Learning From Data
Lecture Notes

Tobias Pohlen

November 17, 2014

Abstract

These lecture notes are based on the lecture *Learning From Data* held by Prof. Yaser Abu-Mostafa at the California Institute of Technology in 2012. The lecture is available via YouTube [AM]. Prof. Abu-Mostafa as well as CalTech are in no way affiliated with this script.

There is also a book [AMMIL12] that covers most of the lecture.

These lecture notes will be updated every week after the lectures have been released. The solution to the exercises will be added after the respective deadlines. You can find the current version of the script as well as the solutions to the programming exercises here: `http://bit.ly/1rZvzYg`.

Please report errors to `tobias.pohlen@rwth-aachen.de`.

Contents

1 Introduction **7**
- 1.1 The Learning Problem . . . 7
- 1.2 The Perceptron Learning Algorithm . . . 8

2 Learning Theory **11**
- 2.1 Introduction . . . 11
- 2.2 Is Learning Feasible? . . . 11
- 2.3 Noisy Targets . . . 15
- 2.4 Error Measures Revisited . . . 17
- 2.5 The Vapnik-Chervonenkis Inequality . . . 19
- 2.6 The VC-Dimension . . . 28
- 2.7 Bias-Variance Tradeoff . . . 31
- 2.8 Overfitting . . . 33
  - 2.8.1 Bias-Variance Tradeoff for noisy Data . . . 33
  - 2.8.2 Regularization . . . 34
  - 2.8.3 Validation . . . 35

3 Learning Algorithms **37**
- 3.1 The Linear Model . . . 37
  - 3.1.1 The Pocket Algorithm . . . 37
  - 3.1.2 Linear Regression . . . 37
  - 3.1.3 Linear Regression for Classification . . . 39
  - 3.1.4 Non-linear Transform . . . 39
  - 3.1.5 Logistic Regression . . . 40
- 3.2 Artificial Neural Networks and Support Vector Machines . . . 43

Appendices **45**

A Exercises **47**
- A.1 Exercise 1 . . . 47
- A.2 Exercise 2 . . . 51
- A.3 Exercise 3 . . . 54
- A.4 Exercise 4 . . . 60
- A.5 Exercise 5 . . . 65
- A.6 Exercise 6 . . . 68
- A.7 Exercise 7 . . . 69

Chapter 1

Introduction

1.1 The Learning Problem

Machine learning techniques have become a major building block of modern computer applications. There are many areas of application ranging from fraud detection to pedestrian recognition. The basic idea behind all approaches is that a functional relationship between some measurements and some output values is *learned* from data. Hence, instead of actually giving the computer detailed instructions on how to solve a certain task, we present the computer with the data and let it choose *the best* way to deal with the problem. The following definition formalizes this idea.

Definition 1.1.1 (The learning problem)**.** *There is an unknown* target functional f *that assigns measurements from an* input space $\mathcal{X}$ *to values from an* output space $\mathcal{Y}$

$$f : \mathcal{X} \mapsto \mathcal{Y} \tag{1.1}$$

A learning algorithm *is supposed to select a* hypothesis $g \in \mathcal{H}$ *from the set of all hypotheses* $\mathcal{H}$ *based on a finite number* training examples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \in \mathcal{X} \times \mathcal{Y}$ *such that*

$$g \approx f \tag{1.2}$$

In order to illustrate the the individual components of the learning problem, we give several examples for the case of supervised learning.

- Assume our target function is a real valued function $f : \mathbb{R} \mapsto \mathbb{R}$. In this case, the input space $\mathcal{X}$ as well as the output space $\mathcal{Y}$ correspond to the set of real numbers $\mathbb{R}$. Hence, our training examples are tuples $(\mathbf{x}_i, y_i) \in \mathbb{R} \times \mathbb{R}$. Technically, we can define the hypothesis set $\mathcal{H}$ to be the set of all real valued function (i.e. $\mathcal{H} = \{\mathbb{R} \mapsto \mathbb{R}\}$). However, this makes it difficult to define a proper learning algorithm that selects the best hypothesis. Therefore, we define $\mathcal{H}$ to be set of all polynomials up to degree D (i.e. $\mathcal{H} = \Pi_D$). By restricting the hypothesis set to polynomials, we can use interpolation as a learning algorithm. In summary, we have
  - Input space: $\mathcal{X} = \mathbb{R}$
  - Output space: $\mathcal{Y} = \mathbb{R}$
  - Unknown target function: $f : \mathbb{R} \mapsto \mathbb{R}$
  - Hypothesis set: $\mathcal{H} = \Pi_D$
  - Learning algorithm: Interpolation

- Assume we want to perform fraud detection. All of our measurements are D-dimensional real valued vectors. In this example, we only consider two possible classes: A measurement is a fraud or it is not. We can encode these two classes using -1 for fraud and 1 for no fraud. We define our hypothesis space to be the set of *linear classifiers*.

$$\mathcal{H} = \{\text{sign}(\mathbf{w}^T\mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}\} \tag{1.3}$$

  In this case, we could use the Perceptron Learning Algorithm, which we discuss in the next section, as our learning algorithm of choice. In summary:

  - Input space: $\mathcal{X} = \mathbb{R}^D$
  - Output space: $\mathcal{Y} = \{-1, 1\}$
  - Unknown target function: $f : \mathbb{R}^D \mapsto \{-1, 1\}$
  - Hypothesis set: $\mathcal{H} = \{\text{sign}(\mathbf{w}^T\mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}\}$
  - Learning algorithm: Perceptron Learning Algorithm (PLA)

It is important to understand which parts we can model ourselves and which we cannot when we deal with a learning problem. When we are presented with a new problem, we typically have a fixed amount of data for training, testing, and evaluation. Hence, we cannot control the quality or the amount of data. Therefore, we also cannot influence the input and output spaces[1]. Finally, the target function is fixed but unknown. The only things we can control ourselves are the hypothesis set $\mathcal{H}$ and the learning algorithm.

1.2 The Perceptron Learning Algorithm

One of the first learning algorithm that was proposed is the so called *Perceptron Learning Algorithm (PLA)* [Ros57]. It was proposed by Rosenblatt in the 1950s. It selects a hypothesis from the set of *linear classifiers*.

Definition 1.2.1 (Linear classifier)**.** *A* linear classifier *is a function*

$$h : \mathbb{R}^D \mapsto \{-1, 1\}, \mathbf{x} \mapsto \textit{sign}(\mathbf{w}^T\mathbf{x} + b) \tag{1.4}$$

where $\mathbf{w} \in \mathbb{R}^D$ *is the* weight vector *and* b *is the* bias *term.*

Remark 1. *In the context of linear classifiers, one often uses homogeneous vectors by defining* $\tilde{\mathbf{w}}_0 = b$ *and* $\tilde{\mathbf{x}}_0 = 1$*. This allows us to write a linear classifier simply in terms of an inner product*

$$h(\mathbf{x}) = \textit{sign}(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}) \tag{1.5}$$

We will refer to the homogeneous vector of $\mathbf{w}$ *by* $\tilde{\mathbf{w}}$*.*

One often refers to these linear classifiers as *(single layer) Perceptrons*. They are the basis for the more powerful neural networks. Given a data set, the Perceptron Learning Algorithm finds a hypothesis that classifies all training points correctly under the assumption that the data set is *linearly separable*.

[1] However, we can use a different representation of the data by extracting features.

Definition 1.2.2 (Linear separability)**.** *A training set* $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \in \mathbb{R}^D \times \{-1, 1\}$ *is called* linearly separable *if there exists a linear classifier* h *such that*

$$h(\mathbf{x}_i) = y_i \tag{1.6}$$

for all $i = 1, ..., N$*.*

This assumption basically means that there is some hypothesis that explains all training points correctly. It is important to note that this does not imply that f is a linear classifier. Although the converse holds, a training set can be linearly separable although the target function is not a linear classifier.
The algorithm works as follows:

- Initialize the weight vector $\tilde{\mathbf{w}}$ randomly
- While there are misclassified training examples
  - Choose a misclassified example $(\tilde{\mathbf{x}}_i, y_i)$ and update the weight vector as follows

$$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} + \alpha y_i \tilde{\mathbf{x}}_i \tag{1.7}$$

    where $\alpha \in (0, 1]$ is the so called *learning rate*

Remark 2. *The PLA finds a hypothesis that classifies all data points correctly in the case when the data is linearly separable. Later in the lecture, we will see that a hypothesis that explains all training points correctly does not necessarily need to be among best hypotheses for a certain problem. This is due to the fact that these hypotheses might adapt strongly to noisy measurements and therefore do not generalize well to unseen data.*

We presented the algorithm at this point as a first example of a working learning algorithm for a simple case. In the first exercise you will be asked to implement the algorithm. Because the algorithm is not important for the rest of the lectures, we skip the proof of convergence at this point and only refer to [Col].

Chapter 2

Learning Theory

2.1 Introduction

In order to discuss learning in a mathematical formal way, we have to make some assumptions about the data we are presented with. It can be shown that without any assumptions, learning would not be feasible[1]. Therefore, we make the following assumptions: $(\mathcal{X}, \mathcal{A}, \mathbb{P})$ is a probability space and our data is generated independently and identically from $\mathbb{P}$. This means that $\mathcal{A}$ is a σ-algebra on $\mathcal{X}$ and $\mathbb{P}$ is a probability measure on $\mathcal{X}$. This makes sure that previous measurements and new measurements are in some way *related*.

2.2 Is Learning Feasible?

In this chapter, we discuss the question whether or not learning is feasible in a mathematical sense. We can reformulate the question of feasibility as follows:

> Can we select a hypothesis based on a finite number independent samples that performs *well* on unseen data? What does the *training error* tell us about the true *performance*?

We start the discussion by introducing a method of measuring the performance of a hypothesis $h \in \mathcal{H}$. For this purpose, we introduce the concept of a *loss function*. The idea is that the loss function defines a point-wise penalty. If the output of the hypothesis deviates $h(\mathbf{x})$ from the true target y on some point $\mathbf{x}$, the loss function returns some penalty.

Definition 2.2.1. *A* loss function $\mathcal{L}(h, \mathbf{x}, y)$ *is a mapping*

$$\mathcal{L} : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R} \tag{2.1}$$

Many loss functions for different purposes have been proposed in the literature. Here, we list some of the most commonly used ones in order to give you an intuition.

- In the case of binary classification (i.e. $\mathcal{Y} = \{-1, 1\}$), a typical loss function is the so called *0-1-loss*

$$\mathcal{L}_{0-1}(h, \mathbf{x}, y) = I(h(\mathbf{x}) \neq y) \tag{2.2}$$

[1] Look up the *No Free Lunch* theorem.

where I is the *indicator function*

$$I(B) = \begin{cases} 1 & \text{if } B \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

The functions returns a loss of 1 if $\mathbf{x}$ is misclassified and a loss of 0 if the point is classified correctly.

- For the case of regression (i.e. $\mathcal{Y} = \mathbb{R}^p$), the so called *squared loss* is a common choice

$$\mathcal{L}_{squared}(h, \mathbf{x}, \mathbf{y}) = \|h(\mathbf{x}) - \mathbf{y}\|_2^2 \tag{2.4}$$

The loss is proportional to the squared distance from the prediction to the true output. We will use this loss function later for the case of linear regression.

- Later in the lecture, we will discuss *Support Vector Machines (SVMs).* We will see that they optimize the *hinge loss*

$$\mathcal{L}_{hinge}(h, \mathbf{x}, y) = \max(0, 1 - yh(\mathbf{x})) \tag{2.5}$$

where h is a *linear discriminant function*

$$h : \mathbb{R}^D \mapsto \mathbb{R}, \mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} + b \tag{2.6}$$

and $y \in \{-1, 1\}$ is a class label. The hinge loss does not give a penalty to points that are classified correctly and returns a penalty that is proportional to the distance from the point to the decision boundary if the point is misclassified.

We can use these loss functions in order to quantify the error that a hypothesis makes on the entire input space as well as on the training set.

Definition 2.2.2 (Error measures)**.** *Let $\mathbf{x}_1, ..., \mathbf{x}_N$ be i.i.d. sampled from $\mathbb{P}$, let $\mathcal{L}$ a loss function, and let $h \in \mathcal{H}$ be some hypothesis. We define the* in-sample-error *$E_{in}(h)$ as the sample mean*

$$E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(h, \mathbf{x}_i, y_i) \tag{2.7}$$

Analog, we define the out-of-sample *error as the expected value*

$$E_{out}(h) = \mathbb{E}[\mathcal{L}(h, \mathbf{x}, y)] \tag{2.8}$$

Remark 3 (Nomenclature)**.** *There are two important aspects of the chosen name that we want to highlight*

1. *The error measures are not measures in a measure theoretical sense.*

2. *In the literature, there can be other names for these measures. One often refers to the in-sample-error as the* training error*. In many publications about statistical learning theory you will read* empirical risk *for the in-sample-error and* actual *or* expected risk *for the out-of-sample error.*

Because the in-sample-error is the sample mean of independently and identically sampled values and the out-of-sample error is the expected value, we know from basic probability theory that

$$E_{in}(h) \to E_{out}(h) \text{ for } N \to \infty \text{ (almost surely)} \tag{2.9}$$

While this is certainly useful, it does not help us with the initial question. When we ask if learning is feasible, we want to know if there is any relationship between E_{in} and E_{out}. In the ideal case, the in-sample-error gives us an impression of how well the hypothesis performs on previously unseen data. Hence, we would like to quantify the relationship

$$E_{in}(h) = E_{out}(h) \tag{2.10}$$

Of course, this relation only holds in the limit (with probability 1). However, because of the convergence, we know that $|E_{in}(h) - E_{out}(h)| \to 0$. Since we do not know $E_{out}(h)$ (and can never measure it in practice), we quantify this relationship by finding an upper bound on the probability that the two quantities are **not** approximately equal. Hence, we want to find a bound on

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \tag{2.11}$$

for some $\epsilon > 0$. This probability quantifies the event that the in-sample-error is misleading in the sense that it is not a good representation for the out-of-sample error.
Fortunately, we can use a result from basic probability theory. *Hoeffding's Inequality* is an upper bound on the probability that the sample mean deviates from the expected value more than ϵ.

Theorem 2.2.3 (Hoeffding's Inequality)**.** *Let $X_1, ..., X_N$ be i.i.d. random variables with $f(X) \in [a, b]$. Then for all $\epsilon > 0$, we have*

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - \mathbb{E}[f(X)]\right| > \epsilon\right] \leq 2\exp\left(-\frac{2\epsilon^2 N}{(b-a)^2}\right) \tag{2.12}$$

Obviously, we can only apply Hoeffding's inequality if $\mathcal{L}$ is bounded by some constant. In the following discussion, we apply the inequality for the case of binary classification. For this purpose, we use the 0-1-loss that we introduced earlier

$$\mathcal{L}_{0-1}(h, \mathbf{x}, y) = I(h(\mathbf{x}) \neq y) \tag{2.13}$$

Obviously, $\mathcal{L}_{0-1} \in [0, 1]$. Therefore, we get

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq \underbrace{2\exp\left(-2\epsilon^2 N\right)}_{=:\delta} \tag{2.14}$$

This means that with a **probability** of at least $1 - \delta$ the statement $E_{in}(h) = E_{out}$ is **approximately correct**. Such a bound is called a P.A.C. bound (P.A.C. = probably approximately correct). The bound says that if we have a large amount of training examples N, we can be confident that the training error is a good representation of the out-of-sample error. Note however that our precision ϵ is squared in the exponential. Hence, if we want to make sure that we perform well on unseen data, our N has to be very high in order to compensate for the squared precision.
There is one important property about this bound that we want to highlight at this point: The bound is independent of the probability measure $\mathbb{P}$. This property is called *model-free*. Hence, regardless of the underlying probability distribution, this bound holds. This is important for our case because we only assumed that there is some underlying probability distribution that we do not know.

Remark 4. *There is an alternative form of the bound that allows us to make assumptions about the out-of-sample error. For this purpose, we rearrange the right hand side as follows*

$$\delta = 2\exp\left(-2\epsilon^2 N\right) \Leftrightarrow \epsilon = \sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{2.15}$$

Therefore

$$\mathbb{P}\left[|E_{in}(h) - E_{out}(h)| > \sqrt{\frac{\log\frac{2}{\delta}}{2N}}\right] \leq \delta \tag{2.16}$$

Or with probability $1-\delta$ *it holds*

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{2.17}$$

Hence, we have a probabilistic bound on the out-of-sample error that depends on the in-sample-error as well as an additional penalty term.

Up until now, we only considered a single fixed hypothesis $h \in \mathcal{H}$. However, our training algorithm picks one final hypothesis $g \in \mathcal{H}$ based on the data. In this case, Hoeffding's inequality does no longer apply. In order to see this, assume that our training algorithm is a function

$$T : (\mathcal{X} \times \mathcal{Y})^N \mapsto \mathcal{H} \tag{2.18}$$

that selects one hypothesis based on the training data. Hence, the probability that we are interested in is

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}((T(X_1,...,X_N))(X_i), X_i, Y_i) - \mathbb{E}[(\mathcal{L}(T(X_1,...,X_N))(X), X, Y)]\right| > \epsilon\right] \tag{2.19}$$

Hoeffding's inequality does no longer apply in this case because the function ($f = T$) depends the random variables. At this point, we can derive an upper bound on this probability for finite $\mathcal{H}$. Let $g = T(X_1, ..., X_N)$. Then it holds

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \mathbb{P}\left[\bigcup_{h\in\mathcal{H}} |E_{in}(h) - E_{out}(h)| > \epsilon\right] \tag{2.20}$$

$$= \sum_{h\in\mathcal{H}} \mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \tag{2.21}$$

$$\leq 2M\exp(-2\epsilon^2 N) \tag{2.22}$$

where $M = |\mathcal{H}|$. This is called the *union bound*.
Let us now return to our initial question:

Is learning feasible?

The best answer we can give is "Probably, yes". Although our derivations showed that we can never be sure to learn anything from a finite number of training examples, the upper bound on the deviation of the errors showed that we can be confident (with a certain probability) that the hypothesis g that we learned from our training data performs as well on unseen data as it does on the training set. We saw that this confidence decreases with the size M of our finite hypothesis space $\mathcal{H}$ and the required precision ϵ. However, the confidence increase with the number of training examples N. Therefore, if we have a sufficient amount of training data, we can be confident that the in-sample-error is close to the out-of-sample error.
In the following sections, we will discuss the case when $\mathcal{H}$ is infinite. This will lead us to the Vapnik-Chervonenkis theory.

2.3 Noisy Targets

In our initial definition of the learning problem, we stated that there is a fixed but unknown target function f. We implicitly assumed that the label for a data point $\mathbf{x}$ is entirely determined by this function; that is

$$y = f(\mathbf{x}) \tag{2.23}$$

However, this assumption does not necessarily hold in all practical applications. In order so see this, consider the following example: We want to build a system that automates credit-card approval based on some measurements such as the annual salary or the age of the customer. As training data, we use manually labeled decisions from different employees of the credit institution. Obviously, it might happen that the decisions of several employees differ on the same customer. Hence, for one data point there can be several labels. In this case, assuming that there is a fixed target function f is no longer valid. In order to cope with noisy targets, we have to alter the learning problem slightly. To be specific, we assume that instead of a target function, there is a *target distribution* $\mathbb{P}[y \mid \mathbf{x}]$. This means that our labels are no longer determined by a function but are drawn from this distribution; that is

$$y \sim \mathbb{P}[y \mid \mathbf{x}] \tag{2.24}$$

This is a generalization of the case when we assumed a deterministic target function. If we defined

$$\mathbb{P}[y \mid \mathbf{x}] = I(y = f(\mathbf{x})) \tag{2.25}$$

we would be in the deterministic case.
We can combine this target distribution with the assumed distribution over the data points to get one joint distribution. Thus, it holds

$$\mathbb{P}[\mathbf{x}, y] = \mathbb{P}[\mathbf{x}]\mathbb{P}[y \mid \mathbf{x}] \tag{2.26}$$

For the rest of the lecture we assume that our data is generated according to $\mathbb{P}[\mathbf{x}, y]$.
This generalization from a target function to a target distribution raises one important question.

What do we learn?

