Contents

Introduction

Programme of the course

Arguments treated:

1. **Motivation**: components of the learning problem and applications of Machine Learning. Supervised and unsupervised learning.
2. **Introduction**: the supervised learning problem, data, classes of models, losses.
3. **Probabilistic models and assumptions on the data**: the regression function. Regression and classification.
4. **When is a model good?**: model complexity, bias variance tradeoff/generalization (VC dimension, generalization error).
5. **Models for regression**: linear regression (scalar and multivariate), subset selection, linear-in-the-parameters models, regularization.
6. **Simple models for classification**: logistic regression, perceptron, naïve bayes classifier.
7. **Kernel methods**: Support Vector Machines.
8. **Neural Networks**.
9. **Deep Learning**: Convolutional Neural Networks.
10. **Validation and model selection**: generalization error, bias-variance tradeoff, cross validation. Model complexity determination.
11. **Unsupervised learning**: cluster analysis, K-means clustering, EM estimation.
12. **Dimensionality reduction**: Principal Component Analysis (PCA).

Chapter 1

Machine Learning framework

1.1 A formal model

We begin with the description of a formal model in order to capture what could be the learning tasks. The fundamental points are:

- **The learner's input**:
  - A domain set $\mathcal{X}$, whose points are the istances we want to label.
  - A label set $\mathcal{Y}$.
  - The training dataset $S = \mathcal{X} \times \mathcal{Y}$. It is a finite sequence of label domain points.
- **The learner's output**:
  - A prediction rule $h : \mathcal{X} \to \mathcal{Y}$ (also called predictor or hyothesis or classifier). It is used to predict the label of new domain points. Therefore $A(S)$ is the hypothesis, where A represents the algorithm.
  - **Simple data-generation model**:
    Assume that the training data are generated by a probability distribution $\mathbb{D}$ (over $\mathcal{X}$). Moreover, suppose that the learner doesn't know anything about the distribution and that there exists some "correct" labeling function $f : \mathcal{X} \to \mathcal{Y}$ such that $y_i = f(x_i)\ \forall i$ (and it is unknown to the learner as well). So, in summary each S is generated by first sampling a point x_i according to $\mathcal{D}$ and then labeling it with the function f.
- **Measures of success**:
  A definition of "**Error of the classifier**" sohuld be introduced: it is the probability to draw a random instance x, according to D, such that $h(x) \neq f(x)$. Formally: $A \subset \mathcal{X} \implies \mathcal{D}(A)$ determines how likely it is to observe a $x \in A$, where:
  $$A = \{x \in \mathcal{X} \ : \ \pi(x) = 1\} \quad \text{with} \quad \pi \ : \ \mathcal{X} \to \{0,1\}$$
  We refer to A as an event and therefore:
  $$\mathcal{D}(A) \equiv \mathbb{P}_{x \sim \mathcal{D}}[\pi(x)]$$
  The error of $h \ : \ \mathcal{X} \to \mathcal{Y}$ is finally:
  $$L_{\mathcal{D},f}(h) \equiv \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq f(x)] \equiv \mathcal{D}(x \ : \ h(x) \neq f(x))$$
  Note that $(\mathcal{D}, f)$ means that the error is computed with respect to the probability distribution $\mathcal{D}$ and the correct labeling function f. Moreover, remind that the learner is blind to the $\mathcal{D}$ and f.

1.2 Empirical Risk Minimization (ERM)

As mentioned before, a learning algorithm receives as input a training set S, sampled from an unknown distriibution D and labeled by some target function f, and should output a predictor $h_S : \mathcal{X} \to \mathcal{Y}$. The goal of the algorithm is to find h_S that minimizes the error with respect to the unknown D and f. Since the learner doesn't know what D and f are, the true error is not directly available to the learner. However, we can define the useful notion of **training error**, i.e. the error the classifier incurs over the training sample:

$$L_S(h) := \frac{|\{i \in [m] \ : \ h(x_i) \neq y_i\}|}{m} \tag{1.1}$$

where $[m] = \{1, \dots, m\}$. Note that the terms *empirical error* and *empirical risk* are often used with the same meaning to indicate Eq. 1.1.

1.2.1 Something may go wrong: Overfitting

This approach can fail miserably if naively used and lead to this phenomenon. Our aim is to find some conditions that guarantee the absence of overfitting.
Intuitively, overfitting occurs when our hypotesis fits the training data "too well" and so our algorithm won't have good performances on a new never-seen dataset.

1.2.2 Empirical Risk Minimization with inductive bias

The ERM rule might lead to overfitting, so we have to find a way to rectify it. In other words, we have to find some conditions that guarantee no overfitting. It means that if the algorithm has good performances with respect to the training data, it is also highly likely to perform well over the underlying data distribution.
A common solution is to apply the ERM learning rule over a restricted search space. Formally, the learner should choose a set of predictors $\mathcal{H}$ before seeing the data. This is a **hypotesis class**. Then, given a training sample S, the ERM$_\mathcal{H}$ learner uses the ERM rule to choose a predictor $h \in \mathcal{H}$ with the lowest possible error over S, which means mathematically:

$$\text{ERM}_\mathcal{H}(S) \in \underset{h \in \mathcal{H}}{\text{argmin}} \ L_S(h) \tag{1.2}$$

Such restrictions are often denoted as **inductive bias**.

Finite hypotesis classes

Now we consider a finite hypothesis class $\mathcal{H}$ and denote with h_S the result obtained applying ERM$_\mathcal{H}$ to S, namely Eq. 1.2. We also make the following simplifying assumption:

Definition 1.2.1 (The realizability assumption)**.** There exists $h^* \in \mathcal{H}$ such that $L_{(\mathcal{D},f)}(h^*) = 0$.

