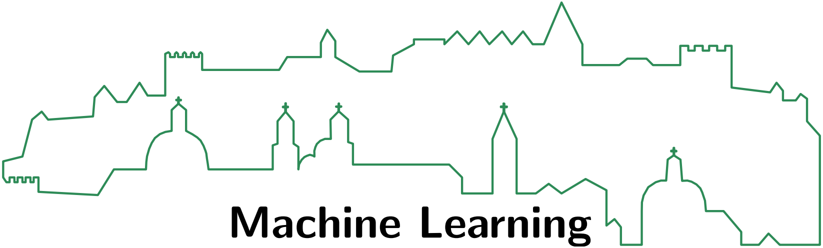


Machine Learning

Roland Kwitt[1]

[1] Department of Computer Science
University of Salzburg

Course: **911.235**

Overview

Quick personal introduction

2005–2007	MSc (Univ. of Salzburg)
2007–2010	PhD (Univ. of Salzburg)
2011–2013	Kitware Inc., NC, USA
2013–2017	Ass.-Prof. (Univ. Salzburg)
2017–2020	Assoc.-Prof. (Univ. Salzburg)
2020–now	Univ.-Prof. (Univ. Salzburg)

How can you reach me?

E-Mail (best): `Roland.Kwitt@sbg.ac.at`

Overview

Some notes on the slide coloring

Important notes, remarks, etc.

Exemplary title

Mostly used for introducing new concepts/terminology.

Theorem 1.1: Theorem title

Also used for lemmata, corollaries, etc.

Questions?

Important stuff

Also important stuff

Comments

Overview

Classroom discussions

We use **SLACK** for course related discussions/questions. Our slack channel is `#machinelearning` at

`https://visel.slack.com/home`

> Please **do not** write e-mails to my personal e-mail account regarding *course related* questions (since they might be interesting for all)!

For those of you who do not want to enroll (for personal or privacy reasons), all classroom material can also be found on my website:

`https://github.com/rkwitt/teaching/tree/master/SS21/ML`

Overview

Grading

Vorlesung (VO): Final exam at the end of the summer term!

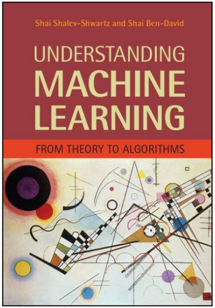
Proseminar (PS): You can work in groups of size ≤ 3!

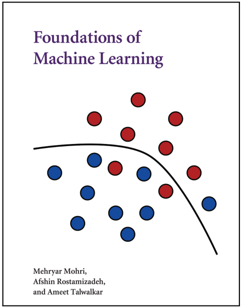
- Exercises are marked with points (more points $\equiv$ harder to solve)
- You have to have > 50 points for a positive grade

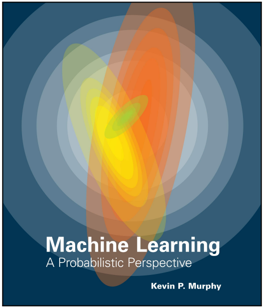
Overview

Textbooks

Most of the material of this lecture is compiled from the following books [4, 2, 3]:


Shai Shalev-Shwartz and Shai Ben-David
UNDERSTANDING MACHINE LEARNING
FROM THEORY TO ALGORITHMS





What is machine learning?

What is machine learning?

Mohri *et al*. [2] define machine learning as:

> *"Computational methods that use experience to improve performance or to make accurate predictions."*

In other words, our goal is to teach a machine to use

$$\textbf{Experience} \xrightarrow{\text{to generate}} \textbf{Expertise} \text{ (or knowledge)}$$

How do we measure the quality of a learner?:

- Space/time complexity (as in many other fields of CS)
- Sample complexity (special to machine learning)

What is machine learning?

Here's another definition from Mitchell [1]:

Machine learning

A computer program is said to learn from

- experience E

with respect to (w.r.t.) some

- class of tasks T

and

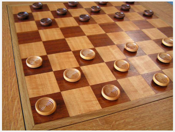
- performance measure P

if its performance, measured by P, improves with experience E.

What is machine learning?

A simple example

Learning objective



Learn to play checkers.

- **T**ask T: playing checkers
- **P**erformance measure P: percentage of games won
- **E**xperience E: playing practice against itself

What is machine learning?

A motivating example

Learning objective

Program a machine that learns to filter SPAM e-mails.

(Naive) “learning-by-memorization”:

1. Memorize all e-mails previously labeled as “spam” by a human user
2. Once a new e-mail arrives, search for it in this memorized set
3. If found, mark as “spam”, otherwise as mark as “no-spam”

However, by this strategy, we *cannot* label previously unseen e-mails!

What is machine learning?

Inductive inference/reasoning and inductive bias

Looking at the previous example, we desire[1]:

Inductive reasoning

A successful learner should be able to generalize. This is also known as *inductive reasoning* (or inductive inference).

In practice, the instance space is often so large that any learning algorithm only sees a fraction of it during training.

Inductive bias

We need to bias the learning process via the incorporation of prior knowledge. This is known as *inductive bias*.

[1] We will make these concepts more crisp, when we take a closer look at the "No-Free-Lunch" theorem(s) later in this lecture.

What is machine learning?

Learning scenarios – (Semi-)supervised/unsupervised

Supervised learning

Experience (in the form of training instances) contains significant information (*e.g.*, labels in “spam” *vs.* “no-spam” classification).

Unsupervised learning

No supervisory signal! We aim for a summary representation (*e.g.*, a clustering) of the data (useful for downstream tasks).

Semi-supervised learning

Training data contains both labeled *and* unlabeled instances (*e.g.*, in situations where unlabeled data is easy to acquire).

In all three scenarios, predictions are made for unseen (test) instances.

What is machine learning?

Learning scenarios – Active/passive

Active learning

Active learners interact with the environment (at training time), *e.g.*, by querying an oracle (human) to request labels for specific instances.

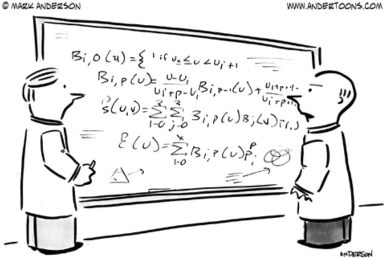
Passive learning

Passive learners only observe information provided to them.

Example (spam *vs.* no-spam):

Passive	Simply use all the provided labeled instances
Active	Ask the user to label specific e-mails

Background material



"What the hell is *that* supposed to mean?!"

Probability review

σ-algebra

Let Ω be some (non-empty) set. A family of sets[2] $\mathcal{F} \subset \mathcal{P}(\Omega)$ is called a **σ-algebra** over Ω, if the following conditions hold:

1. $\Omega \in \mathcal{F}$
2. From $A \in \mathcal{F}$, it follows that $A^c = \Omega \setminus A \in \mathcal{F}$

   (i.e., closed under complement operation)
3. From $A_i \in \mathcal{F}$, $i \in \mathbb{N}$, it follows that

   (i.e., closed under countable union)

$$\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$$

[2] Any subset $\mathcal{F} \subset \mathcal{P}(\Omega)$ of the power set $\mathcal{P}$ of Ω is called a *family of sets*.

Probability review

Borel σ-algebra & Generating sets

Important example: The σ-algebra of Borel sets over $\mathbb{R}^n$.

Let Ω be a topological space and $\mathcal{O}$ the system of *open subsets* of Ω. Then

$$\mathcal{B}(\Omega) = \sigma(\mathcal{O})$$

is called the Borel σ-algebra on Ω. Elements $A \in \mathcal{B}(\Omega)$ are called Borel sets. For $\Omega = \mathbb{R}^n$, we write $\mathcal{B}^n = \mathcal{B}(\mathbb{R}^n)$.

All of the following set systems are *generators* of $\mathcal{B}^n$

- $\mathcal{O}^n = \{U \subset \mathbb{R}^n : U \texttt{ open}\}$,
- $\mathcal{C}^n = \{A \subset \mathbb{R}^n : A \texttt{ closed}\}$,
- $\mathcal{I}^n = \{]a, b] : a, b \in \mathbb{R}^n, a \leq b\}$,
- $\mathcal{I}^n_\infty = \{]-\infty, c] : c \in \mathbb{R}^n\}$,

with $]-\infty, c] =]-\infty, c_1] \times \cdots \times]-\infty, c_n] \subset \mathbb{R}^n$ for $c \in \mathbb{R}^n$.

Probability review

Measurable space & Measurable function

If $\mathcal{F}$ is a σ-algebra over Ω, we call the tuple

$$(\Omega, \mathcal{F})$$

a **measurable space** (e.g., $(\mathbb{R}, \mathcal{B})$).

Given two measure spaces, $(\Omega, \mathcal{F}_1)$ and $(\Omega, \mathcal{F}_2)$, we call a mapping

$$f : \Omega_1 \rightarrow \Omega_2$$

measurable if

$$\forall B \in \mathcal{F}_2 : f^{-1}(B) \in \mathcal{F}_1 \ \ .$$

For measurability, it suffices to check measurability w.r.t. a generating set system, E, of $\mathcal{F}_2 = \sigma(E)$. For example, given $(\Omega, \mathcal{F})$ and $f : \Omega \rightarrow \mathbb{R}$, then f is measurable if, for all $c \in \mathbb{R}$

$$f^{-1}(]-\infty, c]) = \{w \in \Omega : f(w) \leq c\} \in \mathcal{F} \ \ .$$

Probability review

Measurable function – Example

Example: Indicator function of a set $A \subset \Omega$, i.e.,

$$I_A : \Omega \to \mathbb{R}, \quad w \mapsto I_A(w) = \begin{cases} 1, & \text{if } w \in A \\ 0, & \text{else} \end{cases} .$$

All pre-images of I_A are $\emptyset, A, A^c, \Omega$. These are in $\mathcal{F}$ if $A \in \mathcal{F}$.

Intuitively, we refer to A as a **measurable set** if $A \in \mathcal{F}$.

Probability review

Measures

Say $(\Omega, \mathcal{F})$ is a measurable space. A function

$$\mu : \mathcal{F} \to \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$$

is called a **measure** on $\mathcal{F}$ if the following conditions hold:

1. $\mu(\emptyset) = 0$
2. $\mu(A) \geq 0$ for all $A \in \mathcal{F}$
3. For any sequence of disjoint sets from $\mathcal{F}$, we have

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$$

Example: Counting measure: $\mu : \mathcal{F} \to \mathbb{R}$, $A \mapsto |A|$ if A is finite and ∞ otherwise.

Probability review

Measure space & Properties of measures

If $(\Omega, \mathcal{F})$ is a measurable space and $\mu : \mathcal{F} \to \overline{\mathbb{R}}$ a measure, then we call

$$(\Omega, \mathcal{F}, \mu)$$

a **measure space**.

Elementary properties of measures:

- $A, B \in \mathcal{F}$ disjoint: $\mu(A \cup B) = \mu(A) + \mu(B)$
- $A \subset B, \mu(A) < \infty$: $\mu(B \setminus A) = \mu(B) - \mu(A)$
- $A_1, \dots, A_n \in \mathcal{F}$:
$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} \mu(A_i)$$

Probability review

Pushforward measure

Based on a measure, we can easily construct a new measure by means of a measurable function.

Given a measure space $(\Omega_1, \mathcal{F}_1, \mu)$, a measurable space $(\Omega_2, \mathcal{F}_2)$ and a *measurable function*

$$f : \Omega_1 \to \Omega_2$$

we call

$$\mu_f : \mathcal{F}_2 \to [0, \infty], \quad B \mapsto \mu_f(B) = \mu(f^{-1}(B))$$

the **pushforward measure** of μ under f.

Probability review

Probability space

A **probability space** is measure space $(\Omega, \mathcal{F}, \mathcal{D})$ with $\mathcal{D}(\Omega) = 1$.

Interpretation:

- Ω is the sample space
- the σ-algebra $\mathcal{F}$ is the set of possible events
- $\mathcal{D} : \mathcal{F} \to [0, 1]$ assigns a probability to each event

Probability review

Random variables

Given a probability space $(\Omega_1, \mathcal{F}_1, \mathcal{D})$ and a measurable space $(\Omega_2, \mathcal{F}_2)$, we call the measurable function

$$X : \Omega_1 \to \Omega_2$$

a **random variable**.

In case $\Omega_2 = \mathbb{R}^n$, X is a n-dimensional real random variable.

The pushforward measure $\mathbb{P}$ of $\mathcal{D}$ under X is again a probability measure, since

$$\mathbb{P}(\Omega_2) = \mathcal{D}(X^{-1}(\Omega_2)) = \mathcal{D}(\Omega_1) = 1$$

We call the pushforward $\mathbb{P}$ the **distribution** of X.

Probability review

Notation

We write

$$\begin{aligned}\{X \in A\}&\text{for } \{w \in \Omega : X(w) \in A\}\\ \{X = c\}&\text{for } \{w \in \Omega : X(w) = c\}\end{aligned}$$

and, accordingly,

$$\begin{aligned}\mathcal{D}(\{X \in A\}) &= \mathcal{D}(\{w \in \Omega : X(w) \in A\})\\ \mathcal{D}(\{X = c\}) &= \mathcal{D}(\{w \in \Omega : X(w) = c\})\end{aligned}$$

It is further customary to write:

- $z \sim \mathcal{D}$ ... "z is drawn (randomly) according to $\mathcal{D}$"
- $\mathbb{P}_{z\sim\mathcal{D}}[X(z)] = \mathcal{D}(\{z \in \Omega : X(z) = 1\})$ for $X : \Omega \to \{0, 1\}$

Probability review

Indicator function as a random variable

Let $(\Omega, \mathcal{F}, \mathcal{D})$ be a probability space and $A \in \mathcal{F}$. The **indicator function**

$$U(z) = 1_A(z) = \begin{cases} 1, & z \in A \\ 0, & z \in A^c \end{cases}$$

is a random variable. This is easy to verify; let $a \in \mathbb{R}$ and consider the events

$$\{z \in \Omega : U(z) \leq a\} = \begin{cases} \Omega, & a \geq 1 \\ \emptyset, & a < 0 \\ A^c, & a \in [0, 1) \end{cases}$$

We have $\Omega, \emptyset, A^c \in \mathcal{F}$, as $A \in \mathcal{F}$ and $\mathcal{F}$ is a σ algebra. Hence U is measurable.

Probability review

Conditional probability, independence, i.i.d.

Conditional probability of event A, given B (with $\mathcal{D}(B) \neq 0$):

$$\mathcal{D}(A|B) = \frac{\mathcal{D}(A \cap B)}{\mathcal{D}(B)}$$

The events A and B are said to be independent, if

$$\mathcal{D}(A \cap B) = \mathcal{D}(A)\mathcal{D}(B)$$

Independently and identically distributed (i.i.d.) RVs

A sequence of RVs $X_1, \ldots, X_n$ is said to be independently and identically distributed (i.i.d.) when the X_i are (1) mutually independent and (2) follow the same distribution.

Probability review

Some basic formula

Let A, B, $A_1, \ldots, A_n$ be **events** (in $\mathcal{F}$) with $\mathcal{D}(B) \neq 0$. Then,

Sum rule

$$\mathcal{D}(A \cup B) = \mathcal{D}(A) + \mathcal{D}(B) - \mathcal{D}(A \cap B) \tag{B.1}$$

Union bound

$$\mathcal{D}\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} \mathcal{D}(A_i) \tag{B.2}$$

Bayes formula

$$\mathcal{D}(A|B) = \frac{\mathcal{D}(B|A)\mathcal{D}(A)}{\mathcal{D}(B)} \tag{B.3}$$

Chain rule

$$\mathcal{D}\left(\bigcap_{i=1}^{n} A_i\right) = \mathcal{D}(A_1)\mathcal{D}(A_2|A_1)\cdots\mathcal{D}\left(A_n \middle| \bigcap_{i=1}^{n-1} A_i\right) \tag{B.4}$$

Probability review

Expectation

We denote by $\mathbb{E}\left[X\right]$ the expectation[3] of the random variable X, *i.e.*,

$$\mathbb{E}\left[X\right] = \int_{-\infty}^{\infty} x \; f(x) dx \tag{B.5}$$

if X admits a probability density function[4] f. In the discrete case:

$$\mathbb{E}\left[X\right] = \sum_{x} x \; \mathbb{P}\left[X = x\right] \tag{B.6}$$

Properties (of expectation):

- Linearity: $\mathbb{E}\left[aX + bY\right] = a\mathbb{E}\left[X\right] + b\mathbb{E}\left[Y\right]$, for $a, b \in \mathbb{R}$
- If X, Y are independent: $\mathbb{E}\left[XY\right] = \mathbb{E}\left[X\right]\mathbb{E}\left[Y\right]$

[3] we omit the subscript $_{z \sim \mathcal{D}}$ if it's clear from context!

[4] This can be defined without a density, but for our purpose this is enough.

Concentration inequalities

What are concentration inequalities?

Concentration inequalities give bounds for a RV to be concentrated around its mean, or for it to deviate from its mean.

Concentration inequalities

Markov's inequality

Let Z be a *non-negative* RV. We have

$$\begin{aligned}\mathbb{E}[Z] = \int_{x=0}^{\infty} \mathbb{P}[Z \geq x]dx &\geq \int_{x=0}^{a} \mathbb{P}[Z \geq x]dx \\ &\geq \int_{x=0}^{a} \mathbb{P}[Z \geq a]dx = a \cdot \mathbb{P}[Z \geq a]\end{aligned}$$

as $\mathbb{P}[Z \geq x]$ is monotonically *non-increasing*.

Markov's inequality

$$\Rightarrow \forall a > 0 : \mathbb{P}[Z \geq a] \leq \frac{\mathbb{E}[Z]}{a} \tag{B.7}$$

Concentration inequalities

Chebyshev's inequality

What about $|Z - \mathbb{E}[Z]|$?

Let's take a closer look at $(Z - \mathbb{E}[Z])^2$:

$$\forall a > 0 : \mathbb{P}[|Z - \mathbb{E}[Z]| \geq a] = \mathbb{P}\left[(Z - \mathbb{E}[Z])^2 \geq a^2\right]$$

Note that $\mathbb{V}[Z] = \mathbb{E}\left[(Z - \mathbb{E}[Z])^2\right]$. Invoking Markov's inequality yields

Chebychev's inequality

$$\forall a > 0 : \mathbb{P}[|Z - \mathbb{E}[Z]| \geq a] \leq \frac{\mathbb{V}[Z]}{a^2} \tag{B.8}$$

Concentration inequalities

Hoeffding's inequality

Let

- $Z_1, \ldots, Z_m$ be a sequence of m i.i.d. RV's
- $\overline{Z} = {}^1/_m \sum_i Z_i$ (*i.e.*, empirical average)
- $\mathbb{E}\left[\overline{Z}\right] = \mu$
- $\forall i : \mathbb{P}\left[a \leq Z_i \leq b\right] = 1$

Then, for any $\epsilon > 0$

Hoeffding's inequality

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right| \geq \epsilon\right] \leq 2e^{-2m\epsilon^2/(b-a)^2} \tag{B.9}$$

Concentration inequalities

Proof of Hoeffding's inequality

Set RV $X_i = Z_i - \mu$ and let $\overline{X} = {}^1\!/_m \sum_i X_i$. Then,

$$\mathbb{P}\left[\overline{X} \geq \epsilon\right] = \underbrace{\mathbb{P}\left[e^{\lambda \overline{X}} \geq e^{\lambda\epsilon}\right]}_{e^x \text{ is monotonically increasing}}$$

$$\overset{(\text{B.7})}{\leq} e^{-\lambda\epsilon}\mathbb{E}\left[e^{\lambda\overline{X}}\right] \text{ for } \epsilon > 0, \lambda > 0$$

Lemma B.1: Hoeffding's lemma

Let X be a RV with values in $[a,b]$, such that $\mathbb{E}[X] = 0$. Then, for every $\lambda > 0$

$$\mathbb{E}\left[e^{\lambda X}\right] \leq e^{\frac{\lambda^2(b-a)^2}{8}}$$

For a proof, see [4].

Concentration inequalities

Hoeffding's inequality - Proof (contd.)

Due to the *i.i.d. assumption* we have

$$\mathbb{E}\left[e^{\lambda\overline{X}}\right] = \mathbb{E}\left[e^{\lambda 1/m \sum_i X_i}\right] = \mathbb{E}\left[\prod_i e^{\lambda X_i/m}\right] = \prod_i \mathbb{E}\left[e^{\lambda X_i/m}\right]$$

Hence, by virtue of Lemma B.1, we can bound each term in the last product as follows:

$$\mathbb{E}\left[e^{\lambda/m X_i}\right] \leq e^{\frac{\lambda^2(b-a)^2}{8m^2}}$$

Consequently,

$$\mathbb{P}\left[\overline{X} \geq \epsilon\right] \leq e^{-\lambda\epsilon}\prod_i e^{\frac{\lambda^2(b-a)^2}{8m^2}} = e^{-\lambda\epsilon + \frac{\lambda^2(b-a)^2}{8m}}$$

Concentration inequalities

Hoeffding's inequality - Proof (contd.)

By setting $\lambda = \frac{4m\epsilon}{(b-a)^2}$ we get

$$\mathbb{P}\left[\overline{X} \geq \epsilon\right] \leq e^{\frac{-2m\epsilon^2}{(b-a)^2}} \quad .$$

However, this is only part I, since Hoeffding's inequality is of the form

$$\mathbb{P}\left[|\cdot| \ \geq \epsilon\right] \leq \cdots \quad \text{(note the absolute value)}$$

Hence, we also need $\mathbb{P}\left[\overline{X} \leq -\epsilon\right]$. Similarly, we get $\mathbb{P}\left[\overline{X} \leq -\epsilon\right] \leq e^{\frac{-2m\epsilon^2}{(b-a)^2}}$. Now, invoking the union bound of Eq. (B.2) gives

$$\underbrace{\mathbb{P}\left[(\overline{X} \geq \epsilon) \cup (\overline{X} \leq -\epsilon)\right]}_{\mathbb{P}\left[|\overline{X}| \geq \epsilon\right] \text{ with } \overline{X} = 1/m \sum_i X_i} \leq 2e^{-2m\epsilon^2/(b-a)^2}$$

This concludes our proof of Eq. (B.9).

Introduction to the learning framework

Introduction to the learning framework

Definitions – The **domain set** $\mathcal{X}$

Points in $\mathcal{X}$ are referred to as instances. Typically, instances are represented by a vector of features (e.g., in $\mathbb{R}^d$).

Example (papayas):

Instances could be represented by the 2-tuple (softness, color).

Introduction to the learning framework

Definitions – The **label set** $\mathcal{Y}$

Example: Papayas could be *tasty (0) vs. non-tasty (1)*, *i.e.*, $\mathcal{Y} = \{0, 1\}$.

For now, we limit ourselves to such binary classification tasks. We later extend this to more general settings.

Introduction to the learning framework

Definitions – **Training data S**

We define *training data* as a finite sequence of instances + label tuples $S = ((x_1, y_1), \dots, (x_m, y_m))$, with $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

Example: Papaya training data

$\mathcal{X}$	$\mathcal{Y}$
(green,firm)$= x_1$	$y_1 = 0$ (tasty)
(green,soft)$= x_2$	$y_2 = 0$ (tasty)
$\vdots$	$\vdots$
(yellow,soft)$= x_m$	$y_m = 1$ (non-tasty)

This is the data the *learner* (in a supervised setting) has access to.

Introduction to the learning framework

Definition – **Learner output**

The output of the learner is a prediction rule (a.k.a. hypothesis)

$$\boxed{h : \mathcal{X} \to \mathcal{Y}}$$

Say we have a learning algorithm A, then h is the result of running the learning algorithm[5] on the training data S, denoted by $A(S)$.

[5] we will write h_S to indicate the dependency on the training data S when required

Introduction to the learning framework

Assumptions about the **data generation** process

Our assumptions are:

- Let $\mathcal{D}$ be a probability distribution over $\mathcal{X}$

  (Note that the learner has **no knowledge** of $\mathcal{D}$)

For now, we also assume a correct labeling function $f : \mathcal{X} \to \mathcal{Y}$.

We say that the **data** is generated as follows:

1. Instances x_i are drawn i.i.d. from $\mathcal{D}$
2. Then, instances are labeled by f, *i.e.*, $\forall i : y_i = f(x_i)$

The learner has only access to $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_m)$.

Introduction to the learning framework

Definition – **Generalization error**

Definition 1.1: Generalization error

Given a hypothesis h, a label function f and a probability distribution $\mathcal{D}$ over $\mathcal{X}$, the generalization error (or risk) of h is defined as

$$L_{\mathcal{D},f}(h) = \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)] = \mathbb{E}_{x\sim\mathcal{D}}[1_{h(x)\neq f(x)}] \tag{1.1}$$

In fact, the generalization error can also be written as:

$$L_{\mathcal{D},f}(h) = \mathcal{D}(\underbrace{\{x \in \mathcal{X} : h(x) \neq f(x)\}}_{\text{the event we care about}})$$

The generalization error is the probability to predict the wrong label on a random point drawn from the underlying probability distribution.

Introduction to the learning framework

Definition – **Empirical risk minimization (ERM)**

Importantly, while we would like to minimize the error (by selecting h) w.r.t. the unknown $(\mathcal{D}, f)$, we only have access to S.

Definition 1.2: Empirical error

Given a hypothesis h, a label function f and a training sample S, the empirical error (or empirical risk) of h is defined as [a]

$$L_S(h) = \frac{|\{i \in [m] : h(x_i) \neq f(x_i)\}|}{m} = \frac{1}{m}\sum_{i=1}^{m} 1_{h(x_i)\neq f(x_i)} \tag{1.2}$$

[a] $[m] \equiv \{1, \ldots, m\}$

The learning paradigm to choose a h that minimizes the empirical error is referred to as **empirical risk minimization (ERM)**.

Introduction to the learning framework

Generalization *vs.* empirical error

Lets review the distinction between the two errors:

The empirical error of h is the average error over the sample S.

$\uparrow\downarrow$

The generalization error of h is the expected error based on $\mathcal{D}$.

We will later see how to relate these two quantities to each other. In fact, we already know that for a hypothesis h

$$L_{\mathcal{D},f}(h) = \mathbb{E}\left[L_S(h)\right]$$

(shown on the next slide)

Introduction to the learning framework

Generalization *vs.* empirical error (contd.)

Assume $x_1, \ldots, x_m$ (our training set S) be i.i.d. w.r.t. $\mathcal{D}$ and let $y_i = f(x_i)$. We denote this assumption by $S \sim \mathcal{D}^m$ from here on.

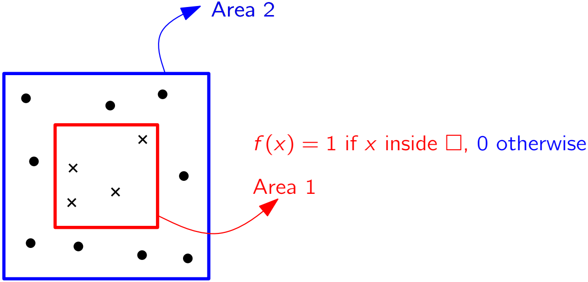
We are now ready to show that $L_{\mathcal{D},f}(h) = \mathbb{E}\left[L_S(h)\right]$

$$
\begin{aligned}
\mathbb{E}_{S\sim\mathcal{D}^m}[L_S(h)] &= \frac{1}{m}\sum_i \mathbb{E}_{S\sim\mathcal{D}^m}[1_{h(x_i)\neq f(x_i)}] && \text{(by linearity)}\\
&= \frac{1}{m}\sum_i \mathbb{E}_{S\sim\mathcal{D}^m}[1_{h(x)\neq f(x)}] && \text{(by i.i.d.)}\\
&= m \cdot \frac{1}{m} \cdot \mathbb{E}_{S\sim\mathcal{D}^m}[1_{h(x)\neq f(x)}] && (\text{since } \sum_i = m)\\
&= \mathbb{E}_{x\sim\mathcal{D}}[1_{h(x)\neq f(x)}]\\
&= L_{\mathcal{D},f}(h) && \text{(by definition)}
\end{aligned}
$$

Introduction to the learning framework

A (naive) example of learning via ERM

Let $\mathcal{D}$ be such that points are uniformly distributed in the square □.



