for every $\mathbf{B} \in \mathbb{R}^{N \times d}$ of rank at most n we have

$$\|\mathbf{A} - \mathbf{A}_n\|_F \leq \|\mathbf{A} - \mathbf{B}\|_F.$$

To show this statement we have to show the following lemma first

Lemma 22.4. Let $\mathbf{A} \in \mathbb{R}^{N\times k}$ and $\mathbf{A}_n$ defined by (22.5). Then the rows of $\mathbf{A}_n$ are the projections of $\mathbf{A}$ onto V_n, spanned by the first n singular vectors of $\mathbf{A}$.

Proof. Let $\mathbf{a}$ be an arbitrary row vector of $\mathbf{A}$. Then the projection of $\mathbf{a}$ onto V_n is given by $\sum_{k=1}^{n}(\mathbf{a}\cdot\mathbf{v}_k)\mathbf{v}_k^T$. Therefore the matrix whose rows are projections of rows of $\mathbf{A}$ onto V_n is given by $\sum_{k=1}^{n} A\mathbf{v}_k\mathbf{v}_k^T$. But this is actually $\mathbf{A}_n$ since

$$\sum_{k=1}^{n}\mathbf{A}\mathbf{v}_k\mathbf{v}_k^T = \sum_{k=1}^{n}\sigma_k\mathbf{u}_k\mathbf{v}_k = \mathbf{A}_n.$$

□

Now we are able to prove Theorem 22.3.

Proof of Theorem 22.3. Assume that $\mathbf{B}$ minimises $\|\mathbf{A}-\mathbf{A}_n\|_F$ among all rank n matrices. We denote the space spanned by the rows of $\mathbf{B}$ by V (note that the dimension of V is at most n). Since $\mathbf{B}$ is the best rank approximation is row of $\mathbf{B}$ is the projection of a row of $\mathbf{A}$ onto V. Therefore $\|\mathbf{A}-\mathbf{B}\|_F^2$ is the sum of the squared distances of rows of $\mathbf{A}$ on V. Since A_n minimises the sum of the squared distance of rows of $\mathbf{A}$ to any n-dimensional subspace, we have that $\|\mathbf{A}-\mathbf{A}_n\| \leq |\mathbf{A}-\mathbf{B}\|_F$. □

The matrix $\mathbf{A}_n$ is also the best rank n approximation in the 2-norm. Before we state the corresponding theorem we compute the square of the 2 norm of $\mathbf{A}-\mathbf{A}_n$.
From the definition of A_n (22.5) we obtain that $\mathbf{A}-\mathbf{A}_n = \sum_{k=n+1}^{r}\sigma_k\mathbf{u}_k\mathbf{v}_k^T$. Let $\mathbf{v}$ be the first singular vector of $\mathbf{A}-\mathbf{A}_n$, which we can write as $\mathbf{v} = \sum_{j=1}^{r} c_j\mathbf{v}_j$. We compute

$$\begin{aligned}|(\mathbf{A}-\mathbf{A}_n)\mathbf{v}| &= |\sum_{k=n+1}^{r}\sigma_k\mathbf{u}_k\mathbf{v}_k^T\sum_{j=1}^{r}c_j\mathbf{v}_j| = |\sum_{k=n+1}^{r}c_k\sigma_k\mathbf{u}_k\mathbf{v}_k^T\mathbf{v}_k \\ &= |\sum_{k=n+1}^{r}c_k\sigma_k\mathbf{u}_k| = \sqrt{\sum_{k=n+1}^{r}c_k^2\sigma_k^2}\end{aligned}$$

since $\mathbf{u}_i$ are orthogonal. To calculate the 2-norm of this expression we have to find a vector $\mathbf{v}$ with $\sum_{k=1}^{r}c_k^2 = 1$ (since it has to have norm one). Hence we choose $c_{n+1} = 1$ and $c_i = 0$ for $i \neq n+1$. Then $\|\mathbf{A}-\mathbf{A}_n\|_2^2 = \sigma_{n+1}^2$.

Theorem 22.5. Let $\mathbf{A} \in \mathbb{R}^{N\times d}$ with real entries, and $\mathbf{B} \in \mathbb{R}^{N\times d}$. Then

$$\|\mathbf{A}-\mathbf{A}_n\|_2 \leq \|\mathbf{A}-\mathbf{B}\|_2.$$

Proof. We now from the previous lemma that $\|\mathbf{A}-\mathbf{A}_n\|_2 = \sigma_{n+1}^2$. Assume that there is a matrix $\mathbf{B}$ that is a better rank n approximation, then $\|\mathbf{A}-\mathbf{B}\|_2^2 < \sigma_{n+1}^2$. The null space of $\mathbf{B}$ has (at least) dimension $d-n$. Let $\mathbf{v}_1,\dots\mathbf{v}_{n+1}$ be the first $n+1$ singular vectors of $\mathbf{A}$. Then there exists a $\mathbf{z} \neq 0 \in$ Null$(\mathbf{B}) \cap \{\mathbf{v}_1,\dots,\mathbf{v}_{n+1}\}$. W.l.o.g. we assume that $|\mathbf{z}| = 1$.
We will show that $(\mathbf{A}-\mathbf{B})\mathbf{z} \geq \sigma_{n+1}$. Because then the 2-norm of $\mathbf{A}-\mathbf{B}$ is at least σ_{n+1}, contradicting $\|\mathbf{A}-\mathbf{B}\|_2 < \sigma_{n+1}$.
From the definition of the 2 norm we have that

$$\|\mathbf{A}-\mathbf{B}\|_2^2 \leq |(\mathbf{A}-\mathbf{B})\mathbf{z}|^2$$

and since $\mathbf{Bz} = 0$ we have

$$|\mathbf{A} - \mathbf{B}\|_2^2 \geq |\mathbf{Az}|^2.$$

Since $\mathbf{z}$ is in span$\{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_{n+1}\}$ we have

$$|\mathbf{Az}| = \sum_{k=1}^{N} \sigma_k \mathbf{u}_k \mathbf{v}_k^T \mathbf{z}|^2 = \sum_{k=1}^{N} \sigma_k^2 (\mathbf{v}_k^T \mathbf{z})^2 = \sum_{k=1}^{n+1} \sigma_k^2 (\mathbf{v}_k^T \mathbf{z})^2 > \sigma_{n+1}^2 \sum_{k=1}^{n+1} (\mathbf{v}_k^T \mathbf{z})^2 = \sigma_{n+1}^2.$$

Therefore $\|\mathbf{A} - \mathbf{B}\|_2^2 \geq \sigma_{n+1}^2$, which is a contradiction. □