Note that this assumption implies that with probability 1 over random samples S we have $L_S(h^*) = 0$, where the istances of S are sampled according to $\mathcal{D}$ and are labeled by f.
Another assumption has to be done, presented in the following definition.

Definition 1.2.2 (Independently and identically distributed)**.** The examples in the training set are independently and identically distributed (i.i.d.) according to the distribution $\mathcal{D}$ if every x_i in S is freshly sampled according to $\mathcal{D}$ and then labeled according to the labeling function f. We denote this assumption by $S \sim \mathcal{D}^m$, where m is the size of S and $\mathcal{D}^m$ denotes the probability over m-tuples induced by applying $\mathcal{D}$ to pick each element of the tuple independently of the other members of the tuple.

We want to give an upper bound of the probability to sample a m-tuple of istances that will lead to the failure of the learner. Therefore we consider the probability δ of getting a non representative sample S of the distribution. Through δ we define the **confidence parameter** $1 - \delta$. We also introduce an **accuracy parameter** ε, with this meaning:

- $L_{(\mathcal{D},f)}(h_S) > \varepsilon \Longrightarrow$ failure of the learner;
- $L_{(\mathcal{D},f)}(h_S) \leq \varepsilon \Longrightarrow$ success of the learner.

So we would like to upper bound:

$$\mathcal{D}^m(\{S|_x \ : \ L_{(\mathcal{D},f)}(h_S) > \varepsilon\}) \tag{1.3}$$

For this reason we consider the set of "bad" hypotheses $\mathcal{H}_B$:

$$\mathcal{H} = \{h \in \mathcal{H} \ : \ L_{(\mathcal{D},f)}(h_S) > \varepsilon\} \tag{1.4}$$

In addition, we consider the set of misleading samples:

$$M = \{S|_x \ : \ \exists h \in \mathcal{H}_B, \ L_S(h) = 0\} \tag{1.5}$$

namely, $\forall S|_x \in M \ \exists h \in \mathcal{H}_B$ that looks like a "good" hypothesis on $S|_x$, which means:

$$\{S|_x \ : \ L_{(\mathcal{D},f)}(h_S) > \varepsilon\} \subseteq M \Longrightarrow M = \bigcup_{h \in \mathcal{H}_B} \{S|_x \ : \ L_S(h) = 0\} \tag{1.6}$$

Hence:

$$\mathcal{D}^m(\{S_x \ : \ L_{(\mathcal{D},f)}(h_S) > \varepsilon\}) \leq \mathcal{D}^m(M) = \mathcal{D}^m\left(\bigcup_{h \in \mathcal{H}_B} \{S|_x \ : \ L_S(h) = 0\}\right) \tag{1.7}$$

Next, we upper bound the right-hand side of Eq. 1.7 using the following lemma, derived from basic properties of probability.

Lemma 1.2.1 (Union Bound)**.** *For any two sets A, B and a distribution $\mathcal{D}$ we have:*

$$\mathcal{D}(A \cup B) \leq \mathcal{D}(A) + \mathcal{D}(B) \tag{1.8}$$

Applying Lemma 1.2.1 to the right-hand side of Eq. 1.7 yields:

$$\mathcal{D}^m(\{S_x \ : \ L_{(\mathcal{D},f)}(h_S) > \varepsilon\}) \leq \sum_{h \in \mathcal{H}_B} \mathcal{D}^m(\{S|_x \ : \ L_S(h) = 0\}) \tag{1.9}$$

The next step is to bound each summand of the right-hand side of Ineq. 1.9. Let's fix some "bad" hypothesis $h \in \mathcal{H}_B$. The event $L_S(h) = 0$ is equivalent to the event $\forall i, \ h(x_i) = f(x_i)$. Since the examples in the training dataset are sampled i.i.d. we get:

$$\begin{aligned} \mathcal{D}^m(\{S|_x \ : \ L_S(h) = 0\}) &= \mathcal{D}^m(\{S|_x \ : \ \forall i, \ h(x_i) = f(x_i)\}) && (1.10) \\ &= \prod_{i=1}^{m} \mathcal{D}(\{x_i \ : \ h(x_i) = f(x_i)\}) && (1.11) \end{aligned}$$

Lastly, for each individual sampling of an element of the training set we have:

$$\mathcal{D}(\{x_i \ : \ h(x_i) = y_i\}) = 1 - L_{(\mathcal{D},f)}(h) \leq 1 - \varepsilon \tag{1.12}$$

The last inequality follows from the fact that $h \in \mathcal{H}_B$. So we combine Eq. 1.12 with Eq. 1.11 and we use the inequality $1 - \varepsilon \leq e^{-\varepsilon}$ to get $\forall h \in \mathcal{H}_B$:

$$\mathcal{D}^m(\{S|_x \ : \ L_S(h) = 0\}) \leq (1 - \varepsilon)^m \leq e^{-\varepsilon m} \tag{1.13}$$

Combining Eq. 1.13 with Eq. 1.9 we conclude that:

$$\mathcal{D}^m(\{S_x \ : \ L_{(\mathcal{D},f)}(h_S) > \varepsilon\}) \leq |\mathcal{H}_B| e^{-\varepsilon m} \leq |\mathcal{H}| e^{-\varepsilon m} \tag{1.14}$$

Finally, we get the following useful corollary.

Corollary 1.2.1.1. *Let $\mathcal{H}$ be a finite hypothesis class. Let $\delta \in (0, 1)$ and $\varepsilon > 0$ and let m be an integer that satisfies:*

$$m \geq \frac{\log (|\mathcal{H}|/\delta)}{\varepsilon}$$

Then, for any labeling function, f, and for any distribution, $\mathcal{D}$, for which the realizability assumption holds (for some $h \in \mathcal{H}$, $L_{(\mathcal{H},f)}(h) = 0$), with probability of at least $1 - \delta$ over the choice of an i.i.d. sample S of size m, we have that for every ERM hypothesis, h_S, it holds that:

$$L_{(\mathcal{H},f)}(h_S) \leq \varepsilon$$

The Corollary 1.2.1.1 tells us that for a sufficiently large m, the $\text{ERM}_{\mathcal{H}}$ rule over a finite hypotesis class will be *probably* (with confidence $1 - \delta$) *approximately* (up to an error of ε) correct.

