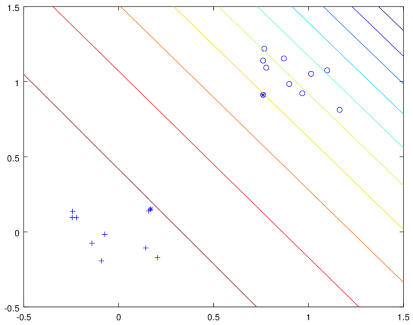
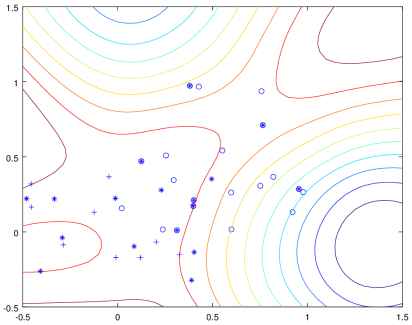
Learning ***Theory***

Lecture, first given in winter semester 2015/16

Tomas Sauer

Version 1.1
Last modified on: 21.10.2019





Statt einer Leerseite ... 0

Denn die Doktrin der geschlechtergerechten Sprache macht das Lesen solchermassen "'gerechter"' Texte nicht nur fast unerträglich. Sie basiert auch auf einem linguistischen Grundirrtum, weil es das biologische Geschlecht mit dem grammatischen Genus gleichsetzt.

C. Wirz, "'Neusprech für Fortgeschrittene"', *NZZ Online, 12.7.2013*

Die wahren Analphabeten sind schließlich diejenigen, die zwar lesen können, es aber nicht tun. Weil sie gerade fernsehen.

L. Volkert, *SZ–Online, 11.7.2009*

$$\frac{(a+b)!}{a!b!} \geq \sqrt{\frac{(a+b)^{a+b}}{a^a b^b}}.$$

And it didn't stop being magic just because you found out how it was done.

T. Pratchett, *Wee Free Men*

Oh mein Gott! Schon wieder ein englischsprachiges Skript. Aber im Zeitalter der Internationalisierung von Studiengängen, insbesondere des ITS–Schwerpunkts im Master "Informatik" bleibt einem nichts anderes übrig und letztendlich gilt halt doch: *The official language of science is broken English*. In diesem Sinne: Viel Spass damit.

Tomas Sauer
Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverabeitung
Universität Passau
Innstr. 43
94032 Passau

Contents | 0

The Vertrauensmann is man we are trusting. Not yesterday, maybe not tomorrow. But today we are trusting him for ever.

J. le Carré, *A Perfect Spy*

1 Introduction – What is Learning Theory

Machine Learning is definitely one of the most popular fields of research in Computer Science, with a lot of promises and announcements and even some real progress. Algorithms that gather and classify data and even make decisions on the basis of this data and its classification are widely available and often reported in the press. In this chapter we will try to learn about the basic mathematical formulation of the problem and some oversimplified ideas how to approach it, as a motivation for the more detailed mathematical facts we are deriving thereafter.

The simplest and most accepted way to distinguisch *Machine Learning* from other methods of articifical intelligence is to say that it is **data driven**. This means that the methods are not based on expert knowledge of the underlying data but try to *learn* the answer to the problem from data that has been gathered somehow and somewhere. Of course, *amount* and *quality* of that data play a fundamental role in what can really be learnt from them.

1.1 Supervised and unsupervised

Most common in machine learning are so–called **classification problems** which can be, in wide generality, be described as follows.

Definition 1.1 (Classification problem)**.** Given a set[1] $\mathscr{X}$ and a finite[2] subset $X \subset \mathscr{X}$ of **training data**, determine from X classes $X_1, \dots, X_n$ and a **discriminant function** $f : \mathscr{X} \to \{1, \dots, n\}$ with

$$f(X_j) = \{j\}, \qquad j = 1, \dots, n,$$

that determines to which class an arbitrary element of $\mathscr{X}$ belongs.

Learning problems can arise in two ways:

[1] "Set", in contrast to "space", is no accident here, in many problems the data to work on can be quite unstructured.

[2] Try to work with infinite sets in practice.

1. For each element $x \in X$ of the training data we know a **label** $y_x \in \{1, \dots, n\}$ that tells us which class the object belongs to. This is called **supervised learning**.

2. We have only the training data and have to detect the "hidden structure" and the classification automatically, maybe even the number n of classes. This is called **unsupervised learning**.

Any supervised learning problem can be easily turned into an unsupervised one by setting

$$X' = X \times y, \qquad y = \{y_x : x \in X\} \subset \{1, \dots, n\}^X,$$

with the hope that the unsupervised classification function $f : X' \to \{1, \dots, n\}$ satisfies

$$f(x, y_x) = y_x, \qquad x \in X.$$

1.2 An simple classification problem

In this section we will introduce some of the basic ideas of learning theory by merging the introductionary chapters of [51], [6] and [19]. The problem is a simple supervised one with two categories. The data is given as

$$X \subset \mathscr{X}, \qquad y_x \subset \{\pm 1\}, \quad x \in X.$$

For $\mathscr{X}$ we will consider three options:

1. unstructured set $\mathscr{X}$,

2. "vector space": $\mathscr{X} = \mathbb{R}^d$,

3. one parameter: $\mathscr{X} = \mathbb{R}$,

and we will learn a mechanism from [51] to extend the vector space ideas to unstructured sets by means of kernels. The case $\mathscr{X} = \mathbb{R}$ will help to test the approaches by means of a very simple example. What the chapter will also show is that there are various approaches that lead to the same thing in the end, but with quite different justifications *why* one does it in this particular way.

1.2.1 Discrimination by interpolation

But let us begin simple, with $X \subset \mathscr{X} = \mathbb{R}^d$ and $y \subset \{\pm 1\}^X$. We note that the problem is solved once we find the **discriminant function** $f : \mathscr{X} \to \{\pm 1\}$, then the classification is $f(x)$ for any $x \in \mathscr{X}$. The most naive way is to find a function that exactly "learns" the information provided by y, i.e., a function with the property

$$f(x) = y_x, \qquad x \in X. \tag{1.1}$$

Such functions are well known and studied for a long time[3].

[3] According to [5], the term "**interpolation**" has been introduced by Wallis in 1655. Hence, it is older than "derivative" or "matrix".

Definition 1.2 (Interpolation)**.** A function f is said to **interpolate** the data Y at X or is called an **interpolant** for Y at X if it satisfies (1.1) which we abbreviate as $f(X) = y$.

Of course, the **interpolation problem** (1.1) has infinitely many solutions, even among continuous or well–behaved functions of whichever sort. Normally it is turned into a linear problem by considering a **vector space** of functions spanned by a finite set Φ of functions $\mathscr{X} \to \mathbb{R}$. There is a point in considering **real valued functions**, we want a vector space and its structural[4] properties; for simplicity, we choose the most common underlying field, namely $\mathbb{R}$.

Exercise 1.1 Show that $\{f : [0,1] \to \{\pm 1\}\}$ is no vector space. Give at least three reasons. ◊

Any element f of the vector space $\mathscr{F}$ spanned by Φ can now be written as

$$f = \sum_{\varphi\in\Phi} a_\varphi\, \varphi, \qquad a_\varphi \in \mathbb{R},\ \varphi \in \Phi,$$

and our interpolation problem (1.1) can be rewritten as[5]

$$y_x = f(x) = \sum_{\varphi\in\Phi} a_\varphi\, \varphi(x) = \Phi(x)^T\, a, \qquad a := [a_\varphi : \varphi \in \Phi]\,,$$

or, in matrix form

$$y = \begin{bmatrix} \varphi(x) : & x \in X \\ & \varphi \in \Phi \end{bmatrix} [a_\varphi : \varphi \in \Phi] =: \Phi^T(X)\, a, \tag{1.2}$$

which is a **linear system** based on the **collocation matrix**

$$\Phi(X) := \begin{bmatrix} \varphi(x) : & \varphi \in \Phi \\ & x \in X \end{bmatrix} = \begin{bmatrix} \varphi_1(x_1) & \dots & \varphi_1(x_{\#X}) \\ \vdots & \ddots & \vdots \\ \varphi_{\#\Phi}(x_1) & \dots & \varphi_{\#\Phi}(x_{\#X}) \end{bmatrix}, \tag{1.3}$$

where in the right hand side expression we number the functions and points to indicate the structure in which the matrix is arranged. Now we can easily rely on linear algebra.

Proposition 1.3 (Solvability of (1.2))**.**

1. *The linear system*[6] *has a solution if the* ***rank*** *of the matrix* $\Phi(X)$ *is at least* $\#X$ *which implies that* $\#\Phi \geq \#X$*.*
2. *The linear system has a* unique *solution if the matrix is* ***invertible****, i.e.,* $\#\Phi = \#X$ *and* $\det \Phi(X) \neq 0$*.*

[4] *Numerical* computations in the real world cannot get enough structure if they are supposed to work well.

[5] We write vectors as column vectors and inner products in the form $a^T b$.

[6] And therefore the interpolation problem.

3. *With our notation, in the case of a unique solution the* ***interpolant*** *can be written as*

$$g(x) = \sum_{\varphi\in\Phi} \left(\Phi(X)^{-1}y\right)_\varphi \ \varphi(x) = \Phi(x)^T\,\Phi(X)^{-1}y. \tag{1.4}$$

The identity (1.4) is of purely theoretical use, it is not a numerically safe or recommendable way to compute the interpolant, not even in the simplest situations. Moreover, even if it perfectly captures the behavior on y, the function g is **not** a discriminant function since its values do not restrict to $\{\pm 1\}$; however, we can simply define the discriminant function as[7]

$$f(x) = \operatorname{sgn} g(x) = \begin{cases} 1, & g(x) \geq 0, \\ -1, & g(x) < 0. \end{cases} \tag{1.5}$$

Note that

1. f is discontinuous, and behaves erratical if $g(x) \sim 0$, i.e., if the interpolant is "undecided",
2. the handling of $g(x) = 0$ is arbitrary,

so that this discriminant function will have jumps as soon as the interpolant is at least continuous and not all training data belongs to one class.

Example 1.4 ("Learning" by interpolation)**.** The most common interpolation method for $d = 1$ is to use **polynomials** of degree $< \#X$, setting $\Phi = \{1, x, \ldots, x^m\}$, $m = \#X - 1$. It is well–known, cf. [10, 22, 46, 56], that there always exists a unique solution that can be computed in a mostly stable way. Let us look at different configurations:

1. Two training values, one from $(-\infty, 0)$ with label -1 and one from $(0, \infty)$ with label $+1$. As shown in Fig. 1.1, the discriminant function changes its sign in the middle.
2. The points $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$ and their negatives labeled according to sign. The interpolant still does the job quite well, but the interpolant grows dramatically outside the interval, see Fig. 1.2.
3. With sufficiently many randomly chosen points, we quite easily see in Fig 1.3 that the approach is of no use any more.

Exercise 1.2 Find out what happens in one of the good cases if one element of the training is mislabeled. ♢

What we learn from Example 1.4 is obvious: polynomial interpolation is **not** a good choice. The question, of course, is "why" and there are quite a few reasons:

1. polynomials of high degree oscillate very strongly and should be avoided in numerical applications in general. This carries over to the interpolant and leads to the behavior seen in Fig. 1.3.

[7] Be aware of the fact that the handling of zero by the **sign function** sgn can vary in literature and software libraries, the most common choices being $\operatorname{sgn} 0 = 0$ and $\operatorname{sgn} 0 = 1$.

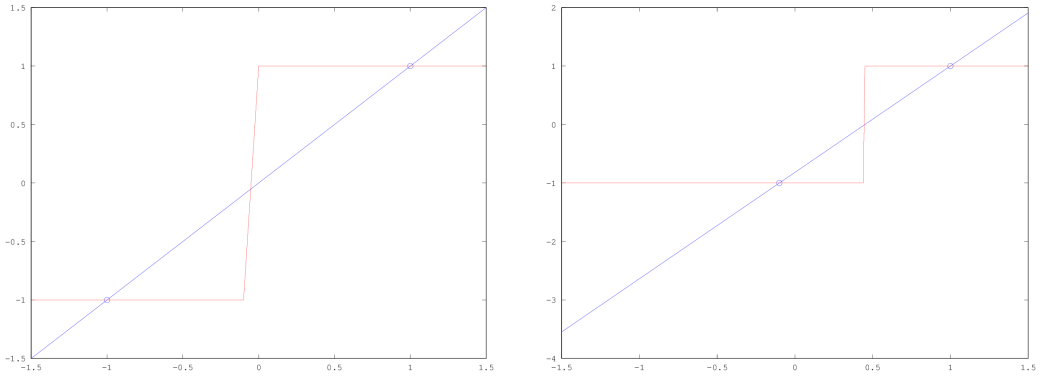


Figure 1.1: Learning one positive and one negative point. But the discriminant function changes with the choice of points.

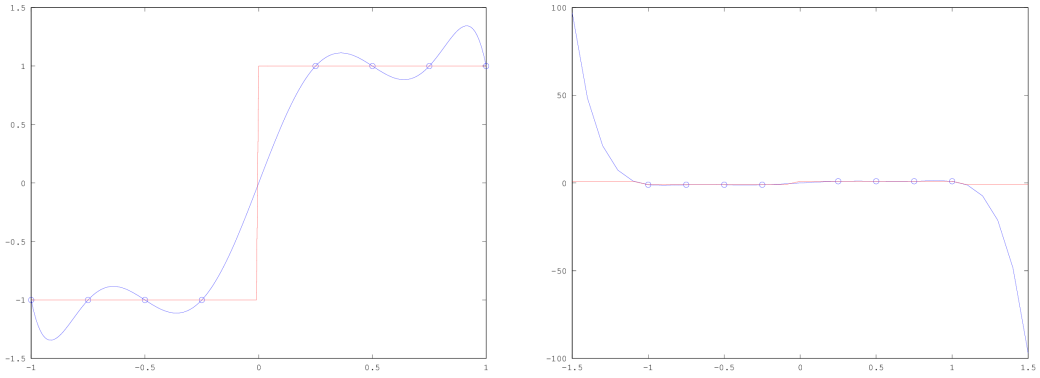


Figure 1.2: Several points on the left and on the right, looks quite OK on $[-1, 1]$ *(left)* but the interpolant becomes large outside *(right)*. Still, the resulting discriminant function is perfect.

2. Even if this can be overcome by choosing other types of functions for interpolation, for example splines, see [7, 50], there is a more fundamental problem with interpolation that we already mentioned in Proposition 1.3, 1): The number of functions, $\#\Phi$, has to be greater than or equal to the number of training points $\#X$. In other words, we need at least as much information[8] to explain the data as we need for the data itself, and this is the main problem.

Remark 1.5 (Occam's razor, overlearning)**. Occam's razor** or **lex parsimoniae** is a scientific principle formulated as *"Non sunt multiplicanda entia sine necessitate"*[9] by William of Ockham[10]. It can be rephrased as that among all

[8] Coefficients with respect to Φ.

[9] *"Entities must not be multiplied beyond necessity"*

[10] $\sim$ 1287–1347, English monk and philosopher.

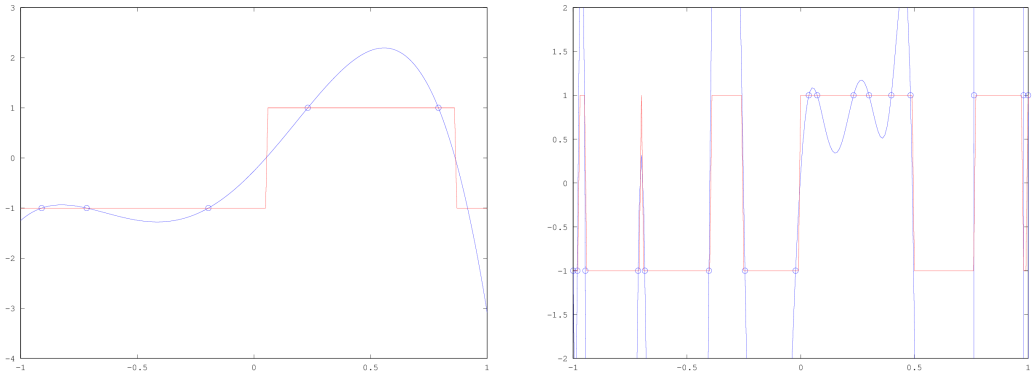


Figure 1.3: Random choices of the x values, labeled according to their sign. Three negative and two positive abscissas *(left)* yield a strange behavior, but typical for interpolation of an odd number of points, seven random points on the left and eight on the right together with ±1 yield a totally erratic behavior *(right)*.

explanations of a phenomenon the simplest, i.e., the one with the smallest number of parameters should be chosen. In view of that principle, interpolation is as complicated as the data itself and therefore a bad interpretation. In Learning Theory such a solution is called **overlearning**.

1.2.2 Simpler models

In **linear regression**, a common principle in statistics, the discriminant function is modeled as

$$f(x) = \phi(x, a) = a_0 + \sum_{\varphi \in \Phi} a_\varphi \, \varphi(x) \tag{1.6}$$

for a given set Φ of real valued[11] functions. The constant offset a_0 is called **bias** and can be dropped by assuming that $1 \in \Phi$. Hence, we consider functions of the form

$$\phi(x, a) = \sum_{\varphi \in \Phi} a_\varphi \, \varphi(x) = \Phi(x)^T \, a \tag{1.7}$$

where the coefficient vector a describes the discriminant function completely. Note, however, that the quality of the model clearly depends on the set Φ. Since we cannot guarantee any more that $\phi(x, a) = y_x$, $x \in X$, we can only try to fit the function as good as possible. To describe "good", we use a concept from information theory which slightly weakens the notion of a **norm** known from Analysis and Linear Algebra.

Definition 1.6 (Loss function)**.** A function $\Lambda : \mathbb{R} \to \mathbb{R}$ is called **loss function** if

1. $\Lambda(t) \geq 0$, $t \in \mathbb{R}$,

[11] For simplicity! This can be easily generalized.

2. $\Lambda(-t) = \Lambda(t)$, $t \in \mathbb{R}$,

3. $\Lambda(t) = 0$ iff[12] $t = 0$.

Standard examples for loss functions are $\Lambda(t) = t^2$ and $\Lambda(t) = |t|$.

Give a loss function Λ we can now define the **average loss**[13]

$$\Lambda(\Phi, a, X, y) = \Lambda_a(\Phi, a, X, y) := \frac{1}{\#X} \sum_{x \in X} \Lambda\left(\phi(x, a) - y_x\right), \tag{1.8}$$

and the **maximal loss**

$$\Lambda(\Phi, a, X, y) = \Lambda_\infty(\Phi, a, X, y) := \max_{x \in X} \Lambda\left(\phi(x, a) - y_x\right) \tag{1.9}$$

The best explanation is then the coefficient vector a that solves the minimization problem

$$\min_a \Lambda(\Phi, a, X, y). \tag{1.10}$$

We will have a closer look at optimization problems, which form a fundamental part of Learning Theory, later, for now we set $\Lambda(t) = t^2$ and obtain by substituting (1.7) into (1.8), that

$$\begin{aligned}
\Lambda(\Phi, a, X, y) &= \sum_{x \in X} \left(\phi(x, a) - y_x\right)^2 = \sum_{x \in X} \left(\Phi(x)^T a - y_x\right)^2 \\
&= \sum_{x \in X} \left(a^T \Phi(x) \Phi(x)^T a - 2 y_x \Phi(x)^T a + y_x^2\right) \\
&= a^T \left(\sum_{x \in X} \Phi(x) \Phi(x)^T\right) a - 2 \left(\sum_{x \in X} y_x \Phi(x)\right)^T a + \sum_{x \in X} y_x^2 \\
&=: a^T A a - 2 b^T a + c,
\end{aligned}$$

which is a **quadratic form** in a with a symmetric square matrix $A \in \mathbb{R}^{\#\Phi \times \#\Phi}$, the vector $b \in \mathbb{R}^{\#\Phi}$ and $c \in \mathbb{R}$.

Lemma 1.7. *A is a positive semidefinite matrix, i.e., $a^T A a \geq 0$, and the function*

$$G(a) := a^T A a - 2 b^T a + c$$

assumes its minimum at a if and only if

$$A a = b \qquad \Leftrightarrow \qquad \Phi(X)^T \Phi(X) a = \Phi(X)^T y \tag{1.11}$$

[12] A common abbreviation for "if and only if" invented by Paul Halmos, see [16].

[13] It becomes average by normalizing, but we could also consider it an ℓ_1–like loss function by dropping the $1/\#X$ factor. For the optimization that follows, the constant factor is irrelevant anyway.

Proof: Positive semidefiniteness of A follows directly by

$$a^T A a = \sum_{x \in X} a^T \Phi(x) \Phi(x)^T a = \sum_{x \in X} \left(\Phi(x)^T a \right)^2 \geq 0$$

with equality if and only $\Phi(x)^T a = 0$ for all $x \in X$ which is equivalent to[14]

$$0 = \left[\Phi(x)^T : x \in X \right] a = \Phi(X) a.$$

Since the first and second derivatives, that is, the **gradient** and the **Hessian**, of G, see [49], are of the form

$$\nabla G(a) = 2(Aa - b), \qquad \nabla^2 G(a) = A,$$

the function has a unique local, hence global minimum at any point a such that $Aa = b$. To derive the **normal form equations** on the right hand side of (1.11), we only have to note again that in "stacked" notation we have $\Phi(X) = [\Phi(x)^T : x \in X]$ and therefore[15]

$$\Phi(X)^T \Phi(x) = \sum_{x \in X} \Phi(x)\, \Phi(x)^T \qquad \text{and} \qquad \Phi(X)^T y = \sum_{x \in X} \Phi(x) y_x.$$

In particular, this implies that the **quadratic loss** takes the form

$$\Lambda(\Phi, a, X, y) = a^T \Phi(X)^T \Phi(X) a - 2\Phi(X)^T y + \|y\|_2^2, \tag{1.12}$$

which we record for later use. □

Therefore, given any basis Φ for a **model space** and training data (X, y), we can compute the best model $\Phi^T a$ by solving (1.11), i.e.,

$$a = \left(\Phi(X)^T \Phi(X) \right)^{-1} \Phi(X)^T y, \tag{1.13}$$

and the use the discriminant function

$$f(x) = \operatorname{sgn} \phi(x, a) = \operatorname{sgn} \left(\Phi(X)^T \Phi(X) \right)^{-1} \Phi(X)^T y.$$

Remark 1.8.

1. To use the explicit formula (1.13) is numerical suicide, even solving the system (1.11) directly is not the best idea. There are numerically more stable methods.

2. The quality of the method depends on the choice of Φ, on the number and type of functions. Here a priori knowledge on the problem to be solved can be helpful. We will consider some methods and ideas for choosing good bases.

[14] We simply stack all conditions into a vector and recognize the resulting matrix as the collocation matrix.

[15] This is purely formal and notational!

3. If $\#\Phi = \#X$ and $\Phi(X)$ is nonsingular, then we immediately get that

$$a = \Phi(X)^{-1}\Phi(X)^{-T}\Phi(X)^{T}y = \Phi(X)^{-1}y$$

and we are back to interpolation which therefore is a special case.

4. To avoid **overlearning**, it is recommendable to keep $\#\Phi$ small, but if it is too small the discrimination power of the model space will of course be limited.

Example 1.9. We apply the **least squares** approach to the data considered before and use as model space the polynomials of degree $\leq m$ spanned by $\Phi_m := \{1, x, \ldots, x^m\}$.

1. For the data $X = \{\pm 1\}$ we can interpolate whenever $m \geq 1$ and thus get the same result as in Fig 1.1. In the case of "overoverfitting", $m \geq 2$, the solution of (1.11) is not unique any more but `Matlab` automatically picks the simplest one which is the linear function, so that the implementation[16] behaves the same for all $m \geq 2$.

2. For the second example, $\pm\left\{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\right\}$, the results are shown in Fig. 1.4. For $m = 1, 5$ the discriminant function is what we expect, for $m = 3$ it is at least reasonable in the region covered by the training data.

3. The biggest difference occurs when we face many unstructured points as shown in Fig. 1.5 where we get no visible difference to the four points considered before.

The clear observation is: simple models help, at least for simple data. In all examples the linear discriminant seems to be the best choice.

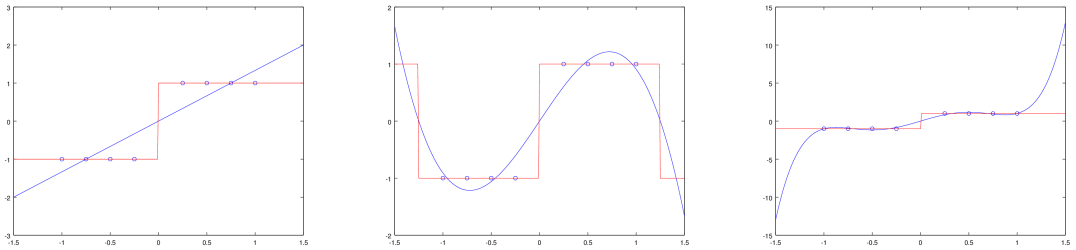


Figure 1.4: The four points of Fig. 1.2, this time with $m = 1$ *(left)*, $m = 3$ *(middle)* and $m = 5$ *(right)*.

[16] Mathematics says that there are many solution, an implementation always has to pick one. And it is not always easy to tune the implementation in such a way that this choosing process behaves reasonably.

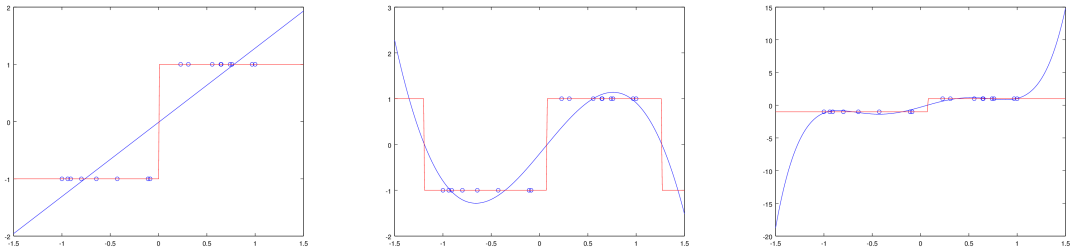


Figure 1.5: The same choices as in Fig 1.4, this time however with the random abscissas. The "randomness" and the larger number of these points visibly does not really play a role.

1.2.3 A "different" concept: likelihood

Next, we approach our problem from a more statistical perspective. To that end, we recall some very basic notions from probability.

Definition 1.10. A **probability density** function $p : \mathbb{R} \to \mathbb{R}$ is any measurable[17] function with

$$p(t) \geq 0, \quad t \in \mathbb{R}, \qquad \int_{\mathbb{R}} p(t)\, dt = 1. \tag{1.14}$$

The **Gaussian distribution** with **mean** μ and **variance** σ^2 is defined as

$$N(t|\mu, \sigma^2) := \left(2\pi\sigma^2\right)^{-1/2} e^{-\frac{1}{2\sigma^2}(t-\mu)^2}. \tag{1.15}$$

Exercise 1.3 Verify that the Gaussian distribution is a probability distribution. ◊

We return to our model $\phi(\cdot, a)$, but now we assume that training data stems from inexact measurements of that model and that the errors are distributed at $x \in X$ as $N(\cdot|\phi(x, a), \beta^{-1})$. This means that each single measurement or training datum has the largest probability at the "exact" $\phi(x, a)$ but varies around this value with nonzero probability. This deviation is, however, *independent* of the location x and describe by β^{-1}, $\beta > 0$, where the situation becomes more unpredictable if β becomes smaller[18].

We also assume[19] that the measurements are independent, so that the **likelihood** of the measurement y takes the form

$$p(y|X, a, \beta) = \prod_{x \in X} N\left(y_x|\phi(x, a), \beta^{-1}\right). \tag{1.16}$$

[17] All integrals over reasonable subsets must be well–defined. Since a careful definition needs some terminology, we remain deliberately informal here.

[18] Therefore, β measures how predictable the measurement is.

[19] Even if they are sometimes hard to justify in practice, these are standard assumptions in all these approaches.

To understand what this function means, let us recall that we know the measurements X and y, that β is a procedural parameter that we can choose freely and that we want to determine the model parameters α that best explain the measurements in the sense that they maximize the likelihood.

Since the location[20] of a maximum is not affected by monotonic functions, we can as well maximize the function $\log p(y|X, \alpha, \beta)$ or minimize the function

$$\begin{aligned} -\log p(y|X, \alpha, \beta) &= -\sum_{x \in X} \log N\left(y_x | \phi(x, \alpha), \beta^{-1}\right) \\ &= -\sum_{x \in X} \left(\frac{1}{2}\log \beta - \frac{1}{2}\log 2\pi - \frac{\beta}{2}\left(y_x - \phi(x, \alpha)\right)^2\right) \\ &= \frac{\beta}{2}\sum_{x \in X}\left(y_x - \phi(x, \alpha)\right)^2 - \frac{\#X}{2}\log \beta + \frac{\#X}{2}\log 2\pi \end{aligned} \tag{1.17}$$

with respect to α. The same computations as in the quadratic loss function then yield that

$$-\log p(y|X, \alpha, \beta) = \frac{2}{\beta}\Lambda(\Phi, \alpha, X, y) + \frac{\#X}{2}\log \frac{2\pi}{\beta} \tag{1.18}$$

and since constants do not change anything in optimization problems, the solution of our quadratic loss problem is the same as in the maximal likelihood one.

There is, however, a nice twist in this approach: we can play with the parameter β that regulates the width of the Gaussian distribution and we can choose β such that the explanation becomes optimal. To that end, we simply differentiate (1.17) with respect to β and get

$$\frac{d}{d\beta}\left(-\log p(y|X, \alpha, \beta)\right) = \frac{1}{2}\sum_{x \in X}\left(y_x - \phi(x, \alpha)\right)^2 - \frac{\#X}{2\beta},$$

hence the minimum is assumed for

$$\beta = \left(\frac{1}{\#X}\sum_{x \in X}\left(y_x - \phi(x, \alpha)\right)^2\right)^{-1}$$

which can be seen as a (reciprocal) variance of the model. In fact, β^{-1} was exactly chosen as the variance parameter in the model (1.16). The better the model fits, the wider the distribution can be chosen.

1.2.4 Hilbert spaces

We take yet another approach, this time from Functional Analysis or Approximation Theory which is based on norming vector spaces. To do this in a really nice and general way, we introduce some more terminology.

[20] But not the value!

Definition 1.11 (Inner product)**.** Let V a vector space[21] over $\mathbb{R}$. An **inner product** or **scalar product** $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ is a symmetric[22] and definite **bilinear form** which satisfies the following requirements:

1. **(Bilinearity)** For $\alpha, \beta \in \mathbb{R}$ and $v, v', w, w' \in V$,

$$\langle \alpha v + \beta v', w \rangle = \alpha \langle v, w \rangle + \beta \langle v', w \rangle \tag{1.19}$$

   and

$$\langle v, \alpha w + \beta w' \rangle = \alpha \langle v, w \rangle + \beta \langle v, w' \rangle . \tag{1.20}$$

2. **(Symmetry)** For $v, w \in V$,

$$\langle v, w \rangle = \langle w, v \rangle . \tag{1.21}$$

3. **(Definiteness)** For $v \in V \setminus \{0\}$,

$$\langle v, v \rangle > 0. \tag{1.22}$$

Example 1.12 (Inner products)**.** The classical inner products on $\mathbb{R}^d$ or on function spaces[23] are

$$\langle v, w \rangle = \sum_{j=1}^{d} v_j w_j \qquad \text{and} \qquad \langle f, g \rangle = \int f(x) g(x) \, dx,$$

respectively, but there are many more examples of useful inner products.

Exercise 1.4 Show that $\langle v, w \rangle := v^T A w$ is an inner product on $\mathbb{R}^d$ for any symmetric positive definite matrix $A \in \mathbb{R}^{d \times d}$. ◊

Inner products can be used to define geometric quantities like "length" of vector space elements and "angles" between elements. The mathematical concept of "length" is as follows.

Definition 1.13 (Norm)**.** A mapping $\| \cdot \| : V \to \mathbb{R}$ is called a **norm** on V if it satisfies the following conditions:

1. **(Nonnegativity)** $\|v\| \geq 0$ and $\|v\| = 0$ iff $v = 0$.

2. **(Positive homogeneity)** For $v \in V$ and $\alpha \in \mathbb{R}$,

$$\|\alpha v\| = |\alpha| \, \|v\|. \tag{1.23}$$

3. **(Triangle inequality)** For $v, v' \in V$,

$$\|v + v'\| \leq \|v\| + \|v'\|. \tag{1.24}$$

[21] Defined in Linear Algebra. Essentially it means that we can add elements and multiply them by reals and all that in a reasonable way.

[22] We restrict ourselves to $\mathbb{R}$! Over the complex numbers, the story is different.

[23] Vector spaces are **not** only collections of arrows or directed whatsoever-s.

Example 1.14 (Norms)**.** The "standard" norm on $\mathbb{R}^d$ is the **Euclidean norm**

$$\|v\|_2 := \sqrt{\sum_{j=1}^{d} v_j^2},$$

a member of the family of p**–norms**

$$\|v\|_p := \left(\sum_{j=1}^{d} |v_j|^p \right)^{1/p}, \quad 1 \le p < \infty, \qquad \|v\|_\infty := \max_{j=1,\dots,d} |v_j|.$$

Figure 1.6: Three grayscale images. How can we measure their similarity? For example by taking a norm of the difference.

Example 1.15 (PSNR)**.** In digital **image processing**, pictures are often compared, for example after performing operations like denoising. The standard approach is to interpret pictures x and y as pixel sequences, that is, as vectors in $\mathbb{R}^N$, $N =$ # pixels, and to consider the **distance**[24] $\|x-y\|_2$ or, more precisely, the reciprocal of it[25], resulting in the **PSNR** (Peak Signal to Noise Ratio) defined as

$$\mathrm{PSNR}(x,y) = 20 \log_{10} \frac{I_x \sqrt{N}}{\|x-y\|}, \qquad I_x := \max_{j=1,\dots,N} |x_j|,$$

where I_x denotes the **intensity** of x. If PSNR is really a good similarity measure is up to discussion, see Fig 1.7.

The next fundamental result, given without proof[26] here, connects inner products and norms.

Theorem 1.16. *If $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ is an inner product on V, then*

$$\|v\| = \sqrt{\langle v, v \rangle} \tag{1.25}$$

*is a norm, **induced** by the scalar product.*

[24] The mathematical concept for distances is the **metric** and the related concept of a metric space.

[25] Large PSNR means similarity, a PSNR of 0 expresses dissimilarity.

[26] It's not too hard and very elementary. The only point that needs a bit of work is the triangle inequality, see [48, Lemma 8.46]

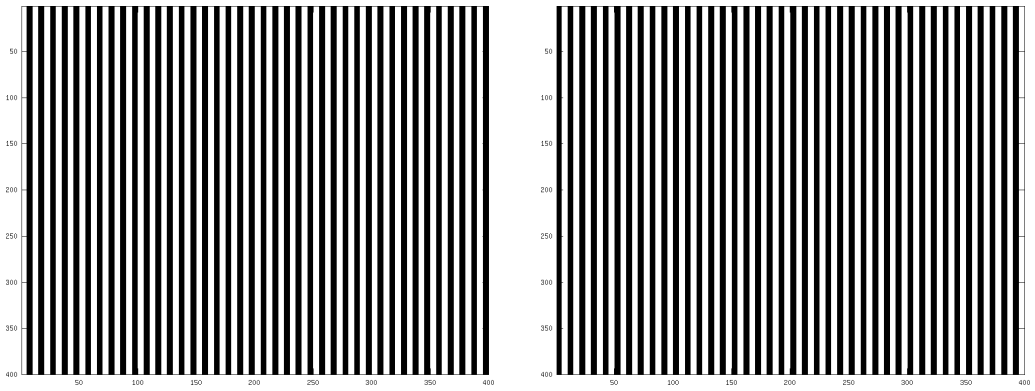


Figure 1.7: The strips in these two images are shifted in such a way that one is black where the other is white, hence the PSNR is 0. So much about measuring similarity

Definition 1.17 (Hilbert space)**.** A complete[27] normed vector space whose norm is induced by an inner product is called a **Hilbert space**.

Example 1.18 ($\mathbb{R}^d$)**.** The simplest example for a Hilbert space is $\mathbb{R}^d$ with the Euclidean norm which is induced by the standard inner product on $\mathbb{R}^d$.

Optimization or best approximation problems in Hilbert spaces are quite simple as the next result shows which we will fully prove as it introduces us to some concepts and ideas that we will need in the sequel anyway.

Proposition 1.19 (Subspace approximation in Hilbert spaces)**.** *Let $W \subseteq V$ be a subspace of the Hilbert space V.*

1. *For each $v \in V$ there exists a* unique *$w \in W$ such that $\|v - w\|$ is minimal, i.e.*

$$\|v - w\| = \min_{w' \in W} \|v - w'\|.$$

2. *w is the minimizing element if and only if*[28]

$$\{0\} = \langle W, v - w\rangle = \{\langle w', v - w\rangle : w' \in W\} \tag{1.26}$$

Proof: The proof consists of recording some elementary but important properties of Hilbert space norms emerging from the **Cauchy–Schwarz inequality**

$$|\langle v, w\rangle| \leq \|v\| \, \|w\|, \qquad v, w \in V, \tag{1.27}$$

with equality iff $v = w$, which holds for *any* norm induced by an inner product[29]

[27] Any **Cauchy sequence** has a limit; this is a *topological* property where the topology is induced by the norm.

[28] Geometrically this is the **orthogonal projection**.

[29] This is really a general and quite powerful approach even it is not too hard to prove.

1. *The norm* $\|\cdot\|$ *is* ***strictly convex****,* i.e., for any $v, v' \in V$ and $0 < \alpha < 1$, the inequality (1.27) yields

$$\begin{aligned}\left\|\alpha v + (1-\alpha)v'\right\|^2 &= \alpha^2\langle v, v\rangle + 2\alpha(1-\alpha)\langle v, v'\rangle + (1-\alpha)^2\langle v', v'\rangle \\ &\leq \alpha^2 \|v\|^2 + (1-\alpha)^2 \|v'\|^2 + 2\alpha(1-\alpha) \|v\| \|v'\| \\ &= (\alpha \|v\| + (1-\alpha) \|v'\|)^2,\end{aligned}$$

and since equality occurs in (1.27) only if $v = v'$, we can conclude that

$$\left\|\alpha v + (1-\alpha)v'\right\| < \alpha \|v\| + (1-\alpha) \|v'\|, \qquad v \neq \lambda v'. \tag{1.28}$$

This yields the *strict* convexity[30].

2. *There exists a* ***best approximation*** *from* W since the continuous function[31] $\|\cdot - v\|$ is bounded from below by zero, hence has an infimum and therefore a minimum since any subspace is closed.

3. *The* ***best approximation*** *is unique:* for any $v \in V$ there is exactly one $w^* \in W$ such that

$$\|v - w^*\| < \|v - w\|, \qquad w \in W \setminus \{w^*\},$$

which follows from strict convexity. Supposing that $w_1 \neq w_2 \in W$ were two best approximations, we set $w := \frac{1}{2}w_1 + \frac{1}{2}w_2 \in W$ and obtain

$$\begin{aligned}\|v - w\| &= \left\|v - \frac{1}{2}(w_1 + w_2)\right\| = \left\|\frac{1}{2}(v - w_1) + \frac{1}{2}(v - w_2)\right\| \\ &< \frac{1}{2}(\|v - w_1\|_2 + \|v - w_2\|) = \min_{w \in W} \|x - w\|,\end{aligned}$$

which is a contradiction because of "<" and proves statement 1).

4. The **best approximation** w has the property that for any $w' \in W$

$$\begin{aligned}0 &< \|v - w'\|^2 - \|v - w\|^2 \\ &= \|v\|^2 - 2\langle v, w'\rangle + \|w'\|^2 - \|v\|^2 + 2\langle v, w\rangle - \|w\|^2 \\ &= \langle w', w'\rangle - \langle w, w\rangle + 2\langle v, w - w'\rangle = 2\langle v, w - w'\rangle - \langle w + w', w - w'\rangle \\ &= \langle (v - w) + (v - w'), w - w'\rangle\end{aligned}$$

Since w' was arbitrary in this inequality, we can replace it[32] by the **convex combination** $\alpha w' + (1-\alpha)w$, $\alpha \in (0,1)$, and get

$$\begin{aligned}0 &< \langle (v - w) + (v - \alpha w' - (1-\alpha)w), w - \alpha w' - (1-\alpha)w\rangle \\ &= \langle 2v - \alpha w' - (2-\alpha)w, \alpha w - \alpha w'\rangle \\ &= \alpha\langle \alpha(v - w') + (2-\alpha)(v - w), w - w'\rangle.\end{aligned}$$

[30] Recall that a real valued function f is called **convex** if $f(\alpha x + (1-\alpha)x') \leq \alpha f(x) + (1-\alpha)f(x')$.
[31] See Exercise 1.5.
[32] Yes, that's a trick!