In our initial setting, we wanted to find a hypothesis that somewhat *approximates* the target function. Now that we have a target distribution, what should our hypothesis do? The confusing answer is: The *target function*. In order to see this, we define the target function in a new way. Before, the target function determined the labels for our data points. Therefore, we could say

$$E_{out}(f) = 0 \tag{2.27}$$

Or in other terms, f minimized the out-of-sample error. In the context of a target distribution this still holds. We define the target function to be the function f achieves the minimal out-of-sample error over all measurable functions.

$$E_{out}(f) = \inf_g E_{out}(g) \tag{2.28}$$

Therefore, the goal of our learning algorithm is still to find a hypothesis $g \in \mathcal{H}$ such that

$$g \approx f \tag{2.29}$$

Example 1 (Binary classification)**.** *We want to elaborate the idea of a target function in the context of a target distribution for the case of binary classification of real valued vectors. For this purpose, we use the 0-1-loss function in order to calculate the error*

$$\mathcal{L}_{0-1}(h, \mathbf{x}, y) = I(h(\mathbf{x}) \neq y) \tag{2.30}$$

where $y \in \mathcal{Y} = \{-1, 1\}$ *and* $\mathbf{x} \in \mathcal{X} = \mathbb{R}^D$*. Our target function is the function that minimizes the out-of-sample error*

$$E_{out}(f) = \mathbb{E}[\mathcal{L}_{0-1}(f, \mathbf{x}, y)] = \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}_{0-1}(f, \mathbf{x}, y)\, d\mathbb{P}[\mathbf{x}, y] \tag{2.31}$$

Obviously, f *divides the space* $\mathcal{X}$ *into two disjoint decision regions* $\mathcal{C}_1$ *and* $\mathcal{C}_{-1}$*.*

$$\int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}_{0-1}(f, \mathbf{x}, y)\, d\mathbb{P}[\mathbf{x}, y] = \int_{\mathcal{C}_1\times\mathcal{Y}} \mathcal{L}_{0-1}(f, \mathbf{x}, y)\, d\mathbb{P}[\mathbf{x}, y] + \int_{\mathcal{C}_{-1}\times\mathcal{Y}} \mathcal{L}_{0-1}(f, \mathbf{x}, y)\, d\mathbb{P}[\mathbf{x}, y] \tag{2.32}$$

We now use that $\mathcal{L}_{0-1}(f, \mathbf{x}, y) = 0$ *for* $f(\mathbf{x}) = y$ *and* $\mathcal{L}_{0-1}(f, \mathbf{x}, y) = 1$ *for* $f(\mathbf{x}) \neq y$*. Therefore, we get*

$$\int_{\mathcal{C}_1\times\mathcal{Y}} \mathcal{L}_{0-1}(f, \mathbf{x}, y)\, d\mathbb{P}[\mathbf{x}, y] + \int_{\mathcal{C}_{-1}\times\mathcal{Y}} \mathcal{L}_{0-1}(f, \mathbf{x}, y)\, d\mathbb{P}[\mathbf{x}, y] \tag{2.33}$$

$$= \int_{\mathcal{C}_1} 1\, d\mathbb{P}[\mathbf{x}, y = -1] + \int_{\mathcal{C}_{-1}} 1\, d\mathbb{P}[\mathbf{x}, y = 1] \tag{2.34}$$

$$= \mathbb{P}[\mathcal{C}_1, y = -1] + \mathbb{P}[\mathcal{C}_{-1}, y = 1] \tag{2.35}$$

$$= \mathbb{P}[\mathcal{C}_1 \mid y = -1]\mathbb{P}[y = -1] + \mathbb{P}[\mathcal{C}_{-1} \mid y = 1]\mathbb{P}[y = 1] \tag{2.36}$$

Hence, for all measurable sets $X \subset \mathcal{C}_i$ *with* $\mathbb{P}[X] > 0$ *it must hold*

$$\mathbb{P}[X \mid y = i]\mathbb{P}[y = i] \geq \mathbb{P}[X \mid y = -i]\mathbb{P}[y = -i] \tag{2.37}$$

Otherwise, there would be a function for which the out-of-sample error was smaller. However, we defined f *to be the function that minimizes the out-of-sample error. Therefore, it must hold. Using Bayes' theorem we can derive further*

$$\mathbb{P}[X \mid y = i]\mathbb{P}[y = i] \geq \mathbb{P}[X \mid y = -i]\mathbb{P}[y = -i] \tag{2.38}$$

$$\Leftrightarrow \quad \frac{\mathbb{P}[y = i \mid X]\mathbb{P}[X]}{\mathbb{P}[y = i]}\mathbb{P}[y = i] \geq \frac{\mathbb{P}[y = -i \mid X]\mathbb{P}[X]}{\mathbb{P}[y = -i]}\mathbb{P}[y = -i] \tag{2.39}$$

$$\Leftrightarrow \quad \mathbb{P}[y = i \mid X] \geq \mathbb{P}[y = -i \mid X] \tag{2.40}$$

Hence, the function f *that minimizes the out-of-sample error assigns a point to the label that has the higher posterior probability. This is called* Bayes' decision rule*. We can state* f *as follows*

$$f(\mathbf{x}) = sign(\mathbb{E}[y \mid \mathbf{x}]) = sign(\mathbb{P}[y = 1 \mid \mathbf{x}] - \mathbb{P}[y = -1 \mid \mathbf{x}]) \tag{2.41}$$

Example 2 (Regression)**.** *Let us consider a real valued target distribution* $p(y \mid \mathbf{x})$ *and the squared loss function. The out-of-sample error can be stated as follows*

$$E_{out}(f) = \int_{\mathbb{R}}\int_{\mathbb{R}} \mathcal{L}_{squared}(f, \mathbf{x}, y)p(\mathbf{x}, y)\, dy\, d\mathbf{x} = \int_{\mathbb{R}}\int_{\mathbb{R}} (f(\mathbf{x}) - y)^2 p(\mathbf{x}, y)\, dy\, d\mathbf{x} \tag{2.42}$$

If we assume f is smooth, we can solve for f using the calculus of variations. Taking the functional derivative w.r.t. to f yields.

$$\frac{\partial E_{out}}{\partial f}(f) = 2\int_{\mathbb{R}} (f(\mathbf{x}) - y)p(\mathbf{x}, y)\,dy \stackrel{!}{=} 0 \tag{2.43}$$

We can now simply solve for f. It holds

$$2\int_{\mathbb{R}} (f(\mathbf{x}) - y)p(\mathbf{x}, y)\,dy = 0 \tag{2.44}$$

$$\Leftrightarrow \quad \int_{\mathbb{R}} f(\mathbf{x})p(\mathbf{x}, y)\,dy - \int_{\mathbb{R}} yp(\mathbf{x}, y)\,dy = 0 \tag{2.45}$$

$$\Leftrightarrow \quad f(\mathbf{x}) \underbrace{\int_{\mathbb{R}} p(\mathbf{x}, y)\,dy}_{=p(\mathbf{x})} = \int_{\mathbb{R}} yp(\mathbf{x}, y)\,dy \tag{2.46}$$

$$\Leftrightarrow \quad f(\mathbf{x}) = \int_{\mathbb{R}} y \underbrace{\frac{p(\mathbf{x}, y)}{p(\mathbf{x})}}_{=p(y \mid \mathbf{x})}\,dy \tag{2.47}$$

$$\Leftrightarrow \quad f(\mathbf{x}) = \underbrace{\int_{\mathbb{R}} yp(y \mid \mathbf{x})\,dy}_{=\mathbb{E}[y \mid \mathbf{x}]} \tag{2.48}$$

$$\Leftrightarrow \quad f(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}] \tag{2.49}$$

Thus, the target function is the conditional expected value.

2.4 Error Measures Revisited

We introduced two error measures E_{in} and E_{out} that were both defined in terms of a loss function $\mathcal{L}$. We should always choose the loss function depending on the individual learning task. In this section, we want to highlight what kind of situations lead to what kind of loss functions.
In order to start the discussion, let us consider the task of binary classification. So far, we only considered the 0-1-loss. However, depending on the specific learning problem, there can be better options. For this purpose, suppose that we ought to build a machine learning system that recognizes fingerprints for authentication. There are four possible ways our system decides:

True positive A fingerprint is recognized correctly

True negative A fingerprint is rejected correctly

False positive A fingerprint is falsely recognized

False negative A fingerprint is falsely rejected

It seem intuitive that some of these options should be penalized more than others. For example, if a fingerprint is used as a method of authentication to access security relevant data, a false positive is far more severe than a false negative. If a false positive happens, a person that is not supposed to have access to the data is granted access. On the other side, if a false negative happens, a person that has access to the data is denied access. Obviously, in this case, the person could just retry.

We can encode this kind of information in a loss matrix $L \in \mathbb{R}^{2\times 2}$. Where $L_{i,j}$ is the loss induced when the target is j but the hypothesis predicted i. The loss function then is

$$\mathcal{L}(h, \mathbf{x}, y) = L_{h(\mathbf{x}),y} \tag{2.50}$$

Where we use the convention $L_{-1,-1} \equiv L_{2,2}$ etc.

Example 3. *Assume we design a loss function for the fingerprint application. A loss matrix that penalizes the false positive more than the false negatives could look like this*

$$L = \begin{pmatrix} 0 & 1 \\ 1000 & 0 \end{pmatrix} \tag{2.51}$$

Hence, if a false positive happens, the loss is 1000, and if a false negative happens, the loss is only 1.

Unfortunately, it is not always possible to manually define the loss for individual outcomes. In these cases, one often follows one of two strategies:

1. Use a *plausible* loss function.
2. Use a *friendly* loss function.

Because these to choices are very popular, we want to elaborate.
A *plausible* loss function is used if we make further assumptions about the nature of the data. For this purpose, let us consider the case of regression. We can make the plausible assumption that our target distribution is a Gaussian distribution centered around the "true" target value; that is

$$\mathbb{P}[y \mid \mathbf{x}] = \mathcal{N}(y \mid \mu_{\mathbf{x}}, \sigma^2) \tag{2.52}$$

where we assume a fixed noise level σ^2.
One way to interpret the assumption is that there is a true target value that is measured with Gaussian noise. We now derive a loss function under this assumption for the case of linear regression. The idea is that we want to *estimate* $\tilde{\mathbf{w}}$ such that the likelihood that all data points have been generated from the distribution $\mathcal{N}(y \mid \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}, \sigma)$ is maximal. The statistical framework of maximum likelihood estimation allows us to derive the error function. The goal is to maximize the joint likelihood with respect to $\tilde{\mathbf{w}}$.

$$\tilde{\mathbf{w}}^* = \arg\max_{\tilde{\mathbf{w}}} \prod_{i=1}^{N} \mathcal{N}(y_i \mid \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i, \sigma^2) \tag{2.53}$$

We know from calculus that applying a monotonic transformation does not change the location of the maxima. Therefore, this is equivalent to minimizing the negative logarithm.

$$\arg\max_{\tilde{\mathbf{w}}} \prod_{i=1}^{N} \mathcal{N}(y_i \mid \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i, \sigma^2) = \arg\min_{\tilde{\mathbf{w}}} -\ln\left(\prod_{i=1}^{N} \mathcal{N}(y_i \mid \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i, \sigma^2)\right) \tag{2.54}$$

$$= \arg\min_{\tilde{\mathbf{w}}} -\sum_{i=1}^{N} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i - y_i)^2}{\sigma^2} \tag{2.55}$$

$$= \arg\min_{\tilde{\mathbf{w}}} -\sum_{i=1}^{N} -\frac{(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i - y_i)^2}{\sigma^2} \tag{2.56}$$

$$= \arg\min_{\tilde{\mathbf{w}}} \sum_{i=1}^{N} (\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i - y_i)^2 \tag{2.57}$$

Hence, the plausible assumption about the Gaussian nature of the target distribution directly leads to the squared loss function.
The other popular choice that we highlighted is the use of a *friendly* loss function. By friendly, we mean that the loss function has some properties that make handling it easier. For example, a friendly loss function could be a loss function that leads to a close form solution of the optimization problem. We already saw that the squares loss function allows exactly this. Another convenient property is convexity. Because convexity implies that there is a unique global minimum, standard optimization algorithms are able to optimize the function efficiently. A very popular convex loss function for the case of binary classification is the so called *cross entropy error*. The error is often used for *logistic regression* and neural network training.

2.5 The Vapnik-Chervonenkis Inequality

In this section, we refine the union bound that we derived in section 2.2. As before, we only treat the case where $\mathcal{L} \equiv \mathcal{L}_{0-1}$. Hence, we only discuss the case of binary classification ($\mathcal{Y} = \{-1, 1\}$). Our goal is to bound the probability that the in-sample error $E_{in}(g)$ of a learned hypothesis $g \in \mathcal{H}$ deviates more than ϵ from the out-of-sample error. Because g is learned from the sample, we could not apply Hoeffding's inequality directly. Instead, we bounded the probability as follows

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \mathbb{P}\left[\sup_{h \in \mathcal{H}} |E_{in}(h) - E_{out}(h)| > \epsilon\right] \tag{2.58}$$

In the final step we assumed $|\mathcal{H}| = M$ to be finite which yields

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \sum_{h \in \mathcal{H}} \mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2M \exp(-2\epsilon^2 N) \tag{2.59}$$

where N was the sample size.
While this seems like a good result, it has several problems in practice. Firstly, in most cases the hypothesis set is not finite. For example, consider the set of linear classifiers. Because each classifier corresponds to a weight vector $\tilde{\mathbf{w}} \in \mathbb{R}^{D+1}$, we have $|\mathcal{H}| = |\mathbb{R}^{D+1}|$. Obviously, even in this simple case we have uncountably many hypotheses. Secondly, we bounded the maximum error over all hypotheses by the sum over all hypotheses. In most cases, this is a very loose bound. In order to see this, consider a hypothesis set that only contains two classifiers $\mathcal{H} = \{h_1, h_2\}$

$$h_1(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \tag{2.60}$$

$$h_2(\mathbf{x}) = \text{sign}((\tilde{\mathbf{w}} + \epsilon \mathbf{e}_1)^T \tilde{\mathbf{x}}) \tag{2.61}$$

where $\mathbf{e}$ is some unit vector and $0 < \epsilon \ll 1$. Obviously, h_1 and h_2 agree on most of the data points, which means that $\mathbb{P}[h_1(\mathbf{x}) \neq h_2(\mathbf{x})]$ is very small. Therefore, the probability

$$\mathbb{P}\left[|E_{in}(h_1) - E_{out}(h_1)| > \epsilon, |E_{in}(h_2) - E_{out}(h_2)| \leq \epsilon\right] \tag{2.62}$$

is also very small. This means that the cases where the in-sample error is not a good representation of the out-of-sample error are approximately the same for h_1 and h_2. Therefore

$$\mathbb{P}\left[\sup_{h \in \mathcal{H}} |E_{in}(h) - E_{out}(h)| > \epsilon\right] \approx \mathbb{P}[|E_{in}(h_1) - E_{out}(h_1)| > \epsilon] \tag{2.63}$$

However, in our derivation of the union bound we ignored such *overlaps*. If we apply the union bound to this case, we get

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |E_{in}(h) - E_{out}(h)| > \epsilon\right] \lesssim 2\mathbb{P}[|E_{in}(h_1) - E_{out}(h_1)| > \epsilon] \tag{2.64}$$

This gives us some intuition that the bound is not very tight.
Both of these problems are caused by the additional factor $M = |\mathcal{H}|$ that captures the size of the hypothesis set. Therefore, our qualitative goal is to *replace* M with a quantity that resolves the problems. To be specific, we will replace M with a function $m(N)$ that grows polynomially in N. It is easy to see why this will resolve the two problems.

1. Even for infinite hypothesis sets, $m(N)$ is finite. Hence, infinite hypothesis sets do not render the bound useless.

2. If $m(N)$ grows polynomially in N, then we do not lose the convergence speed of the negative exponential. Hence, for large N, the bound is very small regardless of the hypothesis set[2].

In the following discussion, we derive the so called *Vapnik-Chervonenkis Inequality* that is a generalization of the union bound to cases where $\mathcal{H}$ is infinite. We start this discussion by only considering the training data $\mathbf{x}_1, ..., \mathbf{x}_N$. Regardless of which hypothesis we consider, there is only a finite number of ways these data points can be classified. Each of these ways defines a function

$$d : \{\mathbf{x}_1, ..., \mathbf{x}_N\} \mapsto \{-1, 1\} \tag{2.65}$$

which we call a *dichotomy*. Obviously, each hypothesis h induces a dichotomy as follows

$$d_{h,\mathbf{x}1,...,\mathbf{x}_N}(\mathbf{x}_i) := h(\mathbf{x}_i) \tag{2.66}$$

You can think of a dichotomy as the effect a hypothesis has on the finitely many training points disregarding the entire input space $\mathcal{X}$.

Example 4. *Assume that we have two training samples* $\mathbf{x}_1 = -3, \mathbf{x}_2 = 2 \in \mathbb{R}$ *and a hypothesis*

$$h : \mathbb{R} \mapsto \{-1, 1\}, \mathbf{x} \mapsto sign(\mathbf{x}) \tag{2.67}$$

Then, the dichotomy $d_{h,\mathbf{x}_1,\mathbf{x}_2}$ *that is induced by* h *is*

$$d_{h,\mathbf{x}_1,\mathbf{x}_2}(-3) = -1 \tag{2.68}$$
$$d_{h,\mathbf{x}_1,\mathbf{x}_2}(2) = 1 \tag{2.69}$$

For a fixed training set $\mathbf{x}_1, ..., \mathbf{x}_N$ we define

$$\mathcal{H}(\mathbf{x}_1, ..., \mathbf{x}_N) := \{d_{h,\mathbf{x}_1,...,\mathbf{x}_N} : h \in \mathcal{H}\} \tag{2.70}$$

Hence, $\mathcal{H}(\mathbf{x}_1, ..., \mathbf{x}_N)$ is the set of all possible dichotomies that can be induced by the hypotheses on the specified points.

[2]However, the hypothesis set has to have a finite break point.

Example 5. *Reconsider the case where we have two training examples* $\mathbf{x}_1 = -3, \mathbf{x}_2 = 2 \in \mathbb{R}$. *We choose our hypothesis set to be*

$$\mathcal{H} = \{\mathbf{x} \mapsto sign(\mathbf{x} - a) : a \in \mathbb{R}\} \tag{2.71}$$

Hence, a hypothesis $h \in \mathcal{H}$ *is always of the form*

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \geq a \\ -1 & \text{otherwise} \end{cases} \tag{2.72}$$

where we agreed upon the convention $sign(0) = 1$.
We can construct the set of dichotomies $\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2)$ *by considering the different values of the parameter* a.

- *If* $a \in (-\infty, -3]$, *then* $h(\mathbf{x}_1) = 1$ *and* $h(\mathbf{x}_2) = 1$. *Therefore, all these hypotheses induce the same dichotomy*

$$d_1 := d_{h,\mathbf{x}_1,\mathbf{x}_2}(\mathbf{x}) = 1 \tag{2.73}$$

- *If* $a \in (-3, 2]$, *then* $h(\mathbf{x}_1) = -1$ *and* $h(\mathbf{x}_2) = 1$. *Therefore, all these hypothesis induce the dichotomy*

$$d_2 := d_{h,\mathbf{x}_1,\mathbf{x}_2}(\mathbf{x}) = \begin{cases} -1 & \text{if } \mathbf{x} = \mathbf{x}_1 \\ 1 & \text{if } \mathbf{x} = \mathbf{x}_2 \end{cases} \tag{2.74}$$

- *Finally, consider the case* $a \in (2, \infty)$. *In this cases, we get* $h(\mathbf{x}_1) = h(\mathbf{x}_2) = -1$. *Therefore, the dichotomy that is induced by these hypotheses is*

$$d_3 := d_{h,\mathbf{x}_1,\mathbf{x}_2}(\mathbf{x}) = -1 \tag{2.75}$$

Hence, our set of dichotomies is

$$\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2) = \{d_1, d_2, d_3\} \tag{2.76}$$

Note that although $\mathcal{H}$ *is infinite, the set of dichotomies is finite.*

Example 5 shows clearly that the set of dichotomies depends on the chosen data points. If we chose $\mathbf{x}_1 = \mathbf{x}_2$, then we would only have found two different dichotomies. We can remove this dependence on the data points by simply taking the supremum over all possible N points. This leads us to the definition of the so called *growth function.*

Definition 2.5.1 (Growth function)**.** *Let* $\mathcal{H}$ *be a hypothesis set. The* growth function $m_{\mathcal{H}}(H)$ *is the maximum number of dichotomies on* N *points.*

$$m_{\mathcal{H}}(N) := \sup_{\mathbf{x}_1,...,\mathbf{x}_N \in \mathcal{H}} |\mathcal{H}(\mathbf{x}_1, ..., \mathbf{x}_N)| \tag{2.77}$$

Obviously, all growth functions are bounded by 2^N.
In the following examples, we consider hypothesis sets for which we can state the growth function explicitly.

