# Basics of Mathematics in Machine Learning II

Toni Karvonen

Exactum B326 — University of Helsinki
toni.karvonen@helsinki.fi

March 6, 2024

## Contents

# 1 Introduction

These are lecture notes for the course *Basics of Mathematics in Machine Learning II* (MAT11015) lectured in Spring 2024. Th notes approximately contain the material in Chapter 5 and Sections 6.1–6.4 and 7.1 of *Mathematics for Machine Learning* by Deisenroth, Faisal and Oong (freely available at https://mml-book.github.io/). We cover the following three topics:

**I** *Univariate calculus* — Section 2

**II** *Vector calculus* — Section 3

**III** *Discrete probability* — Section 4

Topics **I** and **II** cover the tools and results from mathematical analysis that are essential for machine learning. The primary learning objective of these topics is to (a) understand the basics of mathematical optimisation and (b) be able to implement some optimisation methods. Our approach to calculus is not completely rigorous, as we do not properly define some of the notions that we encounter or use and occasionally sweep certain finer points under the rug. A rigorous treatment would require *much* more time than we have. Those interested should consider some of the following courses:

- *Raja-arvot* (MAT11003)
- *Differentiaalilaskenta* (MAT11004)
- *Integraalilaskenta* (MAT11005)
- *Calculus IA: Limits and differentiation* (MAT11006)
- *Calculus IB: Integration* (MAT11007)
- *Advanced calculus* (MAT11008)
- *Vektorianalyysi I* (MAT21003)
- *Vektorianalyysi II* (MAT21020)

Topic **III** covers the basics definitions, concepts and results of discrete probability theory. The objective of this topic is to provide the essentials upon which to build on subsequent courses on statistical machine learning and data science. We have time to cover only the bare minimum about probability, and you should seriously consider taking a course dedicated to probability theory, such as

- *Todennäköisyyslaskenta I* (MAT12003)

## 1.1 Motivation

Let us begin by motivating the these topics with three simple examples.

**Ordinary Least Squares.** Suppose that we have observations (or outputs) $y_1, \dots, y_n \in \mathbb{R}$ at some distinct locations (or inputs) $x_1, \dots, x_n \in \mathbb{R}$. This is our *training data* (suom. *opetusjoukko*). To name a few examples, the locations could describe spatial coordinates of some physical sensors, they could be time instances [in which case with a *time series* (suom. *aikasarja*)], or they could be different parameters that an experiment is performed with. Given these data, we want to predict what the observation might be at a location that is not included in the dataset. To do this, we can postulate that observations have linear relationship to the locations:

$$y_i = a + bx_i + \varepsilon_i \quad \text{for} \quad i = 1, \dots n, \tag{1.1}$$

where $a$ and $b$ determine the relationship between $x_i$ and $y_i$ and the residual $\varepsilon_i$ accounts for observation noise or influence on $y_i$ of sources other than $x_i$. Our task is to find a "good" or

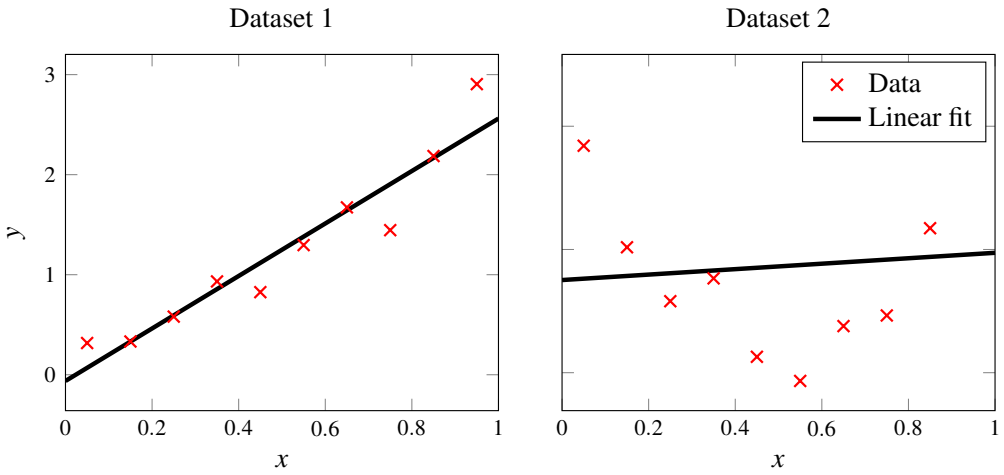


**Figure 1:** The linear fits $f(x) = a^* + bx^*$ for two different datasets. Here $a^*$ and $b^*$ are computed from (1.3). The fit is quite good for the first dataset but completely useless for the second.