We divide this expression by 2α and let $\alpha \to 0$, which directly brings us to the **Kolmogoroff criterion**[33]

$$0 \le \langle v - w, w - w' \rangle. \tag{1.29}$$

Now we are practically done since we can replay w' in (1.29) by $2w - w'$ yielding

$$0 \le \langle v - w, w - (2w - w') \rangle = \langle v - w, w' - w \rangle = -\langle v - w, w - w' \rangle \le 0,$$

which implies that $\langle v - w, w' - w \rangle = 0$ for any $w' \in W$ from which (1.26) and therefore statement 2) follow. □

Exercise 1.5 Prove that the norm is a continuous function. Find the definition of a **topological vector space** and show that any subspace of a finite dimensional topological vector space is closed. ♢

Now suppose we have a *finite dimensional* subspace W of V, again generated by $\Phi \subset V$, so that any element of W can now be written as

$$w = \phi(a) = \sum_{\varphi \in \Phi} a_\varphi \, \varphi.$$

The Hilbert space elements *can* be functions $\mathscr{X} \to \mathbb{R}$, but they do not have to, which allows for different structural approaches.

By Proposition 1.19, the best approximation w to a given v is characterized by

$$\begin{aligned} 0 &= \langle v - w, w' \rangle = \langle v - \phi(a), \phi(a') \rangle \\ &= \left\langle v - \sum_{\varphi \in \Phi} a_\varphi \, \varphi, \sum_{\varphi' \in \Phi} a'_{\varphi'} \, \varphi' \right\rangle = \sum_{\varphi' \in \Phi} a'_{\varphi'} \langle v, \varphi' \rangle - \sum_{\varphi, \varphi' \in \Phi} a_\varphi \, a'_{\varphi'} \, \langle \varphi, \varphi' \rangle \\ &=: g_{\Phi,v}^T a' - a^T G_\Phi a' = (g_{\Phi,v} - G_\Phi a)^T a', \end{aligned}$$

holding for any **coefficient vector** a', which is in turn equivalent to

$$[\langle v, \varphi \rangle : \varphi \in \Phi] = [\langle \varphi, \varphi' \rangle : \varphi, \varphi' \in \Phi] \; a. \tag{1.30}$$

This is again a rule that allows us to compute the best approximation or the minimizer of $\|v - \phi(a)\|$ by simply[34] solving a linear system.

Definition 1.20 (Gram matrix)**.** The square matrix

$$G_\Phi := [\langle \varphi, \varphi' \rangle : \varphi, \varphi' \in \Phi] \in \mathbb{R}^{\#\Phi \times \#\Phi} \tag{1.31}$$

is called the **Gram matrix** or **Gramian** of the **generating system** Φ of W.

Remark 1.21 (Gram matrix)**.** The Gramian has some interesting properties:

[33] Attention: strict inequalities loose their strictness in the limit.

[34] Of course, this is **not** simple, otherwise an area like Numerical Linear Algebra would not exist.

1. it is a symmetric and positive semidefinite matrix.
2. the rank of the Gramian is the dimension of span $\Phi = W$.
3. it takes the role of the matrix $\Phi(X)^T\Phi(X)$ known from the **normal form equations** (1.11).
4. if Φ is an **orthonormal basis**, that is

$$\langle \varphi, \varphi' \rangle = \delta_{\varphi,\varphi'}, \qquad \varphi, \varphi' \in \Phi, \tag{1.32}$$

   then $G_\Phi = I$ and vice versa.

We can recover our quadratic loss function also in terms of Hilbert spaces if we make the following simple setup:

Space: $V = \{f : X \to \mathbb{R}\}$.

Inner product:

$$\langle f, g \rangle = \sum_{x \in X} f(x)\, g(x).$$

Model space: Φ.

Approximation target: $v \in V$ defined by $v(x) := y_x$, $x \in X$.

Then we indeed have

$$[\langle v, \varphi \rangle : \varphi \in \Phi] = \left[\sum_{x \in X} y_x \varphi : \varphi \in \Phi \right] = \Phi(X)^T y$$

and

$$[\langle \varphi, \varphi' \rangle : \varphi, \varphi' \in \Phi] = \left[\sum_{x \in X} \varphi(x)\varphi'(x) : \varphi, \varphi' \in \Phi \right] = \Phi(X)^T\Phi(X).$$

and the best approximation in the Hilbert space is *exactly* our quadratic loss minimizer.

1.2.5 Summary

In this section we encountered three[35] seemingly different approaches to the problem of determining a discriminant function and always ended up with the same thing: Solve a linear system[36] to obtain the solution of a least squares problem. So it appears that the whole effort is pointless and that there is only one thing: Simple least squares. This is, of course not the case and even if the three approaches indeed coincide for a certain configuration of parameters, the allow for generalizations in different directions which then lead to significantly different methods:

[35] Even four if we count interpolation, but it is included in the least squares approach.

[36] How this should be done reasonably is well–known in Numerical Linear Algebra, see [15, 46].

Model Space: use a different **loss function** to obtain a different and for example more robust behavior.

Maximal Likelihood: use a different **probability density** to reflect different perturbation models.

Hilbert Space: use different spaces and different inner products.

All this we will consider in greater detail later.

1.3 Separating hyperplanes and support vector machines

Now we get to a separation method for clusters based on hyperplanes that in fact works in any Hilbert space.

Definition 1.22 (Hyperplane)**.** A **hyperplane** with **normal** $n \in \mathscr{H}$, $\|n\| = 1$, and **offset** $c \in \mathbb{R}$ in a Hilbert space[37] $\mathscr{H}$ is the set of all elements given as[38]

$$H(n, c) := \{x \in \mathscr{H} : \langle n, x\rangle = c\}. \tag{1.33}$$

It is the intersection and separation of the two **half spaces** $H_\pm(n, c)$ defined as

$$H_+(n, c) = \{\langle n, x\rangle - c > 0\} \qquad \text{and} \qquad H_-(n, c) = \{\langle n, x\rangle - c < 0\}.$$

Returning to the classification problem (X, y) from the preceding chapter, $y \in \{\pm 1\}^X$, where now $X \subset \mathscr{H}$ is a finite subset of the Hilbert space $\mathscr{H}$, we can form the **barycenter** of the sets $X_\pm := \{x \in X : y_x = \pm 1\}$ as

$$x_\pm := \frac{1}{\#X_\pm} \sum_{x \in X_\pm} x. \tag{1.34}$$

Again this is the solution of a least squares problem.

Lemma 1.23. *Given a finite set $V \subset \mathscr{H}$, we have that*

$$w = \operatorname{argmin} \sum_{v \in V} \|v - w\|^2 \qquad \Leftrightarrow \qquad w = \frac{1}{\#V} \sum_{v \in V} v. \tag{1.35}$$

Proof: We consider the derivative of

$$\sum_{v \in V} \|v - w\|^2 = \sum_{v \in V} \langle v - w, v - w\rangle = \sum_{v \in V} \langle v, v\rangle - 2\langle v, w\rangle + \langle w, w\rangle$$

with respect to w in terms of directional[39] or **Fréchet derivative**

$$D_u f = \lim_{h \to 0} \frac{f(\cdot + hu) - f(\cdot)}{h}.$$

[37] From now on, we use the letter $\mathscr{H}$ for Hilbert spaces, moving away from the general vector space V.

[38] For those who are interested in it: it is an **affine subspace**, for $c = 0$ even a **subspace**.

[39] In a generic Hilbert space we do not have unit vectors like in $\mathbb{R}^d$ which may define partial derivatives.

For our functions under consideration we find that[40]

$$D_u\langle v,\cdot\rangle(w) = \lim_{h\to 0}\frac{1}{h}\left(\langle v, w+hu\rangle - \langle v,w\rangle\right) = \langle v,u\rangle$$

and

$$\begin{aligned} D_u\langle\cdot,\cdot\rangle(w) &= \lim_{h\to 0}\frac{1}{h}\left(\langle w+hu, w+hu\rangle - \langle w,w\rangle\right) \\ &= \lim_{h\to 0}\frac{1}{h}\Big(\langle w,w\rangle + 2h\langle w,u\rangle + h^2\langle u,u\rangle - \langle w,w\rangle\Big) \\ &= \lim_{h\to 0}\left(2\langle w,u\rangle + h\langle u,u\rangle\right) = 2\langle w,u\rangle. \end{aligned}$$

Hence, the unique minimum is characterized by

$$0 = D_u\sum_{v\in V}\|v-\cdot\|^2(w) = -2\sum_{v\in v}\langle v,u\rangle + 2\#V\langle w,u\rangle = -2\left\langle\sum_{v\in v} v - \#V\, w, u\right\rangle$$

holding for all $u\in\mathscr{H}$ which is equivalent to the right hand side of (1.35). □

Remark 1.24. The equivalence (1.35) can also be used to define an average on *nonlinear* structures as long as these are normed or equipped with a reasonable metric, like the surface of a sphere. The **average** is simply the minimizer of a certain sum of distances.

The simplest classifier we can build on the basis of $x_\pm$ is based on taking the midpoint $x_0 = \frac{1}{2}(x_+ + x_-)$ and checking the in which of the half planes defined by the hyperplane with normal $x_+ - x_-$ passing through x_0 the point lies. Let us compute its offset c first. We have that $n = \frac{x_+-x_-}{\|x_+-x_-\|}$ and obtain c by requiring that

$$\begin{aligned} c &= \left\langle\frac{x_+-x_-}{\|x_+-x_-\|}, x_0\right\rangle = \frac{1}{\|x_+-x_-\|}\left\langle x_+-x_-, \frac{1}{2}(x_++x_-)\right\rangle \\ &= \frac{1}{2\|x_+-x_-\|}\langle x_+-x_-, x_++x_-\rangle = \frac{1}{2\|x_+-x_-\|}\left(\langle x_+,x_+\rangle - \langle x_-,x_-\rangle\right) \\ &= \frac{\|x_+\|^2-\|x_-\|^2}{2\|x_+-x_-\|} \end{aligned}$$

The resulting discriminant function is then

$$\begin{aligned} f(x) &= \operatorname{sgn}\left(\langle n,x\rangle - c\right) = \operatorname{sgn}\left(\left\langle\frac{x_+-x_-}{\|x_+-x_-\|}, x\right\rangle\frac{\|x_+\|^2-\|x_-\|^2}{2\|x_+-x_-\|}\right) \\ &= \operatorname{sgn}\left(\langle x_+-x_-, x\rangle - \frac{\|x_+\|^2-\|x_-\|^2}{2}\right). \end{aligned} \tag{1.36}$$

A very simple insight from this formula is that we do not have to normalize the vector n since we are only interested in signs at the end and can capture this in the offset or **threshold** c.

[40]This means that in *any* Hilbert space, as abstract as it may be, we can define linear and quadratic functions as differentiable functions and compute their derivatives.

If we drive the "separating hyperplane" idea a little bit further, we already get an idea of some of the classics of supervised learning, namely support vector machines. Here, we try to find a hyperplane $\langle n, \cdot \rangle + c$ that separates the two sets X_+ and X_- determined by the labeling. The distance of a point $x \in X$ to the hyperplane is[41]

$$\min\{\|x - v\| : \langle n, v \rangle = c\}.$$

To solve this constraint optimization we deal with **Lagrange multipliers** by formally forming [42] the **Lagrangian**[43]

$$g(v) := \|x - v\|^2 + 2\alpha\left(\langle n, v \rangle - c\right) = \langle x, x \rangle - 2\langle x, v \rangle + \langle v, v \rangle + 2\alpha\left(\langle n, v \rangle - c\right)$$

and taking its formal derivative with respect to v,

$$D_u g(v) = -2\langle x, u \rangle + 2\langle v, u \rangle + 2\alpha\langle n, u \rangle = 2\langle (v - x) + \alpha n, u \rangle$$

which is zero for all u iff $v = x - \alpha n$ and we can determine α from

$$c = \langle n, v \rangle = \langle n, x - \alpha n \rangle = \langle n, x \rangle - \alpha\|n\|^2 \qquad \Rightarrow \qquad \alpha = \frac{\langle n, x \rangle - c}{\|n\|^2},$$

which yields

$$v = x - \frac{\langle n, x \rangle - c}{\|n\|^2}\, n \qquad \Rightarrow \qquad \|x - v\| = \left\| \frac{\langle n, x \rangle - c}{\|n\|^2}\, n \right\| = \frac{|\langle n, x \rangle - c|}{\|n\|} \tag{1.37}$$

and holds even if the normal vector n for the hyperplane is not **normalized**, i.e., if we only require[44] that $\|n\| \neq 0$.

We now want to choose the hyperplane in such a way that it offers **optimal separation** by maximizing the distance to the closest point, i.e., by solving

$$\max_{n,c} \min_{x \in X} \min_{\langle n, v \rangle = c} \|x - v\| = \max_{n,c} \min\{\|x - v\| : \langle n, v \rangle = c, x \in X\}$$

In addition, however, we have to maintain the side condition that

$$y_x = f(x) = \operatorname{sgn}\left(\langle n, x \rangle - c\right), \qquad x \in X,$$

for the linear discriminant function defined by the hyperplane. As in [51], the optimization problem can be written as

$$\min_{n \in \mathscr{H}, c \in \mathbb{R}} \frac{1}{2}\|n\|^2 = \frac{\langle n, n \rangle}{2} \qquad \text{subject to } y_x\left(\langle n, x \rangle - c\right) \geq 1, \quad x \in X. \tag{1.38}$$

The side condition ensures that f behaves as wanted and the constant 1 on the right hand side of the inequality is arbitrary: any other positive constant would

[41] Another optimization problem. This is not with the intention to bore the reader by repetition, but to slowly make the reader acquainted with the main concepts and the need for optimization.

[42] The norm square is no accident, it makes things computationally easier by avoiding the square root as well as the factor 2 in front of the α

[43] We will get to this in the next chapter.

[44] $\|n\| = 0$ or $n = 0$, which is the same, cannot define a valid hyperplane.

do the job as well, we would only have to multiply n and c accordingly and make the almost trivial observation that $H(n, c)$ and $H(\alpha n, \alpha c)$, $\alpha \in \mathbb{R} \setminus \{0\}$, are the same.

To derive the form (1.38) we first write the side condition as

$$y_x \left(\langle n, x \rangle - c \right) = 0, \qquad x \in X, \tag{1.39}$$

and set

$$\varepsilon := \min_{x \in X} y_x \left(\langle n, x \rangle - c \right),$$

which is well-defined as long as $\#X < \infty$. Thus, (1.39) takes the form

$$y_x \left(\langle n, x \rangle - c \right) \geq \varepsilon, \quad x \in X \qquad \Leftrightarrow \qquad y_x \left(\left\langle \frac{n}{\varepsilon}, x \right\rangle - \frac{c}{\varepsilon} \right) \geq 1, \quad x \in X. \tag{1.40}$$

The **margin** ε becomes maximal if $\frac{n}{\varepsilon}$ becomes minimal among all non-normalized normals that satisfy the right hand side of (1.40); replacing $\frac{n}{\varepsilon}$ by n and $\frac{c}{\varepsilon}$ by c, this leads precisely to the optimization problem (1.38).

The Lagrangian[45] of (1.38) is

$$g(n, c) = \frac{\langle n, n \rangle}{2} - \sum_{x \in X} \alpha_x \left(y_x \left(\langle n, x \rangle - c \right) - 1 \right)$$

which has to be differentiated with respect to n and c. The first one is

$$D_n g(\cdot, c)(u) = \langle n, u \rangle - \sum_{x \in X} \alpha_x y_x \langle x, u \rangle = \left\langle n - \sum_{x \in X} \alpha_x y_x x, u \right\rangle \tag{1.41}$$

and

$$\frac{\partial}{\partial c} g(n, c) = \sum_{x \in X} \alpha_x y_x. \tag{1.42}$$

Setting them equal to zero leads to

$$n = \sum_{x \in X} \alpha_x y_x x \qquad \text{and} \qquad \sum_{x \in X} \alpha_x y_x = 0. \tag{1.43}$$

In addition, the α_x have to satisfy the **KKT conditions**[46]

$$\alpha_x \left(y_x \left(\langle n, x \rangle - c \right) - 1 \right) = 0, \qquad x \in X, \tag{1.44}$$

which say that either $\alpha_x = 0$ or the respective side condition is satisfied with equality. Hence, the points with $\alpha_x \neq 0$ are the ones where x is closest to the separating hyperplane and the ones that decide about the hyperplane which is the reason why they are named **support vector**.

[45] We still do not know precisely what we are doing here, but we will find out quite soon.

[46] To be introduced soon.

To determine the parameters n and c, we substitute the expression for n from (1.43) into g and get

$$\begin{aligned} g(n,c) &= \frac{1}{2}\left\langle \sum_{x'\in X} \alpha_{x'} y_{x'} x', \sum_{x\in X} \alpha_x y_x x \right\rangle - \sum_{x\in X} \alpha_x \left(y_x \left(\sum_{x'\in X} \langle \alpha_{x'} y_{x'} x', x\rangle - c \right) - 1 \right) \\ &= \frac{1}{2} \sum_{x,x'\in X} \alpha_x y_x \alpha_{x'} y_{x'} \langle x, x'\rangle - \sum_{x,x'\in X} \alpha_x y_x \alpha_{x'} y_{x'} \langle x, x'\rangle - c \underbrace{\sum_{x\in X} \alpha_x y_x}_{=0} + \sum_{x\in X} \alpha_x \\ &= \sum_{x\in X} \alpha_x - \frac{1}{2} \sum_{x,x'\in X} \alpha_x \alpha_{x'} \langle y_x\, x, y_{x'}\, x'\rangle =: 1^T\alpha - \frac{1}{2}\alpha^T G_{y\cdot X}\alpha. \end{aligned}$$

This **dual form** of the problem (1.38) has to be maximized subject to $\alpha \geq 0$,

$$\max_{\alpha} 1^T\alpha - \frac{1}{2}\alpha^T G_{y\cdot X}\alpha \qquad \text{subject to } \alpha \geq 0,\ y^T\alpha = 0, \tag{1.45}$$

and the two solutions are equivalent. The resulting system is

$$\begin{array}{rcl} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} & = & G_{y\cdot X}\alpha - \lambda - \mu y, \\ 0 & = & \lambda^T\alpha, \\ 0 & = & y^T\alpha, \end{array} \qquad 0 \leq \alpha, \lambda \in \mathbb{R}^{\#X}, \quad \mu \in \mathbb{R}.$$

This is a nonlinear problem and we will have to learn some methods and principles from optimization to solve it, see, for example [40]. After having determined α and therefore n, we obtain c by means of (1.44) as $c = y_x - \langle n, x\rangle$ for some x with $\alpha_x \neq 0$.

The advantage of Hilbert space methods is their application in **kernel learning**: if $\mathscr{X}$ is an unstructured space, for example images, we can pick a suitable[47] Hilbert space $\mathscr{H}$ and a function $\psi : \mathscr{X} \to \mathscr{H}$ and replace the Gramian-s above by

$$[\langle \psi(x), \psi(x')\rangle : x, x' \in X]$$

leading to the discriminant function

$$f = \operatorname{sgn}\left(\sum_{x\in X} y_x \alpha_x \langle \psi(\cdot), \psi(x)\rangle - c \right).$$

The **kernel** $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is then the function

$$k(x,x') = \langle \psi(x), \psi(x')\rangle, \qquad x, x' \in \mathscr{X}, \tag{1.46}$$

that will play a fundamental role in support vector machines.

[47] This has to be exploited in more detail.

I have the simplest tastes. I am always satisfied with the best.

O. Wilde

2 Introduction to Optimization

We have seen that optimization is obviously an important issue in learning as in most cases we had to determine parameters or configurations in an optimal way, minimizing loss functions, for example. Since optimization is a lecture of its own, we can only give some of the basic ideas and some elementary proofs.

In optimization, one considers functions $f : D \to \mathbb{R}$ and is interested in finding a location x such that

$$f(x) = \min_{x' \in D} f(x'), \qquad x := \operatorname*{argmin}_{x' \in D} f(x'). \tag{2.1}$$

The following distinctions are common:

Smooth vs. nonsmooth: Once we can take derivatives, high school math tells us that finding an extremum implies finding a zero of the derivative.

Constrained vs. non-constrained: Is the **domain** D a region without boundaries or do we have to take care of side conditions? In the latter case, what are the points that satisfy these conditions and do they exist at all?

Convex vs. nonconvex: Convex functions must have a unique local minimum[48] while nonconvex ones can behave "as they want".

We will explore some of these facts in the sequel and see how to deal with them.

2.1 Basic observations

Remark 2.1. Looking for minima as in (2.1) is sufficient since any minimum of f is a maximum of $-f$ and vice versa.

Definition 2.2. $x \in D$ is called a **local minimum** of $f : D \to \mathbb{R}$ if there exists a neighborhood[49] U of x such that

$$f(x) \leq f(x'), \qquad x' \in U, \tag{2.2}$$

and it is called a **global minimum** if we can choose $U = D$. A minimum is called **strict** if the strict inequality holds in (2.2) for all $x' \neq x$.

[48] And concave functions a local maximum. This is taking into account the second derivative in our high school math.

[49] A set that contains an open set containing x. This requires a **topology** on D, but how else could one talk about "local" effects.

Next we introduce differentiability in a way that works for function on arbitrary vector spaces[50], which we already needed in our Hilbert space context.

Definition 2.3 (Derivative)**.** Let V be a **vector space** and $f : V \to \mathbb{R}$.

1. A function f is called **directionally differentiable** in $v \in V$ if for all $u \in V$ the limit
$$D_u f(v) := \lim_{h \to 0^+} \frac{f(v + hu) - f(v)}{h} \tag{2.3}$$
exists.

2. The function is called **differentiable** in v if the **Gateaux variation** of f at v,
$$G(f, v) : V \to \mathbb{R}, \qquad u \mapsto G(f, v)(u) := D_u f(v) \tag{2.4}$$
is **linear**.

3. If V is a Hilbert space, and f is differentiable at v, then the **derivative** $Df(v) \in V$ is the unique vector[51] such that
$$D_u f(v) = \langle Df(v), u \rangle, \qquad u \in V. \tag{2.5}$$

Exercise 2.1 Show that any directionally differentiable function is continuous. ◊

Remark 2.4. Keep in mind that the directional derivative is the classical one sided univariate derivative of the function
$$g(t) = f(v + tu), \qquad t \in [0, \varepsilon)$$
at the position $t = 0$. Hence, we can use the standard theory to compute, for example,
$$D_u \|\cdot\|(v) = \frac{d}{dt} \sqrt{\langle v + tu, v + tu \rangle}\Big|_{t=0} = \frac{1}{2\sqrt{\langle v, v \rangle}} D_u \langle \cdot, \cdot \rangle(v) = \frac{\langle v, u \rangle}{\|v\|}. \tag{2.6}$$

Remark 2.5 (Differentiabilities in $\mathbb{R}^d$)**.** In the "standard case" $V = \mathbb{R}^d$, the Gateaux variation of a differentiable function is the $u^T \nabla f(v)$ with the **gradient** ∇f which is formed by vectorizing the partial derivatives. Moreover, the gradient is the unique vector that defines the linear function by means of an inner product.

Note that directional differentiability does not imply differentiability as one should learn in basic calculus, see Fig 2.1.

Exercise 2.2 Show that on $\mathbb{R}^d$, $d > 1$, the function $f(x) = \|x\|_\infty$ is directionally differentiable at $x = 0$, but the Gateaux variation is not linear. ◊

[50] Where we do not necessarily have a basis of unit vectors like in $\mathbb{R}^d$.

[51] This uses the fact that any Hilbert space is **reflexive**, which means that any vector also takes the role of a functional and vice versa and that all bounded functionals can be expressed as inner products, see [27, 61].

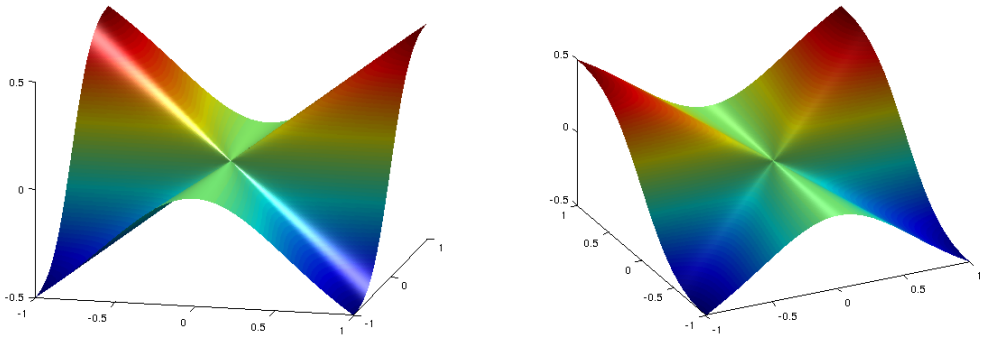


Figure 2.1: The function

$$f(x,y) = \begin{cases} \dfrac{xy^2}{x^2+y^2}, & (x,y) \neq (0,0), \\ 0, & (x,y) = (0,0), \end{cases} \tag{2.7}$$

shown here from two perspectives, is directionally differentiable, but not differentiable at the origin, which can be seen quite nicely from the plots.

Proposition 2.6. *If* $f : V \to \mathbb{R}$ *is directionally differentiable and has local minimum at* $v \in V$*, then*

$$D_u f(v) \geq 0, \qquad u \in V. \tag{2.8}$$

If f *is differentiable, then* $D_u f(v) = 0$, $u \in V$.

Proof: If v is local minimum, then $f(v) \leq f(v + hu)$ for all $u \in V$ and all sufficiently small values of h. This yields

$$0 \leq \frac{f(v+hu) - f(v)}{h} \quad \to \quad D_u f(v).$$

If f is differentiable, then $D_u f(v)$ is linear and we have for any $u \in V$ that

$$0 \leq D_{-u} f(v) = -D_u f(v) \leq 0,$$

hence $D_u f(v) = 0$. □

2.2 Convexity and its consequences

Convex functions play a fundamental role in optimization and it is no surprise that Optimization and Convex Analysis have a lot in common, cf. [43, 57].

Definition 2.7 (Convexity)**.** A function $f : V \to \mathbb{R}$ is called **convex** if

$$f\left((1-\alpha)x + \alpha x'\right) \leq (1-\alpha)\, f(x) + \alpha f(x'), \qquad \alpha \in [0,1], \tag{2.9}$$

holds for any $x, x' \in V$. A subset X of V is called **convex**[52] if

$$x, x' \in X \qquad \Rightarrow \qquad (1-\alpha)x + \alpha x' \in X, \quad \alpha \in [0,1]. \tag{2.10}$$

The **boundary** ∂X of a convex set X consists of all points $y \in X$ that cannot be written as a convex combination $y = (1-\alpha)x + \alpha x'$, $\alpha \in [0,1]$, $x, x' \in X \setminus \{y\}$.

Exercise 2.3 Show that $f : \mathscr{H} \to \mathbb{R}$ is convex if and only if for any *finite* subset $X \subset \mathscr{H}$ we have

$$f\left(\sum_{x\in X} \alpha_x x\right) \leq \sum_{x\in X} \alpha_x f(x), \qquad \alpha_x \geq 0, \ \sum_{x\in X} \alpha_x = 1.$$

♢

We already know two important types of convex functions on vector and Hilbert spaces.

Proposition 2.8 (Norms are convex)**.**

1. *The function $\|\cdot\|$ is convex on any vector space.*
2. *The function $\|\cdot\|^2$ is convex on any Hilbert space.*

Proof: For 1) we simply use the norm axioms triangle inequality and positive homogeneity:

$$\|(1-\alpha)x + \alpha x'\| \leq \|(1-\alpha)x\| + \|\alpha x'\| = (1-\alpha)\|x\| + \alpha\|x'\|,$$

while (2.10)) is exactly the argument that we used in step 1. of the proof of Proposition 1.19. □

Interesting local extrema of convex functions have to be minima. The proof is simple and gets us acquainted to the concept.

Lemma 2.9. *Let $f : V \to \mathbb{R}$ be a convex function and $X \subseteq V$ a convex subset*[53] *of V.*

1. *If $x^* \in X$ a strict local **maximum** of f, then $x^* \in \partial X$.*
2. *Any local minimum of f on X is also a global minimum on X*

Proof: 1): Suppose that $x^* \notin \partial X$, then there are $x, x' \in X$ and $\alpha \in [0,1]$ such that $x^* = (1-\alpha)x + \alpha x'$. By moving x, x' closer to x^* along the line through that points, we can ensure that $f(x) < f(x^*)$ and $f(x') < f(x^*)$ since x^* is a strict local maximum. It then follows from convexity that

$$\begin{aligned} f(x^*) &= f\left((1-\alpha)x + \alpha x'\right) \leq (1-\alpha)\underbrace{f(x)}_{<f(x^*)} + \alpha\underbrace{f(x')}_{<f(x^*)} \\ &< (1-\alpha)f(x^*) + \alpha f(x^*) = f(x^*) \end{aligned}$$

[52] The same word for two different concepts which are, however, closely related.

[53] Since vector spaces are trivially convex, $X = V$ is well included, even if statement 1) becomes meaningless then since $\partial V = \emptyset$.

which is a contradiction.

For 2) assume that $x \in X$ is a local minimum and $x' \in X$ the global minimum with $f(x') < f(x)$. Since X is convex, all the points

$$x_t := (1-t)x + tx', \qquad t \in [0,1], \qquad x_0 = x,\ x_1 = x',$$

are contained in X and we have that

$$f(x_t) = f\left((1-t)x + tx'\right) \le (1-t)f(x) + t\underbrace{f(x')}_{<f(x)} < \left((1-t)+t\right) f(x) = f(x)$$

holds for any $t \in [0,1]$ and contradicts the assumption that x is a local minimum if we let $t \to 0$. □

Exercise 2.4 Show that a convex function has to be constant on the convex hull of its minima. ◇

For convex functions on Hilbert spaces there exists a slightly more involved but very powerful concept, see [43].

Definition 2.10 (Subgradient)**.** Let $f : \mathscr{H} \to \mathbb{R}$. An element $v \in \mathscr{H}$ is called a **subgradient** of a convex function $f : \mathscr{H} \to \mathbb{R}$ at $x \in \mathscr{H}$ if

$$f(x') \ge f(x) + \langle v, x' - x\rangle, \qquad x' \in \mathscr{H}. \tag{2.11}$$

The set $\partial f(x)$ of all subgradients at x is called the **subdifferential** and f is said to be **subdifferentiable** at x if $\partial f(x) \neq \emptyset$.

Example 2.11. Let us consider the subdifferential of the convex function $\|\cdot\| : \mathscr{H} \to \mathbb{R}$. In the case $x = 0$ a vector v belongs to the subgradient if and only if

$$\langle v, x'\rangle = \langle v, x' - 0\rangle \le \|x'\| - \|0\| = \|x'\|, \qquad x' \in \mathscr{H},$$

which happens if and only if $\|v\| \le 1$. This is easy to see: if $\|v\| \le 1$ then the Cauchy–Schwarz inequality yields

$$\langle v, x'\rangle \le |\langle v, x'\rangle| \le \|v\|\|x'\| \le \|x'\|, \qquad x' \in \mathscr{H},$$

while for $\|v\| > 1$ the choice $x' = v$ gives

$$\langle v, x'\rangle = \langle v, v\rangle = \|v\|^2 > \|v\| = \|x'\|$$

and establishes the converse.

The case $x \neq 0$ follows from a more general principle as the norm is differentiable there with derivative[54] $\|x\|^{-1}x$ as shown in (2.6).

Proposition 2.12. *Suppose that* $f : \mathscr{H} \to \mathbb{R}$ *is convex.*

1. *If* f *is* ***directionally differentiable*** *at* x *then*

$$v \in \partial f(x) \qquad \Leftrightarrow \qquad D_u f(x) \ge \langle v, u\rangle. \tag{2.12}$$

[54] Keep in mind: The derivative is a linear form!

2. *If* f *is* ***differentiable*** *at* x *then* $\partial f(x) = \{Df(x)\}$.

3. x *is a* ***local minimum*** *of* f *if and only if* $0 \in \partial f$.

Proof: For 1) we set $x' = x + hu$ for $h > 0$ and $u \in \mathscr{H}$. Then, for any $v \in \mathscr{H}$

$$\begin{aligned} f(x') - f(x) - \langle v, x' - x\rangle &= f(x + hu) - f(x) - h\langle v, u\rangle \\ &= h\left(\frac{f(x + hu) - f(x)}{h} - \langle v, u\rangle\right). \end{aligned}$$

Since $v \in \partial f(x)$ iff the above expression is nonnegative for all $u \in \mathscr{H}$ and $h \geq 0$, we have $v \in \partial f(x)$ iff

$$\langle v, u\rangle \leq \frac{f(x + hu) - f(x)}{h}, \qquad u \in \mathscr{H},\ h \geq 0,$$

and (2.12) follows from passing to the limit $h \to 0^+$.

For 2) we use (2.12) and note that in the case of differentiability we get for any $u \in \mathscr{H}$ that

$$\langle v, u\rangle \leq D_u f(x) = \langle Df(x), u\rangle$$

as well as

$$-\langle v, u\rangle = \langle v, -u\rangle \leq D_{-u} f(x) = \langle Df(x), -u\rangle = -\langle Df(x), u\rangle$$

which implies $\langle v, u\rangle \geq \langle Df(x), u\rangle$ and hence

$$\langle v, u\rangle = \langle Df(x), u\rangle, \qquad u \in \mathscr{H},$$

which is equivalent to $v = Df(x)$.

For 3) we recall Lemma 2.9 that allows us to assume that x is a **global minimum** so that $f(x + u) \geq f(x)$ for all $u \in \mathscr{H}$. This means that

$$f(x + u) - f(x) \geq 0 = \langle 0, u\rangle, \qquad u \in \mathscr{H},$$

which is equivalent to $0 \in \partial f(x)$. □

Example 2.13 (Example 2.11, continued)**.** The subdifferential of the norm function $\|\cdot\| = \langle\cdot,\cdot\rangle$ on a Hilbert space $\mathscr{H}$ is

$$\partial\|\cdot\|(x) = \begin{cases} \{v : \|v\| \leq 1\}, & x = 0, \\ \{\|x\|^{-1}x\}, & x \neq 0, \end{cases}$$

with exactly one minimum at $x = 0$ since $0 \in \{v : \|v\| \leq 1\}$.

2.3 Constrained optimization

So far we mainly considered **unconstrained** optimization problems which were characterized by a "$f'(x) = 0$" type of argument[55]. In many cases however, for example in the support vector machine, a certain function had to be optimized subject to certain **side conditions** or **constraints**, and one even uses these constraints to model the optimization problem appropriately.

[55]Indeed, most optimization tries to determine optima by checking where the derivative equals zero in some sense.

Definition 2.14 (Constrained optimization problem)**.** Given $f : V \to \mathbb{R}$, $g : V \to \mathbb{R}^p$, $h : V \to \mathbb{R}^q$, the **constrained** optimization problem is

$$\min f(x) \qquad \text{subject to } g(x) = 0,\ h(x) \geq 0, \tag{2.13}$$

with the p **equality constraints** g and the q **inequality constraints** h. The **feasible set** of this problem is

$$F := \{x \in V : g(x) = 0,\ h(x) \geq 0\} \tag{2.14}$$

and the problem (2.13) is called **feasible** if $F \neq \emptyset$.

Remark 2.15. Feasibility of an optimization problem is not easy to verify in general. As we will see, equality constraints are relatively easy to handle but very restrictive while inequality constraints are less restrictive for the prize of a more complex treatment.

As already mentioned, the "standard" way to treat differentiable constraint optimization problems is by means of **Lagrange multipliers** which we will formulate in the next theorem in the context of Hilbert spaces.

To that end, recall that for $g : \mathscr{H} \to \mathbb{R}$ the derivative $Dg : \mathscr{H} \to \mathscr{H}$ is a Hilbert space valued function[56] and therefore

$$g = \begin{bmatrix} g_1 \\ \vdots \\ g_p \end{bmatrix} : \mathscr{H} \to \mathbb{R}^p \qquad \Rightarrow \qquad Dg = \begin{bmatrix} Dg_1 \\ \vdots \\ Dg_p \end{bmatrix} : \mathscr{H} \to \mathscr{H}^n.$$

Would we write the derivatives with respect to a basis of $\mathscr{H}$, for example when $\mathscr{H} = \mathbb{R}^n$ for some n, then the derivative Dg is most conveniently written as matrix, called the **Jacobian** of g, cf. [49].

To formulate our main theorem, we need some more insight into the geometry of the **feasible set** F which again we can do in arbitrary Hilbert spaces.

Definition 2.16 (Tangent cone)**.**

1. For $M \subset \mathscr{H}$ und $x \in \mathscr{H}$ the (closed) **tangent cone** to M in x is the set

$$T(M, x) := \bigcap_{\varepsilon > 0} \overline{\{(y - x)\ \mathbb{R}_+ \ :\ y \in M,\ \|y - x\| \leq \varepsilon\}}.$$

2. For $M \subset \mathscr{H}$ the set

$$M' = \{y \in \mathscr{H} \ :\ \langle y, M \rangle \geq 0\}$$

is called **positive normal cone** to M.

The tangent cone plays a fundamental role in constrained optimization.

[56] In general, the derivative maps V to the **algebraic dual** V^*, the space of all continuous linear functionals, but we do not want to do so much Functional Analysis here.

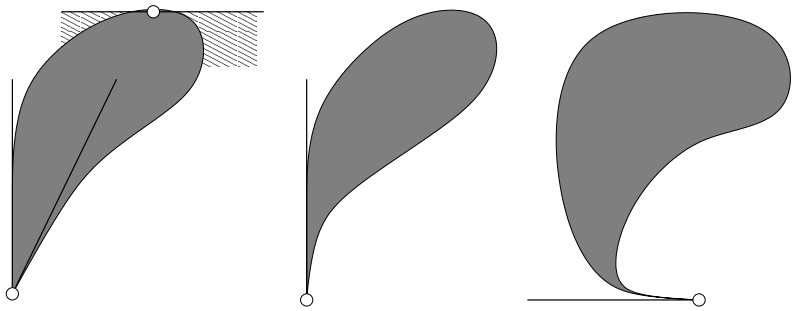


Figure 2.2: Illustration of the tangential cones of several shapes.

Proposition 2.17. *If $x \in M \subset \mathscr{H}$ is a local minimum of the differentiable function $f : M \to \mathbb{R}$ with continuous derivative Df, then*

$$Df \in T(M,x)' = \{y : \langle y, T(M,x)\rangle \geq 0\} \tag{2.15}$$

Proof: For $y \in T(M,x)$ there exist sequences[57] $M \ni x_k \to x$ and $\alpha_k \in \mathbb{R}_+$, $k \in \mathbb{N}$, such that $\alpha_k (x_k - x) \to y$ which in particular implies that $\alpha_k \to \infty$ or, equivalently, $\alpha_k^{-1} \to 0$.
Since $x_k \to x$ and since x is a local minimum, we get for sufficiently large $k \in \mathbb{N}$ that

$$\begin{aligned} 0 \;&\leq\; f(x_k) - f(x) = f(x + (x_k - x)) - f(x) = f\left(x + \alpha_k^{-1}(\alpha_k(x_k - x))\right) - f(x) \\ &=:\; f\left(x + \alpha_k^{-1} y_k\right) - f(x), \end{aligned}$$

hence

$$0 \leq \frac{f\left(x + \alpha_k^{-1} y_k\right) - f(x)}{\alpha_k^{-1}} \to D_y f = \langle Df(x), y\rangle,$$

since $y_k \to y$ and f has a continuous derivative. Thus, the assumption of a minimum yields

$$y \in T(M,x) \quad \Rightarrow \quad \langle Df(x), y\rangle \geq 0 \qquad \text{or} \qquad \langle \nabla f(x), T(M,x)\rangle \geq 0,$$

which is (2.15). □

Two more definitions and we are ready to go.

Definition 2.18 (Active constraints and linearizing cones)**.**

1. For a feasible point $x \in F$ where F is defined by (2.14), we denote by

$$J(x) = \{1 \leq j \leq q \;:\; h_j(x) = 0\} \subseteq \{1, \ldots, q\}$$

the set of **active constraints**.

