

# Machine Learning

Anders Jonsson & Vicenç Gómez

Master in Intelligence Interactive Systems  
2021-22

Lecture 1  
Introduction to Machine Learning

# Content

- 1 General information**
- 2 Introduction to Machine Learning**
- 3 Mathematical formulation of supervised learning**
- 4 Basic algorithms**
- 5 Exercises**

## Content

- 1 General information**
  - 2 Introduction to Machine Learning**
  - 3 Mathematical formulation of supervised learning**
  - 4 Basic algorithms**
  - 5 Exercises**

## Organization

- Teachers: Anders Jonsson, Vicenç Gómez
    - Office: 55.130
    - Email: {anders.jonsson,vicen.gomez}@upf.edu
  - 10 lectures, 2.5 hours each
    - Mostly theory in lectures
    - Several lab sessions (lectures 3, 5, 7 and 8)
  - Homeworks and programming exercises
  - Final exam in December

# Evaluation

- Final grade is calculated as follows:
  - 1 Homeworks:  $4 \times 5\% = 20\%$
  - 2 Programming exercises:  $4 \times 10\% = 40\%$
  - 3 Final exam: 40%
- To pass the course, need a passing grade in all parts

# Objective

- Understand the mathematical principles of machine learning
- Solve mathematical exercises related to machine learning theory
- Recognize different types of machine learning problem
- Choose and implement appropriate machine learning algorithms
- Evaluate learning outcome and compare different algorithms
- Select appropriate values of hyperparameters through validation

# Prerequisites

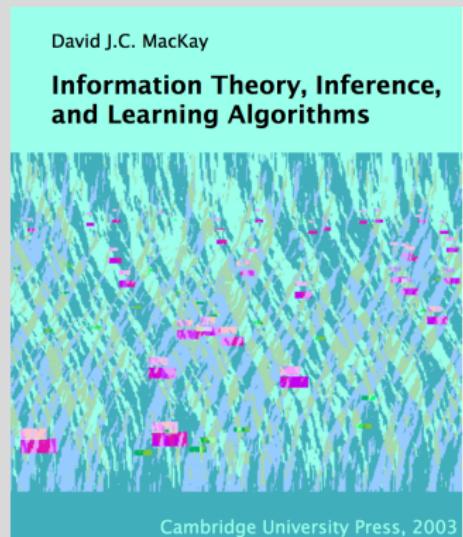
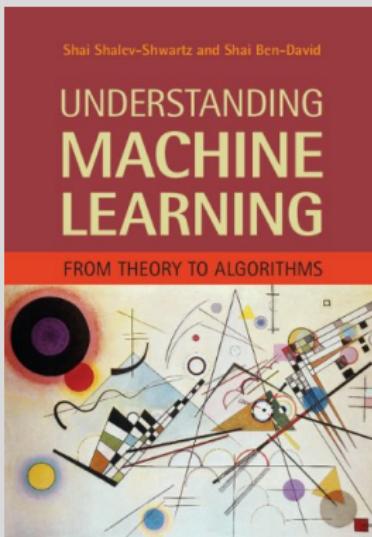
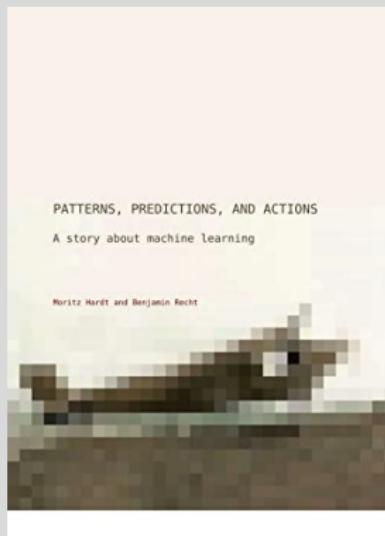
Students are expected to have basic knowledge in these topics:

- Linear algebra
- Calculus
- Probability theory/statistics

# Topics

1. Introduction to machine learning (Anders)
2. Linear models (Anders)
3. Bias-variance tradeoff and overfitting (Anders) + LS #1
4. Optimization and gradient descent (Anders)
5. Common algorithms for supervised learning (Anders) + LS #2
6. Unsupervised learning (Anders)
7. Deep Learning and Applications I (Anders) + LS #3
8. Deep Learning and Applications II (Vicenç) + LS #4
9. Bayesian learning and inference (Vicenç)
10. Introduction to probabilistic graphical models (Vicenç)

# Books



# Content

1 General information

2 Introduction to Machine Learning

3 Mathematical formulation of supervised learning

4 Basic algorithms

5 Exercises

# Definition of machine learning

From Wikipedia: “Machine learning (ML) is the study of **computer algorithms** that can **improve automatically** through experience and by the use of data. Machine learning algorithms build a **model** based on **sample data** [...] in order to make **predictions or decisions**.”