Example 6 (Positive rays)**.** *In example 5 we considered a hypothesis set of the form*

$$\mathcal{H} = \{\mathbf{x} \mapsto sign(\mathbf{x} + a) : a \in \mathbb{R}\} \tag{2.78}$$

Each hypothesis classifies a point depending on whether or not it is greater than the parameter a.
Let $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}$ be our data points. In order to maximize the number of dichotomies, we assume they are pairwise distinct (i.e. $\mathbf{x}_i \neq \mathbf{x}_j$ for $i \neq j$). Without loss of generality we can further assume $\mathbf{x}_i < \mathbf{x}_j$ for $i < j$ (i.e. the points are ordered).
Then for each interval $(\mathbf{x}_i, \mathbf{x}_{i+1}], i = 0, ..., N$ we get one dichotomy where we define $\mathbf{x}_0 = -\infty$ and $\mathbf{x}_{N+1} = \infty$. Let $a \in (\mathbf{x}_i, \mathbf{x}_{i+1}]$, $i \in \{0, ..., N\}$. Then

$$h(\mathbf{x}_j) = -1 \text{ for } j < i \tag{2.79}$$

$$h(\mathbf{x}_j) = 1 \text{ for } j \geq i \tag{2.80}$$

There are $N + 1$ possible choices of i. Therefore, there are $N + 1$ intervals that induce one dichotomy each. Hence, the growth function is

$$m_{\mathcal{H}}(N) = N + 1 \tag{2.81}$$

One observation that we will discuss later is that $m_{\mathcal{H}} = o(2^N)$.

Example 7 (Convex sets)**.** *In this example, our hypothesis set is based on* convex sets *in $\mathbb{R}^2$.*

Definition 2.5.2 (Convexity)**.** *A set $X \subset \mathbb{R}^D$ is* convex *if for all points $\mathbf{x}_1, \mathbf{x}_2 \in X$*

$$\{\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 : \lambda \in [0, 1]\} \subset X \tag{2.82}$$

Each classifier is associated with a convex set X. It assigns a point to class 1 if it is in X. Formally

$$\mathcal{H} = \{sign(2\chi_X(\mathbf{x}) - 1) : X \subset \mathbb{R}^2, X \text{ convex}\} \tag{2.83}$$

where χ_X is the so called characteristic function *of X*

$$\chi_X(\mathbf{x}) = I(\mathbf{x} \in X) \tag{2.84}$$

We can maximize the number of dichotomies for a set of N pairwise distinct points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^2$ if we arrange the points on a circle centered at the origin. That means

$$\|\mathbf{x}_i\|_2 = r, i = 1, ..., N \tag{2.85}$$

Assuming this arrangement, it turns out that we can classify the points in all possible ways. Let d be some dichotomy on $\mathbf{x}_1, ..., \mathbf{x}_N$. We can find a hypothesis $h \in \mathcal{H}$ with

$$d_{h,\mathbf{x}_1,...,\mathbf{x}_N}(\mathbf{x}_i) = d(\mathbf{x}_i) \tag{2.86}$$

using the convex hull *of all points with $d(\mathbf{x}) = 1$.*

Definition 2.5.3 (Convex hull)**.** *Let $P \subset \mathbb{R}^D$. The* convex hull *$C(P)$ is the smallest convex set that contains P.*

$$C(P) = \bigcap_{X \subset \mathbb{R}^D : X convex, P \subset X} X \tag{2.87}$$

It is clear that for

$$h = \chi_{C(\{d=1\})} \tag{2.88}$$

it holds $h(\mathbf{x}) = 1$ *for* $\mathbf{x} \in \{d = 1\}$[3]*. Assume there was a point* $\mathbf{x}_i \notin \{d = 1\}$ *with* $h(\mathbf{x}_i) = 1$*. Then,* $\mathbf{x}_i$ *must lie in the convex hull of the other points.* $\mathbf{x}_i$ *cannot be an inner point of* $C(P)$[4]*. Because if this was the case, we would get*

$$\|\mathbf{x}_i\|_2 < \sup_{\mathbf{x}\in C(P)} \|\mathbf{x}\|_2 = \max_{\mathbf{x}\in C(P)} \|\mathbf{x}\| = r \tag{2.89}$$

which violates our assumption $\|\mathbf{x}_i\|_2 = r$*. Therefore,* $\mathbf{x}_i$ *must lie on the boundary which means that* $\mathbf{x}_i$ *lies on a line segment that connects two neighbouring points* $\mathbf{x}_j, \mathbf{x}_{j+1}$*. Thus, we get*

$$\|\mathbf{x}_i\|_2^2 = \|\lambda\mathbf{x}_j + (1-\lambda)\mathbf{x}_{j+1}\|_2^2 \tag{2.90}$$

with $\lambda \in (0, 1)$*. We can derive further*

$$\begin{aligned}
\|\mathbf{x}_i\|_2^2 &= \|\lambda\mathbf{x}_j + (1-\lambda)\mathbf{x}_{j+1}\|_2^2 && (2.91)\\
&= \lambda^2\|\mathbf{x}_j\|_2^2 + 2\lambda(1-\lambda)\mathbf{x}_j^T\mathbf{x}_{j+1} + (1-\lambda)^2\|\mathbf{x}_{j+1}\|_2^2 && (2.92)\\
&= \lambda^2 r^2 + 2\lambda(1-\lambda)r^2\cos(\mathbf{x}_j, \mathbf{x}_{j+1}) + (1-\lambda)^2 r^2 && (2.93)
\end{aligned}$$

where we used $\|\mathbf{x}_j\|_2^2 = \|\mathbf{x}_{j+1}\|_2^2 = r^2$ *and* $\mathbf{x}_j^T\mathbf{x}_{j+1} = \|\mathbf{x}_j\|_2\|\mathbf{x}_{j+1}\|_2\cos(\mathbf{x}_j, \mathbf{x}_{j+1})$*. Because the points* $\mathbf{x}_j$ *and* $\mathbf{x}_{j+1}$ *lie on a circle centered at the origin and are distinct, we know* $\cos(\mathbf{x}_j, \mathbf{x}_{j+1}) < 1$ *and thus we get*

$$\begin{aligned}
\lambda^2 r^2 + 2\lambda(1-\lambda)r^2\cos(\mathbf{x}_j, \mathbf{x}_{j+1}) + (1-\lambda)^2 r^2 &< \lambda^2 r^2 + 2\lambda(1-\lambda)r^2 + (1-\lambda)^2 r^2 && (2.94)\\
&= r^2(\lambda^2 + 2\lambda - 2\lambda^2 + \lambda^2 - 2\lambda + 1) && (2.95)\\
&= r^2 && (2.96)
\end{aligned}$$

which is a contradiction. Therefore, the hypothesis $h = \chi_{C(\{d=1\})}$ *induces the dichotomy* d*. We now proved that if the points lie on a circle around the origin, for each possible dichotomy we can find a hypothesis that classifies the points according to this dichotomy. Thus, the growth function is*

$$m_{\mathcal{H}}(N) = 2^N \tag{2.97}$$

We mentioned, that we want to replace the quantity M in the union bound with a function $m(N)$ that grows polynomially in N. Obviously, $m_{\mathcal{H}}(N)$ could be a candidate. We already saw that in the case where $\mathcal{H}$ is the set of positive intervals $m_{\mathcal{H}}(N)$ grows polynomially. We now prove a fundamental property of the growth function that allows us to determine whether or not the function grows polynomially. For this purpose, we introduce the notion of *break points*.

Definition 2.5.4 (Break point)**.** *Let* $m_{\mathcal{H}}(N)$ *be a growth function. The* break point k *of* $m_{\mathcal{H}}(N)$ *is the smallest* $k \in \mathbb{N}$ *such that*

$$m_{\mathcal{H}}(k) < 2^k \tag{2.98}$$

If there is no k *for which* $m_{\mathcal{H}}(k) < 2^k$*, then we say that the break point is infinite ('*$k = \infty$*').*

[3] $\{d = 1\}$ is the *level set*. $\{d = 1\} := \{\mathbf{x} : d(\mathbf{x}) = 1\}$.

[4] $\mathbf{x}$ is an inner point of C if C contains an open ϵ neighbourhood of $\mathbf{x}$.

Remark 5 (Shattering)**.** *Another way to introduce the break point of a growth function is to consider the smallest number for which a data set cannot be* shattered *by $\mathcal{H}$.*

Definition 2.5.5 (Shattering)**.** *Let $\mathcal{H}$ be a hypothesis space. We say $\mathcal{H}$* shatters *a data set $\mathbf{x}_1, ..., \mathbf{x}_N$ if the data points can be classified in all possible 2^N ways.*

Hence, $\mathcal{H}$ shatters a data set $\mathbf{x}_1, ..., \mathbf{x}_N$ if for each dichotomy $d : \{\mathbf{x}_1, ..., \mathbf{x}_N\} \mapsto \{-1, 1\}$ there is a hypothesis $h \in \mathcal{H}$ with

$$d = d_{h,\mathbf{x}_1,...,\mathbf{x}_N} \tag{2.99}$$

Using the notion of shattering, we can define a break point as follows.

Corollary 1. *The break point k of the $m_{\mathcal{H}}(N)$ is the smallest k such that no data set of k points can be shattered by $\mathcal{H}$.*

Before we determine the break point for the two examples we discussed before, we explain the break point in terms of a two player game. If we have a *winning strategy* for a certain number k, then we know that the break point is at least k. The game works as follows:

Opponent Picks the number k

We Pick the points $\mathbf{x}_1, ..., \mathbf{x}_k$

Opponent Picks a dichotomy $d : \{\mathbf{x}_1, ..., \mathbf{x}_k\} \mapsto \{-1, 1\}$

We Pick a hypothesis h

Winner If $h(\mathbf{x}_i) = d(\mathbf{x}_i)$ for all $i = 1, ..., k$, then we win. Otherwise, the opponent wins.

It is important to highlight that we need a *winning strategy*. Only winning some rounds for a certain k, does not imply that the break point is at least k.
Let us now return to the two examples that we discussed before and determine the break point one time using the definition and one time using the game.

Example 8 (Positive rays)**.** *We established that the growth function for the hypothesis set of positive rays is given by*

$$m_{\mathcal{H}}(N) = N + 1 \tag{2.100}$$

The break point is the smallest k such that $m_{\mathcal{H}}(k) < 2^k$. We can subsequently try different values for k in order to determine the break point.

$$k = 1 \Rightarrow m_{\mathcal{H}}(1) = 1 = 2^1 \tag{2.101}$$

$$k = 2 \Rightarrow m_{\mathcal{H}}(2) = 3 < 2^2 = 4 \tag{2.102}$$

Hence, the break point is 2.
Let us now play the game for the cases $k = 1$ and $k = 2$. For $k = 1$, we develop the winning strategy and for $k = 2$ we show one run that we cannot win regardless of our strategy. We start with $k = 1$. A winning strategy looks as follows.

Opponent *Picks the number $k = 1$*

We *Pick a random point $\mathbf{x}_1 \in \mathbb{R}$*

Opponent *Picks a dichotomy $d : \{\mathbf{x}_1\} \mapsto \{-1, 1\}$*

We *Pick a hypothesis h as follows: If $d(\mathbf{x}_1) = -1$, then we choose $h(\mathbf{x}) = sign(\mathbf{x} - \mathbf{x}_1 - 1)$. If $d(\mathbf{x}_1) = 1$, then we choose $h(\mathbf{x}) = sign(\mathbf{x} - \mathbf{x}_1 + 1)$.*

Winner *Because $d(\mathbf{x}_1) = h(\mathbf{x}_1)$ we win regardless of what dichotomy our opponent chose.*

Therefore, the break point is at least 1. Now, we consider a run that we cannot win.

Opponent *Picks the number $k = 2$*

We *Pick two random points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}$ with $\mathbf{x}_1 \neq \mathbf{x}_2$. We choose them such that they are distinct because otherwise we cannot win if the opponent picks a dichotomy for which $d(\mathbf{x}_1) \neq d(\mathbf{x}_2)$*

Opponent *Let $\mathbf{x}_1 < \mathbf{x}_2$. Our opponent picks a dichotomy d with $d(\mathbf{x}_1) = 1$ and $d(\mathbf{x}_2) = -1$*

We *Because $\mathbf{x}_1 < \mathbf{x}_2$, for each hypothesis h it holds $h(\mathbf{x}_1) = 1 \Rightarrow h(\mathbf{x}_2) = 1$. Hence, we can pick any hypothesis.*

Winner *We were not able to pick a hypothesis that induces the dichotomy chosen by the opponent. Hence, we lose this round.*

Because we could not win a round, we do not have a winning strategy for $k = 2$. Since we had a winning strategy for $k = 1$, the break point is $k = 2$.

Example 9 (Convex sets)**.** *We established that for the hypothesis set that is based on convex sets we have*

$$m_{\mathcal{H}}(N) = 2^N \tag{2.103}$$

Therefore, there is no break point. According to our definition, we say the break point is infinite. This means that we are supposed to have a winning strategy for all choices of k. Indeed, we already saw the winning strategy.

Opponent *Picks a number $k \in \mathbb{N}$*

We *Pick a random distinct points $\mathbf{x}_1, ..., \mathbf{x}_k \in \mathbb{R}^2$ on a circle around the origin*

Opponent *Picks a dichotomy $d : \{\mathbf{x}_1\} \mapsto \{-1, 1\}$*

We *Pick the following hypothesis*

$$h(\mathbf{x}) = \chi_{C(\{d=1\})}(\mathbf{x}) \tag{2.104}$$

Winner *We already proved that $h(\mathbf{x}_i) = d(\mathbf{x}_i)$. Therefore, we win.*

Using the notion of break points, we can prove the following lemma.

Lemma 1. *Let $\mathcal{H}$ be a hypothesis set with a finite break point k. Then for all $N \in \mathbb{N}$*

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k} \begin{pmatrix} N \\ i \end{pmatrix} \tag{2.105}$$

Proof. We prove the proposition by deriving a recursively defined upper bound on the growth function. First of all, let us consider a bound on

$$m_{\mathcal{H}}(N) = \sup_{\mathbf{x}_1,...,\mathbf{x}_N} |\mathcal{H}(\mathbf{x}_1, ..., \mathbf{x}_N)| \tag{2.106}$$

This quantity must be less or equal than the maximum number of dichotomies such that no k points are classified in all possible 2^k ways. We call this upper bound $B(N, k)$. Thus,

$$m_{\mathcal{H}}(N) \leq B(N, k) \tag{2.107}$$

We can bound $B(N, k)$ recursively. Assume we know the points $\mathbf{x}_1, ..., \mathbf{x}_N$ that give rise to the maximum number of dichotomies S. We can split the dichotomies into two three sets depending on the class label of $\mathbf{x}_N$.

$$S_1 := \{d \in S : \nexists \tilde{d} \in S \backslash \{d\} \text{ with } d(\mathbf{x}_i) = \tilde{d}(\mathbf{x}_i) \text{ for } i = 1, ..., N-1\} \tag{2.108}$$

$$S_2^+ := \{d \in S : d(\mathbf{x}_N) = 1 \wedge \exists \tilde{d} \in S \backslash \{d\} \text{ with } d(\mathbf{x}_i) = \tilde{d}(\mathbf{x}_i) \text{ for } i = 1, ..., N-1\} \tag{2.109}$$

$$S_2^- := \{d \in S : d(\mathbf{x}_N) = -1 \wedge \exists \tilde{d} \in S \backslash \{d\} \text{ with } d(\mathbf{x}_i) = \tilde{d}(\mathbf{x}_i) \text{ for } i = 1, ..., N-1\} \tag{2.110}$$

The three sets contain the following dichotomies:

- S_1 is the set of all dichotomies d for which there is no other dichotomy $\tilde{d}$ such that d and $\tilde{d}$ only disagree on $\mathbf{x}_N$
- S_2^+ is the set of all dichotomies d for which there is another dichotomy $\tilde{d}$ such that d and $\tilde{d}$ only disagree on $\mathbf{x}_N$ and d assigns $\mathbf{x}_N$ to class 1
- S_2^+ is the set of all dichotomies d for which there is another dichotomy $\tilde{d}$ such that d and $\tilde{d}$ only disagree on $\mathbf{x}_N$ and d assigns $\mathbf{x}_N$ to class -1

Because the three sets are distinct, we get

$$B(N, k) = |S_1| + |S_2^+| + |S_2^-| \tag{2.111}$$

Furthermore, we see that $|S_2^+| = |S_2^-|$. With $\alpha := |S_1|$ and $\beta := |S_2^+| (= |S_2^-|)$ we get

$$B(N, k) = \alpha + 2\beta \tag{2.112}$$

We now derive upper bounds on $\alpha + \beta$ and β.
Let us first consider the set $S_1 \cup S_2^+$. This is a set of dichotomies such that no k of $N-1$ points are classified in all 2^k possible ways. Hence, the number of dichotomies in this set must be less or equal to the **maximum** number of dichotomies such that no k of $N-1$ points are classified in all ways. This quantity is captured by $B(N-1, k)$. Thus, we get

$$|S_1 \cup S_2^+| = |S_1| + |S_2^+| = \alpha + \beta \leq B(N-1, k) \tag{2.113}$$

Let us now consider the set S_2^-. This is a set of dichotomies such that no $k-1$ of $N-1$ points are classified in all possible ways. In order to see this, assume that there existed a selection of points $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_{k-1}}$ such that for each dichotomy d on these points there existed a dichotomy $\tilde{d} \in S_2^-$ with $d(\mathbf{x}_{i_j}) = \tilde{d}(\mathbf{x}_{i_j})$. Because for each of these dichotomies $\tilde{d}$ there exists another one in S_2^+ that only disagrees with $\tilde{d}$ on the first $N-1$ points, there would be a selection of k points that are shattered by $S_2^+ \cup S_2^-$. However, this contradicts our assumption that we consider the

largest set of dichotomies S such that no k of N points are shattered. Thus, no $k-1$ of $N-1$ points are shattered by S_2^-. Thus, we get

$$|S_2^-| = \beta \leq B(N-1, k-1) \tag{2.114}$$

Putting the two bounds together yields

$$m_{\mathcal{H}}(N) \leq B(N,k) \leq B(N-1,k) + B(N-1,k-1) \tag{2.115}$$

Finally, we have to prove that $B(N,k) \leq \sum_{i=0}^{k} \left(\begin{array}{c} N \\ i \end{array} \right)$. We do this using two induction steps.

Base case $N = 1$
If we only have a single point, there are at most 2 possible ways we can classify the point. If $k = 1$, we have

$$B(1,1) = 1 \leq \sum_{i=0}^{1} \left(\begin{array}{c} 1 \\ i \end{array} \right) = 2 \tag{2.116}$$

And if $k > 1$, we get

$$B(1,k) = 2 \leq \sum_{i=0}^{1} \left(\begin{array}{c} 1 \\ i \end{array} \right) = 2 \tag{2.117}$$

Base case $k = 1$
If the break point is k, we cannot classify a single point in all possible ways. Hence, for each N, all points have to be assigned to the same class. Therefore, we get

$$B(N,1) = 1 \leq \sum_{i=0}^{1} \left(\begin{array}{c} N \\ i \end{array} \right) = 1 + N \tag{2.118}$$

Hypothesis
Let the proposition be true for a fixed $k, N \in \mathbb{N}$.
For the inductive step, we use the following recursive definition of the binomial coefficients.