Chapter 2

A formal learning model

2.1 Probably Approximated Correct (PAC) learning

Definition 2.1.1 (PAC learnability). A hypothesis class $\mathcal{H}$ is PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: $\forall \varepsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labeling function $f : \mathcal{X} \to \{0,1\}$, if the realizable assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypotesis h such that, with probability of at least $1-\delta$ (over the choice of the examples), $L_{(\mathcal{D},f)}(h) \leq \varepsilon$.

The function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ determines the sample complexity of learning $\mathcal{H}$. So it determines how many samples are required to guarantee a probably approximately correct solution.

Corollary 2.1.0.1. *Every finite class is PAC learnable with sample complexity:*

$$m_{\mathcal{H}}(\varepsilon, \delta) \leq \left\lceil \frac{1}{\varepsilon} \log\left(\frac{|\mathcal{H}|}{\delta}\right) \right\rceil \tag{2.1}$$

Now we want to generalize by:

- Removing the realizability assumption.
  For practical learning tasks, this assumption may be too strong and it isn't necessarily met.

- Extending to learning problems beyond the simple binary classification.
  One may wish to predict real valued numbers or a label picked from a finite set of labels. Therefore our analysis can be extended to many othere scenarios by allowing a variety of loss functions.

2.2 Agnostic PAC learning

We introduce a more realistic model for the data-generating distribution. From now on, let $\mathcal{D}$ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$ (the meaning of the notation hasn't changed). $\mathcal{D}$ is a **joint distribution** over domain points and labels and it can be viewed as the composition of two parts:

- A distribution $\mathcal{D}_x$ over unlabeled domain points (also called **marginal distribution**).

- A **conditional probability** $\mathcal{D}((x,y)|x)$ over labels for each domain point.

2.2.1 Empirical and true error revised

One can measure how likely h is to make an error when labeled points are randomly drawn according to $\mathcal{D}$ previously introduced. So we redefine the true error of a prediction rule h:

$$L_{\mathcal{D}}(h) := \underset{(x,y)\sim\mathcal{D}}{\mathbb{P}}[h(x) \neq y] := \mathcal{D}(\{(x,y) \ : \ h(x) \neq y\}) \tag{2.2}$$

We would like to find a predictor h for which the error in Eq. 2.2 will be minimized. However, the learner doesn't know the data generating $\mathcal{D}$, but he has access only to the training set S. Hence the definition of the empirical risk remains the same as before:

$$L_S(h) := \frac{|\{i \in [m] \ : \ h(x_i) \neq y_i\}|}{m} \tag{2.3}$$

The goal is to find some hypotesis $h \ : \ \mathcal{X} \to \mathcal{Y}$ that (probably approximately) minimizes the true risk, $L_{\mathcal{D}}(h)$.

2.2.2 Bayes optimal predictor

Given any probability distribution $\mathcal{D}$ over $\mathcal{X}\times\{0,1\}$, the best label predicting function from $\mathcal{X}$ to $\{0,1\}$ will be:

$$f_{\mathcal{D}}(x) = \begin{cases} 1 & \text{if } \mathbb{P}[y=1|x] \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Proposition 2.2.1. *For every probability distribution $\mathcal{D}$, the Bayes optimal predictor $f_{\mathcal{D}}$ is optimal, in the sense that no other classifier $g \ : \ \mathcal{X} \to \{0,1\}$ has a lower error. Mathematically:*

$$L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(g) \qquad \forall g \tag{2.5}$$

Clearly, we can't hope that the learning algorithm will find a hypotesis whose error is smaller than the minimal possible error of the Bayes predictor. We require that the learning algorithm will find a predictor whose error is not much larger than the best possible error of a predictor in some given benchmark hypotesis class. So, we generalize the definition of PAC learning.

Definition 2.2.1 (Agnostic PAC learnability)**.** A hypotesis class is PAC learnable if there exist a function $m_{\mathcal{H}} \ : \ (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: for every $\varepsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X}\times\mathcal{Y}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\varepsilon,\delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypotesis h such that, with probability of at least $1-\delta$ (over the choice of the m training examples):

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon \tag{2.6}$$

In the agnostic case, the learner is required to achieve a small error relative to the best error achievable by the hypotesis class and not in absolute terms. In other words, we drop the requirement of finding the best predictor, but we don't want to be too far from it.

2.2.3 Scope of the learning problems

We want to extend our model so that it can be applied to a wide variety of learning tasks, such as:

- **Multiclass Classification**:
  Our classification doesn't have to be binary. The label set can be much more complex, for example it can be a set of real numbers.

- **Regression**:11 In this task, one wishes to find some simple pattern in the data, i.e. a functional relationship between the $\mathcal{X}$ and $\mathcal{Y}$ components of the data. For example, one wishes to find a linear function that describes the correlation between a sample of $\mathcal{X}$ and the corresponding value inside $\mathcal{Y}$. However, for this type of tasks, the measure of success has to be changed. We may evaluate the quality of a hypotesis function, $h : \mathcal{X} \to \mathcal{Y}$, by the *expected square difference* between the true labels and their predicted values, namely:

$$L_{\mathcal{D}}(h) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[h(x) - y]^2 \tag{2.7}$$

To extend the reach of machine learning techniques we must generalize our formalism of the measure of success.