[57] This is essentially the definition of the tangent cone plus the fact that we took the closure.

2. Der **linearizing cone** of the constraint functions g, h is defined as

$$\begin{aligned} L(x) &= L(x, g, h) \\ &= \{y \in \mathscr{H} : \langle Dg_j(x), y\rangle = 0,\ \langle \nabla h_k(x), y\rangle \geq 0,\ j = 1, \dots, p,\ k \in J(x)\}. \end{aligned}$$

The geometric intuition of the linearizing cone is to approach the tangent cone by means of tangents of the functions that determine it implicitly. For $\mathscr{H} = \mathbb{R}^n$ one always has $T(\Omega, x) \subseteq L(x)$ while the converse is not true in general. However, the respective sets are "academic", in most practical cases both agree.

Theorem 2.19. *If $x \in F$ is a local minimum of the continuously differentiable function $f : \mathscr{H} \to \mathbb{R}$ on a finite dimensional*[58] *Hilbert space $\mathscr{H}$ with*

$$L(x)' = T(F, x)', \tag{2.16}$$

then there exist $\lambda \in \mathbb{R}^p$ und $\mu \in \mathbb{R}^q_+$ such that

$$\begin{aligned} Df(x) - Dg(x)^T \lambda - Dh(x)^T \mu &= 0 && (2.17) \\ \mu^T h(x) &= 0 && (2.18) \end{aligned}$$

Remark 2.20. The vectors λ, μ are called **Lagrange multipliers** and the inner products are to be understood in the sense

$$Dg(x)^T \lambda = \sum_{j=1}^{p} \lambda_j \, Dg_j \in \mathscr{H}, \qquad Dh(x)^T \mu = \sum_{j=1}^{q} \mu_j \, Dh_j \in \mathscr{H}.$$

The conditions (2.17) and (2.18) are known as the **Kuresh–Kuhn–Tucker conditions** or **KKT conditions**; note that (2.17) is a Hilbert space identity while (2.18) is concerned with real numbers.

The main tool for the proof is a nice lemma from Linear Algebra that we do not want to prove here. It can be found in [60, Theorem 1.9, S. 17] oder [52, A2.1.4, S. 40] or in [40].

Lemma 2.21 ("**Farkas–Lemma**"). *For $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ there exists $0 \leq x \in \mathbb{R}^n$ with $Ax = b$ iff*

$$A^T y \geq 0 \qquad \Longrightarrow \qquad b^T y \geq 0. \tag{2.19}$$

Proof of Theorem 2.19: If $\mathscr{H}$ is finite dimensional, we can introduce an orthonormal basis H and represent the derivative as a **gradient**

$$\nabla f(x) := [\langle h, Df(x)\rangle : h \in H] \in \mathbb{R}^H \simeq \mathbb{R}^{\#H}.$$

If $x \in F$ is a local minimum of f we know, by Proposition 2.17 and our assumption (2.16) that

$$Df(x) \in T(F, x)' = L(x)', \qquad \Rightarrow \qquad \langle z, Df(x)\rangle \geq 0, \quad z \in L(x).$$

[58] We will use a tool from Linear Algebra soon and this works better with a matrix.

If we define the matrix $A \in \mathbb{R}^{H\times m}$, $m := 2p + \#J(x)$ as

$$A := A(x) = [\nabla g(x), -\nabla g(x), [\nabla h_j(x) \ : \ j \in J(x)]]$$

then the definition of $L(x)$ yields

$$z \in L(x) \qquad \Leftrightarrow \qquad A^T z \geq 0,$$

and we can rephrase the condition $\nabla f(x) \in L(x)'$ into[59]

$$A^T z \geq 0 \qquad \Rightarrow \qquad z^T \nabla f(x) \geq 0. \tag{2.20}$$

Now the **Farkas–Lemma**, Lemma 2.21 says that the set

$$\{\gamma \in \mathbb{R}^m \ : \ A\gamma = \nabla f(x),\ \gamma \geq 0\}$$

is nonempty and therefore there exists

$$\gamma = \left[\gamma_j^{(1)}, \gamma_j^{(2)}, \gamma_k^{(3)} \ : \ j = 1, \ldots, p,\ k \in J(x)\right] \in \mathbb{R}_+^{2p+\#J(x)},$$

such that

$$\begin{aligned} \nabla f(x) &= \sum_{j=1}^{p} \gamma_j^{(1)} \nabla g_j(x) - \sum_{j=1}^{p} \gamma_j^{(2)} \nabla g_j(x) + \sum_{j\in J(x)} \gamma_j^{(3)} \nabla h_j(x) \\ &= \sum_{j=1}^{p} \underbrace{\left(\gamma_j^{(1)} - \gamma_j^{(2)}\right)}_{=:\lambda_j} \nabla g_j(x) + \sum_{j\in J(x)} \underbrace{\gamma_j^{(3)}}_{=:\mu_j} \nabla h_j(x), \end{aligned}$$

and with $\mu_j = 0$, $j \in \{1, \ldots, q\} \setminus J(x)$, we get (2.17). The second conclusion (2.18), is obtained via

$$\mu^T h(x) = \sum_{j\in J(x)} \mu_j \underbrace{h_j(x)}_{=0} + \sum_{j\notin J(x)} \underbrace{\mu_j}_{=0} h_j(x) = 0.$$

□

There is a simple interpretation of Theorem 2.19 which we already used before. If one defines the **Lagrangian**

$$\mathscr{L} : \mathscr{H} \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}, \qquad \mathscr{L}(x, \lambda, \mu) := f(x) - \lambda^T g(x) - \mu^T h(x), \tag{2.21}$$

then the requirement (2.17) becomes

$$D\mathscr{L}(\cdot, \lambda, \mu) = 0, \tag{2.22}$$

a condition that is satisfied at any **minimum** or **maximum**[60] of $\mathscr{L}(\cdot, \lambda, \mu)$. We will consider this problem more carefully in the next section for particular choices of f, g, h.

[59] Now with respect to basis and coordinates.

[60] Which may or may not exist for some choice of λ, μ as x is also constrained by $g(x) = 0$ and $h(x) \geq 0$.

2.4 Convex problems and duality

To obtain more structure and stronger statements in that particular situation, we follow [40] and restrict the constrained optimization problem in the following way:

1. f is a **convex** function on $\mathscr{H}$,
2. we only have **inequality constraints** h,
3. the functions h_j are **concave**, i.e., $-h_j$ is **convex**, $j = 1, \ldots, q$.

Remark 2.22 (Normal forms)**.** Requirement 2) is not really a restriction as any equality constraint $g(x) = 0$ can be rewritten as $g(x) \geq 0$ and $-g(x) \geq 0$ and therefore is encoded in two inequality constraints[61]. Conversely, one can also rephrase inequality constraints as equality constraints by introducing **slack variables** y_j and

$$h_j(x) \to \widetilde{h}_j(x, y) := h_j(x) - y_j^2$$

and $h(x) \geq 0$ iff $0 = h(x) - y^2 = \widetilde{h}(x, y)$. However, the optimization runs over x and y now and the free variable y only enters the optimization problem *indirectly* via the constraint.

Nevertheless, these two operations can be used to transform[62] optimization problems between different normal forms used, for example, by different software[63] packages.

Exercise 2.5 Compare the KKT conditions for the optimization problem

$$\min_x f(x) \qquad \text{subject to } g(x) = 0,\ h(x) \geq 0,$$

and the modified, equivalent problem

$$\min_{x,y} f(x) \qquad \text{subject to } \begin{bmatrix} g(x) \\ \widetilde{h}(x, y) \end{bmatrix} = 0$$

after the introduction of slack variables. ◊

For $x, x' \in F$ and $\alpha \in [0, 1]$ the concave functions satisfy

$$h_j\left((1-\alpha)x + \alpha x'\right) \geq (1-\alpha)\underbrace{h_j(x)}_{\geq 0} + \alpha\underbrace{h_j(x')}_{\geq 0} \geq 0,$$

it follows that the set F is **convex** and also that

$$\mathscr{L}(x, \mu) = f(x) - \sum_{j=1}^{q} \mu_j\, h_j(x)$$

[61] A cheap trick, but frequently used in optimization.

[62] At least formally, the computational performance can differ.

[63] This is particularly true in the case of linear optimization where a lot of different programs with different normal forms are available.

is convex as long as $\mu \geq 0$.

Exercise 2.6 Show that if f, g are convex then so is $f + g$. ◊

Consequently, for $\mu \geq 0$ it is relatively easy to compute the function

$$q(\mu) := \inf_{x \in F} \mathscr{L}(x, \mu). \tag{2.23}$$

The value $q(\mu)$ is finite if F is bounded but in general we have $q : \mathbb{R}^q_+ \to \mathbb{R} \cup \{-\infty\}$, so that the definition

$$D_q := \{\mu \in \mathbb{R}^q_+ : q(\mu) > -\infty\} \tag{2.24}$$

makes sense.

Proposition 2.23. *The function* q *from* (2.23) *is* ***concave*** *and* D_q *is* ***convex****.*

Proof: Since $\mathscr{L}$ is linear in μ, we have that

$$\mathscr{L}(x, (1-\alpha)\mu + \alpha\mu') = (1-\alpha)\mathscr{L}(x, \mu) + \alpha\mathscr{L}(x, \mu')$$

so that

$$\begin{aligned} q((1-\alpha)\mu + \alpha\mu') &= \inf_{x \in F} \mathscr{L}(x, (1-\alpha)\mu + \alpha\mu') \\ &\geq (1-\alpha) \inf_{x \in F} \mathscr{L}(x, \mu) + \alpha \inf_{x \in F} \mathscr{L}(x, \mu') \\ &= (1-\alpha)q(\mu) + \alpha q(\mu'), \end{aligned}$$

showing that q is concave. If $\mu, \mu' \in D_q$ then

$$q((1-\alpha)\mu + \alpha\mu') \geq (1-\alpha)\underbrace{q(\mu)}_{>-\infty} + \alpha\underbrace{q(\mu')}_{>-\infty} > -\infty$$

also verifies the convexity of the set D_q. □

Definition 2.24 (Dual problem)**.** For the **primal problem**

$$\min_x f(x) \qquad \text{subject to } h(x) \geq 0, \tag{2.25}$$

its associated **dual problem** is defined as

$$\max_\mu q(\mu) \qquad \text{subject to } \mu \geq 0. \tag{2.26}$$

Lemma 2.25 (Weak duality)**.** *For each feasible choice of* x *and* μ *we have*

$$q(\mu) \leq f(x). \tag{2.27}$$

Proof: Almost trivial:

$$q(\mu) = \inf_{x'} \Big(f(x') - \mu^T h(x') \Big) \leq f(x) - \underbrace{\mu^T h(x)}_{\geq 0} \leq f(x)$$

holds as long as $\mu \geq 0$ and $h(x) \geq 0$, which are the feasibility conditions. □

Hence, the weak duality means that the function to be maximized is always bounded from above by the function to be minimized. We will see that the optimal solutions coincide and yield the optimal parameters.

The first result closes the gap we left with the optimization problem (1.45) in the support vector machine and says that indeed we can derive the proper multipliers by maximizing that dual problem[64]

Theorem 2.26 (Duality, "⇒"). *Suppose that* x *solves the minimization problem* (2.25) *and that* f *and* $-h_j$, $j = 1, \dots, q$ *are convex functions and differentiable at* x. *Then any* μ *that satisfies the* ***primal KKT conditions***

$$\begin{aligned} Df(x) - Dh(x)^T\mu &= 0 \\ \mu^T h(x) &= 0 \\ h(x) &\geq 0 \\ \mu &\geq 0 \end{aligned} \tag{2.28}$$

is a solution of the ***dual problem*** (2.26).

Proof: Since $\mathscr{L}(\cdot, \mu)$ is convex and differentiable[65], its subgradient satisfies

$$\partial\mathscr{L}(\cdot, \mu)(x) = \{D\mathscr{L}(\cdot, \mu)(x)\}$$

and (2.11) yields together with the fact that $D\mathscr{L}(\cdot, \mu)(x) = Df(x) - Dh(x)^T\mu = 0$ by (2.28) the estimate

$$\mathscr{L}(x', \mu) \geq \mathscr{L}(x, \mu) + \underbrace{\langle D\mathscr{L}(\cdot, \mu)(x), x' - x\rangle}_{=0} = \mathscr{L}(x, \mu), \qquad h(x') \geq 0,$$

hence, again by (2.28),

$$q(\mu) = \inf_{h(x')\geq 0} \mathscr{L}(x', \mu) = \mathscr{L}(x, \mu) = f(x) - \underbrace{\mu^T h(x)}_{=0} = f(x).$$

Since $q(\mu') < f(x')$ for all $\mu' \geq 0$ and $h(x') \geq 0$ by the **weak duality** (2.27), it follows that the above μ indeed maximizes q. □

The converse is slightly trickier and requires some more assumptions and notions. The first is a nonsingularity condition on the constraints.

Definition 2.27 (LICQ). The constraints $h : \mathscr{H} \to \mathbb{R}^q$ are said to satisfy the *linear independence constraint qualification*[66] (**LICQ**) or simply are called **nonsingular**[67] at $x \in F$ if the set

$$\{Dh_j(x) : h_j(x) = 0\} \tag{2.29}$$

of active constraint derivatives is linearly independent.

[64] Keep in mind that we computed the dual there by substituting the minimum of the expression!

[65] To be differentiable at a point is always a property of the function in some small neighborhood of the point.

[66] Terminology from [40].

[67] Terminology from [57].

Exercise 2.7 Show that the LICQ condition implies that $L(x)' = T(F, x)'$ in the sense of (2.16). ◊

Exercise 2.8 Show that if the constraint functions are continuously differentiable and the LICQ holds at x, then there exists an open set U with $x \in U$ such that the LICQ holds for

$$\{x' : x' \in U, J(x') = J(x)\}, \qquad J(x) = \{x : h_j(x) > 0\},$$

i.e. for any point from the neighborhood with the same active set. ◊

The "converse" result is only a partial converse with more assumptions[68]

Theorem 2.28 (Duality, "$\Leftarrow$"). *Suppose that f and $-h_j$, $j = 1, \dots, q$, are convex and continuously differentiable and that x is a solution of the* ***primal problem*** (2.25) *where LICQ is satisfied.*

If μ is a solution of the dual problem (2.26) *for which $\mathscr{L}(\cdot, \mu)$ is a* ***strictly convex*** *function that assumes its infimum*[69] *at x', then $x' = x$ and $f(x) = \mathscr{L}(x', \mu)$.*

Proof: Assume, for the contrary, that $x \neq x'$. Since x is a minimum and since LICQ guarantees[70] (2.16), we can apply Theorem 2.19 to obtain the existence of some $\mu' \geq 0$ that satisfies the **KKT conditions** (2.28). From Theorem 2.26 we can then conclude that[71]

$$\mathscr{L}(x, \mu') = q(\mu') = q(\mu) = \mathscr{L}(x', \mu)$$

Since x' minimizes $\mathscr{L}(\cdot, \mu)$, differentiability yields $D\mathscr{L}(\cdot, \mu)(x) = 0$ and the *strict* convexity of $\mathscr{L}(\cdot, \mu)$ yields that

$$\mathscr{L}(x, \mu) - \mathscr{L}(x', \mu) > \langle D\mathscr{L}(\cdot, \mu)(x'), x - x' \rangle = 0,$$

hence

$$\mathscr{L}(x, \mu) > \mathscr{L}(x', \mu) = \mathscr{L}(x, \mu')$$

or, by (2.28),

$$f(x) - h(x)^T \mu > f(x) - h(x)^T \mu' \qquad \Rightarrow \qquad 0 \leq h(x)^T \mu < h(x)^T \mu' = 0$$

which is a contradiction. Hence $x = x'$ as claimed. □

More on convex optimization, the duality in convex constraint optimization and the **saddle point** interpretation of duality can be found in [43, 57].

2.5 How to compute optimizers?

After having had plenty of fun with the theory of optimization, convexity and duality, we return to the more realistic issue of how to really compute solutions of optimization problems. In fact, we will briefly recall the main algorithms for **unconstrained optimization**, the ones for constraint approximation usually combine those with feasibility checks.

Moreover, we assume that f is differentiable and that Df can be computed. If this is not the case, derivative free methods have to be applied, see [47].

[68] Including the existence of a solution of the primal problem.
[69] It could be negative infinite.
[70] See or better do Exercise 2.7.
[71] μ and μ' are both locations of maxima of q, the first by duality, the second by assumption.

2.5.1 Descent algorithms

The standard method in smooth nonlinear optimization are so called **descent** methods that approach a local minimum by always walking "downhill". Mathematically, this is based on the following concept.

Definition 2.29. An element $u \in \mathscr{H}$ is called a **descent direction** for $f : \mathscr{H} \to \mathbb{R}$ at x if $D_u f(x) < 0$; if f is differentiable this is equivalently described by $\langle Df(x), u\rangle < 0$.

If u is a descent direction, we can consider the function $f_u(t) := f(x + tu)$, $t \geq 0$, which satisfies

$$f_u'(0) = D_u f(x) = \langle Df(x), u\rangle.$$

If f has a continuous[72] derivative, then there exists $\tau > 0$ such that the continuous function $f_u'(t)$ is negative for $t \in [0, \tau]$

$$f(x + tu) = f(x) + \int_0^t f_u'(s)\,ds = f(x) + \int_0^t \underbrace{\langle Df(x + su), u\rangle}_{<0}\,ds < f(x) \qquad (2.30)$$

This is the basic iterative algorithm:

1. Choose any initial x_0.
2. For $j = 1, 2, \ldots$
   (a) determine a descent direction u_j for f at x_{j-1},
   (b) determine $t_j > 0$ such that $f(x_{j-1} + t_j u_j) < f(x_{j-1})$,
   (c) set $x_j = x_{j-1} + t_j u_j$.

The argument stops if there is no descent direction any more, i.e., if $D_u f(x) \geq 0$, $u \in \mathscr{H}$, which is the necessary condition for a minimum considered (2.8). If f is continuously differentiable, the algorithm stops if $Df(x) = 0$. Such a point x is called **critical**, and finding critical points is usually the best optimization can do.

Practically, there is a fundamental question: How to choose u_j and t_j? The "natural", but at least naive choice for u_j is the **steepest descent**, $u_j = -Df(x_{j-1})$. Indeed, it is the solution of a minimization problem:

$$\min_{\|u\|=1} \langle Df(x), u\rangle \qquad \Rightarrow \qquad u = \frac{-Df(x)}{\|Df(x)\|}. \qquad (2.31)$$

Nevertheless, steepest descent is not always a good idea as it can lead to directions that do not lead to the minimum very fast but circle around and result in algorithms that converge slowly if at all. This gives rise to other search directions like **conjugate gradients** or **Newton directions**, cf. [47].

For the **stepsize** t_j there are some rules available which we briefly list next.

Definition 2.30 (Stepsize conditions)**.** For two given constants[73] $0 < c_1 < c_2 < 1$

[72] It does not make much sense to consider seriously discontinuous functions as then derivative is not really useful information. In practice, little can be done without continuity and even continuity alone is too weak when looking carefully. We will see this in the context of neural networks.

[73] This are "process parameters" to be chosen by a user.

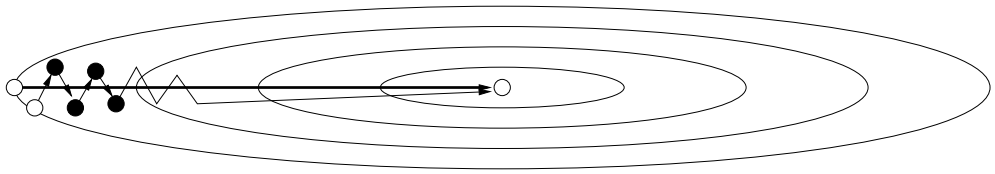


Figure 2.3: The reason why **steepest descent** is not always a good idea: in "flat ellipses" like here (height lines), the steepest direction is almost perpendicular to the best direction that would point directly to the minimum.

and a **descent direction** u the stepsize t is said to satisfy

1. the **Armijo condition** if

$$f(x+tu) - f(x) \leq t\, c_1 \langle Df(x), u\rangle. \tag{2.32}$$

2. the **Wolfe conditions**[74], also called **Powell conditions**[75], if

$$\begin{array}{rcl} f(x+tu) - f(x) & \leq & t\, c_1 \langle Df(x), u\rangle, \\ \langle Df\,(x+tu)\,, u\rangle & \geq & c_2 \langle Df(x), u\rangle. \end{array} \tag{2.33}$$

3. the **strong Wolfe conditions** if

$$\begin{array}{rcl} f(x+tu) - f(x) & \leq & t\, c_1 \langle Df(x), u\rangle \\ |\langle Df(x+tu), u\rangle| & \leq & c_2\, |\langle Df(x), u\rangle| \end{array} \tag{2.34}$$

These requirements can be met.

Lemma 2.31. *If f is continuously differentiable, u a descent direction and*

$$\inf\{f\,(x+tu)\ :\ t \in \mathbb{R}_+\} > -\infty,$$

then there exist, for any choice $0 < c_1 < c_2 < 1$, values $t \in \mathbb{R}_+$ that satisfy (2.33) *or* (2.34)*, respectively*[76]*.*

Proof: Denote by $\ell(t) = f(x) + t\, c_1 D_u f(x)$, $t \in \mathbb{R}_+$, the linear function $\ell : \mathscr{H} \to \mathbb{R}$, that interpolates f_u and f_u' at $t = 0$. Since

$$\lim_{t\to\infty} \ell(t) = -\infty < \inf_{t\in\mathbb{R}_+} f_u(t),$$

there exist a *minimal* $t' > 0$, such that

$$f(x) + t'\, c_1 D_u f(x) = \ell\,(t') = f_u\,(t') = f\,(x+t'u)$$

[74] Due to [59].
[75] Due to [42].
[76] And therefore also (2.32).

and $c_1 < 1$ yields $\ell(t) \geq f_u(t)$, $t \leq t'$. In other words,

$$f(x + tu) - f(x) \leq t\, c_1 \langle Df(x), u\rangle, \qquad t \in (0, t')$$

which is (2.32). By the **mean value theorem** there exists $t^* \in (0, t')$ such that

$$f_u(t') - f_u(0) = (t' - 0)\ f_u'(t^*)\,,$$

hence

$$t'\langle Df(x + t^* y), u\rangle = f(x + t'u) - f(x) = t' \underbrace{c_1}_{<c_2} \underbrace{\langle Df(x), u\rangle}_{<0} > t'\, c_2 \langle Df(x), u\rangle.$$

Dividing by $t' > 0$, we obtain the second conditions in (2.33) and also in (2.34) for t in some neighborhood of t^*. □

If the stepsizes are chosen according to Definition 2.30, one can prove that descent methods converge in quite general situations. An example, without proof, is as follows.

Theorem 2.32. *Suppose that $f \in C^1(\mathscr{H})$ has a **Lipschitz continuous** derivative and is bounded from below:*

$$\inf\{f(x) \ :\ x \in \mathscr{H}\} > -\infty.$$

If for an arbitrary starting value x_0 one constructs the sequence

$$x_j = x_{j-1} + t_j\, u_j, \qquad j \in \mathbb{N},$$

with descent directions u_j and t_j satisfying (2.33)*, then*

$$\sum_{j=0}^{\infty} \cos^2\theta_j\ \left\|Df(x_j)\right\|_2^2 < \infty, \tag{2.35}$$

for any starting value x_0, where

$$\cos\theta_j := \frac{\langle Df(x_j), u_j\rangle}{\left\|Df(x_j)\right\|\ \left\|u_j\right\|}, \qquad j \in \mathbb{N}.$$

2.5.2 Variants of Newton's method

Another approach to localizing extrema for differentiable functions is to find zeros of the derivative. This needs methods to compute zeros of nonlinear functions and the most prominent one is **Newton's method**. In one variable this corresponds to the iteration

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}, \qquad j \in \mathbb{N}_0, \tag{2.36}$$

with the geometric interpretation of intersecting the tangent of f at x_j with the x–axis. Newton's method is the "one point variant" of the **secant method**

$$x_{j+1} = x_j - \frac{x_j - x_{j-1}}{f(x_j) - f(x_{j-1})}\ f(x_j)\,, \tag{2.37}$$

that intersects the secant between two preceding points with the x–axis.

If $f : \mathscr{H} \to \mathscr{H}$ is a function between Hilbert spaces and H is a basis of $\mathscr{H}$, then we can still[77] define a **directional derivative**

$$D_u f := \lim_{h \to 0^+} \frac{f(\cdot + hu) - f(\cdot)}{h}$$

and if f is differentiable, then the function $u \mapsto D_u f$ is linear mapping from $\mathscr{H} \to \mathscr{H}$. If $H \subset \mathscr{H}$ is a basis of $\mathscr{H}$, we can write u

$$u = \sum_{h \in H} \langle h^*, u \rangle \, h, \qquad \langle h^*, h' \rangle = \delta_{h,h'}, \quad h, h' \in H, \tag{2.38}$$

where $h^* \in \mathscr{H}$ denotes the **dual** or **biorthogonal** element[78] for h which must exist due to the **reflexivity** of the Hilbert space. If H is an **orthonormal basis**, then $h^* = h$. Consequently, linearity and an application of the decomposition (2.38) yield for any $v \in \mathscr{H}$

$$\langle D_u f, v \rangle \;=\; \sum_{h \in H} \langle h^*, u \rangle \langle D_h f, v \rangle \sum_{h, h' \in H} \langle h^*, u \rangle \langle D_h f, h' \rangle \langle h'^*, v \rangle$$

Hence,

$$Df = \left[\langle h', D_h f \rangle : \begin{array}{c} h' \in H \\ h \in H \end{array} \right] \tag{2.39}$$

describes a bilinear form or the counterpiece of a matrix.

Example 2.33. If $\mathscr{H} = \mathbb{R}^d$ and e_j are the canonical standard basis, then

$$\langle e_j, D_{e_k} f \rangle = \frac{\partial f_j}{\partial x_k}$$

and Df is the well known **Jacobian** of $f : \mathbb{R}^d \to \mathbb{R}^d$.

Based on (2.39) we can define the **linear system** in the **weak form**

$$\langle f(x), v \rangle = \langle Df(x)\, y, v \rangle = \langle D_y f(x), v \rangle, \qquad v \in \mathscr{H},$$

which leads to

$$\begin{aligned} 0 \;&=\; \sum_{h' \in H} \langle f(x), h' \rangle \langle h'^*, v \rangle - \sum_{h, h' \in H} \langle h^*, y \rangle \langle D_h f(x), h' \rangle \langle h'^*, v \rangle \\ &=\; \sum_{h' \in H} \left\langle f(x) - \sum_{h \in H} \langle h^*, y \rangle \langle D_h f(x), h' \rangle, v \right\rangle \langle h'^*, v \rangle \end{aligned}$$

and therefore to

$$f(x) = \sum_{h \in H} \langle h^*, y \rangle \langle D_h f(x), h' \rangle = \left(Df(x) \langle H, y \rangle \right)_{h'}, \qquad h' \in H, \tag{2.40}$$

which is a linear system in coefficients of y with respect to H with respect to the **Jacobian** of f at x.

[77] These are all Hilbert space operation.

[78] In this notation, the * is an operator like transposition for matrices.

Definition 2.34. The solution y of (2.40) is called the **Newton direction** of f at x.

The strategy for optimization methods based on Newton directions corresponds to finding critical points of the derivative:

1. Given $f : \mathscr{H} \to \mathbb{R}$ and the problem $\min_x f(x)$
2. use the derivative $F : \mathscr{H} \to \mathscr{H}$, i.e. $F(x) = Df(x)$ and a **starting point** x_0.
3. For $k = 1, 2, \ldots$
   (a) compute the **Newton direction** $y_k = DF(x_{k-1})^{-1}F(x_{k-1})$ by solving (2.40),
   (b) compute a **stepsize** t_k, for example[79] $t_k = 1$,
   (c) set $x_k = x_{k-1} - t_k y_k$.

Under certain circumstances[80] Newtons method provides **local convergence**, i.e., it converges if the starting value is close enough to the zero. Since computing zeros of a derivative cannot distinguish between minima and maxima, the quality of the starting value is fundamental for the success of the method.

A typical theorem in this regard looks as follows, formulated for $\mathscr{H} = \mathbb{R}^d$.

Theorem 2.35. *If $f \in C^2(\mathbb{R}^d)$ and $\nabla^2 f$ is Lipschitz continuous in a neighborhood of a **strict minimum** x^* of f. Then there exists a neighborhood U of x^*, such that for all $x_0 \in U$ the Newton iteration with stepsize 1*

1. *converges to x^*:*
$$\lim_{k\to\infty} x_k = x^*. \tag{2.41}$$
2. *is **quadratically convergent**:*
$$\sup_{k\in\mathbb{N}_0} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < \infty. \tag{2.42}$$
3. *has **quadratically convergent** gradients with limit zero:*
$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0, \qquad \sup_{k\in\mathbb{N}_0} \frac{\left\|\nabla f(x_{k+1})\right\|}{\left\|\nabla f(x_k)\right\|^2} < \infty. \tag{2.43}$$

Remark 2.36. Another computational problem is that in each iteration the matrix DF has to be recomputed which is a $\#H \times \#H$ "matrix" which can be a substantial effort for larger spaces. Moreover, in many cases it is not so easy to compute the derivatives *explicitly* and numerical differentiation methods of limited accuracy have to be applied.

To overcome that problem, people in Optimization have developed a lot of algorithms which only apply a partial rank one update to the matrix; this leads to the huge family of **quasi Newton methods** that are well documented in the literature.

[79] That would be Newton's method.

[80] Essentially some smoothness of Df and boundedness of certain expressions, see [46, 47].

2.6 Penalty and regularization

In the last part of this section we return to constraint optimization problems but now with "soft" constraints. One immediate reason is explained quite easily

Remark 2.37 (Sensitivity of equality constraints)**.** In the presence of **equality constraints** the **feasible set** $\{x : g(x) = 0\}$ is usually a "thin" set and often very sensitive to perturbation. In fact, if $Dg(x) \neq 0$ for some feasible x with $g(x) = 0$, then there exist points arbitrarily close to x with $g(x) \neq 0$, hence these points are not feasible any more.

Definition 2.38 (Penalty)**.** The **penalty function** for f and the equality constraints g is

$$f_\lambda = f(x) + \lambda \|g(x)\|^2 = f(x) + \lambda \sum_{j=1}^{p} g_j^2(x), \qquad \lambda \in \mathbb{R}_+, \tag{2.44}$$

and for equality and inequality constraints

$$f_{\lambda,\mu} = f(x)+\lambda\|g(x)\|^2+\mu\|h_-\|^2 = f(x)+\lambda \sum_{j=1}^{p} g_j^2(x)+\mu \sum_{j=1}^{q} (h_j)_-^2(x), \qquad \lambda, \mu \in \mathbb{R}_+, \tag{2.45}$$

where for $f : \mathscr{H} \to \mathbb{R}$

$$f_-(x) := \frac{1}{2}\left(|f(x)| - f(x)\right) = \begin{cases} 0, & f(x) \geq 0, \\ -f(x), & f(x) < 0. \end{cases}$$

Remark 2.39.

1. At first view, the penalty function looks quite like a **Lagrangian**, but first it is not the function that appears but its square[81] and the parameters are only scalar ones[82].

2. Since we are still considering *minimization* problems, this approach penalized violation of the constraints and the effect of this penalty is controlled by the variable λ or the variables λ, μ, respectively. If they are set to zero, the constraint is ignored, the larger they get, the more relevant the constraints become.

3. In any way, the minimization problem $\min_x f_{\lambda,\mu}(x)$ is an unconstrained one now, hence could, in the case of (2.44) be solved in the "usual" manner by solving

$$0 = Df_\lambda(x) = Df(x) + 2\lambda \sum_{j=1}^{p} g_j(x)\, Dg_j(x)$$

with respect to x for the fixed parameter λ.

[81] The absolute value would do as well, in fact any even function $\mathbb{R} \to \mathbb{R}$ that is monotonically increasing on $\mathbb{R}_+$ and zero at the origin would do the job, in other words, any **loss function**.

[82] Even if one could model the influence of any side condition by a separate factor as well.

Penalty methods in optimization approximately solve the unconstrained minimization problems with respect to (2.44) or (2.45) and at the same time increase the parameters λ, μ. More precisely, in the case of (2.44), one choose two sequences ε_k, τ_k converging to zero and computes[83], in the kth iteration, a point x_k such that

$$\left\| Df_{\varepsilon_k^{-1} f(x_k)} \right\| \leq \tau_k. \tag{2.46}$$

A typical result then looks as follows, again in the context of finite dimensional Hilbert spaces.

Theorem 2.40. *For $g \in C^1(\mathbb{R}^n)$, and nonnegative sequences ε_k, τ_k, $k \in \mathbb{N}$, converging to zero let x^* be an accumulation point of a sequence $x_k \in \mathscr{H}$, satisfying (2.46). If the derivatives $\nabla g_j(x^*)$, $j = 1, \ldots, p$, are linearly independent then there exists a **Lagrange multiplier** $\lambda \in \mathbb{R}^p$ such that*

$$\nabla f(x^*) - \lambda^T \nabla g(x^*) = 0 \tag{2.47}$$

and we have

$$\lambda = \lim_{j \to \infty} -\frac{g\left(x_{k_j}\right)}{\varepsilon_{k_j}}, \qquad x^* = \lim_{j \to \infty} x_{k_j}. \tag{2.48}$$

There is different interpretation of this approach as the **regularization** of a linear system which in fact brings us back to where we started in learning and is actually used quite a bit in the context of learning which makes it a nice end to the optimization chapter.

Let us return to the problem of approximating data $v \in \mathscr{H}$ by a finite dimensional **model space** $W \subset \mathscr{H}$ spanned by a finite set Φ. We now explicitly drop the assumption that Φ were a **basis** of W, it only has to be a generating set[84]. Nevertheless, note that the coefficient vector $a \in \mathbb{R}^\Phi$ such that

$$\phi(a) = \sum_{\varphi \in \Phi} a_\varphi \varphi$$

is the best approximant to v in the sense

$$\min_{a \in \mathbb{R}^\Phi} \|v - \phi(a)\|$$

iff it is the solution of the linear system (1.30), which we recall to be

$$[\langle v, \varphi \rangle : \varphi \in \Phi] = [\langle \varphi, \varphi' \rangle : \varphi, \varphi' \in \Phi]\ a. \tag{2.49}$$

If $\operatorname{rank} G_\Phi < \#\Phi$, then (2.49) has infinitely many solutions[85] and we have to choose among them, so why not take the best. To that end, we consider (2.49) only as a constraint, pick any $f : \mathbb{R}^\Phi \to \mathbb{R}$ and solve

$$\min_a f(a), \qquad \text{subject to } [\langle v, \varphi \rangle : \varphi \in \Phi] = [\langle \varphi, \varphi' \rangle : \varphi, \varphi' \in \Phi]\ a. \tag{2.50}$$

[83] Mostly by the methods described above, i.e., Newton or descent.

[84] In the context of dictionaries and **basis pursuit** this will become relevant.

[85] This is sometimes called **small data**: We have a lot of possible explanations but not enough data to really verify or falsify them which a *unique* solution a would.

This could either be solved by means of **Lagrange multipliers** or be turned into the penalized optimization problem

$$\min_{a} f(a) + \lambda \|G_\Phi a - g_{\Phi,v}\|^2, \qquad \lambda \in \mathbb{R}_+.$$

Replacing λ by λ^{-1} and multiplying with λ, we get the more common form

$$\min_{a} \|G_\Phi a - g_{\Phi,v}\|^2 + \lambda f(a), \qquad \lambda \in \mathbb{R}_+, \tag{2.51}$$

where, depending on the field of application f is called **regularizer**, **cost function**, **loss function** or **energy functional** while $\|G_\Phi a - g_{\Phi,v}\|^2$ is often referred to as the **data fidelity** term.

The most classical regularizer is

$$f(a) = \|a\|_2^2 = \sum_{\varphi\in\Phi} a_\varphi^2$$

which leads to the minimization of

$$\begin{aligned} \|G_\Phi a - g_{\Phi,v}\|_2^2 + \lambda\|a\|_2^2 &= a^T G_\Phi^T G_\Phi a - 2(G_\Phi^T g_{\Phi,v})^T a + \|g_{\Phi,v}\|_2^2 + \lambda\, a^T a \\ &= a^T \left(G_\Phi^T G_\Phi + \lambda I\right) a - 2(G_\Phi^T g_{\Phi,v})^T a + \|g_{\Phi,v}\|_2^2 \end{aligned}$$

with respect to a which is done by the solution of

$$\left(G_\Phi^T G_\Phi + \lambda I\right) a = G_\Phi^T g_{\Phi,v}.$$

Since $G_\Phi^T G_\Phi$ is positive semidefinite, the matrix $G_\Phi^T G_\Phi + \lambda I$ is invertible for any $\lambda > 0$ since

$$y^T \left(G_\Phi^T G_\Phi + \lambda I\right) y = \underbrace{\|G_\Phi y\|^2}_{\geq 0} + \underbrace{\|y\|_2^2}_{>0} > 0, \qquad 0 \neq y \in \mathbb{R}^\Phi.$$

Another possible regularizer is

$$f(a) = \|\phi(a)\|^2 = \langle \phi(a), \phi(a)\rangle = a^T G_\Phi a$$

and we can either consider the expression

$$\begin{aligned} &\|v - \phi(a)\|^2 + \lambda\,\|\phi(a)\|^2 = \langle v - \phi(a), v - \phi(a)\rangle + \lambda\,\langle \phi(a), \phi(a)\rangle \\ &= \|v\|^2 - 2\langle v, \phi(a)\rangle + (1+\lambda)\|\phi(a)\|^2 = \|v\|^2 - 2g_{\Phi,v}^T a + (1+\lambda) a^T G_\Phi a \end{aligned}$$

which is minimized by the solution of

$$G_\Phi a = \frac{1}{1+\lambda} g_{\Phi,v}$$

giving a[86] rescaled version of the solution of the original problem, or we can consider

$$\|G_\Phi a - g_{\Phi,v}\|_2^2 + \lambda\|\Phi(a)\|_2^2 = a^T \left(G_\Phi^T G_\Phi + \lambda G_\Phi\right) a - 2(G_\Phi^T g_{\Phi,v})^T a + \|g_{\Phi,v}\|_2^2$$

[86] Quite boring ...

which, however, suffers from the same problem as the original minimizer: the matrix is only positive *semi*definite as any y with $G_\Phi y = 0$ also satisfies $G_\Phi^T G_\Phi y = 0$.

This simple example already shows that the choice of the **loss function** strongly influences the optimization process. A really interesting one happens to be

$$\|G_\Phi a - g_{\Phi,\nu}\|_2^2 + \lambda\|a\|_1 = \|G_\Phi a - g_{\Phi,\nu}\|_2^2 + \sum_{\varphi\in\Phi} |a_\varphi|$$

which often leads to a **sparse solution** with only very few nonzero components of a. We'll get to this later when considering sparsity.

Which is probably one of the reasons those of us who love contemporary fiction love it as we do. We're alone with it. It arrives without references, without credentials we can trust.

M. Cunningham, *The New Yorker Online, 10.7.2012*

3 Kernels

Now we will switch to one of the most important concepts in Learning Theory, namely kernel methods. There are several ways to introduce this concept: We could start with an intuitive definition of the SVM and then define the tools like kernels as it is done in the "learning" literature. On the other hand, kernels are much more classical from Functional Analysis, cf. [61], and even used in the theory of continuous games [23]. So I prefer to follow the historical track and first introduce the theory that was available at the time when kernel learning was born[87].

Again we consider the problem of learning labels y_x obtained on a *finite* training set $X \subset \mathscr{X}$, where $\mathscr{X}$ can be a fairly unstructured **metric space**[88]. For completeness, let us recall the definition.

Definition 3.1 (Metric)**.** A **metric** on a set $\mathscr{X}$ is a mapping $d : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ such that

1. $d(x, x') \geq 0$ and $d(x, x') = 0$ iff $x = x'$.
2. $d(x, x') = d(x', x)$, $x, x' \in \mathscr{X}$.
3. $d(x, x') \leq d(x, y) + d(y, x')$,$x, x', y \in \mathscr{X}$.

A set $\mathscr{X}$ endowed with a metric d is called a **metric space**.

We want to learn functions $f : \mathscr{X} \to \mathbb{R}$ such that $f(x) = y_x$, $x \in X$, and we want to use Hilbert space structures, even if $\mathscr{X}$ does not allow for integration or inner products. Fortunately, there is a tool for that.

[87] So much for the disclaimer.