Corollary 2.

$$\left(\begin{array}{c} N \\ i \end{array} \right) = \left(\begin{array}{c} N-1 \\ i-1 \end{array} \right) + \left(\begin{array}{c} N-1 \\ i \end{array} \right) \tag{2.119}$$

Inductive step: $N \to N+1$

$$B(N+1,k) \leq B(N,k) + B(N,k-1) \tag{2.120}$$

$$\overset{hyp.}{\leq} \sum_{i=0}^{k} \left(\begin{array}{c} N \\ i \end{array} \right) + \sum_{i=0}^{k-1} \left(\begin{array}{c} N \\ i \end{array} \right) \tag{2.121}$$

$$= 1 + \sum_{i=1}^{k} \left(\begin{array}{c} N \\ i \end{array} \right) + \sum_{i=1}^{k} \left(\begin{array}{c} N \\ i-1 \end{array} \right) \tag{2.122}$$

$$= 1 + \sum_{i=1}^{k} \left(\begin{array}{c} N+1 \\ i \end{array} \right) \tag{2.123}$$

$$= \sum_{i=0}^{k} \left(\begin{array}{c} N+1 \\ i \end{array} \right) \tag{2.124}$$

Inductive step: $k \to k+1$

$$B(N, k+1) \leq B(N-1, k+1) + B(N-1, k) \tag{2.125}$$

$$\overset{hyp.}{\leq} \sum_{i=0}^{k+1} \binom{N-1}{i} + \sum_{i=0}^{k} \binom{N-1}{i} \tag{2.126}$$

$$= 1 + \sum_{i=1}^{k+1} \binom{N-1}{i} + \sum_{i=1}^{k+1} \binom{N-1}{i-1} \tag{2.127}$$

$$= 1 + \sum_{i=1}^{k+1} \binom{N}{i} \tag{2.128}$$

$$= \sum_{i=0}^{k+1} \binom{N}{i} \tag{2.129}$$

□

Lemma 1 is particularly interesting because it implies that $m_{\mathcal{H}}(N)$ grows polynomially in N. It is easy to see that

$$m_{\mathcal{H}}(N) = \mathcal{O}(N^{k-1}) \tag{2.130}$$

if $k < \infty$ is the **fixed** break point. Therefore, $m_{\mathcal{H}}$ could be the desired substitution for M that we were looking for. Using this result, we will now state the VC inequality.

Theorem 2.5.6 (Vapnik-Chervonenkis Inequality)**.** *Let $\mathcal{H}$ be a class of hypothesis, let $X_1, ..., X_N$ be i.i.d. sampled, and let $g \in \mathcal{H}$ be a hypothesis that is chosen based on the samples. Then for all $\epsilon > 0$ it holds*

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 4 m_{\mathcal{H}}(2N) \exp\left(-\frac{1}{8}\epsilon^2 N\right) \tag{2.131}$$

The difficult task in proving theorem 2.5.6 is that $E_{out}(g)$ depends on the entire input space as well as on the probability measure $\mathbb{P}[\mathbf{x}, y]$. Vapnik found a way to work around this problem. The idea is to *replace* $E_{out}(g)$ with another E^*_{in} which is based on a second independent set of samples. This argument is called *symmetrization*. Intuitively, if E_{in} deviates significantly from E_{out}, then for a large sample size E_{in} should also deviate significantly from E^*_{in}. The following lemma, formalizes this idea.

Lemma 2 (Symmetrization)**.** *For all $\epsilon > 0$ with $N\epsilon^2 \geq 2$,*

$$\mathbb{P}\left[\sup_{h \in \mathcal{H}} |E_{in}(h) - E_{out}(h)| > \epsilon\right] \leq 2\mathbb{P}\left[\sup_{h \in \mathcal{H}} |E_{in}(h) - E^*_{in}(h)| > \frac{\epsilon}{2}\right] \tag{2.132}$$

Proof. A proof can be found in [LAM] □

Using this argument, one can prove the VC-inequality. A detailed proof can be found in [LAM].

2.6 The VC-Dimension

In the previous section we saw that we can bound the probability that the in-sample error deviates significantly from the out-of-sample error using the growth function $m_{\mathcal{H}}(N)$ of the hypothesis set $\mathcal{H}$. One important observation that we made was that if the growth function has

a finite break point k, then it grows polynomially in N. This is beneficial because in this case the negative exponential dominates the bound for large N. It is important to note that the growth function as well as the break point are properties of the hypothesis set $\mathcal{H}$. Therefore, the VC-inequality does not depend on the properties of the learning problem such as the target function f and the probability measure $\mathbb{P}$ on $\mathcal{X} \times \mathcal{Y}$. This is a desirable result because this shows that although we have no control over the learning problem, we can control the error by choosing an appropriate hypothesis set.
We already saw that we can characterize the break point of a growth function as the smallest number k such that for any set of k points there is a dichotomy that we cannot express using any of our hypotheses. Another way to measure the power of a hypothesis set is by using the *VC-dimension*.

Definition 2.6.1 (VC-Dimension)**.** *Let $\mathcal{H}$ be a hypothesis set. The* VC-dimension $d_{VC}(\mathcal{H})$ *is the largest number such that there is a set of $d_{VC}(\mathcal{H})$ points that is shattered by $\mathcal{H}$.*

Obviously, the following relation between the break point and the VC-dimension holds.

$$k = d_{VC} + 1 \tag{2.133}$$

In this section, we want to explore what the VC-dimension measures. Our goal is to develop an intuition for this quantity. We start the discussion by finding the VC-dimension of the set of linear classifiers (Perceptrons).

Theorem 2.6.2. *The hypothesis set of linear classifiers in $\mathbb{R}^D$ has VC-dimension $D + 1$.*

Proof. We prove the proposition using two steps. First, we show that $d_{VC} \leq D + 1$. Then, we show that $d_{VC} \geq D + 1$. This implies $d_{VC} = D + 1$.
Show $d_{VC} \geq D + 1$
We can show this by finding a set of $D + 1$ points that can be shattered. We choose the points as follows:

$$\tilde{\mathbf{x}}_1 = (0, ..., 0, 1)^T \tag{2.134}$$

$$\tilde{\mathbf{x}}_i = (\delta_{1,i}, ..., \delta_{D,i}, 1)^T \text{ for } i = 2, ..., D + 1 \tag{2.135}$$

Note that the last (or the first) entry of each vector is 1 because we consider homogeneous vectors. We can stack these points into a matrix $\mathbf{X}$.

$$\mathbf{X} = (\tilde{\mathbf{x}}_1 \dots \tilde{\mathbf{x}}_{D+1})^T \tag{2.136}$$

Given any dichotomy d on $\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_{D+1}$, we want to determine the weight vector $\tilde{\mathbf{w}}$ such that

$$\text{sign}(\mathbf{X}\tilde{\mathbf{w}}) = \underbrace{(d(\mathbf{x}_1), ..., d(\mathbf{x}_{D+1}))^T}_{=\mathbf{y}} \tag{2.137}$$

Obviously it holds

$$\mathbf{X}\tilde{\mathbf{w}} = \mathbf{y} \Rightarrow \text{sign}(\mathbf{X}\tilde{\mathbf{w}}) = \mathbf{y} \tag{2.138}$$

Because $\det(\mathbf{X}) = 1$, the matrix is invertible. Therefore, for any possible $\mathbf{y}$ defined by a dichotomy d, the weight vector $\tilde{\mathbf{w}}^* = \mathbf{X}^{-1}\mathbf{y}$ induces the dichotomy. Therefore,

$$d_{VC} \geq D + 1 \tag{2.139}$$

Show $d_{VC} \leq D + 2$
We need to show that no set of $D+2$ points can be shattered by the set of linear classifiers. Let $\mathbf{x}_1, ..., \mathbf{x}_{D+2}$ be arbitrary. Because there are more points than there are dimensions, we know that they are linearly dependent. Let i be such that

$$\mathbf{x}_i = \sum_{i \neq j} \lambda_j \mathbf{x}_j \tag{2.140}$$

The linear dependency implies $\lambda_j \neq 0$ for at least one j. We now construct a dichotomy that cannot be expressed by a linear classifier. We define the dichotomy as follows

$$d(\mathbf{x}_l) = \begin{cases} \text{sign}(\lambda_l) & \text{if } l \neq i \\ -1 & \text{otherwise} \end{cases} \tag{2.141}$$

Let $h \in \mathcal{H}$ be any hypothesis with weight vector $\tilde{\mathbf{w}}$. Assume that $h(\mathbf{x}_j) = d(\mathbf{x}_j)$ for $j \neq i$. This has to hold if h induces d. However, this yields

$$h(\mathbf{x}_i) = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i) = \text{sign}\left(\sum_{i \neq j} \lambda_j \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_j\right) > 0 > d(\mathbf{x}_i) = -1 \tag{2.142}$$

Because $d(\mathbf{x}_j) = \text{sign}(\lambda_j) = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_j) \Rightarrow \lambda_j \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_j > 0$. Therefore, h does not classify $\mathbf{x}_i$ correctly. Because we assumed the points to be arbitrary, we showed that no set of $D+2$ points can be shattered. Therefore, we get

$$d_{VC} \leq D + 1 \tag{2.143}$$

□

One observation that we can make in the case of linear classifiers is that the VC-dimension coincides with the number of parameters. There are $D+1$ weights for the individual dimensions and the VC-dimension is also $D + 1$. We also observe this in the case of positive rays. Each positive ray is characterized by one parameter a and the VC-dimension is also one. This raises the following question:

> Does the VC-dimension coincide with the number of parameter?

The answer is: No, it does not. We discuss two examples where this observation cannot be made. First first example shows that the number of parameters can be higher than the VC-dimension and the second example shows that the number of parameters can be lower than the VC-dimension.

Example 10. *Consider the following hypothesis set in* $\mathbb{R}^2$

$$\mathcal{H} = \{\mathbf{x} \mapsto h(g(\mathbf{x})) : h, g \textit{ linear classifiers}\} \tag{2.144}$$

Each hypothesis consists of two nested linear classifiers h, g*. Obviously, each hypothesis requires* $2 \cdot 3 = 6$ *parameters. However, the VC-dimension of the set is still 3. In order to see this, let us think about the actions that can be performed by the outer classifier* h*. Once* g *has made a decision,* h *can either accept the decision (*$h(\mathbf{x}) = g(\mathbf{x})$*) or invert it (*$h(\mathbf{x}) = -g(\mathbf{x})$*). Both cases can be "simulated" using only a single linear classifier.*

Example 11 (From chapter 2.3 of [Bur98]). *Consider the following hypothesis set in* $\mathbb{R}$

$$\mathcal{H} = \{\mathbf{x} \mapsto sign(\sin(\alpha x - 0.5\pi)) : \alpha \in \mathbb{R}\} \tag{2.145}$$

We show that this hypothesis set, which is controlled by a single parameter, has an infinite VC-dimension. For this purpose, we choose N *points as follows*

$$x_i = 10^{-i}, \text{ for } i = 1, ..., N \tag{2.146}$$

Let d *by any dichotomy on* $x_1, ..., x_N$. *We choose* α *as follows*

$$\alpha = \pi \left(1 + \sum_{i=1}^{N} \frac{(1 - d(x_i))10^i}{2} \right) \tag{2.147}$$

Then, it holds

$$h(x_j) = sign\left(\sin\left(\alpha x_j - 0.5\pi\right)\right) \tag{2.148}$$

$$= sign(\sin(\pi(1 + \sum_{i=1}^{N} \frac{(1 - d(x_i))10^i}{2})10^{-j} - 0.5\pi)) \tag{2.149}$$

$$= sign(\sin(\pi(1 + \sum_{i=1}^{N} \frac{(1 - d(x_i))10^{i-j}}{2}) - 0.5\pi)) \tag{2.150}$$

$$= sign(\sin(\pi(1 + \overbrace{\underbrace{\sum_{i=1:d(x_i)=-1}^{N} 10^{i-j}}_{A}}^{B}) - 0.5\pi)) \tag{2.151}$$

- *If* $d(x_j) = 1$, *then* A *is even and* B *is odd. Thus* $\sin(\pi B - 0.5\pi) = 1$.
- *If* $d(x_j) = -1$, *then* A *is odd (because it includes* 10^0*) and* B *is even. Thus* $\sin(\pi B - 0.5\pi) = -1$.

Therefore, $h(x_j) = d(x_j)$ *for all* j. *This means that the VC-dimension of* $\mathcal{H}$ *is infinite although it only has a single parameter.*

Therefore, the VC-dimension does not characterize the number of parameters. The VC-dimension is measures the *effective number of parameters* or the *degrees of freedom*. Think of it this way: We can define the set of linear classifiers in any way we want. We can choose to use $1000D$ or only $D + 1$ parameters. The parametrization is just a technical formalization. Regardless of the way we parametrize the hypothesis set, the expressive power does not change. The VC-dimension is an abstract measure based solely on the expressive power (i.e. how large a data set can we shatter).

2.7 Bias-Variance Tradeoff

In the previous sections, we investigated the behaviour of the in-sample error and the out-of-sample error. We saw that we can bound the generalization error using the VC-inequality. In all of our discussions, we focused only on the hypothesis set $\mathcal{H}$ and did not take into account a practical learning algorithm. In this section, we investigate the so called *expected* out-of-sample

error w.r.t to the squared loss $\mathcal{L}_{squared}$ and our training algorithm T. If the learning algorithm is deterministic[5], then it can be seen as a mapping

$$T : (\mathcal{X} \times \mathcal{Y})^N \mapsto \mathcal{H} \tag{2.152}$$

that takes N training examples $(\mathbf{x}_i, y_i)$ and returns one of our hypotheses g. In the VC analysis, we found a bound on the generalization error by taking the supremum over the hypothesis set.

$$|E_{out}(g) - E_{in}(g)| \leq \sup_{h \in \mathcal{H}} |E_{out}(h) - E_{in}(h)| \tag{2.153}$$

where $g = T(X)$ is the result of our training algorithm on the data set $X \in (\mathcal{X} \times \mathcal{Y})^N$. By performing this step, we removed the dependency on the training algorithm. Now, we look at the *expected* out-of-sample error that is derived by taking the expected value over the training sets X.

$$\mathbb{E}_X[E_{out}(T(X))] = \int_{(\mathcal{X}\times\mathcal{Y})^N} E_{out}(T(X))p(X)dX \tag{2.154}$$

Because the order of integration is irrelevant, it holds

$$\begin{aligned} \mathbb{E}_X[E_{out}(T(X))] &= \mathbb{E}_X[\mathbb{E}_{\mathbf{x}}[\mathcal{L}_{squared}(T(X), \mathbf{x}, f(\mathbf{x}))]] & (2.155)\\ &= \mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[\mathcal{L}_{squared}(T(X), \mathbf{x}, f(\mathbf{x}))]] & (2.156) \end{aligned}$$

Let us focus on the inner part

$$\mathbb{E}_X[\mathcal{L}_{squared}(T(X), \mathbf{x}, f(\mathbf{x}))] = \mathbb{E}_X[(T(X)(\mathbf{x}) - f(\mathbf{x}))^2] \tag{2.157}$$

The key ingredient to the bias-variance decomposition is the *expected hypothesis* that results from our training algorithm.

$$\overline{g}(\mathbf{x}) := \mathbb{E}_X[T(X)] \tag{2.158}$$

This is defined because a our hypothesis set consists of real valued functions. We derive the decomposition by adding $\overline{g}(\mathbf{x}) - \overline{g}(\mathbf{x})$.

$$\begin{aligned} &\mathbb{E}_X[(T(X)(\mathbf{x}) - f(\mathbf{x}))^2] & (2.159)\\ =&\mathbb{E}_X[(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}) + \overline{g}(\mathbf{x}) - f(\mathbf{x}))^2] & (2.160)\\ =&\mathbb{E}_X[(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}))^2 + (\overline{g}(\mathbf{x}) - f(\mathbf{x}))^2 - 2(\overline{g}(\mathbf{x}) - y)(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}))] & (2.161)\\ =&\mathbb{E}_X[(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}))^2] + \mathbb{E}_X[(\overline{g}(\mathbf{x}) - f(\mathbf{x}))^2] - 2(\overline{g}(\mathbf{x}) - f(\mathbf{x}))(\underbrace{\mathbb{E}_X[T(X)(\mathbf{x})]}_{\overline{g}(\mathbf{x})} - \overline{g}(\mathbf{x}))) & (2.162)\\ =&\underbrace{\mathbb{E}_X[(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}))^2]}_{var(\mathbf{x})} + \underbrace{(\overline{g}(\mathbf{x}) - f(\mathbf{x}))^2}_{bias(\mathbf{x})} & (2.163) \end{aligned}$$

Substituting this result back in (2.156) yields

$$\mathbb{E}_X[E_{out}(T(X))] = \mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}))^2]] + \mathbb{E}_{\mathbf{x}}[(\overline{g}(\mathbf{x}) - f(\mathbf{x}))^2] \tag{2.164}$$

Let us discuss what the two terms describe.

[5]There are learning algorithms that incorporate randomness. For example, random forests only optimize over a random subset of features [Bre01].

The variance $\mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[(T(X)(\mathbf{x})-\overline{g}(\mathbf{x}))^2]]$ quantifies how well our training algorithm on average approximates the best possible hypothesis $\overline{g}$ using this algorithm. If this quantity is large, then depending on the data set, we mostly get a somewhat bad approximation. If this quantity is small, then the algorithm approximates the best hypothesis quite well on many data sets. Note, however, that the best possible hypothesis $\overline{g}$ does not necessarily have to be a good approximation of the target function.

The (squared) bias $\mathbb{E}_{\mathbf{x}}[(\overline{g}(\mathbf{x}) - f(\mathbf{x}))^2]$ quantifies how well the best possible hypothesis $\overline{g}$ approximates the true target values. If this quantity is small, then in principle it should be possible to find a hypothesis using our learning algorithm that approximates the target function well.

2.8 Overfitting

In this chapter, we want to discuss the problem of *overfitting* and ways to cope with it. The inverse problem of estimating a function from a finite number of independent samples can in general be consider *ill-posed*. This means that the solution to the problem cannot be uniquely identified given the information provided by the samples.
Consider the problem of real valued regression. If we use the typical squared loss function, we end up with an optimization problem of the form

$$\min_h \sum_{i=1}^{N} (h(\mathbf{x}_i) - y_i)^2 \tag{2.165}$$

where the data points $(\mathbf{x}_i, y_i)$ are sampled from the joint probability distribution $p(\mathbf{x}, y)$. In chapter 2.3 we saw that the target function is given by the conditional expectation $\mathbb{E}[y \mid \mathbf{x}]$. Thus, we can rewrite the optimization problem as follows

$$\min_h \sum_{i=1}^{N} (h(\mathbf{x}_i) - (\mathbb{E}[y \mid \mathbf{x}_i] + \epsilon_i))^2 \tag{2.166}$$

where ϵ_i are *noise terms* with mean 0. From this formulation, we see that the best hypothesis is the one that fits all the noisy data points perfectly. Thus, we *overfit* the data. A common way to avoid overfitting is a technique called *regularization*.

2.8.1 Bias-Variance Tradeoff for noisy Data

Before we discuss regularization, we briefly consider the bias-variance tradeoff for noisy data. For this purpose, we consider the noisy target $f(\mathbf{x}) + \epsilon(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}] + \epsilon(\mathbf{x})$.

$$\mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[(T(X)(\mathbf{x}) - f(\mathbf{x}) - \epsilon(\mathbf{x}))^2]] = \mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[(T(X)(\mathbf{x}) - f(\mathbf{x}))^2 - 2((T(X)(\mathbf{x}) - f(\mathbf{x}))\epsilon(\mathbf{x}) + \epsilon(\mathbf{x})^2]] \tag{2.167}$$

$$= \mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[(T(X)(\mathbf{x}) - f(\mathbf{x}))^2]] - 2\mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[((T(X)(\mathbf{x}) - f(\mathbf{x}))]] \underbrace{\mathbb{E}_{\mathbf{x}}[\epsilon(\mathbf{x})]}_{=0} + \mathbb{E}_{\mathbf{x}}[\epsilon(\mathbf{x})^2] \tag{2.168}$$

$$= \underbrace{\mathbb{E}_{\mathbf{x}}[\mathbb{E}_X[(T(X)(\mathbf{x}) - \overline{g}(\mathbf{x}))^2]]}_{var} + \underbrace{\mathbb{E}_{\mathbf{x}}[(\overline{g}(\mathbf{x}) - f(\mathbf{x}))^2]}_{bias} + \underbrace{\mathbb{E}_{\mathbf{x}}[\epsilon(\mathbf{x})^2]}_{noise} \tag{2.169}$$

Let us consider which terms we can control and which cannot control.

- The **noise** term is determined by the joint distribution $p(\mathbf{x}, y)$. We do not have any control over this term.

- The **bias** term is determined by the target function and the average hypothesis. We can influence this term by modifying the training algorithm and therefore the average hypothesis. However, we cannot influence the term in a controlled manner because we do not know the target function.

- The **variance** term is determined only by the training algorithm. Thus, by modifying the algorithm such that the results vary less, we can reduce this term. This is what regularization aims at.