2.2.4 Generalized loss functions and common examples

Given any set $\mathcal{H}$ (our hypotheses or models) and some domain Z, let ℓ be any function from $\mathcal{H} \times Z$ to the set of non-negative real numbers, ℓ, namely $\mathcal{H} \times z \to \mathbb{R}_+$. We call such functions **loss functions**.
We now define the **risk function** to be the expected loss of a classifier $h \in \mathcal{H}$ with respect to a probability distribution $\mathcal{D}$ over Z, namely:

$$L_{\mathcal{D}}(h) := \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\ell(h, z)] \tag{2.8}$$

In other words, we consider the expectation of the loss of h over objects z picked randomly according to $\mathcal{D}$. Similarly, we define the **empirical risk** to be the expected loss over a given sample $S = (z_1, \ldots, z_m) \in Z^m$, namely:

$$L_S(h) := \frac{1}{m}\sum_{i=1}^{m} l(h, z_i) \tag{2.9}$$

Some of the most common examples of losses are given below:

- **0-1 loss**:
  Our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is:

$$\ell_{0-1}(h, (x, y)) := \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases} \tag{2.10}$$

  It is used in binary and multiclass classification.

- **Square loss (L2)**:
  Our loss function is:

$$\ell_{\text{sq}}(h, (x, y)) := [h(x) - y]^2 \tag{2.11}$$

  This loss is used in regression problems to penalize few large errors.

- **Absolute value loss (L1)**:
  Just like before, but with modulus and not the square:

$$\ell_{\text{abs}}(h,(x,y)) := |h(x) - y| \tag{2.12}$$

  It is used in regression tasks to penalize many small errors.

To summarize, we formally define agnostic PAC learnability for general loss functions.

Definition 2.2.2 (Agnostic PAC learnability for general loss functions)**.** A hypotesis class $\mathcal{H}$ is PAC learnable with respect to a set Z and a loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: for every $\varepsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over Z, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\varepsilon,\delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns $h \in \mathcal{H}$ such that, with probability of at least $1-\delta$ (over the choice of the m training examples):

$$l_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon \tag{2.13}$$

where $L_{\mathcal{D}}(h) = \mathbb{E}_{z\sim\mathcal{D}}[\ell(h,z)]$

Remark 1. We can consider $\ell(h, \bullet) : Z \to \mathbb{R}_+$ a random variable. Hence $L_{\mathcal{D}}(h)$ is the expected value of this random variable.

Chapter 3

Learning via uniform convergence

In this Chapter we will develop a more general tool, namely the *uniform convergence*, and apply it to show that any finite class is learnable in the agnostic PAC model with general loss functions, as long as the range loss function is bounded.

3.1 Uniform convergence is sufficient for learnability

Given a hypothesis class $\mathcal{H}$, the ERM learning algorithm:

- Receives a training sample S.
- The learner evaluates the risk of each $h \in \mathcal{H}$ on the given sample S.
- The learner outputs a member h^* of $\mathcal{H}$ that minimizes this empirical risk.
- The hope is that a h that minimizes the empirical risk with respect to S is a risk minimizer with respect to the true data probability distribution as well. For that, it sufficies to ensure that the empirical risks of all members of $\mathcal{H}$ are good approximations of their true risk.

In other words, we need that uniformly over all hypoteses in the hypotesis class, the empirical risk will be close to the true risk. This is formalized in the following definition. Then, a lemma introduces us to a first result.

Definition 3.1.1 (ε-representative sample)**.** A training set S is called ε-representative (with respect to domain Z, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$) if:

$$\forall h \in \mathcal{H},\ |L_S(h) - L_{\mathcal{D}}(h)| \leq \varepsilon \tag{3.1}$$

Lemma 3.1.1. *Assume that a training set S is $\frac{\varepsilon}{2}$-representative (with respect to domain Z, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$). Then any output of ERM$_{\mathcal{H}}(S)$, namely, any $h_S \in \operatorname{argmin}_{h\in\mathcal{H}} L_S(h)$, satisfies:*

$$L_{\mathcal{D}}(h_S) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon \tag{3.2}$$

Proof. For every $h \in \mathcal{H}$:

$$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\varepsilon}{2} \leq L_S(h) + \frac{\varepsilon}{2} \leq L_{\mathcal{D}}(h) + \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = L_{\mathcal{D}}(h) + \varepsilon$$

□

The consequence of this lemma is that, if with probability $1 - \delta$, a random training set S is ε-representative, then the ERM rule is an agnostic PAC learner. The uniform convergence condition formalizes this requirement.

Definition 3.1.2 (Uniform convergence)**.** A hypothesis calss $\mathcal{H}$ has the *uniform convergence property* (with respect to a domain Z and a loss function ℓ) if there exists a function $m_{\mathcal{H}}^{\mathrm{UC}} : (0,1)^2 \to \mathbb{N}$ such that for every $\varepsilon, \delta \in (0,1)$ and for every probability distribution $\mathcal{D}$ over Z, if S is a sample of $m \geq m_{\mathcal{H}}^{\mathrm{UC}}(\varepsilon, \delta)$ examples drawn i.i.d. according to $\mathcal{D}$, then, with probability of at least $1 - \delta$, S is ε-representative.

Corollary 3.1.1.1. *If a class $\mathcal{H}$ has the uniform convergence property with a function $m_{\mathcal{H}}^{UC}$, then the class is agnostically PAC learnable with the sample complexity $m_{\mathcal{H}}(\varepsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\frac{\varepsilon}{2}, \delta)$. Furthermore, in that case, the ERM$_{\mathcal{H}}$ paradigm is a successful agnostic PAC learner for $\mathcal{H}$.*

3.2 Finite classes are agnostic PAC learnable

Let's start with a lemma that will be useful to give a proof of a subsequent theorem.