[88] This is not even a restriction since any set can be equipped with the **discrete metric** $d(x, x') = 1 - \delta_{x,x'}$.

3.1 Reproducing kernel Hilbert spaces

Our goal is to construct a Hilbert space $\mathscr{H}$ whose elements can be considered as functions $\mathscr{X} \to \mathbb{R}$. The trick is based on the following concept.

Definition 3.2 (Kernels)**.** Let $\mathscr{X}$ be a metric space.

1. A **kernel** is a function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$.
2. A kernel is called symmetric if $k(x, x') = k(x', x)$.
3. A kernel is called **positive semidefinite** if for any *finite* subset $X \subset \mathscr{X}$ the matrix
$$K(X) = \left[k(x, x') : \begin{array}{c} x \in X \\ x' \in X \end{array} \right]$$
is positive semidefinite.
4. A kernel is called a **Mercer kernel** if it is continuous[89], symmetric and positive semidefinite.

On the "classical" Hilbert space $\mathscr{X} = L_2(\mathbb{R})$, kernels are used to define bilinear forms $\mathscr{X} \times \mathscr{X} \to \mathbb{R}$ as

$$(f, g)_k := \int_{\mathbb{R}} \int_{\mathbb{R}} f(x) k(x, x') g(x') \, dx' dx$$

which is the continuous counterpart of $(x, y)_A = x^T A y$. The generalization is to define a linear operator $A_k : \mathscr{X} \to \mathscr{X}$ as $A_k g(x) = \langle k(x, \cdot), g \rangle$ and $(f, g)_k = \langle f, A_k g \rangle$. Like matrices, kernels can play an ambiguous role, namely as the **linear operator** A_k or as the **bilinear form** $(\cdot, \cdot)_k$.

Remark 3.3 (Mercer kernels)**.**

1. Any Mercer kernel is nonnegative since the positive semidefiniteness of $K(X)$ with $X = \{x\}$ implies $k(x, x) \geq 0$.
2. Since any symmetric positive semidefinite matrix $A \in \mathbb{R}^{n \times n}$ satisfies
$$a_{jk}^2 \leq a_{jj} \, a_{kk}, \qquad j, k = 1, \ldots, n, \tag{3.1}$$
we also have that $k(x, x')^2 \leq k(x, x) \, k(x', x')$, $x, x' \in \mathscr{X}$.

Exercise 3.1 Prove (3.1). ◊

Mercer kernels always define very interesting Hilbert spaces in a natural and *unique* way.

Theorem 3.4. *If $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is a **Mercer kernel** then there exists a unique Hilbert space $\mathscr{H}$ of functions on $\mathscr{X}$ such that*

1. *For all $x \in \mathscr{X}$ we have that $k(x, \cdot) \in \mathscr{H}$.*

[89] This is a metric property on $\mathscr{X}$ and trivially satisfied for the discrete metric.

2. *The vector space* $\operatorname{span}\{k(x,\cdot) : x \in \mathscr{X}\}$ *is dense in* $\mathscr{H}$.

3. *One has*

$$f(x) = \langle f, k(x,\cdot)\rangle, \qquad f \in \mathscr{H}, x \in \mathscr{X}. \tag{3.2}$$

Moreover,

$$|f(x)| \le \sqrt{k(x,x)}\,\|f\|. \tag{3.3}$$

These special Hilbert spaces that are defined by the Mercer kernel are so important that they have a name of their own.

Definition 3.5 (Reproducing kernel Hilbert space)**.** A Hilbert space $\mathscr{H}$ of functions on a metric space $\mathscr{X}$ is called[90] a **reproducing kernel Hilbert space** or **RKHS**, for short, if there exists a kernel $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ such that

$$k(x,\cdot) \in \mathscr{H} \qquad \text{and} \qquad f(x) = \langle f, k(x,\cdot)\rangle, \qquad x \in X.$$

Remark 3.6. In summary, Theorem 3.4 says that any Mercer kernel defines a reproducing kernel Hilbert space.

Proof of Theorem 3.4: We construct $\mathscr{H}$ by building the inner product[91] for the Hilbert space. To that end, we start on the linear space $H := \operatorname{span}\{k(x,\cdot) : x \in \mathscr{X}\}$. For any function $h \in H$ there exists a finite subset[92] $X \subset \mathscr{X}$ such that

$$h = \sum_{x\in X} h_x\, k(x,\cdot).$$

We now *define* an inner product $\langle\cdot,\cdot\rangle$ on H as

$$\langle h, h'\rangle := \sum_{x\in X}\sum_{x'\in X'} h_x h'_{x'} k(x,x') \qquad h, h' \in H. \tag{3.4}$$

This is obviously symmetric in h and h' and bilinear. Definiteness is a bit trickier: suppose that $\langle h, h\rangle = 0$ and let $x \in \mathscr{X}$ then the positive semidefiniteness[93] of $K(X \cup \{x\})$ tells us that for all $0 \neq t \in \mathbb{R}$

$$\begin{aligned} 0 &\le \left[h_X^T, t\right] K(X\cup\{x\}) \begin{bmatrix} h_X \\ t \end{bmatrix} = \left[h_X^T, t\right] \begin{bmatrix} K(X) & k(X,x) \\ k(X,x)^T & k(x,x) \end{bmatrix} \begin{bmatrix} h_X \\ t \end{bmatrix} \\ &= h_X^T K(X) h_X + 2t h_X^T k(X,x) + t^2 k(x,x) \\ &= \underbrace{\sum_{y,y'\in X} h_y h_{y'} k(y,y')}_{=\langle h,h\rangle=0} + 2t \underbrace{\sum_{x'\in X} h_{x'} k(x',x)}_{=h(x)} + t^2 k(x,x), \end{aligned}$$

[90] There are slightly varying definitions, this one is taken from [61].

[91] No surprising idea as this is what makes the Hilbert space. No inner product, no Hilbert space!

[92] We are talking about "real" bases here, not Schauder bases, so we only consider *finite* linear combinations of the possibly infinitely many elements of $\mathscr{X}$.

[93] Even if already $x \in X$ the *semi*definiteness is not affected as just one row and one column are repeated in the matrix.

hence

$$h(x) + t\,k(x,x) \begin{cases} \geq 0, & t > 0, \\ \leq 0, & t < 0, \end{cases}$$

which implies $h(x) = 0$ by letting $t \to 0$. Hence $\langle h, h\rangle = 0$ implies that $h(x) = 0$, $x \in \mathscr{X}$, and therefore h is the zero function on $\mathscr{X}$.

The reproducing property is immediate from (3.4): just consider

$$\langle h, k(x,\cdot)\rangle = \sum_{x'\in X} h_{x'} k(x', x) = h(x).$$

Having defined the appropriate inner product, we have a norm, hence a metric, and can form the **metric completion** $\mathscr{H}$ of H by adding the limits of all Cauchy sequences and considering equivalence classes with respect to limits[94], so that $\mathscr{H}$ is automatically a complete inner product space, hence a **Hilbert space**.

To prove uniqueness, we assume that $\mathscr{H}'$ is another Hilbert space with $\langle\cdot,\cdot\rangle'$ that also has all the above properties. Then clearly $H \subset \mathscr{H}'$ and for $x, x' \in \mathscr{X}$ we have that

$$\langle k(x,\cdot), k(x',\cdot)\rangle' = k(x,x') = \langle k(x,\cdot), k(x',\cdot)\rangle$$

so that

$$\begin{aligned} \langle h, h'\rangle' &= \sum_{x\in X}\sum_{x'\in X'} h_x h'_{x'} \langle k(x,\cdot), k(x',\cdot)\rangle' = \sum_{x\in X}\sum_{x'\in X'} h_x h'_{x'} \langle k(x,\cdot), k(x',\cdot)\rangle \\ &= \langle h, h'\rangle \end{aligned}$$

and therefore $\mathscr{H}$ and $\mathscr{H}'$ are both completions of H and thus must be the same. Finally, (3.3) is verified by means of the Cauchy–Schwarz inequality

$$|f(x)| = |\langle f, k(x,\cdot)\rangle| \leq \|f\|\,\|k(x,\cdot)\|$$

and the observation that

$$\|k(x,\cdot)\|^2 = \langle k(x,\cdot), k(x,\cdot)\rangle = k(x,x)$$

completes the proof. □

Equation (3.3) is even more important than one might think as it actually is a characterization of the RKHS as the following famous theorem shows.

Theorem 3.7 (Aronzajn/Bergmann)**.** *A metric space $\mathscr{X}$ has a reproducing kernel if and only if for each $x \in \mathscr{X}$ there exists a constant C_x such that*

$$|f(x)| \leq C_x \|f\|. \tag{3.5}$$

Proof: We have just shown that the existence of a reproducing kernel implies (3.5) and that C_x can be chosen as $\sqrt{k(x,x)}$. For the converse, we apply the

[94] This process should be known from basic calculus, it is exactly the process that generates $\mathbb{R}$ from $\mathbb{Q}$, cf. [48].

Riesz representation theorem[95] to the functional $\delta_x : f \mapsto f(x)$ which ensures the existence of a uniquely defined element $h_x \in \mathscr{H}$ such that

$$f(x) = \langle f, h_x \rangle$$

and the function $k(x, \cdot) := h_x$ is the desired reproducing kernel and even unique. □

Definition 3.8. The reproducing kernel Hilbert space with respect to a kernel $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ will be denoted by $\mathscr{H}_k$ and, if necessary, we will use $\langle \cdot, \cdot \rangle_k$ and $\| \cdot \|_k$ for the resulting inner product and the norm.

3.2 Examples of Mercer kernels

The main result from Theorem 3.4 can be summarized as "whenever there is a Mercer kernel, there is a reproducing kernel Hilbert space", but of course we have to make sure that we are not talking about the empty set. Fortunately, there is even a large variety of candidates for Mercer kernels in $\mathbb{R}^d$ or compact subsets thereof.

The simplest mercer kernels are the so called dot product kernels. Let, for $R > 0$, denote

$$B_R := \{x \in \mathbb{R}^d : \|x\| \le R\}$$

the **ball** of **radius** R in $\mathbb{R}^d$ and set $\mathscr{X} = B_R(\mathbb{R}^d)$, so that now $\mathscr{X}$ is a **compact** metric space.

Definition 3.9 (Dot product kernel)**.** For a sequence $a : \mathbb{N}_0 \to \mathbb{R}$ with the properties

$$a_n \ge 0, \qquad \sum_{n=0}^{\infty} a_n R^{2n} < \infty, \tag{3.6}$$

the **dot product kernel** $k_a : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is defined as[96]

$$k_a(x, x') = \sum_{n=0}^{\infty} a_n (x \cdot x')^n, \qquad x, x' \in \mathscr{X}. \tag{3.7}$$

The task of property (3.6) is to ensure that the kernel in (3.7) is well defined as the series converges absolutely[97]:

$$\sum_{n=0}^{\infty} |a_n (x \cdot x')^n| \le \sum_{n=0}^{\infty} a_n \underbrace{(\|x\| \|x'\|)^n}_{\le R^{2n}} < \infty, \qquad x, x' \in \mathscr{X},$$

which works because we picked $\mathscr{X} = B_R(\mathbb{R}^d)$. Note, however, that we can choose $R = \infty$ if a_n decays fast enough, for example $a_n \le \rho^n$ for some $0 < \rho < 1$, and especially for sequences with only finitely many nonzero entries.

[95] See [27, 61] or any other good book on Functional Analysis.

[96] The inner product $x^T x' = \sum x_j x'_j$ on $\mathbb{R}^d$ is often written as $x \cdot x'$ and then called the **dot product**.

[97] Again Cauchy–Schwarz.

Proposition 3.10. *Any dot product kernel is a Mercer kernel.*

Proof: By the multinomial theorem we have

$$(x \cdot x')^n = \left(\sum_{k=1}^{d} x_k x'_k \right)^n = \sum_{|\alpha|=n} \binom{n}{\alpha} x^\alpha x'^\alpha,$$

and therefore, for any $X \subset \mathscr{X}$ and any $c \in \mathbb{R}^X$,

$$\begin{aligned} c^T K(X) c &= \sum_{x,x' \in X} c_x c_{x'} k(x,x') = \sum_{x,x' \in X} c_x c_{x'} \sum_{n=0}^{\infty} a_n (x \cdot x')^n \\ &= \sum_{x,x' \in X} c_x c_{x'} \sum_{n=0}^{\infty} a_n \sum_{|\alpha|=n} \binom{n}{\alpha} x^\alpha x'^\alpha = \sum_{n=0}^{\infty} a_n \sum_{|\alpha|=n} \binom{n}{\alpha} \sum_{x,x' \in X} c_x c_{x'} x^\alpha x'^\alpha \\ &= \sum_{n=0}^{\infty} a_n \sum_{|\alpha|=n} \binom{n}{\alpha} \underbrace{\left(\sum_{x \in X} c_x x^\alpha \right)^2}_{\geq 0} \geq 0, \end{aligned}$$

which shows that the kernel is positive semidefinite. The other properties are easily verified. □

Exercise 3.2 Prove the **multinomial theorem**

$$\left(\sum_{j=1}^{d} x_j \right)^n = \sum_{|\alpha|=n} \binom{n}{\alpha} x^\alpha, \qquad x^\alpha := x_1^{\alpha_1} \cdots x_d^{\alpha_d}, \quad \binom{n}{\alpha} = \frac{n!}{\alpha_1! \cdots \alpha_d!}.$$

♢

If the sequence a from Definition 3.9 is finite, i.e., $a_n = 0$ for n sufficiently large, then we can choose R arbitrarily and therefore the kernel works globally.

Corollary 3.11. *Any finite dot product kernel*

$$k_a(x,x') = a_0 + a_1 (x \cdot x') + \cdots + a_n (x \cdot x')^n$$

is a Mercer kernel for any $\mathscr{X} \subseteq \mathbb{R}^d$.

The simplest dot product kernel even works on arbitrary Hilbert spaces, namely $k_{a_0,a_1}(x,x') = a_0 + a_1 \langle x, x' \rangle$. Here we have

$$\begin{aligned} &\sum_{x,x' \in X} c_x c_{x'} k(x,x') = \sum_{x,x' \in X} c_x c_{x'} (a_0 + a_1 \langle x, x' \rangle) \\ &= a_0 \sum_{x,x' \in X} c_x c_{x'} + a_1 \sum_{x,x' \in X} c_x c_{x'} \langle x, x' \rangle = a_0 \left(\sum_{x \in X} c_x \right)^2 + a_1 \left\| \sum_{x \in X} c_x x \right\|^2 \geq 0. \end{aligned}$$

This allows for a very peculiar *kernel trick*: For an arbitrary **metric space** $\mathscr{X}$ let $\psi : \mathscr{X} \to \mathscr{H}$ be a continuous mapping to an *intermediate Hilbert space* $\mathscr{H}$ on which the linear kernel k_{a_0,a_1} is well–defined. Then

$$k_{a_0,a_1}(x,x') := a_0 + a_1 \langle \psi(x), \psi(x') \rangle, \qquad a_0, a_1 \geq 0,$$

is a family of nontrivial Mercer kernels on $\mathscr{X}$ since we do not need any multinomial theorem here. As a consequence, we can record that we are definitely not talking about an empty set.

Corollary 3.12. *For any metric space $\mathscr{X}$ there exists a reproducing kernel Hilbert space.*

There is another famous dot product kernel, namely the one obtained if we choose $a_n = \frac{1}{n!}$, i.e.

$$k(x, x') = \sum_{n=0}^{\infty} \frac{1}{n!}(x \cdot x')^n = e^{x \cdot x'}, \qquad x, x' \in \mathbb{R}^d,$$

giving the exponential **ridge function**[98] and proving that the matrices

$$E(X) := \left[e^{x \cdot x'} \begin{array}{c} x \in X \\ x' \in X \end{array} \right], \qquad X \subset \mathbb{R}^d, \, \#X < \infty,$$

are positive semidefinite. Since this implies that $\det E(X) \geq 0$ and since the property carries over to any subset $X' \subseteq X$ of X, it also follows that

$$\det E(X') \geq 0, \qquad X' \subseteq X, \tag{3.8}$$

which means that any such matrix $E(X)$ is **totally nonnegative**, quite a special property among matrices, see [24, 46]. Another proof of this fact can be found in [36].

This approach can easily be extended to arbitrary analytic functions $f : \mathbb{R} \to \mathbb{R}$ with nonnegative Taylor coefficients and the resulting $k(x, x') = f(x \cdot x')$. We will encounter functions of this form later.
The second class of kernels will be **shift invariant** kernels or **stationary** kernels. They will be defined on all of $\mathbb{R}^d$ but need other properties of the function.

Definition 3.13. A **stationary** kernel is a kernel defined as

$$k(x, x') := k_g(x, x') := g(x - x'), \qquad x, x' \in \mathbb{R}^d, \tag{3.9}$$

where $g \in C(\mathbb{R}^d)$ is a continuous **even function**, i.e., $g(-x) = g(x)$, $x \in \mathbb{R}^d$.

To understand stationary kernels, we need another definition.

Definition 3.14 (Fourier transform)**.** The **Fourier transform** of a function $f \in L_1(\mathbb{R}^d)$, i.e., a function such that

$$\int_{\mathbb{R}^d} |f(x)| dx < \infty,$$

is defined as

$$\hat{f}(\xi) := f^{\wedge}(\xi) := \int_{\mathbb{R}^d} f(x) e^{-i\xi \cdot x} \, dx, \qquad \xi \in \mathbb{R}^d, \tag{3.10}$$

and the **inverse Fourier transform** as

$$f^{\vee}(x) := \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} f(\xi) e^{i\xi \cdot x} \, d\xi, \qquad x \in \mathbb{R}^d. \tag{3.11}$$

[98] Once again: most of the objects that we are considering have been discovered or developed in various fields of mathematics.

Remark 3.15 (Fourier transform)**.** We will not consider details of the Fourier transform[99] here, but some remarks are nevertheless necessary.

1. The definitions (3.10) and (3.11) of the Fourier transform and its inverse are one choice of many. In particular, the constant $(2\pi)^{-d}$ can quite arbitrarily distributed over the two transforms and is handled differently in the literature.

2. The name "inverse Fourier transform" is justified since $(\widehat{f})^{\vee} = f$ for any function f such that $f, \widehat{f} \in L_1(\mathbb{R}^d)$.

3. The Fourier transform and its inverse can be *extended* to $L_2(\mathbb{R}^d)$ and the extension can be applied in a rather naive way.

Proposition 3.16 (Stationary kernels)**.** *If $g \in L_2(\mathbb{R}^d) \cap C(\mathbb{R}^d)$ is an even function[100] with nonnegative Fourier transform, i.e. $\widehat{g} \geq 0$, then the stationary kernel k_g is a Mercer kernel.*

Proof: Since g is real and even, it follows that

$$\overline{\widehat{g}(\xi)} = \int_{\mathbb{R}^d} \overline{g(x)\, e^{-i\xi\cdot x}}\, dx = \int_{\mathbb{R}^d} g(x)\, e^{i\xi\cdot x}\, dx = \int_{\mathbb{R}^d} g(-x)\, e^{-i\xi\cdot x}\, dx = \widehat{g}(\xi)$$

hence $\widehat{g}$ is real and therefore $\widehat{g} \geq 0$ makes sense[101]. Now consider, again for a finite $X \subset \mathbb{R}^d$ and $c \in \mathbb{R}^X$,

$$\begin{aligned}
&\sum_{x,x'\in X} c_x c_{x'} k(x,x') = \sum_{x,x'\in X} c_x c_{x'} g(x-x') \\
&= \sum_{x,x'\in X} c_x c_{x'} (\widehat{g})^{\vee}(x-x') = \sum_{x,x'\in X} c_x c_{x'} \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \widehat{g}(\xi) \underbrace{e^{i\xi\cdot(x-x')}}_{=e^{i\xi\cdot x}\, e^{-i\xi\cdot x'}}\, d\xi \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \widehat{g}(\xi) \left(\sum_{x\in X} c_x e^{i\xi\cdot x}\right) \underbrace{\left(\sum_{x'\in X} c_{x'} e^{-i\xi\cdot x'}\right)}_{=\overline{\sum_x \cdots}} d\xi = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \underbrace{\widehat{g}(\xi)}_{\geq 0} \underbrace{\left|\sum_{x\in X} c_x e^{i\xi\cdot x}\right|^2}_{\geq 0} d\xi \\
&\geq 0.
\end{aligned}$$

The other properties of the Mercer kernel are easy to check. □

The appealing point about this approach is that it immediately provides a really large class of kernel functions, among them some really well–known functions.

The concept needed here is that of a **convolution** which is defined, for $f, g \in L_2(\mathbb{R}^d)$ as

$$(f * g)(x) = \int_{\mathbb{R}^d} f(x-t) g(t)\, dt. \tag{3.12}$$

[99] See, for example, [25, 31, 45] for some information.

[100] We assume continuity to guarantee that the **point evaluation** at $x - x'$ are well–defined.

[101] Of course, this is a standard argument and fact from Fourier Analysis.

Definition 3.17 (B–Splines)**.** The (centered) **cardinal tensor product B–Spline** N_m of **degree** m is defined as the $m+1$–fold of the characteristic function

$$\chi := \chi_{[-\frac{1}{2},\frac{1}{2}]^d}(x) = \begin{cases} 1, & -\frac{1}{2} \le x_j \le \frac{1}{2},\ j = 1,\dots,d, \\ 0, & \text{otherwise,} \end{cases} \tag{3.13}$$

of the centered **unit cube**[102] as

$$N_m(x) = (\chi * N_{m-1})(x) = \underbrace{(\chi * \cdots * \chi)}_{m+1}(x). \tag{3.14}$$

Based on the two observations

$$(f * g)^\wedge(\xi) = \widehat{f}(\xi)\,\widehat{g}(\xi) \qquad \text{and} \qquad \widehat{\chi}(\xi) = \prod_{j=1}^{d} \frac{\sin \xi_j/2}{\xi_j/2}, \qquad \xi \in \mathbb{R}^d, \tag{3.15}$$

we can explicitly give a family of stationary Mercer kernels.

Exercise 3.3 Prove (3.15). ◊

Proposition 3.18. *The kernels* $k(x,x') := N_m(x-x')$ *are stationary Mercer kernels for any odd* $m \in 2\mathbb{N}_0 + 1$.

Proof: By (3.15),

$$\widehat{N}_m(\xi) = \prod_{j=1}^{d} \left(\frac{\sin \xi_j/2}{\xi_j/2} \right)^{m+1}$$

which is ≥ 0 if m is odd. Since $N_0 = \chi$ is an even function and since for even functions f, g one has

$$(f*g)(-x) = \int_{\mathbb{R}^d} \underbrace{f(-x-t)}_{=f(x+t)}\, \underbrace{g(t)}_{=g(-t)}\, dt = \int_{\mathbb{R}^d} f(x-t)g(t)\,dt = (f*g)(x), \qquad x \in \mathbb{R}^d,$$

the splines are even functions even for any $m \in \mathbb{N}_0$. □

The last and almost most popular example of Mercer kernels is a restriction of the stationary and chooses $g = g_0(\|\cdot\|^2)$. Since now $g(x)$ only depends on $\|x\|$, it is constant on spheres which is the reason why such a functions is called a **radial function**.

Definition 3.19 (Radial kernel)**.** A **radial kernel** is a kernel of the form

$$k(x,x') = g_0\left(\|x-x'\|_2^2\right), \qquad x, x' \in \mathbb{R}^d, \tag{3.16}$$

with $g_0 : \mathbb{R}_+ \to \mathbb{R}$.

The proof of the fact that these are Mercer kernels is based on yet another classical and very interesting concept.

[102] "Unit" because it has volume 1.

Definition 3.20 (Complete monotonicity)**.** A function $f : \mathbb{R}_+ \to \mathbb{R}$ is called **completely monotonic**[103] if $f \in C(\mathbb{R}_+) \cap C^\infty(\mathbb{R}_+ \setminus \{0\})$ and

$$(-1)^j f^{(j)}(x) \geq 0, \qquad j \in \mathbb{N}_0, \quad x \in \mathbb{R}_+. \tag{3.17}$$

Hence, a completely monotonic function is nonnegative ($f(x) \geq 0$), monotonically decreasing ($f'(x) \leq 0$), convex ($f''(x) \geq 0$), and so on.

The location of this definition suggests that completely monotonic functions are the property class of kernels and indeed there is the following result due to Micchelli [35] in the context of **radial basis function**[104] interpolation.

Proposition 3.21. *If $g_0 : \mathbb{R}_+ \to \mathbb{R}$ is completely monotonic, then the radial kernel k from* (3.16) *is a **Mercer kernel** for $\mathscr{X} = \mathbb{R}^d$.*

The proof is based on yet another classical theorem that we only state here without proof.

Theorem 3.22 (Bochner's theorem)**.** *A function $f : \mathbb{R}_+ \to \mathbb{R}$ is completely monotonic if and only if there exist a finite nonnegative **Borel measure**[105] μ such that*[106]

$$f(x) = \int_0^\infty e^{-xt}\, d\mu(t), \qquad x \in \mathbb{R}_+. \tag{3.18}$$

Proof of Proposition 3.21: The proof is an application of Theorem 3.22. By (3.18), the kernel takes the form $k(x, x') = g(x - x')$ with

$$g(x) = \int_0^\infty e^{-t\|x\|^2}\, d\mu(t), \qquad x \in \mathbb{R}^d. \tag{3.19}$$

Since

$$\left(e^{-t\|\cdot\|_2^2}\right)^\wedge(\xi) = \left(\frac{\pi}{t}\right)^{d/2} e^{-\|\xi\|_2^2/(4t)}, \qquad t > 0, \quad \xi \in \mathbb{R}^d, \tag{3.20}$$

it follows like in the proof of Proposition 3.16 that

$$\begin{aligned}
\sum_{x,x' \in X} c_x c_{x'} k(x, x') &= \sum_{x,x' \in X} c_x c_{x'} \int_0^\infty e^{-t\|x - x'\|^2}\, d\mu(t) \\
&= \int_0^\infty \sum_{x,x' \in X} c_x c_{x'} \left(\frac{\pi}{t}\right)^{d/2} \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{-\|\xi\|_2^2/(4t)} e^{i\xi(x - x')}\, d\xi\, d\mu(t) \\
&= \frac{1}{(2\pi)^d} \int_0^\infty \left(\frac{\pi}{t}\right)^{d/2} \int_{\mathbb{R}^d} e^{-\|\xi\|_2^2/(4t)} \left| \sum_{x \in X} c_x e^{i\xi \cdot x} \right|^2 d\xi\, d\mu(t) \geq 0
\end{aligned}$$

which again verifies the positive semidefiniteness of the kernel. □

Exercise 3.4 Prove (3.20). What happens for $t = 0$? ◇

[103] A **completely monotone**. i.e. boring, function is for example $f = 0$.

[104] Usually abbreviated as **RBF**

[105] Essentially this is a **nonnegative measure**, like probability densities without proper normalization.

[106] This, in turn, is essentially the **Laplace transform** of the measure. The Laplace transform is a concept of independent interest, especially in Signal Processing.

Example 3.23 (Radial kernels)**.** Famous examples of a **radial Mercer kernel** on $\mathbb{R}^d$ are the following functions:

1. **Gaussian** $k(x, x') = e^{-\|x-x'\|_2^2/c^2}$, $c \in \mathbb{R}$.
2. **inverse multiquadrics** $k(x, x') = \left(c^2 + \|x - x'\|_2^2\right)^{-\alpha}$, $c \in \mathbb{R}$, $\alpha > 0$.

3.3 Mercer's theorem

Finally, a very simple construction of Mercer kernels that we should almost have given in the beginning. Let Φ be a set of continuous functions from $\mathscr{X} \to \mathbb{R}$ and let $a \in \mathbb{R}^\Phi$ be a nonnegative sequence, i.e., $a_\varphi \geq 0$, $\varphi \in \Phi$.

Proposition 3.24 (Yet another Mercer kernel)**.** *If the function*

$$k(x, x') = \sum_{\varphi \in \Phi} a_\varphi \varphi(x)\varphi(x'), \qquad x, x' \in \mathscr{X}, \tag{3.21}$$

is well–defined[107]*, it is a Mercer kernel.*

Proof: For finite $X \subset \mathscr{X}$ and $c \in \mathbb{R}^X$ we get

$$\begin{aligned}\sum_{x,x' \in X} c_x c_{x'} k(x, x') &= \sum_{x,x' \in X} c_x c_{x'} \sum_{\varphi \in \Phi} a_\varphi \varphi(x)\varphi(x') \\ &= \sum_{\varphi \in \Phi} a_\varphi \sum_{x,x' \in X} c_x \varphi(x)\, c_{x'} \varphi(x') = \sum_{\varphi \in \Phi} \underbrace{a_\varphi}_{\geq 0} \underbrace{\left(\sum_{x \in X} c_x \varphi(x)\right)^2}_{\geq 0} \geq 0,\end{aligned}$$

which is almost too simple to be true. □

Remark 3.25. Proposition 3.24 defines Mercer kernels, hence reproducing kernel Hilbert spaces for *any* **metric space** $\mathscr{X}$. All we need is a continuous, real valued[108] function on $\mathscr{X}$. Equation (3.21) defines the kernel by **separation of variables** or as a **tensor product** function on $\mathscr{X} \times \mathscr{X}$. We will see later that this seemingly simple approach is surprisingly general.

Nevertheless, the above *concrete* examples of admissible Mercer kernels are valuable because they include classical functions with well understood properties.

The kernels of the type (3.21) are the reason for the name "Mercer kernels". If we define a Hilbert space $L_2(\mathscr{X}, \mu)$ with the inner product

$$\langle f, g \rangle_{\mathscr{X}} := \int_{\mathscr{X}} f(x)g(x)\, d\mu(x)$$

[107] That means, if the above series converges in the case of infinite Φ which may make it necessary to restrict the size of x, x' like in the case of the **dot product kernel**.

[108] Complex valued if one wants a complex RKHS.

based on a positive **Borel measure**[109] μ, the kernel k defines the **integral transform**

$$Kf := \int_{\mathscr{X}} k(x,\cdot) f(x)\, d\mu(x).$$

Since the symmetry of k implies

$$\langle f, Kg\rangle_{\mathscr{X}} = \int_{\mathscr{X}}\int_{\mathscr{X}} f(x)k(x,x')g(x)\, d\mu(x)\, d\mu(x') = \langle Kf, g\rangle_{\mathscr{X}},$$

the operator K is **self adjoint**. If moreover k is continuous and $\mathscr{X}$ is compact and bounded[110], then

$$\begin{aligned} |Kf(x) - Kf(x')| &= \left|\int_{\mathscr{X}} (k(x,t) - k(x',t))\, f(t)\, d\mu(t)\right| \\ &\le \int_{\mathscr{X}} |k(x,t) - k(x',t)|\, |f(t)|\, d\mu(t) \le \|k(x,\dot{}) - k(x',\cdot)\|_{L_2(\mathscr{X},\mu)} \|f\|_{L_2(\mathscr{X},\mu)} \\ &\le \sqrt{\mu(\mathscr{X})}\max_{t\in\mathscr{X}} |k(x,t) - k(x',t)|\ \|f\|_{L_2(\mathscr{X},\mu)}, \end{aligned}$$

so that Kf is a continuous function. By the **spectral theorem**, cf.[9] there exists an orthonormal basis Ψ of $L_2(\mathscr{X},\mu)$ such that

$$K\psi = \lambda_\psi \psi, \quad \psi\in\Psi, \qquad \text{and} \qquad \langle \psi, \psi'\rangle_{\mathscr{X}} = \delta_{\psi,\psi'}. \tag{3.22}$$

These eigenfunctions can then be used to expand the kernel itself, showing that in this sense under certain circumstances *every* Mercer kernel is of the simple form (3.21). This is Mercer's theorem.

Theorem 3.26 (Mercer's theorem)**.** *If μ is a nondegenerate Borel measure on $\mathscr{X}$, $k : \mathscr{X}\times\mathscr{X} \to \mathbb{R}$ a Mercer kernel and $\Psi \subset L_2(\mathscr{X},\mu)$ a basis of nonnegative eigenfunctions of K, then*

$$k(x,x') = \sum_{\psi\in\Psi} \lambda_\psi\, \psi(x)\psi(x'), \tag{3.23}$$

and the convergence of the series is absolute and uniform.

A fully detailed proof of Mercers theorem would need too many requirements from Functional Analysis for which reason we only refer to [9].

3.4 Learning with kernels

Now that we have the RKHS mechanism available, the learning problems with **training data** $X \subset \mathscr{X}$ and **labels** $y \in \mathbb{R}^X$ can be written by means of a model space Φ as the penalized or smoothened quadratic functional

$$\min_a \sum_{x\in X} \left(y_x - \langle k(x,\cdot), \phi(a)\rangle\right)^2 + \lambda a^T a, \qquad \lambda \ge 0. \tag{3.24}$$

[109] We need positivity of the measure for the definiteness of the inner product.

[110] Metrically and in measure, i.e. $d(\mathscr{X}) := \sup\{x - x' : x, x' \in \mathscr{X}\} < \infty$ and $\mu(\mathscr{X}) < \infty$

The functional

$$\sum_{x\in X} (y_x - \langle k(x,\cdot), \phi(a)\rangle)^2 \tag{3.25}$$

is often called the **empirical error**, cf. [9], or **square loss**. The target function can be rewritten in the usual way as

$$\begin{aligned}
&\sum_{x\in X} (y_x - \langle k(x,\cdot), \phi(a)\rangle)^2 + \lambda a^T a \\
&= \sum_{x\in X} y_x^2 - 2y_x \underbrace{[\langle k(x,\cdot), \varphi\rangle : \varphi \in \Phi]}_{=:\Phi(x)} a + (\Phi(x)a)^T (\Phi(x)a) + \lambda a^T a \\
&= y^T y - 2y^T \Phi(X)a + a^T \left(\Phi(x)^T \Phi(X) + \lambda I\right) a
\end{aligned}$$

which once more leads to

$$\left(\Phi(x)^T \Phi(X) + \lambda I\right) a = \Phi(X)^T y, \tag{3.26}$$

as before. So what's new with this approach? It still appears unnecessary[111] to define all these abstract spaces, but we haven't discussed the **model space** yet, where we could also take *kernels* as generating functions for the model space, i.e.,

$$\varphi = \varphi_\xi = k(\xi,\cdot), \qquad \xi \in \Xi \subseteq \mathscr{X},$$

where Ξ can depend on the test set X or not. We then have that the **Gramian**

$$G_\Phi = \left[\langle k(\xi,\cdot), k(\xi',\cdot)\rangle : \begin{array}{c}\xi\in\Xi\\ \xi'\in\Xi\end{array}\right] = \left[k(\xi,\xi') : \begin{array}{c}\xi\in\Xi\\ \xi'\in\Xi\end{array}\right] = K(\Xi) \tag{3.27}$$

is the kernel **collocation matrix** with respect to Ξ while

$$\Phi(X) = \left[k(x,\xi) : \begin{array}{c}x\in X\\ \xi\in\Xi\end{array}\right] =: K(X,\Xi)$$

is a "mixed collocation" between X and Ξ, and (3.26) becomes

$$\left(K(X,\Xi)^T K(X,\Xi) + \lambda I\right) a = K(X,\Xi)^T y. \tag{3.28}$$

The prediction function then takes the form[112]

$$\phi(a) = \sum_{\xi\in\Xi} a_\xi k(\xi,\cdot) = K(\cdot,\Xi)^T \left(K(X,\Xi)^T K(X,\Xi) + \lambda I\right)^{-1} K(X,\Xi)^T y \tag{3.29}$$

and only depends on the behavior of k on $(X\cup\Xi)\times(X\cup\Xi)$.

Remark 3.27. The most frequently used case, motivated by its use in the theory of **radial basis functions** is to choose $X = \Xi$, which makes all the matrices square matrices and simplifies (3.29) to

$$\phi_y = K(\cdot,X).(K(X) + \lambda I)^{-1} K(X)^T y \tag{3.30}$$

[111] Even if it may be mathematically pleasant.

[112] This is *not* the way to compute this function, of course, never *compute* inverses of matrices!

In the special case $\Xi = X$, the **orthogonal projection** Pf of $f \in \mathscr{H}$ to $\operatorname{span}\Phi = \operatorname{span}\{k(x,\cdot) : x \in X\}$, defined by

$$0 = \langle f - Pf, \Phi\rangle \qquad \Leftrightarrow \qquad 0 = \big\langle f - Pf, k(x,\cdot)\big\rangle, \quad x \in X,$$

satisfies $f(x) = Pf(x)$, $x \in X$, and the **empirical error** (3.25) takes the same value for f and Pf. This already proves the following observation from [9].

Proposition 3.28. *If $\mathscr{H}' \subset \mathscr{H}$ satisfies $P(\mathscr{H}') \subseteq \mathscr{H}'$ and the empirical error can be minimized on $\mathscr{H}'$, then the minimizer can be chosen from $P(\mathscr{H}')$.*

We could also do a "direct" approximation in the RKHS space by modeling a label function

$$y = \sum_{\xi\in\Xi} c_\xi k(\xi,\cdot), \qquad y : \mathscr{X} \to \mathbb{R},$$

and rewrite the side conditions

$$y_x = y(x) = \sum_{\xi\in\Xi} c_\xi k(\xi, x), \qquad x \in X,$$

as $y = K(X,\Xi)c$. As long as the kernel is a **Mercer kernel** and thus the RKHS is well–defined, we then minimize

$$\begin{aligned}\big\|y - \phi(a)\big\|^2 + \lambda a^T a &= \left\|\sum_{\xi\in\Xi} (c_\xi - a_\xi)\, k(\xi,\cdot)\right\|^2 + \lambda a^T a \\ &= \sum_{\xi,\xi'} (a_\xi - c_\xi)(a_{\xi'} - c_{\xi'}) \underbrace{\big\langle k(\xi,\cdot), k(\xi',\cdot)\big\rangle}_{=k(\xi,\xi')} = (a-c)^T K(\Xi)(a-c) + \lambda a^T a\end{aligned}$$

subject to the **equality constraint** $K(X,\Xi)c = y$. Since the constraint is independent of a, we have to solve

$$0 = 2K(\Xi)(a-c) + 2\lambda a = 2(K(\Xi) + \lambda I)a - 2K(\Xi)c$$

subject to $y = K(X,\Xi)c$ which leads to the **augmented system**

$$\begin{bmatrix} K(\Xi) + \lambda I & -K(\Xi) \\ 0 & K(X,\Xi) \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{3.31}$$

which has a unique solution provided that $K(X,\Xi)$ is of maximal rank, otherwise it is to be solved in the least square sense again[113]

3.5 Make your own kernel

Once a Mercer kernel has been determined, there is a lot of methods, some listed in [6], to build new kernels. The trick is usually to use properties of positive semidefinite matrices. Let us list and prove at least one example.

[113] `Matlab` does this almost automatically.

Proposition 3.29. *If* k, k' *are Mercer kernels on* $\mathcal{X}$ *then*

$$\alpha k + \beta k', \quad \alpha, \beta \geq 0, \qquad \textit{and} \qquad k\,k' \tag{3.32}$$

are Mercer kernels as well.

Proof: The first result is quite obvious:

$$\begin{aligned}
&\sum_{x,x' \in X} c_x c_{x'} \left(\alpha\, k(x,x') + \beta\, k'(x,x')\right) \\
&\quad = \underbrace{\alpha \sum_{x,x' \in X} c_x c_{x'}\, k(x,x')}_{\geq 0} + \underbrace{\beta \sum_{x,x' \in X} c_x c_{x'}\, k'(x,x')}_{\geq 0} \geq 0,
\end{aligned}$$

while the second one follows from the property that any **symmetric matrix** has an **orthonormal basis** of eigenvectors, i.e.,

$$A = \sum_{j=1}^{n} v_j v_j^T, \qquad A = A^T \in \mathbb{R}^{n \times n}, \quad v_j \in \mathbb{R}^n,\ v_j^T v_k = \delta_{jk} \lambda_j \geq 0.$$

Then the **Hadamard product**

$$A \odot A' = \left[a_{jk} a'_{jk} : j, k = 1, \ldots, n\right]$$

of two symmetric matrices takes, for $r, s \in \{1, \ldots, n\}$, the form

$$\begin{aligned}
(A \odot A')_{rs} &= \sum_{j=1}^{n} \underbrace{\left(v_j v_j^T\right)}_{= v_{jr} v_{js}} \sum_{k=1}^{n} \underbrace{\left(w_k w_k^T\right)}_{= w_{kr} w_{ks}} = \sum_{j,k=1}^{n} v_{jr} w_{kr} v_{js} w_{ks} \\
&= \sum_{j,k=1}^{n} (v_j \odot w_k)_r\, (v_j \odot w_k)_s,
\end{aligned}$$

hence

$$A \odot A' = \sum_{j,k=1}^{n} (v_j \odot w_k)(v_j \odot w_k)^T$$

is a sum of positive definite rank 1 matrices and therefore positive semidefinite. Since $kk'(X) = K(X) \odot K'(X)$, this completes the proof of the second claim. □

Exercise 3.5 Prove that $A \odot A'$ is positive definite if A and A' are positive definite. ♢

Some more simple properties which are proved in a straightforward way are listed in the following proposition.