Area 2

$f(x) = 1$ if x inside □, 0 otherwise

Area 1

Now, let

$$h_S(x) = \begin{cases} y_i, & \text{if } \exists i \in [m] \text{ such that } x_i = x \\ 0, & \text{else} \end{cases}$$

Clearly, $L_S(h_S) = 0$ (and thus a valid ERM algorithm).

Introduction to the learning framework

ERM Example (contd.)

However, the true (generalization) error is

$$L_{\mathcal{D},f}(h_S) = 1/2, \text{ compared to } L_S(h_S) = 0 \ .$$

$\Rightarrow$ we would do a bad job in predicting the label of new instances.

Explanation: Unless new samples are from the training data, we always predict 0. As the probability to draw a new sample with label 0 or 1 is 1/2 (by def.), $L_{\mathcal{D},f}(h_S) = 1/2$.

This phenomenon is called overfitting.

In some sense, the hypothesis fits the data “too well”.

Introduction to the learning framework

ERM with inductive bias[6]

We look for conditions under which the ERM predictor does not overfit!
Idea: Restrict the search space for h

Hypothesis class $\mathcal{H}$

We denote by $\mathcal{H}$ a hypothesis class, where $\forall h \in \mathcal{H} : h : \mathcal{X} \to \mathcal{Y}$.

Formally,

$$\mathrm{ERM}_{\mathcal{H}}(S) \in \arg\min_{h \in \mathcal{H}} L_S(h)$$

where $\mathrm{ERM}_{\mathcal{H}}(S)$ denotes the ERM learner over $\mathcal{H}$, based on S (with output h_S).

Where is the inductive bias? In the restriction of the ERM learner to select h from the class $\mathcal{H}$.

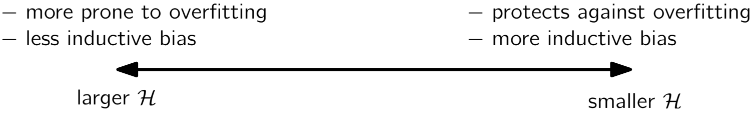
[6] Remember, inductive bias $\equiv$ incorporating prior knowledge!

Introduction to the learning framework

ERM with inductive bias (contd.)

One fundamental question of learning theory is:

> Over which hypothesis classes does ERM not overfit?



– more prone to overfitting
– less inductive bias

– protects against overfitting
– more inductive bias

larger $\mathcal{H}$

smaller $\mathcal{H}$

Introduction to the learning framework

Probably Approximately Correct (PAC) learning – Finite hypothesis classes $\mathcal{H}$

Lets make the following **assumptions** (for now):

- $\mathcal{H}$ is a finite hypothesis class and every ERM hypothesis

$$h_S \in \arg\min_{h\in\mathcal{H}} L_S(h)$$

- Realizability[7]: $\exists h^* \in \mathcal{H}$, such that

$$L_{\mathcal{D},f}(h^*) = 0$$

  with probability 1 over all random samples S from $\mathcal{D}$. Hence, $L_S(h^*) = 0$.

[7] this is a *strong* assumption and we will relax it later

Introduction to the learning framework

Probably **A**pproximately **C**orrect (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Can we guarantee to find h_S, such that $L_{\mathcal{D},f}(h_S) = 0$?

Claim 1

We can only be approximately correct.

Why? We cannot guarantee perfect label prediction, since $L_{\mathcal{D},f}(h_S)$ depends on S (which is randomly picked).

Here is an example:

- Let $\mathcal{X} = \{x_1, x_2\}, \mathcal{D}(\{x_1\}) = 1 - \epsilon$ and $\mathcal{D}(\{x_2\}) = \epsilon$ with $\epsilon \in (0, 1)$
- Probability of not seeing x_2 among m i.i.d. samples is

$$(1-\epsilon)^m \approx e^{-\epsilon m} \quad (\text{since } 1 - \epsilon \leq e^{-\epsilon})$$

- $\Rightarrow$ If $\epsilon \ll 1/m$; it's likely not to see x_2 at all (and also its label)

Relax to $L_{\mathcal{D},f}(h_S) \leq \epsilon, \epsilon \in (0, 1)$, *i.e.*, be **approximately** correct.

Introduction to the learning framework

Probably **A**pproximately **C**orrect (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Can we succeed at this goal with certainty?

Claim 2

We can't guarantee $L_{\mathcal{D},f}(h_S) \leq \epsilon$ with certainty.

Why? There is always a probability that S is not representative of $\mathcal{D}$

Relax to $L_{\mathcal{D},f}(h_S) \leq \epsilon$ with probability $\delta \in (0, 1)$, *i.e.*, be **probably** correct.

We are interested in bounding the probability to sample a m-tuple of training instances such that the learner fails, *i.e.*, $L_{\mathcal{D},f}(h_S) > \epsilon$.

Introduction to the learning framework

Probably **A**pproximately **C**orrect (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Formally, this means we want to bound $\mathcal{D}^m(\{S|_x : L_{\mathcal{D},f}(h_S) > \epsilon\})$[a].

[a] $S|_x = (x_1, \dots, x_m)$, *i.e.*, the restriction of S to the data instances

Let's do this! First, we define two sets:

Set of **bad hypotheses** (*i.e.*, the learner fails on those h)

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{\mathcal{D},f}(h) > \epsilon\}$$

Set of **misleading samples** (*i.e.*, the learner fails, but empirical risk is 0)

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

Introduction to the learning framework

Probably **A**pproximately **C**orrect (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Second, since realizability implies $L_S(h_S) = 0$ (for every ERM hypothesis), the case

$$L_{\mathcal{D},f}(h) > \epsilon$$

can only happen if $h \in \mathcal{H}_B$ and $L_S(h) = 0$. We have

$$\begin{aligned}\{S|_x : L_{\mathcal{D},f}(h_S) > \epsilon\} \subseteq M &= \{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\} \\ &= \bigcup_{h \in \mathcal{H}_B} \{S|_x : L_S(h) = 0\}\end{aligned}$$

Why? In $M = \{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\}$, we say *there exists* a $h \in \mathcal{H}_B$, such that $L_S(h) = 0$. Yet, such h might *not* be chosen by ERM $\Rightarrow \subseteq$.

Introduction to the learning framework

Probably Approximately Correct (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Now, lets look at the probability of these events:

$$\begin{aligned}\mathcal{D}^m(\{S|_x : L_{\mathcal{D},f}(h_S) > \epsilon\}) &\leq \mathcal{D}^m(\cup_{h \in \mathcal{H}_B}\{S|_x : L_S(h) = 0\}) \\ &\underbrace{\leq \sum_{h \in \mathcal{H}_B} \mathcal{D}^m(\{S|_x : L_S(h) = 0\})}_{\text{by virtue of the union bound (B.2)}}\end{aligned}$$

Introduction to the learning framework

Probably **A**pproximately **C**orrect (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Third, we try to bound the terms $\mathcal{D}^m(\{S|_x : L_S(h) = 0\})$ in the last sum (on the previous slide). We know that samples in S are i.i.d. Thus,

$$\begin{aligned}
\mathcal{D}^m(\{S|_x : L_S(h) = 0\}) &= \mathcal{D}^m(\{S|_x : \forall i : h(x_i) = f(x_i)\}) \\
&\overset{\text{i.i.d}}{=} \prod_{i=1}^{m} \mathcal{D}(\{x_i : h(x_i) = f(x_i)\}) \\
&= \prod_{i=1}^{m} \underbrace{[1 - L_{\mathcal{D},f}(h)]}_{\leq 1-\epsilon, \text{ since } h \in \mathcal{H}_B} \\
&\leq (1-\epsilon)^m \\
&\leq e^{-\epsilon m}
\end{aligned}$$

Introduction to the learning framework

Probably **A**pproximately **C**orrect (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

We are now ready to combine what we have so far:

$$\begin{aligned}
\mathcal{D}^m(\{S|_x : L_{\mathcal{D},f}(h_S) > \epsilon\}) &\leq \mathcal{D}^m(\cup_{h\in\mathcal{H}_B}\{S|_x : L_S(h) = 0\}) \\
&\leq \sum_{h\in\mathcal{H}_B} \mathcal{D}^m(\{S|_x : L_S(h) = 0\}) \\
&\leq \sum_{h\in\mathcal{H}_B} e^{-\epsilon m} \\
&= |\mathcal{H}_B|e^{-\epsilon m} \\
&\leq |\mathcal{H}|e^{-\epsilon m}
\end{aligned}$$

Now, what do we need for the last inequality to be $\leq \delta$?

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

Introduction to the learning framework

Probably Approximately Correct (PAC) learning - Finite hypothesis classes $\mathcal{H}$ (contd.)

Let's summarize this result in the following corollary:

Corollary 1.1: PAC learning of finite hypothesis classes $\mathcal{H}$

Fix $\epsilon, \delta \in (0,1)$. *If* $m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$, *then for any* $\mathcal{D}, f$ *(assuming realizability), with probability* $1-\delta$ *(over the choice of* S *of size* m*), we have (for every ERM hypothesis* h_S*)*

$$L_{\mathcal{D},f}(h_S) \leq \epsilon \ .$$

In other words, the ERM paradigm over finite $\mathcal{H}$ will output a hypothesis that will probably (with $1-\delta$) be approximately (up to error ϵ) correct[8].

[8] we will more formally define PAC learning next

PAC Learning

(*cf.* [4, Chapter 3])

Leslie Valiant, ACM Turing Award 2010

PAC Learning

Definition

Definition 2.1: PAC learnability

A hypothesis class $\mathcal{H}$ is PAC learnable if

$$\exists m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$$

and a learning algorithm A with the following property: For

- *every $\epsilon, \delta \in (0,1)$, and*
- *every distribution $\mathcal{D}$ over $\mathcal{X}$, and*
- *every label function $f : \mathcal{X} \to \mathcal{Y}$,*

if realizability holds w.r.t. $\mathcal{H}, \mathcal{D}, f$, then running A on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ labeled (by f) i.i.d. samples of $\mathcal{D}$, A returns a hypothesis h s.t. with probability $1 - \delta$

$$L_{\mathcal{D},f}(h) \leq \epsilon \quad .$$

PAC Learning

Sample complexity

The function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ measures the **sample complexity**, *i.e.*, how many samples are required to guarantee PAC learnability of $\mathcal{H}$.

Sample complexity (of learning $\mathcal{H}$)

The function $m_{\mathcal{H}}(\epsilon, \delta)$ which, for any $\epsilon, \delta \in (0,1)$, returns the smallest integer such that $\mathcal{H}$ is PAC learnable.

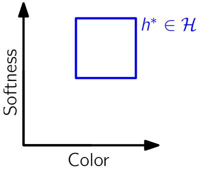
PAC Learning

Relaxing the "realizability" assumption

Formally, "realizability" means

$$\exists h^* \in \mathcal{H} : \mathbb{P}_{x \sim \mathcal{D}}[h^*(x) = f(x)] = 1$$

Example (papayas): say $\mathcal{H}$ is the class of rectangles



Can we really guarantee that there exists a $h^* \in \mathcal{H}$ that fully determines if papayas are tasty?

Waiving "realizability" leads to the notion of **agnostic** PAC learning.

PAC Learning

Agnostic PAC learning

- $\mathcal{D}$ is now a probability distribution over $\mathcal{X} \times \mathcal{Y}$ with two parts:
    1. $\mathcal{D}_x$ is a distribution over unlabled instances
    2. $\mathcal{D}((x, y)|x)$ is the conditional probability over labels for each x[9]

Generalization error (in agnostic PAC learning)

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] = \mathcal{D}(\{(x, y) : h(x) \neq y\}) \tag{2.1}$$

Empirical error (in agnostic PAC learning)

$$L_S(h) = \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m} \tag{2.2}$$

[9] **Note**: this allows, e.g., for papayas with same (softness, color) to have different labels (here: taste)!

PAC Learning

Agnostic PAC learning (contd.)

Definition 2.2: Agnostic PAC learnability

A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if

$$\exists m_{\mathcal{H}} : (0,1)^2 \rightarrow \mathbb{N}$$

and a learning algorithm A with the following property: For

- *every $\epsilon, \delta \in (0,1)$, and*
- *every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$,*

running A on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. samples of $\mathcal{D}$ returns a hypothesis h such that with probability $1 - \delta$ (over the choice of S)

$$\underbrace{L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon}_{\text{i.e., measured relative to the \underline{best} achievable error}}$$

PAC Learning

Agnostic PAC learning (contd.)

	basic PAC	**agnostic** PAC
Distribution	$\mathcal{D}$ over $\mathcal{X}$	$\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$
Truth	$f \in \mathcal{H}$	not in $\mathcal{H}$ or does not exist
Risk	$\mathcal{D}(\{x : h(x) \neq f(x)\})$	$\mathcal{D}(\{(x,y) : h(x) \neq y\})$
Training	$(x_1, \ldots, x_m) \sim \mathcal{D}^m$, $y_i = f(x_i)$	$(x_1, y_1), \ldots, (x_m, y_m) \sim \mathcal{D}^m$
Goal	$L_{\mathcal{D},f}(A(S)) \leq \epsilon$	$L_{\mathcal{D}}(A(S)) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$

Exercise: rewrite the goal of agnostic PAC learning as

$$\mathcal{D}^m \left(\left\{ S \in Z^m : L_{\mathcal{D}}(A(S)) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon \right\} \right) \geq 1 - \delta$$

where $z = (x, y) \in Z = \mathcal{X} \times \mathcal{Y}$.

PAC Learning

Introducing (general) **loss functions**

Loss function

Given a hypothesis class $\mathcal{H}$ and some domain Z, the function

$$\ell : \mathcal{H} \times Z \to \mathbb{R}_+$$

is called a loss function[a].

[a] in prediction problems, we know that $Z = \mathcal{X} \times \mathcal{Y}$

Generalization error (for a loss function ℓ)

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

PAC Learning

Introducing (general) **loss functions** (contd.)

Empirical error (for loss functions ℓ)

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i)$$

Examples: here, z ranges over the set of pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$

$0 - 1$ loss

$$\ell_{0-1}(h, (x, y)) = 1_{h(x)=y} := \begin{cases} 0, & \text{if } h(x) = y \\ 1, & \text{otherwise} \end{cases} \tag{2.3}$$

Squared loss

$$\ell_{sq}(h, (x, y)) = (h(x) - y)^2 \tag{2.4}$$

Note that ℓ_{sq} is typically used in **regression problems**.

Learning via Uniform Convergence

(*cf*. [4, Chapter 4])

Learning via Uniform Convergence

The basic idea of uniform convergence

At this point, one of our motivating questions is:

> *"Are finite hypothesis classes $\mathcal{H}$ agnostic PAC learnable?"*

We already know that

- finite hypothesis classes $\mathcal{H}$ are (basic) PAC learnable (w.r.t. ℓ_{0-1})

Definition 2.3: ϵ-representative sample

A training set S is called ϵ-representative w.r.t. $\mathcal{H}, \mathcal{D}$ and ℓ if

$$\forall h \in \mathcal{H} : |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon \tag{2.5}$$

Intuitively, we want to make sure that the empirical risks of the hypotheses are close to their true risks (not a big surprise :)

Learning via Uniform Convergence

${}^{\epsilon}/_{2}$-representativeness ensures "good" ERM hypotheses

Lemma 2.1

Assume that S is ${}^{\epsilon}/_{2}$-representative in the sense of Def. 2.3. Then, any output $h_S \in \arg\min_{h \in \mathcal{H}} L_S(h)$ of an ERM algorithm satisfies

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

Proof.

$$\begin{aligned}
L_{\mathcal{D}}(h_S) &\leq L_S(h_S) + {}^{\epsilon}/_{2} && \text{(since } S \text{ is } \epsilon/2\text{-rep.)} \\
&\leq L_S(h) + {}^{\epsilon}/_{2} && \text{(since } h_S \text{ is ERM hypothesis)} \\
&\leq \underbrace{L_{\mathcal{D}}(h) + {}^{\epsilon}/_{2} + {}^{\epsilon}/_{2}}_{L_{\mathcal{D}}(h)+\epsilon} && \text{(since } S \text{ is } \epsilon/2\text{-rep.)}
\end{aligned}$$

□

Learning via Uniform Convergence

Uniform convergence is sufficient for PAC learnability

If, with probability $1-\delta$ (over the random choice of S), we get a ϵ-representative S, the ERM rule is an agnostic PAC learner.

Definition 2.4: Uniform convergence

A hypothesis class $\mathcal{H}$ has the uniform convergence property if

$$\exists m_{\mathcal{H}}^{UC} : (0,1)^2 \to \mathbb{N}$$

such that for

- *every $\epsilon, \delta \in (0,1)$, and*
- *every probability distribution $\mathcal{D}$ over Z and loss ℓ*

the following holds: if S is an i.i.d. sample of size $m \geq m_{\mathcal{H}}^{UC}(\epsilon, \delta)$, then, with probability $1-\delta$, the sample S is ϵ-representative.

Learning via Uniform Convergence

Uniform convergence ↔ Agnostic PAC learnability

Corollary 2.1

If a class $\mathcal{H}$ has the uniform convergence property with $m_{\mathcal{H}}^{UC}$ then $\mathcal{H}$ is agnostic PAC learnable with $m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta)$. In that case, the ERM paradigm also is an agnostic PAC learner.

Learning via Uniform Convergence

What about our finite hypothesis classes

We want to establish that uniform convergence holds for **finite** $\mathcal{H}$. This would imply that finite $\mathcal{H}$ are agnostic PAC learnable.

First, fix $\epsilon, \delta \in (0,1)$. We need to show that

$$\mathcal{D}^m(\{S : \forall h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon\}) \geq 1 - \delta$$

(where $\mathcal{H}$ is finite). In other words, we need to find m that guarantees

- for any $\mathcal{D}$
- with probability $1-\delta$ over the choice of $S = (z_1, \ldots, z_m)$
- we have that $\forall h \in \mathcal{H}$: $|L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$

Learning via Uniform Convergence

What about our finite hypothesis classes (contd.)

Equivalently, we can show that

$$\underbrace{\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\})}_{=: K} < \delta$$

Recall that

$$\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon|\} = \cup_{h \in \mathcal{H}} \{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}$$

and, by virtue of the union bound Eq. (B.2), we have

$$K \leq \sum_{h \in \mathcal{H}} \mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\})$$

Learning via Uniform Convergence

What about our finite hypothesis classes (contd.)

We next have to show that for sufficiently large m

- and any fixed $h \in \mathcal{H}$ (chosen prior to sampling S)
- the terms in the sum on the right-hand side, *i.e.*,

$$|L_S(h) - L_{\mathcal{D}}(h)|$$

  are likely to be small.

Strategy: we will use one of our concentration inequalities!

Learning via Uniform Convergence

What about our finite hypothesis classes (contd.)

Lets recall that

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

Since, the z_i are i.i.d. and by the linearity of expectation, we have

$$\begin{aligned}
\mathbb{E}_{S \sim \mathcal{D}^m} \underbrace{\left[\frac{1}{m} \sum_i \ell(h, z_i) \right]}_{\text{empirical risk}} &= \frac{1}{m} \sum_i \mathbb{E}_{z_i \sim \mathcal{D}}[\ell(h, z_i)] \\
&= \frac{1}{m} \sum_i \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)] \\
&= \frac{1}{m} \cdot m \cdot \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)] \\
&= L_{\mathcal{D}}(h)
\end{aligned}$$

Hence, $L_{\mathcal{D}}(h)$ is the expected value of $L_S(h)$.

Learning via Uniform Convergence

What about our finite hypothesis classes (contd.)

As a matter of fact, our summation terms

$$|L_S(h) - L_{\mathcal{D}}(h)|$$

are the deviations of $L_S(h)$ from its mean. Fortunately, we can bound these terms via Hoeffding's inequality, see Eq. (B.9):

$$\mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) = \mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i) - \underbrace{\mu}_{L_{\mathcal{D}}(h)}\right| > \epsilon\right] \leq 2e^{-2m\epsilon^2} \tag{2.6}$$

Note: we assume the loss ℓ is in $[0, 1]$ (which is why the denominator $(b-a)^2$ vanishes).

Learning via Uniform Convergence

What about our finite hypothesis classes (contd.)

Now that we have a bound on the summation terms, we have

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

Finally, requiring that the left-hand side $\leq \delta$ yields $m \geq \frac{\log(2|\mathcal{H}|/\delta|)}{2\epsilon^2}$.

Corollary 2.2

Let $\mathcal{H}$ be a finite hypothesis class, Z be a domain and $\ell : \mathcal{H} \times Z \to [0,1]$ a loss function. Then $\mathcal{H}$ has the uniform convergence property with

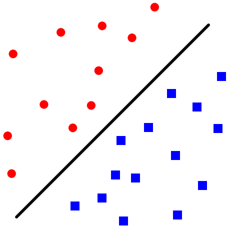
$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{\log(2|\mathcal{H}|/\delta|)}{2\epsilon^2} \right\rceil$$

and $\mathcal{H}$ is agnostically PAC learnable with

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \ .$$

Linear predictors

(*cf.* [4, Chapter 9])



Linear predictors

Definitions

The class of affine functions L_d

Let

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

with

$$h_{\mathbf{w},b} : \mathbb{R}^d \to \mathbb{R}, \mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b$$

be the class of **affine functions**.

Hypothesis classes of linear predictors

The classes of linear predictors are compositions of functions

$$\phi : \mathbb{R} \to \mathcal{Y}$$

on L_d, *e.g.*, $\phi = \mathrm{sign}$.

Linear predictors

Terminology

We call b the bias. Also, by extending $\mathbf{x} \in \mathbb{R}^d$ to

$$\mathbf{x}' = (1, x_1, \dots, x_d) \in \mathbb{R}^{d+1}$$

and letting

$$\mathbf{w}' = (b, w_1, \dots, w_d) \in \mathbb{R}^{d+1}$$

we can write

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle,$$

i.e., a homogeneous linear function. Under this transformation[10], L_d becomes the class of homogeneous linear functions $h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$.

[10] *i.e.*, adding 1 to each input vector

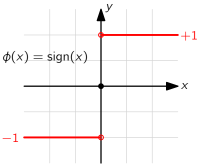
Linear predictors

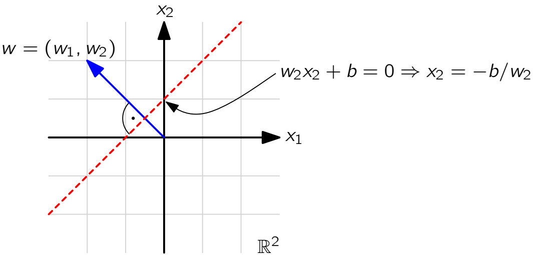
Halfspaces

Let

- $\mathcal{X} = \mathbb{R}^d$,
- $\mathcal{Y} = \{+1, -1\}$, and
- $\mathcal{HS}_d = \text{sign} \circ L_d = \{\mathbf{x} \mapsto \text{sign}(h_{\mathbf{w},b}(x)) : h_{\mathbf{w},b} \in L_d\}$

Geometric interpretation in $\mathbb{R}^2$:





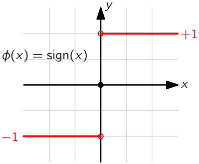
Linear predictors

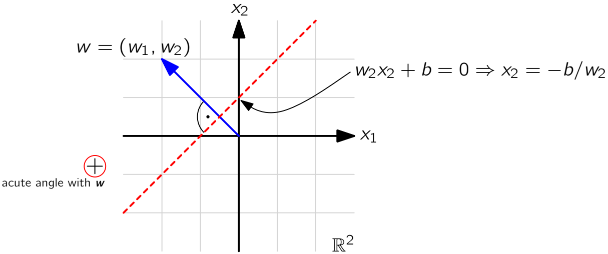
Halfspaces

Let

- $\mathcal{X} = \mathbb{R}^d$,
- $\mathcal{Y} = \{+1, -1\}$, and
- $\mathcal{HS}_d = \text{sign} \circ L_d = \{\mathbf{x} \mapsto \text{sign}(h_{\mathbf{w},b}(x)) : h_{\mathbf{w},b} \in L_d\}$

Geometric interpretation in $\mathbb{R}^2$:




x_2
$w = (w_1, w_2)$
$w_2 x_2 + b = 0 \Rightarrow x_2 = -b/w_2$
x_1
$\oplus$
acute angle with $\boldsymbol{w}$
$\mathbb{R}^2$

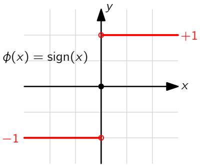
Linear predictors

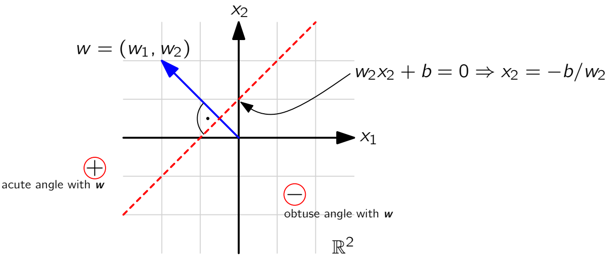
Halfspaces

Let

- $\mathcal{X} = \mathbb{R}^d$,
- $\mathcal{Y} = \{+1, -1\}$, and
- $\mathcal{HS}_d = \text{sign } \circ L_d = \{\mathbf{x} \mapsto \text{sign}(h_{\mathbf{w},b}(x)) : h_{\mathbf{w},b} \in L_d\}$

Geometric interpretation in $\mathbb{R}^2$:





x_2
$w = (w_1, w_2)$
$w_2 x_2 + b = 0 \Rightarrow x_2 = -b/w_2$
x_1
$+$
acute angle with $\boldsymbol{w}$
$-$
obtuse angle with $\boldsymbol{w}$
$\mathbb{R}^2$

Linear predictors

ERM procedure(s) for halfspaces – Linear programming

First, what is a linear program (LP)?

Linear program (LP)

Let $\mathbf{A} \in \mathbb{R}^{m \times d}$ be a matrix and $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{u} \in \mathbb{R}^d$ vectors. Problems of the form:

$$\begin{aligned} \max_{\mathbf{w} \in \mathbb{R}^d} \quad & \langle \mathbf{w}, \mathbf{u} \rangle \\ \text{subject to} \quad & \mathbf{A}\mathbf{w} \geq \mathbf{v} \end{aligned}$$

are referred to as linear programs where $\mathbf{w} \in \mathbf{R}^d$ is the unknown vector of variables.

Linear predictors

ERM procedure(s) for halfspaces – Linear programming (contd.)

Assumptions:

- Realizability[11] (*i.e.*, the data is separable by a hyperplane)
- Training data $S = \{(x_i, y_i)\}_{i=1}^m$ with $y_i \in \{+1, -1\}$

$\Rightarrow$ ERM predictor should have 0 error on S. We look for $\mathbf{w} \in \mathbb{R}^d$ such that

$$\forall i \in [m] : \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle) = y_i,$$

or, equivalently

$$\forall i \in [m] : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$$

[11] in the non-realizable case, this is **computationally hard** for ℓ_{0-1} loss

Linear predictors

ERM procedure(s) for halfspaces – Linear programming (contd.)