Lemma 3.2.1 (Hoeffding's inequality)**.** *Let $\theta_1, \ldots, \theta_m$ be a sequence of i.i.d. random variables and assume that $\forall i$, $\mathbb{E}[\theta_i] = \mu$ and $\mathbb{P}[a \leq \theta_i \leq b] = 1$. Then for any $\varepsilon > 0$:*

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \varepsilon\right] \leq 2e^{-\frac{2m\varepsilon^2}{(b-a)^2}} \tag{3.3}$$

Theorem 3.2.2. *Let $\mathcal{H}$ be a finite hypothesis class, let Z be a domain, and let $\ell : \mathcal{H} \times Z \to [0,1]$ be a loss function. Then:*

- *$\mathcal{H}$ enjoys the uniform convergence property with sample complexity:*

$$m_{\mathcal{H}}^{UC}(\varepsilon, \delta) \leq \left\lceil \frac{\log\left(\frac{2|\mathcal{H}|}{\delta}\right)}{2\varepsilon^2} \right\rceil \tag{3.4}$$

- *$\mathcal{H}$ is agnostically PAC learnable using the ERM algorithm with sample complexity:*

$$m_{\mathcal{H}}(\varepsilon, \delta) \leq m_{\mathcal{H}}^{UC}\left(\frac{\varepsilon}{2}, \delta\right) \leq \left\lceil \frac{2\log\left(\frac{2|\mathcal{H}|}{\delta}\right)}{\varepsilon^2} \right\rceil \tag{3.5}$$

Proof. Let's give the proof of the previous theorem by steps.

- ▷ $\mathcal{D}^m(\{S : \forall h \in \mathcal{H},\ |L_S(h) - L_{\mathcal{D}}(h)| \leq \varepsilon\}) \geq 1 - \delta$
- ▷ $\mathbb{P}_{\text{bad}} = \mathcal{D}^m(\{s : \exists h \in \mathcal{H},\ |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) \leq \delta$
- ▷ $\{S : \exists h \in \mathcal{H},\ |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\} = \bigcup_{h \in \mathcal{H}} \{S : |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}$
- ▷ Union bound:
  $\mathcal{D}^m(\{S : \exists h \in \mathcal{H},\ |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) \leq \sum_{h \in \mathcal{H}} \mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\})$
- ▷ We now demonstrate that for any fixed h, the difference $|L_S(h) - L_{\mathcal{D}}(h)|$ is small enough.

▷ Recall that:

$$L_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\ell(h,z)]$$
$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}\ell(h,z_i)$$

By the linearity of expectation, it follows that $L_{\mathcal{D}}(h)$ is also the expected value of $L_S(h)$. Hence, the quantity $|L_S(h) - L_{\mathcal{D}}(h)|$ is the deviation of the random variable $L_S(h)$ from its expectation.
We need to show that the measure of $L_S(h)$ is concentrated around its expectation value.

▷ Law of large numbers: if m is large, the average converges to the expectation.

▷ From Lemma 3.2.1 it follows:

$$\mathcal{D}^m(\{S \,:\, |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) = \mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \varepsilon\right] \leq 2e^{-2m\varepsilon^2}$$

▷ $\sum_{h\in\mathcal{H}} \mathcal{D}^m(\{S \,:\, |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) \leq |\mathcal{H}|2e^{-2m\varepsilon^2} = \sum_{h\in\mathcal{H}} 2e^{-2m\varepsilon^2}$

▷ We impose:

$$|\mathcal{H}|2e^{-2m\varepsilon^2} \overset{!}{\leq} \delta \Longrightarrow m \geq \log\left(\frac{2|\mathcal{H}|}{\delta}\right)\frac{1}{2\varepsilon^2}$$
$$\Longrightarrow \sum_{h\in\mathcal{H}} \mathcal{D}^m(\{S \,:\, |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) \leq \delta$$

□

Remark 2 (The discretization trick). There is a simple trick that allows us to get a good estimate of the practical sample complexity of infinite hypothesis class:

- Consider a hypothesis class determined by d real number parameters.
- In principle, we have a hypothesis class of infinite size.
- In practice, real numbers are represented with 64 bits double precision variables.
- For d parameters: $|\mathcal{H}| = 2^{64d}$, so $|\mathcal{H}|$ is larger but finite.
- We have:

$$m_{\mathcal{H}}(\varepsilon,\delta) \leq m_{\mathcal{H}}^{\mathrm{UC}}\left(\frac{\varepsilon}{2}\right) \leq \frac{2\log\left(2\frac{2^{64d}}{\delta}\right)}{\varepsilon^2}$$

- So the bound depends on the chosen number representation.

Chapter 4

The bias complexity trade-off

Let A be the learning algorithm, S the training set, $\mathcal{H}$ the hypothesis class and $\hat{h} \in \mathcal{H}$ such that $L_{\mathcal{D}}(\hat{h})$ is small. What we want to know is if there is a universal learner, i.e. an algorithm A that predicts the best $\hat{h}$ for any distribution $\mathcal{D}$.

4.1 No-Free-Lunch theorem

Theorem 4.1.1 (No-Free-Lunch)**.** *Let A be any learning algorithm for the task of binary classification with respect to the 0-1 loss over a domain $\mathcal{X}$. Let m be any number smaller than $\frac{|\mathcal{X}|}{2}$, representing a training set size. Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:*

- *There exists a function $f : \mathcal{X} \to \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$.*
- *With probability of at least $\frac{1}{7}$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq \frac{1}{8}$.*

The meaning of this theorem is clear: a universal learner can't exist. In fact, no learner can succeed on all learning tasks. FIn other words, for every learner, there exists a task on which it fails, even though that task can be successfully learned by another learner.

Corollary 4.1.1.1. *Let $\mathcal{X}$ be an infinite domain set and let $\mathcal{H}$ be the set of all functions from $\mathcal{X}$ to $\{0,1\}$. Then, $\mathcal{H}$ is not PAC learnable.*

Firstly, we give the idea of the proof. Our training set is smaller than half of the domain, so we have no information on what happens on the other half. There exists some target function f that works on the other half in a way that contradicts our estimated labels.