# Machine learning problems

In which of the following problems can machine learning help?

- Predict on which weekday the next market crash will occur
- Classify an email as spam or not spam
- Find the shortest path between two nodes in a graph
- Recognize cancerous cells in the X-ray image of a patient
- Guess when the next world-wide pandemic will occur

# Characteristics of machine learning problems

- There exists an underlying **pattern** that we can exploit
- Describing (or programming) the pattern is difficult
- However, one can obtain many **observations (data)** of the pattern
- Machine learning algorithms can **learn** the pattern (or an approximation) from data

# Machine learning problems

In which of the following problems can machine learning help?

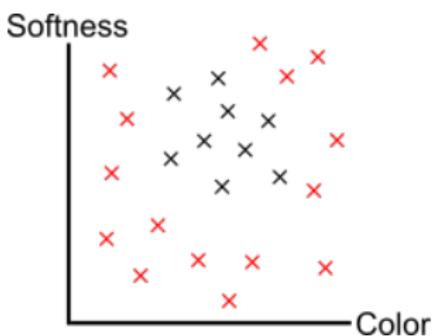
- Predict on which weekday the next market crash will occur
- Classify an email as spam or not spam
- Find the shortest path between two nodes in a graph
- Recognize cancerous cells in the X-ray image of a patient
- Guess when the next world-wide pandemic will occur

# Intuition



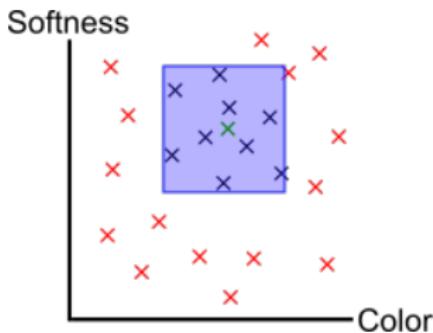
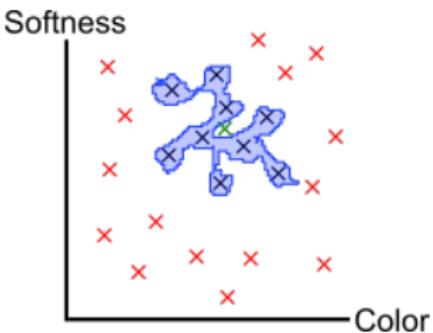
- There exists a set of objects or concepts that we want to analyze
- **Key assumption:** objects with similar features behave similarly
- If this assumption does not hold, learning is hopeless

# Example [Understanding Machine Learning]



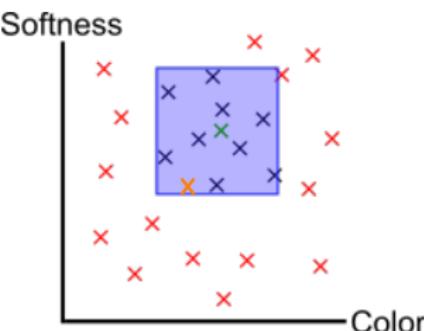
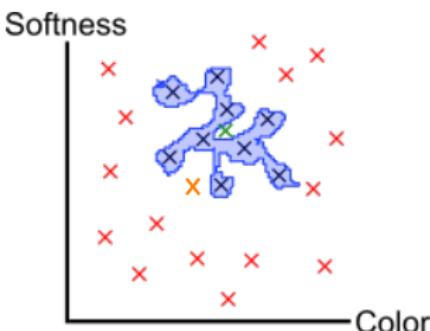
- Papayas can be described by two features: softness and color
- Papayas are either tasty (x) or not tasty (✗)
- We can obtain many **observations** of papayas: softness, color, and tastiness

# Example (cont.)



- Machine learning algorithms attempt to find a **pattern** that explains the relationship between **features** (softness, color) and **label** (tastiness)
- Many possible patterns exist for explaining this relationship!