2.8.2 Regularization

Suppose our hypothesis set $\mathcal{H}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and basis $\{\phi_i\}_{i=1}^{D}$ where D can be infinite. We can decrease the variance by constraining the solution of the minimization problem to a subset of $\mathcal{H}$. To be specific, in this discussion we focus on the so called *squared regularizer* that restricts the squared norm of a hypothesis. The constrained optimization problem can be stated as follows

$$\min_h \sum_{i=1}^{N} \mathcal{L}(h, \mathbf{x}_i, y_i) \text{ s.t.} \|h\|_{\mathcal{H}}^2 \leq C \tag{2.170}$$

where C is the restriction parameter. Obviously, the smaller C is, the smaller the variance becomes. Using the KKT conditions (See for example chapter 5 of [NW06]) this is equivalent to the following unconstrained optimization problem.

$$\min_h \sum_{i=1}^{N} \mathcal{L}(h, \mathbf{x}_i, y_i) + \lambda \|h\|_{\mathcal{H}}^2 \tag{2.171}$$

where $\lambda \geq 0$ is the so called *regularization parameter*.

Example 12. *Let us consider the example of regularized polynomial regression which corresponds to linear regression under a non-linear transformation (as discussed in chapter 3). We consider polynomials up to the fixed order D on the interval $[-1, 1]$. As basis we choose the* Legendre polynomials P_i *which are orthonormal w.r.t. to the inner product*

$$\langle f, g \rangle_{\mathcal{H}} = \int_{-1}^{1} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x} \tag{2.172}$$

i.e.

$$\langle P_i, P_j \rangle_{\mathcal{H}} = \delta_{i,j} \tag{2.173}$$

Our goal is to solve (2.171) for this setting where we choose the squared loss function. Thus, the optimization problem is

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{N} (h(\mathbf{x}_i) - y_i)^2 + \lambda \|h\|_{\mathcal{H}}^2 \tag{2.174}$$

Let $h \in \mathcal{H}$. We can rewrite $\|h\|_{\mathcal{H}}^2$ as follows

$$\|h\|_{\mathcal{H}}^2 = \left\| \sum_{i=1}^{D} w_i P_i(\mathbf{x}) \right\|_{\mathcal{H}}^2 \tag{2.175}$$

$$= \left\langle \sum_{i=1}^{D} w_i P_i(\mathbf{x}), \sum_{i=1}^{D} w_i P_i(\mathbf{x}) \right\rangle_{\mathcal{H}} \tag{2.176}$$

$$= \sum_{i=1}^{D} \sum_{j=1}^{D} w_i w_j \underbrace{\langle P_i(\mathbf{x}), P_i(\mathbf{x}) \rangle_{\mathcal{H}}}_{=\delta_{i,j}} \tag{2.177}$$

$$= \sum_{i=1}^{D} w_i^2 \tag{2.178}$$

$$= \|\mathbf{w}\|_2^2 \tag{2.179}$$

Thus, the squared norm of h in the Hilbert space is equal to the squared l_2 norm of the weight vector $\mathbf{w}$. Now, let $(\mathbf{X})_{i=1}^N = (P_1(\mathbf{x}_i), \ldots, P_D(\mathbf{x}_i))$. Then the optimization problem is equivalent to

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \tag{2.180}$$

Hence, the problem of finding a function in the Hilbert space that allows regularization transforms into the problem of solving a linear system. Taking the derivative w.r.t $\mathbf{w}$ yields

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.181}$$

2.8.3 Validation

The main question we still have to answer is:

> How to choose the regularization parameter λ?

Obviously, for $\lambda \to \infty$ the variance goes towards 0. However, the bias term is likely to increase if $f \neq 0$. Thus, finding a reasonable value for lambda is not trivial. In this section, we discuss the method of *validation* in order to solve the general problem of *model selection*.
The problem of model selection is concerned with selection one of several final hypotheses that are obtained using different models and/or training algorithms. For example, training with M different values for λ yields M final hypotheses. Obviously, we want to select the model that has the smallest out-of-sample error E_{out}.
The first quantity that comes to mind when selecting the models is the in-sample-error E_{in}. However, this quantity is inherently biased. During training, our main concern is to minimize E_{in}. Thus, E_{in} is most likely a very poor estimate for E_{out}.
Validation is the idea of keeping a separate data set on which we estimate the performance of the models. We obtain such a separate data set by splitting our initial data set X into a training set X_{train} and a *validation set* X_{val} such that

$$X = X_{train} \cup X_{val} \text{ and } X_{train} \cap X_{val} = \emptyset \tag{2.182}$$

The problem that we are facing here is the following: If we our validation set is small, then the estimated out-of-sample error based on the validation set has a high variance and therefore is unreliable. Thus, we want out validation set to be large. However, if the validation set is large,

then the training set is small. We know from the VC-inequality that the in-sample-error in this case is a poor estimate of the out-of-sample error and therefore our training algorithm is likely to yield a poor final hypothesis. A solution to this problem is *K-fold-cross-validation.*
In K-fold-cross validation we split our initial data set into K pairs of training and validation set $(X_{train,i}, X_{val,i})_{i=1}^{K}$ by dividing X into K disjoint subsets $X_1, ..., X_K$ of the same size and define

$$X_{train,i} = X \backslash X_i \tag{2.183}$$

$$X_{val,i} = X_i \tag{2.184}$$

Then, for each i, we train a model on $X_{train,i}$ and evaluate the performance on $X_{val,i}$. The resulting estimated out-of-sample error for a specific model is the average estimation over the K runs. We then pick the model that has the smallest estimated out-of-sample error.

Remark 6 (Rule of thumb)**.** *10-fold-cross-validation usually yields reliable estimates in practice.*

Chapter 3

Learning Algorithms

For the most part, this chapter is detached from the theory part of the previous chapter. Here, we introduce some basic learning algorithms that can actually be used in practice.

3.1 The Linear Model

In the introductory chapter, we introduced linear classifiers and the Perceptron Learning Algorithm as a first example of a learning algorithm. In this section, we discuss the linear model in greater detail.

3.1.1 The Pocket Algorithm

Recall that at each iteration of the PLA the algorithm selects a single misclassified point and updates the weight vector using this point. While we know that the algorithm converges in the case of linearly separable data, the convergence might be very slow. Therefore, in practice one often stops the algorithm after a fixed number of iterations. In the light of the previous chapter, we can say that the PLA tries to optimize the in-sample-error using the 0-1-loss function; that is

$$E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{0-1}(h, \mathbf{x}_i, y_i) = \frac{1}{N}\sum_{i=1}^{N} I(h(\mathbf{x}_i) \neq y_i) \tag{3.1}$$

Unfortunately, the update at some iteration might **in**crease the error instead of **de**creasing it. Hence, if we stop after a fixed number of iterations, the final hypothesis might not be the best hypothesis (in terms of the in-sample-error) that we encountered over all iterations.
There is a simple variation of the PLA that is called the *Pocket Algorithm*. The idea is that we perform the PLA as usual and save the best hypothesis in terms of the in-sample-error. If we stop, the algorithm after a fixed number of iterations, the final hypothesis of the Pocket Algorithm will be the best hypothesis that has been found over all iterations.

3.1.2 Linear Regression

In this section, we deal with the problem of *regression*. All this means is that our output space $\mathcal{Y}$ is the set of real numbers (i.e. $\mathcal{Y} = \mathbb{R}$). We can solve such a learning problem using *linear regression*. In linear regression we assume a linear relationship between our input vectors and the output value. Hence, the set of hypotheses is

$$\mathcal{H} = \{\tilde{\mathbf{x}} \mapsto \tilde{\mathbf{w}}^T\tilde{\mathbf{x}} : \tilde{\mathbf{w}} \in \mathbb{R}^{D+1}\} \tag{3.2}$$

There are many possible learning algorithms for this problem and this hypothesis set. One of the simplest algorithm tries to optimize the squared loss. We follow this approach here. Hence, our objective is to minimize

$$E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{squared}(h, \mathbf{x}_i, y_i) = \frac{1}{N}\sum_{i=1}^{N}(h(\mathbf{x}_i) - y_i)^2 \tag{3.3}$$

We can write this in a more compact form using matrix notation. For this purpose, we introduce the matrix $\tilde{\mathbf{X}}$ and the vector $\mathbf{y}$.

$$\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1 \dots \tilde{\mathbf{x}}_N)^T \tag{3.4}$$

$$\mathbf{y} = (y_1 \dots y_N)^T \tag{3.5}$$

Because our hypothesis set is isomorphic to $\mathbb{R}^{D+1}$, we can say that our in-sample-error is a function of $\tilde{\mathbf{w}}$. Using these introduced matrices, we can write the error as

$$E_{in}(\tilde{\mathbf{w}}) = \frac{1}{N}\|\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y}\|_2^2 \tag{3.6}$$

This error function has two convenient properties. Firstly, it is convex if $\tilde{\mathbf{X}}$ is not rank deficient. In order to see this, we can look at the second derivative w.r.t. to $\tilde{\mathbf{w}}$. It holds

$$\nabla^2_{\tilde{\mathbf{w}}} E_{in}(\tilde{\mathbf{w}}) = \frac{1}{N}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} \tag{3.7}$$

From basic calculus, we now that a function is convex, if its hessian is symmetric positive definite. If $\tilde{\mathbf{X}}$ has rank $D+1$, it holds $\mathbb{L}_0(\tilde{\mathbf{X}}) = \{0\}$ and thus

$$\mathbf{z}^T\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\mathbf{z} = \|\tilde{\mathbf{X}}\mathbf{z}\|_2^2 > 0 \ \forall \mathbf{z} \neq 0 \tag{3.8}$$

Because $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is obviously symmetric, the error function is convex. The convexity is particularly convenient because it implies that the function has a unique local minimum that corresponds to the optimal weight vector $\tilde{\mathbf{w}}^*$. The second convenient property is that we can find this optimum in a closed form. For this purpose, we take the first derivative with respect to $\tilde{\mathbf{w}}$.

$$\frac{\partial}{\partial\tilde{\mathbf{w}}}E_{in}(\tilde{\mathbf{w}}) = \frac{1}{N}\frac{\partial}{\partial\tilde{\mathbf{w}}}\left(\|\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y}\|_2^2\right) \tag{3.9}$$

$$= \frac{1}{N}\frac{\partial}{\partial\tilde{\mathbf{w}}}\left((\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y})^T(\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{y})\right) \tag{3.10}$$

$$= \frac{1}{N}\frac{\partial}{\partial\tilde{\mathbf{w}}}\left(\tilde{\mathbf{w}}^T\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}} - 2\tilde{\mathbf{w}}^T\tilde{\mathbf{X}}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}\right) \tag{3.11}$$

$$= \frac{1}{N}(2\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}} - 2\tilde{\mathbf{X}}^T\mathbf{y}) \tag{3.12}$$

A necessary condition for a local minimum is $\nabla_{\tilde{\mathbf{w}}}E_{in}(\tilde{\mathbf{w}}^*) = 0$. Therefore

$$\nabla_{\tilde{\mathbf{w}}}E_{in}(\tilde{\mathbf{w}}^*) \stackrel{!}{=} 0 \tag{3.13}$$

$$\Leftrightarrow \quad \frac{1}{N}(2\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}}^* - 2\tilde{\mathbf{X}}^T\mathbf{y}) = 0 \tag{3.14}$$

$$\Leftrightarrow \quad \tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^T\mathbf{y} \tag{3.15}$$

Hence, the weight vector that minimizes the error function corresponds to the solution of the equation $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^T\mathbf{y}$. In general, this solution is given by the so called *pseudo inverse* $\tilde{\mathbf{X}}^+$ of $\tilde{\mathbf{X}}$.

$$\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^+\mathbf{y} \tag{3.16}$$

If $\tilde{\mathbf{X}}$ has rank $D+1$, then the following identity holds

$$\tilde{\mathbf{X}}^{+} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T \tag{3.17}$$

Remark 7 (Normal equations)**.** *The system in equation (3.15) is known in the field of numerical analysis as the* normal equations*. In general, one should avoid optimizing the squared error using the normal equations because they are ill conditioned. This is due to the fact that the matrix $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ has the squared condition number of $\tilde{\mathbf{X}}$. There are two stable ways to solve the system. If the matrix has rank $D+1$, one can use a QR-decomposition of $\tilde{\mathbf{X}}$. If the matrix is rank deficient, the solution is no longer unique. In this case, one can calculate the singular value decomposition (SVD) of the matrix and select the solution that has the smallest norm. More information about least squares systems can be found in [GVL13].*

3.1.3 Linear Regression for Classification

We can use the linear regression framework that we developed in the previous section for the case of binary classification. Because a classifier is still a real valued function, we can directly apply the optimization procedure using the squared loss. In order to get a hard decision from the learned function, we simply take the sign. Hence, we learn a linear classifier.
However, there is one caveat to this method. Because we use a loss function that is not designed for classification (squared loss), the final hypothesis tends to *overfit* the data. In order to see this, assume that the final hypothesis classifies all data points correctly. However, the in-sample-error is not 0 when using the squared loss. All points induce a penalty that is proportional to the squared distance from the point to the decision boundary. Hence, the learning algorithm does not try to minimize the classification error (using the 0-1-loss function).
Using linear regression for classification is useful in practice in several scenarios. Firstly, it can give us an impression of how difficult a certain classification problem is. Because it is so simple to apply, we can quickly evaluate the performance of this method as a baseline approach. Secondly, the the resulting weights can be used a an initialization for more sophisticated learning algorithm. For example, we said that the PLA converges very slowly. However, if we initialize the weights using linear regression for classification, many points will already be classified correctly when the algorithm starts. In the following iterations, it only has to adjust the decision boundary.

3.1.4 Non-linear Transform

We often think of a linear model as a function that is linear in $\tilde{\mathbf{x}}$. While this is true, it is also linear in $\tilde{\mathbf{w}}$. To be specific, during training, the data points are fixed and only the weights vary. We can use this observation to introduce a so called *feature space map*

$$\phi : \mathbb{R}^D \mapsto \mathcal{Z} \tag{3.18}$$

where $\mathcal{Z}$ is some Hilbert space. By doing so, we change our linear model to

$$h(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle \tag{3.19}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of $\mathcal{V}$ and $\mathbf{w} \in \mathcal{V}$ is the weight vector. Although this new model is non-linear in $\mathbf{x}$, it is still linear in $\mathbf{w}$. Therefore, we can still use the training algorithms that we discussed in the previous sections.

3.1.5 Logistic Regression

In section 2.3 we saw that for the case of noisy targets, we can minimize the expected loss, if we assign the vector $\mathbf{x}$ to the class y that has the higher *posterior probability* $\mathbb{P}[y \mid \mathbf{x}]$. There are many approaches from probability theory that try to do this by using Bayes' theorem and model the posterior probability as a product of *likelihood* and *prior*. Let p be the corresponding probability density (or mass) function. Then it holds

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y)p(y)}{p(\mathbf{x})} \tag{3.20}$$

The statistical approaches try to model $p(\mathbf{x} \mid y)$ and $p(y)$ explicitly using probability density functions. We will return to this general approach at the end of the section and compare it to the approach of *logistic regression* that we discuss now.
Instead of using Bayes' theorem for finding the posterior probability, we try to model the posterior probability directly. Hence, our goal is to find a hypothesis h such that

$$p(1 \mid \mathbf{x}) = h(\mathbf{x}) \tag{3.21}$$

Because p is a probability mass function for a binary event, it holds

$$p(1 \mid \mathbf{x}) = 1 - p(-1 \mid \mathbf{x}) \tag{3.22}$$

And therefore, we can write our approach as

$$p(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } y = 1 \\ 1 - h(\mathbf{x}) & \text{otherwise} \end{cases} \tag{3.23}$$

This is the general setup. In logistic regression, we want to use a linear model h. To be specific, we use a so called *generalized linear model*.

Definition 3.1.1 (Generalized linear model)**.** *Let $\theta : \mathbb{R} \mapsto \mathbb{R}$ be a function. The* generalized linear model *with weight vector $\tilde{\mathbf{w}} \in \mathbb{R}^{D+1}$ is defined as*

$$h(\mathbf{x}) = \theta(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \tag{3.24}$$

The model is called generalized linear model, because all the linear models that have seen so far are special cases.

Perceptron/Linear classifier If we take $\theta \equiv \text{sign}$, then we get a linear classifier

$$h(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \tag{3.25}$$

Linear regression If we take θ to be the identity function id, then we get the usual linear regressor

$$h(\mathbf{x}) = \text{id}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} \tag{3.26}$$

In the case of logistic regression, we want to choose θ such that $h(\mathbf{x})$ can be interpreted as a probability. Thus, it should satisfy

$$\theta(\mathbf{x}) \in [0, 1] \ \forall \mathbf{x} \tag{3.27}$$

We use the so called *logistic sigmoid function* σ that is defined as follows

$$\sigma(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x})} \tag{3.28}$$

Obviously, this function fulfills the requirements. Thus, our model is of the form

$$h(\mathbf{x}) = \sigma(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) = \frac{1}{1 + \exp(-\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})} \tag{3.29}$$

As in the other cases, we determine the weight vector $\tilde{\mathbf{w}}$ by minimizing a loss function $\mathcal{L}$. We already saw that in the case of linear regression, maximizing the *likelihood* under a Gaussian noise assumption corresponds to minimizing the squared loss function. Let us consider the likelihood function in this case. Under the assumption of independently and identically sampled points, we get

$$p(y_1, ..., y_N \mid \mathbf{x}_1, ..., \mathbf{x}_N) = \prod_{i=1}^{N} p(y_i \mid \mathbf{x}_i) \tag{3.30}$$

If we use the fact that $\sigma(-\mathbf{x}) = 1 - \sigma(\mathbf{x})$, it holds

$$p(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } y = 1 \\ 1 - h(\mathbf{x}) & \text{otherwise} \end{cases} = \begin{cases} h(\mathbf{x}) & \text{if } y = 1 \\ h(-\mathbf{x}) & \text{otherwise} \end{cases} = h(y\mathbf{x}) \tag{3.31}$$

because $y \in \{-1, 1\}$. Thus, our likelihood function transforms to

$$L(\mathbf{y} \mid \mathbf{X}) = \prod_{i=1}^{N} h(y_i \mathbf{x}_i) = \prod_{i=1}^{N} \sigma(y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \tag{3.32}$$

We want to maximize the likelihood with respect to $\tilde{\mathbf{w}}$ it holds

$$\arg\max_{\tilde{\mathbf{w}}} \prod_{i=1}^{N} \sigma(y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) = \arg\min_{\tilde{\mathbf{w}}} - \ln \left(\prod_{i=1}^{N} \sigma(y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \right) \tag{3.33}$$

$$= \arg\min_{\tilde{\mathbf{w}}} - \sum_{i=1}^{N} \ln \left(\sigma(y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \right) \tag{3.34}$$

$$= \arg\min_{\tilde{\mathbf{w}}} - \sum_{i=1}^{N} \ln \left(\frac{1}{1 + \exp(-y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}})} \right) \tag{3.35}$$

$$= \arg\min_{\tilde{\mathbf{w}}} \sum_{i=1}^{N} \underbrace{\ln \left(1 + \exp(-y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \right)}_{\mathcal{L}_{cross}(h, \mathbf{x}, y)} \tag{3.36}$$

Thus, maximizing the likelihood corresponds to minimizing the cross entropy loss function $\mathcal{L}_{cross}(h, \mathbf{x}, y)$[1]. In summary, our goal is to minimize the following error function with respect to $\tilde{\mathbf{w}}$.

$$E(\tilde{w}) = \sum_{i=1}^{N} \ln \left(1 + \exp(-y_i \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) \right) \tag{3.37}$$

One very convenient property of the cross entry loss is that it is convex.

[1] This is just a variant of the usual cross entropy loss function.

Corollary 3. *The cross entropy loss function* $\ln\left(1+\exp(-y_i\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})\right)$ *is convex in* $\tilde{\mathbf{w}}$.