Let $\mathbf{w}^*$ be such a vector. Set

$$\begin{aligned}\gamma = \min_i y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle &\Rightarrow \forall i \in [m] : y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq \gamma \\ &\Rightarrow \forall i \in [m] : \frac{1}{\gamma} y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1\end{aligned}$$

Now, set $\overline{\mathbf{w}} = \mathbf{w}^*/\gamma$, then

$$\forall i \in [m] : y_i \langle \overline{\mathbf{w}}, \mathbf{x}_i \rangle \geq 1$$

which shows that there exists a vector $\mathbf{w}$, such that

$$\forall i \in [m] : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$$

By design, this is a valid ERM predictor.

Linear predictors

ERM procedure(s) for halfspaces – Linear programming (contd.)

Now, we frame the search for such a $\mathbf{w}$ as a linear program. Let

$$A = \begin{pmatrix} \mathbf{x}_1 y_1 \\ \mathbf{x}_2 y_2 \\ \vdots \\ \mathbf{x}_m y_m \end{pmatrix} \in \mathbb{R}^{m \times d} \text{ and } \mathbf{v} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^m,$$

then our problem of finding $\mathbf{w}$ (see previous slide) yields

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

Any $\mathbf{w}$ which satisfies this is valid. Hence, our objective (for the LP) can simply be a "dummy" $\langle \mathbf{w}, \mathbf{u} \rangle$ with $\mathbf{u} = \mathbf{0}$ (depends on the LP solver, though).

Linear predictors

ERM procedure(s) for halfspaces – Linear programming (contd.)

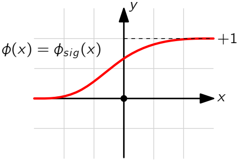
See the lecture GIT repository for a full implementation of this (on the IRIS dataset) in Python.

Linear predictors

ERM procedure(s) for halfspaces – Logistic regression

Let

- $\mathcal{X} = \mathbb{R}^d$
- $\mathcal{Y} = \{+1, -1\}$ (despite the word "regression")
- $\mathcal{LR}_d = \phi_{sig} \circ L_d = \{\mathbf{x} \mapsto \phi_{sig}(\langle \mathbf{w}, \mathbf{x} \rangle), \mathbf{w} \in \mathbb{R}^d\}$



$$\boxed{\phi_{sig}(x) = \frac{1}{1+e^{-x}}}$$

If $u = \langle \mathbf{w}, \mathbf{x} \rangle$ is

- large, $\phi_{sig}(u)$ is close to 1
- small, $\phi_{sig}(u)$ is close to 0

Linear predictors

ERM procedure(s) for halfspaces – Logistic regression (contd.)

What about a suitable loss function?

We would want

- $h_{\mathbf{w}}(\mathbf{x})$ to be large if $y = +1$, and
- $h_{\mathbf{w}}(\mathbf{x})$ to be small if $y = -1$ (or $1 - h_{\mathbf{w}}(\mathbf{x})$ large, resp.)

Consequently, a reasonable loss function would increase monotonically with

$$\frac{1}{1 + e^{y\langle \mathbf{w}, \mathbf{x}\rangle}}, \text{ or } 1 + e^{-y\langle \mathbf{w}, \mathbf{x}\rangle}$$

The loss function used in logistic regression is

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = \underbrace{\log(1 + e^{-y\langle \mathbf{w}, \mathbf{x}\rangle})}_{\text{convex w.r.t. } \mathbf{w}} \quad \text{(log is monotonic)}$$

Linear predictors

ERM procedure(s) for halfspaces – Logistic regression (contd.)

Due to the convexity of the loss function[12], the ERM rule can be implemented efficiently

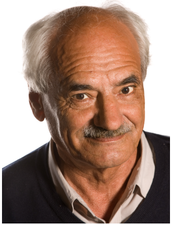
$$\arg\min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + e^{-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle}\right)$$

Logistic regression implements the ERM rule efficiently without relying on the realizability assumption (via the use of a different loss).

[12] we discuss later why this is important

Vapnik-Chervonenkis (VC) Dimension

(*cf.* [4, Chapter 6])



Vladimir Vapnik & Alexey Chevronenkis

VC-Dimension

Motivation

A question that we have not yet answered is:

"What hypothesis classes $\mathcal{H}$ are PAC learnable?"

We do already know that:

- finite $\mathcal{H}$ are PAC learnable
- the class of all functions over an infinite domain is not PAC learnable (will be shown in the "No-Free-Lunch" lecture)

We will see that the size of the hypothesis class is not the right characterization to answer that question.

VC-Dimension

An example of PAC learnability for an infinite hypothesis class $\mathcal{H}$

Lets look at the class of threshold functions on $\mathbb{R}$, *i.e.*

$$\mathcal{H} = \{h_a : a \in \mathbb{R}\}, \text{ with } h_a(x) = \begin{cases} 1, & \text{if } x < a \\ 0, & \text{else} \end{cases}$$

Note that $|\mathcal{H}| = \infty$.

Lemma 4.1

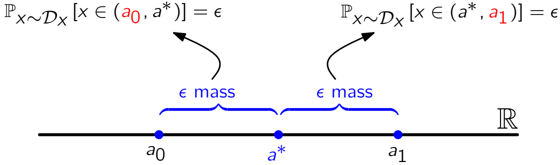
Let $\mathcal{H}$ be defined as above. Then $\mathcal{H}$ is PAC learnable, using the ERM rule, with sample complexity

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(2/\delta)}{\epsilon} \right\rceil \ .$$

VC-Dimension

An example of PAC learnability for an infinite hypothesis class $\mathcal{H}$ (contd.)

Proof. Fix a^* such that $h^*(x) = 1_{x<a^*}$ and $L_{\mathcal{D}}(h^*) = 0$. Further, let a_0, a_1 be such that


$\mathbb{P}_{x\sim\mathcal{D}_X}[x \in (a_0, a^*)] = \epsilon$
$\mathbb{P}_{x\sim\mathcal{D}_X}[x \in (a^*, a_1)] = \epsilon$
ϵ mass
ϵ mass
$\mathbb{R}$
a_0
a^*
a_1

Given a training set S, we let

- $b_0 = \max\{x : (x, 1) \in S\}$, $b_0 = -\infty$ if no $(x, y) \in S$ with $y = 1$
- $b_1 = \min\{x : (x, 0) \in S\}$, $b_1 = +\infty$ if no $(x, y) \in S$ with $y = 0$

For an ERM hypothesis h_S (corresponding to a threshold b_S), we have that $b_S \in (b_0, b_1)$ must hold (since we choose b_S based on S).

VC-Dimension

An example of PAC learnability for an infinite hypothesis class $\mathcal{H}$ (contd.)

For PAC learnability, we need $L_{\mathcal{D}}(h_S) \leq \epsilon$. From the graphical construction of the previous slide, we see that

$$b_0 \geq a_0 \text{ and } b_1 \leq a_1$$

is sufficient to guarantee this. By switching to $L_{\mathcal{D}}(h_S) > \epsilon$, we have[13]

$$\begin{aligned}\mathbb{P}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S) > \epsilon] &\leq \mathbb{P}_{S\sim\mathcal{D}^m}[(b_0 < a_0) \ \vee \ (b_1 > a_1)] \\ &\overset{(B.2)}{\leq} \mathbb{P}_{S\sim\mathcal{D}^m}[b_0 < a_0] + \mathbb{P}_{S\sim\mathcal{D}^m}[b_1 > a_1]\end{aligned}$$

By construction (with prob. mass ϵ on (a_0, a^*)) we know that

$$\mathbb{P}_{S\sim\mathcal{D}^m}[b_0 < a_0] = (1-\epsilon)^m \leq e^{-\epsilon m}$$

[13] $\vee$ is a logical **or**

VC-Dimension

An example of PAC learnability for an infinite hypothesis class $\mathcal{H}$ (contd.)

What did we assume?

We did assume that

$$m > \frac{\log(2/\delta)}{\epsilon}$$

Consequently,

$$e^{-\epsilon \frac{\log(2/\delta)}{\epsilon}} = e^{-\log(2/\delta)} = e^{\log(\delta/2)} = {}^{\delta}\!/_{2}$$

and thus

$$\mathbb{P}_{S \sim \mathcal{D}^m}[b_0 < a_0] \leq {}^{\delta}\!/_{2}$$

The same holds for $\mathbb{P}_{S \sim \mathcal{D}^m}[b_1 > a_1]$ and consequently

$$L_{\mathcal{D}}(h_S) \leq \epsilon$$

with probability $1 - \delta$ follows (since we started with $L_{\mathcal{D}}(h_S) > \epsilon$). □

VC-Dimension

Shattering

Definition 4.1: Restricting $\mathcal{H}$ to C

Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0,1\}$ and $C = \{c_1, \ldots, c_m\} \subset \mathcal{X}$. By restricting $\mathcal{H}$ to C, denoted by $\mathcal{H}_C$, we mean

$$\mathcal{H}_C = \{(h(c_1), \ldots, h(c_m)) : h \in \mathcal{H}\}$$

i.e., the set of functions from C to $\{0,1\}$ that can be derived from $\mathcal{H}$.

Definition 4.2: Shattering

A hypothesis class $\mathcal{H}$ shatters a set C if $|\mathcal{H}_C| = 2^{|C|}$, i.e., it comprises all functions from C to $\{0,1\}$.

VC-Dimension

Shattering (contd.)

Example (threshold functions over $\mathbb{R}$):

$$\mathcal{H} = \{h_a : a \in \mathbb{R}\}, \quad h_a(x) = \begin{cases} 1, & \text{if } x < a \\ 0, & \text{else} \end{cases} \tag{4.1}$$

Let $C_1 = \{c_1\}$, hence, there are two functions from C_1 to $\{0, 1\}$

- If $a = c_1 + 1$, h_a maps to 1
- If $a = c_1 - 1$, h_a maps to 0
- Consequently, $|\mathcal{H}_{C_1}| = 2^{|C_1|} = 2$ and $\mathcal{H}$ shatters C_1

VC-Dimension

Shattering (contd.)

Now, let $C_2 = \{c_1, c_2\}$ and w.l.o.g. $c_1 \leq c_2$

- There are $2^2 = 4$ possible label configurations
- $(0,0), (1,1), (1,0)$ are all possible with $h_a \in \mathcal{H}$
- However, $(0,1)$ **cannot** be realized, hence $\mathcal{H}$ does not shatter C_2

Based on the "No-Free-Lunch" theorem, we have the following corollary:

Corollary 4.1

Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0,1\}$. Assume there exists a set $C \subset \mathcal{X}$ of size $2m$ that is shattered by $\mathcal{H}$. Then for any learning algorithm A, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ and $h \in \mathcal{H}$ such that $L_{\mathcal{D}}(h) = 0$, but with probability of $1/7$ over the choice of $S \sim \mathcal{D}^m$, we have $L_{\mathcal{D}}(A(S)) \geq 1/8$.

VC-Dimension

Shattering (contd.)

Why is that so?

1. Since $\mathcal{H}$ shatters C, it contains all functions from C to $\{0, 1\}$.
2. We can construct $\mathcal{D}$ and h such that

$$L_{\mathcal{D}}(h) = 0$$

3. Since, $|S| = m$ and $|C| = 2m$ the result follows from the "No-Free-Lunch" theorem

In other words, if $\mathcal{H}$ shatters some set of size $2m$, we cannot expect to learn from a training set of size m.

VC-Dimension

VC-Dimension – Definition

Definition 4.3: VC-Dimension of a hypothesis class $\mathcal{H}$

The VC-Dimension of a hypothesis class $\mathcal{H}$, denoted by VCdim($\mathcal{H}$), is the maximal size of a set $C \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$.

Theorem 4.1

Let $\mathcal{H}$ be such that VCdim($\mathcal{H}$) $= \infty$. Then $\mathcal{H}$ is not PAC learnable.

Proof. Since VCdim($\mathcal{H}$) $= \infty$, for any training set of size m, there always exists a shattered set of size $2m$. The claim follows from Corollary 4.1. □

VC-Dimension

VC-Dimension – How do we show that $\text{VCdim}(\mathcal{H}) = d$

What do we need to show for $\text{VCdim}(\mathcal{H}) = d$?

1. $\exists C$ with $|C| = d$ that is shattered by $\mathcal{H}$
2. Every set C with $|C| = d + 1$ is not shattered by $\mathcal{H}$

   (this is typically the hard(er) part)

VC-Dimension

VC-Dimension – Examples

Example (threshold functions over $\mathbb{R}$, see Eq. (4.1)):
We know that

- $\mathcal{H}$ shatters an arbitrary set of size 1, *e.g.*, $C = \{c_1\}$
  Consequently, $\text{VCdim}(\mathcal{H}) \geq 1$
- $\mathcal{H}$ does not shatter an arbitrary set $C = \{c_1, c_2\}$ with $c_1 \leq c_2$
  Consequently, $\text{VCdim}(\mathcal{H}) < 2$

$$\boxed{\Rightarrow \text{VCdim}(\mathcal{H}) = 1}$$

VC-Dimension

VC-Dimension – Examples

Example (hypotheses are intervals over $\mathbb{R}$):

$$\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{R}, a < b\}, \quad h_{a,b}(x) = 1_{a \le x \le b}(x) \tag{4.2}$$

Let, $C_1 = \{c_1\}$ be an arbitrary set of size 1.

- $a = c_1 - 1$, $b = c_1 + 1$ allows to realize 1
- $a = c_1 + 1$, $b = c_1 + 2$ allows to realize 0

Hence, $\mathcal{H}$ shatters C_1 and we conclude $\text{VCdim}(\mathcal{H}) \geq 1$

Let $C_2 = \{c_1, c_2\}$ with $c_1 \leq c_2$ w.l.o.g.

- $a = c_1 - 1, b = c_2 + 1$ realizes $(1, 1)$
- $a = c_2 + 1, b = c_2 + 2$ realizes $(0, 0)$
- $a = c_1 - 1, b = c_1 + (c_2 - c_1)/2$ realizes $(1, 0)$
- $a = c_1 + (c_2 - c_1)/2, b = c_2 + 1$ realizes $(0, 1)$

Hence, $\mathcal{H}$ shatters C_2 and we conclude $\text{VCdim}(\mathcal{H}) \geq 2$

VC-Dimension

VC-Dimension – Examples

Now, let $C_3 = \{c_1, c_2, c_3\}$, $c_1 \leq c_2 \leq c_3$ w.l.o.g.

- $2^3 = 8$ possible labelings
- $(1, 0, 1)$ cannot be realized with $\mathcal{H}$

  Hence, $\mathcal{H}$ does not shatter C_3 and $\text{VCdim}(\mathcal{H}) < 3$

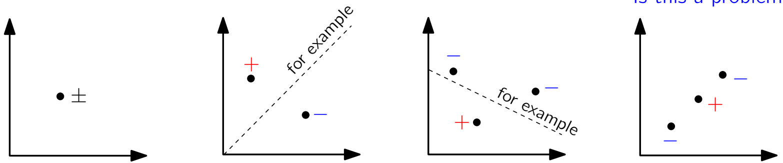
$$\boxed{\Rightarrow \text{VCdim}(\mathcal{H}) = 2}$$

VC-Dimension

VC-Dimension – **Non-homogenous** halfspaces in $\mathbb{R}^d$

$$\boxed{\mathcal{X} = \mathbb{R}^d, \quad \mathcal{H} = \{\mathbf{x} \mapsto \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}}$$

As an introduction, look at the case $d = 2$, i.e., $\mathbb{R}^2$:



Is this a problem?

no problem! $\Rightarrow \text{VCdim}(\mathcal{H}) \geq 1$

no problem! $\Rightarrow \text{VCdim}(\mathcal{H}) \geq 2$

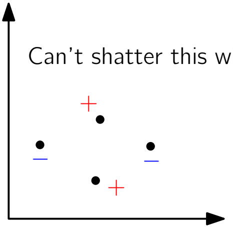
no problem! $\Rightarrow \text{VCdim}(\mathcal{H}) \geq 3$

The case on the right is **no problem** for $\text{VCdim}(\mathcal{H}) \geq 3$, since we only need to find one set that can be shattered (for the lower bound)!

VC-Dimension

VC-Dimension – **Non-homogenous** halfspaces in $\mathbb{R}^d$

Now, what about sets with 4 points?



$\Rightarrow$ VCdim$(\mathcal{H}) < 4$

While (visually showing)

- VCdim$(\mathcal{H}) \geq 3$
- VCdim$(\mathcal{H}) < 4$

is not a formal proof, it "suggests":

$$\boxed{\Rightarrow \text{VCdim}(\mathcal{H}) = 3}$$

VC-Dimension

VC-Dimension – **Non-homogenous** halfspaces in $\mathbb{R}^d$

For general d, we first show the **lower bound**: Lets set

$$\forall i \in [d] : \mathbf{e}_i = \underbrace{[0, \ldots, 1, \ldots, 0]}_{\text{1 at i-th position}} \in \mathbb{R}^d$$

Claim 4.1

$\mathbf{0}, \mathbf{e}_1, \ldots, \mathbf{e}_d$ can be shattered by $\mathcal{H}$.

Proof. For any arbitrary labeling $y_0, \ldots, y_d$, set $b = y_0$ and $\mathbf{w} = [y_1 - b, \ldots, y_d - b]$ with $y_i \in \{-1, +1\}$. We get

- $\text{sign}(\langle \mathbf{0}, \mathbf{w} \rangle + b) = \text{sign}(b) = \text{sign}(y_0) = y_0$ ✓
- $\text{sign}(\langle \mathbf{e}_i, \mathbf{w} \rangle + b) = \text{sign}(y_i - b + b) = \text{sign}(y_i) = y_i$ ✓

□

VC-Dimension

VC-Dimension – **Non-homogenous** halfspaces in $\mathbb{R}^d$

For the **upper bound**, we first state:

Lemma 4.2: Radon's lemma

Let $X \subset \mathbb{R}^d$ be a set of size $d + 2$. Then there exist two disjoint subsets X_1, X_2 of X such that $convh(X_1) \cap convh(X_2) \neq \emptyset$ where $convh(\cdot)$ denotes the convex hull.

Let $X = \mathbf{x}_1, \ldots, \mathbf{x}_{d+2}$ and X_1, X_2 be two disjoint subsets of X. Label

- all points in X_1 with $+1$
- all points in X_2 with -1

From Randon's lemma, the intersection is nonempty. Hence, a halfspace can never be correct in the **intersection** $\Rightarrow \text{VCdim}(\mathcal{H}) < d + 2$

$$\boxed{\Rightarrow \text{VCdim}(\mathcal{H}) = d + 1}$$

VC-Dimension

Finite hypothesis classes

Lets consider the case of finite $\mathcal{H}$. For any set C, restricting $\mathcal{H}$ to C implies that

$$|\mathcal{H}_C| \leq |\mathcal{H}|$$

We know that there are $2^{|C|}$ possible functions from C to $\{0,1\}$. So, if

$$|\mathcal{H}| < 2^{|C|}$$

C cannot be shattered (since, $\mathcal{H}_C$ does not contain all functions from C to $\{0,1\}$, see Definition 4.2). In other words, **we need**

$$\log_2(|\mathcal{H}|) \geq |C|$$

so that we **can shatter** C. This implies

$$\boxed{\log_2(|\mathcal{H}|) \geq \mathrm{VCdim}(\mathcal{H})}$$

VC-Dimension

Finite hypothesis classes

Since, $|\mathcal{H}|$ is finite, VCdim($\mathcal{H}$) is finite and we will see that:

Proposition

PAC learnability of finite $\mathcal{H}$ follows from PAC learnability of hypothesis classes with finite VC-dimension.

(In fact, we have already shown that finite hypothesis classes are PAC learnable before, but this result is more general)

VC-Dimension

Fundamental theorem of statistical learning

Theorem 4.2: Fundamental theorem of statistical learning

Let $\mathcal{H}$ be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0, 1\}$ and let the loss function be the $0 - 1$ loss ℓ_{0-1}. Then the following are equivalent:

1. *$\mathcal{H}$ has the uniform convergence property*
2. *Any ERM rule is a successful agnostic PAC learner for $\mathcal{H}$*
3. *$\mathcal{H}$ is agnostic PAC learnable*
4. *$\mathcal{H}$ is PAC learnable*
5. *Any ERM rule is a successful PAC learner for $\mathcal{H}$*
6. *$\mathcal{H}$ has finite VC-dimension*

VC-Dimension

Fundamental theorem of statistical learning – Quantitative (without proofs)

Assume that $\mathrm{VCdim}(\mathcal{H}) = d < \infty$. Then, there are absolute constants C_1, C_2, such that

1. $\mathcal{H}$ has the uniform convergence property with

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2} \tag{4.3}$$

2. $\mathcal{H}$ is agnostically PAC learnable with

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2} \tag{4.4}$$

3. $\mathcal{H}$ is PAC learnable with

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon} \tag{4.5}$$

VC-Dimension

Fundamental theorem of statistical learning (contd.)

Lets take a closer look at $6 \rightarrow 1$: To do so, we need some definitions and theorems (without proof).

Definition 4.4: Growth function $\tau_{\mathcal{H}}$

Let $\mathcal{H}$ be a hypothesis class. Then the growth function of $\mathcal{H}$, denoted by $\tau_{\mathcal{H}} : \mathbb{N} \rightarrow \mathbb{N}$, is defined as

$$\tau_{\mathcal{H}}(m) = \max_{C \subset \mathcal{X} : |C| = m} |\mathcal{H}_C|$$

Interpretation:

- $|\mathcal{H}_C|$ (number of hypothesis when restricting $\mathcal{H}$ to C)
- $\max_{C \subset \mathcal{X} : |C| = m}$ (max. taken over all subsets of size m)

So, $\tau_{\mathcal{H}}(m) \equiv$ number of functions from C (of size m) to $\{0, 1\}$ w.r.t. $\mathcal{H}_C$.

VC-Dimension

Fundamental theorem of statistical learning (contd.)

Lemma 4.3: (Sauer-Shelah-Perles, aka "Sauer's lemma")

Let $\mathcal{H}$ be a hypothesis class with VCdim$(\mathcal{H}) \leq d < \infty$. Then,

$$\forall m \in \mathbb{N} : \tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}$$

and, in particular, if $m > d + 1$, then

$$\tau_{\mathcal{H}}(m) \leq \left(\frac{em}{d}\right)^{d} \quad .$$

Interpretation: If m becomes larger than the VC-dimension of $\mathcal{H}$, $\tau_{\mathcal{H}}$ increases polynomially, not exponentially, with m.

VC-Dimension

Fundamental theorem of statistical learning (contd.)

Theorem 4.3

Let $\mathcal{H}$ be a hypothesis class and $\tau_{\mathcal{H}}$ be its growth function. Then,

- *for every $\mathcal{D}$*
- *and every $\delta \in (0, 1)$,*

with probability $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$, we have

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \frac{4 + \sqrt{\log(\tau_{\mathcal{H}}(2m))}}{\delta\sqrt{2m}}$$

(without proof)

VC-Dimension

Fundamental theorem of statistical learning (contd.)

With these tools in hand, we are ready to show 6 → 1, namely that finite VC-dimension of $\mathcal{H}$ guarantees that $\mathcal{H}$ has uniform convergence property!

Proof. Let us start with

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \frac{4 + \sqrt{\log(\tau_{\mathcal{H}}(2m))}}{\delta\sqrt{2m}} \quad \text{(from Theorem 4.3)}$$

and apply Sauer's lemma for the case $d > m$. We get

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \frac{4 + \sqrt{d\log(2em/d)}}{\delta\sqrt{2m}}$$

For simplicity, we can assume $\sqrt{d\log(2em/d)} \geq 4$. Why? This guarantees the inequality to hold, since we want to get rid of the 4 (which is then accounted for within the square root).

VC-Dimension

Fundamental theorem of statistical learning (contd.)

This yields

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \frac{1}{\delta}\sqrt{\frac{2d\log(2em/d)}{m}}$$

The question now is, what m needs to be for the left-hand size to be at most ϵ, *i.e.*,

$$\begin{aligned}
\frac{1}{\delta}\sqrt{\frac{2d\log(2em/d)}{m}} &\leq \epsilon \\
\frac{2d\log(2em/d)}{m} &\leq (\epsilon\delta)^2 \\
\frac{m}{2d\log(2em/d)} &\geq \frac{1}{(\epsilon\delta)^2} \\
m &\geq \frac{2d\log(m)}{(\epsilon\delta)^2} + \frac{2d\log(2e/d)}{(\epsilon\delta)^2}
\end{aligned}$$

VC-Dimension

Fundamental theorem of statistical learning (contd.)

We need the following lemma (without proof), to establish when the preceding inequality holds:

Lemma 4.4

Let $a \geq 1$ and $b > 0$. Then: $x \geq 4a\log(2a)+2b \Rightarrow x \geq a\log(x)+b$.

By virtue of this lemma, we get

$$m \geq \underbrace{\frac{2d}{(\epsilon\delta)^2}\log(m)}_{\equiv a\log(x)} + \underbrace{\frac{2d}{(\epsilon\delta)^2}\log(2e/d)}_{\equiv b}$$

$$\Rightarrow \boxed{m \geq 4\frac{2d}{(\epsilon\delta)^2}\log\left(\frac{4d}{(\epsilon\delta)^2}\right) + \frac{4d}{(\epsilon\delta)^2}\log(2e/d)}$$

VC-Dimension

Fundamental theorem of statistical learning (contd.)

This, obviously, guarantees that

$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq 4\frac{16d}{(\epsilon\delta)^2}\log\left(\frac{16d}{(\epsilon\delta)^2}\right) + \frac{16d}{(\epsilon\delta)^2}\log(2e/d)$$

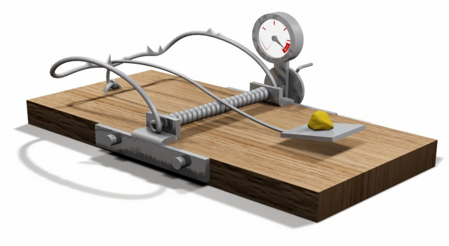
i.e., an upper bound on $m_{\mathcal{H}}^{UC}(\epsilon, \delta)$. □

Note that a lower bound can be derived as well; also

- tighter bounds are available, specialized to
  - agnostic PAC learners
  - (basic) PAC learners (*i.e.*, the realizable case)

No-Free-Lunch / Bias-Complexity Trade-off

(*cf.* [4, Chapter 5])



No-Free-Lunch / Bias-Complexity Trade-off

Motivation

We agree that introducing prior knowledge, *e.g.*, in the form of certain hypothesis classes, is essential to learning, right?

Do **universal learner's** exist? (learners which can be challenged by any task and, without introducing prior knowledge, achieve small $L_{\mathcal{D}}(h)$?)

Summary of the "No-Free-Lunch" theorem

No such universal learner exists.

Note: "without prior knowledge" in fact means consider all functions from some domain to $\{0, 1\}$.

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem

Theorem 5.1

Let A be any learning algorithm for the task of binary classification w.r.t. ℓ_{0-1} over a domain $\mathcal{X}$. Let m be any number of samples $< |\mathcal{X}|/2$. Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:

1. $\exists f : \mathcal{X} \to \{0,1\}$ *with* $L_{\mathcal{D}}(f) = 0$
2. *With probability* $\geq 1/7$ *over the choice of* $S \sim \mathcal{D}^m$ *we have*

$$L_{\mathcal{D}}(A(S)) \geq 1/8$$

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Interpretation

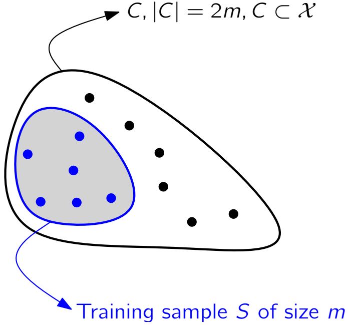
What does this theorem really tell us?