Remark 3. A class $\mathcal{H}$ of all possible functions from $\mathcal{X}$ to $\{0,1\}$ (i.e. assuming no prior knowledge) is not a good idea since ii isn't PAC learnable.

Proof. Let's demonstate the previous corollary. We proceed by contradiction:

1. Assume PAC learnability:
   - We choose $\varepsilon < \frac{1}{8}$, $\delta < \frac{1}{7}$ (Note that $\mathcal{H}$ includes all functions).
   - By definition of PAC: there exists an algorithm A such that for very distribution $\mathcal{D}$, if realizable, when running the algorithm on m i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypothesis h such that, with probability $\geq 1 - \delta$, $L_{\mathcal{D}}(A(S)) \leq \varepsilon$.
2. Apply No-Free-Lunch theorem to A:

  - $|\mathcal{X}| > 2m$: for any ML algorithm (including A), there exists a distribution $\mathcal{D}$ for which with probability $\geq \frac{1}{7} > \delta$, $L_{\mathcal{D}}(A_S) \geq \frac{1}{8} > \varepsilon$.

3. Contradiction!

□

Remark 4. We want to choose a good hypotesis set:

- We need to use our prior knowledge about $\mathcal{D}$ to pick a good hypothesis set.
- We would like $\mathcal{H}$ to be large, so that it may contain a function h with small $L_S(h)$ and hopefully a small $L_{\mathcal{D}}(h)$.
- No-Free-Lunch theorem tells us that $\mathcal{H}$ can't be too large! In fact:
  - ▷ $\mathcal{H}$ too small $\implies$ large L_S and $L_{\mathcal{D}} \sim L_S$.
  - ▷ $\mathcal{H}$ too large $\implies$ small L_S and $L_{\mathcal{D}} \gg L_S$ (risk of overfitting).

4.2 Error decomposition

To have a more clear vision, we decompose the error of an ERM$_{\mathcal{H}}$ predictor into two components as follows. Let h_S be an ERM$_{\mathcal{H}}$ hypotesis. Then, we can write:

$$L_{\mathcal{D}}(h_S) = \varepsilon_{\text{app}} + \varepsilon_{\text{est}} \tag{4.1}$$

where:

$$\varepsilon_{\text{app}} = \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \tag{4.2a}$$

$$\varepsilon_{\text{est}} = L_{\mathcal{D}}(h_S) - \varepsilon_{\text{app}} \tag{4.2b}$$

- **Approximation Error**:
  It is the minimum risk achievable by a predictor in the hypotesis class. This term measures how much risk we have because we restrict ourselves to a specific class, namely how much **inductive bias** we have. Other informations on ε_{app}:
  - ▷ It depends on the choice of $\mathcal{H}$.
  - ▷ It doesn't depend on the sample size m.
  - ▷ Under the realizability assumption: $\varepsilon_{\text{app}} = 0$
  - ▷ The larger $\mathcal{H}$ is, the smaller $\varepsilon_{\text{app}} = 0$ is.
- **Estimation Error**:
  It is the difference between the approximation error and the error achieved by the ERM predictor. It results because the empirical risk (i.e. the training error) is only an estimate of the true risk and so the predictor minimizing the empirical risk is only an estimate of the predictor minimizing the true risk. Other informations on ε_{est}:
  - ▷ It depends on the training set size m and on the size (complexity) of the hypothesis class.
  - ▷ ε_{est} increases with $|\mathcal{H}|$ and decreases with m.

Since our goal is to minimize the total risk, we face a tradeoff, called the **bias-complexity tradeoff**:

- Choosing $\mathcal{H}$ large $\implies$ decreases ε_{app}, increases ε_{est}. This leads to overfitting.
- Choosing $\mathcal{H}$ small $\implies$ decreases ε_{est}, increases ε_{app}. This leads to underfitting.

4.3 Train, test and validation sets

In most practical applications, the available set of examples is split into more sets with different purposes. In order:

- We estimate $L_{\mathcal{D}}(h)$ (with h selected with ERM).
- We extract a **test set** from the original one, namely a new set of samples not used for picking h. It is different from the training set and it leads to a more reliable estimation of $L_{\mathcal{D}}(h)$. We must not look to the test set until we have picked our final hypothesis. In practice, one set of sample is splitted into training set and test set.
- Sometimes the training set is divided into training set and **validation set**. The ladder is used for selecting the hyperparameters of the algorithm.

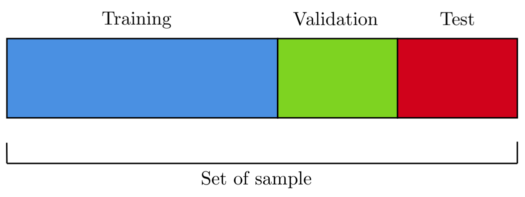

Figure 4.1: Scheme of the division of the set of examples S.

Hence, the training routine is:

- Train model on training set.
- Evaluate model on validation set.
- Tweak model according to the results on validation set. Restart from the first point.
- When the learning procedure has finished due to a particular ending condition, pick the model that performs better on validation set.
- Confirm the result on test set.

Chapter 5

Linear predictors

In this Chapter we will study the family of linear predictors, one of the most useful families of hypothesis classes. Many learning algorithms that are being widely used in practice rely on linear predictors, for their ability to learn them efficiently in many cases and because they are intuitive and easy to interpret.
In particular, we focus on learning linear predictors using the ERM approach. The hypothesis classes can include:

- Halfspaces (classification).
- Linear regression (regression).
- Logistic regression (classification moduled as a regression problem).

The algorithms can include:

- Linear programming (for halfspaces).
- Perceptron (for halfspaces).
- Least squares (for regression).