Proof. A sufficient condition for convexity is a positive definite Hessian. It holds

$$\nabla_{\tilde{\mathbf{w}}}E(\tilde{\mathbf{w}}) = \frac{-y}{1+\exp(y\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})}\tilde{\mathbf{x}} \tag{3.38}$$

$$\nabla^2_{\tilde{\mathbf{w}}}E(\tilde{\mathbf{w}}) = \frac{y^2\exp(y\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})}{(1+\exp(y\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}))^2}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T \tag{3.39}$$

Let $\tilde{\mathbf{z}} \in \mathbb{R}^{D+1}$, then it holds

$$\tilde{\mathbf{z}}^T\nabla^2_{\tilde{\mathbf{w}}}E(\mathbf{x})\tilde{\mathbf{z}} = \underbrace{\frac{y^2\exp(y\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})}{(1+\exp(y\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}))^2}}_{\geq 0}\underbrace{(\tilde{\mathbf{z}}^T\tilde{\mathbf{x}})^2}_{\geq 0} \geq 0 \tag{3.40}$$

□

Unfortunately, in contrast to the squared loss function, no closed form solution can be obtained. We have to use more sophisticated optimization algorithms in this case. But because of the convexity of the loss function, even relatively simple methods can yield reasonable approximations of the optimum. We use a method that is called *gradient descent*. In the following, we discuss the method somewhat informally to give an understanding. A more rigorous discussion can be found in chapter 3 of [NW06].
The general idea behind gradient descent is that we iteratively approximate the optimum. At each iteration we want to make a step in the *direction of the steepest descent*. Hence, we want to determine the direction ξ (with $\|\xi\|_2 = 1$) that maximizes the directional derivative

$$\lim_{t\to 0}\frac{E(\tilde{\mathbf{w}}+t\xi)-E(\tilde{\mathbf{w}})}{t} \tag{3.41}$$

If E has a total derivative ∇E , then it holds

$$\lim_{t\to 0}\frac{E(\tilde{\mathbf{w}}+t\xi)-E(\tilde{\mathbf{w}})}{t} = (\nabla E(\tilde{\mathbf{w}}))^T\xi \tag{3.42}$$

Using the Cauchy-Schwarz inequality, we get

$$|(\nabla E(\tilde{\mathbf{w}}))^T\xi| \leq \|(\nabla E(\tilde{\mathbf{w}}))\|_2^2\underbrace{\|\xi\|_2^2}_{=1} \tag{3.43}$$

And thus

$$-\|\nabla E(\tilde{\mathbf{w}})\|_2^2 \leq (\nabla E(\tilde{\mathbf{w}}))^T\xi \tag{3.44}$$

for all directions ξ. Therefore, the direction of the steepest descent corresponds to the direction of the total derivative. Obviously, in our case the total derivative corresponds to the gradient. Hence, our optimization scheme has the following form

$$\tilde{\mathbf{w}}^{(n+1)} = \tilde{\mathbf{w}}^{(n)} - \eta\nabla_{\tilde{\mathbf{w}}}E(\tilde{\mathbf{w}}^{(n)}) \tag{3.45}$$

where $\eta > 0$ is the so called *learning rate*. Using a second order Taylor expansion it can be shown that there exists an η such that the update decreases the error function if have not reached the optimum yet.
In summary, our learning algorithm for logistic regression has the following form.

- Initialize $\tilde{\mathbf{w}}^{(0)}$
- For $n = 1, 2, ..., K$ do
  - Perform the update step

$$\tilde{\mathbf{w}}^{(n+1)} = \tilde{\mathbf{w}}^{(n)} + \eta \nabla_{\tilde{\mathbf{w}}} E(\tilde{\mathbf{w}}^{(n)}) \tag{3.46}$$

- Return $\tilde{\mathbf{w}}^{(K)}$

Instead of using a fixed number of iteration K, we can use an adaptive stopping criterion. There are many criteria available in the literature. One of the simplest methods considers the norm of the gradient. Because $\nabla_{\tilde{\mathbf{w}}} E(\tilde{\mathbf{w}}^*) = 0$ is a necessary condition for $\tilde{\mathbf{w}}^*$ to be the optimum, the norm of the gradient can be a good indicator of the quality of the current solution.

Remark 8. *In order to make logistic regression robust in practice, one usually needs to apply some kind of* regularization. *We discuss this matter in a later lecture.*

Let us now return to our initial discussion about the posterior probability that can be factorized into a product of likelihood and prior.

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y) p(y)}{p(\mathbf{x})} \tag{3.47}$$

Consider the number of parameters to estimate if we model the likelihood $p(\mathbf{x} \mid y)$ using a simple multivariant Gaussian $\mathcal{N}(\mu, \Sigma)$ distribution.

- $\mu \in \mathbb{R}^{D+1}$: D+1 parameters
- $\Sigma \in \mathbb{R}^{D+1 \times D+1}$ symmetric: D(D+1) parameters.

Thus, the number of parameters to estimate when using this very simple kind of probabilistic modelling grows quadratically in the number of dimensions. In comparison, in the case of logistic regression we only need to estimate $D + 1$ parameters. Hence, the number of parameters grows linearly in the dimensionality of the problem. This indicates that logistic regression might be a suitable for learning problems where $\mathcal{X}$ is very high dimensional.

Example 13 (Bag of words model)**.** *Assume we want to do document classification. For this purpose, we create (or learn) a* dictionary *consisting of 10,000 words. Given some text, we can generate a feature vector using the so called* bag of words model. *Let $w_1, ..., w_D$ be the words in the dictionary. For each word w_i, we define x_i to be the number of occurrences of w_i in the text. Thus, we get a 10,000 dimensional feature vector.*
This is a very frequently used model which gives rise to very high dimensional feature vectors. If we used a Gaussian likelihood model, then the we would have to estimate approximately 10^8 parameters. In contrast, using logistic regression, we only need to estimate 10^4 parameters.

3.2 Artificial Neural Networks and Support Vector Machines

The professor published excellent chapters from his book that cover these two topics.

Appendices

Appendix A

Exercises

A.1 Exercise 1

Exercise 1.1

Correct answer: [d]
Answer

(i) Although it seems like this is a machine learning approach because a statistical model is built in order to classify the points, technically it is not learning. Because we are given the exact ground truth measurements, we do not have to *learn* anything from *data*. The statistical model is just a convenient representation of the knowledge. Therefore, this is not learning.

(ii) This is true supervised learning because we are given nothing but *labeled training data*. We use this data in order to train a classifier that is based on some measurements. These measurements typically are given as points in some high dimensional vector space. The classifier divides this space into decision regions that are separated by decision boundaries.

(iii) This is reinforcement learning. In this case, we do not have labeled training data that allows us to learn the game strategy. Instead, we use a weaker form of learning. During training, we try different moves and a teacher tells us whether this was a good or a bad move.

Exercise 1.2

Correct answer: [a]
Explanation

(i) Although this is a classification problem, it is not suitable for machine learning. Unless prime numbers are generated according to a fixed pattern that can be learned, the best a machine learning algorithm could do is to memorize a finite number of primes.

(ii) This problem is highly suitable for a learning approach. There are certain measurements that belong to one of two classes (fraud vs no fraud). However, one does not know the mapping from the measurements to the classes. A machine learning algorithm can learn this mapping from labeled training data.

(iii) Technically, this can be seen as a regression problem. One wants to find a function that calculates the time it takes for an object to fall given its height from the ground. However,

because the true functional dependency between the input and output is known, a machine learning approach would not be the best choice. A better choice would be to assume the fixed relationship

$$h = \frac{1}{2}gt^2 \tag{A.1}$$

and then use a sequence of measurements $(h_1, t_1), ..., (h_N, t_N)$ to estimate g in a least-squares sense.

(iv) This is a machine learning problem. Obviously, there is some pattern that underlies the traffic at a certain location at a certain time. However, the exact pattern is unknown. We can only perform noisy *measurements*.

Exercise 1.3

Correct answer: [d]

Explanation Let $B \in \{1, 2\}$ be a random variable that corresponds to the event of choosing a bag and let $X_1, X_2 \in \{b, w\}$ be random variables that correspond to the event of picking a black b or white w ball. We want to determine the probability of picking a black ball given that we already picked a black ball. We make the following assumptions:

- $P(B = 1) = \frac{1}{2}$
- $P(X_i = b) = \frac{3}{4}$

Using Bayes' Theorem as well as the law of total probability, we get

$$\begin{aligned}
P(X_1 = b \mid X_2 = b) &= \sum_{i \in \{1,2\}} P(X_1 = b, B = i \mid X_2 = b) && \text{(A.2)} \\
&= \sum_{i \in \{1,2\}} P(X_1 = b \mid B = i, X_2 = b) P(B = i \mid X_2 = b) && \text{(A.3)} \\
&= \sum_{i \in \{1,2\}} P(X_1 = b \mid B = i, X_2 = b) \frac{P(X_2 = b \mid B = i) P(B = i)}{P(X_2 = b)} && \text{(A.4)} \\
&= P(X_1 = b \mid B = 1, X_2 = b) \frac{P(X_2 = b \mid B = 1) P(B = 1)}{P(X_2 = b)} && \text{(A.5)} \\
&= 1 \frac{1\frac{1}{2}}{\frac{3}{4}} = \frac{2}{3} && \text{(A.6)}
\end{aligned}$$

Exercise 1.4

Correct answer: [b]

Explanation

We want to know the probability that the sample mean is 0 if we draw 10 independent samples. Let $X_i \in \{0, 1\}$ be a random variable that corresponds to the event of drawing a sample. From

the exercise we have $P(X_i = 1) = 0.55$. Therefore,

$$P(\nu = 0) = P\left(\frac{1}{10}\sum_{i=1}^{10} X_i = 0\right) \tag{A.7}$$

$$= P(X_1 = 0, ..., X_{10} = 0) \tag{A.8}$$

$$= \prod_{i=1}^{10} P(X_i = 0) \tag{A.9}$$

$$= \prod_{i=1}^{10} 0.45 = 0.45^{10} = 3.405 \times 10^{-4} \tag{A.10}$$

Exercise 1.5

Correct answer: [c]
Explanation The probability that at least one draw has sample mean 0 is the opposite event of all samples having mean greater than 0. Therefore,

$$P(\exists i \in \{1, ..., 1000\} : \nu_i = 0) = 1 - P(\nu_1 > 0, ..., \nu_{1000} > 0) \tag{A.11}$$

$$= 1 - \prod_{i=1}^{1000} P(\nu_i > 0) \tag{A.12}$$

$$= 1 - \prod_{i=1}^{1000} (1 - P(\nu_i = 0)) \tag{A.13}$$

$$= 1 - \prod_{i=1}^{1000} (1 - 3.405 \times 10^{-4}) \tag{A.14}$$

$$= 1 - (1 - 3.405 \times 10^{-4})^{1000} = 0.289 \tag{A.15}$$

Exercise 1.6

Correct answer: [e]
Explanation There are 8 possible functions that agree with the training data. In order to count the agreements between the hypotheses, simple make a table

	101	110	111	#0	#1	#2	#3
1	0	0	0	1	3	3	1
2	0	0	1	1	3	3	1
3	0	1	0	1	3	3	1
4	0	1	1	1	3	3	1
5	1	0	0	1	3	3	1
6	1	0	1	1	3	3	1
7	1	1	0	1	3	3	1
8	1	1	1	1	3	3	1

As you can tell from the table, all hypotheses have the same number of other hypotheses they agree with.

Exercise 1.7

Correct answer: [b]
Explanation Execute `ex1_7`

Exercise 1.8

Correct answer: [c]
Explanation Execute `ex1_8`

Exercise 1.9

Correct answer: [b]
Explanation Execute `ex1_9`

Exercise 1.10

Correct answer: [b]
Explanation Execute `ex1_10`

A.2 Exercise 2

Exercise 2.1

Correct answer: [b]
Explanation Since this is a practical exercise, you can answer it by running the function `ex2_1` from the solutions directory. The function will produce three plots that show the histograms over the individual averages and will print out the actual averages.

Exercise 2.2

Correct answer: [d]
Explanation Both, ν_1 and ν_{rand} have distributions that satisfy Hoeffding's Inequality. This is not due to the fact that they both have an average that is close to the expected value, but because the distributions are not data dependent. You can think of ν_{min} as the result of a learning algorithm: At each run, we pick the value based on the data (Namely, we pick the smallest sample). We highlighted in section 2.2 that Hoeffding's Inequality cannot be used for these kinds of situations.
The function `ex2_2` plots the probability $\mathbb{P}\left[\left|\frac{1}{N}\sum_{n=1}^{N} X_n - \mathbb{E}[X]\right| > \epsilon\right]$ together with the Hoeffding bound for $\epsilon \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

Exercise 2.3

Correct answer: [e]
Explanation We want to calculate the probability that $h(\mathbf{x})$ deviates from y.

$$\mathbb{P}[h(\mathbf{x}) \neq y] \tag{A.16}$$

Because there are only two possible values for y ($\mathcal{Y} = \{-1, 1\}$), there are two cases when this happens

1. $h(\mathbf{x}) = f(\mathbf{x}) \wedge y \neq f(\mathbf{x}) \Rightarrow h(\mathbf{x}) \neq y$
2. $h(\mathbf{x}) \neq f(\mathbf{x}) \wedge y = f(\mathbf{x}) \Rightarrow h(\mathbf{x}) \neq y$

It is easy to see that

$$h(\mathbf{x}) \neq y \Leftrightarrow (h(\mathbf{x}) = f(\mathbf{x}) \wedge y \neq f(\mathbf{x})) \vee (h(\mathbf{x}) \neq f(\mathbf{x}) \wedge y = f(\mathbf{x})) \tag{A.17}$$

Therefore, we get

$$\begin{aligned}\mathbb{P}[h(\mathbf{x}) \neq y] =& \mathbb{P}[h(\mathbf{x}) = f(\mathbf{x}), y \neq f(\mathbf{x})] + \mathbb{P}[h(\mathbf{x}) \neq f(\mathbf{x}), y = f(\mathbf{x})] &\text{(A.18)}\\ =& \mathbb{P}[h(\mathbf{x}) = f(\mathbf{x})]\mathbb{P}[y \neq f(\mathbf{x})] + \mathbb{P}[h(\mathbf{x}) \neq f(\mathbf{x})]\mathbb{P}[y = f(\mathbf{x})] &\text{(A.19)}\end{aligned}$$

From the exercise sheet we know

- $\mathbb{P}[h(\mathbf{x}) = f(\mathbf{x})] = 1 - \mu$ (The probability that h does not make an error)
- $\mathbb{P}[h(\mathbf{x}) \neq f(\mathbf{x})] = \mu$
- $\mathbb{P}[y = f(\mathbf{x})] = \lambda$
- $\mathbb{P}[y \neq f(\mathbf{x})] = 1 - \lambda$

Therefore, the resulting probability is

$$\mathbb{P}[h(\mathbf{x}) \neq y] = (1 - \mu)(1 - \lambda) + \mu\lambda \tag{A.20}$$

Exercise 2.4

Correct answer: [b]
Explanation
There are two methods which lead to the correct answer.

Trial and error We can subsequently plug in the different options. If we plug in $\lambda = 0.5$, we get

$$\begin{aligned}\mathbb{P}[h(\mathbf{x}) \neq y] =& (1-\mu)(1-0.5) + \mu 0.5 && \text{(A.21)}\\ =& (1-\mu)0.5 + 0.5\mu = 0.5 - 0.5\mu + 0.5\mu = 0.5 && \text{(A.22)}\end{aligned}$$

Therefore, the performance of h is independent of μ for $\lambda = 0.5$.

Determine the value of λ Obviously, such a trial and error approach cannot work in practice where we do not have possible answers. A better way is to determine the value of λ such that

$$\mathbb{P}[h(\mathbf{x}) \neq y] = \int_0^1 \mathbb{P}[h(\mathbf{x}) \neq y] \mathrm{d}\mu \tag{A.23}$$

For the right hand side we get

$$\begin{aligned}\int_0^1 \mathbb{P}[h(\mathbf{x}) \neq y] \mathrm{d}\mu =& \int_0^1 1 - \mu - \lambda + 2\mu\lambda \mathrm{d}\mu && \text{(A.24)}\\ =& \left[\mu - 0.5\mu^2 - \lambda\mu + \mu^2\lambda\right]_0^1 && \text{(A.25)}\\ =& 1 - 0.5 = 0.5 && \text{(A.26)}\end{aligned}$$

Hence, we have to solve the following equation for λ

$$\begin{aligned}& 1 - \mu - \lambda + 2\mu\lambda = 0.5 && \text{(A.27)}\\ \Leftrightarrow \quad & \lambda(2\mu - 1) = 0.5 - 1 + \mu && \text{(A.28)}\\ \Leftrightarrow \quad & \lambda = \frac{\mu - 0.5}{(2\mu - 1)} = \frac{\mu - 0.5}{2(\mu - 0.5)} = \frac{1}{2} = 0.5 && \text{(A.29)}\end{aligned}$$

Therefore, for $\lambda = 0.5$ the performance of h is independent of μ.

Exercise 2.5

Correct answer: [c]
Explanation Execute `ex2_56`

Exercise 2.6

Correct answer: [c]
Explanation Execute `ex2_56`

Exercise 2.7

Correct answer: [a]
Explanation Execute `ex2_7`

Exercise 2.8

Correct answer: [d]
Explanation Execute `ex2_8`

Exercise 2.9

Correct answer: [a]
Explanation Execute `ex2_9`

Exercise 2.10

Correct answer: [b]
Explanation Execute `ex2_10`

A.3 Exercise 3

Exercise 3.1

Correct answer: [b]
Explanation We have to solve the following equation for N where δ is our desired upper bound.

$$2Me^{-2\epsilon^2 N} \overset{!}{\leq} \delta \tag{A.30}$$

$$\Leftrightarrow \quad e^{-2\epsilon^2 N} \leq \frac{\delta}{2M} \tag{A.31}$$

$$\Leftrightarrow \quad -2\epsilon^2 N \leq \ln\left(\frac{\delta}{2M}\right) \tag{A.32}$$

$$\Leftrightarrow \quad N \geq -\frac{\ln\left(\frac{\delta}{2M}\right)}{2\epsilon^2} \tag{A.33}$$

In order to get the answer, we have to plug in the following values:

- $M = 1$
- $\delta = 0.03$
- $\epsilon = 0.05$

Thus, we get

$$N \geq -\frac{\ln\left(\frac{0.03}{2\cdot 1}\right)}{2 \cdot 0.05^2} = 839.9 \tag{A.34}$$

Exercise 3.2

Correct answer: [c]
Explanation We can reuse the derivations from the previous exercise. We only have to plug in $M = 10$ instead of $M = 1$. Thus, we get

$$N \geq -\frac{\ln\left(\frac{0.03}{2\cdot 10}\right)}{2 \cdot 0.05^2} = 1300.4 \tag{A.35}$$

Exercise 3.3

Correct answer: [d]
Explanation We can reuse the derivations from the previous exercise. We only have to plug in $M = 100$ instead of $M = 1$. Thus, we get

$$N \geq -\frac{\ln\left(\frac{0.03}{2\cdot 100}\right)}{2 \cdot 0.05^2} = 1760.9 \tag{A.36}$$

Exercise 3.4

Correct answer: [b]
Explanation
We consider the general case of the Perceptron Model (linear classifiers) in $\mathbb{R}^D$. First, we show that the break point is greater than $D + 1$. We do this, by stating a winning strategy for the game that we discussed in section 2.5.

Opponent Picks the number $k = D + 1$

We Pick the points $\mathbf{x}_{D+1} = 0$ and $\mathbf{x}_i = \mathbf{e}_i = (\delta_{i,j})_{j=1}^{D}$ for $i = 1, ..., D$

Opponent Picks a dichotomy $d : \{\mathbf{x}_1, ..., \mathbf{x}_{D+1}\} \mapsto \{-1, 1\}$

We We choose the weight vector $\tilde{\mathbf{w}}$ as follows:

$$\tilde{w}_i = d(\mathbf{x}_i) \text{ for } i = 1, ..., D \tag{A.37}$$
$$\tilde{w}_{D+1} = 0.5 d(\mathbf{x}_{D+1}) \tag{A.38}$$

Winner If we can show $h(\mathbf{x}_i) = d(\mathbf{x}_i)$, we have a winning strategy. First, let $i \in \{1, ..., D\}$. Then we get

$$h(\mathbf{x}_i) = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i) = \text{sign}\left(\sum_{j=1}^{D+1} \tilde{w}_j \tilde{x}_{i,j}\right) \tag{A.39}$$
$$= \text{sign}\left(\sum_{j=1}^{D+1} d(\mathbf{x}_j) \tilde{x}_{i,j}\right) \tag{A.40}$$
$$= \text{sign}\left(\sum_{j=1}^{D} d(\mathbf{x}_j) \delta_{i,j} + 0.5 d(\mathbf{x}_{D+1})\right) \tag{A.41}$$
$$= \text{sign}\left(d(\mathbf{x}_i) + d(\mathbf{x}_{D+1})\right) = d(\mathbf{x}_i) \tag{A.42}$$

Now consider $\mathbf{x}_{D+1} = 0$.

$$h(\mathbf{x}_{D+1}) = \text{sign}(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_{D+1}) = \text{sign}(d(\mathbf{x}_{D+1})) = d(\mathbf{x}_{D+1}) \tag{A.43}$$

Therefore, the break point is greater than $D+1$. Now, we show that for $k = D+2$, we no longer have a winning strategy. Obviously, if we did not choose the points to be distinct, then we cannot have a winning strategy. If we chose the $D+2$ distinct points , there are at least 3 points that are linearly dependent. Hence, the opponent can divide the points into two sets T_1 and T_{-1} such that their convex hulls intersect. If this is the case, we cannot find a separating hyper plane. Otherwise, the convex hulls did not intersect. Therefore, the break point is $D + 2$.