- For every learner, there exists a task on which it fails
- However, that same task can be successfully learned by another learner

  (*e.g.*, a learner with $\mathcal{H} = \{f\}$)

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof strategy


$C, |C| = 2m, C \subset \mathcal{X}$
Training sample S of size m

The idea is to show that by observing only half the instances in C we **cannot** tell anything about the unseen instances.

All under the assumption of no prior knowledge!

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

In the following we proof Theorem 5.1 (rather technical, but instructive :)

Proof. Let

- $C \subset \mathcal{X}$, with $|C| = 2m$
- $f_1, \dots, f_T$ denote <u>all</u> $T = 2^{2^m}$ functions from $C \rightarrow \{0,1\}$
- $\mathcal{D}_i$ denote distributions over $C \times \{0,1\}$

For each f_i, let $\mathcal{D}_i$ be such that

$$\mathcal{D}_i(\{(x,y)\}) = \begin{cases} 1/|C| & \text{if } y = f_i(x) \\ 0 & \text{otherwise} \end{cases}$$

In other words, the probability of (x, y) is $1/|C|$ if f_i in fact outputs the correct label given x.

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

Observation:

$$\forall i : L_{\mathcal{D}_i}(f_i) = 0 \quad \text{(by construction)}$$

We show that for every algorithm A that receives a sample S of size m from $C \times \{0,1\}$, $A(S) : C \rightarrow \{0,1\}$, it holds that

$$\max_{i \in [T]} \mathbb{E}_{S \sim \mathcal{D}_i^m}[L_{\mathcal{D}_i}(A(S))] \geq 1/4$$

What does that mean? It means that for every A' receiving a sample S of size m from $\mathcal{X} \times \{0,1\}$, there exists a function $f : \mathcal{X} \rightarrow \{0,1\}$ and a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, such that

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A'(S))] \geq 1/4$$

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

We have $k = (2m)^m$ sequences $S_1, \dots, S_k$ of m examples from C.

Let

$$\begin{array}{ll} S_1^1 & \dots\ S_1 \text{ labeled by } f_1 \\ S_1^2 & \dots\ S_1 \text{ labeled by } f_2 \\ \vdots & \\ S_j^i & \dots\ S_j \text{ labeled by } f_i \\ \vdots & \end{array}$$

i.e., $S_j^i = ((x_1, f_i(x_1)), \dots, (x_m, f_i(x_m)))$.

$\Rightarrow$ Possible training sets under $\mathcal{D}_i$: $S_1^i, \dots, S_k^i$ (with equal probability)

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

Next, we are going to look for a first lower-bound on the max. of the expected generalization error over all $i \in [T]$ w.r.t. $\mathcal{D}_i$, *i.e.*,

$$\begin{aligned}
\max_{i\in[T]} \mathbb{E}_{S\sim\mathcal{D}_i^m}[L_{\mathcal{D}_i}(A(S))] &= \max_{i\in[T]} \frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}[A(S_j^i)] \\
&\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}[A(S_j^i)] \\
&= \frac{1}{k}\sum_{j=1}^{k}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}[A(S_j^i)] \\
&\geq \min_{j\in[k]} \frac{1}{T}\sum_{j=1}^{k} L_{\mathcal{D}_i}[A(S_j^i)]
\end{aligned}$$

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

Fix $j \in [k]$ and $S_j = (x_1, \ldots, x_m)$. However, $|C| = 2m$; hence, the samples that remain unseen are $v_1, \ldots, v_p$ with $p = m$ (or, in general $p \geq m$).

This leads to the following inequality chain $\forall i \in [T]$ and $h : C \to \{0, 1\}$:

$$\begin{aligned} L_{\mathcal{D}_i}(h) &= \frac{1}{2m} \sum_{x \in C} 1_{h(x) \neq f_i(x)} \\ &\geq \frac{1}{2m} \sum_{r=1}^{p} 1_{h(v_r) \neq f_i(v_r)} \\ &\geq \frac{1}{2p} \sum_{r=1}^{p} 1_{h(v_r) \neq f_i(v_r)} \end{aligned}$$

Why is there no expectation in the first line? Can be done because we constructed $\mathcal{D}_i$!

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

A combination of the last two inequalities gives us:

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}[A(S_j^i)] &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p} 1_{A(S_j^i)(v_r)\neq f_i(v_r)}\\ &= \frac{1}{2}\frac{1}{p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T} 1_{A(S_j^i)(v_r)\neq f_i(v_r)}\\ &\geq \frac{1}{2}\min_{r\in[p]}\frac{1}{T}\sum_{i=1}^{T} 1_{A(S_j^i)(v_r)\neq f_i(v_r)}\end{aligned}$$

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

Now, fix some $r \in [p]$ and divide $f_1, \dots, f_T$ into $T/2$ disjoint pairs P.

We follow the construction scheme for a pair $(f_i, f_{i'})$

$$\forall c \in C : f_i(c) \neq f_{i'}(c) \iff c = v_r$$

Obviously, f_i and $f_{i'}$ agree on the remaining samples, so they also agree on the training sets, *i.e.*, $S_j^i = S_j^{i'}$ for such a pair.

We get

$$\frac{1}{T} \sum_{i=1}^{T} 1_{A(S_j^i)(v_r) \neq f_i(v_r)} = \frac{1}{T} \sum_{(f_i, f_{i'}) \in P} 1_{A(S_j^i)(v_r) \neq f_i(v_r)} + 1_{A(S_j^{i'})(v_r) \neq f_{i'}(v_r)}$$

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

But, what is

$$1_{A(S_j^i)(v_r) \neq f_i(v_r)} + 1_{A(S_j^{i'})(v_r) \neq f_{i'}(v_r)} = ?$$

The pairs $(f_i, f_{i'})$ are constructed such that the functions disagree if and only if $c = v_r$. So, one term in the sum needs to be 1, the other 0.

$$\Rightarrow 1_{A(S_j^i)(v_r) \neq f_i(v_r)} + 1_{A(S_j^{i'})(v_r) \neq f_{i'}(v_r)} = 1$$

This yields

$$\frac{1}{T} \sum_{i=1}^{T} 1_{A(S_j^i)(v_r) \neq f_i(v_r)} = \frac{1}{2}$$

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Proof

Finally, we again combine these results and get:

$$
\begin{aligned}
\max_{i\in[T]} \ \mathbb{E}_{S\sim\mathcal{D}_i^m}[L_{\mathcal{D}_i}(A(S))] &\geq \min_{j\in[k]} \frac{1}{T}\sum_{j=1}^{k} L_{\mathcal{D}_i}[A(S_j^i)] \\
&\geq \min_{j\in[k]} \frac{1}{2} \min_{r\in[p]} \frac{1}{T}\sum_{i=1}^{T} 1_{A(S_j^i)(v_r)\neq f_i(v_r)} \\
&= \min_{j\in[k]} \frac{1}{2} \min_{r\in[p]} \frac{1}{2} = \frac{1}{4}
\end{aligned}
$$

This concludes our proof. □

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Recap

So, our conclusion is that when we receive only half the samples (here: m) in C, we have no information to infer the labels of the remaining instances!

Corollary 5.1

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.

Here, the set of all functions in fact means **lack of prior knowledge** (in other words, the "universal learner" we imagined in the beginning).

No-Free-Lunch / Bias-Complexity Trade-off

No-Free-Lunch theorem – Exercise

In Exercise sheet D you will have to show that

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A'(S))] \geq 1/4$$

is sufficient such that

$$\mathbb{P}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A'(S)) \geq 1/8] \geq 1/7$$

holds for every learning algorithm A' that receives m samples from $C \times \{0,1\}$. The following lemma is useful here (without proof):

Lemma 5.1

Let Z be a random variable that takes values in $[0,1]$. Assume that $\mathbb{E}[Z] = \mu$. Then, for any $a \in (0,1)$,

$$\mathbb{P}[Z > 1 - a] \geq \frac{\mu - (1-a)}{a}$$

No-Free-Lunch / Bias-Complexity Trade-off

Error decomposition

Prior knowledge $\equiv$ choosing the "right" hypothesis class!

We pay a price for choosing a rich hypothesis class (in fact the richest class of all functions gives us the most problems).

$$L_{\mathcal{D}}(h_S) = \epsilon_{app} + \epsilon_{est} \quad (h_S \ldots \text{ERM hypothesis})$$

with

- **Approximation error** $\epsilon_{app} := \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$
- **Estimation error** $\epsilon_{est} := L_{\mathcal{D}}(h_S) - \epsilon_{app}$

No-Free-Lunch / Bias-Complexity Trade-off

Error decomposition – Approximation *vs.* Estimation error

Approximation error ϵ_{app}

- measures how much risk we have if we restrict ourself to $\mathcal{H}$
- equivalently, it measures how much **inductive bias** we have
- does not depend on sample size $\Rightarrow$ larger $\mathcal{H}$ can lead to less ϵ_{app}

Estimation error ϵ_{est}

- is a result of empirical risk only being an estimate of the true risk
- quality of h depends on the $|S|$ and the complexity of $\mathcal{H}$

  (we know that the complexity will be measured as VCdim($\mathcal{H}$))
- increases logarithmically with $|\mathcal{H}|$ for **finite** $\mathcal{H}$

This trade-off is called the **bias-complexity trade-off**.

Boosting

(*cf.* [4, Chapter 10])

Michael Kearns, Leslie Valiant, Robert Schapire, Yoav Freund (from left to right)

Boosting

Motivation

Boosting aims to address two issues:

Bias-complexity tradeoff

- large $\mathcal{H} \rightarrow$ small approximation, but large estimation error!

  (see our discussion on the bias-complexity trade-off)

Computational complexity

- *e.g.*, ERM for learning halfspaces NP-Hard (under 0-1 loss)!

What is the key idea?

> Amplify the accuracy of weak learners (WL), *i.e.*, “easy-to-learn” hypothesis classes that slightly perform better than random guessing.

Boosting

Motivation

Can we boost efficient weak learners into an efficient strong learner?
(solved in 1990 by grad. student R. Schapire while at MIT)

Our plan:

- Introduce the notion of γ-Weak-Learnability
- Decision stumps as weak learners
- AdaBoost for boosting weak learners
- Simple implementation of AdaBoost (in Python)

Boosting

γ-Weak-Learnability

Definition 6.1: γ-Weak-Learnability

A learning algorithm A is a γ-weak learner for a hypothesis class $\mathcal{H}$ if there exists a function

$$m_{\mathcal{H}} : (0,1) \to \mathbb{N}$$

such that

- *for every $\delta \in (0,1)$,*
- *for every distribution $\mathcal{D}$ over $\mathcal{X}$, and*
- *for every labeling function $f : \mathcal{X} \to \{+1,-1\}$*

– if realizability holds w.r.t. $\mathcal{D}, \mathcal{H}, f$ – running A on a sample of size $m \geq m_{\mathcal{H}}(\delta)$ we have

$$\mathcal{D}^m(\{S : L_{\mathcal{D},f}(A(S)) \leq 1/2 - \gamma\}) \geq 1 - \delta$$

Boosting

γ-Weak-Learnability

Some **observations**:

- very similar to (strong) PAC learning
- (Strong) PAC learning implies finding arbitrarily good classifiers

  setting $\epsilon = 1/2 - \gamma$: if $d = \infty$, then $\mathcal{H}$ not γ-weakly learnable!

  (by virtue of the fundamental theorem of learning)

However, PAC learning ignores computational complexity!

Boosting

"Simple" hypothesis classes

Goal: efficiently implementable algorithm A satisfying γ-weak-learnability!

Here's a possible approach:

- take a simple hypothesis class B (aka the base hypothesis class)
- apply ERM to B (*i.e.*, denoted as ERM_B)
- Requirements (w.r.t. B):
  1. ERM_B efficiently implementable
  2. for all samples S labeled by some $h \in \mathcal{H}$

$$L_{\mathcal{D},h}(ERM_B(S)) \leq 1/2 - \gamma$$

Boosting

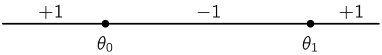
"Simple" hypothesis classes – Example

- $\mathcal{H}$ is the class of 3-piece classifiers, *i.e.*

$$\mathcal{H} = \{h_{\theta_0,\theta_1,b} : \theta_0, \theta_1 \in \mathbb{R}, b \in \{\pm 1\}\}$$

$$h_{\theta_0,\theta_1,b} = \begin{cases} +b, & \text{if } x < \theta_0 \vee x > \theta_1 \\ -b, & \text{if } \theta_0 \leq x \leq \theta_1 \end{cases}$$

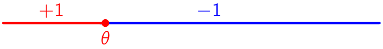
Class $\mathcal{H}$ of 3-piece classifiers



- B is the class of "decision stumps", *i.e.*

$$B = \{x \mapsto \text{sign}(x - \xi) \cdot b : \xi \in \mathbb{R}, b \in \{\pm 1\}\}$$

(Simple) class B of "decision stumps"



Boosting

“Simple” hypothesis classes – Example

Claim 6.1

ERM_B is a γ-weak learner for $\mathcal{H}$ with $\gamma = 1/12$.

Proof. First, we ask the question “what is the generalization error of a ERM_B hypothesis?” In fact, for any $\mathcal{D}$ consistent with $\mathcal{H}$

$$L_{\mathcal{D},f}(h) \leq 1/3$$

with $h \in \mathrm{ERM}_B$ and $f \in \mathcal{H}$. This is because we can choose ξ s.t. the labeling always agrees with (at least) two components.

Next, “What is the VC dimension of B?” We know that $\mathrm{VCdim}(B) = 2$!

Boosting

“Simple” hypothesis classes – Example

The **fundamental theorem of learning** tells us that if the sample size is greater than[14]

$$\Omega(\log(1/\delta)/\epsilon^2)$$

then, with probability $1-\delta$ over the choice of S, running ERM_B on S returns a hypothesis with error at most ${}^1\!/_3 + \epsilon$.

However, we are free to choose ϵ. So we set $\epsilon = {}^1\!/_{12}$ which yields

$$\begin{aligned} L_{\mathcal{D},f}(\text{ERM}_B(S)) &\leq {}^1\!/_3 + {}^1\!/_{12} \\ &= {}^1\!/_2 - {}^1\!/_{12} \end{aligned}$$

which concludes the proof. □

[14] $f(n) \in \Omega(g(n))$ means $f(n)$ is bounded below by $g(n)$.

Boosting

ERM learning for decision stumps over $\mathbb{R}^d$

Let

$$\boxed{\mathcal{H}_{DS} = \{\mathbf{x} \mapsto \text{sign}(\theta - x_i) \cdot b : \theta \in \mathbb{R}, b \in \{\pm 1\}, i \in [d]\}}$$

be the class of decision stumps over $\mathbb{R}^d$ and let S be our training set of size m. Also, let $\mathbf{D}$ be a probability vector, *i.e.*, $\sum_i D_i = 1$.

We will describe a weak learner that receives $(\mathbf{D}, S)$ and outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}$, minimizing

$$L_{\mathbf{D}}(h) = \sum_{i=1}^{m} D_i 1_{h(\mathbf{x}_i) \neq y_i}$$

over $\mathbf{D}$. If $\forall i : D_i = 1/m$, then $L_{\mathbf{D}}(h) = L_S(h)$.

Boosting

ERM learning for decision stumps over $\mathbb{R}^d$

Minimizing the empirical risk w.r.t. **D** (assume $b = 1$) is equivalent to

$$\min_{j\in[d]} \min_{\theta\in\mathbb{R}} \left(\sum_{i:y_i=+1} D_i 1_{x_{i,j}>\theta} + \sum_{i:y_i=-1} D_i 1_{x_{i,j}\leq\theta} \right)$$

Lets fix $j \in [d]$

$$\begin{pmatrix} x_{1,1} & \dots & \boxed{x_{1,j}} & \dots x_{1,d} \\ \vdots & \vdots & \vdots & \\ x_{m,1} & \dots & x_{m,j} & \dots x_{m,d} \end{pmatrix}$$

sorted: $x_{s_1,j} \leq x_{s_2,j} \leq \cdots \leq x_{s_m,j}$

and define

$$\Theta_j = \{(x_{s_i,j} + x_{s_{i+1},j})/2 : i \in [m-1]\} \cup \{x_{s_1,j} - 1\} \cup \{x_{s_m,j} + 1\}$$

Boosting

ERM learning for decision stumps over $\mathbb{R}^d$

Observation: for any $\theta \in \mathbb{R}$ that we choose, we can equivalently find a $\theta' \in \Theta_j$ that yields the same predictions.

Naive algorithm: search over all $j \in [d]$, then over all $\theta' \in \Theta_j$ and compute the sums $\Rightarrow$ $\mathcal{O}(dm^2)$[15].

[15] sorting is done as a preprocessing step; in general, this can be done also in $\mathcal{O}(dm)$

Boosting

AdaBoost – ERM learning for decision stumps over $\mathbb{R}^d$ in Python

Implementation of the stump:

```python
def stumpClassify(X,dim,thr,ineq):
    # set all labels to 1
    ret = ones((shape(X)[0],1))
    if ineq == 'lt':
        ret[X[:,dim] <= thr] = -1.0
    else:
        ret[X[:,dim] > thr] = -1.0
    return ret
```

Boosting

AdaBoost – ERM learning for decision stumps over $\mathbb{R}^d$ in Python

```
def buildStump(dataArr,classLabels,D):
    Y = mat(dataArr) # original data
    X = mat(dataArr) # original data (to be sorted)
    L = mat(classLabels).T # labels

    X.sort(axis=0) # sort along columns
    m,n = X.shape
    # we get m-1 values from the data + 2 additional entries/column
    Theta = zeros((m+1,n))

    # construct \Theta
    for dim in range(n): # dim=0,...,n-1
        Theta[0,dim] = X[0,dim]-1
        for j in range(m-1): #j=0,...,m-1
            Theta[j+1,dim] = (X[j,dim]+X[j+1,dim])/2.0
        Theta[m,dim] = X[m-1,dim]+1

    minError = inf; bestStump = {}; bestClasEst = mat(zeros((m,1)))
```

Boosting

AdaBoost – ERM learning for decision stumps over $\mathbb{R}^d$ in Python

(continued from previous slide ...)

```
        for dim in range(n):
            for j in range(m):
            thr = Theta[j,dim]
            for inequal in ['lt', 'gt']:
                predictedVals = stumpClassify(Y,dim,thr,inequal)
                errArr = mat(ones((m,1)))
                errArr[predictedVals == L] = 0
                weightedError = D.T*errArr

                if weightedError < minError:
                    minError = weightedError
                    bestClasEst = predictedVals.copy()
                    bestStump['dim'] = dim
                    bestStump['thresh'] = thr
                    bestStump['ineq'] = inequal
    return bestStump,minError,bestClasEst
```

Boosting

AdaBoost – Algorithm

Algorithm 6.1: AdaBoost

```
Input: Training set $S$ (size $m$); Weak learner $WL$; #Rounds $T$
Initialization: $\mathbf{D}^{(1)} = (1/m, \ldots, 1/m)$
for $t = 1, \ldots, T$
    ▶ invoke weak learner $h_t = WL(\mathbf{D}^{(t)}, S)$ and store $h_t$
    ▶ compute error $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} 1_{h_t(\mathbf{x}_i) \neq y_i}$
    ▶ compute hypothesis weighting $w_t = {}^1/_2 \log({}^1/_{\epsilon_t} - 1)$
    ▶ update
        $\forall i = 1, \ldots, m : D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_j D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$
Output: $h_S(\mathbf{x}) = \mathrm{sign}(\sum_t w_t h_t(\mathbf{x}))$
```

Boosting

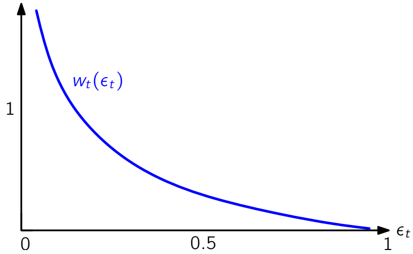
AdaBoost – Algorithm in Python

```python
def adaBoost(dataArr,classLabels,numIt=40):
    weakClassArr = []; m = shape(dataArr)[0]
    D = mat(ones((m,1))/m)
    aggClassEst = mat(zeros((m,1)))
    for i in range(numIt):
        bestStump,error,classEst = buildStump(dataArr,classLabels,D)
        alpha = float(0.5*log((1.0-error)/max(error,1e-16)))
        bestStump['alpha'] = alpha
        weakClassArr.append(bestStump)
        D = multiply(D,exp(multiply(-1*alpha*mat(classLabels).T,classEst)))
        D = D/D.sum()
        aggClassEst += alpha*classEst
        aggErrors = multiply(sign(aggClassEst) != mat(classLabels).T,ones((m,1)))
        errorRate = aggErrors.sum()/m
        if errorRate == 0.0: break
    return weakClassArr,aggClassEst
```

Boosting

AdaBoost – Algorithm

- The weight w_t is inversely proportional to the error



$\Rightarrow$ the update

$$\forall i = 1, \ldots, m : D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_j D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$$

assigns more "probability mass" to samples where h_t erred!

Boosting

AdaBoost – Analysis

Lets state the main theorem first:

Theorem 6.1

Let S be a training set and assume that at each iteration of AdaBoost the weak learner returns a hypothesis for which $\epsilon_t \leq 1/2 - \gamma$. Then, the training error of the output hypothesis h_S is at most

$$L_S(h_S) = \frac{1}{m}\sum_{i=1}^{m} 1_{h_S(\mathbf{x}_i)\neq y_i} \leq \exp(-2\gamma^2 T)$$

Proof. We let

$$\forall t : f_t = \sum_{j\leq t} w_j h_j \quad \text{and} \quad \forall t : Z_t = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f_t(\mathbf{x}_i)}$$

meaning that $\text{sign}(f_T) = h_S$.

Boosting

AdaBoost – Analysis

Part I: Note that, for any hypothesis h, we have

$$1_{h(\mathbf{x}) \neq y} \leq \exp(-yh(\mathbf{x}))$$

since the right-hand side is $\exp(1)$ if $h(\mathbf{x}) \neq y$, hence $1 \leq \exp(1)$ and $0 \leq \exp(0)$ otherwise. Consequently,

$$L_S(f_T) \leq Z_T$$

and it suffices to show that $Z_T \leq \exp(-2\gamma^2 T)$ (in Part **II** of the proof).

Boosting

AdaBoost – Analysis

Part II: Note that $Z_0 = 1$, since

$$f_0 = w_0 h_0 \equiv 0$$

So, Z_T can be re-written as

$$Z_T = \frac{Z_T}{Z_{T-1}} \cdot \frac{Z_{T-1}}{Z_{T-2}} \cdots \frac{Z_2}{Z_1} \cdot \frac{Z_1}{Z_0} = \frac{Z_T}{Z_0}$$

and it suffices to show that

$$\frac{Z_{t+1}}{Z_t} \leq \exp(-2\gamma^2)$$

which results from $Z_{t+1} \leq \exp(-2\gamma^2(t+1))$ and $Z_t \leq \exp(-2\gamma^2 t)$.

Boosting

AdaBoost – Analysis

Now, lets take a closer look at the update rule

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_j D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$$

We repeatedly replace $D_i^{(t)}$ to obtain

$$\begin{aligned} D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i)) &= \frac{D_i^{(t-1)} \exp(-w_{t-1} y_i h_{t-1}(\mathbf{x}_i))}{\sum_{k=1}^m D_k^{(t-1)} \exp(-w_{t-1} y_k h_{t-1}(\mathbf{x}_k))} \exp(-w_t y_i h_t(\mathbf{x}_i)) \\ &= \frac{D_i^{(t-1)} \exp\left(-y_i \sum_{j=t-1}^t w_j h_j(\mathbf{x}_i)\right)}{\sum_{k=1}^m D_k^{(t-1)} \exp(-w_{t-1} y_k h_{t-1}(\mathbf{x}_k))} \\ &\vdots \end{aligned}$$

$$\Rightarrow D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_j D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))} = \frac{\exp(-y_i f_t(\mathbf{x}_i))}{\sum_{j=1}^m \exp(-y_j f_t(\mathbf{x}_j))}$$

Boosting

AdaBoost – Analysis

With this result in mind, we have

$$\begin{aligned}\frac{Z_{t+1}}{Z_t} &= \frac{\sum_{i=1}^m e^{-y_i f_{t+1}(\mathbf{x}_i)}}{\sum_{i=1}^m e^{-y_i f_t(\mathbf{x}_i)}} \\ &= \frac{\sum_{i=1}^m e^{-y_i f_t(\mathbf{x}_i)} e^{-y_i w_{t+1} h_{t+1}(\mathbf{x}_i)}}{\sum_{i=1}^m e^{-y_i f_t(\mathbf{x}_i)}} \\ &= \sum_{i=1}^m D_i^{(t+1)} e^{-y_i w_{t+1} h_{t+1}(\mathbf{x}_i)}\end{aligned}$$

Now, lets distinguish between

$$y_i h_{t+1}(\mathbf{x}_i) = +1 \Rightarrow e^{-w_{t+1}} \text{ and } y_i h_{t+1}(\mathbf{x}_i) = -1 \Rightarrow e^{+w_{t+1}}$$

Hence,

$$\frac{Z_{t+1}}{Z_t} = e^{-w_{t+1}} \left[\sum_{i: y_i h_{t+1}(\mathbf{x}_i) = +1}^m D_i^{(t+1)} \right] + e^{+w_{t+1}} \left[\sum_{i: y_i h_{t+1}(\mathbf{x}_i) = -1}^m D_i^{(t+1)} \right]$$

Boosting

AdaBoost – Analysis

This looks familiar, since if we recall

$$\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} 1_{h(\mathbf{x}_t) \neq y_i}$$

So,

$$\begin{aligned}
\frac{Z_{t+1}}{Z_t} &= e^{-w_{t+1}}(1-\epsilon_{t+1}) + e^{+w_{t+1}}\epsilon_{t+1} \\
&= \frac{1}{\sqrt{1/\epsilon_{t+1}-1}}(1-\epsilon_{t+1}) + \sqrt{1/\epsilon_{t+1}-1}\epsilon_{t+1} \quad \text{(by def. of } w_t\text{)} \\
&= \sqrt{\frac{\epsilon_{t+1}}{1-\epsilon_{t+1}}}(1-\epsilon_{t+1}) + \sqrt{\frac{1-\epsilon_{t+1}}{\epsilon_{t+1}}}\epsilon_{t+1}
\end{aligned}$$

Boosting

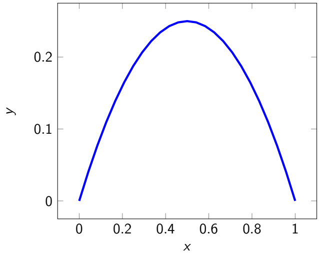
AdaBoost – Analysis

This has the form

$$\sqrt{\frac{a}{1-a}}(1-a)+\sqrt{\frac{1-a}{a}}a=\sqrt{\frac{a}{1-a}(1-a)^2}+\sqrt{\frac{1-a}{a}a^2}$$
$$=2\sqrt{a(1-a)}$$

Note: We assumed $\epsilon_{t+1} \leq 1/2-\gamma$

$f(x) = x(1-x)$ monontonically inc. on [0,1/2]



$$\begin{aligned}2\sqrt{\epsilon_{t+1}(1-\epsilon_{t+1})} &\leq 2\sqrt{(1/2-\gamma)(1/2+\gamma)}\\ &= \sqrt{(1-4\gamma^2)}\\ &\leq e^{-4\gamma^2/2}\\ &= e^{-2\gamma^2}\end{aligned}$$

□

Boosting

AdaBoost – Analysis

The output of AdaBoost will be a hypothesis in

$$L(B,T) = \left\{ \mathbf{x} \mapsto \operatorname{sign}\left(\sum_{t=1}^{T} w_t h_t(\mathbf{x}) \right) : \mathbf{w} \in \mathbb{R}^T, \forall t : h_t \in B \right\} \tag{6.1}$$

The **output** of a $h \in L(B,T)$ is generated by:

1. apply T base hypotheses to generate $\psi(\mathbf{x}) = [h_1(\mathbf{x}), \dots, h_T(\mathbf{x})]$
2. apply homogeneous halfspace $\mathbf{w}$ on $\psi(\mathbf{x})$ via $\mathbf{w}^\top \psi(\mathbf{x})$

We next look at the VC dimension of $L(B,T)$.