First, we define the class of **affine functions** as:

$$L_d = \{h_{\mathbf{w},b} \ : \ \mathbf{w} \in \mathbb{R}^d, \ b \in \mathbb{R}\} \tag{5.1}$$

where:

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b \tag{5.2}$$

It is convenient to use the following notation for L_d:

$$L_d = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b \ : \ \mathbf{w} \in \mathbb{R}^d, \ b \in \mathbb{R}\} \tag{5.3}$$

which reads as follows: L_d is a set of functions, where each function is parametrized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and each such function takes as input a vector $\mathbf{x}$ and returns as output the scalar $\langle \mathbf{w}, \mathbf{x} \rangle + b$. Therefore the dimension of the parameter space increases from d to $d+1$.
The different hypothesis classes of linear predictors are composition of function $\varphi : \mathbb{R} \to \mathcal{Y}$ on L_d. For example:

- For binary classification: $\mathcal{Y} = \{-1, 1\} \to \varphi(z) = \text{sign}(z)$
- For regression: $\mathcal{Y} = \mathbb{R} \to \varphi(z) = z$

$$\begin{aligned} \mathbf{w}' &= (b, w_1, w_2, \dots, w_d) \in \mathbb{R}^{d+1} \\ \mathbf{x}' &= (1, x_1, x_2, \dots, x_d) \in \mathbb{R}^{d+1} \end{aligned} \tag{5.4}$$

Therefore $h_{\mathbf{w},b}(\mathbf{x})$ becomes a linear function:

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle \tag{5.5}$$

5.1 Halfspaces

They are exploited for binary classification problems, namely $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$. The class of halfspaces is defined as follows:

$$HS_d = \text{sign} \circ L_d = \{\mathbf{x} \mapsto \text{sign}(h_{\mathbf{w},b}(\mathbf{x})) \ : \ h_{\mathbf{w},b} \in L_d\} \tag{5.6}$$

In other words, each halfspace hypothesis in HS_d is parametrized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ and upon receiving a vector $\mathbf{x}$ the hypotesis returns the label sign $(\langle \mathbf{w}, \mathbf{x} \rangle + b)$. The realizability can be obtained only if the space is linearly separable.
To illustrate this hypothesis class geometrically, let's consider the case $d = 2$. Each hypothesis forms a hyperplane that is perpendicular to the vector $\mathbf{w}$ and intersects the vertical axis at the point $(0, -\frac{b}{w_2})$. The istances that are "above" the hyperplane, that is, share an acute angle with $\mathbf{w}$, are labeled positively. Istances that are "below" the hyperplane, that is, share an obtuse angle with $\mathbf{w}$, are labeled negatively.

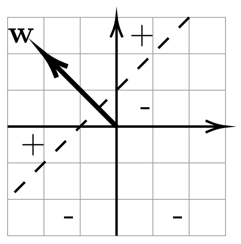

Figure 5.1: Example of halfspace with $d = 2$.

From Fig. 5.1 it is clear the condition needed for the realizability. Implementing the ERM rule in non-separable case (i.e. in the agnostic case) is known to be computationally hard, but there are several approaches to tackle problems with non-separable data.

5.1.1 Linear programming for the class of halfspaces

Linear programs (LP) are problems that can be expressed as maximizing a linear function subject to linear inequalities (constraints). That is:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \langle \mathbf{u}, \mathbf{w} \rangle \qquad \text{subject to } A\mathbf{w} \geq \mathbf{v} \tag{5.7}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the vector of variables we wish to determine, A is an $m \times d$ matrix, and $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{u} \in \mathbb{R}^d$ are vectors. Linear programming can be solved efficiently[1], and furthermore, there are publicly implementations of LP solvers.
The ERM problem for halfspaces in the realizable case can be expressed as a linear program. To show this, we assume for simplicity the homogeneous case. Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be a training set of size m. We can say:

[1]Namely, in time polynomial in m, d, and in the representation size of real numbers.

- For the realizable assumption, an ERM predictor should have zero errors on the training set. That is, we are looking for some vector $\mathbf{w} \in \mathbb{R}^d$ for which:

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0 \qquad \forall i = 1, \dots, m \tag{5.8}$$

- There exists also a vector $\mathbf{w}$ that satisfy

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \qquad \forall i = 1, \dots, m \tag{5.9}$$

  Proof. Let $\mathbf{w}^*$ be a vector that satisfies $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0 \; \forall i$. We define $\gamma = \min_i (y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle)$ and let $\bar{\mathbf{w}} = \frac{\mathbf{w}^*}{\gamma}$. Therefore, $\forall i$ we have:

$$y_i \langle \bar{\mathbf{w}}, \mathbf{x}_i \rangle = \frac{1}{\gamma} y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1$$

  We have thus shown that there exists a vector that satisfies Eq. 5.9. □

- We want to express the ERM model with linear programming. Let A be a $m \times d$ matrix such that $A_{ij} = y_i x_{ij}$, with $x_i j$ the j^{th} element of the vector $\mathbf{x}_i$. Let $\mathbf{v}$ be the vector $(1, \dots, 1) \in \mathbb{R}^m$. Then, Eq. 5.9 can be rewritten as:

$$A\mathbf{w} \geq \mathbf{v}$$

  The LP form requires a maximization objective, yet all the $\mathbf{w}$ that satisfy the constraints are equal candidates as output hypotheses. Thus, we set a "dummy" objective $\mathbf{u} = (0, \dots, 0) \in \mathbb{R}^d$.