In the exercise, we have $D = 3$. Therefore, the break point is $D + 2 = 3 + 2 = 5$.

Exercise 3.5

Correct answer: [b]
Explanation Let us check the individual answer options and see whether or not they are possible growth functions.

i) We saw in the lecture that this is the growth function of the positive rays hypothesis set.

ii) This is the growth function if we combine the positive rays set with the positive intervals set.

iii) This is not a growth function. First, we see that $k = 1$ is the break point of this function because $m_3(1) = 1 < 2^1 = 2$. Thus, if this was a growth function, it must grow polynomially. Formally, this means

$$m_3 = \bigcup_{k \in \mathbb{N}} \mathcal{O}(N^k) \tag{A.44}$$

If this holds, then there must be a smallest k_0 such that

$$m_3 = \mathcal{O}(N^k) \; \forall \; k \geq k_0 \tag{A.45}$$

This is equivalent to

$$\lim_{N \to \infty} \frac{m_3(N)}{N^k} < \infty \; \forall \; k \geq k_0 \tag{A.46}$$

If we find a lower bound on m_3 for which this condition does not hold, then it cannot hold for m_3 either. We now derive such a lower bound.

First, we note that we can lower bound a binomial coefficient as follows

$$\left(\begin{array}{c} N \\ k \end{array} \right) = \frac{N(N-1)\dots(N-k+1)}{1 \dots k} = \underbrace{\frac{N}{k}}_{\geq \frac{N}{k}} \cdot \underbrace{\frac{N-1}{k-1}}_{\geq \frac{N}{k}} \cdots \underbrace{\frac{N-k+1}{1}}_{\geq \frac{N}{k}} \geq \left(\frac{N}{k} \right)^k \tag{A.47}$$

Obviously, m_3 is greater or equal to the largest term of the sum.

$$m_3(N) \geq \left(\begin{array}{c} N \\ \lfloor \sqrt{N} \rfloor \end{array} \right) \tag{A.48}$$

We can now apply the bound on the binomial coefficient that we derived before.

$$\left(\begin{array}{c} N \\ \lfloor \sqrt{N} \rfloor \end{array} \right) \geq \left(\frac{N}{\lfloor \sqrt{N} \rfloor} \right)^{\lfloor \sqrt{N} \rfloor} = \Theta \left(\sqrt{N}^{\sqrt{N}} \right) \tag{A.49}$$

where $\Theta(f)$ is the class of all functions that grow as fast as f (The floor operation is not important in this asymptotic discussion). If we can show that $f(N) = \sqrt{N}^{\sqrt{N}}$ does not grow polynomially, then we showed that m_3 does not grow polynomially. For this purpose, let us assume that f grows polynomially. Hence, there is a $k_0 \in \mathbb{N}$ such that

$$\lim_{N \to \infty} \frac{f(N)}{N^{k_0}} < \infty \tag{A.50}$$

Instead of looking at this limit, we look at the limit

$$\lim_{x \to \infty} \frac{\tilde{f}(x)}{x^{k_0}} \tag{A.51}$$

where $\tilde{f} : \mathbb{R} \mapsto \mathbb{R}, x \mapsto \sqrt{x}^{\sqrt{x}}$. If this limit diverges, then the sequence also diverges. It holds

- $\lim_{x \to \infty} \tilde{f}(x) = \infty$
- $\lim_{x \to \infty} x^{k_0} = \infty$
- $\tilde{f}$ and x^{k_0} are differentiable

- $x^{k_0} \neq 0$ for $x > 0$

This allows us to use the theorem of l'Hospital. Thus, if

$$\lim_{x\to\infty} \frac{\frac{d}{dx}\tilde{f}(x)}{\frac{d}{dx}x^{k_0}} \tag{A.52}$$

converges, then the limit in (A.51) also converges and limits are equal. We can write $\tilde{f}$ as follows

$$\tilde{f}(x) = \exp(\sqrt{x}\ln\sqrt{x}) \tag{A.53}$$

This allows us to make the following observations.

- $\frac{d^k}{dx^k}\tilde{f}$ and x^{k_0} are differentiable for $k \leq k_0$
- $\lim_{x\to\infty} \frac{d^k}{dx^k}\tilde{f}(x) = \infty$ for $k < k_0$
- $\lim_{x\to\infty} \frac{d^k}{dx^k}x^{k_0} = \infty$ for $k < k_0$
- $\frac{d^k}{dx^k}x^{k_0} \neq 0$ for $x > 0$ for $k \leq k_0$

Therefore, we can apply the theorem k_0 times. Hence, we investigate the following limit.

$$\lim_{x\to\infty} \frac{\frac{d^{k_0}}{dx^{k_0}}\tilde{f}(x)}{\frac{d^{k_0}}{dx^{k_0}}x^{k_0}} = \lim_{x\to\infty} \frac{\frac{d^{k_0}}{dx^{k_0}}\tilde{f}(x)}{k_0!} = \infty \tag{A.54}$$

This implies (l'Hospital)

$$\lim_{x\to\infty} \frac{\frac{d^k}{dx^k}\tilde{f}(x)}{\frac{d}{dx}x^{k_0}} = \infty \tag{A.55}$$

for all $0 \leq k \leq k_0$. This is a contradiction to the assumption that k_0 was chosen such that the limit is finite. Hence, the assumption was wrong and f does not grow polynomially. Because f is a lower bound on m_3, m_3 also does not grow polynomially. Since m_3 had a finite break point, it cannot be a growth function.

iv) We see that $k = 1$ is a break point. Hence, m_4 is supposed to grow polynomially. However, we see that it grows exponentially in N. Therefore, it is not a growth function.

v) This is the growth function of the convex set hypothesis set.

Exercise 3.6

Correct answer: [c]
Explanation We use the result of exercise 3.7 and test different values for k.

$$k = 1 \Rightarrow m_{\mathcal{H}}(4) = 2 = 2^1 \tag{A.56}$$
$$k = 2 \Rightarrow m_{\mathcal{H}}(4) = 4 = 2^2 \tag{A.57}$$
$$k = 3 \Rightarrow m_{\mathcal{H}}(4) = 8 = 2^3 \tag{A.58}$$
$$k = 4 \Rightarrow m_{\mathcal{H}}(4) = 16 = 2^4 \tag{A.59}$$
$$k = 5 \Rightarrow m_{\mathcal{H}}(5) = 31 < 2^5 \tag{A.60}$$

Therefore, the break point is $k = 5$.

Exercise 3.7

Correct answer: [c]
Explanation We follow the technique used in the lecture and pick N distinct points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}$ such that $\mathbf{x}_i < \mathbf{x}_j$ for $i < j$. The dichotomies induced by two intervals can be counted as follows

- Both intervals do not cover any of the points
- The first interval I_1 is contained in the second interval I_2. In this case, the interval I_2 determines the number of hypothesis. There are $\begin{pmatrix} N+1 \\ 2 \end{pmatrix}$ possible choices for the boundaries.
- Both intervals are disjoint. There are $\begin{pmatrix} N+1 \\ 4 \end{pmatrix}$ possible choices for the boundaries.

Therefore, the growth function is given by

$$m_{\mathcal{H}}(N) = \begin{pmatrix} N+1 \\ 4 \end{pmatrix} + \begin{pmatrix} N+1 \\ 2 \end{pmatrix} + 1 \tag{A.61}$$

Exercise 3.8

Correct answer: [d]
Explanation If we consider $2M$ points $\mathbf{x}_1, ..., \mathbf{x}_{2M}$ with $\mathbf{x}_i < \mathbf{x}_j$ for $i < j$, for all possible dichotomies there are at most M blocks of points which are labeled 1. Hence, the break point is greater than $2M$. However, we cannot shatter $2M + 1$ points. If two points $\mathbf{x}_i$ and $\mathbf{x}_j$ overlap, we cannot find a hypothesis with $h(\mathbf{x}_i) \neq h(\mathbf{x}_j)$. If all points are distinct, then they are ordered. In this case, we cannot find a hypothesis with $h(\mathbf{x}_i) = 2(i \bmod 2) - 1$. Therefore, the break point is $2M + 1$.

Exercise 3.9

Correct answer: [d]
Explanation First of all, we show that we can shatter 7 points. Let $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^2$ be distinct points on a circle centered at the origin. For any dichotomy d on $\mathbf{x}_1, ..., \mathbf{x}_N$ there can be at most 3 blocks of points labeled 1. We can separate these points using three lines that form a triangle.
Now, we show that we cannot shatter 8 points. We consider two cases. If one point lies within the convex hull of the other, we cannot label the inner point -1 and the outer points 1. Hence, the points must be arranged such that no point lies within the convex hull of the others. If this is the case, we can order the points according to their angle α_i. However, in this case, we cannot label the points such as $d(\mathbf{x}_i) = 2(i \bmod 2) - 1$.

Exercise 3.10

Correct answer: [b]
Explanation
We show that this hypothesis set has the same expressive power as the hypothesis set of positive intervals that was discussed in the lecture. We do so, by showing that any positive interval can be expressed by the circles and vice versa. Let $\mathcal{H}_C$ be the hypothesis set of concentric positive circles and let $\mathcal{H}_I$ be the hypothesis set of positive intervals.

Let $\mathbf{x} \in \mathbb{R}^2$ and let $h_C \in \mathcal{H}_C$. It holds.

$$h_C(\mathbf{x}) = 1 \Leftrightarrow a^2 \leq x_1^2 + x_2^2 \leq b^2 \tag{A.62}$$
$$\Leftrightarrow a^2 \leq \|\mathbf{x}\|_2^2 \leq b^2 \tag{A.63}$$

Thus, we get an equivalent positive interval by applying the feature map $\phi(\mathbf{x}) = \|\mathbf{x}\|_2^2$ and then setting the boundaries to a^2 and b^2.
Now, let $\mathbf{x} \in \mathbb{R}$ and let $h_I \in \mathcal{H}_I$. It holds

$$h_i(\mathbf{x}) = 1 \Leftrightarrow \mathbf{x} \in [a, b] \tag{A.64}$$
$$\Leftrightarrow a \leq \mathbf{x} \leq b \tag{A.65}$$
$$\Leftrightarrow a^2 \leq \mathbf{x}^2 \leq b^2 \tag{A.66}$$
$$\Leftrightarrow a^2 \leq \|(\mathbf{x}, 0)^T\|_2^2 \leq b^2 \tag{A.67}$$

Thus, we get an equivalent concentric circle classifier by applying the feature map $\phi(\mathbf{x}) = (\mathbf{x}, 0)^T$ and then setting the boundaries to a^2 and b^2.
In the lecture, we saw that $\mathcal{H}_I$ has the growth function

$$m_{\mathcal{H}_I}(N) = \begin{pmatrix} N+1 \\ 2 \end{pmatrix} + 1 \tag{A.68}$$

A.4 Exercise 4

Exercise 4.1

Correct answer: [d]
Explanation We use the standard bound that we derived from the VC-inequality.

$$E_{out}(h) - E_{in}(h) \leq \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \tag{A.69}$$

Using the approximation $m_{\mathcal{H}}(N) = N^{d_{VC}}$ we get

$$E_{out}(h) - E_{in}(h) \leq \sqrt{\frac{8}{N} \ln \frac{4 \cdot 2^{d_{VC}} N^{d_{VC}}}{\delta}} \tag{A.70}$$

$$\leq \sqrt{\frac{8}{N}(\ln 4 + d_{VC}(\ln 2 + \ln N) - \ln \delta)} \tag{A.71}$$

where we did the second step for numerical stability. We want to determine N such that with probability $1 - \delta$ the bound is less than η.

$$\sqrt{\frac{8}{N}(\ln 4 + d_{VC}(\ln 2 + \ln N) - \ln \delta)} \overset{!}{\leq} \eta \tag{A.72}$$

$$\Leftrightarrow \quad \frac{8}{N}(\ln 4 + d_{VC}(\ln 2 + \ln N) - \ln \delta) \leq \eta^2 \tag{A.73}$$

$$\Leftrightarrow \quad 8(\ln 4 + d_{VC} \ln 2 - \ln \delta) \leq N\eta^2 - 8d_{VC} \ln N \tag{A.74}$$

Obviously, there is no closed form solution. However, we can approximate the minimal N. For the smallest $N^* \in \mathbb{R}$ it holds

$$8(\ln 4 + d_{VC} \ln 2 - \ln \delta) - N\eta^2 + 8d_{VC} \ln N = 0 \tag{A.75}$$

$$\Leftrightarrow \quad \underbrace{8(\ln 4 + d_{VC} \ln 2 - \ln \delta) - N\eta^2 + 8d_{VC} \ln N + N}_{\Phi(N)} = N \tag{A.76}$$

Therefore, the smallest N^* is a fixed point of Φ. Assuming that Φ is differentiable, we get that Φ is a contraction for all large N. Therefore, the fixed point iteration

$$N_0 := \text{ some large } N \in \mathbb{N} \tag{A.77}$$

$$N_{i+1} := \Phi(N_i) \tag{A.78}$$

converges and $N_i \to N^*$. After a sufficient number of iterations we get

$$N^* \approx 452957 \tag{A.79}$$

Execute `ex4_1`.

Exercise 4.2

Correct answer: [d]
Explanation We first have to rearrange the formulas to make them numerically stable.

(i)

$$\sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} = \sqrt{\frac{8}{N}(\ln 4 + \ln m_{\mathcal{H}}(2N) - \ln \delta)} \tag{A.80}$$

$$= \sqrt{\frac{8}{N}(\ln 4 + d_{VC}(\ln 2 + \ln N) - \ln \delta)} \tag{A.81}$$

(ii)

$$\sqrt{\frac{2\ln(2Nm_{\mathcal{H}}(N))}{N}}+\sqrt{\frac{2}{N}\ln\frac{1}{\delta}}+\frac{1}{N}=\sqrt{\frac{2(\ln 2+\ln N+d_{VC}\ln N)}{N}}+\sqrt{\frac{2}{N}\ln\frac{1}{\delta}}+\frac{1}{N} \tag{A.82}$$

(iii) This is an implicit bound on ϵ. We plot it for the largest ϵ for which the bound holds. It holds

$$\epsilon \leq \sqrt{\frac{1}{N}\left(2\epsilon+\ln\frac{6m_{\mathcal{H}}(2N)}{\delta}\right)} \tag{A.83}$$

$$\Rightarrow \quad \epsilon^2 \leq \frac{1}{N}\left(2\epsilon+\ln\frac{6m_{\mathcal{H}}(2N)}{\delta}\right) \tag{A.84}$$

$$\Rightarrow \quad \epsilon^2-\frac{1}{N}2\epsilon-\frac{1}{N}\ln\frac{6m_{\mathcal{H}}(2N)}{\delta}\leq 0 \tag{A.85}$$

$$\Rightarrow \quad \epsilon \gtrless \frac{1}{N}\pm\sqrt{\frac{1}{N^2}+\frac{1}{N}\ln\frac{6m_{\mathcal{H}}(2N)}{\delta}} \tag{A.86}$$

We define a function $\epsilon(N) := \frac{1}{N}\pm\sqrt{\frac{1}{N^2}+\frac{1}{N}\ln\frac{6m_{\mathcal{H}}(2N)}{\delta}}$. Then, we can rearrange the bound as follows.

$$\sqrt{\frac{1}{N}\left(2\epsilon(N)+\ln\frac{6m_{\mathcal{H}}(2N)}{\delta}\right)}=\sqrt{\frac{1}{N}\left(2\epsilon(N)+\ln 6+d_{VC}(\ln 2+\ln N)-\ln\delta\right)} \tag{A.87}$$

(iv) This is also an implicit bound on ϵ. We again determine the largest ϵ for which the bound holds.

$$\epsilon \leq \sqrt{\frac{1}{2N}\left(4\epsilon(1+\epsilon)+\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}\right)} \tag{A.88}$$

$$\Rightarrow \quad \epsilon^2 \leq \frac{1}{2N}\left(4\epsilon(1+\epsilon)+\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}\right) \tag{A.89}$$

$$\Rightarrow \quad \epsilon^2-\frac{2}{N}\epsilon-\frac{2}{N}\epsilon^2-\frac{1}{2N}\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}\leq 0 \tag{A.90}$$

$$\Rightarrow \quad \epsilon^2-\frac{2}{N-2}\epsilon-\frac{1}{2N-4}\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}\leq 0 \tag{A.91}$$

$$\Rightarrow \quad \epsilon \gtrless \frac{1}{N-2}\pm\sqrt{\frac{1}{(N-2)^2}+\frac{1}{2N-4}\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}} \tag{A.92}$$

With $\epsilon(N) := \frac{1}{N-2}+\sqrt{\frac{1}{(N-2)^2}+\frac{1}{2N-4}\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}}$ we get the following bound that we plot.

$$\sqrt{\frac{1}{2N}\left(4\epsilon(N)(1+\epsilon(N))+\ln\frac{4m_{\mathcal{H}}(N^2)}{\delta}\right)}=\sqrt{\frac{1}{2N}\left(4\epsilon(N)(1+\epsilon(N))+\ln 4+2d_{VC}\ln N-\ln\delta\right)} \tag{A.93}$$

Execute `ex4_2`.

Exercise 4.3

Correct answer: [c]
Explanation Because $N < d_{VC}$ in this case, we have $m_{\mathcal{H}}(5) = 2^5$. [1]
Execute `ex4_3`.

Exercise 4.4

Correct answer: [e]
Explanation We want to fit the line $h(x) = ax$ such that the mean squared error is minimized. Hence, out objective is to determine

$$\min_{a \in \mathbb{R}} \underbrace{(ax_1 - f(x_1))^2 + (ax_2 - f(x_2))^2}_{E(a)} \tag{A.94}$$

A necessary condition is $E'(a) = 0$.

$$E'(a) = 2(ax_1 - f(x_1))x_1 + 2(ax_2 - f(x_2))x_2 \stackrel{!}{=} 0 \tag{A.95}$$

$$\Leftrightarrow \quad (x_1^2 + x_2^2)a = f(x_1)x_1 + f(x_2)x_2 \tag{A.96}$$

$$\Leftrightarrow \quad a = \frac{f(x_1)x_1 + f(x_2)x_2}{x_1^2 + x_2^2} \tag{A.97}$$

Because E is convex, this is the global optimum. For the exercise, we randomly choose two points and then average over a. Execute `ex4_4`.

Exercise 4.5

Correct answer: [b]
Explanation Execute `ex4_5`.

Exercise 4.6

Correct answer: [a]
Explanation Execute `ex4_6`.

Exercise 4.7

Correct answer: [b]
Explanation Execute `ex4_7`.

Exercise 4.8

Correct answer: [c]
Explanation We note that $\begin{pmatrix} N \\ q \end{pmatrix} = 0$ for $q > N$. Hence, for all $N < q$, it holds

$$m_{\mathcal{H}}(N+1) = 2m_{\mathcal{H}}(N) = 2^{N+1} \tag{A.98}$$

For $N = q$, we get $\begin{pmatrix} N \\ q \end{pmatrix} = \begin{pmatrix} N \\ N \end{pmatrix} = 1$ and this yields

$$m_{\mathcal{H}}(q) = 2^q - 1 < 2^q \tag{A.99}$$

Therefore, the break point is q.

[1] Thanks to jlloyd for pointing this out.

Exercise 4.9

Correct answer: [b]
Explanation
For this exercise, we need the following lemmas and corollaries.