"optimal" values for $a$ and $b$. This is called *linear regression* (suom. *lineaarinen regressio*). One way to achieve this is to select $a$ and $b$ that minimise the *sum of squared residuals*

$$L(a,b) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2. \tag{1.2}$$

The function $L$ is an important example of a *loss function* (suom. *tappiofunktio*), or *cost function*. The selection of a loss function is arbitrary (e.g., we could alternatively try to minimise $\sum_{i=1}^{n} |\varepsilon_i|$), but the *quadratic* loss function in (1.2) is mathematically extremely convenient, yielding the *ordinary least squares* method (suom. *pienimmän neliösumman menetelmä*). During the course we will learn how to minimise $L$ and that (as long as $n \geq 2$)

$$\begin{bmatrix} a^* \\ b^* \end{bmatrix} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}, \quad \text{where} \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \in \mathbb{R}^{n\times 2} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n, \tag{1.3}$$

solve this minimisation problem *uniquely*, in that $L(a^*,b^*) < L(a,b)$ for any other pair $(a,b)$. The *linear fits* $f(x) = a^* + bx^*$ produced by linear regression are depicted in Figure 1 for two rather different datasets. The figure shows that the linear fit can be quite useful if the data indeed exhibit a linear trend, as is the case for Dataset 1. That is, for an input $x_0$ not contained in the training data it seems plausible that $f(x_0)$ would be close to the corresponding output, $y_0$. However, the fit is useless if no such trend is present, as is the case for Dataset 2. This is because the relationship (1.1) between $x_i$ and $y_i$ that we have postulated is not particularly expressive, having only two parameters, $a$ and $b$.

**Neural Networks.** At their core, *neural networks* (suom. *neuroverkko*) consist of nothing more than (i) a flexible postulate like (1.1) for the relationship between the inputs and outputs; (ii) selection of some parameters by minimising a loss function; and (iii) computation of a fit that is then used to predict the output at unobserved inputs.[1] Let us look at a simple two-layer network based on the *sigmoid activation function* (suom. *aktivaatiofunktio*)

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{1.4}$$

[1]The following article gives a decent introduction to neural networks: HIGHAM & HIGHAM (2019). Deep learning: An introduction for applied mathematicians. *SIAM Review* 61(4):860–891.

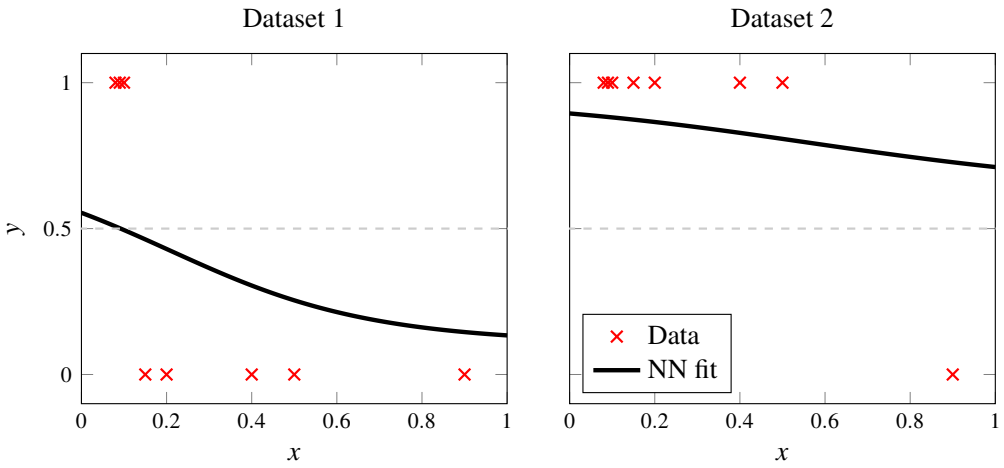


**Figure 2:** The neural network (NN) fit $f(x) = F(x \mid w_1^*, w_2^*, b_1^*, b_2^*) = \sigma(w_2^*\sigma(w_1^* x + b_1^*) + b_2^*)$ for two different datasets. Here $w_1^*, w_2^*, b_1^*$ and $b_2^*$ have been computed using gradient descent. The datasets consist of $n = 8$ *labels* $y \in \{0, 1\}$ (i.e., points labelled $y = 1$ are in a category $A$ and points those labelled $y = 0$ in category $B$). After computing the neural network fit $f$, we may classify a new point by, for example, saying it is in category $A$ if $f(x) > \frac{1}{2}$ and in category $B$ if $f(x) \leq \frac{1}{2}$. This *decision boundary* (suom. *päätöspinta*) is plotted as a dashed grey line. Such classification works for Dataset 1 (at least barely), but not for Dataset 2. This is understandable, as the neural network we have used is extremely simple.