Proposition 3.30 (Kernel building)**.** *If* k, k' *are Mercer kernels, then the following kernels are Mercer kernels as well:*

1. $q(k(x,x'))$ *where* $q(t) = q_0 + \cdots + q_n t^n$, $q_j \geq 0$.

2. $e^{k(x,x')}$.

3. $\phi(x)^T A \phi(x')$ *with* $\phi : \mathscr{X} \to \mathbb{R}^d$ *and* $A \in \mathbb{R}^{d\times d}$ *positive semidefinite.*

4. *any decomposition of the form*

$$k(x, x') = \sum_{j=1}^{n} k_j(x_j, x'_j), \qquad x = (x_1, \ldots, x_n)$$

and k_j *a Mercer kernel on the "part" of the variable* x*.*

Exercise 3.6 Prove Proposition 3.30. ◊

3.6 Nodal functions and packing

This section lists some results on kernels that can be used to derive quantitative results of the following sort:

1. how many training values does one need to reliably learn a certain function?

2. can learning be guaranteed if the function is sampled on a denser and denser set of points to provide training data?

3. are there quantitative statements about the *rate* of decay of the approximation error?

Many of these questions are considered in [9], but require a somewhat sophisticated interplay between Statistics and Approximation Theory that exceeds the scope of this lecture and would be a lecture of its own. However, we will aim for Theorem 3.39 in this subsection that can and will be interpreted as a first estimate how well kernels can distinguish between different labelings.

In many cases, kernels should be able to interpolate on the training set $X \subset \mathscr{X}$. This leads to the following concept in the the context of an RKHS $\mathscr{H}$.

Definition 3.31 (Nodal functions)**.** A vector $\Psi \in \mathscr{H}^X$, $\Psi = (\psi_x : x \in X)$ is called a set of **nodal functions** for a finite set X if

$$\psi_x \in \operatorname{span}\{k(x', \cdot) : x' \in X\} \tag{3.33}$$

and

$$\delta_{x,x'} = \psi_x(x') = \langle \psi_x, k(x', \cdot)\rangle. \tag{3.34}$$

In other words, nodal functions are the ones that vanish at all points except one where they are one; in **polynomial interpolation**, these functions are often called **Lagrange functions** since the interpolant to $f : \mathscr{X} \to \mathbb{R}$ can be simply written as

$$L_\Psi f = \sum_{x\in X} f(x)\, \psi_x \qquad \Rightarrow \qquad f(X) = L_\Psi f(X).$$

The existence of nodal functions can be characterized as follows.

Theorem 3.32 (Existence of nodal functions)**.** *For a Mercer kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *and* $X \subseteq \mathcal{X}$, $\#X < \infty$, *the following statements are equivalent:*

1. *There exists a vector* Ψ *of nodal functions.*
2. *The functions* $k(x, \cdot)$, $x \in X$, *are* ***linearly independent****.*
3. *The collocation matrix* $K(X)$ *is invertible.*
4. *There exists* $F \in \mathcal{H}$ *such that* $f_x(x') = \delta_{x,x'}$, $x, x' \in X$.

In that case the nodal functions are given as

$$\psi_x = \left(K(X)^{-1}K(X, \cdot)\right)_x = \sum_{x' \in X} \left(K(X)^{-1}\right)_{x,x'} k(x', \cdot) \tag{3.35}$$

Proof: "1) $\Rightarrow$ 2)": The nodal functions are linearly independent since

$$0 = \psi := \sum_{x \in X} c_x \psi_x \qquad \Rightarrow \qquad 0 = \psi(x') = \sum_{x \in X} c_x \underbrace{\psi_x(x')}_{=\delta_{x,x'}} = c_{x'}, \quad x' \in X,$$

and $\Psi \subset \operatorname{span} k(X, \cdot)$ gives

$$\#X = \dim \operatorname{span} \Psi \leq \dim \operatorname{span} k(X, \cdot) \leq \#X$$

yields $\dim \operatorname{span} k(X, \cdot) = \#X$, hence the functions are linearly independent.

"2) $\Rightarrow$ 3)": If $K(X)$ were not invertible, there exists $c \in \mathbb{R}^X \setminus \{0\}$ such that $K(X)c = 0$ and therefore

$$\begin{aligned} \left\| \sum_{x \in X} c_x k(x, \cdot) \right\|^2 &= \left\langle \sum_{x \in X} c_x k(x, \cdot), \sum_{x \in X} c_x k(x, \cdot) \right\rangle = \sum_{x,x' \in X} c_x c_{x'} \langle k(x, \cdot), k(x', \cdot) \rangle \\ &= \sum_{x,x' \in X} c_x c_{x'} k(x, x') = c^T \underbrace{K(X)c}_{=0} = 0, \end{aligned}$$

hence $\sum_x c_x k(x, \cdot) = 0$ which contradicts the linear independence of $k(X, \cdot)$.

"3) $\Rightarrow$ 4)": Just define

$$f_x = \sum_{y \in X} \left(K(X)^{-1}\right)_{x,y} k(y, \cdot),$$

as in (3.35), then

$$f_x(x') = \sum_{y \in X} \left(K(X)^{-1}\right)_{x,y} k(y, x') = \left(K(X)^{-1}K(X)\right)_{x,x'} = \delta_{x,x'}.$$

"4) $\Rightarrow$ 1)": Let P denote the orthogonal projection from $\mathcal{H}$ to $\operatorname{span} k(X, \cdot)$, then the same argument as the one for Proposition 3.28 tells us that $Pf(x) = f(x)$, $x \in X$, hence $\psi_x = Pf_x$ has the desired properties. □

We want to use nodal functions to get an idea how large the training set must be if we want to be able to distinguish features. This will be related to the **operator norm** $\|K(X)^{-1}\|_2$. The respective formal object is as follows.

Definition 3.33. Let S be a compact metric space and $\eta > 0$. The **packing number** $p(S,\eta)$ is defined as

$$p(S,\eta) = \max \#\{X \subset S : d(x,x') > \eta,\ x,x' \in X\}. \tag{3.36}$$

The **covering number**[114] $c(S,\eta)$ is defined as

$$\min \# \left\{ x \subset S : S \subseteq \bigcup_{x\in X} B_\eta(x) \right\}, \tag{3.37}$$

where

$$B_\eta(x) := \{x' \in S : d(x,x') \le \eta\}$$

denotes the closed ball of radius η around x.

Remark 3.34. Intuitively, packing and covering numbers for $[0,1]^d$ grow like η^{-d} when η becomes smaller and smaller. Of course, the precise numbers are a more difficult thing.

Clearly, $p(S,\eta)$ and $c(S,\eta)$ are monotonically decreasing with respect to η. Another interesting relationship is the following that shows that covering and packing are essentially the same.

Lemma 3.35. $p(S,2\eta) \le c(S,\eta) \le p(S,\eta)$.

Proof: Let $X \subset S$ be such that $d(x,x') > 2\eta$ for $x,x' \in X$. Then $B_\eta(x) \cap B_\eta(x') = \emptyset$, hence $c(S,\eta) \ge \#X$; this holds in particular for some X with $p(S,\eta) = \#X$ and therefore $p(S,2\eta) \le c(S,\eta)$

For the other inequality, let $X \subset S$ be such that $d(x,x') > \eta$ and $\#X = p(S,\eta)$. Then

$$S \subseteq \bigcup_{x\in X} B_\eta(x) \qquad \Rightarrow \qquad c(S,\eta) \le p(S,\eta)$$

as otherwise there would exist $x' \in S$ such that $d(x,x') > \eta$, $x \in X$, and that would yield the contradiction $p(S,\eta) \ge \#X + 1$. □

The next theorem needs a slightly strange space which we are going to define first.

Definition 3.36. For an RKHS $\mathscr{H}$ defined on a compact metric space $\mathscr{X}$, we define the metric space

$$B_R(\mathscr{H})_\infty := \{f \in \mathscr{H} : \|f\|_{\mathscr{H}} \le R\}, \qquad d(f,f') = \|f - f'\|_\infty = \max_{x\in\mathscr{X}} |f(x) - f'(x)| \tag{3.38}$$

obtained by the R–ball in $\mathscr{H}$, equipped with the $\|\cdot\|_\infty$–norm used for continuous functions.

[114] This is why we need compactness as then this number is finite.

Remark 3.37. If the kernel k for the RKHS is a Mercer kernel and in particular continuous, then (3.3) yields that

$$\|f - f'\|_\infty = \max_{x \in \mathscr{X}} \sqrt{k(x,x)}\, \|f - f'\|_{\mathscr{H}}$$

which is **uniformly bounded** as long as k is continuous and $\mathscr{X}$ is compact[115].

Definition 3.38. The 2 **operator norm** of a matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$\|A\|_2 := \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2 = 1} \|Ax\|_2.$$

Theorem 3.39 (Packing with kernels)**.** *If $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is a **Mercer kernel** for the compact metric space $\mathscr{X}$ and $X \subset \mathscr{X}$ is a finite set such that $K(X)$ is invertible, then*

$$p\left(B_R(\mathscr{H})_\infty, \eta\right) \geq 2^{\#X} - 1 \qquad \textit{whenever} \qquad \left\|K(X)^{-1}\right\|_2 < \frac{1}{\#X}\left(\frac{R}{\eta}\right)^2 \tag{3.39}$$

and $\eta > 0$.

The condition on η can be rephrased as

$$\eta \leq R \sqrt{\frac{1}{\#X \left\|K(X)^{-1}\right\|_2}},$$

hence, the larger $\left\|K(X)^{-1}\right\|_2$, i.e., the more difficult it is to solve the optimization problem, the smaller we must choose η to have the optimal separation of the functions in the ball

$$B_R(\mathscr{H}) = \{f \in \mathscr{H} : \|f\|_{\mathscr{H}} \leq R\}.$$

Proof: For $\emptyset \neq Y \subset X$ and $\gamma > \eta$ define

$$\psi_Y = \sum_{y \in Y} \gamma\, \psi_y,$$

where ψ_x, $x \in X$, are the **nodal functions** with respect to X. The $2^{\#X} - 1$ functions ψ_Y satisfy[116]

$$\eta < \|\psi_Y - \psi_{Y'}\|_\infty := \max_{x \in \mathscr{X}} |\psi_Y(x) - \psi_{Y'}(x)| \tag{3.40}$$

since for any $Y' \neq Y$ there exists $y \in Y \setminus Y'$ and therefore

$$\|\psi_Y - \psi_{Y'}\|_\infty \geq |\underbrace{\psi_Y(y)}_{=\gamma} - \underbrace{\psi_{Y'}(y)}_{=0}| = \gamma > \eta.$$

[115] It would be sufficient to guarantee that $k(x,x)$ is uniformly bounded, but that is not so easy for "classical" kernels.

[116] Again we use compactness of $\mathscr{X}$ to be able to apply the maximum.

To show that $\psi_Y \in B_R(\mathscr{H})$ for any Y, we compute

$$\begin{aligned}
\|\psi_Y\|_{\mathscr{H}}^2 &= \left\| \sum_{y\in Y} \gamma\, \psi_y \right\|^2 \\
&= \left\| \gamma \sum_{y\in Y} \sum_{x\in X} \left(K(X)^{-1}\right)_{y,x} k(x,\cdot) \right\|^2 \\
&= \gamma^2 \sum_{y,y'\in Y} \sum_{x,x'\in X} \left(K(X)^{-1}\right)_{y,x} \left(K(X)^{-1}\right)_{y',x'} \underbrace{\langle k(x,\cdot), k(x',\cdot)\rangle}_{=K(X)_{x,x'}} \\
&= \gamma^2 \sum_{y,y'\in Y} \sum_{x'\in X} \left(K(X)^{-1}\right)_{y',x'} \underbrace{\sum_{x\in X} \left(K(X)^{-1}\right)_{y,x} K(X)_{x,x'}}_{=(K(X)^{-1}K(X))_{y,x'}=I_{y,x'}=\delta_{y,x'}} \\
&= \gamma^2 \sum_{y,y'\in Y} \left(K(X)^{-1}\right)_{y,y'} = \gamma^2 \sum_{y\in Y} \left(K(X)^{-1} 1_Y\right)_y \le \gamma^2 \sum_{y\in Y} \left| \left(K(X)^{-1} 1_Y\right)_y \right| \\
&\le \gamma^2 \sqrt{\#Y} \left(\sum_{y\in Y} \left(\left(K(X)^{-1} 1_Y\right)_y \right)^2 \right)^{1/2} = \gamma^2 \sqrt{\#Y} \left\| K(X)^{-1} 1_Y \right\|_2 \\
&\le \gamma^2 \sqrt{\#Y} \underbrace{\|1_Y\|_2}_{=\sqrt{\#Y}} \left\| K(X)^{-1} \right\|_2 = \gamma^2 \#Y \left\| K(X)^{-1} \right\|_2,
\end{aligned}$$

where $1_Y \in \mathbb{R}^X$ is the vector with

$$(1_Y)_x = \begin{cases} 1, & x \in Y, \\ 0, & x \notin Y. \end{cases}$$

Since this has to hold for *any* $\gamma > \eta$, it follows that $\psi_Y \in B_R(\mathscr{H})$ if

$$\eta^2 \#Y \left\| K(X)^{-1} \right\|_2 \le R^2 \qquad \Leftrightarrow \qquad \left\| K(X)^{-1} \right\|_2 \le \frac{1}{\#Y} \frac{R^2}{\eta^2}.$$

Since $\#Y < \#X$ this is satisfied whenever the second condition in (3.39) is fulfilled. □

Remark 3.40. Theorem 3.39 is not only a statement on packing numbers. The proof shows that if the $\|K(X)^{-1}\|_2$ is sufficiently small, then the characteristic functions for different labelings, here in $\{0,1\}^X$ are separated *everywhere* on $[0,1]^d$ by the prescribed quantity η.

For any **stationary kernel** we can now bound the norm of the inverse *explicitly*.

Theorem 3.41 (Stationary kernels)**.** *If* $k(x,x') = g(x-x')$ *is a* ***stationary Mercer kernel*** *on* $\mathscr{X} = [0,1]^d$ *and if* $\hat{g}(\xi) > 0$, $\xi \in \mathbb{R}^d$, *then*

$$\left\| K(X_n)^{-1} \right\|_2 \le n^{-d} \left(\inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi) \right)^{-1} \tag{3.41}$$

where

$$X_n := \left(\frac{\{0,1,\dots,n-1\}}{n} \right)^d. \tag{3.42}$$

Remark 3.42. The point set X_n in (3.42) is the regular grid of width $1/n$ in the unit cube. Any such point can be written as

$$x = \frac{\alpha}{n}, \qquad \alpha < n1_d.$$

Proof: The **inverse Fourier transform** yields

$$g(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(\xi) e^{i\xi\cdot x}\, d\xi$$

and therefore, for $c \in \mathbb{R}^{X_n}$,

$$\begin{aligned}
c^T K(X_n) c &= \sum_{x,x'\in X_n} c_x c_{x'} \underbrace{g(x-x')}_{=k(x,x')} = \sum_{x,x'\in X_n} c_x c_{x'} \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(\xi) e^{i\xi\cdot(x-x')}\, d\xi \\
&= \sum_{\alpha,\alpha'\le n1_d} c_\alpha c_{\alpha'} \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(\xi) e^{i\xi\cdot(\alpha-\alpha')/n}\, d\xi \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(\xi) \left| \sum_{\alpha<n1_d} c_\alpha e^{i(\xi\cdot\alpha)/n} \right|^2 d\xi = \frac{n^d}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{g}(n\xi) \left| \sum_{\alpha<n1_d} c_\alpha e^{i\xi\cdot\alpha} \right|^2 d\xi \\
&\ge \frac{n^d}{(2\pi)^d} \int_{[-\pi,\pi]^d} \hat{g}(n\xi) \left| \sum_{\alpha<n1_d} c_\alpha e^{i\xi\cdot\alpha} \right|^2 d\xi \\
&\ge \frac{n^d}{(2\pi)^d} \inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi) \int_{[-\pi,\pi]^d} \left| \sum_{\alpha<n1_d} c_\alpha e^{i\xi\cdot\alpha} \right|^2 d\xi \\
&= \frac{n^d}{(2\pi)^d} \inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi) \sum_{\alpha,\alpha'<n1_d} c_\alpha c_{\alpha'} \underbrace{\int_{[-\pi,\pi]^d} e^{i\xi\cdot(\alpha-\alpha')}\, d\xi}_{=(2\pi)^d \delta_{\alpha,\alpha'}} \\
&= n^d \inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi) \sum_{\alpha<n1_d} c_\alpha^2 = n^d \inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi) \sum_{x\in X_n} c_x^2
\end{aligned}$$

which bounds the smallest eigenvalue λ_- of $K(X)$ from below by

$$n^d \inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi).$$

Therefore the largest eigenvalue and hence the 2–norm of $K(X)^{-1}$ is bounded from above by

$$n^{-d} \left(\inf_{\xi\in[-n\pi,n\pi]^d} \hat{g}(\xi) \right)^{-1},$$

which is (3.41). □

Combining Theorem 3.39 with Theorem 3.41 and taking into account that $\#X_n = n^d$, we get the following observation.

Corollary 3.43 (Packing with stationary kernels). *If* $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ *is a* ***stationary Mercer kernel*** *for compact metric space* $\mathscr{X} = [0,1]^d$ *then*

$$p\left(B_R(\mathscr{H})_\infty, \eta\right) \geq 2^{n^d} - 1 \qquad \textit{whenever} \qquad 0 < \eta < \left(\inf_{\xi \in [-n\pi, n\pi]^d} \hat{g}(\xi)\right)^{1/2}. \quad (3.43)$$

Remark 3.44. The problem with (3.43) is the fact that whenever the kernel is smooth[117], its Fourier transform decays[118] rapidly and therefore large training sets become troublesome.

3.7 Kernel support vector machines

We finally apply the kernel approach to define a more complex **support vector machine** than the linear one in the introduction section where one considered **separating hyperplanes**[119] from (1.33) which was of the form $f(x) = n \cdot x + c$ and actually is a **dot product kernel** itself. Now the nonlinear **separation function** takes the form[120]

$$\phi(a) + c = \sum_{\xi \in \Xi} a_\xi \, k(\xi, \cdot) + c = a^T K(\Xi, \cdot) + c,$$

where we explicitly define the **bias** $c \in \mathbb{R}$. So far, we leave it open whether we set $\Xi = X$ or not. The separation function defines a **margin**

$$M_{a,c} := \{x \in \mathscr{X} : \phi(a)(x) + c = 0\}$$

and the positive and negative regions

$$M^{\pm}_{a,c} := \{x \in \mathscr{X} : (\pm 1)\,(\phi(a)(x) + c) \geq 0\}.$$

The **decision function** will eventually be

$$f(x) = \operatorname{sgn}\,(\phi(a)(x) + c) = \operatorname{sgn}\left(\sum_{\xi \in \Xi} a_\xi \, k(\xi, x) + c\right) \quad (3.44)$$

and we have to recall the construction process for a. This coefficient vector now replaces the **normal** n in (1.38) and we face the **primal problem**

$$\min_{a,c} \frac{1}{2}\|a\|_2^2 \qquad \text{subject to} \quad y_x\,(\phi(a)(x) + c) \geq 1, \quad x \in X, \quad (3.45)$$

[117] Like differentiable, possibly of higher order.

[118] Functions with compactly supported Fourier transform, so called **bandlimited function** that appear in the Shannon sampling theorem, cf. [31], fail in this theorem whenever the sampling rate gets beyond the Nyquist frequency.

[119] We slightly change the notation here and use the one developed for the dot product kernels.

[120] In contrast to the earlier theory, we replace c by $-c$. Anyway, the sign of the offset does not matter.

with the **Lagrangian**[121]

$$\begin{aligned} g(a,c) &= \frac{a^T a}{2} - \sum_{x\in X} \alpha_x \left(y_x \left(\phi(a)(x)+c\right)-1\right) \\ &= \frac{a^T a}{2} - \sum_{x\in X} \alpha_x \left(y_x \left(\sum_{\xi\in\Xi} a_\xi k(\xi,x)+c\right)-1\right) \end{aligned}$$

and its partial derivatives

$$\frac{\partial g}{\partial a_\xi} = a_\xi - \sum_{x\in X} \alpha_x y_x k(\xi,x), \qquad \xi\in\Xi, \tag{3.46}$$

$$-\frac{\partial g}{\partial c} = \sum_{x\in X} \alpha_x y_x. \tag{3.47}$$

Hence, the resulting **KKT conditions** are

$$\begin{aligned} K(\Xi,X)(\alpha\odot y) - a &= 0, \\ 1^T(\alpha\odot y) &= 0, \\ \alpha_x\left(a^T K(\Xi,x)+c-y_x\right) &= 0, \qquad x\in X, \\ \alpha &\geq 0, \end{aligned}$$

which we again solve by switching to the **dual problem** that is obtained by substituting $a = K(\Xi,X)(\alpha\odot y) =: K(\Xi,X)\hat{\alpha}$ into $g(a,c)$, giving

$$\begin{aligned} g(a,c) &= \frac{\hat{\alpha}^T K(\Xi,X)^T K(\Xi,X)\hat{\alpha}}{2} - \sum_{x\in X}\alpha_x\left(y_x\left(\hat{\alpha}^T K(\Xi,X)^T K(\Xi,x)+c\right)-1\right) \\ &= \frac{1}{2}\hat{\alpha}^T K(\Xi,X)^T K(\Xi,X)\hat{\alpha} - \hat{\alpha}^T K(\Xi,X)^T K(\Xi,X)\hat{\alpha} - c\underbrace{1^T\hat{\alpha}}_{=0} + 1^T\alpha \\ &= 1^T\alpha - \frac{1}{2}\alpha^T Y K(\Xi,X)^T K(\Xi,X) Y \alpha, \end{aligned}$$

where

$$Y = \operatorname{diag}\left[y_x : x\in X\right].$$

We can summarize that observation in the following way.

Lemma 3.45. *Instead of solving the **primal problem** (3.45), we can solve the **dual problem***

$$\max_\alpha 1^T\alpha - \frac{1}{2}\alpha^T Y K(\Xi,X)^T K(\Xi,X) Y \alpha \qquad \textit{subject to} \quad \alpha\geq 0,\ y^T\alpha = 0, \tag{3.48}$$

and set

$$a_\xi = \sum_{x\in X} \alpha_x y_x\, k(\xi,x), \qquad \xi\in\Xi, \tag{3.49}$$

$$c = y_x - \sum_{\xi\in\Xi} a_\xi k(\xi,x), \qquad \alpha_x > 0. \tag{3.50}$$

[121] We take the "intuitive" approach here, but of course we could also use Theorem 2.19. But keep in mind that this is always only the **necessary condition** for a minimum.

The disadvantage of (3.48) is that the matrix $K(\Xi, X)^T K(\Xi, X)$ has to be computed as the product of two matrices of size $\#X \times \#\Xi$ and $\#\Xi \times \#X$ which is quite an effort for large Ξ and X, even if $\Xi = X$.

Because of that, one substitutes the **kernel** directly to the hyperplane method, sets $X = \Xi$ and solves

$$\max_{\alpha} 1^T\alpha - \alpha^T Y K(X) Y \alpha \qquad \text{subject to} \quad \alpha \geq 0,\ y^T\alpha = 0, \tag{3.51}$$

and uses the discriminant function based on

$$f = \sum_{x\in X} \alpha_x y_x\, k(x, \cdot),$$

see e.g. [51]. Since[122]

$$\|f\|^2_{\mathscr{H}} = \left\langle \sum_{x\in X} \alpha_x y_x\, k(x,\cdot), \sum_{x\in X} \alpha_x y_x\, k(x,\cdot) \right\rangle = \alpha^T Y K(X) Y \alpha,$$

this is the dual problem to the primal problem

$$\min_{a,c} \|\phi\,(a)\,\|^2_{\mathscr{H}} \qquad \text{subject to} \quad y_x\,(\phi(a)(x) + c) \geq 1, \quad x \in X, \tag{3.52}$$

with its Lagrangian

$$\begin{aligned} g(a,c) &= \frac{\|\phi\,(a)\,\|^2_{\mathscr{H}}}{2} - \sum_{x\in X} \alpha_x\,(y_x\,(\phi(a)(x) + c) - 1) \\ &= \frac{1}{2} a^T \left[\langle k(x,\cdot), k(x',\cdot)\rangle : \begin{array}{c} x \in X \\ x' \in X \end{array} \right] a - \sum_{x\in X} \alpha_x \left(y_x \left(a^T K(X,x) + c \right) - 1 \right) \\ &= \frac{1}{2} a^T K(X) a - \left(a^T K(X) + 1 \right)(\alpha \cdot y) + 1^T\alpha \end{aligned}$$

yielding the KKT conditions

$$\begin{aligned} K(X)\,((\alpha \odot y) - a) &= 0, \\ 1^T(\alpha \odot y) &= 0, \\ a^T K(X)(\alpha \odot y) + c(\alpha \odot y) &= \alpha, \\ \alpha &\geq 0, \end{aligned}$$

from which the dual problem (3.51) can be deduced.

Lemma 3.46. *Instead of solving the* ***primal problem*** (3.52)*, we can solve the* ***dual problem*** (3.51) *and compute* $a = \alpha \cdot y$ *and* c *like in Lemma 3.45.*

[122] We do not have to write down every step of this standard computation, do we?

3.8 Examples for kernel learning

Let us apply the theory developed before to some labeling problems that we want to solve by kernel based support vector machines. We will always set $\Xi = X$, so that our dual problem takes the form

$$\max_{\alpha} 1^T\alpha - \frac{1}{2}\alpha^T YG(X)Y\alpha \qquad \text{s.t.} \quad \begin{array}{rcl} y^T\alpha & = & 0, \\ \alpha & \geq & 0, \end{array} \qquad G(X) = \begin{cases} K(X)^TK(X), \\ K(X), \end{cases} \tag{3.53}$$

depending on which type of optimization we chose in the preceding subsection. Let us look at $YG(X)Y$ first. Since Y is a labeling, the respective diagonal element satisfies

$$Y_{xx} = y_x = \begin{cases} 1, & x \in X_+, \\ -1, & x \in X_-, \end{cases} \qquad x \in X,$$

and therefore

$$YK(X)Y = \begin{bmatrix} K(X_+) & -K(X_+, X_-) \\ -K(X_-, X_+) & K(X_-) \end{bmatrix}$$

if we order X as $X = (X_+, X_-)$. Similarly, we can apply the sign pattern to the partitioning

$$\begin{aligned} &K(X)^TK(X) = K(X)^2 \\ &= \begin{bmatrix} K(X_+)K(X_+) + K(X_+, X_-)K(X_-, X_+) & K(X_+)K(X_+, X_-) + K(X_+, X_-)K(X_-) \\ K(X_-)K(X_-, X_+) + K(X_-, X_+)K(X_+) & K(X_-)K(X_-) + K(X_-, X_+)K(X_+, X_-) \end{bmatrix} \end{aligned}$$

Let us start to implement this in `octave` where we will make use of the built-in function `qp` for **quadratic programming**. The **normal form** used by `qp` is of the form

$$\min_x \frac{1}{2}x^THx + q^Tx \qquad \text{s.t.} \quad \begin{array}{c} Ax = b, \\ \ell \leq x \leq u. \end{array} \tag{3.54}$$

This means we have to define

$$H = YG(X)Y, \quad q = -1, \qquad A = y^T, \qquad \ell = 0.$$

Moreover, the routine `qp` needs a **feasible** initial value x_0 which we can choose as $x_0 = 0$, in particular since the gradient of the target function at $x_0 = 0$ is 1 and therefore this point cannot be a local extremum. For the solution x of (3.54) we then get that

$$a = y \cdot \alpha = Y\alpha.$$

Let us start with building the matrix G needed in the dual problem. Having at hand a kernel function like in Prog. 3.1, it is easy to set up the matrix in the "standard" kernel case, see Prog. 3.2. Then one only has to call `qp` with the appropriate parameters as is done in Prog. 3.3. We can use this function to also compute c by first finding an index $x \in X$ such that $\alpha_x > 0$ and then setting

$$\begin{aligned} c &= \underbrace{y_x}_{\in\{\pm 1\}} - \sum_{x' \in X} k(x, x')a_{x'} = y_x - (K(X)a)_x = y_x - (\underbrace{Y^2}_{=I}K(X)Y^2a)_x \\ &= y_x - (Ha)_x = y_x - \underbrace{Y_{xx}}_{=y_x}(HYa)_x = y_x\,(1 - ((H(y \odot a))_x) \\ &= y_x\,(1 + (H(y \odot a))_x) = y_x\,(1 + (H\alpha)_x)\,, \end{aligned}$$

```
% Kernel learning
% Setup kernel matrix, inefficient implementaion

% Data:
%    k   : Function handle for kernel
%    X,Y : Data sets (column matrices)

function K = SVMKernMat( k,X,Y )
  [mX,nX] = size(X);
  [my,nY] = size(Y);

  K = zeros( nX,nY );

  for jX = 1:nX
    for jY = 1:nY
      K( jX,jY ) = k( X(:,jX), Y(:,jY) );
    end
  end
```

Program 3.1: Computation of the kernel matrix $K(X, X')$.

which can be implemented very easily in `octave`.

We begin with a simple example of two well–separated point sets, one centered around the origin, the other around $(1, 1)$, and we plot them

```
> X = ( rand(2,10) .- .5)/2; Y = 1 .+ ( rand(2,10) .- .5 )/2;
> hold on; plot( X(1,:),X(2,:),"+" ); plot( Y(1,:),Y(2,:),"o" );
```

Next, we define the kernel and to keep it simple we first pick the simple dot product kernel $k(x, x') = x \cdot x'$:

```
> k = @(x,y) x'*y;
```

Determining a and c is now a simple call of our function since `X` takes the role of X_+ and `Y` the one of X_-:

```
% Kernel learning
% Setup optimization matrix, uses KernMat

% Data:
%    k   : Function handle for kernel
%    X,Y : Data sets (column matrices)

function K = SVMOptMat( k,X,Y )
  KK = -SVMKernMat( k,X,Y ); %% labeling!

  K = [ SVMKernMat(k,X,X), KK ; KK', SVMKernMat(k,Y,Y) ];
```

Program 3.2: Computation of the matrix $YK(X)Y$ needed in optimization.

```
% Kernel learning
% Solve Dual optimization problem via qp

% Data:
%    k   : Function handle for kernel
%    X,Y : Data sets (column matrices) for +/- labels

function [a,c] = SVMSolveDual( k,X,Y )
  [mX,nX] = size(X);
  [mY,nY] = size(Y);
  n = nX + nY;

  H = SVMOptMat(k,X,Y);
  q = -ones( n,1 );
  x0 = zeros( n,1 );
  y = [ ones( 1,nX ), -ones( 1,nY) ]';
  l = zeros( n,1 );
  u = [];

  a = qp( x0,H,q,y',0,l,u );
  [amx,j] = max( abs(a) ); % find location != 0

  c = y(j) * ( 1-H( j,: )*a );
  a = y .* a;
```

Program 3.3: Computation of a and c by means of qp.

```
> [a,c] = SVMSolveDual( k,X,Y );
```

In the vector a we can now look for the support vectors, i.e., the points where $\alpha_x \neq 0$

```
> sv = find( abs(a) > 10*eps )
sv =

    7
   16
> Z = [X,Y]; plot( Z(1,sv),Z(2,sv),"*" );
```

where we also plotted the points belonging to the active constraints. The rest is simple, we evaluate the discriminant function on a subgrid of $\left[-\frac{1}{2}, \frac{3}{2}\right]^2$ and plot the contour lines of the function:

```
> Xs = (-.5:.1:1.5); M = SVMDiscrFun( k,[X,Y],a,c,Xs,Xs);
> contour( Xs,Xs,M );
```

The result can be seen in Fig. 3.1 and it is quite what one expects it to be. No surprise: In this case the support vector machine computes *exactly* the separating hyperplane.

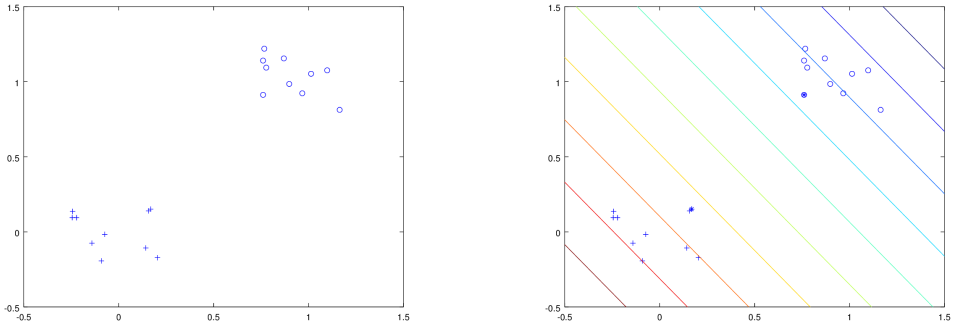


Figure 3.1: The results of the first test, showing the points only *(left)* and the contour plot as well as the support vectors *(right)*

What happens if we approach the same problem with a nonlinear SVM. Let us try another dot product kernel, for example an initial segment of the exponential

$$k(x, x') = (x \cdot x') + \frac{1}{2}(x \cdot x')^2 + \frac{1}{6}(x \cdot x')^3.$$

The application is quite the same

```
> k2 = @(x,y) x'*y + (x'*y)^2/2 + (x'*y)^3/6;
> [a2,c2] = SVMSolveDual( k2,X,Y );
> sv = find( abs(a2) > 10*eps )
sv =

    7
   16
```

but gives different support vectors. So let us look what happens:

```
> clf; hold on;
> plot( X(1,:),X(2,:),"+" ); plot( Y(1,:),Y(2,:),"o" );
> Z = [X,Y]; plot( Z(1,sv),Z(2,sv),"*" );
> Xs = (-.5:.1:1.5); M = SVMDiscrFun( k2,[X,Y],a2,c2,Xs,Xs);
> contour( Xs,Xs,M );
```

The result can be seen in Fig 3.2.

Next, we make it a bit tougher with a weaker separation of points which we now choose as perturbed points around the origin and $(\frac{1}{2}, \frac{1}{2})$ even with possible overlap:

```
> X = (rand(2,20) .- .5 ); Y = .5 .+ (rand(2,20) .- .5 );
> [a,c] = SVMSolveDual( k,X,Y );
> [a2,c2] = SVMSolveDual( k2,X,Y );
```

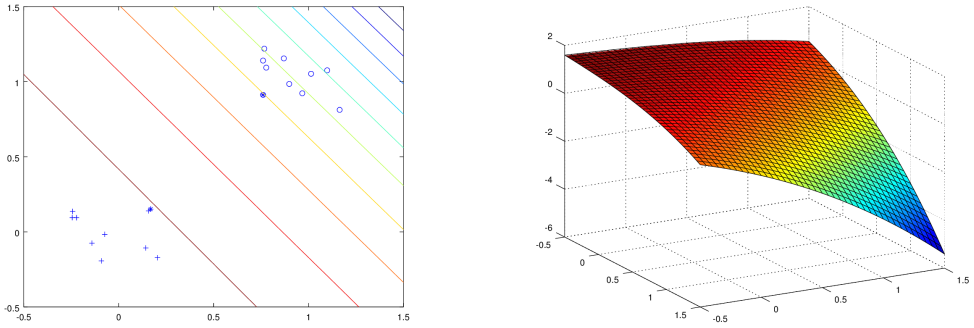


Figure 3.2: The SVM based on the nonlinear kernel *(left)* whose contour plot is now slightly curved. This can also be seen from the surface *(right)*.

If we now consider the two support vectors, we find that they even have totally different length:

```
> sv = find( abs(a) > 10*eps );
> sv2 = find( abs(a2) > 10*eps );
> [ length(sv) length(sv2) ]
ans =

   12    34
```

And the plot of the two kernels shows quite a different behavior as can be seen

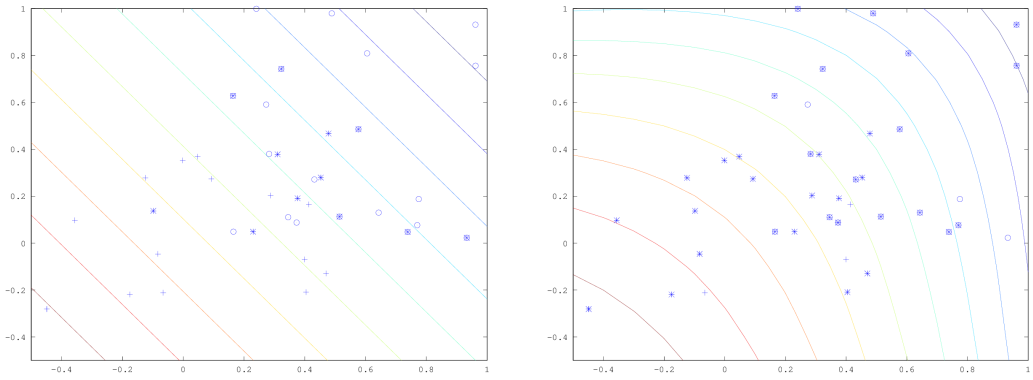


Figure 3.3: Classification by the two different kernels

in Fig. 3.3. Of course, the result will vary with the random data, but in one test[123] the correct classification by the hyperplane method gave 25% for X_+ and 70% for X_- while the cubic dot product kernel provided 90% for X_+ and 95% for X_-. Some results for 100 tests are shown in table 1.

[123]The routines are there and can be downloaded on `Stud.IP`. So test by yourself.