5.1.2 Perceptron for halfspaces

It is a different implementation of the ERM rule by Rosenblatt (1958). The perceptron is an iterative algorithm that constructs a sequence of vectors $\mathbf{w}^{(1)}$, $\mathbf{w}^{(2)}$, ... Initially, $\mathbf{w}^{(1)}$ is set to be the all-zeros vector. At iteration time t, the perceptron finds an example i that is mislabeled by $\mathbf{w}^{(t)}$, namely, an example for which $\text{sign}(\langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle) \neq y_i$. Then, the perceptron updates $\mathbf{w}^{(t)}$ by adding to it the istance $\mathbf{x}_i$ scaled by the label y_i. That is:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$$

Our goal is to have $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0 \; \forall i$ and note that:

$$y_i \left\langle \mathbf{w}^{(t+1)}, \mathbf{x}_i \right\rangle = y_i \left\langle \mathbf{w}^{(t)} + y_i \mathbf{x}_i, \mathbf{x}_i \right\rangle = y_i \left\langle \mathbf{w}^{(t)}, \mathbf{x}_i \right\rangle + \|\mathbf{x}_i\|^2$$

Hence, the update of the perceptron guides the solution to be "more correct" on the i^{th} example.

Algorithm 1: Batch Perceptron.

```
  input     : A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$
  output    : $\mathbf{w}^{(t)}$
  initialize: $\mathbf{w}^{(1)} = (0, \dots, 0)$
1 for $t = 1, 2, \dots$ do
2     if $\exists i$ s.t. $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \leq 0$ then
3         $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_i \mathbf{x}_i$
4     else
5         return $\mathbf{w}^{(t)}$
```

The following theorem guarantees that in the realizable case, the algorithm stops with all sample points correctly classified.

Theorem 5.1.1. *Assume that $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ is separable, let:*

$$\begin{aligned}
&B = \min \{\|\mathbf{w}\| \ : \ \forall i \in [m], \ y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1\} \\
&R = \max{}_i \|\mathbf{x}_i\|
\end{aligned}$$

Then, the perceptron algorithm stops after at most $(RB)^2$ iterations, and when it stops it holds that $\forall i \in [m], \ y_i \left\langle \mathbf{w}^{(t)}, \mathbf{x}_i \right\rangle > 0$.

Proof. We give the proof in steps:

▷ Let's define a vector $\mathbf{w}^*$ achieving the minimum in B, and T the number of iterations before stopping.

▷ Consider:

$$\frac{\left\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \right\rangle}{\|\mathbf{w}^*\| \left\|\mathbf{w}^{(t+1)}\right\|} \leq 1 \tag{5.10}$$

▷ We need to demonstate:

$$\frac{\sqrt{T}}{RB} \leq \frac{\left\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \right\rangle}{\|\mathbf{w}^*\| \left\|\mathbf{w}^{(T+1)}\right\|} \leq 1 \Longrightarrow T < (RB)^2 \tag{5.11}$$

▷ We divide this step into two parts:

a) Numerator:
We want to demonstate that: $\left\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \right\rangle \geq T$.

- First iteration: $\mathbf{w}^{(1)} = (0, \ldots, 0) \Longrightarrow \left\langle \mathbf{w}^*, \mathbf{w}^{(1)} \right\rangle = 0$
- At each step: $\left\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \right\rangle - \left\langle \mathbf{w}^*, \mathbf{w}^{(t)} \right\rangle \geq 1$
- After T iterations: $\left\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \right\rangle \geq T$
In fact:

$$\begin{aligned}
\left\langle \mathbf{w}^*, \mathbf{w}^{(T+1)} \right\rangle &= \sum_{t=1}^{T} \left(\left\langle \mathbf{w}^*, \mathbf{w}^{(t+1)} \right\rangle - \left\langle \mathbf{w}^*, \mathbf{w}^{(t)} \right\rangle \right) \\
&= \sum_{t=1}^{T} \langle \mathbf{w}^*, y_i \mathbf{x}_i \rangle \geq T
\end{aligned} \tag{5.12}$$

b) Denominator:
We want to demonstate that: $\|\mathbf{w}^*\| \left\|\mathbf{w}^{(T+1)}\right\| \leq \sqrt{T} RB$.

- We have:

$$\begin{aligned}
\left\|\mathbf{w}^{t+1}\right\| = \left\|\mathbf{w}^{(t)} + y_i \mathbf{x}_i\right\|^2 &= \left\|\mathbf{w}^{(t)}\right\|^2 + 2y_i \left\langle \mathbf{w}^{(t)}, \mathbf{x}_i \right\rangle + y_i^2 \|\mathbf{x}_i\|^2 \\
&\leq \left\|\mathbf{w}^{(t)}\right\|^2 + R^2
\end{aligned} \tag{5.13}$$

Therefore:

$$\begin{aligned}
\left\|\mathbf{w}^{(T+1)}\right\| &= \sum_{t=1}^{T}\left(\left\|\mathbf{w}^{(t+1)}\right\|^2 - \left\|\mathbf{w}^{(t)}\right\|^2\right) \\
&= \sum_{t=1}^{T}\left(\left\|\mathbf{w}^{(t)} + y_i\mathbf{x}_i\right\|^2 - \left\|\mathbf{w}^{(t)}\right\|^2\right) \\
&= \sum_{t=1}^{T}\left(2y_i\left\langle\mathbf{w}^{(t)}, \mathbf{x}_i\right\rangle + \|\mathbf{x}_i\|^2\right) \leq TR^2 \qquad (5.14)
\end{aligned}$$

▷ Combining the two previous results we obtain:

$$\frac{\left\langle\mathbf{w}^*, \mathbf{w}^{(T+1)}\right\rangle}{\|\mathbf{w}^*\|\left\|\mathbf{w}^{(T+1)}\right\|} \geq \frac{T}{BR\sqrt{T}} = \frac{\sqrt{T}}{BR} \tag{5.15}$$

□

Remark 5. Convergence is guaranteed and it depends on B, which can be exponential in d.

5.2 Linear regression

Chapter 6

The Vapnick-Chervonenkis dimension

Bibliography