Boosting

AdaBoost – VC dimension of $L(B, T)$

Lemma 6.1

Let B be a base hypothesis class and $L(B, T)$ defined as in Eq. (6.1). Assume that $VCdim(B) \geq 3$ and $T \geq 3$. Then,

$$VCdim(L(B, T)) \leq T(VCdim(B) + 1)(3\log(T(VCdim(B) + 1)) + 2)$$

Ignoring constant and logarithmic factors (denoted by $\tilde{\mathcal{O}}$ notation), we get

$$\mathrm{VCdim}(L(B, T)) \in \tilde{\mathcal{O}}(\mathrm{VCdim}(B)T)$$

Proof. Let $d = \mathrm{VCdim}(B)$ and let $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be shattered by $L(B, T)$.

Boosting

AdaBoost – VC dimension of $L(B, T)$

How many ways are there to label C?

For each h_i we know, from Sauer's lemma, that there are

$$(em/d)^d$$

possible labelings. Since we need to select T such hypothesis we obtain

$$(em/d)^{dT}$$

possibilities. Once this is done, we apply a homogeneous halfspace $\mathbf{w} \in \mathbb{R}^T$ on $\psi(\mathbf{x})$. Since homogeneous halfspaces have VC dimension T, this yields

$$(em/d)^{dT}(em/T)^T$$

possible labelings of C induced by $L(B, T)$ in total.

Boosting

AdaBoost – VC dimension of $L(B, T)$

It is straightforward to upper-bound

$$(em/d)^{dT}(em/T)^{T}$$

Note that $e = 2.7183$, so $d, T \geq 3$ guarantees that $(em/d) < m$ and $(em/T) < m$, so

$$(em/d)^{dT}(em/T)^{T} \leq m^{T(d+1)}$$

Also, since C is shattered by $L(B, T)$ we know that the left-hand side of the previous inequality has to be at least 2^m. Hence,

$$2^m \leq m^{T(d+1)} \Rightarrow m \leq \log(m)\frac{T(d+1)}{\log(2)}$$

The question is when does this hold (since we have m on both sides)?

Boosting

AdaBoost – VC dimension of $L(B, T)$

Here's a useful lemma (without proof):

Lemma 6.2

Let $a > 0$. Then: $x \geq 2a\log(a) \Rightarrow x \geq a\log(x)$. It follows that a necessary condition for $x < a\log(x)$ to hold is that $x < 2a\log(a)$.

Invoking Lemma 6.2 on our previous inequality yields

$$
\begin{aligned}
m &\leq 2\frac{(d+1)T}{\log(2)}\log\left[\frac{(d+1)T}{\log(2)}\right] \\
&= (d+1)T\underbrace{\frac{2}{\log(2)}}_{\approx 2.89}\left[\log((d+1)T) - \underbrace{\log(\log(2))}_{\approx -0.36}\right] \\
&\leq (d+1)T(3\log((d+1)T)+2)
\end{aligned}
$$

□

Boosting

AdaBoost – Summary

So, what have we shown?

- VC dimension of $L(B, T)$ is bounded by T times $\text{VCdim}(B)$

  (up to constant and logarithmic factors)
- $\Rightarrow$ Estimation error of AdaBoost grows linearly with T
- $\Rightarrow$ Empirical risk decreases with T (in fact, goes to 0)

> The parameter T of AdaBoost constols the bias-complexity tradeoff!

Non-Uniform Learnability

(*cf.* [4, Chapter 7])

Non-Uniform learnability

Motivation

In all of our previous considerations, we discussed **PAC learnability** with sample complexity depending on

- accuracy parameter ϵ
- confidence parameter δ

and uniform[16] w.r.t. the labeling rule and distribution.

In the following, we relax/weaken this notion of learnability.

[16] *i.e.*, statements about the sample complexity had to hold for all hypotheses.

Non-Uniform learnability

Motivation – PAC learnability *vs.* Non-uniform learnability

Definition 7.1: Non-uniform learnability

A hypothesis class $\mathcal{H}$ is non-uniformly learnable if there exists a learning algorithm A and a function

$$m_{\mathcal{H}}^{NUL} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$$

such that, for

- *every $\epsilon, \delta \in (0,1)$ and*
- *every hypothesis $h \in \mathcal{H}$,*

if $m \geq m_{\mathcal{H}}^{NUL}(\epsilon, \delta, h)$, then for every distribution $\mathcal{D}$ with probability $1-\delta$ over all $S \sim \mathcal{D}^m$, it holds that

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon \ .$$

Non-Uniform learnability

Motivation – PAC learnability *vs.* Non-uniform learnability

- Compare

$$\boxed{m_{\mathcal{H}}^{NUL}: (0,1)^2 \times \mathcal{H} \to \mathbb{N}}$$

  to all of our previous functions of the form $m_{\mathcal{H}}(0,1)^2 \to \mathbb{N}$

- Also, compare

$$\boxed{L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon}$$

  to

$$L_{\mathcal{D},f}(A(S)) \leq \epsilon \qquad \text{PAC learnability (if realizable)}$$

$$L_{\mathcal{D}}(A(S)) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon \qquad \text{agnostic PAC learnability}$$

In other words, we required a output hypothesis to be (ϵ, δ)-competitive with all hypotheses in $\mathcal{H}$.

Non-Uniform learnability

Motivation – PAC learnability *vs.* Non-uniform learnability

Non-uniform learnability strictly relaxes agnostic PAC learnability since

$$L_{\mathcal{D}}(A(S)) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

guarantees

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + \epsilon$$

for all $h \in \mathcal{H}$.

If $\mathcal{H}$ is agnostically PAC learnable it is also non-uniformly learnable.

(we will see an example later)

Non-Uniform learnability

Characterization of non-uniform learnability

Theorem 7.1

A hypothesis class $\mathcal{H}$ of binary classifiers is non-uniformly learnable iff ($\Leftrightarrow$) it is a countable union of agnostic PAC learnable hypothesis classes.

Theorem 7.2

Let $\mathcal{H}$ be a hypothesis class that can be written as a countable union $\bigcup_{n\in\mathbb{N}}\mathcal{H}_n$ of hypothesis classes, where each $\mathcal{H}_n$ enjoys the uniform convergence property. Then, $\mathcal{H}$ is non-uniformly learnable.

Non-Uniform learnability

Characterization of non-uniform learnability – Proof of Theorem 7.1

First, we show the "$\Leftarrow$" part.
Proof. Assume

$$\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n, \quad \text{with each } \mathcal{H}_n \text{ agnostic PAC learnable}$$

By Theorem 4.2, each $\mathcal{H}_n$ has the uniform convergence property and thus $\mathcal{H}$ is non-uniformly learnable (by virtue of Theorem 7.2). □

Second, we show the "$\Rightarrow$" part.
Proof. Assume non-uniform learnability of $\mathcal{H}$ via algorithm A. For every $n \in \mathbb{N}$, let

$$\mathcal{H}_n = \{h \in \mathcal{H} : m_{\mathcal{H}}^{NUL}(1/8, 1/7, h) \leq n\}$$

Clearly, $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$.

Non-Uniform learnability

Characterization of non-uniform learnability – Proof of Theorem 7.1 (contd.)

Recall that if $n \geq m_{\mathcal{H}}^{NUL}(1/8, 1/7, h)$ then

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(h) + 1/8$$

for every distribution $\mathcal{D}$ with probability $6/7$ over the choice of $S \sim \mathcal{D}^n$. Hence, for every $\mathcal{D}$ that satisfies realizability w.r.t. $\mathcal{H}_n$, we have

$$L_{\mathcal{D}}(A(S)) \leq 1/8 \quad (\text{since } L_{\mathcal{D}}(h) = 0)$$

with probability $6/7$ over the choice of $S \sim \mathcal{D}^n \Rightarrow \text{VCdim}(\mathcal{H}_n) < d < \infty$ and thus $\mathcal{H}_n$ is agnostic PAC learnable (by virtue of Theorem 4.2). □

Non-Uniform learnability

Structural risk minimization (SRM)

So far, we introduced prior knowledge by selecting a specific class $\mathcal{H}$.

Structural risk mimimization

Specify preferences over the hypotheses within $\mathcal{H}$. Specifically,

- $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$ (each $\mathcal{H}_n$ has uniform convergence property)
- weigh each $\mathcal{H}_n$ via a weighting function $w : \mathbb{N} \to [0, 1]$

Non-Uniform learnability

Structural risk minimization (SRM)

Let us define

$$\begin{aligned} \epsilon_n &: \mathbb{N} \times (0,1) \to (0,1) \\ \epsilon_n(m,\delta) &= \min\{\epsilon \in (0,1) : m^{UC}_{\mathcal{H}_n}(\epsilon,\delta) \leq m\} \end{aligned} \tag{7.1}$$

to return the lowest upper bound on the gap between generalization and empirical error for a fixed sample size m.

In particular, for every m and $\delta \in (0,1)$, with probability of at least $1-\delta$ over the choice of $S \sim \mathcal{D}^m$, we have

$$\forall h \in \mathcal{H}_n : |L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m,\delta)$$

Non-Uniform learnability

The weight function w

Weight function

The weight function

$$w : \mathbb{N} \to [0,1]$$

reflects the importance (or some measure of complexity) of each $\mathcal{H}_1, \mathcal{H}_2, \dots$ and has to satisfy

$$\sum_{n=1}^{\infty} w(n) \leq 1$$

Examples:

- if $\mathcal{H}$ is a finite union of N classes: $w(n) = {}^{1}/{N}$
- if $\mathcal{H}$ is not finite: $w(n) = 2^{-n}$

Non-Uniform learnability

Theorem 7.3

Let $w : \mathbb{N} \to (0,1)$ *be a function such that* $\sum_{n=1}^{\infty} w(n) \leq 1$ *and let* $\mathcal{H}$ *be a hypothesis class that can be written as*

$$\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n,$$

where each $\mathcal{H}_n$ *has the uniform convergence property with sample complexity* $m^{UC}_{\mathcal{H}_n}$*. Also let* ϵ_n *be defined as in Eq.* (7.1)*. Then, for*

- *every* $\delta \in (0,1)$*, and*
- *every distribution* $\mathcal{D}$*,*

with probability of at least $1 - \delta$ *(over the choice of* $S \sim \mathcal{D}^m$*), the following bound holds for every* $n \in \mathbb{N}$ *and* $h \in \mathcal{H}_n$*:*

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m, w(n) \cdot \delta)$$

Non-Uniform learnability

Proof. If we fix n, by the UC property of each $\mathcal{H}_n$ we have

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}_n : L_{\mathcal{D}}(h) < L_S(h) + \epsilon_n(m, \delta_n)\} \leq \delta_n$$

with $\delta_n = w(n)\delta$. Now, taking the union bound over $n = 1, 2, \ldots$ yields

$$\mathcal{D}^m(\{S : \exists n, h \in \mathcal{H}_n : L_{\mathcal{D}}(h) < L_S(h) + \epsilon_n(m, \delta_n)\} \leq \sum_n \delta_n$$

The fact that

$$\sum_n \delta_n = \delta \underbrace{\sum_n w(n)}_{\leq 1} \leq \delta$$

and thus $1 - \sum_n \delta_n \geq 1 - \delta$ concludes the proof. □

Non-Uniform learnability

SRM as a generic non-uniform learning rule

From Theorem 7.3 we see that for every $\delta \in (0,1)$ and distribution $\mathcal{D}$, with probability $1-\delta$ over the choice of $S \sim \mathcal{D}^m$, we have

$$\forall h \in \mathcal{H}, L_{\mathcal{D}}(h) \leq L_S(h) + \min_{n: h \in \mathcal{H}_n} \epsilon_n(m, w(n) \cdot \delta)$$

With

$$n(h) = \min\{n : h \in \mathcal{H}_n\} \tag{7.2}$$

this implies that

$$\boxed{L_{\mathcal{D}}(h) \leq L_S(h) + \epsilon_{n(h)}(m, w(n(h)) \cdot \delta)}$$

The SRM learning rule searches for h that minimizes this bound!

Non-Uniform learnability

SRM as a generic non-uniform learning rule (contd.)

In pseudocode, the SRM learning rule is:

Structural risk minimization

```
Prior knowledge:
    $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$
    $w : \mathbb{N} \to [0,1]$, where $\sum_n w(n) \leq 1$
Define: $\epsilon_n$ as in Eq. (7.1), $n(h)$ as in Eq. (7.2)
Input: Training set $S \sim \mathcal{D}^m$, confidence $\delta$
Output:
    $h \in \arg\min_{h\in\mathcal{H}} \left[ L_S(h) + \epsilon_{n(h)}(m, w(n(h) \cdot \delta) \right]$
```

Non-Uniform learnability

SRM as a generic non-uniform learning rule (contd.)

Theorem 7.4

Let $\mathcal{H}$ be a hypothesis class such that $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$ where each $\mathcal{H}$ has the uniform convergence property. Further, let $w : \mathbb{N} \to [0,1]$ be the weighting function

$$w(n) = \frac{6}{\pi^2 n^2} \enspace .$$

Then, $\mathcal{H}$ is non-uniformly learnable using the SRM rule with sample complexity

$$m_{\mathcal{H}}^{NUL}(\epsilon, \delta, h) \leq m_{\mathcal{H}_{n(h)}}\left({}^{\epsilon}\!/_{2}, \frac{6\delta}{(\pi n(h))^2} \right) \enspace .$$

Non-Uniform learnability

SRM as a generic non-uniform learning rule (contd.)

Proof. For every ϵ, δ, h let $m \geq m_{\mathcal{H}_{n(h)}}(\epsilon, w(n(h)) \cdot \delta)$. Also, let A be an SRM algorithm.

First, note that

$$\sum_{n=1}^{\infty} w(n) = \sum_{n=1}^{\infty} \frac{6}{(\pi n)^2} = 1$$

hence we can use Theorem 7.3. That is, for every $h' \in \mathcal{H}$, with probability $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$, we have

$$L_{\mathcal{D}}(h') \leq L_S(h') + \epsilon_{n(h')}(m, w(n(h')) \cdot \delta)$$

Non-Uniform learnability

SRM as a generic non-uniform learning rule (contd.)

When we look at the SRM rule, we see that the preceding inequality obviously holds for a hypothesis returned by $A(S)$, *i.e.*,

$$\begin{aligned} L_{\mathcal{D}}(A(S)) &\leq \min_{h' \in \mathcal{H}} \left[L_S(h') + \epsilon_{n(h')}(m, w(n(h')) \cdot \delta) \right] \\ &\leq L_S(h) + \epsilon_{n(h)}(m, w(n(h)) \cdot \delta) \end{aligned}$$

If

$$\underbrace{m \geq m^{UC}_{\mathcal{H}_{n(h)}}(\epsilon/2, w(n(h)) \cdot \delta) \Rightarrow \epsilon_{n(h)}(m, w(n(h)) \cdot \delta) \leq \epsilon/2}_{\text{since } \epsilon_{n(h)} \text{ exactly measures the lowest possible upper bound}}$$

Finally, by the UC property of each $\mathcal{H}_n$, we have $L_S(h) \leq L_{\mathcal{D}}(h) + \epsilon/2 \Rightarrow$

$$\begin{aligned} L_{\mathcal{D}}(A(S)) &\leq L_S(h) + \epsilon/2 \\ &\leq L_{\mathcal{D}}(h) + \epsilon/2 + \epsilon/2 = L_{\mathcal{D}}(h) + \epsilon \end{aligned}$$

□

Non-Uniform learnability

Summary

Note that our last proof also proofs Theorem 7.2.

What price do we pay for non-uniform learnability?

- in agnostic PAC learning, we specialize to one $\mathcal{H}_n$
- in NUL we search the entire $\mathcal{H}$ (*i.e.*, weaker prior knowledge)

We pay in terms of sample complexity!

Convex learning problems

(*cf.* [4, Chapter 12])

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Quick reminder of the gradient of a function $f : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{x} \in \mathcal{X}$, denoted by $\nabla f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_d}(\mathbf{x}) \end{pmatrix}$$

Let f be twice differentiable, then the Hessian of f at $\mathbf{x} \in \mathcal{X}$, denoted by $\nabla^2 f(\mathbf{x})$ is defined by

$$\nabla^2 f(\mathbf{x}) = \left[\frac{\partial^2 f}{\partial x_i, x_j} \right]_{1 \leq i,j \leq d} \in \mathbb{R}^{d \times d}$$

Convex learning problems

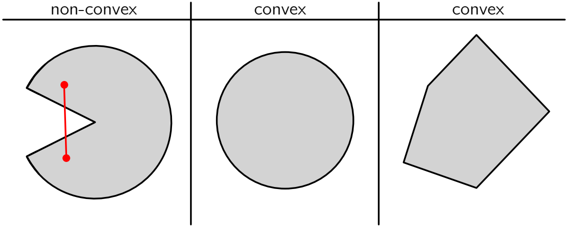
Background

Definition 8.1: Convex set

A set C in a vector space (e.g., $\mathbb{R}^n$) is convex if for any two vectors $\mathbf{u}, \mathbf{v} \in C$, the line segment between $\mathbf{u}$ and $\mathbf{v}$ is contained in C, i.e.,

$$\forall \alpha \in [0,1] : \alpha \mathbf{u} + (1-\alpha)\mathbf{v} \ .$$

Examples:



Convex learning problems

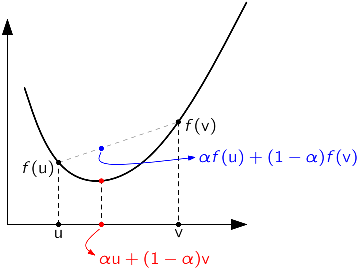
Background – Convexity, Lipschitzness, smoothness

Definition 8.2: Convex function

Let C be a convex set. A function $f : C \to \mathbb{R}$ is convex if for every $\mathbf{u}, \mathbf{v} \in C$ and $\alpha \in [0,1]$,

$$f(\alpha \mathbf{u} + (1-\alpha)\mathbf{v}) \leq \alpha f(\mathbf{u}) + (1-\alpha) f(\mathbf{v})$$

Example (a function $f : \mathbb{R} \to \mathbb{R}$):


$f(\mathrm{v})$
$\alpha f(\mathrm{u}) + (1-\alpha) f(\mathrm{v})$
$f(\mathrm{u})$
u
v
$\alpha \mathrm{u} + (1-\alpha)\mathrm{v}$

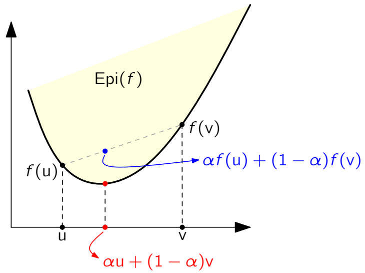
Convex learning problems

Background – Convexity, Lipschitzness, smoothness

The epigraph of a function f is the set

$$\boxed{\text{Epi}(f) = \{(\mathbf{x}, \beta) : f(\mathbf{x}) \leq \beta\}}$$

Visually:



Epi(f)
$f(\text{v})$
$\alpha f(\text{u}) + (1-\alpha) f(\text{v})$
$f(\text{u})$
u
v
$\alpha \text{u} + (1-\alpha)\text{v}$

Note: $f : C \to \mathbb{R}$ is convex $\Leftrightarrow$ Epi(f) is a convex set.

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Let $B(\mathbf{u}, r) = \{\mathbf{v} : \|\mathbf{v} - \mathbf{u}\| \leq r\}$ be a ball of radius r at $\mathbf{u}$. We say $f(\mathbf{u})$ is a local minimum of f at $\mathbf{u}$ if $\exists r > 0,$ s.t. $\forall \mathbf{v} \in B(\mathbf{u}, r) : f(\mathbf{u}) \leq f(\mathbf{v})$.

Claim 8.1

A local minimum of a convex function is a global minimum.

Proof. For any $\mathbf{v}$ there is a small enough $\alpha > 0$ such that $\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u}) \in B(\mathbf{u}, r)$. Hence,

$$f(\mathbf{u}) \leq f(\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u})) \ .$$

By definition of convexity we get

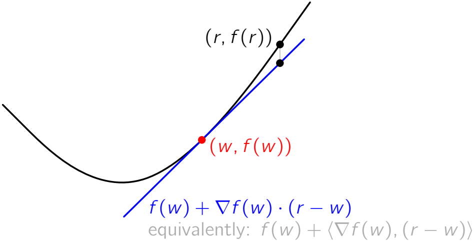
$$f(\mathbf{u} + \alpha(\mathbf{v} - \mathbf{u})) \leq (1 - \alpha) f(\mathbf{u}) + \alpha f(\mathbf{v}).$$

Thus $f(\mathbf{u}) \leq f(\mathbf{v})$ which holds for all $\mathbf{v}$ and the claim follows. □

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

For convex functions f, we can construct a tangent to f at $\mathbf{r}$, such that the tangent lies below f everywhere. Visually, for $f : \mathbb{R} \rightarrow \mathbb{R}$:



$(r, f(r))$
$(w, f(w))$
$f(w) + \nabla f(w) \cdot (r - w)$
equivalently: $f(w) + \langle \nabla f(w), (r - w) \rangle$

In general, for differentiable f, the tangent is the linear function

$$l(\mathbf{r}) = f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{r} - \mathbf{w} \rangle$$

and we have

$$\boxed{\forall \mathbf{r} : f(\mathbf{r}) \geq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{r} - \mathbf{w} \rangle}$$

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

An easy way to check if $f: \mathbb{R} \to \mathbb{R}$ is convex is to leverage that the following statements are equivalent:

1. f is convex
2. f' is monotonically non-decreasing
3. f'' is non-negative

Example: $f: \mathbb{R} \to \mathbb{R}, x \mapsto x^2$. We have $f''(x) = 2 > 0 \Rightarrow$ convex

Lemma 8.1

Let $f: \mathbb{R}^d \to \mathbb{R}$ be such that $f = g(\langle \mathbf{w}, \mathbf{x} \rangle + y)$ for some $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$ and $g: \mathbb{R} \to \mathbb{R}$. Then convexity of g implies convexity of f.

(without proof)

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Definition 8.3: Lipschitzness

Let $C \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over C if for every $\mathbf{w}_1, \mathbf{w}_2 \in C$ we have

$$\|f(\mathbf{w}_1) - f(\mathbf{w}_2)\| \leq \rho \|\mathbf{w}_1 - \mathbf{w}_2\| \ .$$

In fact, Lipschnitzness can be defined w.r.t. to any norm (**here**: always the Euclidean norm in $\mathbb{R}^d$)

Example: $f : \mathbb{R} \to \mathbb{R}, x \mapsto |x|$

$$\begin{aligned} |x_1| - |x_2| &= |x_1 - x_2 + x_2| - |x_2| && \text{(simple expansion)} \\ &\leq |x_1 - x_2| + |x_2| - |x_2| && \text{(by triangle inequality)} \\ &= |x_1 - x_2| \end{aligned}$$

$$\Rightarrow ||x_1| - |x_2|| \leq 1 \cdot |x_1 - x_2| \Rightarrow \text{1-Lipschitz}$$

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Example: $f : \mathbb{R}^d \to \mathbb{R}, \mathbf{w} \mapsto f(\mathbf{w}) = \langle \mathbf{w}, \mathbf{v} \rangle + b$ with $\mathbf{v} \in \mathbb{R}^d$

$$\begin{aligned}
\|f(\mathbf{w}_1) - f(\mathbf{w}_2)\| &= \|\langle \mathbf{w}_1, \mathbf{v} \rangle + b - \langle \mathbf{w}_2, \mathbf{v} \rangle - b\| && \text{(expand)} \\
&= \|\langle \mathbf{w}_1, \mathbf{v} \rangle - \langle \mathbf{w}_2, \mathbf{v} \rangle\| \\
&= \|\langle \mathbf{w}_1 - \mathbf{w}_2, \mathbf{v} \rangle\| \\
&\leq ||\mathbf{v}|| \cdot \|\mathbf{w}_1 - \mathbf{w}_2\| && \text{(by Cauchy-Schwartz)}
\end{aligned}$$

$\Rightarrow \|\mathbf{v}\|$-Lipschitz

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Claim 8.2

Let $f(\mathbf{x}) = g_1(g_2(\mathbf{x})) = g_2 \circ g_1(\mathbf{x})$ where g_1 ρ_1-Lipschitz and g_2 ρ_2-Lipschitz. Then f is $(\rho_1\rho_2)$-Lipschitz.

(proof similar to previous one)

Definition 8.4

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth if its gradient is β-Lipschitz, i.e.,

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^d : \|\nabla f(\mathbf{v}) - \nabla f(\mathbf{w})\| \leq \beta\|\mathbf{v} - \mathbf{w}\| \ .$$

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

In fact, β-smoothness, implies that:

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^d : f(\mathbf{v}) \leq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle + \frac{\beta}{2} \|\mathbf{v} - \mathbf{w}\|^2 \qquad (8.1)$$

Proof. The “gradient theorem“ tells us that

$$f(\mathbf{v}) = f(\mathbf{w}) + \int_0^1 \underbrace{\nabla f(t\mathbf{v} + (1-t)\mathbf{w})^\top (\mathbf{v} - \mathbf{w})}_{\text{note that this is } \langle \cdot, \cdot \rangle} dt$$

Lets re-arrange and subtract $\nabla f(\mathbf{w})^\top (\mathbf{v} - \mathbf{w})$ on both sides

$$\begin{aligned} f(\mathbf{v}) - f(\mathbf{w}) - \nabla f(\mathbf{w})^\top (\mathbf{v} - \mathbf{w}) &= \int_0^1 \left[\nabla f(t\mathbf{v} + (1-t)\mathbf{w}) - \nabla f(\mathbf{w})\right]^\top (\mathbf{v} - \mathbf{w}) dt \\ &= \int_0^1 \langle \nabla f(t\mathbf{v} + (1-t)\mathbf{w}) - \nabla f(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle dt \\ &\leq \|\mathbf{v} - \mathbf{w}\| \int_0^1 \|\nabla f(t\mathbf{v} + (1-t)\mathbf{w}) - \nabla f(\mathbf{w})\| dt \end{aligned}$$

(note that we have used the Cauchy-Schwartz inequality again)

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Lets leverage Def. 8.4 now: set $\mathbf{a} = t\mathbf{v} + (1-t)\mathbf{w}$

$$\begin{aligned}
f(\mathbf{v}) - f(\mathbf{w}) - \nabla f(\mathbf{w})^\top(\mathbf{v}-\mathbf{w}) &\leq \|\mathbf{v}-\mathbf{w}\| \int_0^1 \|\nabla f(\mathbf{a}) - \nabla f(\mathbf{w})\| dt \\
&\leq \beta \|\mathbf{v}-\mathbf{w}\| \int_0^1 \|t(\mathbf{v}-\mathbf{w})\| dt \\
&= \beta \|\mathbf{v}-\mathbf{w}\|^2 \int_0^1 t \; dt \\
&= \frac{\beta}{2} \|\mathbf{v}-\mathbf{w}\|^2
\end{aligned}$$

Upon arranging the terms back, the inequality of Eq. (8.1) follows. □

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Lets summarize these results:

- Convexity of f implied:

$$f(\mathbf{v}) \geq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle$$

- Now, smoothness adds

$$f(\mathbf{v}) \leq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle + \frac{\beta}{2} \|\mathbf{v} - \mathbf{w}\|^2$$

Consequently, we have a **lower** and an **upper** bound on the difference of the function and its 1st order approximation.