Kernel	X_+ min	X_+ max	X_+ avg	X_+ min	X_+ max	X_+ avg
Linear	0%	100%	69.1%	0%	100%	67.5%
Cubic	35%	100%	88.3%	30%	100%	84.3%

Table 1: Random points with two different kernels

Of course, we should also try at least one stationary and one radial kernel. For the stationary kernel we can use, according to Proposition 3.18, odd degree cardinal B–splines with knots at $\pm\frac{2k+1}{2}$, $k = 0, \ldots, m$, which we evaluate by the **B–spline recurrence**, cf. [7, 8, 50, 46] and use an `Octave` function from the "spline lecture toolbox". From this function, we can build a routine `SVMBSplEval (m,x)` which

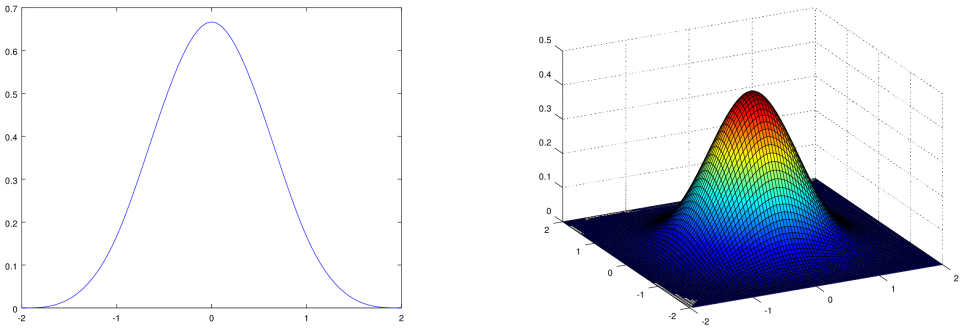


Figure 3.4: A centered B–spline *(left)* and a centered tensor product B–spline *(right)*, both of degree 3. There is no visible difference to the degree 5 B–spline.

is given the degree `m` and as `x` a point $x \in \mathbb{R}^d$ for which it computes the value $M_m(x)$ of the tensor product B–Spline at x. The setup is as usual

```
>> X = (rand(2,20) .- .5 ); Y = .5 .+ (rand(2,20) .- .5 );
>> k1 = @(x,y) x'*y;
>> k3 = @(x,y) x'*y + (x'*y)^2/2 + (x'*y)^3/6;
>> ks3 = @(x,y) SVMBSplEval( 3,x-y );
>> ks5 = @(x,y) SVMBSplEval( 5,x-y );
```

and for the generation of the plots we use a function `SVMPlot (k,X,Y,Xs,Ys)` that plots the data points, support vectors and the contour of the discriminant for the given parameters. The success rates are

Kernel	X_+	X_-
Linear	55 %	100 %
Cubic	95 %	70 %
M_3	70%	75%
M_5	40 %	90 %

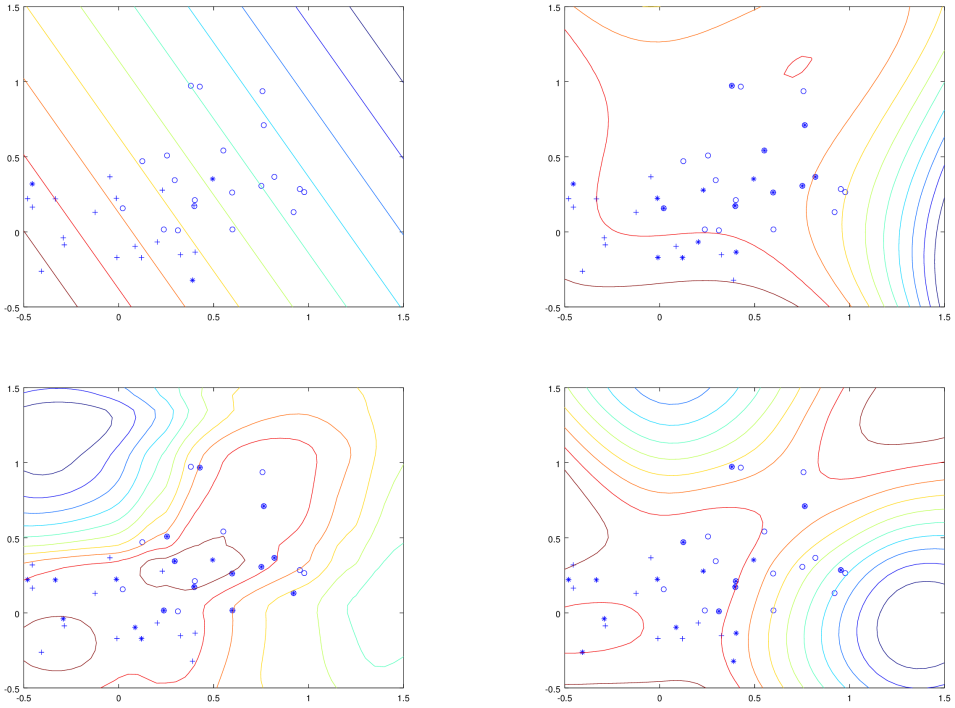


Figure 3.5: Points, supports and contours for the (irreproducible) example The kernels are linear *(top left)*, cubic *(top right)*, cubic spline *(bottom left)* and quintic spline *(bottom right)*.

Keep in mind that there are no real "overlearning" effects here as we always consider spaces with the same number of basis functions and matrices of the same size. What changes is the flexibility of the involved kernel functions.

For a slightly more "statistical" approach we again run our test 100 times with the four kernels, this time with 20 random points in a circle with radius $\frac{1}{2}$ around the origin and 20 random points in a circle with radius $\frac{1}{2}$ around $[t, t]$ for different values of t. The results are listed in Table 2. Finally, we can also try radial kernels, for example some exponentials

```
>> ke1 = @(x,y) exp( -norm(x-y,2)^2 );
>> ke2 = @(x,y) exp( -norm(x-y,2)^2/10 );
```

or some inverse multiquadrics

```
>> km1 = @(x,y) (.1 + norm(x-y,2)^2 )^(-1/2);
>> km2 = @(x,y) (.1 + norm(x-y,2)^2 )^(-1);
```

Surprisingly enough, the radials methods are not really affected by the distance between the sampling circles, at least not as far as their classification rate is concerned.

	t = 1		t = .7		t = .5	
Kernel	X_+	X_-	X_+	X_-	X_+	X_-
Linear	100 %	100 %	84.35 %	84.05 %	69.10 %	73.55 %
Cubic	100 %	100 %	95.50 %	94.45 %	86.60 %	84.45 %
M_3	100 %	100 %	100 %	100 %	97.60 %	98.25 %
M_5	100 %	100 %	96.85 %	97.05 %	84.65 %	83.85 %
$e^{-\|x-x'\|^2}$	65 %	63.75 %	65.1 %	65.9 %	68 %	64.75 %
$e^{-\|x-x'\|^2/10}$	56.75 %	54.9 %	53.15 %	60 %	56.75 %	58.7 %
$c = 0.1$, $\alpha = \frac{1}{2}$	99.2 %	98.25 %	98.85 %	97.85 %	99 %	99.2 %
$c = 0.1$, $\alpha = 1$	99.45 %	99.55 %	99.7 %	99.8 %	100 %	100 %

Table 2: Testing different kernels on different distances.

3.9 Choosing the kernel

There remains an important question: How to choose the kernel? Normally, this is just a deliberate choice made by the user of the SVM library, but there exists, of course, also some theory on how to choose the kernel.

The oldest ideas in kernel selection are older than machine learning and were introduced in Geostatistics in a process called **kriging**[124]. In a nutshell, kriging is exactly our well–known approximation or interpolation process of the form

$$\min_a \sum_{x\in X} (\phi(a)(x) - y_x)^2 + \lambda a^T B a,$$

or

$$\min_a a^T B a, \qquad \text{subject to} \quad \phi(a)(x) = y_x, \quad x \in X,$$

where B is a symmetric positive semidefinite matrix describing the **energy functional**. Also here it is requested that $\Phi = \{k(\xi,\cdot) : \xi \in \Xi\}$ and that k is a **Mercer kernel**[125]. The difference is that the parameters in the kernel, for example the ones in the exponential and inverse multiquadric kernels from Example 3.23 are determined in a *data dependent* way.

Definition 3.47 (Variogram)**.** Given a set finite $X \subset \mathscr{X}$ of a metric space $\mathscr{X}$ and $y \in \mathbb{R}^X$,

1. we denote by
$$\mathbb{R} \supset d(X) = \{d(x,x') : x,x' \in X\} \setminus \{0\}$$
the (finite) set of all distances occurring in X.

2. The (empiric) **variogram** of y, $V(y) : d(X) \to \mathbb{R}$ is the defined as
$$v(d) := \frac{1}{\#X_d} \sum_{(x,x')\in X_d} (y_x - y_{x'})^2, \qquad X_d := \{(x,x') : d(x,x') = d\}. \quad (3.55)$$

[124]Named after the South African mining engineer D. G. Krige.
[125]In our terminology, in the kriging context it is just "positive definite".

Given the data, one can compute the empirical variogram and consider this to be yet another **training set**: just set $D = d(X)$ and $\tilde{y}_d := \nu(d)$, $d \in D$. If now the kernel is a parameterized **stationary kernel**[126] i.e.,

$$k(x, x') = g_a(x - x'), \qquad a \subset \mathbb{R}^n,$$

like the **dot product kernel** from (3.7) or the radial kernels from Definition 3.23, we can fix the parameter by another least squares fit

$$\min_a \sum_{d \in D} \frac{1}{\#X_d} \sum_{(x,x') \in X_d} \left(g_a(x - x') - \nu(d)\right)^2,$$

which is supposed to determine the "best" kernel for the problem. In Geostatistics there often is a distinction between the variogram and the **semivariogram** $\frac{1}{2}\nu(d)$, which makes some sense from a probabilistic perspective but is not relevant for our optimization here.

Learning the kernel has also been considered in "native" learning theory, for example in [37]. There, the regression approach[127]

$$\min_a \sum_{x \in X} \left(\phi(a)(x) - y_x\right)^2 + \lambda \|\phi(a)\|^2_{\mathscr{H}}, \qquad \Phi = \operatorname{span}\{k(x, \cdot) : x \in X\},$$

is considered and the resulting optimal function $\phi^* = \phi(a^*)$ is now taken to be a function of the kernel k and denoted by $\phi^*(k)$. Now, we can minimize this expression over all Mercer kernels,

$$\min_{k \in \mathscr{K}} \phi^*(k) = \min_k \left(K(X)^T K(X) + \lambda K(X)\right)^{-1} K(X)^T y,$$

which looks complicated enough, in particular when taking into account that $\mathscr{K}$ stands for the set of all Mercer kernels here. Alternatively, we can perform the double minimization

$$\min_{a \in \mathbb{R}^{\#X}, k \in \mathscr{K}} \sum_{x \in X} \left(a^T K(X, x) - y_x\right)^2 + \lambda\, a^T K(X) a. \tag{3.56}$$

This still looks[128] nontrivial, but there are two fundamental observations:

1. the space $\mathscr{K}$ of Mercer kernels is **convex**, see Proposition 3.29 that says that $k, k' \in \mathscr{K}$ implies $\alpha k + (1 - \alpha)k' \in \mathscr{K}$ as well.

2. The functional in (3.56) is also convex in k under suitable assumptions.

These two observations allow to make conclusions about the existence and uniqueness of minima, see Lemma 2.9. The rest is a lot of nice and interesting mathematics in [37], including fixpoint theorems and the famous **minimax theorem** due to John von Neumann, cf. [38, 39]. Unfortunately, this goes too far for an introductionary lecture.

[126] Stationary is important here since stationary kernels depend on a distance only. On the other hand, stationarity is one of the fundamental assumptions in kriging.

[127] The paper considers this in wider generality

[128] And also is.

Networking is rubbish; have friends instead.

S. Winwood

4 Neural networks

The basic idea of neural networks is to mimic the human brain that consists of connected neurons which transfer information forward and react according to

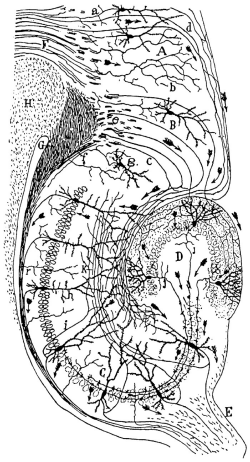


Figure 4.1: **Hippocampus** of a rodent, due to Ramón y Cajal (1911), showing the connection of neurons into a complex network. The hippocampus is one of the oldest and simplest parts of the brain. *(Source: Wikipedia, gemeinfrei)*

this accumulated information.

The main idea of this concept is that a single neuron is an extremely simple object and that the combination and interaction of the network finally results in a system that can handle complicated tasks, in particular in pattern recognition.

4.1 Definition and basic facts

Let us begin with the main concept, the mathematical model of a neuron.

Definition 4.1 (Neuron)**.** A **neuron** $\varphi : \mathbb{R}^m \to \mathbb{R}$ is a function of the form

$$\varphi(x) = \varphi_{\sigma,w}(x) = \sigma(b + w \cdot x), \tag{4.1}$$

with a **weight** $w \in \mathbb{R}^m$, a **bias** b and an **activation function** $\sigma : \mathbb{R} \to \mathbb{R}$. Extending x and w into $\hat{x} = \begin{bmatrix} 1 \\ x \end{bmatrix} \in \mathbb{R}^{m+1}$ and $\hat{w} = \begin{bmatrix} b \\ w \end{bmatrix} \in \mathbb{R}^{m+1}$, we can also use the more compact way of writing $\varphi(x) = \sigma(\hat{w} \cdot \hat{x})$.

A neural network is now a collection of such neurons with varying weights[129] and *a single* activation function σ. Such a network is organized in the following way.

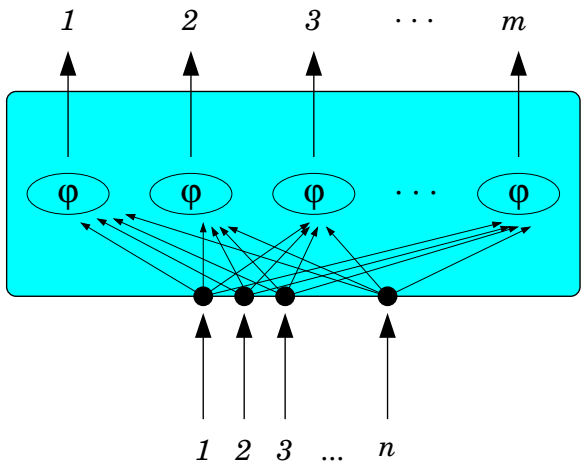


Figure 4.2: The layout of a neural network with just one hidden layer and no output layer. It also illustrates how a single hidden layer works.

Definition 4.2 (Layers and networks)**.** A **neural network** of **depth** $d \in \mathbb{N}$ consists of several layers of the following types:

1. the **input layer** consists of n_0 neurons φ with weights $\hat{w}_1^0, \ldots, \hat{w}_{n_0}^0 \in \mathbb{R}^n$. Each neuron of the input layer is a function from $\mathbb{R}^n$ to $\mathbb{R}$ where n is the original number of variables.

2. A **hidden layer** on level j, $j = 1, \ldots, d$, consists of n_j neurons with weights $\hat{w}_1^j, \ldots, \hat{w}_{n_j}^j \in \mathbb{R}^{n_{j-1}}$, being able to process the output from level $j-1$ of the network.

3. The **output layer** is a single neuron with weight $\hat{w}^{d+1} \in \mathbb{R}^{n_d}$ that has the purpose of combining the output from the preceding levels into a single value.

Any such combination of layers defines a **feedforward neural network**[130] func-

[129] The weights can and will depend on the respective neuron.
[130] The "feedforward" is pretty much self explaining.

tion f by setting

$$\begin{aligned} x_0 &:= \left[\sigma(\hat{w}_j^0 \cdot \hat{x}) : j = 1, \dots, n_0\right] \in \mathbb{R}^{n_0} \\ x_k &:= \left[\sigma(\hat{w}_j^k \cdot \hat{x}_{k-1}) : j = 1, \dots, n_k\right] \in \mathbb{R}^{n_k}, \qquad k = 1, \dots, d, \\ f(x) := f_w(x) &:= \hat{w}^{d+1}\hat{x}_k = \sigma\left(w_0^{d+1} + \sum_{j=1}^{n_d} w_j^d x_{d,j}\right), \end{aligned}$$

where $w = \left(w_j^k : j = 1, \dots, n_k,\ k = 1, \dots, d+1\right)$ denotes the **weight vector** of the neural network.

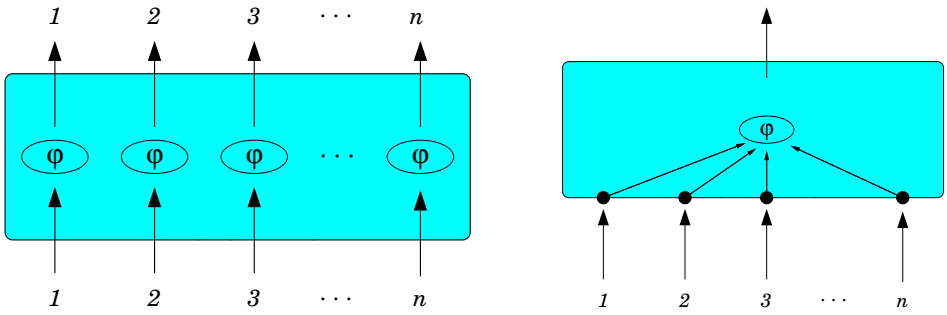


Figure 4.3: An input layer whose weights are unit vectors *(left)* and an output layer *(right)*.

Remark 4.3. In the beginning, neural networks were usually "flat" with quite few hidden layers. This was mainly motivated by **Kolmogorov's theorem**[131] that could be interpreted as the observation that a single layer with a modest number of neuron can represent any function of almost arbitrary complexity. What is true and false about this and why this has to be taken with care, will be discussed later. A the moment, **deep learning** with more structured networks and several hidden layers is more popular.

Remark 4.4 (Ridge functions)**.** A function of the form $\varphi(x) = \sigma(\hat{w} \cdot \hat{x})$, $w, x \in \mathbb{R}^n$, is often called a **ridge function**. Such functions are constant along any **hyperplane** of the form

$$y + w^\perp, \qquad w^\perp := \{x \in \mathbb{R}^n : w \cdot x = 0\},$$

since for any $x' \in w^\perp$

$$\begin{aligned} \varphi(x + x') &= \sigma(w_0 + w \cdot (x + x')) = \sigma(w_0 + w \cdot x + \underbrace{w \cdot x'}_{=0}) \\ &= \sigma(w_0 + w \cdot x) = \varphi(x). \end{aligned}$$

[131] We will learn about that later.

Example 4.5 (Activation functions)**.** Typical activation functions for neural networks are

1. $\sigma(x) = \operatorname{sgn} x$, i.e.,

$$\varphi(x) = 1 \quad \Leftrightarrow \quad 0 < \hat{w} \cdot \hat{x} = b + \sum_{j=1}^{n} w_j x_j \quad \Leftrightarrow \quad \sum_{j=1}^{n} w_j x_j > -b,$$

   and $\varphi(x) = 0$ if $w \cdot x \leq -b$, hence the neuron **fires** if the inner product $w \cdot x$ is larger than the **threshold** $-b$.

2. $\sigma : \mathbb{R} \to [0,1]$ is any monotonically increasing function, forming a continuous counterpart to the **jump function** $x \mapsto \operatorname{sgn} x$. Such a function is often called a **sigmoidal function** if it has the additional property $\sigma(0) = 0$ and $\sigma(1) = 1$. Note that the two locations 0 and 1 are a totally deliberate choice here but no loss of generality as any other activation levels can be obtained by choosing $\hat{w}$ appropriately.

3. A concrete and frequently used candidate is the **sigmoid function** of the form

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{4.2}$$

   see [2] and Fig. 4.4. It has no compact support but satisfies

$$\lim_{x \to -\infty} \sigma(x) = 0 \qquad \text{and} \qquad \lim_{x \to \infty} \sigma(x) = 1.$$

   Nevertheless it is close to 0 and 1 quite early as Fig. 4.4 shows.

The notation from Definition 4.2 turns into an index battle quite soon. To avoid that, we arrange the (row) vectors w_j^k, $j = 1, \ldots, n_k$, into a matrix

$$W_k := \left[w_1^k, \ldots, w_{n_k}^k \right]^T \in \mathbb{R}^{n_k \times n_{k-1}}, \qquad \hat{W}_k := \left[\begin{array}{c|c} b_0 & \\ \vdots & W_k \\ b_{n_k} & \end{array} \right],$$

and extend $\sigma : \mathbb{R} \to \mathbb{R}$ in a simple "`Matlab` fashion" into $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ by setting

$$\sigma(x) = [\sigma(x_j) : j = 1, \ldots, n] \qquad \text{and} \qquad \hat{\sigma}(x) = \left[\begin{array}{c} 1 \\ \sigma(x) \end{array} \right].$$

Then we have that

$$\begin{aligned} \hat{x}_0 &= \hat{\sigma}(\hat{W}_0 \hat{x}) && (4.3) \\ \hat{x}_k &= \hat{\sigma}(\hat{W}_k \hat{x}_{k-1}), \qquad k = 1, \ldots, d, && (4.4) \\ f_w(x) &= \hat{\sigma}(\hat{W}_{d+1} \hat{x}_d) && (4.5) \end{aligned}$$

and therefore

$$f_w(x) = \hat{\sigma}\left(\hat{W}_{d+1} \hat{\sigma}\left(\cdots \hat{\sigma}\left(\hat{W}_1 \hat{\sigma}(\hat{W}_0 \hat{x}) \right) \cdots \right) \right). \tag{4.6}$$

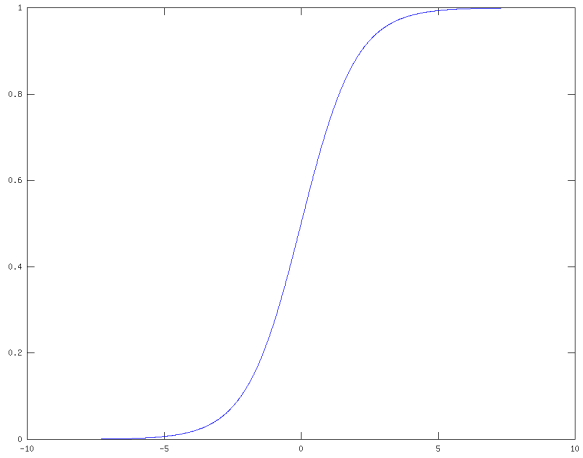


Figure 4.4: The **sigmoid function** (4.2). By rescaling into $\sigma(\lambda\cdot)$, $\lambda > 0$, the slope can be made arbitrarily steep or flat. And although the function never assumes the values 0 and 1 exactly, it "practically" gets close to them very soon.

This expression shows that **deep learning** is a little bit awkward to write down but also the layered recursion clearly offers some intricate nonlinearities.

For a single hidden layer network, the expression is still fairly simple:

$$f_w(x) = \hat{\sigma}\left(\hat{W}_2\hat{\sigma}\left(\hat{W}_1\hat{\sigma}(\hat{W}_0\hat{x})\right)\right). \tag{4.7}$$

We will use this simpler identity later when considering how neural networks learn.

It is easy to see that we can implement Boolean functions by means of neurons, simply setting, for $x, y \in \{0, 1\}$

$$x \vee y = \sigma(x + y), \qquad x \wedge y = \sigma(x + y - 1), \qquad \neg x = \sigma(1 - x)$$

where σ is any strictly monotonically increasing function with $\sigma(0) = 0$ and $\sigma(1) = 1$. Since any **Boolean function**, i.e., any hard decision can be written in terms of these three elementary expressions, neural networks are capable, at least in principle, to learn any function $f : \{0, 1\}^n \to \{0, 1\}$ that performs such lists of decisions. This can even be made more quantitative as the following result from [2, Theorem 7.1] shows.

Theorem 4.6. *Any **Boolean function** $f : \{0, 1\}^n \to \{0, 1\}$ with n parameters can be realized by a neural network consisting of one **hidden layer** of 2^n neurons and one output layer with suitable σ and weights.*

The simple Boolean operations above correspond to the weights $[0, 1, 1]$, $[-1, 1, 1]$, $[1, -1]$, respectively and σ can actually be chosen quite liberal.

Remark 4.7. Proper piecewise polynomial activation functions for the above construction could be

$$\sigma(x) = \begin{cases} 0, & x \le 0, \\ 2^{m-1}x^m, & 0 \le x \le \frac{1}{2}, \\ 1 - 2^{m-1}(1-x)^m, & \frac{1}{2} \le x \le 1, \\ 1, & x \ge 1, \end{cases}$$

where $m > 0$ is an arbitrary parameter. The construction has the advantage that

$$\sigma^{(k)}(0^+) = \sigma^{(k)}(1^-) = 0, \qquad k = 0, \ldots, m-1,$$

so that the sigmoidal function σ is really **flat** at $x = 0, 1$.

The focus on sigmoidal functions for a neural network seems overly restrictive: why don't we use activation functions that range, say, from -1 to 1 and also allow **inhibition** in addition to activation. The answer is that the next layer is actually able to compensate that. To that end, we note that the range of

$$\tilde{\sigma} := 2\sigma(\cdot) - 1$$

is $[-1, 1]$ if σ is sigmoidal, then, for $j = 1, \ldots, n_k$,

$$x_{kj} := (x_k)_j = \tilde{\sigma}(w_j^k \cdot x_{k-1} + b_j^k) = 2\sigma(w_j^k \cdot x_{k-1} + b_j) - 1$$

and[132]

$$\begin{aligned} x_{k+1,j} &= \tilde{\sigma}(w_j^{k+1} \cdot x_k + b_j^{k+1}) = \tilde{\sigma}\left(w_j^{k+1} \cdot (2\sigma(w^k \cdot x_{k-1} + b_j) - 1) + b_j^{k+1}\right) \\ &= \tilde{\sigma}\left(2w_j^{k+1} \cdot \sigma(w^k \cdot x_{k-1} + b_j) + b_j^{k+1} - w_j^{k+1} \cdot 1\right) \\ &=: \tilde{\sigma}\left(\tilde{w}_j^{k+1} \cdot \sigma(w^k \cdot x_{k-1} + b_j) + \tilde{b}_j^{k+1}\right), \end{aligned}$$

so that $\tilde{\sigma}$ can be replaced by σ in all levels of the neural network except the output level.

4.2 What to learn?

The learning problem to be handled by neural networks is again the penalized quadratic regression one, i.e.,

$$F(w) := \min_w \sum_{x \in X} (f_w(x) - y_x)^2 + \lambda w^T B w \tag{4.8}$$

with a symmetric positive definite regularization matrix B. Hence, we have to minimize the expression[133]

$$\sum_{x \in X} f_w^2(x) - 2y_x f_w(x) + \lambda w^T B w$$

[132] We use the `Matlab` like vectorization of σ.

[133] The constant part $\sum y_x^2$ has been dropped already.

which leads to

$$0 = \frac{1}{2}\frac{\partial F(w)}{\partial w} = \sum_{x\in X} f_w(x)\,\frac{\partial f_w(x)}{\partial w} - y_x\frac{\partial f_w(x)}{\partial w} + \lambda B w, \tag{4.9}$$

which, however, is hard to solve directly for w. But also if we want to apply a descent method as introduced in Section 2.5.1, a **descent direction** is needed, i.e., a direction y such that

$$y^T \nabla F(w) = y^T \frac{\partial F}{\partial w}(w) < 0.$$

To that end, we have to evaluate the expression in (4.9) which means that we need to be to differentiate $f(w, x)$ with respect to w.

Since we know from basic calculus that the **concatenation**[134] of differentiable functions is again differentiable, the heart of the **chain rule**, we can easily make the following conclusion, taking into account that the roles of w and x in (4.1) are actually *symmetric*.

Proposition 4.8. *If the activation function σ is differentiable, then the function $f(w, x)$ is differentiable with respect to w and x.*

4.3 The simplest case: the perceptron

To understand how a neural network learns and to get the basic idea of the **backpropagation** method that we will encounter in Section 4.4, we will consider the simplest neural network, namely one that consist of a single neuron; according to [11], this is the **perceptron** model introduced by Rosenblatt in [44]. This consists of

$$f_{w,b}(x) = \sigma(w \cdot x + b), \qquad x, w \in \mathbb{R}^n,\ b \in \mathbb{R},$$

where σ is again a sigmoidal function, i.e., monotonically increasing with

$$\lim_{x\to-\infty} \sigma(x) = 0, \qquad \lim_{x\to\infty} \sigma(x) = 1.$$

We will show that like the simplest support vector machine this approach is capable of finding a **separating hyperplane** for labeled data (X, y) with $y_x \in \{0, 1\}$, at least whenever such a hyperplane exists. Again we have a optimization problem of the form

$$F(w, b) = \frac{1}{2}\sum_{x\in X} (y_x - f_{w,b}(x))^2 + \frac{\lambda}{2}\left\|\begin{bmatrix} w \\ b \end{bmatrix}\right\|_2^2 \tag{4.10}$$

with a regularizing parameter λ, and get

$$\nabla F(w) = \sum_{x\in X} \nabla_{w,b} f_{w,b}(x)\,(y_x - f_{w,b}(x)) + \lambda \begin{bmatrix} w \\ b \end{bmatrix}.$$

[134] This means $f \circ g$ with $(f \circ g)(x) := f(g(x))$.

Now,

$$\begin{aligned}\nabla_w f_{w,b}(x) &= \nabla_w \sigma(w\cdot x+b) = \left[\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) : j=1,\dots,n\right]\\ &= [\sigma'(w\cdot x+b)\,x_j : j=1,\dots,n] = \sigma'(w\cdot x+b)\,x\end{aligned}$$

and

$$\nabla_b f_{w,b}(x) = \frac{\partial}{\partial b}\sigma(w\cdot x+b) = \sigma'(w\cdot x+b),$$

so that

$$\nabla_{w,b} f_{w,b}(x) = \sigma'(w\cdot x+b)\begin{bmatrix} x\\ 1\end{bmatrix} \tag{4.11}$$

and therefore

$$\nabla F(w) = \sum_{x\in X}\sigma'(w\cdot x+b)\,(y_x - f_{w,b}(x))\begin{bmatrix} x\\ 1\end{bmatrix} + \lambda\begin{bmatrix} w\\ b\end{bmatrix}. \tag{4.12}$$

Now we can give a weak form of Rosenblatt's theorem saying that separating hyperplanes can always be computed by a perceptron in a finite number of steps.

Theorem 4.9. *If the two sets* $X_+ := \{x\in X : y_x = 1\}$ *and* $X_0 := \{x\in X : y_x = 1\}$ *can be separated by a hyperplane, then any optimal solution of* (4.10) *separates hyperplanes for sufficiently small* $\lambda\in\mathbb{R}_+$.

Proof: Suppose that $0<\sigma(0)<1$, otherwise we can simply shift the bias b and let $v\in\mathbb{R}^n$ and $c\in\mathbb{R}$ define a separating hyperplane, i.e.,

$$v\cdot X_0 < c < v\cdot X_+ \qquad\Leftrightarrow\qquad (\alpha v)\cdot X_0 < \alpha c < (\alpha v)\cdot X_+,\quad \alpha>0.$$

Therefore, with $w_\alpha = \alpha v$ and $b_\alpha = -\alpha c$, the continuity of σ and the finiteness of X yield

$$\lim_{\alpha\to\infty}\sigma(w_\alpha\cdot X_0 + b_\alpha) = \sigma\Big(\lim_{\alpha\to\infty}\alpha\underbrace{(v\cdot X_0 - c)}_{<0}\Big) = 0$$

and, analogously,

$$\lim_{\alpha\to\infty}\sigma(w_\alpha\cdot X_0 + b_\alpha) = 1.$$

Hence, for any $\varepsilon>0$ there exists $\alpha=\alpha(\varepsilon)>0$ such that

$$F(w_\alpha,b_\alpha) < \frac{\varepsilon}{2} + \frac{\lambda\alpha^2}{2}\left\|\begin{bmatrix} v\\ c\end{bmatrix}\right\|^2,$$

so that for $\lambda=\lambda(\varepsilon) < \dfrac{\varepsilon}{\alpha^2\|(v,c)\|^2}$ we have that

$$\min_{w,b} F(w,b) \le F(w_\alpha,b_\alpha) < \varepsilon.$$

On the other hand, any misclassification contributes to the error, namely a factor of $>\sigma(0)^2$ if an element of X_0 is falsely classified as 1 and $>(1-\sigma(0))^2$ if an element of X_1 is falsely classified as 0. Consequently, for

$$\varepsilon < \min\left\{\sigma(0)^2, (1-\sigma(0))^2\right\}$$

and $\lambda<\lambda(\varepsilon)$, no misclassifying choice of w can become minimal. □

Remark 4.10. The proof also shows that whenever $0 < \sigma(x) < 1$ for $x \in \mathbb{R}$, then the error will never be exactly zero and $\|w\|$ would diverge to infinity. This is why a regularization term becomes necessary that limits the size of w.

Generally, a perceptron computation consists of fixing $\lambda > 0$ and then solving the optimization problem by a descent algorithm[135] with proper stepsize control, see Setion 2.5.1, but one may also use a coupling of "minimality" and regularization parameter as described in Section 2.6 on penalized optimization.

4.4 Training by backpropagation

Now it is time to compute the derivative of a general neural network f_w. This is a little bit more complicated due to the nested structure of the function. Let us recall the setup

$$\begin{aligned} x_{0j} &= \sigma\left(w_j^0 \cdot x + b_j^0\right), \\ x_{kj} &= \sigma\left(w_j^k \cdot x_{k-1} + b_j^k\right), \qquad k = 1, \dots, d+1, \\ f(w,x) = x_{d+1} &= \sigma\left(w^{d+1} \cdot x_d + b^{d+1}\right) \end{aligned} \tag{4.13}$$

which shows that x_k depends on the variables $x_0, \dots, x_{k-1}$ and $w^0, \dots, w^{k-1}$ as well as of $b^0, \dots, b^{k-1}$. To take the derivative of $f(w,x)$ with respect to w^{d+1} we have to consider, like above

$$\frac{\partial f}{\partial w^{d+1}}(w,x) = \sigma'\left(w^{d+1} \cdot x_d + b^{d+1}\right) x_d$$

and

$$\frac{\partial f}{\partial b^{d+1}}(w,x) = \sigma'\left(w^{d+1} \cdot x_d + b^{d+1}\right),$$

hence, simple and easy,

$$\frac{\partial f}{\partial \widehat{w}^{d+1}}(w,x) = \begin{bmatrix} \frac{\partial f}{\partial w^{d+1}}(w,x) \\ \frac{\partial f}{\partial b^{d+1}}(w,x) \end{bmatrix} = \sigma'\left(w^{d+1} \cdot x_d + b^{d+1}\right) \begin{bmatrix} x_d \\ 1 \end{bmatrix}, \tag{4.14}$$

which can now be evaluated directly. To take a derivative with respect to w^d, we note that exactly in the same way as above we can compute

$$\frac{\partial x_{dj}}{\partial w_j^d}(w,x) = \sigma'\left(w_j^d \cdot x_{d-1} + b_j^d\right) \begin{bmatrix} x_{d-1} \\ 1 \end{bmatrix}, \qquad j = 1, \dots, n_d,$$

and also

$$\frac{\partial f}{\partial x_{dj}}(w,x) = \frac{\partial}{\partial x_{dj}} \sigma\left(\sum_{k=1}^{n_d} w_k^{d+1} x_{dk} + b^{d+1}\right) = \sigma'\left(w^{d+1} \cdot x_d + b^{d+1}\right) w_j^{d+1}.$$

[135] Since σ' is nonlinear, a direct solution of equating (4.12) to zero is not really suitable.

These two identities call for an application of the **chain rule** that leads to

$$\begin{aligned}\frac{\partial f}{\partial \widehat{w}_j^d}(w,x) &= \frac{\partial f}{\partial x_{dj}}(w,x)\,\frac{\partial x_{dj}}{\partial \widehat{w}_j^d}(w,x)\\ &= \sigma'\left(w^{d+1}\cdot x_d + b^{d+1}\right) w_j^{d+1}\sigma'\left(w_j^d\cdot x_{d-1}+b_j^d\right)\begin{bmatrix} x_{d-1}\\ 1\end{bmatrix}\end{aligned}$$

In summary, we have

$$\begin{aligned}\frac{\partial f}{\partial \widehat{w}^d}(w,x) &= \left[\frac{\partial f}{\partial \widehat{w}_j^d}(w,x) : j=1,\dots,n_d\right]\\ &= \sigma'\left(w^{d+1}\cdot x_d + b^{d+1}\right)\begin{bmatrix} w_1^{d+1}\sigma'\left(w_1^d\cdot x_{d-1}+b_1^d\right)\begin{bmatrix} x_{d-1}\\ 1\end{bmatrix}\\ \vdots\\ w_{n_d}^{d+1}\sigma'\left(w_{n_d}^d\cdot x_{d-1}+b_{n_d}^d\right)\begin{bmatrix} x_{d-1}\\ 1\end{bmatrix}\end{bmatrix}\\ &= \sigma'\left(w^{d+1}\cdot x_d + b^{d+1}\right)\left(w^{d+1}\odot\sigma'\left(W^d x + b^d\right)\right)\otimes\begin{bmatrix} x_{d-1}\\ 1\end{bmatrix}.\end{aligned}$$

Here we use for abbreviation purposes two nonstandard matrix products, see for example [20, 21].

Definition 4.11 (Nonstandard matrix products)**.**

1. The **Hadamard product**[136] of two matrices $A, B \in \mathbb{R}^{m\times n}$ is the componentwise product

$$A\odot B := \left[a_{jk}b_{jk} : \begin{matrix} j=1,\dots,m\\ k=1,\dots,n\end{matrix}\right] \in \mathbb{R}^{m\times n}. \tag{4.15}$$

2. The **Kronecker product** $A\otimes B$ of $A\in\mathbb{R}^{m\times n}$ and $B\in\mathbb{R}^{m'\times n'}$ is the block matrix

$$A\otimes B := \begin{bmatrix} a_{11}B & \dots & a_{1n}B\\ \vdots & \ddots & \vdots\\ a_{m1}B & \dots & a_{mn}B\end{bmatrix} \in \mathbb{R}^{mm'\times nn'}. \tag{4.16}$$

Exercise 4.1 Which of the two products is commutative? ◊

The general trick for backpropagation is now relatively easy and mainly bookkeeping with matrices. It is based on the fact that for

$$x_k = [x_{kj} : k=1,\dots,n_d] = \sigma\left(W^k x_{k-1} + b^k\right) \in \mathbb{R}^{n_k}$$

with $W_k \in \mathbb{R}^{n_k\times n_{k-1}}$ we again have

$$\frac{\partial x_{kj}}{\partial \widehat{w}_j^k} = \sigma'\left(w_j^k\cdot x_{k-1}+b_j^k\right)\begin{bmatrix} x_{k-1}\\ 1\end{bmatrix}, \qquad j=1,\dots,n_k,$$

[136] We already used it in the proof of Proposition 3.29.

which we can arrange as columns into the matrix

$$\frac{\partial x_k}{\partial \widehat{w}^k} =: \left[\frac{\partial x_{kj}}{\partial \widehat{w}^k_j} : j = 1, \dots, n_k\right] = \begin{bmatrix} x_{k-1} \\ 1 \end{bmatrix} \otimes \sigma'\left(W^k \cdot x_{k-1} + b^k\right)^T \in \mathbb{R}^{n_{k-1}+1 \times n_k}.$$

On the other hand,

$$\frac{\partial x_{kj}}{\partial x_{k-1,\ell}} = \frac{\partial}{\partial x_{k-1,\ell}} \sigma(w^k_j \cdot x_{k-1} + b^k_j) = \sigma'\left(w^k_j \cdot x_{k-1} + b^k_j\right)(w^k_j)_\ell,$$

hence

$$\frac{\partial x_{kj}}{\partial x_{k-1}} = \sigma'\left(w^k_j \cdot x_{k-1} + b^k_j\right) w^k_j$$

which gives the (transposed) Jacobi matrix[137]

$$\frac{\partial x_k}{\partial x_{k-1}}^T = \operatorname{diag}\left(\sigma'\left(W^k x_{k-1} + b^k\right)\right) W^k \in \mathbb{R}^{n_k \times n_{k-1}}$$

whose jth column is $\frac{\partial x_{kj}}{\partial x_{k-1}}$. Here,

$$\begin{aligned} D_k &:= \operatorname{diag}\left(\sigma'\left(W^k \cdot x_{k-1} + b^k\right)\right) \\ &= \begin{bmatrix} \sigma'\left(w^k_1 \cdot x_{k-1} + b^k_1\right) & & \\ & \ddots & \\ & & \sigma'\left(w^k_{n_k} \cdot x_{k-1} + b^k_{n_k}\right) \end{bmatrix}. \end{aligned}$$

Now we assume that we already computed

$$\frac{\partial f}{\partial x_k} = \left[\frac{\partial f}{\partial x_{kj}} : j = 1, \dots, n_k\right] \in \mathbb{R}^{n_k}$$

initialized with

$$\frac{\partial f}{\partial x_d} := \sigma'\left(w^{d+1} \cdot x_d + b^{d+1}\right) w^{d+1}, \tag{4.17}$$

and use the chain rule for

$$\frac{\partial f}{\partial \widehat{w}^k_j} = \frac{\partial f}{\partial x_{kj}} \frac{\partial x_{kj}}{\partial \widehat{w}^k_j} = \left(\frac{\partial f}{\partial x_k}\right)_j \frac{\partial x_k}{\partial \widehat{w}^k_j} e_j, \qquad j = 1, \dots, n_k.$$

Since this is the jth column of the matrix $\frac{\partial f}{\partial \widehat{w}^k}$, we can group them again and obtain that

$$\begin{aligned} \frac{\partial f}{\partial \widehat{w}^k} &= \left(1_{n_{k-1}+1} \otimes \frac{\partial f}{\partial x_k}^T\right) \odot \frac{\partial x_k}{\partial \widehat{w}^k} \\ &= \left(1_{n_{k-1}+1} \otimes \frac{\partial f}{\partial x_k}^T\right) \odot \left(\begin{bmatrix} x_{k-1} \\ 1 \end{bmatrix} \otimes \sigma'\left(W^k x_{k-1} + b^k_j\right)^T\right), \end{aligned}$$

[137] Keep in mind that here we arrange the gradients $\frac{\partial x_{kj}}{\partial x_{k-1}}$ as rows of a matrix to be compatible with W^k.

hence, using Exercise 4.2,

$$\frac{\partial f}{\partial \widehat{w}^k} = \begin{bmatrix} x_{k-1} \\ 1 \end{bmatrix} \otimes \left(\frac{\partial f}{\partial x_k} \odot \sigma' \left(W^k x_{k-1} + b_j^k \right) \right)^T . \tag{4.18}$$

Exercise 4.2 Show that for arbitrary vectors $a, c \in \mathbb{R}^m$ and $b, d \in \mathbb{R}^n$ one has

$$\left(a \otimes b^T \right) \odot \left(c \otimes d^T \right) = (a \odot c) \otimes (b \odot d)^T . \tag{4.19}$$

♢

Finally, we also get[138]

$$\frac{\partial f}{\partial x_{k-1}} = \frac{\partial x_k}{\partial x_{k-1}} \frac{\partial f}{\partial x_k} = \left(D_k W^k \right)^T \frac{\partial f}{\partial x_k} = (W^k)^T \left(D_k \frac{\partial f}{\partial x_k} \right),$$

that is,

$$\frac{\partial f}{\partial x_{k-1}} = (W^k)^T \left(\sigma' \left(W^k \cdot x_{k-1} + b^k \right) \odot \frac{\partial f}{\partial x_k} \right). \tag{4.20}$$

This gives the famous **backpropagation algorithm**, the "working horse" of neural network training. In the algorithm we use the abbreviation $F_k := \frac{\partial f}{\partial x_k} \in \mathbb{R}^{n_k}$ and $W_k := \frac{\partial f}{\partial \widehat{w}_k}$.