Lemma 3. *Let $\mathcal{H}_1, \mathcal{H}_2$ be two hypothesis sets with finite VC-dimensions. Then it holds*

$$d_{VC}(\mathcal{H}_1 \cup \mathcal{H}_2) \leq d_{VC}(\mathcal{H}_1) + d_{VC}(\mathcal{H}_2) + 1 \tag{A.100}$$

Proof. Obviously, it holds

$$m_{\mathcal{H}_1 \cup \mathcal{H}_2}(N) \leq m_{\mathcal{H}_1}(N) + m_{\mathcal{H}_2}(N) \tag{A.101}$$

Because $\mathcal{H}_1$ and $\mathcal{H}_2$ have finite VC-dimensions, we know from the lecture that the growth function of the union is bounded as follows

$$m_{\mathcal{H}_1 \cup \mathcal{H}_2}(N) \leq m_{\mathcal{H}_1}(N) + m_{\mathcal{H}_2}(N) \leq \sum_{i=0}^{d_{VC}(\mathcal{H}_1)} \binom{N}{i} + \sum_{i=0}^{d_{VC}(\mathcal{H}_2)} \binom{N}{i} \tag{A.102}$$

$$= \sum_{i=0}^{d_{VC}(\mathcal{H}_1)} \binom{N}{i} + \sum_{i=0}^{d_{VC}(\mathcal{H}_2)} \binom{N}{N-i} \tag{A.103}$$

$$= \sum_{i=0}^{d_{VC}(\mathcal{H}_1)} \binom{N}{i} + \sum_{i=N-d_{VC}(\mathcal{H}_2)}^{N} \binom{N}{i} \tag{A.104}$$

If $m_{\mathcal{H}_1 \cup \mathcal{H}_2}(d_{VC}(\mathcal{H}_1) + d_{VC}(\mathcal{H}_2) + 2) < 2^{d_{VC}(\mathcal{H}_1)+d_{VC}(\mathcal{H}_2)+2}$, then the VC-dimension of $\mathcal{H}_1 \cup \mathcal{H}_2$ is at most $d_{VC}(\mathcal{H}_1) + d_{VC}(\mathcal{H}_2) + 1$. Let us examine the growth function for $N = d_{VC}(\mathcal{H}_1) + d_{VC}(\mathcal{H}_2) + 2$. It holds

$$m_{\mathcal{H}_1 \cup \mathcal{H}_2}(\mathcal{H}_1 + d_{VC}(\mathcal{H}_2) + 2) \leq \sum_{i=0}^{d_{VC}(\mathcal{H}_1)} \binom{N}{i} + \sum_{i=d_{VC}(\mathcal{H}_1)+2}^{N} \binom{N}{i} \tag{A.105}$$

$$= \sum_{i=0}^{N} \binom{N}{i} - \underbrace{\binom{N}{d_{VC}(\mathcal{H}_1)+1}}_{>0} < 2^N \tag{A.106}$$

□

Corollary 4. *Let $\mathcal{H}_1, \mathcal{H}_2$ be two hypothesis sets with finite VC-dimensions. Then it holds*

$$\min(d_{VC}(\mathcal{H}_1), d_{VC}(\mathcal{H}_2)) \leq \max(d_{VC}(\mathcal{H}_1), d_{VC}(\mathcal{H}_2)) \leq d_{VC}(\mathcal{H}_1 \cup \mathcal{H}_2) \tag{A.107}$$

Corollary 5. *Let $\mathcal{H}$ be a hypothesis set with finite VC-dimension. For all $H \subseteq \mathcal{H}$ it holds*

$$d_{VC}(H) \leq d_{VC}(\mathcal{H}) \tag{A.108}$$

Corollary 6. *Let $\mathcal{H}_1, \mathcal{H}_2$ be two hypothesis sets with finite VC-dimensions. Then for all $H \subseteq \mathcal{H}_1 \cap \mathcal{H}_2$ it holds*

$$d_{VC}(H) \leq d_{VC}(\mathcal{H}_1) + d_{VC}(\mathcal{H}_2) \tag{A.109}$$

a) This bound is correct:

$$0 = d_{VC}(\emptyset) \overset{Cor.5}{\leq} d_{VC}\left(\bigcap_{k=1}^{K} \mathcal{H}_k\right) \overset{Cor.6}{\leq} \sum_{k=1}^{K} d_{VC}(\mathcal{H}_k) \tag{A.110}$$

b) This bound is correct because $\bigcap_{k=1}^{K} \mathcal{H}_k \subseteq \mathcal{H}_i$ for all i. Therefore, the bound follows with corollary 5.

c) This bound is correct for the same reasons the last one was correct.

d) This bound is not correct. Let $\mathcal{H}_1, \mathcal{H}_2$ be two disjoint hypothesis sets. Then it holds

$$d_{VC}(\mathcal{H}_1 \cap \mathcal{H}_2) = d_{VC}(\emptyset) = 0 \leq \min(d_{VC}(\mathcal{H}_1), d_{VC}(\mathcal{H}_2)) \tag{A.111}$$

which contradicts the bound.

e) This is incorrect for the reason the last one was incorrect.

The tightest bound is bound b.

Exercise 4.10

Correct answer: [e]
Explanation

a) This bound is incorrect because the bound of lemma 3 is tight.

b) This bound is correct.

$$0 = d_{VC}(\emptyset) \overset{Cor.5}{\leq} d_{VC}\left(\bigcup_{k=1}^{K} \mathcal{H}_k\right) \overset{Lem.3}{\leq} K - 1 + \sum_{k=1}^{K} d_{VC}(\mathcal{H}_k) \tag{A.112}$$

c) This bound is not correct for the same reason the first bound was not correct.

d) This bound is not correct for the same reason the first bound was not correct.

e) This bound is correct.

$$\max_{k=1,\ldots,K} d_{VC}(\mathcal{H}_k) \overset{Cor.4}{\leq} d_{VC}\left(\bigcup_{k=1}^{K} \mathcal{H}_k\right) \overset{Lem.3}{\leq} K - 1 + \sum_{k=1}^{K} d_{VC}(\mathcal{H}_k) \tag{A.113}$$

The tightest bound is bound e.

A.5 Exercise 5

Exercise 5.1

Correct answer: [c]
Explanation We have to solve the following equation for N where δ is our desired lower bound on $\mathbb{E}[E_{in}(\mathbf{w}_{lin})]$.

$$\mathbb{E}[E_{in}(\mathbf{w}_{lin})] \geq \delta \tag{A.114}$$

$$\Leftrightarrow \quad \sigma^2 \left(1 - \frac{d+1}{N}\right) \geq \delta \tag{A.115}$$

$$\Leftrightarrow \quad \frac{d+1}{N} \leq 1 - \frac{\delta}{\sigma^2} \tag{A.116}$$

$$\Leftrightarrow \quad \frac{d+1}{1 - \frac{\delta}{\sigma^2}} \leq N \tag{A.117}$$

In order to get the answer, we have to plug in the values that are given by the exercise.

- $\sigma = 0.1$
- $d = 8$
- $\delta = 0.008$

Thus, we get

$$N \geq 45 \tag{A.118}$$

Exercise 5.2

Correct answer: [d]
Explanation We can already exclude any answers with $w_1 = 0$ or $w_2 = 0$ because in these cases the decision boundary is not hyperbolic. Thus, we assume $w_1 \neq 0 \neq w_2$.
We know that the decision boundary is defined as

$$\mathbf{w}^T \Phi(\mathbf{x}) = 0 \tag{A.119}$$

Thus, the decision boundary is the 0 level set of

$$F : \mathbb{R}^2 \mapsto \mathbb{R}, \mathbf{x} \mapsto w_0 + w_1 x_1^2 + w_2 x_2^2 \tag{A.120}$$

We know from the implicit function theorem that there is a function

$$T : U \mapsto V \tag{A.121}$$

such that $F(T(x_2), x_2) = 0$ for all $x_2 \in U$ where $V \times U$ is a neighborhood of a point $\mathbf{x}$ on the decision boundary. In this case, we can solve explicitly for T. It holds

$$w_0 + w_1 T(x_2)^2 + w_1 x_2^2 = 0 \tag{A.122}$$

$$\overset{w_1 \neq 0}{\Leftrightarrow} \quad T(x_2)^2 = \frac{-w_2 x_2^2 - w_0}{w_1} \tag{A.123}$$

$$\Leftrightarrow \quad T(x_2) = \pm \sqrt{\frac{-w_2 x_2^2 - w_0}{w_1}} \tag{A.124}$$

In order for this term to be a real valued function, it must hold

$$-\frac{w_2 x_2^2 + w_0}{w_1} \geq 0 \; \forall x_2 \in U \tag{A.125}$$

We can see two things: (1) The implicit function holds for all $x_2 \in \mathbb{R}$; (2) The answer depends on the sign of w_0. From the graphic, we see that the point $(0, 0)$ is assigned to class 1. Therefore, we get $w_0 > 0$.
Because the implicit function holds for all $x_2 \in \mathbb{R}$, it must hold for $x_2 = 0$ and thus

$$-\frac{w_2 \cdot 0 + w_0}{w_1} > 0 \Rightarrow w_1 < 0 \tag{A.126}$$

Because for large x_2 it holds $|w_2 x_2^2| > w_0$ it must also hold that $w_2 > 0$. Therefore, answer d is correct.

Exercise 5.3

Correct answer: [c]
Explanation The feature map has $D = 14$ entries plus the constant term. From the lecture we know that the VC-dimension is $1 + D$. Therefore, answer c is correct.

Exercise 5.4

Correct answer: [e]
Explanation It holds

$$\frac{\partial}{\partial u} E(u, v) = \frac{\partial}{\partial u} (u e^v - 2v e^{-u})^2 = 2(u e^v - 2v e^{-u})(e^v + 2v e^{-u}) \tag{A.127}$$

Exercise 5.5

Correct answer: [d]
Explanation For this exercise, we need the gradient. Because we already know $\frac{\partial}{\partial u} E(u, v)$, we only need to compute $\frac{\partial}{\partial v} E(u, v)$. It holds

$$\frac{\partial}{\partial v} E(u, v) = \frac{\partial}{\partial v} (u e^v - 2v e^{-u})^2 = 2(u e^v - 2v e^{-u})(u e^v - 2e^{-u}) \tag{A.128}$$

Thus, the update formula for gradient descent is given by

$$\begin{pmatrix} u^{(n+1)} \\ v^{(n+1)} \end{pmatrix} = \begin{pmatrix} u^{(n)} \\ v^{(n)} \end{pmatrix} - \eta \begin{pmatrix} 2(u^{(n)} e^{v^{(n)}} - 2v^{(n)} e^{-u^{(n)}})(e^{v^{(n)}} + 2v^{(n)} e^{-u^{(n)}}) \\ 2(u^{(n)} e^{v^{(n)}} - 2v^{(n)} e^{-u^{(n)}})(u^{(n)} e^{v^{(n)}} - 2e^{-u^{(n)}}) \end{pmatrix} \tag{A.129}$$

Execute `ex5_5`

Exercise 5.6

Correct answer: [e]
Explanation Execute `ex5_6`.

Exercise 5.7

Correct answer: [a]
Explanation Execute `ex5_7`.

Exercise 5.8

Correct answer: [d]
Explanation Execute `ex5_8`

Exercise 5.9

Correct answer: [a]
Explanation Execute `ex5_9`

Exercise 5.10

Correct answer: [e]
Explanation The loss function should be designed such that

$$\nabla e_n(\mathbf{w}) = \begin{cases} -y\mathbf{x} & \text{if } \operatorname{sign}(\mathbf{w}^T\mathbf{x}) \neq y \\ 0 & \text{otherwise} \end{cases} \tag{A.130}$$

Because the PLA always picks a misclassified point. If we chose a point at random, we can simulate this behaviour, if the update is in this case 0.
Obviously, answer e fulfills these requirements. If $\operatorname{sign}(\mathbf{w}^T\mathbf{x}) \neq y$, then $y \cdot \operatorname{sign}(\mathbf{w}^T\mathbf{x}) = -1$ and therefore $y\mathbf{w}^T\mathbf{x} < 0$. Thus, we get $e_n(\mathbf{w}) = -\min(0, y\mathbf{w}^T\mathbf{x}) = 0$ in this case. If however $y = \operatorname{sign}(\mathbf{w}^T\mathbf{x})$, then $y\mathbf{w}^T\mathbf{x} > 0$ and $e_n(\mathbf{w}) = -\min(0, y\mathbf{w}^T\mathbf{x}) = -y\mathbf{w}^T\mathbf{x}$.

A.6 Exercise 6

While I am writing this script, I am still a full time graduate student in computer science and mathematics. Unfortunately, I did not have the time to write the solutions to this exercise during week 6 due to other lectures I am taking at my university. I will publish the solutions in the week after the finals.

A.7 Exercise 7

Exercise 7.1

Correct answer: [d]
Explanation Execute `ex7_1`.

Exercise 7.2

Correct answer: [e]
Explanation Execute `ex7_2`.

Exercise 7.3

Correct answer: [d]
Explanation Execute `ex7_3`.

Exercise 7.4

Correct answer: [d]
Explanation Execute `ex7_4`.

Exercise 7.5

Correct answer: [b]
Explanation Execute `ex7_2` and `ex7_4`.

Exercise 7.6

Correct answer: [d]
Explanation Let $e_1, e_2 \overset{iid}{\sim} \mathcal{U}_{[0,1]}$ and let $e := \min(e_1, e_2)$. We have

$$\mathbb{E}[e_1] = \mathbb{E}[e_2] = \int_{-\infty}^{\infty} xp(x)\mathrm{d}x = \int_0^1 x = \frac{1}{2}x^2|_0^1 = \frac{1}{2} \tag{A.131}$$

Furthermore, we get

$$\mathbb{E}[e] = \mathbb{E}[\min(e_1, e_2)] = \int_0^1 \int_0^1 \min(x, y) \underbrace{p(x)p(y)}_{=1} \mathrm{d}x\mathrm{d}y \tag{A.132}$$

Let $y \in [0, 1]$ be fixed. Then we have

$$\int_0^1 \min(x, y)\mathrm{d}x = \int_0^y x\mathrm{d}x + \int_y^1 y\mathrm{d}x = \frac{1}{2}y^2 + y - y^2 = y - \frac{1}{2}y^2 \tag{A.133}$$

Thus, we can calculate the original integral.

$$\int_0^1 \int_0^1 \min(x, y)\mathrm{d}x\mathrm{d}y = \int_0^1 y - \frac{1}{2}y^2\mathrm{d}y = \frac{1}{3} \tag{A.134}$$

Therefore, answer d is correct.

Exercise 7.7

Correct answer: [c]
Explanation We want to determine ρ such that

$$\sum_{i=1}^{3}(h_0^i(\mathbf{x}_i) - y_i)^2 = \sum_{i=1}^{3}(h_1^i(\mathbf{x}_i) - y_i)^2 \tag{A.135}$$

where h_j^i is the final hypothesis that is obtained by using all points but $\mathbf{x}_i$. Let us determine the individual hypotheses. Firstly, let us consider h_0. We want to minimize the squared error on two points. Thus, our goal is to solve the following optimization problem.

$$\min_{h(\mathbf{x})=b}(b - y_{i_1})^2 + (b - y_{i_2})^2 \tag{A.136}$$

The necessary condition yields

$$2(b - y_{i_1}) + 2(b - y_{i_2}) \stackrel{!}{=} 0 \tag{A.137}$$

$$\Leftrightarrow \quad 2b = y_{i_1} + y_{i_2} \tag{A.138}$$

$$\Leftrightarrow \quad b = \frac{1}{2}(y_{i_1} + y_{i_2}) \tag{A.139}$$

- $y_{i_1} = 1, y_{i_2} = 0$

  $$h_0^1(\mathbf{x}) = \frac{1}{2}(1 + 0) = \frac{1}{2} \tag{A.140}$$

  Error:

  $$e_0^1 = \left(\frac{1}{2} - 0\right)^2 = \frac{1}{4} \tag{A.141}$$

- $y_{i_1} = 0, y_{i_2} = 0$

  $$h_0^2(\mathbf{x}) = \frac{1}{2}(0 + 0) = 0 \tag{A.142}$$

  Error:

  $$e_0^2 = (0 - 1)^2 = 1 \tag{A.143}$$

- $y_{i_1} = 0, y_{i_2} = 1$

  $$h_0^3(\mathbf{x}) = \frac{1}{2}(0 + 1) = \frac{1}{2} \tag{A.144}$$

  Error:

  $$e_0^3 = \left(\frac{1}{2} - 0\right)^2 = \frac{1}{4} \tag{A.145}$$

Let us now consider the the second hypothesis. We again want to minimize the squared error. Thus, the optimization problem is

$$\min_{h(\mathbf{x})=m\mathbf{x}+b}\underbrace{(m\mathbf{x}_{i_1} + b - y_{i_1})^2 + (m\mathbf{x}_{i_2} + b - y_{i_2})^2}_{E(m,b)} \tag{A.146}$$

Obviously, we can achieve the minimum by fitting a line that passes through both points. Thus, we have

$$m = \frac{y_{i_1} - y_{i_2}}{x_{i_1} - x_{i_2}} \tag{A.147}$$

$$b = y_{i_1} - m x_{i_1} \text{ or } b = y_{i_2} - m x_{i_2} \tag{A.148}$$

- $x_{i_1} = \rho, x_{i_2} = 1, y_{i_1} = 1, y_{i_2} = 0.$

$$m = \frac{1 - 0}{\rho - 1} \tag{A.149}$$

$$b = -\frac{1}{\rho - 1} \tag{A.150}$$

  Error:

$$e_1^1 = \left(-\frac{1}{\rho - 1} - \frac{1}{\rho - 1} - 0 \right)^2 = \frac{4}{(\rho - 1)^2} \tag{A.151}$$

- $x_{i_1} = -1, x_{i_2} = 1, y_{i_1} = 0, y_{i_2} = 0.$

$$m = 0 \tag{A.152}$$

$$b = 0 \tag{A.153}$$

  Error:

$$e_1^2 = (0 - 1)^2 = 1 \tag{A.154}$$

- $x_{i_1} = \rho, x_{i_2} = -1, y_{i_1} = 1, y_{i_2} = 0.$

$$m = \frac{1 - 0}{\rho + 1} \tag{A.155}$$

$$b = \frac{1}{\rho + 1} \tag{A.156}$$

  Error:

$$e_1^3 = \left(\frac{1}{\rho + 1} + \frac{1}{\rho + 1} - 0 \right)^2 = \frac{4}{(\rho + 1)^2} \tag{A.157}$$

We now want to determine ρ such that

$$\sum_{i=1}^{3} e_0^i = \sum_{i=1}^{3} e_1^i \tag{A.158}$$

Solving for ρ yields

$$\frac{3}{2} = \frac{4}{(\rho + 1)^2} + \frac{4}{(\rho - 1)^2} + 1 \tag{A.159}$$

$$\Leftrightarrow \quad \frac{1}{2} = \frac{4}{(\rho + 1)^2} + \frac{4}{(\rho - 1)^2} \tag{A.160}$$

$$\Leftrightarrow \quad 0 = \frac{8(\rho + 1)^2 + 8(\rho - 1)^2 - (\rho + 1)^2(\rho - 1)^2}{(\rho + 1)^2(\rho - 1)^2} \tag{A.161}$$

$$\Leftrightarrow \quad 0 = 16\rho^2 + 16 - (\rho^2 + 2\rho + 1)(\rho^2 - 2\rho + 1) \tag{A.162}$$

$$\Leftrightarrow \quad 0 = 16\rho^2 + 16 - \rho^4 + 2\rho^3 - \rho^2 - 2\rho^3 + 4\rho^2 - 2\rho - \rho^2 + 2\rho - 1 \tag{A.163}$$

$$\Leftrightarrow \quad 0 = -\rho^4 + 18\rho^2 + 15 \tag{A.164}$$

Therefore

$$\rho = \pm\sqrt{9 + 4\sqrt{6}} \tag{A.165}$$

Because $\rho > 0$ and $\rho \in \mathbb{R}$, the answer is c.

Exercise 7.8

Correct answer: [c]
Explanation Execute `ex7_8`.

Exercise 7.9

Correct answer: [d]
Explanation Execute `ex7_9`.

Exercise 7.10

Correct answer: [b]
Explanation Execute `ex7_10`.

Bibliography

[AM] Yaser S. Abu-Mostafa. Learning from data. https://www.youtube.com/playlist?list=PLD63A284B7615313A.

[AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012.

[BBL04] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 169–207. Springer Berlin Heidelberg, 2004.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998.

[Col] Michael Collins. Convergence proof for the perceptron algorithm. http://www.cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf.

[GVL13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (4th Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 2013.

[LAM] Hsuan-Tien Lin and Yaser S. Abu-Mostafa. Proof of the vc inequality. http://www.csie.ntu.edu.tw/~htlin/course/ml08fall/doc/vc_proof.pdf.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[Ros57] Frank Rosenblatt. The perceptron–a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.