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

By substituting $\mathbf{v} = \mathbf{w} - {}^{1}\!/_{2} \nabla f(\mathbf{w})$ into Eq. (8.1), we get

$$\begin{aligned} f(\mathbf{v}) &\leq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), -{}^{1}\!/_{\beta} \nabla f(\mathbf{w}) \rangle + \frac{\beta}{2} \| - {}^{1}\!/_{\beta} \nabla f(\mathbf{w}) \|^2 \\ &= f(\mathbf{w}) - \frac{1}{\beta} \|\nabla f(\mathbf{w})\|^2 + \frac{1}{2\beta} \|\nabla f(\mathbf{w})\|^2 \\ &= f(\mathbf{w}) - \frac{1}{2\beta} \|\nabla f(\mathbf{w})\|^2 \end{aligned}$$

$$f(\mathbf{v}) - f(\mathbf{w}) \leq -\frac{1}{2\beta} \|\nabla f(\mathbf{w})\|^2 \Rightarrow f(\mathbf{w}) - f(\mathbf{v}) \geq \frac{1}{2\beta} \|\nabla f(\mathbf{w})\|^2$$

Now, if $f(\mathbf{v}) \geq 0$, then

$$\boxed{\|\nabla f(\mathbf{w})\|^2 \leq 2\beta f(\mathbf{w})}$$

and we refer to f as being self-bounded.

Convex learning problems

Background – Convexity, Lipschitzness, smoothness

Claim 8.3

Let $f : \mathbb{R}^d \to \mathbb{R}, \mathbf{w} \mapsto f(\mathbf{w}) = g(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ *where* $g : \mathbb{R} \to \mathbb{R}$ *is* β*-smooth,* $\mathbf{x} \in \mathbb{R}^d$ *and* $b \in \mathbb{R}$*. Then* f *is* $(\beta \|\mathbf{x}\|^2)$*-smooth.*

(without proof)

Example: $f(\mathbf{w}) = (\langle \mathbf{w}, \mathbf{v} \rangle - y)^2$. Here $g(x) = x^2$. Since $g'(x) = 2x$, g is 2-smooth $\Rightarrow$ f is $(2\|\mathbf{x}\|^2)$-smooth.

Convex learning problems

Definitions

What was our standard setup so far?

- $Z = \mathcal{X} \times \mathcal{Y}$
- $\mathcal{H} \ldots$ the hypothesis set of functions from $\mathcal{X} \rightarrow \mathcal{Y}$

$\mathcal{H}$ can be any arbitrary set, *e.g.*, subspaces of $\mathbb{R}^d$. In this case, a hypothesis is then associated to a weight vector $\mathbf{w} \in \mathbb{R}^d$.

Convex learning problems

Definitions

Definition 8.5: Convex learning problem

A learning problem $(\mathcal{H}, Z, \ell)$ is called convex if $\mathcal{H}$ is a convex set and for all $z \in Z$, the loss function $\ell(\cdot, z)$ is a convex function.

Example (linear regression with squared loss):

- Objective: model relationship between $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ and $y \in \mathbb{R}$
- $\mathcal{H} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \mathbf{w} \in \mathbb{R}^d\}$, *i.e.*, homogeneous linear functions (functions are parameterized by **w**; we say $\mathcal{H} = \mathbb{R}^d$)
- $\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^2, h \in \mathcal{H} \Rightarrow \ell(\mathbf{w}, (\mathbf{x}, y))$

  $\mathcal{H}$ is a convex set; $\ell(\cdot, z)$ is a convex function w.r.t. 1st argument $\Rightarrow$ convex learning problem!

Convex learning problems

Definitions

Lemma 8.2

If ℓ is a convex loss function and $\mathcal{H}$ is a convex set, then the problem of minimizing the empirical loss over $\mathcal{H}$ is a convex learning problem.

Proof. We need to show that

$$L_S(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell(\mathbf{w}, z_i)$$

with $S = z_1, \ldots, z_m$ is a convex function. Let $g(\mathbf{x}) = \sum_{i=1}^{m} r_i f_i(\mathbf{x})$ with $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ convex and $\forall i : r_i \geq 0$ (in our particular case, $\forall i : r_i = 1$).

$$\begin{aligned} g(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) &= \sum_i r_i f_i(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) \\ &\leq \sum_i r_i \left[\alpha f_i(\mathbf{u}) + (1-\alpha) f_i(\mathbf{v})\right] = \alpha g(\mathbf{u}) + (1-\alpha) g(\mathbf{v}) \end{aligned}$$

□

Convex learning problems

Learnability

From VC theory we know that halfspaces in $\mathbb{R}^d$ are learnable with sample complexity dependent on d.

Are all convex learning problems over $\mathbb{R}^d$ learnable? **No!**[17]

Example:
Linear regression is not PAC learnable (not even for $d = 1$) – Without proof!

[17] no contradiction to VC theory; VC deals with binary classification problems

Convex learning problems

Convex-Lipschitz/smooth-bounded learning problems

Conclusion: Convexity alone is not enough!

Definition 8.6: Convex-Lipschitz bounded

A learning problem $(\mathcal{H}, Z, \ell)$ is called convex-Lipschitz bounded, with parameters ρ, B if:

- *$\mathcal{H}$ is a convex set and $\forall \mathbf{w} \in \mathcal{H} : \|\mathbf{w}\| \leq B$*
- *$\forall z \in Z : \ell(\cdot, z)$ is convex and ρ-Lipschitz*

Example:

- $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq \rho\}$, $\mathcal{Y} = \mathbb{R}$ (*i.e.*, $\mathbf{x}$ in ball of radius ρ)
- $\mathcal{H} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \mathbf{w} \in \mathbb{R}^d, \|\mathbf{w}\| \leq B\}$
- $\ell(h, (\mathbf{x}, y)) = |\langle \mathbf{w}, \mathbf{x} \rangle - y|$ (ρ-Lipschitz, since composition of Lipschitz functions)

  Note that $g(x) = |x|$ is convex (as shown on next slide)!

Convex learning problems

Convex-Lipschitz/smooth-bounded learning problems

Note that $g(x) = \max\{-x, x\}$. Let $f_1(x) = x$ and $f_2(x) = -x$, then $g(x) = \max_{i\in\{1,2\}} f_i(x)$. We show convexity of the latter:

$$\begin{aligned} g(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) &= \max_{i\in\{1,2\}} f_i(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) \\ &\leq \max_{i\in\{1,2\}} [\alpha f_i(\mathbf{u}) + (1-\alpha) f_i(\mathbf{v})] \\ &\leq \alpha \max_{i\in\{1,2\}} f_i(\mathbf{u}) + (1-\alpha) \max_{i\in\{1,2\}} f_i(\mathbf{v}) \\ &= \alpha g(\mathbf{u}) + (1-\alpha) g(\mathbf{v}) \end{aligned}$$

□

We only showed this for $i \in \{1, 2\}$ but the result holds for $f_1, \ldots, f_n$ convex.

Convex learning problems

Convex-Lipschitz/smooth-bounded learning problems

Definition 8.7: Convex smooth-bounded

A learning problem $(\mathcal{H}, Z, \ell)$ *is called convex smooth-bounded, with parameters* β, B *if:*

- $\mathcal{H}$ *is a convex set and* $\forall \mathbf{w} \in \mathcal{H} : \|\mathbf{w}\| \leq B$
- $\forall z \in Z : \ell(\cdot, z)$ *is non-negative*[a]*, convex and* β*-smooth*

[a] ensures self-boundedness

Example: previous example with $\|\mathbf{x}\| \leq \beta/2$

- $\ell(\mathbf{w}, (\mathbf{x}, y)) = (\langle \mathbf{w}, \mathbf{x} \rangle - y)^2$

  (convex-smooth-boundedness is easily seen from Claim 8.3)

Convex learning problems

Surrogate loss functions

What was our natural loss function again for learning halfspaces?

$$\ell_{0-1}(\mathbf{w},(\mathbf{x},y)) = 1_{y \neq \mathrm{sign}(\langle \mathbf{w},\mathbf{x} \rangle)}$$

Problems:

- not convex!
- in fact, learning with the ERM rule w.r.t. 0-1 loss is NP-hard (in the non-realizable/non-separable case)

Idea of surrogate loss functions

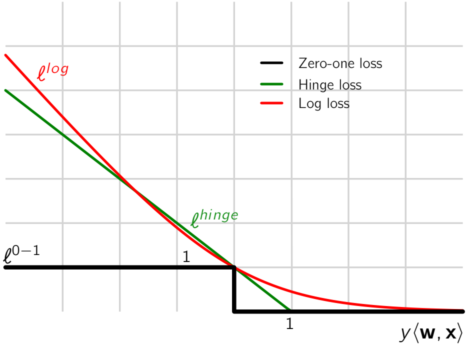
Upper bound the 0-1 loss by a (surrogate) **convex** loss function.

Convex learning problems

Surrogate loss functions

A popular surrogate loss to the 0-1 loss[18] is the hinge loss

$$\boxed{\ell^{hinge}(\mathbf{w},(\mathbf{x},y)) = \max\{0, 1 - y\langle \mathbf{w},\mathbf{x}\rangle\}}$$



[18] in the context of learning halfspaces

Convex learning problems

Surrogate loss functions

The generalization requirement of a hinge-loss learner is

$$\begin{aligned} L_{\mathcal{D}}^{hinge}(A(S)) &\leq \min_{\mathbf{w}\in\mathcal{H}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell^{hinge}(\mathbf{w},(\mathbf{x},y))] + \epsilon \\ &= \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}^{hinge}(\mathbf{w}) + \epsilon \end{aligned}$$

By virtue of the surrogacy property, we have

$$L_{\mathcal{D}}^{0-1}(A(S)) \leq \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}^{hinge}(\mathbf{w}) + \epsilon$$

By expansion (terms in blue), we can write

$$L_{\mathcal{D}}^{0-1}(A(S)) \leq \underbrace{\min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}^{0-1}(\mathbf{w})}_{\text{approx. error}} + \underbrace{\left[\min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}^{hinge}(\mathbf{w}) - \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}^{0-1}(\mathbf{w})\right]}_{\text{optimization error}} + \epsilon$$

Regularization and stability

(*cf*. [4, Chapter 13])

Regularization and stability

Motivation

In our previous discussion(s) ▸ go to frame we (sort of) claimed that

- convex-Lipschitz-bounded, and
- convex-smooth-bounded

problems are learnable.

In the following, we discuss this claim in more detail.

Regularization and stability

Regularized loss minimization

Regularized loss minimization (RLM)

RLM is a learning rule that jointly minimizes the empirical risk and a regularization function $R : \mathbb{R}^d \to \mathbb{R}$. The output of a RLM learning algorithm A, upon inputting a sample S, is

$$A(S) \in \arg\min_{\mathbf{w}} (L_S(\mathbf{w}) + R(\mathbf{w}))$$

Interpretation: The regularization function $R : \mathbb{R}^d \to \mathbb{R}$

- reflects prior knowledge about the problem
- could be interpreted as measuring complexity of a hypothesis
- stabilizes the learning rule[19]

[19] a more crisp definition follows later

Regularization and stability

Tikhonov regularization

A popular and simple regularization function is

$$\boxed{R(\mathbf{w}) = \lambda\|\mathbf{w}\|^2, \quad \text{with } \lambda > 0}$$

commonly referred to as Tikhonov regularization. The RLM rule becomes

$$A(S) \in \arg\min_{\mathbf{w}}(L_S(\mathbf{w}) + \lambda\|\mathbf{w}\|^2)$$

If $\mathcal{H}_i = \{\mathbf{w} : \|\mathbf{w}\|^2 \leq i\}$, we have $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots$, *i.e.*, a nested sequence of hypothesis classes. In this case RLM is similar to SRM (see [4, Chapter 7])

Regularization and stability

Ridge regression as an example of RLM

Ridge regression $\equiv$ linear regression (ℓ^2-loss) + Tikhonov regularization:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\lambda\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^m \frac{1}{2}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2 \right)$$

Let's compute the gradient w.r.t. **w** and set it to zero:

$$-2\lambda\mathbf{w} = \frac{1}{m}\sum_{i=1}^m (\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)\,\mathbf{x}_i$$

$$-2\lambda\mathbf{w}m = \underbrace{-\sum_{i=1}^m y_i\mathbf{x}_i}_{\mathbf{b}} + \underbrace{\left(\sum_{i=1}^m \mathbf{x}_i\mathbf{x}_i^\top\right)}_{\mathbf{A}\text{ (positive semi-definite)}} \mathbf{w}$$

$$\underbrace{(2\lambda m + \mathbf{A})}_{\text{EVs bounded below by }2\lambda m} \mathbf{w} = \mathbf{b} \Rightarrow \boxed{\mathbf{w} = (2\lambda m\mathbf{I} + \mathbf{A})^{-1}\mathbf{b}}$$

Regularization and stability

Ridge regression as an example of RLM

Without having a formal proof yet, we state:

Theorem 9.1

Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times [-1, +1]$, where

$$\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq 1\} \enspace .$$

Further, let

$$\mathcal{H} = \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\| \leq B\}$$

and, for any $\epsilon \in (0, 1)$, let $m \geq 150B^2/\epsilon^2$. Then, applying the ridge regression algorithm with $\lambda = {}^{\epsilon}/_{3B^2}$ satisfies

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A(S))] \leq \min_{\mathbf{w} \in \mathbb{R}^d} L_{\mathcal{D}}(\mathbf{w}) + \epsilon \enspace .$$

Note the expected risk (or generalization error) in the last guarantee!

Regularization and stability

Can we get a PAC-like guarantee from that? (see exercises)

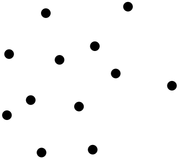
Regularization and stability

Stable rules do not overfit

What do we mean by “overfitting”?

Overfitting $\equiv$ difference between $L_{\mathcal{D}}(A(S))$ and $L_S(A(S))$ is large!

We focus on $\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))]$



$z_1, \ldots, z_m$ with $z_i \sim \mathcal{D}$

What is the analysis principle?

- Sample $S = (z_1, \ldots, z_m)$

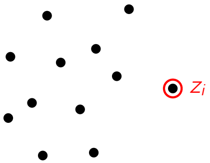
Regularization and stability

Stable rules do not overfit

What do we mean by "overfitting"?

Overfitting $\equiv$ difference between $L_{\mathcal{D}}(A(S))$ and $L_S(A(S))$ is large!

We focus on $\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))]$



$z_1, \ldots, z_m$ with $z_i \sim \mathcal{D}$

What is the analysis principle?

- Sample $S = (z_1, \ldots, z_m)$
- Select a random z_i

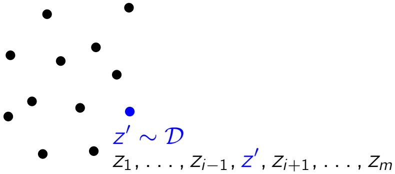
Regularization and stability

Stable rules do not overfit

What do we mean by "overfitting"?

Overfitting $\equiv$ difference between $L_{\mathcal{D}}(A(S))$ and $L_S(A(S))$ is large!

We focus on $\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))]$



What is the analysis principle?

- Sample $S = (z_1, \ldots, z_m)$
- Select a random z_i
- Replace z_i by $z' \sim \mathcal{D} \to S^{(i)}$

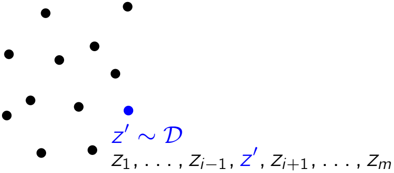
Regularization and stability

Stable rules do not overfit

What do we mean by "overfitting"?

Overfitting $\equiv$ difference between $L_{\mathcal{D}}(A(S))$ and $L_S(A(S))$ is large!

We focus on $\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))]$



What is the analysis principle?

- Sample $S = (z_1, \ldots, z_m)$
- Select a random z_i
- Replace z_i by $z' \sim \mathcal{D} \rightarrow S^{(i)}$

We measure the "effect" of this small change by

$$\underbrace{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}_{\text{if large, then it's likely that we overfit}} \quad \geq 0$$

Regularization and stability

Stable rules do not overfit

Theorem 9.2

Let

- $\mathcal{D}$ *be a distribution*
- $S = (z_1, \ldots, z_m)$ *be an i.i.d. sequence of examples*
- z' *be another sample from* $\mathcal{D}$
- $U(m)$ *be the uniform distribution over* $[m] = \{1, \ldots, m\}$

Then, for any learning algorithm A,

$$\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] = \mathbb{E}_{(S,z')\sim\mathcal{D}^{m+1}, i\sim U(m)}[\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)]$$

Proof.

$$\begin{aligned}\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S))] &= \mathbb{E}_{(S,z')\sim\mathcal{D}^{m+1}}[\ell(A(S), z')] && \text{(by def.)}\\ &= \mathbb{E}_{(S,z')\sim\mathcal{D}^{m+1}}[\ell(A(S^{(i)}), z_i)] && \text{(since, } z_i, z' \text{ i.i.d)}\end{aligned}$$

Regularization and stability

Stable rules do not overfit

We further have

$$\mathbb{E}_{S\sim\mathcal{D}^m}[L_S(A(S))] = \mathbb{E}_{S\sim\mathcal{D}^m, i\sim U(m)}[\ell(A(S), z_i)]$$

Then, by linearity of expectation, Theorem 9.2 follows. □

Definition 9.1: On-Average-Replace-One-Stable

Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be a monotonically decreasing function. We say a learning algorithm A is on-average-replace-one-stable with rate $\epsilon(m)$ if, for every distribution $\mathcal{D}$,

$$\mathbb{E}_{(S,z')\sim\mathcal{D}^{m+1}, i\sim U(m)}[\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)] \leq \epsilon(m)$$

Regularization and stability

Strongly convex functions

Since we will deal with Tikhonov regularization, we introduce a stronger notion of convexity first.

Definition 9.2: Strongly convex functions

A function $f : \mathbb{R}^d \to \mathbb{R}$ is λ-strongly convex if $\forall \mathbf{w}, \mathbf{v} \in \mathbb{R}^d, \alpha > 0$

$$f(\alpha \mathbf{w} + (1-\alpha)\mathbf{v}) \leq \alpha f(\mathbf{w}) + (1-\alpha) f(\mathbf{v}) - \frac{\lambda}{2}\alpha(1-\alpha)\|\mathbf{w}-\mathbf{v}\|^2$$

Example:

- Convex functions are 0-strongly convex (since $\lambda = 0$)

Regularization and stability

Tikhonov regularization as a stabilizer

The following lemma will be useful for our discussions:

Lemma 9.1

Let $f : \mathbb{R}^d \to \mathbb{R}$. Then,

1. $f(\mathbf{w}) = \lambda\|\mathbf{w}\|^2$ *is* 2λ*-strongly convex*
2. *If*
   - f *is* λ*-strongly convex, and*
   - g *is convex,*

   then $f + g$ *is* λ*-strongly convex*
3. *If* f *is* λ*-strongly convex and* $\mathbf{u}$ *is a minimizer of* f*, then*

$$\forall \mathbf{w} : f(\mathbf{w}) - f(\mathbf{u}) \geq \frac{\lambda}{2}\|\mathbf{w} - \mathbf{u}\|^2$$

(proofs on the next slides)

Regularization and stability

Tikhonov regularization as a stabilizer

First, we look at a useful lemma that establishes a relation between λ-strongly convex and convex functions.

Lemma 9.2

A function $f : \mathbb{R}^d \to \mathbb{R}$ is c-strongly convex $\Leftrightarrow$ $f - {}^{c}\!/_{2}\|\mathbf{x}\|^2$ is convex.

Proof. "$\Leftarrow$": Let $f - {}^{\lambda}\!/_{2}\|\mathbf{x}\|^2$ be convex. We have

$$\lambda \left[f(\mathbf{x}) - {}^{c}\!/_{2}\|\mathbf{x}\|^2 \right] + (1-\lambda) \left[f(\mathbf{y}) - {}^{c}\!/_{2}\|\mathbf{y}\|^2 \right] - f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) + \frac{c}{2}\|\lambda\mathbf{x} + (1-\lambda)\mathbf{y}\|^2 \geq 0$$

by definition of convexity. Re-arranging terms yields

$$\begin{aligned} \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) - f(\lambda\mathbf{x} - (1-\lambda)\mathbf{y}) &\geq \frac{c}{2}\left[\lambda\|\mathbf{x}\|^2 + (1-\lambda)\|\mathbf{y}\|^2 - \|\lambda\mathbf{x} + (1-\lambda)\mathbf{y}\|^2\right] \\ &\geq \frac{c}{2}\lambda(1-\lambda)\|\mathbf{x} - \mathbf{y}\|^2 \end{aligned}$$

"$\Leftarrow$" works analogously. □

Regularization and stability

Tikhonov regularization as a stabilizer

We are now ready to proof Lemma 9.1.

Proof.

1. Say we have $g(\mathbf{w}) = {}^{\lambda}\!/_{2}\|\mathbf{w}\|^2$. So, $g(\mathbf{w}) - {}^{\lambda}\!/_{2}\|\mathbf{w}\|^2$ is the null function which is convex. By Lemma 9.2, $g(\mathbf{w})$ is λ-strongly convex.

   Consequently, $f(\mathbf{w}) = \lambda\|\mathbf{w}\|^2$ is 2λ-strongly convex.

□

Regularization and stability

Tikhonov regularization as a stabilizer

Proof.

2. Follows from the definition of convexity and λ-strong convexity. □

Proof.

3. Upon dividing by $\alpha > 0$, we have

$$\frac{f(\mathbf{u}+\alpha(\mathbf{w}-\mathbf{u}))-f(\mathbf{u})}{\alpha} \leq f(\mathbf{w})-f(\mathbf{u})-\frac{\lambda}{2}(1-\alpha)\|\mathbf{w}-\mathbf{u}\|^2$$

Let $\alpha \to 0$: the right-hand side becomes

$$f(\mathbf{w})-f(\mathbf{u})-\frac{\lambda}{2}\|\mathbf{w}-\mathbf{u}\|^2 \ ,$$

and the left-hand side becomes the derivative of f at $\mathbf{u}$, *i.e.*,

$$f'(\mathbf{u}) = \lim_{\alpha\to 0}\frac{f(\mathbf{u}+\alpha(\mathbf{w}-\mathbf{u}))-f(\mathbf{u})}{\alpha} \ .$$

Now, since $\mathbf{u}$ is a minimizer of f, $f'(\mathbf{u}) = 0$ and the claim follows. □

Regularization and stability

Tikhonov regularization as a stabilizer

Next, we discuss our setup to show that RLM learning (using Tikhonov regularization) is stable. Let

- $\mathcal{D}$ be a distribution
- $S = (z_1, \dots, z_m) \sim \mathcal{D}^m$
- $S^{(i)} = (z_1, \dots, z_{i-1}, z', z_{i+1}, \dots, z_m), z' \sim \mathcal{D}$
- $A(S) = \arg\min_{\mathbf{w}}[L_S(\mathbf{w}) + \lambda\|\mathbf{w}\|^2)$
- $f_S(\mathbf{w}) = L_S(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$

Our **1st observation** is:

$$\underbrace{L_S(\mathbf{w})}_{\text{convex}}, \underbrace{\lambda\|\mathbf{w}\|^2}_{\substack{2\lambda\text{-strongly}\\ \text{convex}}} \Rightarrow f_S(\mathbf{w}) \text{ is } 2\lambda\text{-strongly convex}$$

Regularization and stability

Tikhonov regularization as a stabilizer

Our **2nd observation** is:

$$\forall \mathbf{v} : f_S(\mathbf{v}) - f_S(A(S)) \geq \lambda \|\mathbf{v} - A(S)\|^2$$

by Lemma 9.1 (Part 3), since f_S 2λ-strongly convex and $A(S)$ returns the minimizer $\mathbf{w}$ of $L_S(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$.

For arbitrary $\mathbf{v}, \mathbf{u}$, lets expand (and extend) $f_S(\mathbf{v}) - f_S(\mathbf{u})$:

$$\begin{aligned} f_S(\mathbf{v}) - f_S(\mathbf{u}) &= (L_S(\mathbf{v}) + \lambda\|\mathbf{v}\|^2) - (L_S(\mathbf{u}) + \lambda\|\mathbf{u}\|^2) \\ &= (L_{S^{(i)}}(\mathbf{v}) + \lambda\|\mathbf{v}\|^2) - (L_{S^{(i)}}(\mathbf{u}) + \lambda\|\mathbf{u}\|^2) \\ &\quad \underbrace{+ \frac{\ell(\mathbf{v}, z_i) - \ell(\mathbf{u}, z_i)}{m} + \frac{\ell(\mathbf{u}, z') - \ell(\mathbf{v}, z')}{m}}_{\text{subtract loss for } z' \text{ and add loss for } z_i} \end{aligned}$$

Regularization and stability

Tikhonov regularization as a stabilizer

Now, if we let $\mathbf{v} = A(S^{(i)})$, *i.e.*, the output of the RLM rule applied to the sample without z_i but z', and $\mathbf{u} = A(S)$, we get:

$$f_S(A(S^{(i)})) - f_S(A(S)) \leq \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{m} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{m}$$

Together with our 2nd observation (see previous slide), we have:

$$\lambda \|A(S^{(i)}) - A(S)\|^2 \leq \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{m} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{m}$$

Regularization and stability

Tikhonov regularization as a stabilizer – ρ-Lipschitz loss

Lets focus on loss functions $\ell(\cdot, z_i)$ that are ρ-Lipschitz.

Under such loss functions we get:

$$\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i) \leq \rho\|A(S^{(i)}) - A(S)\|$$
$$\ell(A(S^{(i)}), z') - \ell(A(S), z') \leq \rho\|A(S^{(i)}) - A(S)\|$$

Consequently,

$$\lambda\|A(S^{(i)}) - A(S)\|^2 \leq \frac{2\rho\|A(S^{(i)}) - A(S)\|}{m} \Rightarrow \|A(S^{(i)}) - A(S)\| \leq \frac{2\rho}{m\lambda}$$

$$\boxed{\Rightarrow \ell(A(S^{(i)}), z_i) - \ell(A(S), z_i) \leq \frac{2\rho^2}{m\lambda}}$$

Regularization and stability

Tikhonov regularization as a stabilizer – ρ-Lipschitz loss

Corollary 9.1

The RLM learning rule with (Tikhonov) regularization function $\lambda\|\mathbf{w}\|^2$ is on-average-replace-one-stable with rate

$$\epsilon(m) = \frac{2\rho^2}{\lambda m}$$

assuming a convex ρ-Lipschitz loss function.