Let $w_1, w_2 \in \mathbb{R}$ be *weights* (suom. *paino*) and $b_1, b_2 \in \mathbb{R}$ *biases* (suom. *vakiotermi*). Inspired by (1.1), we postulate that the training outputs $0 \leq y_i \leq 1$ are related to the inputs $x_i$ via the equation

$$y_i = \sigma(w_2\sigma(w_1 x_i + b_1) + b_2) + \varepsilon_i \quad \text{for} \quad i = 1, \ldots, n. \tag{1.5}$$

Define

$$F(x \mid w_1, w_2, b_1, b_2) = \sigma(w_2\sigma(w_1 x + b_1) + b_2), \tag{1.6}$$

so that (1.5) becomes $y_i = F(x_i \mid w_1, w_2, b_1, b_2) + \varepsilon_i$. As in the case of linear regression, we may define a quadratic loss function

$$L(w_1, w_2, b_1, b_2) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - F(x_i \mid w_1, w_2, b_1, b_2))^2 \tag{1.7}$$

and select the weights $w_1, w_2$ and biases $b_1, b_2$ by minimising this loss function. But we hit a roadblock. For, unlike in the case of ordinary least squares method that provides the simple expression (1.3) for the minimisers of the loss function (1.2), no such nice formula is available now. Instead, we have to device an *optimisation method* to minimise (1.7). Because in real problems there are thousands of weights and biases and the training data sets are huge, it is essential that this method be efficient, in that it should call the loss function as few times as possible. During the course we will learn how to use *derivatives* and *gradients* to describe the direction and rate of change of functions and how to implement the *gradient descent* (suom. *gradienttimenetelmä*) algorithm, the workhorse of machine learning. Results of selecting the parameters $w_1, w_2, b_1$ and $b_2$ using gradient descent are displayed in Figure 2.

**Statistical Inference and Probability.** Much of machine learning and data analysis is based on *statistical inference* (suom. *tilastollinen päättely*). That is, given some data one seeks to infer properties of an underlying probability distribution and exploit these properties to

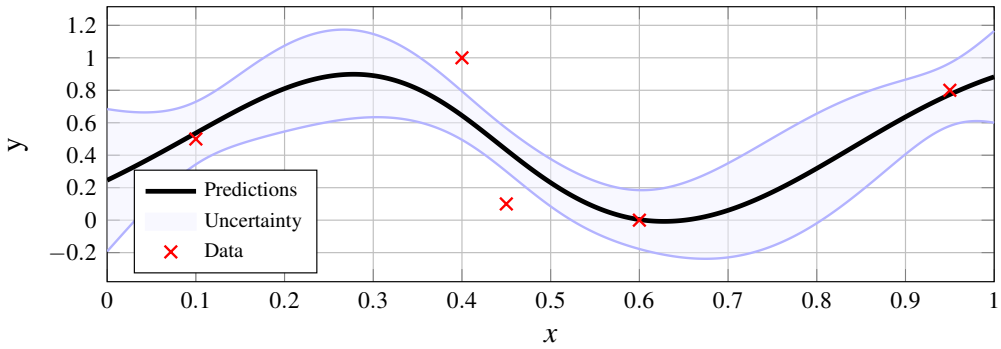


**Figure 3:** Statistical inference and prediction based on *Gaussian processes* (suom. *gaussinen prosessi*). Given some data, we construct a certain *statistical model* (suom. *tilastollinen malli*) that attempts to represent the process that generated the data (i.e., some relationship between $x$ and $y$). Using this model and statistical assumptions about noise present in the data we can perform predictive inference. The predictions are uncertain, as represented by the shaded region. While our best prediction for the data value at, say, $x = 0.7$ is $y \approx 0.05$, we acknowledge that there is significant uncertainty in this prediction, considering "it very likely" that $y \in [-0.2, 0.4]$.

make predictions. The foundations of statistical inference are in the *theory of probability* (suom. *todennäköisyysteoria*), which provides a mathematical language for decision-making and uncertainty. Figure 3 illustrates a certain statistical inference and prediction method popular in machine learning and spatial statistics. During the course we will learn the basic definitions, concepts and results of probability theory.