Algorithm 4.12 (Backpropagation). **Given:** Coefficient matrices $W^0, \ldots, W^{d+1}$ and vectors $x_0, \ldots, x_d$ computed by (4.13).

1. Set $F_{d+1} = 1$.
2. For $k = d, d-1, \ldots, 0$ set
   (a) $F_k = (W^{k+1})^T \left(\sigma'(W^{k+1} x_k + b_k) \odot F_{k+1} \right)$,
   (b) $W_k = \begin{bmatrix} x_{k-1} \\ 1 \end{bmatrix} \otimes \left(\sigma'(W^k x_{k-1} + b^k) \odot F_k \right)^T$.

Result: derivatives $W_k = \dfrac{\partial f}{\partial \widehat{w}_k}$, $k = 0, \ldots, d$.

Remark 4.13. The only additional information needed in the backpropagation algorithm is the derivative σ' of the activation function. In most cases one works with explicit activation functions whose derivatives can easily be computed explicitly.

Training a neural network now means running ones favorite gradient based minimization algorithm on the differentiable functional from (4.8), using backpropagation to compute the derivatives. This, of course, means to resolve issues like finding a proper descent direction[139] and an efficient stepsize. Also it can happen that there are "flat" regions, especially when the optimal solution is not unique.

[138] With Jacobi matrices order matters in the chain rule ...

[139] Steepest descent, using $-\nabla f$ is know to fail in many occasions, see, for example [40, 47].

Remark 4.14. In our perceptron example non–uniqueness of the optimum can be seen quite easily. Suppose that we use $\lambda = 0$ and that the classes can be separated by a hyperplane with parameters ν and c and that σ satisfies $\sigma(-1) = 0$ and $\sigma(1) = 1$. Then we have that $F(w_\alpha) = 0$ for all sufficiently large α and actually also small perturbations of ν will lead to separation. Therefore, the optimum is by far not unique. This is another argument in favor of the regularization term ...

4.5 Kolmorogov's theorem

Kolmogorov's theorem is often used as another, more mathematical, justification why neural networks are so successful[140]. The starting point here is the question: "Do there exist truly multivariate functions?" The naive answer is "yes" since already $f(x, y) = x + y$ depends on both the variables x and y but then in a very simple way, namely by adding them. And addition is a simple operation. But also the seemingly more "complicated" function

$$f(x, y) = xy = e^{\log x + \log y} = \exp(\log x + \log y)$$

is only a sum, however one processed by the *univariate* function exp.

Let us begin with something completely different, namely **Nomography** which is the art of solving mathematical problems by *graphical* methods, i.e., by drawing. The simplest example used in technical applications is logarithmic

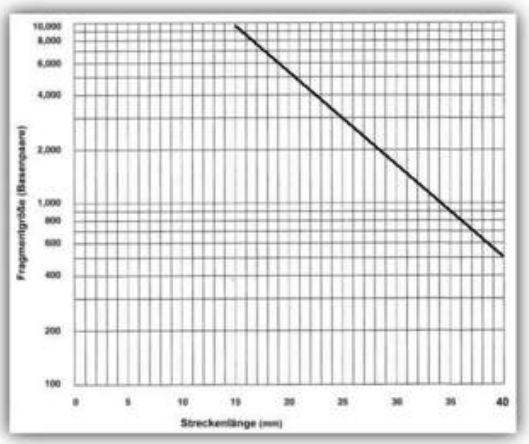


Figure 4.5: Logarithmically scaled sheet for relations of the form $y(x) = C_0 + C_1 e^{-x}$, which are used, for example, to determine the charging time of capacitors.

paper as shown in Fig. 4.5. To determine a relationship

$$y(x) = C_0 + C_1 e^{-x}$$

[140]They definitely are but it is not really clear why.

one only needs two values $y(x_1)$ und $y(x_2)$ and can then read off all other values from the line through these two values. This method has two big advantages:

- There is no need to be able to compute, it's just drawing. And dealing with more complex functions in the time before electronic calculators involved huge tables and interpolation, cf. [1].
- The result automatically has the proper *relative* accuracy and the accuracy can be improved by drawing more accurately or using more finely gridded paper.

Do not underestimate Nomography! Such graphical methods can have a very high complexity for the curve with which one tries to solve the the problem does not have to be a straight line, see Fig 4.6.

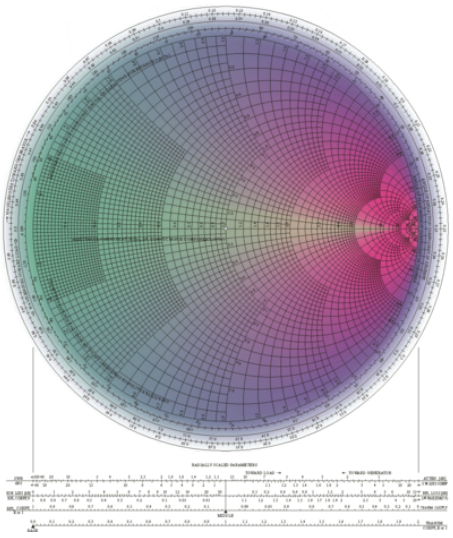


Figure 4.6: A so called **Smith chart** for the computation of the complex impedance of a transmission line. *Source:* Wikipedia.

Curve based Nomography can only work in the *bivariate* case, i.e., for functions $y = f(x)$. To use it for more complicated $y = f(x_1, \ldots, x_2)$ we would have to build it by cascading univariate functions, using only simple operations like addition. This brings us to the International Congress of Mathematicians in Paris in the year 1900 where David Hilbert gave a talk that listed the, in his opinion, 23 most important problems in mathematics. The 13th problem was as follows[141]

> Wir kommen nun zur Algebra; ich nenne im Folgenden ein Problem aus der Gleichungstheorie und eines, auf welches mich die Theorie der algebraischen Invarianten geführt hat.

[141] This quotes the original text, translate yourself.

13. Unmöglichkeit der Lösung der allgemeinen Gleichung 7ten Grades mittelst Functionen von nur 2 Argumenten.

Die Nomographie M. d'Ocagne, Traité de Nomographie, Paris 1899 hat die Aufgabe Gleichungen mittelst gezeichneter Curvenschaaren zu lösen, die von einem willkürlichen Parameter abhängen. Man sieht sofort, daß jede Wurzel einer Gleichung, deren Coefficienten nur von zwei Parametern abhängen, d. h. jede Function von zwei unabhängigen Veränderlichen auf mannigfache Weise durch das der Nomographie zu Grunde liegende Princip darstellbar ist. Ferner sind durch dieses Princip offenbar auch eine große Klasse von Functionen von drei und mehr Veränderlichen darstellbar, nämlich alle diejenigen Functionen, die man dadurch erzeugen kann, daß man zunächst eine Function von zwei Argumenten bildet, dann jedes dieser Argumente wieder gleich Functionen von zwei Argumenten einsetzt, an deren Stelle wiederum Functionen von zwei Argumenten treten u. s. f., wobei eine beliebige endliche Anzahl von Einschachtelungen der Functionen zweier Argumente gestattet ist. So gehört beispielsweise jede rationale Function von beliebig vielen Argumenten zur Klasse dieser durch nomographische Tafeln construirbaren Functionen; denn sie kann durch die Prozesse der Addition, Subtraction, Multiplikation und Division erzeugt werden, und jeder dieser Prozesse repräsentirt eine Function von nur zwei Argumenten. Man sieht leicht ein, daß auch die Wurzeln aller Gleichungen, die in einem natürlichen Rationalitätsbereiche durch Wurzelziehen auflösbar sind, zu der genannten Klasse von Functionen gehören; denn hier kommt zu den vier elementaren Rechnungsoperationen nur noch der Prozeß des Wurzelziehens hinzu, der ja lediglich eine Function eines Argumentes repräsentirt. Desgleichen sind die allgemeinen Gleichungen 5ten und 6ten Grades durch geeignete nomographische Tafeln auflösbar; denn diese können durch solche Tschirnhausentransformationen, die ihrerseits nur Ausziehen von Wurzeln verlangen, in eine Form gebracht werden, deren Coefficienten nur von zwei Parametern abhängig sind.

Wahrscheinlich ist nun die Wurzel der Gleichung 7ten Grades eine solche Function ihrer Coefficienten, die nicht zu der genannten Klasse nomographisch construirbarer Functionen gehört, d. h. die sich nicht durch eine endliche Anzahl von Einschachtelungen von Functionen zweier Argumente erzeugen läßt. Um dieses einzusehen, wäre der Nachweis dafür nötig, daß die Gleichung 7ten Grades

$$f^7 + x\,f^3 + y\,f^2 + z\,f + 1 = 0$$

nicht, mit Hülfe beliebiger stetiger Functionen von nur zwei Argumenten lösbar ist. Daß es überhaupt analytische Functionen von drei Argumenten x, y, z giebt, die nicht durch endlich-malige Verkettung von Functionen von nur zwei Argumenten erhalten werden können, davon habe ich mich, wie ich noch bemerken möchte, durch eine strenge Ueberlegung überzeugt.

It may appear strange that Hilbert mentions a problem that can be solved in two, but not in three variables, but this is not so extraordinary in multivariate

polynomials. However, Hilbert was wrong with his conjecture[142]. In 1957 Kolmogorov showed[143] in [26] that any continuous function can be represented as a **superposition** of univariate functions using exclusively addition and concatenation. This was extended and improved shortly later by his student Arnol'd in [3].

Theorem 4.15 (Kolmogorov's superposition theorem, original version)**.** *There exist* $s(2s+1)$ *continuous functions* ϕ_{jk}, $j = 0, \dots, 2s$, $k = 1, \dots, s$, *such that one can find for any function* $f \in C([0,1]^s)$ *continuous functions* $g_0, \dots, g_{2s}$, *depending on* f, *such that*

$$f(x_1, \dots, x_s) = \sum_{j=0}^{2s} g_j \left(\sum_{k=1}^{s} \phi_{jk}(x_k) \right) \tag{4.21}$$

This result was improved in the sequel, especially by G. G. Lorentz in [29] and by D. Sprecher, [54, 53, 55]. It turned out that there has to be only one function g depending on f and that the functions ϕ_{jk}, $k = 1, \dots, s$ can be chosen as multiples of some universal ϕ_j, $j = 0, \dots, 2s$, which can even be chosen as **Lipschitz continuous**[144] functions. A "final" version of the theorem, as it can also be found in [30] looks as follows.

Theorem 4.16 (**Kolmogorov's theorem** of **superposition**)**.** *There exist universal constants* $\lambda_k \in (0,1]$, $k = 1, \dots, s$, *and functions* ϕ_j, $j = 0, \dots, 2s$, *with the following properties:*

1. *The functions* ϕ_j *are Lipschitz continuous for some exponent*[145] $\alpha > 0$, *i.e., there exists a constant*[146] $C > 0$ *such that*

$$|\phi_j(x) - \phi_j(x')| \le C|x - x'|, \qquad x, x' \in [0,1], \quad j = 0, \dots, 2s.$$

   Moreover, these functions are strictly monotonically increasing.

2. *For each* $f \in C([0,1]^s)$ *there is* $g \in C([0, s])$, *such that*

$$f(x_1, \dots, x_s) = \sum_{j=0}^{2s} g\left(\lambda_1 \phi_j(x_1) + \cdots + \lambda_s \phi_j(x_s)\right). \tag{4.22}$$

Indeed, (4.22) looks like a neural network with one input layer with the universal functions ϕ_j and coefficients λ_k and an activation function g that depends on f. It seems as if everything can be represented by a *flat* neural network. However, there are some drawbacks:

[142] And he explicitly say in the last sentence that he did not (yet) prove it!

[143] This type of publication deserves a footnote of its own. The good old *Doklady* of the USSR were a publication instrument to *announce* results, usually without proof which was given later in a long publication that appeared somewhere else – or, sometimes, not at all.

[144] A function f is called **Lipschitz continuous** if $|f(x) - f(y)| \le C|x - y|$. This is significantly stronger than continuity and can be interpreted as "controllable" continuity.

[145] This is often called **Hölder continuous**, Lipschitz is then Hölder with $\alpha = 1$.

[146] These are finitely many functions, so we just take the largest constant and the smallest exponent.

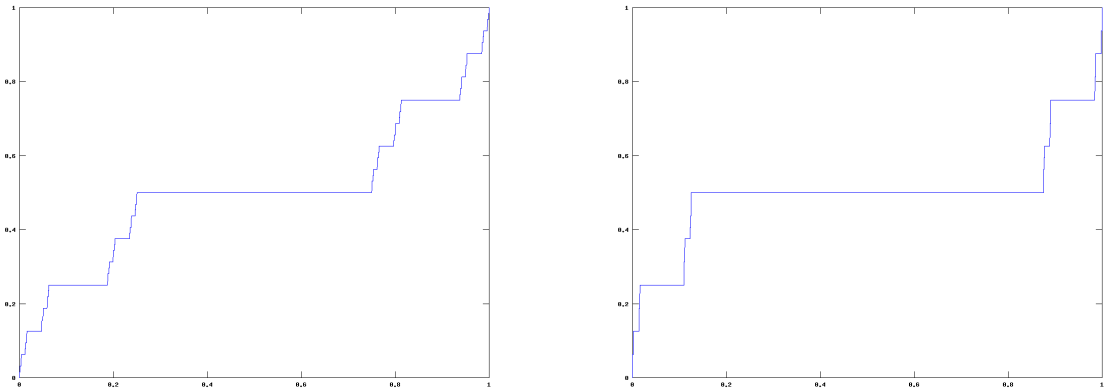


Figure 4.7: Staircase functions ϕ for a decomposition into 4 *(left)* and 8 *(right)* pieces. The fractal nature of the functions can be recognized quite well.

1. As already mentioned, the function g has to depend on f and the proof in [30] shows that g is the limit of an approximation process. On the other hand, one might consider the optimization process to be exactly this approximation, adapting g as good as possible to the finite knowledge one has about f.

2. The really difficult thing is are the universal functions ϕ_j in Theorem 4.16: they are monotonically increasing functions with $\phi(0) = 0$ and $\phi(1) = 1$ that are differentiable almost everywhere and satisfy $f'(x) = 0$ at all such points. Some functions of that sort are depicted in Fig. 4.7 and it can be seen quite well that they have a somewhat fractal behavior. In particular, the **differential quotients** satisfy

$$\sup_{x,x' \in [0,1]} \frac{\phi(x) - \phi(x')}{x - x'} = \infty.$$

3. This particularly means that there are regions of the function where very small changes of the parameter lead to significant changes of the function value which makes it tremendously unstable.

4. The proof of Theorem 4.16 also shows that the universal coefficients λ_j have to be chosen as real numbers that are linearly independent over $\mathbb{Q}$, another property that is not so easily implemented on a computer, because such numbers have an *infinite* B–adic expansion for any basis B.

In summary, even if Kolmogorov's theorem looks a lot like an argument in favor of neural networks, its components are not realizable in practice.

4.6 Convolutional networks

One of the major fashion trends in recent[147] machine learning is the rise of the **deep convolutional neural network** (**CNN**), which combines the ideas of neural networks with a classic from image processing, namely the **filter** or **convolution**.

Definition 4.17 (Signal space)**.** The s–dimensional **signal space** $\ell(\mathbb{Z}^s)$ consists of all sequences with indices in $\mathbb{Z}^s$. We interpret $c \in \ell(\mathbb{Z}^s)$ as a function $c : \mathbb{Z}^s \to \mathbb{R}$ and write $c(\alpha)$ for its value at $\alpha \in \mathbb{Z}^s$. By $\ell_0(\mathbb{Z}^s)$ we denote the subspace[148] of all sequences whose 0**–norm**[149]

$$\|c\|_0 := \#\{\alpha : c(\alpha) \neq 0\} \tag{4.23}$$

is finite.

An example of a one-dimensional signal, $d = 1$, is a sampled audio signal[150], typcial two dimensional examples are images where $\alpha \in \mathbb{Z}^s$ usually stands for an intensity value of the **pixel**.

A fundamental concept in signal processing [17, 32, 58] is the following.

Definition 4.18 (Convolution)**.** Given $f \in \ell_0(\mathbb{Z}^s)$ and $c \in \ell(\mathbb{Z}^s)$, the **convolution** $f * c \in \ell(\mathbb{Z}^s)$ of the two is defined as

$$f * c := \sum_{\alpha \in \mathbb{Z}^s} f(\alpha) c(\cdot - \alpha). \tag{4.24}$$

Remark 4.19. The requirement that the **filter**[151] f has finite support ensures that the sum in (4.24) is **locally finite** and hence the convolution is well–defined. A filter with finite support is called **FIR filter**.

The idea for convolutional neuronal networks emerges from the **scattering transform** introduced in [33] which generalizes the notion of a **filterbank**, see [58]. In a filterbank, a signal c is decomposed into n "components" by the application of filters:

$$c \;\to\; \begin{cases} f_1 * c =: c_1 \\ \vdots \\ f_n * c =: c_n \end{cases}$$

This way however, the filterbank would create a much higher amount of information or at least data than the original signal. Therefore, each of the signals is next processed by means of **downsampling** which just keeps a part of each

[147] Written in 2018, trends and fashion may change while this lecture is taught.

[148] Prove that this is really a subspace. Is it closed?

[149] Is this a norm? The answer, by the way, is "no".

[150] For example in a `wav` file.

[151] To be precise: the filter is usually the operator $F : c \mapsto f * c$ and f is called the **impulse response** fo F. Since we do not do signal processing here, we allow ourselves to be a bit generous with the terminology.

component c_k. The simplest way of downsampling is to use an **expanding dilation matrix** $M \in \mathbb{Z}^{d\times d}$ with all eigenvalues > 1 in modulus[152] an consider $c_k(M\cdot)$ instead.

Example 4.20. Setting $M = m\,I$, $m \in \mathbb{N}$, results in the usual downsampling of an image that just extracts every mth pixel.

The main idea in convolutional neural networks is now twofold: the role of the weighted average

$$w_0 + \sum_{j=1}^{n} w_j x_j$$

for the vector input x in a neuron is now taken by a convolution $f * x$ applied to the signal data x. The replacement of the activation function σ, on the other hand, is a possibly *nonlinear* function ρ that maps the signal to a signal of the same or different size; usually each layer reduces the size of the signal. The joint application of downsampling and nonlinearity is usually called **pooling** in the world of machine learning.

Example 4.21. A simple nonlinear choice, used in *denoising* of images, is to replace any subimage of size $m \times m$ by the *median* of all values in this region.

Mathematically, this idea can be represented quite simply, following [34]. The jth level in the network consists of the application of a *linear* operator W_j between two signal spaces[154], followed by the nonlinear function ρ, i.e,

$$x_j = \rho\left(W_j x_{j-1}\right), \tag{4.25}$$

in our case

$$W_j x = \left(f_j * x\right)(M_j\cdot) = \sum_{\alpha\in\mathbb{Z}^s} f_j(\alpha)\, x\left(M_j\cdot - \alpha\right), \tag{4.26}$$

where again x_{j-1} can be a *vector* of signals $x_{j-1,k}$ and the operators $W_{j,k}$ can be independent of k or depend on k. The new x_j is then again a vector of signals, maybe from another signal space. It is illustrative to write (4.26) as matrix of linear operators:

$$W_j x_{j-1} = \begin{pmatrix} W_{j,1,1} & \dots & W_{j,1,n_{j-1}} \\ \vdots & \ddots & \vdots \\ W_{j,n_j,1} & \dots & W_{j,n_j,n_{j-1}} \end{pmatrix} \begin{pmatrix} x_{j-1,1} \\ \vdots \\ x_{j-1,n_{j-1}} \end{pmatrix}, \tag{4.27}$$

and then to apply ρ componentwise. This, of course, looks very much like the setup in (4.13) and allows for a backpropagation training method where now the variables are the nonzero filter coefficients. Normally, CNNs reduce the size of the images until, at final level d, the images only consist of single pixels and then can be processed by a "normal" neural network.

Remark 4.22. If one would use a downsampling of factor 2 and the same coefficients on all levels as well as $\rho(x) = x$, one would obtain the standard wavelet decomposition by filterbanks.

[152] Alternatively one could request that M^{-1} is a [153], in other words $\|M^{-k}\| \to 0$ as $k \to \infty$.
[154] These can be images of different size.

Nur wissenschaftlich beweisen läßt es sich niemals. Und deswegen kann ich darauf keine Rücksicht nehmen.
(It just can never been proved scientifically. And therefore I cannot consider it.)

K. Laßwitz, *Aspira*

5 Unsupervised learning and clustering

So far we considered methods in **supervised learning** where the **training data** and the **labels** y_x, $x \in X$, were known. In **unsupervised learning**, on the other hand, only the data is known and the algorithm has to define the classification itself. Once the labels are determined, we can then apply our favorite learning method, for example SVM oder neural networks, to that classification and accept that or refine the classification in the case of too many misclassifications by the learning system.

The classical way to treat unsupervised learning is by first **clustering** that data and then use the labels obtained from that clustering. Clustering itself usually combines points that are close to another into a group.

5.1 K–means clustering

We begin with the classical problem of decomposing a finite set $X \subset \mathscr{X}$ into[155] K clusters $X_1, \dots, X_K$ such that

$$X = \bigcup_{j=1}^{K} X_j, \qquad X_j \cap X_k = \emptyset, \quad 1 \le j < k \le K,$$

where K is a user supplied number. As soon as $\mathscr{X}$ is a metric space, one can measure the **dissimilarity** of a cluster as

$$d(X_j) = \sum_{x,x' \in X_j} d(x, x'), \qquad j = 1, \dots, K,$$

if $\mathscr{X} = \mathbb{R}^n$ for some n, we can particularly use

$$d(X_j) = \sum_{x,x' \in X_j} \|x - x'\|_2^2, \qquad j = 1, \dots, K.$$

[155] To use K or k for the number of clusters is "historical" and gave the name to the problem and its major algorithm.

The optimal distribution of the clusters would then solve

$$\min_{X_1,\dots,X_K} \sum_{j=1}^{K} d(X_j) = \sum_{j=1}^{K} \sum_{x,x'\in X_j} \|x - x'\|_2^2. \tag{5.1}$$

Since[156]

$$\begin{aligned}
\sum_{x,x'\in X_j} \|x - x'\|_2^2 &= \sum_{x,x'\in X_j} \left(\|x\|_2^2 - 2\langle x, x'\rangle + \|x'\|_2^2\right) \\
&= 2\,\#X_j \sum_{x\in X_j} \|x\|_2^2 - 2 \sum_{x,x'\in X_j} \langle x, x'\rangle = 2\,\#X_j \left(\sum_{x\in X_j} \|x\|_2^2 - \frac{1}{\#X_j} \sum_{x,x'\in X_j} \langle x, x'\rangle\right) \\
&= 2\,\#X_j \left(\sum_{x\in X_j} \|x\|_2^2 - \frac{2}{\#X_j} \sum_{x,x'\in X_j} \langle x, x'\rangle + \frac{1}{(\#X_j)^2} \sum_{x\in X_j} \sum_{x',x''\in X_j} \langle x', x''\rangle\right) \\
&= 2\,\#X_j \sum_{x\in X_j} \left(\|x\|_2^2 - \frac{2}{\#X_j} \sum_{x'\in X_j} \langle x, x'\rangle + \frac{1}{(\#X_j)^2} \sum_{x',x''\in X_j} \langle x', x''\rangle\right) \\
&= 2\,\#X_j \sum_{x\in X_j} \left\langle x - \frac{1}{\#X_j} \sum_{x'\in X_j} x', x - \frac{1}{\#X_j} \sum_{x'\in X_j} x'\right\rangle = 2\,\#X_j \sum_{x\in X_j} \left\| x - \frac{1}{\#X_j} \sum_{x'\in X_j} x'\right\|_2^2,
\end{aligned}$$

we can as well minimize the distance from the respective **center**

$$c_j := \frac{1}{\#X_j} \sum_{x'\in X_j} x', \qquad j = 1, \dots, K,$$

of the cluster X_j which means that we can as well minimize

$$\min_{X_1,\dots,X_K} \sum_{j=1}^{K} \#X_j \sum_{x\in X_j} \|x - c_j\|_2^2. \tag{5.2}$$

This is done by an simple alternating procedure.

Algorithm 5.1 (K–means clustering)**. Given:** $X \subset \mathscr{X}$ and $K \in \mathbb{N}$.

1. Initialize c_j, $j = 1, \dots, K$, for example randomly in the convex hull $[X]$.
2. Repeat
   (a) Set
   $$X_j := \{x \in X : \|x - c_j\|_2 \le \|x - c_k\|_2,\ k \ne j\}, \qquad j = 1, \dots, K. \tag{5.3}$$
   (b) Recompute
   $$c_j := \frac{1}{\#X_j} \sum_{x'\in X_j} x', \qquad j = 1, \dots, K. \tag{5.4}$$

[156] And this nice identity holds in any Hilbert space.

until the value of (5.3) does not change any more.

Note that by Lemma 1.23 the step (5.4) could be seen as minimizing (5.2) for fixed clusters with respect to the variables c_j while (5.3) minimizes with respect to X_j for fixed centers. In this respect, Algorithm 5.1 is one of the heuristic alternating optimization algorithms for the problem

$$\min_{X_1,\dots,X_K} \min_{c_1,\dots,c_K} \sum_{j=1}^{K} \#X_j \sum_{x\in X_j} \|x - c_j\|_2^2. \tag{5.5}$$

Since in each step such algorithms reduce the target function which in turn is bounded from below by zero, this process trivially guarantees at least convergence of the values of the target function (5.2). However, here more is true.

Theorem 5.2. *Algorithm 5.1 terminates after finitely many steps.*

Proof: There are $\binom{\#X+K-1}{K}$ possible partitions which means that the target function can only be improved for finitely many times. Since the partition determines the centers, the expression (5.2) depends only on the partition. □

The K–means Algorithm is known to work quite well if the clusters are ball–shaped and uniform, but in many cases it can get stuck in a local minimum, even if the stopping criterion

if the partition in some step has been encountered before[157]

is used.

Remark 5.3 (K–means clustering)**.**

1. It is well–known that the K–means algorithm is very sensitive to initialization and there are heuristic strategies to do as good a job as possible there.

2. It could even happen during the algorithm that, because of poor choice of centers, one gets $X_j = \emptyset$ for some j. Also this should be taken into account and handled appropriately. For example, one could give the element with largest dissipation, i.e., the solution of

$$\max_{x\in X} \min_{j=1,\dots,k} \|x - c_j\|_2$$

   a cluster "of its own".

3. Generally, K–means clustering is, like many combinatorial problems, an **NP hard** problem.

4. The algorithm does not provide so called **certificates** for optimality, i.e., a number or a condition from which one can read off that an optimum is reached or not when the algorithm becomes stationary.

[157] If this happens, the algorithm would run in a loop and the target function cannot be improved any further by the algorithm.

The classical application of K–means clustering is **quantization**, the task of reducing a large number of measurements to a small number of values in an efficient way. This is a major issue in image processing.

Example 5.4 (RGB color quantization)**.** Suppose a picture with high resolution RGB[158] has to be encoded with a **color table** of only 256 colors which happens for example in the GIF graphics format and also was used in earlier PC graphics cards. Nevertheless, each value in the table can be a high definition color description which makes the quantization to 256 "best" colors exactly a problem of K–means clustering with $K = 256$.

Because of its discrete structure, the K–means problem cannot be accessed by continuous methods like descent algorithms. To facilitate that, one reformulates the modified problem of average blockwise dissipation

$$\min_{X_1,\dots,X_k} \sum_{j=1}^{K} \sum_{x\in X_j} \|x - c_j\|_2^2 \qquad \Leftrightarrow \qquad \min_{X_1,\dots,X_k} \sum_{j=1}^{K} \frac{1}{\#X_j} \sum_{x,x'\in X_j} \|x - x'\|_2^2 \tag{5.6}$$

in a continuous way for which we need some additional terminology.

Definition 5.5 (Frobenius norm)**.** The **Frobenius norm** of a matrix $A \in \mathbb{R}^{m\times n}$, defined as

$$\|A\|_F := \sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk}^2, \tag{5.7}$$

is the norm of the **Hilbert space** of all $m \times n$ matrices equipped with the inner product

$$\langle A, B\rangle = \sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk}\, b_{jk}. \tag{5.8}$$

Note that the inner product can be rewritten for $A, B \in \mathbb{R}^{m\times n}$ as

$$\begin{aligned} \langle A, B\rangle &= \sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk} b_{jk} = \sum_{\ell=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{n} \delta_{k\ell} a_{jk} b_{j\ell} \\ &= \sum_{k,\ell=1}^{n} \delta_{k\ell} \underbrace{\sum_{j=1}^{m} a_{jk} b_{j\ell}}_{=(A^T B)_{k\ell}} = \sum_{k=1}^{n} (A^T B)_{kk} = \operatorname{trace}(A^T B) \end{aligned}$$

where the **trace** of a square matrix is defined as the sum of its diagonal elements,

$$\operatorname{trace}(A) = \sum_{j=1}^{n} a_{jj}, \qquad A \in \mathbb{R}^{n\times n}. \tag{5.9}$$

Therefore, we can rewrite the target function $\langle E_X, Z\rangle$ also as $\operatorname{trace}(E_X^T Z)$ which is, in fact, done quite frequently.

[158] Red, Green, Blue, one of the standard color models; any color model is the same here, more about color models and transformation matrices between them can, for example, be found in [12].

Definition 5.6 (Distance matrix)**.** The Euclidean **distance matrix** E_X for a finite set $X \subset \mathscr{X}$ is defined as

$$E_X := \left[\|x - x'\|_2^2 : x, x' \in X \right] \in \mathbb{R}^{\#X \times \#X}. \tag{5.10}$$

With this terminology we can express the target function of (5.6) as

$$\langle E_X, Z \rangle, \qquad Z = \sum_{j=1}^{K} \frac{1}{\#X_j} \chi(X_j)\, \chi(X_j)^T,$$

with the discrete **characteristic function**

$$\mathbb{R}^X \ni (\chi(X_j))_x = \chi(X_j)(x) = \begin{cases} 1, & x \in X_j, \\ 0, & x \notin X_j, \end{cases} \qquad j = 1, \ldots, K.$$

Due to this definition it follows immediately that

$$\left(\chi(X_j)\, \chi(X_j)^T\right)_{x,x'} = \chi(X_j)(x)\, \chi(X_j)(x') = \begin{cases} 1, & x, x' \in X_j \\ 0, & \text{otherwise}, \end{cases} \qquad x, x' \in X.$$

The **partition matrix** Z has some properties. First,

$$Z1 = \sum_{j=1}^{K} \frac{1}{\#X_j} \chi(X_j) \underbrace{\chi(X_j)^T 1}_{=\#X_j} = \sum_{j=1}^{K} \chi(X_j) = 1,$$

which is a simple **linear constraint**. Moreover, we have that

$$Z\chi(X_k) = \sum_{j=1}^{K} \frac{1}{\#X_j} \chi(X_j) \underbrace{\chi(X_j)^T \chi(X_k)}_{=\#X_j\, \delta_{jk}} = \chi(X_k), \qquad k = 1, \ldots, K,$$

hence Z is a rank K matrix with the K–fold **eigenvalue** 1. In addition and obviously, $Z^T = Z$ and $Z \geq 0$ in the sense that $z_{x,x'} \geq 0$, $x, x' \in X$. Another property is

$$\begin{aligned} Z^T Z &= Z^2 = \sum_{j=1}^{K} \frac{1}{\#X_j} \chi(X_j)\, \chi(X_j)^T \sum_{k=1}^{K} \frac{1}{\#X_k} \chi(X_k)\, \chi(X_k)^T \\ &= \sum_{j,k=1}^{K} \frac{1}{\#X_j\, \#X_k} \chi(X_j) \underbrace{\chi(X_j)^T \chi(X_k)}_{=\delta_{jk}\#X_j} \chi(X_k)^T = \sum_{j=1}^{K} \frac{1}{\#X_j} \chi(X_j)\, \chi(X_j)^T = Z, \end{aligned}$$

hence Z is a **projection** and since

$$y^T Z y = \sum_{j=1}^{K} \frac{1}{\#X_j} \underbrace{\left(y^T \chi(X_j)\right)^2}_{\geq 0} \geq 0,$$

the matrix is also **positive semidefinite**.

These *nondiscrete* properties allow us to reformulate the clustering problem in various ways and then to *relax* i.e., to drop some of the constraints and to optimize over a larger feasible set.

In this way, we can be begin to reformulate of the optimization problem (5.6), for example as

$$\min \langle E_X, Z\rangle \qquad \text{subject to } Z^T = Z, \quad \begin{array}{ll} Z1 = 1, & \lambda_1(Z) = \cdots = \lambda_K(Z) = 1, \\ Z \geq 0, & \lambda_{K+1}(Z) = \cdots = \lambda_{\#X}(Z) = 0. \end{array} \tag{5.11}$$

We first note that we can drop the symmetry request: if Z is a minimizer of (5.11), then

$$\left\langle E_X, Z^T \right\rangle = \left\langle E_X^T, Z \right\rangle = \langle E_X, Z\rangle \qquad \Rightarrow \qquad \left\langle E_X, \frac{1}{2}(Z + Z^T) \right\rangle = \langle E_X, Z\rangle,$$

hence there always exists a symmetric minimizer, provided that we also require $Z^T 1 = 1$, which is again only a linear constraint. This leads to

$$\min \langle E_X, Z\rangle \qquad \text{subject to } \begin{array}{ll} Z1 = Z^T 1 = 1, & \lambda_1(Z) = \cdots = \lambda_K(Z) = 1, \\ Z \geq 0, & \lambda_{K+1}(Z) = \cdots = \lambda_{\#X}(Z) = 0. \end{array} \tag{5.12}$$

This involves a class of well–known matrices that are useful and well–studied, not only in the world of probability.

Definition 5.7. A square matrix $Z \in \mathbb{R}^{n\times n}$ with $Z \geq 0$ and $Z1_n = Z^T 1_n = 1_n$ is called **doubly stochastic**.

Relaxations and variants of the above problems are considered, for example in [41, 4]. However, the solutions of the relaxed problems may not be partition matrices any more, and there are usually some conditions but fortunately there is another reformulation with a relatively simple converse result. Here we consider partition matrices of the form

$$Y := \sum_{j=1}^{K} \frac{1}{\sqrt{\#X_j}} \chi(X_j) e_j^T \in \mathbb{R}^{\#X\times K}$$

where we write the characteristic vectors as columns of a matrix Y. We have, in the usual manner,

$$\mathbb{R}^{K\times K} \ni Y^T Y = \sum_{j,k=1}^{K} \frac{1}{\sqrt{\#X_j\,\#X_k}} e_j \underbrace{\chi(X_j)^T \chi(X_k)}_{=\delta_{jk}\#X_j} e_k^T = \sum_{j=1}^{K} e_j e_k^T = I,$$

and

$$\mathbb{R}^{\#X\times\#X} \ni YY^T = \sum_{j,k=1}^{K} \frac{1}{\sqrt{\#X_j\,\#X_k}} \chi(X_j) \underbrace{e_j^T e_k}_{=\delta_{jk}} \chi(X_k)^T = \sum_{j=1}^{K} \frac{1}{\#X_j} \chi(X_j)\chi(X_j)^T = Z,$$

hence $YY^T 1 = Z1 = 1$, as well as $Y \geq 0$.

We can also describe the problem by means of Y taking into account that, recalling the computations that related (5.1) and (5.2),

$$\frac{1}{2\,\#X_j}\sum_{x,x'\in X_j}\|x-x'\|_2^2=\sum_{x\in X_j}\|x\|_2^2-\frac{1}{\#X_j}\sum_{x,x'\in X_j}\langle x,x'\rangle$$

and that

$$\sum_{x,x'\in X_j}\langle x,x'\rangle=\chi(X_j)^T\,G_X\,\chi(X_j),\qquad G_X=[\langle x,x'\rangle : x,x'\in X]\,.$$

Therefore,

$$\begin{bmatrix}\frac{\chi(X_1)^T\,G_X\,\chi(X_1)}{\#X_1} & & \\ & \ddots & \\ & & \frac{\chi(X_K)^T\,G_X\,\chi(X_K)}{\#X_K}\end{bmatrix} = \sum_{j=1}^{K}\frac{1}{\#X_j}e_je_j^T\chi(X_j)^T\,G_X\,\chi(X_j)$$
$$= Y^TG_XY,$$

so that we can express the summed **average dissimilarity** of the clusters as

$$\frac{1}{2}\sum_{j=1}^{K}\frac{1}{\#X_j}\sum_{x,x'\in X_j}\|x-x'\|_2^2=\sum_{x\in X}\|x\|_2^2-\operatorname{trace}(Y^TGY)$$

and minimizing the target functions in (5.6) is equivalent to

$$\max_Y \operatorname{trace}(Y^TG_XY)\qquad\text{subject to } Y\geq 0,\ \begin{matrix}Y^TY &=& I,\\ YY^T1 &=& 1.\end{matrix}\tag{5.13}$$

This is a continuous formulation but the constraints in (5.13) are chosen such that only partition matrices are feasible.

Lemma 5.8. *If $A\in\mathbb{R}^{X\times K}$ is a matrix such that $A\geq 0$, $A^TA=I$ and $AA^T1=1$, then A is a partition matrix.*

Proof: The property $A^TA=I$ means that the columns $a_j=[a_{x,j}:x\in X]$ of $A=[a_1,\ldots,a_K]$ are orthogonal and since $A\geq 0$ the property

$$0=a_j^Ta_k,\qquad j\neq k$$

can only be obtained if

$$(a_j)_x>0\qquad\Rightarrow\qquad (a_k)_x=0,\quad j\neq k.$$

Consequently, any row of A contains at most one nonzero element. Now

$$AA^T1 = \left(\sum_{j,k=1}^{K}a_je_j^Te_ka_k^T\right)1=\left(\sum_{j=1}^{K}a_ja_j^T\right)1=\sum_{j=1}^{K}(a_j^T1)a_j,$$

and since the supports of the a_j are disjoint, we have that for any $x \in X$

$$1 = (AA^T 1)_x = \sum_{j=1}^{K} (a_j^T 1) a_{jx} = (a_{j(x)}^T 1) a_{j(x)},$$

if and only if

$$a_{jx} > 0 \qquad \Rightarrow a_{jx} = \frac{1}{a_j^T 1},$$

hence the weight is uniform for all points that belong to the jth cluster. □

Theorem 5.9. *The optimization problems* (5.6) *and* (5.13) *are equivalent.*

5.2 Spectral clustering

This next clustering method was originally developed for graph theoretic considerations but is nowadays used as a good initializer for K–means clustering; more precisely, K–means clustering is even included as a "second step" in spectral clustering methods.

Definition 5.10 (Graphs)**.**

1. A **graph** G consists of a set V of vertices and $E \subset V \times V$ of edges. An **edge** $e = (v, v') \in E$ connects the vertex v with the vertex v'.

2. An **undirected graph** is characterized by a **symmetric** edge set E, i.e., $(v, v') \in E$ iff $(v', v) \in E$.

3. A **complete graph** is a graph where $E = V \times V$.

4. An **adjacency matrix** $W = [w_{v,v'} : v, v' \in V] \in \mathbb{R}_+^{V \times V}$ defines nonnegative weights for the edges and has to satisfy $w_{v,v'} > 0$ if $(v, v') \in E$ and $w_{v,v'} = 0$ if $(v, v') \notin E$.