Taking the expectation w.r.t. $(S, z') \sim \mathcal{D}^{m+1}, i \sim U(m)$, Theorem 9.2 then tells us that

$$\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] \leq \frac{2\rho^2}{m\lambda} \ .$$

Regularization and stability

Fitting *vs.* stability tradeoff

While we do not exercise this here, a similar guarantee for smooth, non-negative loss functions can be obtained.

Upon a simple rephrasing[20] of the expected risk of a learning algorithm

$$\mathbb{E}_S[L_{\mathcal{D}}(A(S))] = \underbrace{\mathbb{E}_S[L_S(A(S))]}_{(a)} + \underbrace{\mathbb{E}_S[L_{\mathcal{D}}(A(S)) - L_S(A(S))]}_{(b)} \tag{9.1}$$

(a) … how well $A(S)$ fits the training data

(b) … how different is the true from the empirical risk $\equiv$ **stability**

(from the previous slide: (b) is equivalent to stability!)

[20] for brevity, $\mathbb{E}_S$ means $\mathbb{E}_{S\sim\mathcal{D}^m}$

Regularization and stability

Fitting *vs*. stability tradeoff

Recall, that the regularization parameter λ controls stability:

- $\lambda \uparrow \equiv$ stability term decreases **but** empirical risk increases
- $\lambda \downarrow \equiv$ stability term increases **but** empirical risk decreases

In summary:

We face a tradeoff between stability *vs*. fitting quality.

Regularization and stability

An "oracle" inequality

Recall that the RLM rule is

$$A(S) = \arg\min_{\mathbf{w}} (L_S(\mathbf{w}) + \lambda \|\mathbf{w}\|^2)$$

Consequently,

$$\begin{aligned} L_S(A(S)) &\leq L_S(A(S)) + \lambda \|A(S)\|^2 \\ &\leq L_S(\mathbf{w}^*) + \lambda \|\mathbf{w}^*\|^2 \end{aligned}$$

for an arbitrary $\mathbf{w}^* \in \mathbb{R}^d$. This yields

$$\begin{aligned} \mathbb{E}_S[L_S(A(S))] &\leq \mathbb{E}_S[L_S(\mathbf{w}^*)] + \lambda \|\mathbf{w}^*\|^2 \\ &= L_{\mathcal{D}}(\mathbf{w}^*) + \lambda \|\mathbf{w}^*\|^2 \end{aligned}$$

$$\boxed{\Rightarrow \mathbb{E}_S[L_{\mathcal{D}}(A(S))] \leq L_{\mathcal{D}}(\mathbf{w}^*) + \lambda \|\mathbf{w}^*\|^2 + \mathbb{E}_S[L_{\mathcal{D}}(A(S)) - L_S(A(S))]}$$

Regularization and stability

An "oracle" inequality

Since, we have an upper bound on $\mathbb{E}_S[L_{\mathcal{D}}(A(S)) - L_S(A(S))]$ from Corollary 9.1, it follows that

$$\forall \mathbf{w}^* : \mathbb{E}_S[L_{\mathcal{D}}(A(S))] \leq L_{\mathcal{D}}(\mathbf{w}^*) + \lambda \|\mathbf{w}^*\|^2 + \frac{2\rho^2}{\lambda m}$$

If $\mathbf{w}^*$ would be a low-risk hypothesis, we could determine the smallest m, s.t., $A(S)$ is at least as good as $\mathbf{w}^*$ (assuming knowledge of $\|\mathbf{w}^*\|$).

In practice: We do not know $\|\mathbf{w}^*\|$, so we tune λ on the basis of a validation set.

Regularization and stability

From "oracle" inequality to a "PAC-like" guarantee

What do we have so far?

- The RLM rule with Tikhonov regularization is 2λ-strongly convex
- This led to on-average-replace-one stability
- We used the stability rate in the "oracle" inequality

We now close the circle, by letting $(\mathcal{H}, Z, \ell)$ be a convex-Lipschitz bounded learning problem with parameters ρ, B. This yields

$$\mathbb{E}_S[L_{\mathcal{D}}(A(S))] \leq \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}(\mathbf{w}) + \lambda B^2 + \frac{2\rho^2}{\lambda m} \enspace ,$$

or (equivalently)

$$\mathbb{E}_S[L_{\mathcal{D}}(A(S))] - \lambda B^2 \leq \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}(\mathbf{w}) + \frac{2\rho^2}{\lambda m} \enspace .$$

Regularization and stability

From "oracle" inequality to a "PAC-like" guarantee

The last inequality definitely holds if λ is such that

$$\frac{2\rho^2}{\lambda m} = \lambda B^2 \Rightarrow \lambda = \sqrt{\frac{2\rho^2}{B^2 m}}$$

Setting λ back in, this yields (on the next slide):

Regularization and stability

From "oracle" inequality to a "PAC-like" guarantee

Corollary 9.2

Let $(\mathcal{H}, Z, \ell)$ be a convex-Lipschitz-bounded learning problem with parameters ρ, B. For any training set size $m > 0$, set

$$\lambda = \sqrt{\frac{2\rho^2}{B^2 m}} \ .$$

Then, the RLM rule with regularization function $\lambda\|\mathbf{w}\|^2$ satisfies

$$\mathbb{E}_S[L_{\mathcal{D}}(A(S))] \leq \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}(\mathbf{w}) + \rho B\sqrt{8/m} \ .$$

In particular, $\forall\epsilon > 0$, if $m \geq {}^{8\rho^2B^2}/_{\epsilon^2}$, then for every distribution $\mathcal{D}$

$$\mathbb{E}_S[L_{\mathcal{D}}(A(S))] \leq \min_{\mathbf{w}\in\mathcal{H}} L_{\mathcal{D}}(\mathbf{w}) + \epsilon$$

Regularized risk minimization

Ridge regression in Python

```
from sklearn.linear_model import Ridge
import numpy as np

n_samples, n_features = 10, 5 # $|S| = 10$ with $\mathbf{x}_i \in \mathbb{R}^5$
np.random.seed(0) # seed random number generator
y = np.random.randn(n_samples) # draw $\mathbf{y}$
X = np.random.randn(n_samples, n_features) # draw $\mathbf{x}_1, \ldots, \mathbf{x}_{10}$

clf = Ridge(alpha=0.5) # configure ridge regression
clf.fit(X, y) # learn
```

Stochastic Gradient Descent (SGD)

(*cf.* [4, Chapter 14])

Stochastic Gradient Descent

A quick recap

Recall that our ultimate goal is to **minimize** the risk

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

However, ...

- this cannot be minimized directly, since it depends on (unknown) $\mathcal{D}$
- so far, we discussed learning methods based on $L_S(h)$[21]

In this chapter, ...

- we focus on convex learning problems (with hypotheses $\mathbf{w} \in \mathbb{R}^d$)
- we introduce a method that allows for "direct minimization" of $L_{\mathcal{D}}(h)$

[21] *e.g.*, ERM, SRM (not discussed in lecture), RLM

Stochastic Gradient Descent

Gradient descent

In our setup, the objective of gradient descent is to minimize a convex, differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, *i.e.*,

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w})$$

Algorithmically, gradient descent

1. starts with $\mathbf{w}^{(0)} = \mathbf{0}$,
2. and then iterates

$$\boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nu \nabla f(\mathbf{w}^{(t)}), \quad \nu > 0}$$

   until convergence (*e.g.*, after T iterations).

3. **Output**: $\overline{\mathbf{w}} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{w}^{(t)}$

Stochastic Gradient Descent

Gradient descent (contd.)

Why is this a good idea?

Lets look at the 1st order Taylor approximation of $f(\mathbf{u})$ at $\mathbf{w}$:

$$f(\mathbf{u}) \approx f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$$

If f is convex, we know that this approximation lower bounds f, *i.e.*,

$$\forall \mathbf{u} \in \mathbb{R}^d : f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$$

- $f(\mathbf{w}) \approx f(\mathbf{w}^{(t)}) + \langle \mathbf{w} - \mathbf{w}^{(t)}, \nabla f(\mathbf{w}^{(t)}) \rangle$ (good, if $\mathbf{w}$ close to $\mathbf{w}^{(t)}$)
  $\Rightarrow$ add "closeness" term: $\|\mathbf{w} - \mathbf{w}^{(t)}\|^2$

This leads to the following **update rule**:

$$\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}^{(t)}\|^2 + \nu \left(f(\mathbf{w}^{(t)}) + \langle \mathbf{w} - \mathbf{w}^{(t)}, \nabla f(\mathbf{w}^{(t)}) \rangle \right)$$

Stochastic Gradient Descent

Gradient descent (contd.)

Lets take the derivative of

$$\frac{1}{2}\|\mathbf{w} - \mathbf{w}^{(t)}\|^2 + \nu \left(f(\mathbf{w}^{(t)}) + \langle \mathbf{w} - \mathbf{w}^{(t)}, \nabla f(\mathbf{w}^{(t)}) \rangle \right)$$

with respect to **w** and set it to 0. We know that

$$\frac{1}{2}\frac{\partial}{\partial \mathbf{w}}\|\mathbf{w} - \mathbf{w}^{(t)}\|^2 = \mathbf{w} - \mathbf{w}^{(t)}$$

and

$$\nu \frac{\partial}{\partial \mathbf{w}} \left(f(\mathbf{w}^{(t)}) + \langle \mathbf{w} - \mathbf{w}^{(t)}, \nabla f(\mathbf{w}^{(t)}) \rangle \right) = \nu \nabla f(\mathbf{w}^{(t)})$$

which, upon solving for **w**, leads to the **update rule**

$$\boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nu \nabla f(\mathbf{w}^{(t)}), \quad \nu > 0}$$

Stochastic Gradient Descent

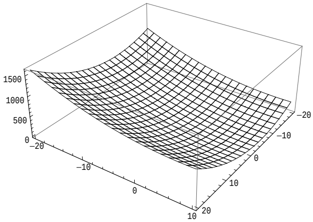
Gradient descent – Example

Lets consider minimizing

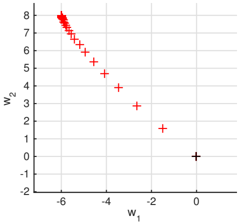
$$f(\mathbf{w}) = 1.25(w_1 + 6)^2 + (w_2 - 8)^2$$

over $\mathbf{w}$. The gradient w.r.t. $\mathbf{w}$ is

$$\nabla f(\mathbf{w}) = \begin{pmatrix} +15 + 2.5w_1 \\ -16 + 2.0w_2 \end{pmatrix}$$



Function $f(\mathbf{w})$



$T = 100$ iterations

Stochastic Gradient Descent

Gradient descent with convex Lipschitz functions

Let $\mathbf{w}^*$ be **any** vector[22] and $\|\mathbf{w}^*\| \leq B$. We have

$$
\begin{aligned}
f(\overline{\mathbf{w}}) - f(\mathbf{w}^*) &= f\left({}^1\!/_T \sum_t \mathbf{w}^{(t)}\right) - f(\mathbf{w}^*) \\
&\leq {}^1\!/_T \sum_t \left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right) \quad \text{(by Jensen's inequality)}
\end{aligned}
$$

Now, remember that (by convexity of f) it holds that $\forall \mathbf{u}$:

$$
\begin{aligned}
f(\mathbf{u}) &\geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle \\
\Rightarrow f(\mathbf{w}) - f(\mathbf{u}) &\leq \langle \mathbf{w} - \mathbf{u}, \nabla f(\mathbf{w}) \rangle
\end{aligned}
$$

and thus

$$
\boxed{f(\overline{\mathbf{w}}) - f(\mathbf{w}^*) \leq {}^1\!/_T \sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \nabla f(\mathbf{w}^{(t)}) \rangle}
$$

[22] not necessarily a minimum of some sort

Stochastic Gradient Descent

Gradient descent with convex Lipschitz-bounded functions

To bound the right-hand side, we use a lemma (without proof):

Lemma 10.1

Let $\mathbf{v}_1, \ldots, \mathbf{v}_T$ *be an arbitrary sequence of vectors. Any algorithm with initialization* $\mathbf{w}^{(0)} = \mathbf{0}$ *and an update rule of the form*

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nu \mathbf{v}_t$$

the following holds: for every $B, \rho > 0$, *if we set* $\nu = \sqrt{B^2/\rho^2 T}$, *then for every* $\mathbf{w}^*$ *with* $\|\mathbf{w}^*\| \leq B$ *we have*

$$1/T \sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t \rangle \leq \frac{B\rho}{\sqrt{T}}$$

under the condition that $\forall t : \|\mathbf{v}_t\| \leq \rho$.

Stochastic Gradient Descent

Gradient descent with convex Lipschitz-bounded functions

Applying this lemma with $\mathbf{v}_t = \nabla f(\mathbf{w}^{(t)})$ we obtain the following corollary:

Corollary 10.1

Let f be convex, ρ-Lipschitz, and $\mathbf{w}^* \in \arg\min_{\mathbf{w}:\|\mathbf{w}\|\leq B} f(\mathbf{w})$. If we run the GD algorithm for T steps with

$$\nu = \sqrt{\frac{B^2}{\rho^2 T}}$$

then the output vector $\overline{\mathbf{w}}$ satisfies

$$f(\overline{\mathbf{w}}) - f(\mathbf{w}^*) \leq \frac{B\rho}{\sqrt{T}}$$

We will see that f ρ-Lipschitz is enough to guarantee $\|\nabla f(\mathbf{w}^{(t)})\| \leq \rho$

Stochastic Gradient Descent

Subgradients (contd.)

To handle non-differentiable functions in gradient descent, we introduce so called subgradients.

Once more, we recall that for convex f, we have

$$\forall \mathbf{u} : f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$$

Lemma 10.2: Alternative characterization of convexity

Let S be an open convex set. A function

$$f : S \to \mathbb{R}$$

is convex $\Leftrightarrow$ for every $\mathbf{w} \in S$ there exists $\mathbf{v}$ such that

$$\forall \mathbf{u} : f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \mathbf{v} \rangle \tag{10.1}$$

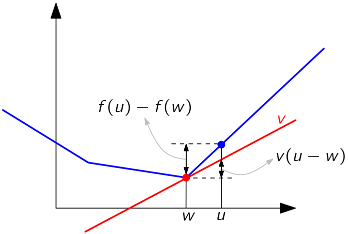
Stochastic Gradient Descent

Subgradients (contd.)

Definition 10.1: Subgradient/Differential set

A vector $\mathbf{v}$ *that satisfies Eq. (10.1) is called a subgradient of* f *at* $\mathbf{w}$*. The set of subgradients at* $\mathbf{w}$ *is called the differential set* $\partial f(\mathbf{w})$*.*

In case $f : \mathbb{R} \to \mathbb{R}$, this means that $f(u) - f(w) \geq v(u - w)$



Stochastic Gradient Descent

Subgradients (contd.)

Claim 10.1

Let $g(\mathbf{w}) = \max_{i\in[r]} g_i(\mathbf{w})$ *for convex differentiable* $g_1, \ldots, g_r$. *Also, let* $j \in \arg\max_i g_i(\mathbf{w})$ *for some* $\mathbf{w}$. *Then,*

$$\nabla g_j(\mathbf{w}) \in \partial g(\mathbf{w}) \enspace .$$

Proof. Since each g_i is convex, we have, in particular,

$$g_j(\mathbf{u}) \geq g_j(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla g_j(\mathbf{w}) \rangle$$

Now, since $g_j = g$ **and** $g(\mathbf{u}) \geq g_j(\mathbf{u})$, we have

$$g(\mathbf{u}) \geq g(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \quad \underbrace{\nabla g_j(\mathbf{w})}_{\text{i.e., a } \mathbf{v}\in\partial g(\mathbf{w})} \quad \rangle$$

□

Stochastic Gradient Descent

Subgradients (Hinge loss)

Based on this claim, it's easy to compute a subgradient of the hinge loss

$$\ell^{hinge}(\mathbf{w}, z) = \max\{0, 1 - y\langle \mathbf{x}, \mathbf{w} \rangle\}$$

Case 1: $1 - y\langle \mathbf{w}, \mathbf{x} \rangle \leq 0$ The max. function returns 0 and the gradient of this constant function w.r.t. $\mathbf{w}$ is $\mathbf{0}$.

Case 2: $1 - y\langle \mathbf{w}, \mathbf{x} \rangle > 0$ The max. function returns $1 - y\langle \mathbf{w}, \mathbf{x} \rangle$ and the gradient w.r.t. $\mathbf{w}$ is $-y\mathbf{x}$.

In summary, we have

$$\mathbf{v} = \begin{cases} \mathbf{0} & \text{if } 1 - y\langle \mathbf{w}, \mathbf{x} \rangle \leq 0 \\ -y\mathbf{x} & \text{if } 1 - y\langle \mathbf{w}, \mathbf{x} \rangle > 0 \end{cases}$$

Stochastic Gradient Descent

Subgradients of Lipschitz functions

Now that we know about subgradients, we come back to the question of whether $\|\nabla f(\mathbf{w})\|$ is bounded if f is ρ-Lipschitz.

Lemma 10.3

Let A be an open convex set and $f : A \to \mathbb{R}$ convex. Then f is ρ-Lipschitz over A $\Leftrightarrow$ $\forall \mathbf{w} \in A$ and $\mathbf{v} \in \partial f(\mathbf{w})$ we have $\|\mathbf{v}\| \leq \rho$.

Proof.

"$\Leftarrow$": We assume that forall $\mathbf{v} \in \partial f(\mathbf{w})$ we have $\|\mathbf{v}\| \leq \rho$. Since $\mathbf{v}$ is an element of the differential set $\partial f(\mathbf{w})$ we have

$$\begin{aligned} f(\mathbf{u}) &\geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \mathbf{v} \rangle \\ \Rightarrow f(\mathbf{w}) - f(\mathbf{u}) &\leq \langle \mathbf{w} - \mathbf{u}, \mathbf{v} \rangle \\ &\leq \|\mathbf{v}\| \|\mathbf{w} - \mathbf{u}\| \\ &\leq \rho \|\mathbf{w} - \mathbf{u}\| \end{aligned}$$

Stochastic Gradient Descent

Subgradients of Lipschitz functions

$f(\mathbf{u}) - f(\mathbf{w}) \leq \rho\|\mathbf{u} - \mathbf{w}\|$ follows analogously. This completes the "$\Leftarrow$" part of the proof.

"$\Rightarrow$": Let f be ρ-Lipschitz. Choose some $\mathbf{w} \in A$ and $\mathbf{v} \in \partial f(\mathbf{w})$.

$$A \text{ open } \Rightarrow \exists \epsilon > 0, \text{ such that } \mathbf{u} = \mathbf{w} + \epsilon \cdot {}^{\mathbf{v}}\!/\!_{\|\mathbf{v}\|} \in A$$

Consequently,

$$\begin{aligned} \langle \mathbf{u} - \mathbf{w}, \mathbf{v} \rangle &= \langle {}^{\epsilon \mathbf{v}}\!/\!_{\|\mathbf{v}\|}, \mathbf{v} \rangle \\ &= \frac{\epsilon}{\|\mathbf{v}\|} \langle \mathbf{v}, \mathbf{v} \rangle \\ &= \frac{\epsilon}{\|\mathbf{v}\|} \|\mathbf{v}\|^2 = \epsilon \|\mathbf{v}\| \end{aligned}$$

Stochastic Gradient Descent

Subgradients of Lipschitz functions

Also,

$$\|\mathbf{u}-\mathbf{w}\| = \epsilon\|\underbrace{\mathbf{v}/\|\mathbf{v}\|}_{1}\| = \epsilon$$

Since $\mathbf{v} \in \partial f(\mathbf{w})$ we have, by definition of subgradients,

$$f(\mathbf{u}) - f(\mathbf{w}) \geq \langle \mathbf{u}-\mathbf{w}, \mathbf{v}\rangle = \epsilon\|\mathbf{v}\|$$

By the Lipschitzness of f, we have

$$f(\mathbf{u}) - f(\mathbf{w}) \leq \rho\|\mathbf{u}-\mathbf{w}\| = \rho\epsilon$$

This establishes the following "chain of inequalities"

$$\epsilon\|\mathbf{v}\| \leq f(\mathbf{u}) - f(\mathbf{w}) \leq \epsilon\rho$$

from which we conclude that $\|\mathbf{v}\| \leq \rho$. □

Stochastic Gradient Descent

Subgradients of Lipschitz functions

We set $\mathbf{v}_t = \nabla f(\mathbf{w}^{(t)})$ to obtain Corollary 10.1. Now, since f is ρ-Lipschitz, any $\mathbf{v} \in \partial f(\mathbf{w})$ is bounded and the Corollary makes sense.

Important: GD works also with subgradients which allows us to handle non-differentiable functions.

Stochastic Gradient Descent

SGD algorithm

We are now ready to state the SGD algorithm.

Algorithm 10.1: SGD for minimization of $f(\mathbf{w})$

```
Input: $\nu > 0$, $T \in \mathbb{N}_+$
Initialization: $\mathbf{w}^{(1)} = \mathbf{0}$
  ▶ Iterate over $t = 1, \ldots, T$
      ▶ choose $\mathbf{v}_t$ at random from a distribution such that
            $\mathbb{E}[\mathbf{v}_t | \mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$   (easy in many cases)
      ▶ update $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nu \mathbf{v}_t$
Output:
            $\overline{\mathbf{w}} = \frac{1}{T} \sum_{t \in [T]} \mathbf{w}^{(t)}$
```

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

Since only the expectation is in the differential set, *i.e.*

$$\mathbb{E}[\mathbf{v}_t|\mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$$

we **cannot** directly use our previous guarantee of Lemma 10.1.

However, we can derive a bound on the **expected** output of the SGD algorithm.

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

Theorem 10.1: SGD for convex-Lipschitz-bounded functions f

Let $B, \rho > 0$ and f be convex. Further, let

$$\mathbf{w}^* \in \arg\min_{\mathbf{w}:\|\mathbf{w}\|\leq B} f(\mathbf{w}) \enspace .$$

Further, assume that $\forall t : \|\mathbf{v}_t\| \leq \rho$ with probability 1 and that the SGD algorithm is run for T iterations with $\nu = \sqrt{B^2/(\rho^2 T)}$. Then,

$$\mathbb{E}[f(\overline{\mathbf{w}})] - f(\mathbf{w}^*) \leq \frac{B\rho}{\sqrt{T}}$$

Hence, to bound the expected output of SGD, for any $\epsilon > 0$, we run the algorithm for $T \geq B^2\rho^2/\epsilon^2$ iterations.

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

Proof. First, note that $\mathbf{w}^*$ is fixed in Theorem 10.1 and we can write

$$\mathbb{E}[f(\overline{\mathbf{w}}) - f(\mathbf{w}^*)] \leq \frac{B\rho}{\sqrt{T}}$$

We take the expectation of a previous result ▸ go to slide

$$f(\overline{\mathbf{w}}) - f(\mathbf{w}^*) \leq {}^1\!/_T \sum_t \left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*) \right)$$

$$\Rightarrow \mathbb{E}_{\mathbf{v}_{1:T}}[f(\overline{\mathbf{w}}) - f(\mathbf{w}^*)] \leq \mathbb{E}_{\mathbf{v}_{1:T}} \left[{}^1\!/_T \sum_t \left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*) \right) \right]$$

We also know that Lemma 10.1 holds for arbitrary $\mathbf{v}_1, \dots, \mathbf{v}_T$, so taking the expectation on the bound gives

$$\mathbb{E}_{\mathbf{v}_{1:T}} \left[{}^1\!/_T \sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t \rangle \right] \leq \frac{B\rho}{\sqrt{T}}$$

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

Hence, **it suffices to show** that

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[{}^1\!/\!{}_T \sum_t \left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right)\right] \leq \mathbb{E}_{\mathbf{v}_{1:T}}\left[{}^1\!/\!{}_T \sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right]$$

Lets focus on the right-hand side term: By linearity of expectation

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[{}^1\!/\!{}_T \sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] = {}^1\!/\!{}_T \sum_t \mathbb{E}_{\mathbf{v}_{1:T}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right]$$

Lets recall the **total law of expectation**: For two random variables X and Y, we have

$$\boxed{\mathbb{E}_X[X] = \mathbb{E}_Y \mathbb{E}_{X|Y}[X|Y]}$$

Now, set $X = \mathbf{v}_{1:t}$ and $Y = \mathbf{v}_{1:t-1}$.

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

We get

$$\begin{aligned}\mathbb{E}_{\mathbf{v}_{1:T}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] &= \mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] \\ &= \mathbb{E}_{\mathbf{v}_{1:t-1}}\mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t|\mathbf{v}_{1:t-1}\rangle\right]\end{aligned}$$

Now, recall our update rule

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nu\mathbf{v}_t \quad .$$

Thus, $\mathbf{w}^{(t)}$ is no longer random once we know $\mathbf{v}_{t-1}$. This yields

$$\mathbb{E}_{\mathbf{v}_{1:t-1}}\mathbb{E}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t|\mathbf{v}_{1:t-1}\rangle\right] = \mathbb{E}_{\mathbf{v}_{1:t-1}}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{v}_{t-1}]\rangle$$

(Note that $\mathbb{E}_{\mathbf{v}_{1:t}}$ reduced to $\mathbb{E}_{\mathbf{v}_t}$)

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

Again, $\mathbf{w}^{(t)}$ only depends on $\mathbf{v}_{1:t-1}$. Also, the SGD algorithm requires

$$\mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$$
$$\Rightarrow \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{v}_{1:t-1}] \in \partial f(\mathbf{w}^{(t)})$$

We repeat (see previous slides):

$$\begin{aligned}\mathbb{E}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] &= \frac{1}{T}\sum_t \mathbb{E}_{\mathbf{v}_{1:T}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] \\ &= \frac{1}{T}\sum_t \mathbb{E}_{\mathbf{v}_{1:t-1}}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{v}_{t-1}]\rangle\end{aligned}$$

Stochastic Gradient Descent

SGD with convex-Lipschitz-bounded functions

Just looking the expectation terms in the sum and remembering that $\mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{v}_{1:t-1}]$ is in the differential set $\partial f(\mathbf{w}^{(t)})$, we have

$$\boxed{\mathbb{E}_{\mathbf{v}_{1:t-1}}\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t|\mathbf{v}_{t-1}]\rangle \geq \mathbb{E}_{\mathbf{v}_{1:t-1}}[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)]}$$

(by the properties of subgradients). In summary

$$\begin{aligned}\mathbb{E}_{\mathbf{v}_{1:T}}\left[\langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right] &\geq \mathbb{E}_{\mathbf{v}_{1:t-1}}[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)] \\ &\geq \mathbb{E}_{\mathbf{v}_{1:T}}[f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)]\end{aligned}$$

By adding back the sum, dividing by T and pulling the expectation to the outside (via linearity), we get:

$$\mathbb{E}_{\mathbf{v}_{1:T}}\left[{}^1/_T \sum_t \left(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)\right)\right] \leq \mathbb{E}_{\mathbf{v}_{1:T}}\left[{}^1/_T \sum_t \langle \mathbf{w}^{(t)} - \mathbf{w}^*, \mathbf{v}_t\rangle\right]$$

□

Stochastic Gradient Descent

SGD for learning problems

Recall that our ultimate goal is to **minimize**

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

but we don't know $\mathcal{D}$.

However, to use SGD we only need a random vector whose conditional expected value is $\nabla L_{\mathcal{D}}(\mathbf{w}^{(t)})$.