## 1.2 Notational Conventions

*Number sets.* The sets $\mathbb{R}$, $\mathbb{N}$, and $\mathbb{Z}$ of real numbers, natural numbers, and integers are

$$\mathbb{R} = (-\infty, \infty), \quad \mathbb{N} = \{1, 2, \ldots\}, \quad \text{and} \quad \mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\},$$

respectively. Note that this definition excludes zero from $\mathbb{N}$ and that the infinities $-\infty$ and $\infty$ are not included in $\mathbb{R}$.

*Scalars, vectors and matrices.* When talking about scalars, vectors, and matrices, we will use plain font (i.e., $x$ or $X$) for scalars, bold lowercase font for vectors (i.e., $\mathbf{x}$), and bold uppercase font for matrices (i.e., $\mathbf{X}$). Note that, depending on the dimensions, $\mathbf{x}$ may happen to be a scalar and $\mathbf{X}$ a vector (or even a scalar). But $x$ will never be a vector or a matrix.

*Constants and functions.* The constant $e \approx 2.718$ is *Euler's constant*, or *Napier's constant* (suom. *Neperin luku*). Moreover, $\exp(x) = e^x$. Euler's constant is the base of the natural logarithm. The natural logarithm of $x$ is always denoted $\log(x)$ in these notes. It is the inverse function of $e^x$, satisfying $\log(e^x) = x \log(e) = x$. We do not use logarithms with other bases.

## 1.3 Preliminaries on Functions

Let us recall some basic notation and concepts that we will use throughout the course.

**Definition 1.1 (FUNCTION).** Let $X$ and $Y$ be sets. A *function* (suom. *funktio*) $f$ from $X$ to $Y$, denoted

$$f \colon X \to Y, \tag{1.8}$$

assigns to every element $x \in X$ exactly one element $f(x) \in Y$. The function $f$ *maps* (suom. *kuvaa*) the *input* $x$ to the *output* $f(x)$. This input-output relationship is often written as

$$x \mapsto f(x). \tag{1.9}$$

In this course we mostly focus on functions $f\colon \mathbb{R}^d \to \mathbb{R}$ (i.e., $X = \mathbb{R}^d$ and $Y = \mathbb{R}$) that map vectors $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ to real numbers $f(\mathbf{x}) \in \mathbb{R}$. Examples of such functions are the bivariate polynomial $f(\mathbf{x}) = f(x_1, x_2) = x_1^2 x_2^3 + x_1 - 3$ (i.e., $f\colon \mathbb{R}^2 \to \mathbb{R}$) and the *ReLU activation function* (suom. *aktivointifunktio*)

$$f(\mathbf{x}) = \max\{0, a + \mathbf{w}\cdot\mathbf{x}\} = \max\{0, a + \textstyle\sum_{i=1}^d w_i x_i\} = \begin{cases} 0 & \text{if} \quad a + \mathbf{w}\cdot\mathbf{x} < 0, \\ a + \mathbf{w}\cdot\mathbf{x} & \text{otherwise,} \end{cases} \tag{1.10}$$

where $a \in \mathbb{R}$ and $\mathbf{w} \in \mathbb{R}^d$ are fixed (i.e., $f\colon \mathbb{R}^d \to \mathbb{R}$). However, a function can be something much more complicated: A computer program that maps user input, such as text or parameter values, to some output, such as a text response or the result of a physics simulation, is also a function.

## 1.4 Function Composition

Complicated functions are often formed by *composing* simple functions.

**Definition 1.2 (FUNCTION COMPOSITION).** Let $g\colon X \to Y$ and $h\colon Y \to Z$ be functions. Their *composite function* (suom. *yhdistetty funktio*) $f = h \circ g$ is a function from $X$ to $Z$ given by

$$f(x) = h(g(x)). \tag{1.11}$$

To compute $f(x) = h(g(x))$ we (i) first map $x$ through $g$ and (ii) after this map the output $g(x)$ through $h$. We can also compose more than two functions, so that

$$f = r \circ h \circ g \quad \text{is given by} \quad f(x) = (r \circ h \circ g)(x) = r(h(g(x))). \tag{1.12}$$

One can think recursively:

$$f = r \circ h \circ g = r \circ f_1, \quad \text{where} \quad f_1 = h \circ g. \tag{1.13}$$