The simplest adjacency matrix is given by the **incidence matrix**

$$w_{v,v'} = \begin{cases} 1, & (v, v') \in E, \\ 0, & (v, v') \notin E, \end{cases} \qquad v, v' \in V.$$

The idea is to consider the complete undirected graph with vertex set X and then to decompose it into subgraphs. As adjacency matrix one uses

$$W = E_X, \qquad \text{i.e.} \qquad w_{x,x'} = \|x - x'\|_2^2, \quad x, x' \in X, \tag{5.14}$$

or more generally

$$w_{x,x'} = f\left(\|x - x'\|_2^2\right), \qquad x, x' \in X, \tag{5.15}$$

where $f : \mathbb{R}_+ \to \mathbb{R}_+$ is a strictly monotonic function like $f(x) = e^x$ or e^{-x}.

The next step is to do analysis on the graph. To that end, we recall the **Laplace operator** or **Laplacian**, defined as

$$\Delta f = \sum_{j=1}^{s} \frac{\partial^2}{\partial x_j^2} f, \qquad f \in C^2(\mathbb{R}^s),$$

which is the most investigated differential operator and the prototype of so called **elliptic differential operators**, cf. [13, 14]. If f is only defined on the grid $\mathbb{Z}^s$, the (centered) **discrete Laplacian** is defined as[159]

$$\Delta f = \frac{1}{s}\left(\sum_{j=1}^{s} f(\cdot + \epsilon_j) - s\, f(\cdot) \right).$$

The bivariate discrete Laplacian is used frequently in image processing as a technique for edge detection, cf. [18, 45]. This concepts is now extended to graphs.

Definition 5.11 (Graph Laplacian)**.** For an **undirected graph** $G = (V, E)$ with **adjacency matrix**[160] W we define

1. the **degree matrix**

$$D := \operatorname{diag}\left[d_v : v \in V \right], \qquad d_v := \sum_{v' \in V} w_{v,v'}. \tag{5.16}$$

2. the **graph Laplacian**

$$L = \Delta G = D - W \in \mathbb{R}^{V \times V}. \tag{5.17}$$

Proposition 5.12. *The graph Laplacian satisfies*

$$y^T L y = \frac{1}{2} \sum_{v,v' \in V} w_{v,v'} \left(y_v - y_{v'} \right)^2, \qquad y \in \mathbb{R}^V \tag{5.18}$$

and therefore

1. L *is symmetric positive semidefinite matrix,*
2. L *has the smallest eigenvalue* 0 *with eigenvector* 1*,*
3. L *has nonnegative eigenvalues* $0 = \lambda_1 \le \cdots \le \lambda_{\#V}$*.*

[159] In one variable, this is the **divided difference**

$$[k-1, k, k+1]f = \frac{f(k-1) - 2f(k) + f(k+1)}{2}.$$

[160] Keep in mind that W is influenced by E.

Proof: We verify (5.18) by computing

$$\begin{aligned} y^T L y &= y^T D y - y^T W y = \sum_{v \in V} d_v y_v^2 - \sum_{v,v' \in V} w_{v,v'} y_v y_{v'} \\ &= \sum_{v,v' \in V} w_{v,v'} y_v^2 - \sum_{v,v' \in V} w_{v,v'} y_v y_{v'} \\ &= \frac{1}{2} \sum_{v,v' \in V} w_{v,v'} y_v^2 + \frac{1}{2} \sum_{v,v' \in V} w_{v,v'} y_{v'}^2 - \sum_{v,v' \in V} w_{v,v'} y_v y_{v'} \\ &= \frac{1}{2} \sum_{v,v' \in V} w_{v,v'} \left(y_v^2 - 2 y_v y_{v'} + y_{v'}^2 \right) = \frac{1}{2} \sum_{v,v' \in V} w_{v,v'} \left(y_v - y_{v'} \right)^2, \end{aligned}$$

where we used the definition (5.16) and the fact that G is undirected. From (5.18) the properties 1) and 3) follow directly while for 2) we note that

$$(L1)_v = d_v - \sum_{v' \in V} w_{v,v'} = 0,$$

which shows that $L1 = 0$. □

Proposition 5.13. *The number of connected components of an undirected graph equals the number of its zero eigenvalues. The eigenspace for 0 is spanned by the characteristic vectors of these components.*

Proof: The proof uses induction on k, the number of connected components.

If $k = 1$, i.e., we have a connected graph, we have to show that 1 is the only eigenvector for the eigenvalue 0. To that end, assume that y is an eigenvector for 0, then

$$0 = y^T \underbrace{L y}_{=0} = \frac{1}{2} \sum_{v,v' \in V} w_{v,v'} \left(y_v - y_{v'} \right)^2$$

which can only vanish if all the summands equal zero. Since f is connected, we have that $w_{v,v'} > 0$, $v, v' \in V$, hence $y_v = y_{v'}$, $v, v' \in V$, and therefore $y = \lambda 1$ for some $\lambda \neq 0$.

Now suppose that $k > 1$ and that $V_1, \ldots, V_k$ are the connected components. By applying a suitable permutation P that orders V in the order $V_1, \ldots, V_k$, and taking into account that D is diagonal, we get that

$$P^T W P = \begin{bmatrix} W_1 & & \\ & \ddots & \\ & & W_k \end{bmatrix} \quad \Rightarrow \quad P^T L P = \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{bmatrix},$$

with $W_j, L_j \in \mathbb{R}^{V_j \times V_j}$, where each L_j is the graph Laplacian for the connected subgraph V_j, hence $L_j 1 = 0$ and 1 is the only zero eigenvalue for L_j as we showed in the first part of the proof. Consequently, the eigenvectors of L for the

eigenvalue 0 are precisely

$$P^T \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1_{L_j} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \chi(V_j), \qquad j = 1, \ldots, k,$$

which completes the proof. □

In the same way as we can sum over dissimilarity or weighted dissimilarity, there is also the concept of a **normalized graph Laplacian**, where one considers

$$L := I - D^{-1/2} W D^{-1/2} \qquad \text{or} \qquad L := I - D^{-1} W,$$

where the version on the left hand side has the advantage of being symmetric.

Spectral clustering again starts with the data $X \in \mathbb{R}^n$ and builds a **similarity graph** from the data. There are different ways to set up such a similarity graph:

1. **neighborhood graph**: for given $\varepsilon > 0$ one sets

   $$(x, x') \in E \qquad \Leftrightarrow \qquad \|x - x'\|_2 \leq \varepsilon,$$

   and extend the edge set symmetrically.

2. **nearest neighbor**: for a given index k one finds the k nearest neighbors of x, i.e.

   $$N_k(x) := \{x' \in X : \#\{x'' \in X : \|x - x''\|_2 \leq \|x - x'\|_2\} \leq k\}$$

   and sets

   $$(x, x') \in E \qquad \Leftrightarrow \qquad x' \in N_{k,x}(x).$$

   Also this graph is then extended symmetrically[161].

3. **connected graph**: one simply sets $E = X \times X$.

Next, the weight matrix is set up according to distance. Since small distance corresponds to *large* similarity and since a cluster should contain similar objects, it is reasonable to use

$$w_{x,x'} = e^{-\|x-x'\|_2^2/\sigma}, \qquad \sigma > 0, \tag{5.19}$$

where the parameter σ controls the decay of the weights or

$$w_{x,x'} = \left(\alpha + \|x - x'\|_2^2\right)^{-\beta}, \qquad \alpha, \beta > 0. \tag{5.20}$$

[161] "Nearest neighbor" is *not* a symmetric relationship.

Having set up the adjacency matrix one can compute the graph Laplacian and its first[162] K eigenvalues $\lambda_1, \dots, \lambda_K$ with eigenvectors $w_1, \dots, w_K$. If the graph would decompose into K subgraphs, which would be the perfect clustering[163], then these eigenvalues would be of the form $\chi(X_j)$ for the clusters indicated by the graph decomposition. That means that the matrix matrix

$$[v_1, \dots, v_K] = [\chi(X_1), \dots, \chi(X_K)] \in \mathbb{R}^{X \times K}$$

has only one nonzero entry of value one in each row and this entry tells us to which cluster the element of X that indexes this row belongs.

Normally, however, the graph Laplacian will only zero as single eigenvalue, but nevertheless one applies the same strategy and tries to use it in a best possible way.

Algorithm 5.14 (Spectral clustering)**.**
Given: $X \subset \mathscr{X}$ and the number K of clusters.

1. Build a similarity graph G and an adjacency matrix W based on X to form a **graph Laplacian** $L \in \mathbb{R}^{X \times X}$.
2. Compute the K smallest eigenvalues[164] of W and the associated eigenvectors $v_1, \dots, v_K \in \mathbb{R}^X$.
3. Consider the rows $y_x \in \mathbb{R}^K$, $x \in X$, of the matrix $V = [v_1, \dots, v_K]$ and cluster them in $\mathbb{R}^K$ with a **K–means clustering** algorithm into clusters $Y_1, \dots, Y_K$.
4. Define clusters $X_1, \dots, X_K$ by

$$X_j := \{x \in X : y_x \in Y_j\}, \qquad j = 1, \dots, K.$$

Result: Decomposition $X = X_1 \cup \dots \cup X_K$ with $X_j \cap X_k = \emptyset$, $j \neq k$.

There is much more to spectral clustering than the above heuristical argument that shows that if the graph decomposes properly then the spectral clustering finds this decomposition. In fact, spectral clustering is closely related to **graph cutting**, see e.g. [28] but that would, unfortunately, lead to far for this lecture here.

5.3 Dimension reduction by PCA

One of the most classical methods to extract features or important components from data is the so called **principal component analysis** or **PCA** for short. It tries to eliminate linear relations between the data.

Example 5.15. Suppose we have data in $\mathbb{R}^n$ but all this data lies in the x_1–x_2–plane, that is, all points are of the form $x = (x_1, x_2, 0, \dots, 0)$. This should be detected before clustering the points. Of course, this should work for any lower dimensional linear space.

[162] Smallest.
[163] For example because by accident the ε had been chosen perfectly in a neighborhood graph
[164] This is a standard though nontrivial task of Numerical Linear Algebra, cf. [15].

The starting point is data $X \subset \mathbb{R}^n$ that is grouped into a matrix

$$X = \begin{bmatrix} x_j : & \begin{matrix} x \in X \\ j = 1, \ldots, n \end{matrix} \end{bmatrix} \in \mathbb{R}^{X \times n}.$$

The main mathematical concept behind the PCA is a classical matrix decomposition from Numerical Linear Algebra, see [15, 20, 46].

Theorem 5.16 (SVD). *For any matrix $A \in \mathbb{R}^{m \times n}$ there exist orthogonal matrices*[165] *$U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ and a **diagonal matrix***[166] *$\Sigma \in \mathbb{R}^{m \times n}$ such that*

$$A = U \Sigma V^T, \qquad \sigma_{11} \geq \cdots \sigma_{kk} \geq 0, \quad k := \min(m, n). \tag{5.21}$$

Definition 5.17 (SVD)**.** The decomposition (5.21) is called **singular value decomposition** or **SVD** of A. The numbers $\sigma_j = \sigma_{jj}$, $j = 1, \ldots, \min(m, n)$, are called the **singular values** of A and the columns[167] of U and V are called left or right **singular vectors** respectively.

Remark 5.18. There are several properties of the **SVD** that make it an important tool in any sort of matrix computations.

1. There exist efficient algorithms to compute the SVD, see [15] and this is much easier than determining all the eigenvalues of the matrix A. `Matlab` contains a function `svd` for that purpose.

2. The **rank** of the matrix can be read off the SVD as the number of positive singular values: if

   $$\sigma_1 \geq \cdots \geq \sigma_k > \sigma_{k+1} = \cdots = \sigma_{\min(m,n)} = 0,$$

   then the rank of A is k. This can also be used for *numerical* rank detection of a matrix by thresholding the singular values.

3. Since the **Frobenius norm** of a matrix A is **orthogonally invariant** it follows immediately that

   $$\|A\|_F^2 = \|U^T A V\|_2^2 = \|\Sigma\|_2^2 = \operatorname{trace}(\Sigma^T \Sigma) = \sum_{j=1}^{\min(m,n)} \sigma_j^2. \tag{5.22}$$

4. The Euclidean **operator norm**

   $$\|A\|_2 := \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2 = 1} \|Ax\|_2 = \max_{\|x\|_2 = 1} \sqrt{x^T A^T A x}$$

[165] A square matrix A is called an **orthogonal matrix** if $A^T A = A A^T = I$, i.e., if its row and column vectors are **orthonormal**, respectively.

[166] A nonsquare matrix A is called *diagonal* if $a_{jk} = 0$ whenever $j \neq k$.

[167] This is the reason for the "V^T" in (5.21): we can then speak of the columns in both cases.

is also orthogonally invariant:

$$\|QAQ'\|_2^2 = \max_{\|x\|_2=1} \|QAQ'x\|_2^2 = \max_{\|Q'x\|_2=1} x^T A \underbrace{Q^T Q}_{=I} Ax = \max_{\|x\|_2=1} \|Ax\|_2^2 = \|A\|_2^2.$$

Therefore,

$$\|A\|_2 = \|U\Sigma V^T\|_2 = \|\Sigma\|_2 = \max_{j=1,\dots,\min(m,n)} \sigma_j = \sigma_1.$$

Rewriting (5.21) with respect to the rows and columns of U and V, we can rephrase Theorem 5.16 in the following way.

Corollary 5.19 (SVD revisited). *For any matrix $A \in \mathbb{R}^{m\times n}$ of **rank** $r = r(A)$ there exist orthonormal vectors $u_1, \dots, u_r \in \mathbb{R}^m$ and $v_1, \dots, v_r \in \mathbb{R}^n$ as well as positive real numbers $\sigma_1 \geq \cdots \geq \sigma_r > 0$ such that*

$$A = \sum_{j=1}^{r(A)} \sigma_j u_j v_j^T. \tag{5.23}$$

The SVD plays a fundamental role in the problem of approximating linear subspaces by linear subspaces of lower dimension. The first concrete question is how to determine so called best fit subspaces that explore the range of a matrix in a best possible way or, in other words, find the dominant lower dimensional subspace of the space defined by the columns of the matrix. And the answer is the SVD.

Definition 5.20. A subspace $W \subset \mathbb{R}^n$ of dimension $k \leq n$ is called a **best fit subspace** of dimension k for $A \in \mathbb{R}^{m\times n}$ if it solves

$$\max_{W=[w_1,\dots,w_k]} \sum_{j=1}^{k} \|Aw_j\|_2^2, \qquad \text{subject to } W^TW = I. \tag{5.24}$$

The constraint in (5.24) requests that the w_j form an **orthonormal basis** for the k–dimensional subspace $W\mathbb{R}^k$.

Proposition 5.21. *The best fit subspace of dimension $k \leq r(A)$ for $A = U\Sigma V^T$ is $V_k = [v_1, \dots, v_k]$ and it is unique if $\sigma_k > \sigma_{k+1}$.*

Proof: The first observation is that

$$\sum_{j=1}^{k} \|Aw_j\|_2^2 = \sum_{j=1}^{k} \|AWe_j\|_2^2 = \sum_{j=1}^{k} e_j^T \underbrace{W^TA^TAW}_{\in\mathbb{R}^{k\times k}} e_j = \operatorname{trace}(W^TA^TAW),$$

that is

$$\sum_{j=1}^{k} \|Aw_j\|_2^2 = \|AW\|_F^2. \tag{5.25}$$

Using the SVD, we next find that

$$\|Aw_j\|_2^2 = \|U\Sigma V^T w_j\|_2^2 = w_j^T V\Sigma \underbrace{U^T U}_{=I} \Sigma V^T w_j\|_2^2 = w_j^T V\Sigma^2 V^T w_j = \|\Sigma V^T w_j\|_2^2$$

i.e.,

$$\|Aw_j\|_2^2 = \sum_{\ell=1}^{r(A)} \sigma_\ell^2 \left(v_\ell^T w_j\right)^2 \tag{5.26}$$

where

$$1 = \|w_j\|_2^2 = w_j^T \underbrace{VV^T}_{=I} w_j = \|V^T w_j\|_2^2 = \sum_{\ell=1}^{n} \left(v_\ell^T w_j\right)^2.$$

After these preliminaries, we use induction on k. In the case $k = 1$ we have to optimize (5.26) and since

$$\|Aw\|_2^2 = \sum_{\ell=1}^{r(A)} \underbrace{\sigma_\ell^2}_{\le\sigma_1} \left(v_\ell^T w\right)^2 \le \sigma_1 \underbrace{\sum_{\ell=1}^{r(A)} \left(v_\ell^T w\right)^2}_{=1} = \sigma_1 = \|Av_1\|_2^2,$$

the vector v_1 is a minimizer and it is unique iff $\sigma_1 > \sigma_2$. For the induction step $k \to k+1$, $k \ge 1$, we suppose that $W \in \mathbb{R}^{m\times k+1}$ maximizes (5.25). Since the homogeneous linear system

$$0 = \underbrace{V_k^T W}_{\in\mathbb{R}^{k\times k+1}} x, \qquad x \in \mathbb{R}^{k+1}$$

has at least one nontrivial solution, $W \cap V_k^\perp \neq \{0\}$ and we can assume that $w_{k+1} \perp V_k$. Then, be the induction hypothesis

$$\|AW\|_F^2 = \sum_{j=1}^{k} \|Aw_j\|_2^2 + \|Aw_{k+1}\|_2^2 \le \sum_{j=1}^{k} \|Av_j\|_2^2 + \|Aw_{k+1}\|_2^2,$$

and $[V\, w_{k+1}]$ performs as least as good as W and, since $v_j^T w_{k+1} = 0$, $j = 1, \ldots, k$,

$$\|AW\|_F^2 = \sum_{j=1}^{k} \sigma_j^2 + \sum_{\ell=k+1}^{r(A)} \underbrace{\sigma_\ell^2}_{\le\sigma_{k+1}} \left(v_\ell^T w_{k+1}\right)^2$$

which is maximized for $w_{k+1} = v_{k+1}$. Uniqueness[168] follows like above. □

Theorem 5.22 (Best approximation of lower rank). *Let* $A = U\Sigma V^T \in \mathbb{R}^{m\times n}$ *and* $k \le \min(m, n)$ *be given. Then the solution of the problem*

$$\min_B \|A - B\|_2 \qquad \textit{subject to } \operatorname{rank} B = k, \tag{5.27}$$

[168] In terms of subspaces, not in term of matrices!

is given by

$$B = A_k := U\Sigma_k V^T := U \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} V^T. \tag{5.28}$$

The solution of (5.27) *is unique iff* $\sigma_k > \sigma_{k+1}$ *and the error is*

$$\|A - B\|_2 = \sigma_{k+1}. \tag{5.29}$$

Proof: The proof is taken from [15]. We can restrict ourselves to $k < r(A)$ as otherwise $B := A$ is the obvious solution of (5.27).

We first note that due to the orthogonal invariance of the Euclidean norm the matrix A_k from (5.28) satisfies

$$\|A - A_k\|_2^2 = \|U(\Sigma - \Sigma_k)V^T\|_2^2 = \|\Sigma - \Sigma_k\|_2^2 = \max_{j>k} \sigma_j^2 = \sigma_{k+1}^2.$$

Now let $B \in \mathbb{R}^{m\times n}$ be any matrix of rank $k < r(A)$. This means[169] that there exist orthonormal vectors $x_1, \ldots, x_{n-k}$ such that $Bx_j = 0$. Since $k < r(A)$ we have $\sigma_1 \geq \cdots \geq \sigma_{k+1} > 0$ and

$$\text{span}\,\{x_1, \ldots, x_{n-k}\} \cap \text{span}\,\{v_1, \ldots, v_{k+1}\} \neq \{0\}$$

since otherwise the two bases in $\mathbb{R}^n$ would span an $n+1$–dimensional space. Hence, there exists

$$z \in \text{span}\,\{x_1, \ldots, x_{n-k}\} \cap \text{span}\,\{v_1, \ldots, v_{k+1}\}, \qquad \|z\|_2 = 1.$$

Since $v_{k+2}, \ldots, v_{r(A)} \in V_{k+1}^{\perp}$, we have that

$$Az = \sum_{j=1}^{r(A)} \sigma_j u_j v_j^T z = \sum_{j=1}^{k+1} \sigma_j u_j v_j^T z$$

it follows that

$$\begin{aligned} \|A - B\|_2^2 &\geq \|(A-B)z\|_2^2 = \|Az\|_2^2 = (Az)^T Az = \left(\sum_{j=1}^{k+1} \sigma_j u_j v_j^T z\right)\left(\sum_{\ell=1}^{k+1} \sigma_\ell u_\ell v_\ell^T z\right) \\ &= \sum_{j,\ell=1}^{k+1} \sigma_j \sigma_\ell z^T v_j \underbrace{u_j^T u_\ell}_{=\delta_{j,\ell}} v_\ell^T z_\ell = \sum_{j=1}^{k+1} \sigma_j^2 (v_j^T z)^2 \geq \sigma_{k+1} \underbrace{\sum_{j=1}^{k+1} (v_j^T z)^2}_{=\|z\|_2^2=1} = \sigma_{k+1} \end{aligned}$$

which shows that $\|A - A_k\|_2 \leq \|A - B\|_2$ for any matrix B of rank k. □

An analogy for Theorem 5.22 also holds in terms of the **Frobenius norm**. We state it without proof next.

[169] The well–known result from Linear Algebra that the rank and the dimension of the kernel always add up to the dimension of the space multiplied to a matrix.

Theorem 5.23 (Best approximation of lower rank). *Let* $A = U\Sigma V^T \in \mathbb{R}^{m\times n}$ *and* $k \le \min r(A)$ *be given. Then the solution of the problem*

$$\min_B \|A - B\|_F \qquad \textit{subject to } \operatorname{rank} B = k, \tag{5.30}$$

is given by

$$B = A_k := U\Sigma_k V^T. \tag{5.31}$$

The solution of (5.30) *is unique iff* $\sigma_k > \sigma_{k+1}$ *and the error is*

$$\|A - B\|_F^2 = \sum_{j=k+1}^{r(A)} \sigma_j^2. \tag{5.32}$$

Therefore, the SVD is an answer to the problem of approximating a vector space, represented by the matrix of a generating system, by a subspace of dimension k: Compute the SVD of the matrix

$$A = \sum_{j=1}^{r(A)} \sigma_j u_j v_j^T$$

and keep

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T = U_k \Sigma_k V_k^T, \qquad U_k \in \mathbb{R}^{m\times k},\ \Sigma_k \in \mathbb{R}^{k\times k},\ V_k \in \mathbb{R}^{m\times k}.$$

Then the columns of U_k are still **orthonormal**, i.e., $UU^T = I$ but not necessarily $U^TU = I$, and span the best k–dimensional subspace of $A\mathbb{R}^m$, while the columns of V_k define the **projection** onto that space.

Figure 5.1: The three grayscale images

Example 5.24. As an example, we apply the SVD to the three images of Fig. 5.1, more precisely to the subsampled[170] versions of Fig. 5.2. The results are depicted in Fig. 5.3. The first and third singular vector show common structure and difference of the two first, very similar images while the one in the middle is more or less the third image which is recognized as "different".

[170] For serious learning applications one needs more sophisticated image processing routines.

Figure 5.2: The subsampled images.

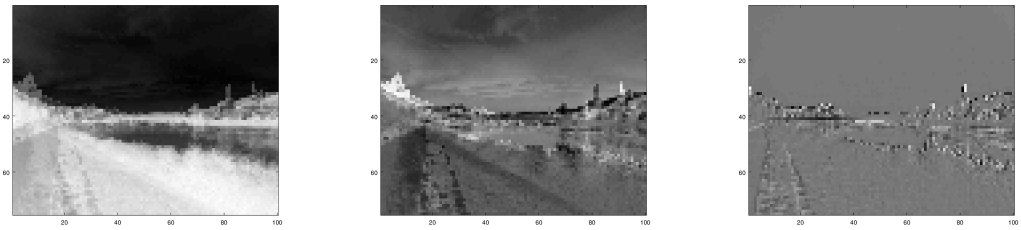


Figure 5.3: The singular vectors of the image matrix. The respective singular values are 2.1×10^4, 2.8×10^3 and 1.2×10^3, hence the first image part dominates significantly.

Thus, the PCA serves as a dimension reduction by projecting any vector x on the subspace spanned by U_k via V_k. This can be used to attribute objects to learned objects, for example testing a picture with the result that it is *"60% image one and 40% image two"*.
There is a simple reason why the images were downsampled as in Fig 5.2: `octave` was refusing to compute the SVD since the columns were too large with 750000 entries; the original image size is 750×1000. On the other hand, especially for huge matrices with big high dimensional data, we are not interested in computing *all* singular values but only the leading, i.e., largest ones. This can be done by the following **greedy procedure**.

Algorithm 5.25 (Greedy SVD)**.**
Given: matrix $A \in \mathbb{R}^{m\times n}$.

1. Set $A_0 := A$ and $k = 1$.
2. Repeat
   (a) Solve the optimization problem[171]

$$\min_{u,v} u^T A_k v \qquad \text{subject to } \|u\|_2 = \|v\|_2 = 1,$$

[171]This is minimizing a **bilinear form**, however not necessarily an inner product which would require A to be **symmetric** and (strictly) **positive definite**.

and call the solutions u_k and v_k

(b) Set $\sigma_k := u_k^T A_k v_k$.

(c) Set[172]

$$A_{k+1} := A_k - \sigma_k u_k v_k^T$$

and replace k by $k+1$.

until $\sigma_k = 0$.

Result: SVD

$$A = [u_1, \ldots, u_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} [v_1, \ldots, v_k]^T.$$

A particular advantage of this algorithm is that it can be stopped even if $\sigma_k \neq 0$, for example if this number is sufficiently small since then we automatically have an estimate on the quality of the approximation with respect to the Euclidean operator norm, see Theorem 5.22.

Theorem 5.26 (Greedy SVD)**.** *The **greedy algorithm**[173] 5.25 computes the SVD.*

Proof: Beginning with the first step, we note that there must be **Lagrange multipliers** λ_v, λ_w such that

$$\begin{aligned} 0 &= Av - 2\lambda_u u, \\ 0 &= A^T u - 2\lambda_v v. \end{aligned}$$

If $A \neq 0$, the maximum must be positive[174] and therefore $\lambda_u \lambda_v \neq 0$. Hence,

$$u = \frac{1}{2\lambda_u} Av \qquad \text{and} \qquad v = \frac{1}{2\lambda_v} A^T u$$

so that[175]

$$u^T Av = 2\lambda_u u^T u = 2\lambda_u = 2\lambda_v \qquad \Rightarrow \qquad \lambda_u = \lambda_v = \frac{\sigma}{2},$$

and

$$u = \frac{1}{4\lambda_u\lambda_v} AA^T u = \frac{1}{\sigma^2} A^T Au, \qquad v = \frac{1}{4\lambda_u\lambda_v} AA^T v = \frac{1}{\sigma^2} AA^T v.$$

Therefore we set $\sigma_1 = \sigma$, $u_1 = u$ and $v_1 = v$ with $Av = \sigma u$ and $A^T u = \sigma v$.

[172] Of course this step is only performed if $\sigma_k \neq 0$.

[173] A greedy algorithm optimizes a subproblem or a local problem in each step. In general greedy algorithms cannot be guaranteed to converge to a global optimum, but under some circumstances they do. Like here.

[174] If it were negative, multiply either u or v by -1; that's the beauty of bilinear forms, even of multilinear forms in general.

[175] By symmetry and definition of σ.

Now suppose that after k steps of the greedy iteration we computed σ_j, u_j and v_j where the u_j and the v_j are mutually orthonormal[176]. Moreover, with

$$A_\ell := A - \sum_{j=1}^{\ell} \sigma_j \, u_j v_j^T, \qquad \ell = 1, \ldots, k,$$

we also assume that

$$\sigma_j := \max_{\|u\|=\|v\|=1} u^T A_{j-1} v \qquad \text{and} \qquad A_{j-1} v_j = \sigma_j u_j, \quad A_{j-1}^T u_j = \sigma_j v_j,$$

and that u_j and v_j are eigenvectors of $A_{j-1}A_{j-1}^T$ and $A_{j-1}^T A_{j-1}$, respectively, for the eigenvalue σ_j^2, $j = 1, \ldots, k$. All that has been ensured for $k = 1$. Since the optimal solution u, v satisfies

$$\sigma := u^T A_k v = u^T A v - \sum_{j=1}^{k} \sigma_j \, (u_j^T u)(v_j^T v), \tag{5.33}$$

the Laplace multiplier requirements become

$$\begin{aligned} 0 &= \left(A - \sum_{j=1}^{k} \sigma_j \, u_j v_j^T \right) v - 2\lambda_u \, u, \\ 0 &= \left(A - \sum_{j=1}^{k} \sigma_j \, u_j v_j^T \right)^T u - 2\lambda_v \, v \end{aligned}$$

and $\lambda_v = 0$ would imply

$$A^T u = \sum_{j=1}^{k} \sigma_j \, (u_j^T u) v_j \qquad \Rightarrow \qquad u^T A v = \sum_{j=1}^{k} \sigma_j \, (u_j^T u)(v_j^T u).$$

Plugging this into (5.33) results in $u^T A_k v = 0$ and therefore $A_k = 0$. Hence $\sigma > 0$ again implies that $\lambda_u \lambda_v \neq 0$. Consequently,

$$u = \frac{1}{2\lambda_u} A_k v \qquad \Rightarrow \qquad v = \frac{1}{4\lambda_u \lambda_v} A_k^T A_k v,$$

and also $2\lambda_v = A_k^T u$. Again,

$$\sigma = u^T A_k v = 2\lambda_u u^T u = 2\lambda_v v^T v \qquad \Rightarrow \qquad \lambda_u = \lambda_v = \frac{\sigma}{2},$$

and u and v are the eigenvectors of $A_k A_k^T$ and $A_k^T A_k$ for the eigenvalue σ^2, respectively. Since $A_k = A_{k-1} - \sigma_k \, u_k v_k^T$, orthogonality gives

$$\begin{aligned} A_k^T A_k &= \left(A_{k-1} - \sigma_k \, u_k v_k^T \right)^T \left(A_{k-1} - \sigma_k \, u_k v_k^T \right) \\ &= A_{k-1}^T A_{k-1} + \sigma_k^2 v_k v_k^T - \sigma_k \left(A_{k-1}^T u_k v_k^T + v_k u_k^T A_{k-1} \right) \\ &= A_{k-1}^T A_{k-1} + \sigma_k^2 v_k v_k^T - \sigma_k \left(\sigma_k v_k v_k^T + \sigma_k v_k v_k^T \right) = A_{k-1}^T A_{k-1} - \sigma_k^2 v_k v_k^T, \end{aligned}$$

[176] Which is trivially satisfied for $k = 1$ where it was only a normalization issue.

so that[177], by induction,

$$A_k^T A_k = A^T A - \sum_{j=1}^{k} \sigma_j^2 v_j v_j^T, \qquad A_k A_k^T = AA^T - \sum_{j=1}^{k} \sigma_j^2 u_j u_j^T. \tag{5.34}$$

It follows that

$$A^T A v_j = \left(A_{j-1}^T A_{j-1} + \sum_{\ell=1}^{j-1} \sigma_\ell^2 v_\ell v_\ell^T \right) v_j = \sigma_j^2 v_j,$$

which yields together with

$$\sigma^2 v = A_k^T A_k v = AA^T v - \sum_{j=1}^{k} \sigma_j^2 \left(v_j^T v \right) v_j$$

that, for $j = 1, \ldots, k$,

$$\sigma^2 v_j^T v = v_j^T AA^T v - \sum_{\ell=1}^{k} \sigma_\ell^2 \left(v_\ell^T v \right) (v_\ell^T v_j) = \sigma_j v_j^T v - \sigma_j v_j^T v = 0.$$

A symmetric argument for u extends orthogonality to $k+1$ and the choice

$$\sigma_{k+1} = \sigma, \quad u_{k+1} = u, \qquad v_{k+1} = v$$

advances the induction. □

In summary, the PCA can be used for the following purposes:

Dimension reduction: The PCA gives the best projection on low(er) dimensional subspaces spanned by the **left singular vectors** $u_1, \ldots, u_k$ and the dual vectors that have to be used for this projection, namely the **right singular vectors** $v_1, \ldots, v_k$. It is then equivalent to work in $\mathbb{R}^k$, replacing $x \in \mathbb{R}^n$ by $\hat{x} = \Sigma_k V_k^T x \in \mathbb{R}^k$. Only for *interpretation* of the result the vectors $U_k \hat{x} \in \mathbb{R}^m$ should be used.

Clustering: For a clustering into K groups compute the K leading singular values and then put x into the cluster j for which $(\Sigma_K V_K^T x)_j$ becomes maximal, either in value or in absolute value. Note, however, that usually the clusters according to large singular values will contain more elements than the clusters corresponding to small singular values.

5.4 Independent component analysis

There is another interpretation of the SVD that should be mentioned briefly. Again, we have measurements $X \subset \mathbb{R}^d$ which we arrange into a matrix

$$X = U\Sigma V^T = AV^T = AW, \qquad A := U\Sigma,\ W = V^T. \tag{5.35}$$

[177] The second formula is proved in an analogous fashion.

Hence, any column $x \in X$ can be written as

$$x = Aw_x, \qquad w_x \in \mathbb{R}^n, \qquad x \in X,$$

so that A explains the behavior of the measured variable x in terms of the "hidden" variable w_x. Since W ist orthogonal, i.e., $I = W^TW = WW^T$ these variables are **uncorrelated** provided that the measurement have **zero mean**:

$$0 = \sum_{x \in X} x = X1 \qquad \Rightarrow \qquad w_x 1 = 0.$$

In that case the matrix $WW^T = I$ means that the variables w_x are *independent* which is the reason why the decomposition from (5.35) is called **independent component analysis** or **ICA** for short. For more details see, for example [6, 19].

Of course, this lecture could only be a small introduction to Learning Theory, i.e., the mathematics behind machine learning. It is far from complete and just one angle from which the issue can be looked at. But for those who are interested in more: There are books, some of them even listed in the references.

References A

[1] A. G. Aitken, *On interpolation by iteration of proportional parts, without the use of differences*, Proc. Edinburgh Math. Soc. **3** (1932), 56–76.

[2] M. Anthony, *Discrete mathematics of neural networks. selected topics*, Monographs on Discrete Mathematics and Applications, SIAM, 2001.

[3] V. I. Arnol'd, *The representation of functions of several variables*, Mat. Prosveščˇ **3** (1958), 41–61.

[4] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnawasmy, S. Villar, and R. Ward, *Relax, no need to round: integrality of clustering formulations*, (2015), arXiv:1408.4045v5.

[5] J. Bauschinger, *Interpolation*, Encyklopädie der Mathematischen Wissenschaften, Bd. I, Teil 2, B. G. Teubner, Leipzig, 1900, pp. 800–821.

[6] Ch. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006.

[7] C. de Boor, *A practical guide to splines*, Springer–Verlag, New York, 1978.

[8] C. de Boor, *Splinefunktionen*, Lectures in Mathematics, ETH Zürich, Birkhäuser, 1990.

[9] F. Cucker and D. X. Zhou, *Learning Theory: An Approximation Theory viewpoint*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2007.

[10] P. J. Davis, *Interpolation and approximation*, Dover Books on Advanced Mathematics, Dover Publications, 1975.

[11] Ke-Lin Du and M. N. S. Swamy, *Neural networks and statistical learning*, Springer, 2014.

[12] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer graphics*, 2nd ed., Addison Wesley, 1990.

[13] O. Forster, *Analysis 3. Integralrechung im* $\mathbb{R}^n$ *mit Anwendungen*, 3. ed., Vieweg, 1984.

[14] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, 2. ed., Grundlehren der mathematischen Wissenschaften, Springer–Verlag, 1983.

[15] G. Golub and C. F. van Loan, *Matrix computations*, 3rd ed., The Johns Hopkins University Press, 1996.

[16] P. Halmos, *I want to be a mathematician. an automathography*, MAA Spectrum Series, Mathematical Association of America, 1988.

[17] R. W. Hamming, *Digital filters*, Prentice–Hall, 1989, Republished by Dover Publications, 1998.

[18] H. Handels, *Medizinische bildverarbeitung*, B. G. Teubner, 2000.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, 2. ed., Springer, 2009.

[20] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[21] R. A Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1991.

[22] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley & Sons, 1966.

[23] S. Karlin, *Mathematical methods and theory in games, programming and economics*, Dover Phoenix Editions, Addison–Wesley, 1959, Dover Reprint 2003.

[24] ———, *Total positivity*, Stanford University Press, Stanford CA, 1968.

[25] Y. Katznelson, *An introduction to harmonic analysis*, 2. ed., Dover Books on advanced Mathematics, Dover Publications, 1976.

[26] A. N. Kolmogoroff, *On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition*, Dokl. Akad. Nauk. SSSR **114** (1957), 369–373.

[27] E. Kreyszig, *Introductionary functional analysis with applications*, John Wiley & Sons, 1978.

[28] E. Lawler, *Combinatorial optimization. networks and matroids*, Holt, Rinehart & Winston, 1974, Dover reprint 2001.

[29] G. G. Lorentz, *Metric entropy, widths, and superpositions of functions*, Amer. Math. Monthly **69** (1962), 469–485.

[30] G. G. Lorentz, *Approximation of functions*, Chelsea Publishing Company, 1966.

[31] S. Mallat, *A wavelet tour of signal processing*, 2. ed., Academic Press, 1999.

[32] ———, *A wavelet tour of signal processing: The sparse way*, 3rd ed., Academic Press, 2009.

[33] ———, *Group invariant scattering*, Commun. Pure Appl. Math. **65** (2012), 1331–1398.

[34] ———, *Understanding deep convolutional networks*, Phil. Tran. R. Soc. A **374** (2016), 1–16.

[35] C. A. Micchelli, *Interpolation of scattered data: distance matrices and conditionally positive definite functions*, Constr. Approx. **2** (1986), 11–22.

[36] ———, *Mathematical aspects of geometric modeling*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 65, SIAM, 1995.

[37] C. A. Micchelli and M. Pontil, *Learning the kernel function via regularization*, Journal of Machine Learning Research **2005** (2005), 1099–1125.

[38] J. von Neumann, *Zur Theorie der Gesellschaftsspiele*, Math. Annalen **100** (1928), 295–320.

[39] J. von Neumann and O. Morgenstern, *Theory of games and economic behavior*, sixth paperback printing, 1990 ed., Princeton University Press, 1944.

[40] J. Nocedal and S. J Wright, *Numerical optimization*, Springer Series in Operations Research, Springer, 1999.

[41] J. Peng and Y. Wei, *Approximating* k*–means clustering via semindefinite programming*, SIAM J. Optim. **18** (2007), 186–205.

[42] M. J. D. Powell, *Some global convergence properties of a variable matric algorithm for minimization without exact line searches*, SIAM–AMS Proceedings 9: Nonlinear Programming (1976), 53–72.

[43] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[44] R. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review **65** (1985), 386–408.

[45] T. Sauer, *Einführung in die Bild- und Signalverarbeitung*, Vorlesungsskript, Universität Passau, 2012.

[46] ———, *Einführung in die Numerische Mathematik*, Vorlesungsskript, Universität Passau, 2013.

[47] ———, *Optimierung*, Vorlesungsskript, Universität Passau, 2013.

[48] ———, *Analysis 1*, Vorlesungsskript, Universität Passau, 2014.

[49] ———, *Analysis 2*, Vorlesungsskript, Universität Passau, 2015.

[50] ———, *Geometric modeling*, Vorlesungsskript, Universität Passau, 2015.

[51] B. Schölkopf and A. J. Smolka, *Learning with kernels*, The MIT Press, 2002.

[52] P. Spellucci, *Numerische Verfahren der nichtlinearen Optimierung*, Internationale Schriftenreihe zu Numerischen Mathematik, Birkhäuser, 1993.

[53] D. A. Sprecher, *On the structure of continuous functions of several variables*, Trans. Amer. Math. Soc. **115** (1965), 340–355.

[54] ———, *A representation theorem for continuous functions of several variables*, Proc. Amer. Math. Soc. **16** (1965), 200–203.

[55] ———, *An improvement in the superposition theorem of kolmogorov*, J. Math. Anal. Appl. **38** (1972), 208–213.

[56] I. F. Steffensen, *Interpolation*, Chelsea Pub., New York, 1927.

[57] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions I*, Springer Verlag, 1970.

[58] M. Vetterli and J. Kovačević, *Wavelets and subband coding*, Prentice Hall, 1995.

[59] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Review **11** (1969), 220–228.

[60] Yinyu Ye, *Interior points algorithms. Theory and analysis*, John Wiley & Sons, 1997.

[61] K. Yosida, *Functional Analysis*, Grundlehren der mathematischen Wissenschaften, Springer–Verlag, 1965.

Index

B