Here's the idea: (1) sample $z \sim \mathcal{D}$ and (2) let $\mathbf{v}_t$ be $\nabla_{\mathbf{w}}\ell(\mathbf{w}^{(t)}, z)$

$$\mathbb{E}[\mathbf{v}_t|\mathbf{w}^{(t)}] = \mathbb{E}_{z\sim\mathcal{D}}[\nabla\ell(\mathbf{w}^{(t)}, z)] = \nabla \underbrace{\mathbb{E}_{z\sim\mathcal{D}}[\ell(\mathbf{w}^{(t)}, z)]}_{L_{\mathcal{D}}(\mathbf{w}^{(t)})} = \nabla L_{\mathcal{D}}(\mathbf{w}^{(t)})$$

Stochastic Gradient Descent

SGD for learning problems

What about non-differentiable losses? Works the same way!

Let $\mathbf{v}_t$ be a subgradient of $\ell(\mathbf{w}, z)$ at $\mathbf{w}^{(t)}$.

$$\forall \mathbf{u} : \ell(\mathbf{u}, z) \geq \ell(\mathbf{w}^{(t)}, z) + \langle \mathbf{u} - \mathbf{w}^{(t)}, \mathbf{v}_t \rangle$$

Consequently,

$$\begin{aligned} \ell(\mathbf{u}, z) - \ell(\mathbf{w}^{(t)}, z) &\geq \langle \mathbf{u} - \mathbf{w}^{(t)}, \mathbf{v}_t \rangle & \qquad \mid \mathbb{E}_{z\sim\mathcal{D}}[\cdot|\mathbf{w}^{(t)}] \\ \Rightarrow L_{\mathcal{D}}(\mathbf{u}) - L_{\mathcal{D}}(\mathbf{w}^{(t)}) &= \mathbb{E}_{z\sim\mathcal{D}}[\ell(\mathbf{u}, z) - \ell(\mathbf{w}^{(t)}, z)|\mathbf{w}^{(t)}] \\ &\geq \mathbb{E}_{z\sim\mathcal{D}}[\langle \mathbf{u} - \mathbf{w}^{(t)}, \mathbf{v}_t \rangle|\mathbf{w}^{(t)}] \\ &= \langle \mathbf{u} - \mathbf{w}^{(t)}, \mathbb{E}_{z\sim\mathcal{D}}[\mathbf{v}_t|\mathbf{w}^{(t)}] \rangle \end{aligned}$$

So, $\mathbb{E}_{z\sim\mathcal{D}}[\mathbf{v}_t|\mathbf{w}^{(t)}]$ is an element of $\partial L_{\mathcal{D}}(\mathbf{w}^{(t)})$ → subgradient

Stochastic Gradient Descent

SGD for learning problems

```
Algorithm 10.2: SGD for minimization of $L_{\mathcal{D}}(\mathbf{w})$

Input: $\nu > 0, T \in \mathbb{N}_+$
Initialization: $\mathbf{w}^{(1)} = \mathbf{0}$
  ▶ Iterate over $t = 1, \ldots, T$
      ▶ sample $z \sim \mathcal{D}$
      ▶ pick $\mathbf{v}_t \in \partial\nabla\ell(\mathbf{w}^{(t)}, z)$
      ▶ update $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \nu\mathbf{v}_t$
Output:
    $\overline{\mathbf{w}} = \frac{1}{T} \sum_{t \in [T]} \mathbf{w}^{(t)}$
```

Stochastic Gradient Descent

SGD for learning problems

Corollary 10.2

Consider a convex-Lipschitz-bounded learning problem with parameters $B, \rho > 0$. Then, for every $\epsilon > 0$, if we run Algorithm 10.2 with a number of iterations (i.e., number of examples)

$$T \geq \frac{B^2 \rho^2}{\epsilon^2}$$

and with

$$\nu = \sqrt{\frac{B^2}{\rho^2 T}}\ ,$$

then the output of SGD satisfies

$$\mathbb{E}[L_{\mathcal{D}}(\overline{\mathbf{w}})] \leq \min_{\mathbf{w} \in \mathcal{H}} L_{\mathcal{D}}(\mathbf{w}) + \epsilon\ .$$

Stochastic Gradient Descent

SGD for regularized loss minimization

Remember, the goal of RLM is

$$\boxed{\min_{\mathbf{w}\in\mathbb{R}^d}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + L_S(\mathbf{w})\right)}$$

We know, from Lemma 9.1, that

$$f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + L_S(\mathbf{w})$$

is λ-strongly convex.

How do we compute a subgradient of f at $\mathbf{w}$? Just pick z at random from S and let $\mathbf{v}_t \in \partial\ell(\mathbf{w}^{(t)}, z)$, then

$$\mathbb{E}_{z\sim\mathcal{D}}[(\lambda\mathbf{w}^{(t)} + \mathbf{v}_t)] \in \partial f(\mathbf{w}^{(t)})$$

since it's a convex combination of a gradient with a subgradient.

Stochastic Gradient Descent

SGD for regularized loss minimization

Let's expand the update rule

$$\begin{aligned}
\mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - \frac{1}{\lambda t}\left(\lambda \mathbf{w}^{(t)} + \mathbf{v}_t\right) \\
&= \mathbf{w}^{(t)} - \frac{1}{t}\mathbf{w}^{(t)} + \frac{1}{t\lambda}\mathbf{v}_t \\
&= \left(1 - \frac{1}{t}\right)\mathbf{w}^{(t)} - \frac{1}{t\lambda}\mathbf{v}_t \\
&= \left(\frac{t-1}{t}\right)\mathbf{w}^{(t)} + \frac{1}{t\lambda}\mathbf{v}_t \\
&= \frac{t-1}{t}\left(\mathbf{w}^{(t-1)} - \frac{1}{\lambda(t-1)}\left[\lambda \mathbf{w}^{(t-1)} + \mathbf{v}_{t-1}\right]\right) + \frac{1}{t\lambda}\mathbf{v}_t \\
&= \frac{t-1}{t}\left(\frac{t-2}{t-1}\mathbf{w}^{(t-1)} - \frac{1}{\lambda(t-1)}\mathbf{v}_{t-1}\right) - \frac{1}{t\lambda}\mathbf{v}_t \\
&= \frac{t-2}{t}\mathbf{w}^{(t-1)} - \frac{1}{t\lambda}\mathbf{v}_{t-1} - \frac{1}{t\lambda)}\mathbf{v}_t \\
&\vdots \qquad\qquad\qquad\qquad \text{(expand further)} \\
&= -\frac{1}{t\lambda}\sum_{i=1}^{t}\mathbf{v}_t \qquad\qquad \text{(since initial } \mathbf{w}^{(0)} \text{ is } \mathbf{0}\text{)}
\end{aligned}$$

Stochastic Gradient Descent

SGD for regularized loss minimization

Let's look at the (extended) SGD algorithm for minimizing λ-strongly convex functions first!

Algorithm 10.3: SGD for minimization of λ-strongly convex f

```
Input: $T \in \mathbb{N}_+$
Initialization: $\mathbf{w}^{(1)} = \mathbf{0}$
  ▶ Iterate over $t = 1, \dots, T$
      ▶ choose random vector $\mathbf{v}_t$ s.t. $\mathbb{E}[\mathbf{v}_t | \mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$
      ▶ set $\nu_t = {}^{1}/_{\lambda t}$ (learning rate)
      ▶ update $\mathbf{w}^{(t+{}^{1}/_{2})} = \mathbf{w}^{(t)} - \nu_t \mathbf{v}_t$
      ▶ update $\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w} \in \mathcal{H}} \|\mathbf{w} - \mathbf{w}^{(t+{}^{1}/_{2})}\|^2$ (projection step)
Output:
    $$\overline{\mathbf{w}} = \frac{1}{T} \sum_{t \in [T]} \mathbf{w}^{(t)}$$
```

Stochastic Gradient Descent

SGD for regularized loss minimization

The following theorem holds in this case (without proof):

Theorem 10.2

Assume that f is λ-strongly convex and that $\mathbb{E}[\|\mathbf{v}_t\|^2\| \leq \rho^2$. Let

$$\mathbf{w}^* \in \arg\min_{\mathbf{w}\in\mathcal{H}} f(\mathbf{w})$$

be an optimal solution. Then

$$\mathbb{E}[f(\overline{\mathbf{w}})] - f(\mathbf{w}^*) \leq \frac{\rho^2}{2\lambda T}(1 + \log(T))$$

Stochastic Gradient Descent

SGD for regularized loss minimization

We know that $\mathbf{v}_t \in \partial\ell(\mathbf{w}^{(t)}, z)$. Assuming the loss function is ρ-Lipschitz, we further know that

$$\|\mathbf{v}_t\| \leq \rho$$

(since not only the gradient but also the subgradients are bounded). We then expand the update rule

$$\begin{aligned}
\mathbf{w}^{(t)} &= -\frac{1}{\lambda(t-1)}\sum_{i=1}^{t-1}\mathbf{v}_t \\
\Rightarrow \lambda\mathbf{w}^{(t)} &= -\frac{1}{(t-1)}\sum_{i=1}^{t-1}\mathbf{v}_t \\
\Rightarrow \|\lambda\mathbf{w}^{(t)}\| &= \left|\frac{-1}{(t-1)}\right| \left\|\sum_{i=1}^{t-1}\mathbf{v}_t\right\| \\
&\leq \left|\frac{-1}{(t-1)}\right| (\|\mathbf{v}_1\| + \cdots + \|\mathbf{v}_{t-1}\|) \\
&\leq \rho
\end{aligned}$$

Thus, $\|\lambda\mathbf{w}^{(t)} + \mathbf{v}_t\| \leq \|\lambda\mathbf{w}^{(t)}\| + \|\mathbf{v}_t\| \leq 2\rho$

Stochastic Gradient Descent

SGD for regularized loss minimization

With the result

$$\|\lambda \mathbf{w}^{(t)} + \mathbf{v}_t\| \leq 2\rho$$

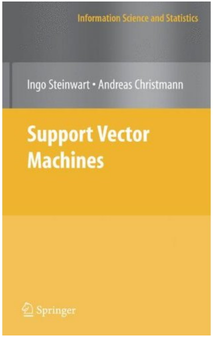
we invoke Theorem 10.2 which says that Algorithm 10.3 outputs $\overline{\mathbf{w}}$ with guarantee

$$\boxed{\mathbb{E}[f(\overline{\mathbf{w}})] - f(\mathbf{w}^*) \leq \frac{2\rho^2}{\lambda T}(1 + \log(T))}$$

This is an important result, especially since many learning algorithms can be cast as variants of RLM (such as SVM's for instance).

Support vector machines

(*cf.* [4, Chapter 15])


Information Science and Statistics
Ingo Steinwart • Andreas Christmann
Support Vector Machines
Springer

Support vector machines

Introduction

Idea

Learn a linear predictor in a high-dimensional space.

Problems:

- Sample complexity
- Computational complexity

We will see how support vector machines address both issues by searching for large margin seperators.

Support vector machines

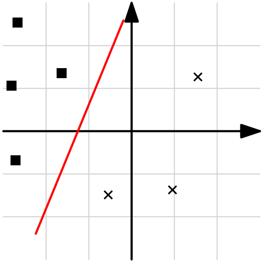
Which linear predictor to choose?

- $S = (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_m, y_m)$ with each tuple in $\mathbb{R}^d \times \{-1, +1\}$
- If S is linearly separable, then

$$\boxed{\forall i \in [m] : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) > 0}$$

- Every halfspace $(\mathbf{w}, b)$ that satisfies this is a valid ERM hypothesis

  But, which one should we choose?



Support vector machines

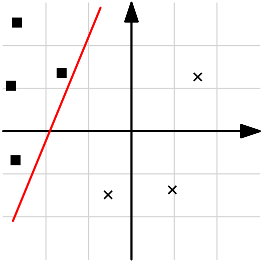
Which linear predictor to choose?

- $S = (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_m, y_m)$ with each tuple in $\mathbb{R}^d \times \{-1, +1\}$
- If S is linearly separable, then

$$\boxed{\forall i \in [m] : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) > 0}$$

- Every halfspace $(\mathbf{w}, b)$ that satisfies this is a valid ERM hypothesis

  But, which one should we choose?



Support vector machines

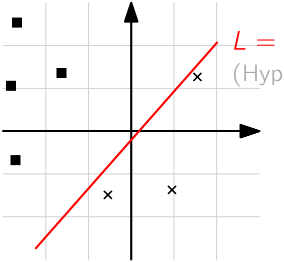
Which linear predictor to choose?

- $S = (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_m, y_m)$ with each tuple in $\mathbb{R}^d \times \{-1, +1\}$
- If S is linearly separable, then

$$\boxed{\forall i \in [m] : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) > 0}$$

- Every halfspace $(\mathbf{w}, b)$ that satisfies this is a valid ERM hypothesis

  But, which one should we choose?



$L = \{\mathbf{v} : \langle \mathbf{w}, \mathbf{v} \rangle + b\}$

(Hyperplane)

Support vector machines

Hard-SVM

Claim 11.1

The distance $d(\mathbf{x}, L)$ between a point $\mathbf{x} \in \mathbb{R}^d$ and the hyperplane $(\mathbf{w}, b)$ with $\|\mathbf{w}\| = 1$ is $d(L, \mathbf{x}) = |\langle \mathbf{w}, \mathbf{x} \rangle + b|$.

Proof. In fact, the distance is

$$d(L, \mathbf{x}) = \min\{\|\mathbf{x} - \mathbf{v}\| : \langle \mathbf{w}, \mathbf{v} \rangle + b = 0\}$$

First, we let $\mathbf{v} = \mathbf{x} - (\langle \mathbf{w}, \mathbf{x} \rangle) + b)\mathbf{w}$ and show that $\mathbf{v}$ is in fact on the hyperplane, *i.e.*,

$$\begin{aligned}\langle \mathbf{w}, \mathbf{v} \rangle + b &= \langle \mathbf{w}, \mathbf{x} - (\langle \mathbf{w}, \mathbf{x} \rangle) + b)\mathbf{w} \rangle \\ &= \langle \mathbf{w}, \mathbf{x} \rangle - \underbrace{\|\mathbf{w}\|^2}_{1}(\langle \mathbf{x}, \mathbf{w} \rangle + b) + b = 0\end{aligned}$$

Support vector machines

Hard-SVM (contd.)

Next, we take a look at $\|\mathbf{x}-\mathbf{v}\|$, *i.e.*,

$$\begin{aligned}\|\mathbf{x}-[\mathbf{x}-(\langle\mathbf{w},\mathbf{x}\rangle)+b]\mathbf{w}\| &= \|(\langle\mathbf{w},\mathbf{x}\rangle)+b)\mathbf{w}\|\\ &= |(\langle\mathbf{w},\mathbf{x}\rangle)+b)|\cdot\|\mathbf{w}\|\\ &= |(\langle\mathbf{w},\mathbf{x}\rangle)+b)|\end{aligned}$$

Hence, the distance is at most $|(\langle\mathbf{w},\mathbf{x}\rangle)+b)|$. To establish our claim, we show that any other point $\mathbf{u}$ on the hyperplane is at least $\|\mathbf{x}-\mathbf{v}\|$ away.

$$\begin{aligned}\|\mathbf{x}-\mathbf{u}\|^2 &= \|\mathbf{x}-\mathbf{v}+\mathbf{v}-\mathbf{u}\|^2\\ &= \|\mathbf{x}-\mathbf{v}\|^2+\|\mathbf{x}-\mathbf{u}\|^2-2\langle\mathbf{x}-\mathbf{v},\mathbf{v}-\mathbf{u}\rangle\\ &\geq \|\mathbf{x}-\mathbf{v}\|^2-2\langle\mathbf{x}-\mathbf{v},\mathbf{v}-\mathbf{u}\rangle \qquad \text{(next, replace first } \mathbf{v}\text{)}\\ &= \|\mathbf{x}-\mathbf{v}\|^2+2(\langle\mathbf{w},\mathbf{x}\rangle+b)\langle\mathbf{w},\mathbf{v}-\mathbf{u}\rangle\\ &= \|\mathbf{x}-\mathbf{v}\|^2 \quad \text{(since, } \langle\mathbf{w},\mathbf{u}\rangle=\langle\mathbf{w},\mathbf{v}\rangle=-b\text{)}\end{aligned}$$

□

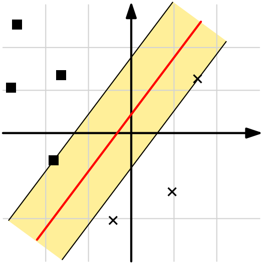
Support vector machines

Hard-SVM – Margin

Margin of a separating hyperplane

The margin of a separating hyperplane L is the distance of the closest example to it, *i.e.*

$$\min_{i \in [m]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$$



Support vector machines

Hard-SVM – The concept of a “large margin”

Hard-SVM

Hard-SVM is the learning rule in which we return the ERM hyperplane that separates the training set with the largest possible margin.

Formally, this corresponds to the following optimization problem:

$$\begin{aligned} &\arg \max_{(\mathbf{w},b):\|\mathbf{w}\|=1} \min_{i\in[m]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \\ \text{subject to} \quad &\forall i, y_i(\langle \mathbf{w}, \mathbf{x} \rangle + b) > 0 \end{aligned}$$

Support vector machines

Hard-SVM – Algorithm

Algorithm 11.1: Hard-SVM

Input: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$

Solve:

$$(\mathbf{w}_0, b_0) = \arg\min_{(\mathbf{w}, b)} \|\mathbf{w}\|^2$$

$$\textit{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

Output:

$$\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|} \quad \textit{and} \quad \hat{b} = \frac{b}{\|\mathbf{w}_0\|}$$

This is a quadratic optimization problem.

Support vector machines

Hard-SVM – Correctness

Claim 11.2

Algorithm 11.1 outputs the hyperplane with the largest margin.

Proof. First, we note that our previous optimization problem

$$\begin{aligned} &\arg \max_{(\mathbf{w},b):\|\mathbf{w}\|=1} \min_{i\in[m]} |\langle \mathbf{w}, \mathbf{x}_i\rangle + b| \\ \text{subject to} \quad &\forall i, y_i(\langle \mathbf{w}, \mathbf{x}\rangle + b) > 0 \end{aligned}$$

is equivalent to (as we assume separability)

$$\boxed{\arg \max_{(\mathbf{w},b):\|\mathbf{w}\|=1} \min_{i\in[m]} y_i(\langle \mathbf{w}, \mathbf{x}\rangle + b)} \qquad (11.1)$$

We show that Algorithm 11.1 outputs an optimal solution to Eq. (11.1).

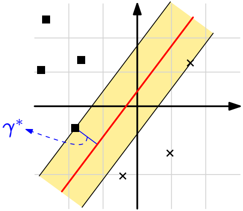
Support vector machines

Hard-SVM – Correctness (contd.)

Let $(\mathbf{w}^*, b^*)$ be a solution to the optimization problem in Eq. (11.1) and let

$$\gamma^* = \min_{i\in[m]} y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)$$

Graphically:



Consequently,

$$\forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle) + b \geq \gamma^*$$
$$\Rightarrow \forall i : y_i\left(\left\langle \frac{\mathbf{w}^*}{\gamma^*}, \mathbf{x}_i\right\rangle + \frac{b^*}{\gamma^*}\right) \geq 1$$

Support vector machines

Hard-SVM – Correctness (contd.)

Hence, the tuple

$$\left(\frac{\mathbf{w}^*}{\gamma^*}, \frac{b^*}{\gamma^*}\right)$$

satisfies the “subject to” requirement in the Hard-SVM algorithm. Also, $\mathbf{w}_0$ is chosen to be the smallest $\mathbf{w}$ s.t. the requirement holds. Thus,

$$\|\mathbf{w}_0\| \leq \left\|\frac{\mathbf{w}^*}{\gamma^*}\right\| \leq \frac{1}{\gamma^*} \quad (\text{since, } \|\mathbf{w}^*\| = 1)$$

and we have

$$\begin{aligned}\forall i : y_i(\langle \hat{\mathbf{w}}, \mathbf{x}_i \rangle + \hat{b}) &= \frac{1}{\|\mathbf{w}_0\|} y_i(\langle \mathbf{w}_0, \mathbf{x}_i \rangle + b_0) \\ &\geq \frac{1}{\|\mathbf{w}_0\|} \geq \gamma^*\end{aligned}$$

Also, $\|\hat{\mathbf{w}}\| = 1$ which concludes the proof. □

Support vector machines

Hard-SVM – Homogenous halfspaces

In the homogeneous case we know that

$$\mathbf{w} = [b, w_1, \ldots, w_d] \in \mathbb{R}^{d+1}$$

and the optimization problem becomes

$$\boxed{\mathbf{w}_0 = \min_{\mathbf{w}} \underbrace{\|\mathbf{w}\|^2}_{\text{regularization}}, \text{ subject to } \forall i : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1}$$

This effectively regularizes the bias b (since, it's included in $\mathbf{w}$) which is not the case in our previous formulation[23].

[23] but typically does not make a big difference in practice

Support vector machines

Hard-SVM – Scale sensitivity of the margin

Assume

$$\underbrace{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)}_{\text{separated by margin } \gamma} \Rightarrow \underbrace{(2\mathbf{x}_1, y_1), \ldots, (2\mathbf{x}_m, y_m)}_{\text{separated by margin } 2\gamma}$$

In general, scaling samples by $\alpha > 0$ scales the margin by α.

For that reason, we introduce the notion of (γ, ρ)-separability.

Definition 11.1: (γ, ρ)-separability

Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{+1, -1\}$. We say $\mathcal{D}$ is separable with a (γ, ρ)-margin if there exists $(\mathbf{w}^*, b^*)$ such that $\|\mathbf{w}^*\| = 1$ and such that, with probability 1 over the choice of $(\mathbf{x}, y) \sim \mathcal{D}$, we have

$$y(\langle \mathbf{w}^*, \mathbf{x} \rangle + b^*) \geq \gamma \text{ and } \|\mathbf{x}\| \leq \rho \ .$$

Support vector machines

Hard-SVM – Sample complexity

We know that the VC-dimension of halfspaces in $\mathbb{R}^d$ is $d+1$. Consequently,

- the sample complexity grows with the dimensionality d
- if $m \ll d/\epsilon$, no algorithm can learn an ϵ-accurate halfspace[24]

The assumption of (γ, ρ)-separability allows us to still have small sample complexity.

Remark: This does not contradict the lower bound of the fundamental theorem, since we introduce more prior knowledge (*i.e.*, inductive bias).

[24] review the fundamental theorem of statistical learning

Support vector machines

Hard-SVM – Sample complexity (contd.)

Theorem 11.1

Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{+1, -1\}$ that satisfies (γ, ρ)-separability using a homogeneous halfspace. Then, with probability $1 - \delta$ over the choice of training set of size m, the 0-1 error of the output of Hard-SVM in Algorithm 11.1 is at most

$$\sqrt{\frac{4(\rho/\gamma)^2}{m}} + \sqrt{\frac{2\log(2/\delta)}{m}}$$

Support vector machines

From Hard-SVM to Soft-SVM

We next relax the separability assumption of Hard-SVM:

$$\forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \overset{\text{relax to}}{\Rightarrow} \forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

with $\xi_i \geq 0$, referred to as slack variables. In fact, ξ_i measures how much the constraint

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

is violated.

Support vector machines

Soft-SVM – Algorithm

Algorithm 11.2: Soft-SVM

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, $\lambda > 0$

Solve:

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right)$$

$$\textit{such that} \quad \forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Output: $\mathbf{w}$, b

Support vector machines

Soft-SVM as a RLM problem

A closer look at Algorithm 11.2 reveals that the Soft-SVM problem is equivalent to the RLM problem

$$\min_{\mathbf{w},b} \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \underbrace{\max\{0, 1 - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)\}}_{\ell^{hinge}((\mathbf{w},b),(\mathbf{x}_i,y_i))} \right)$$

This is easy to see:

- $\xi_i \geq 0 \Rightarrow$ optimally, $\xi_i = 0$ if
$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$
- otherwise, $\xi_i = 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$

$$\boxed{\Rightarrow \xi_i = \ell^{hinge}((\mathbf{w}, b), (\mathbf{x}_i, y_i))}$$

In what follows, we work with homogeneous thes (so, no b)!

Support vector machines

Sample complexity of Soft-SVM

Since we already know a lot about convex-Lipschitz-bounded learning problems, let's see what we have here:

- $\ell^{hinge}(\cdot, z)$ is convex

But, is $\ell^{hinge}(\cdot, z)$ ρ-Lipschitz? And, if so, with what ρ?

Looking at Lemma 10.3, lets consider subgradients $\mathbf{v}$ of the Hinge loss, *i.e.*, $\mathbf{v} = -y\mathbf{x}$ or $\mathbf{0}$ (remember, we had this before). Hence,

$$\|\mathbf{v}\| = \underbrace{|y|}_{1} \|\mathbf{x}\|$$

$\Rightarrow$ The Hinge loss is convex and $\|\mathbf{x}\|$-Lipschitz.

Support vector machines

Sample complexity of Soft-SVM

We also know that the Hinge loss upper bounds the 0-1 loss. Invoking Corollary 9.2 then yields

Corollary 11.1

Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{0,1\}$, where $\mathcal{X} = \{\mathbf{x} : \|\mathbf{x}\| \leq \rho\}$ and let $A(S)$ be the output of running the Soft-SVM algorithm on $S \sim \mathcal{D}^m$. Then, for every $B > 0$

$$\begin{aligned}\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}^{0-1}(A(S))] &\leq \mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}^{hinge}(A(S))] \\ &\leq \min_{\mathbf{w}:\|\mathbf{w}\|\leq B} L_{\mathcal{D}}^{hinge}(\mathbf{w}) + \rho B\sqrt{\frac{8}{m}}\end{aligned}$$

The sample complexity can be controlled by the norm $\|\mathbf{w}\|$ of the halfspace (independent of dimension)!

Support vector machines

Implementing Soft-SVM via SGD

We want to solve the optimization problem (Soft-SVM):

$$\min_{\mathbf{w}} \left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}_i\rangle\} \right)$$

Recall, this is an instance of regularized loss minimization (RLM). Also, we can write the update rule (see SGD lecture) as

$$\mathbf{w}^{(t+1)} = -\frac{1}{\lambda t}\sum_{j=1}^{t} \mathbf{v}_j$$

where $\mathbf{v}_j$ is a subgradient of the loss function at $\mathbf{w}^{(j)}$.

Support vector machines

Implementing Soft-SVM via SGD

Algorithm 11.3: Soft-SVM via SGD

```
Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, $T$
Initialize: $\boldsymbol{\theta}^{(1)} = \mathbf{0}$ and let $\theta^{(t)} = -\sum_{j<t} \mathbf{v}_t$
for $t = 1, \ldots, T$
    Let $\mathbf{w}^{(t)} = {}^1/_{\lambda t}\boldsymbol{\theta}^{(t)}$
    Choose $i$ uniformly at random from $[m]$
    If $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle < 1$
        Set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + y_i \mathbf{x}_i$
    Else
        Set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)}$
Output: $\overline{\mathbf{w}} = {}^1/_T \sum_{t=1}^{T} \mathbf{w}^{(t)}$
```

Support vector machines

Summary

- SVM prior knowledge $\equiv$ preference for large margin
- Hard-SVM seeks a halfspace that separates the data perfectly
- Soft-SVM allows for the constraints to be violated (via slack variables)
- Sample complexity:
  - depends on max. norms of $\mathbf{x}, \mathbf{w}$
  - independent of the dimension of the domain
  - important in the context of **kernels** (not discussed here)

Bibliography I

T. Michtell.
Machine Learning.
McGraw-Hill, 1997.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar.
Foundations of Machine Learning.
The MIT Press, 2012.

Kevin P. Murphy.
Machine Learning: A Probabilistic Perspective.
The MIT Press, 2012.

Shai Shalev-Shwartz and S. Ben-David.
Understanding Machine Learning: From Theory to Algorithms.
Cambridge University Press, 2014.