**Example 1.3.** Define the functions

$$g(x) = 3\pi, \quad h(x) = \log(1+x), \quad r(x) = 2 + x^{-3} \quad \text{and} \quad l(x) = x, \tag{1.14}$$

all from $\mathbb{R}$ to $\mathbb{R}$. Then

$$f(x) = (h \circ g)(x) = h(g(x)) = h(3\pi) = \log(1+3\pi), \tag{1.15a}$$
$$f(x) = (g \circ h)(x) = g(h(x)) = g(\log(1+x)) = 3\pi, \tag{1.15b}$$
$$f(x) = (h \circ r)(x) = h(r(x)) = h(2+x^{-3}) = \log(3+x^{-3}), \tag{1.15c}$$
$$f(x) = (r \circ l)(x) = r(l(x)) = r(x) = 2 + x^{-3}, \tag{1.15d}$$
$$f(x) = (r \circ l \circ h)(x) = r(l(h(x))) = r(h(x)) = r(\log(1+x)) = 2 + \log(1+x)^{-3}. \tag{1.15e}$$

Note that (1.15a) and (1.15b) are simply the *constant functions* $x \mapsto \log(1+3\pi)$ and $x \mapsto 3\pi$, which map every $x \in \mathbb{R}$ to the constants $\log(1+3\pi)$ and $3\pi$, respectively.

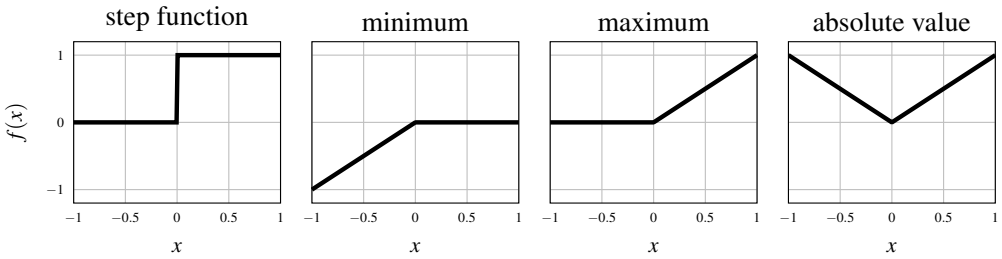


**Figure 4:** Four piecewise defined functions. From left to right: The step function in (1.19), the minimum and maximum functions in (1.20), and the absolute value function in (1.21).

**Example 1.4.** We can write the ReLU activation function (1.10) as the composition

$$f = h \circ g, \tag{1.16}$$

where

$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} \quad \text{and} \quad h(x) = \max\{0, a + x\} \tag{1.17}$$

are functions from $\mathbb{R}^d$ to $\mathbb{R}$ and from $\mathbb{R}$ to $\mathbb{R}$, respectively. The composition is *not* unique. We could have alternatively selected $g(\mathbf{x}) = a + \mathbf{w} \cdot \mathbf{x}$ and $h(x) = \max\{0, x\}$.

**Example 1.5.** Consider the following idiotic piece of pseudocode:

```
1: procedure FUNC(scalar x, integer n ≥ 1)
2:     if n ≥ 1 then
3:         return 2 × FUNC(x, n − 1)
4:     else
5:         return x
6:     end if
7: end procedure
```

Given an integer constant $n \geq 1$, this procedure computes $f_n(x) = \text{FUNC}(x, n) = 2^n x$ by using the $n$-fold composition

$$f_n(x) = \underbrace{(g \circ \cdots \circ g)}_{n \text{ times}}(x), \quad \text{where} \quad g(x) = 2x. \tag{1.18}$$

For example, when $n = 3$ we have $f_n(x) = (g \circ g \circ g)(x) = g(g(g(x))) = 2(2(2x)) = 2^3 x = 8x$.

## 1.5 Piecewise defined functions

We will occasionally encounter *piecewise defined functions* (suom. *paloittain määritelty funktion*), such as the ReLU activation function (1.10). Typical piecewise defined functions include the *step function* (suom. *porrasfunktio*)

$$f(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0, \end{cases} \tag{1.19}$$

the *maximum* and *minimum*

$$pf(x) = \min\{0,x\} = \begin{cases} x & \text{if } x < 0, \\ 0 & \text{if } x \geq 0 \end{cases} \quad \text{and} \quad f(x) = \max\{0,x\} = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0, \end{cases} \tag{1.20}$$

and the *absolute value* (suom. *itseisarvo*)

$$f(x) = |x| = \begin{cases} -x & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases} \tag{1.21}$$

These four functions are plotted in Figure 4.

# 2 Calculus

# 3 Vector Calculus

# 4 Discrete Probability