# Prediction



- Given an unseen papaya (x), predict whether or not it is tasty
- Record features (softness, color) and use pattern model to predict tastiness
- Prediction depends on model! Need a way to measure **performance** of models

# Areas of machine learning

- **Supervised learning**: data contains (feature,label) pairs, objective is to predict target for unseen examples
- **Unsupervised learning**: data only contains feature examples, objective is to find patterns in features
- **Reinforcement learning**: algorithms can perform **actions** that modify features, objective is to perform actions that maximize long-term **reward**

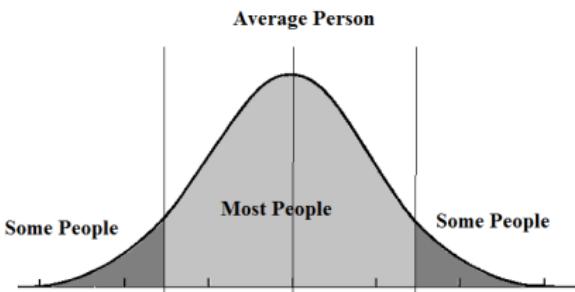
# Content

- 1 General information**
- 2 Introduction to Machine Learning**
- 3 Mathematical formulation of supervised learning**
- 4 Basic algorithms**
- 5 Exercises**

# Domain set

- Assume that there exists a set of **features**  $\mathcal{X}^1, \dots, \mathcal{X}^d$
- Objects or concepts are described by **feature vectors**  $(x^1, \dots, x^d)$
- An **input** is given by  $x = (x^1, \dots, x^d) \in \mathcal{X}^1 \times \dots \times \mathcal{X}^d \equiv \mathcal{X}$
- $\mathcal{X}$  is called the **domain set**, **input set**, or **sample space**
- $d$  is the number of **dimensions**, e.g.  $d = 2$  in the papaya example

# Input distribution



- Not all inputs are equally likely
- Assume that inputs follow a **probability distribution  $\mathcal{D}$**
- We can **sample** an input  $x \sim \mathcal{D}$  from the probability distribution

# Target set

- Target set (label space)  $\mathcal{Y}$  represents what we want to predict
- The machine learning problem depends on the form of  $\mathcal{Y}$ :

$\mathcal{Y} = \{c_1, \dots, c_k\}$ : classification (target is a class)

$\mathcal{Y} = \mathbb{R}$ : regression (target is a real number)

$\mathcal{Y} = [0, 1]$ : logistic regression (target is a probability)

- For example,  $\mathcal{Y} = \{\text{tasty}, \text{not tasty}\}$  in the papaya example

# Labelling function

- The unknown **pattern** that we are trying to learn
- Mathematically, the pattern is a **labelling function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$
- For a given input  $x \in \mathcal{X}$ , the label is  $y = f(x) \in \mathcal{Y}$
- In practice, labelling is noisy, and two identical inputs may have different labels

# Training set

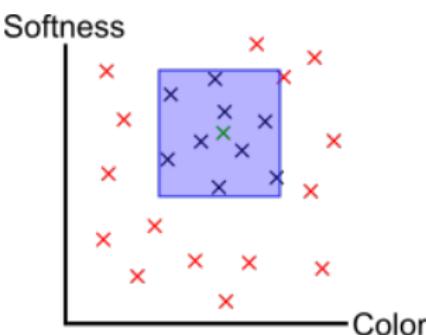
- Data is obtained by observing multiple labelled examples
- The result is a **training set**  $S = ((x_1, y_1), \dots, (x_m, y_m))$
- Each data point  $(x_i, y_i)$ ,  $1 \leq i \leq m$ , consists of an input  $x_i \sim \mathcal{D}$  and a label  $y_i = f(x_i)$
- Technically,  $S$  is a **sequence** and not a set, since two distinct data points  $(x_i, y_i)$  and  $(x_j, y_j)$ ,  $1 \leq i < j \leq N$ , may be identical
- Some algorithms process the data points in  $S$  sequentially

# Supervised learning problem

To summarize, a supervised learning problem consists of:

- A domain set  $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^d$
- An **unknown** probability distribution  $\mathcal{D}$  on  $\mathcal{X}$
- A target set  $\mathcal{Y}$
- An **unknown** labelling function  $f : \mathcal{X} \rightarrow \mathcal{Y}$
- A training set  $S = ((x_1, y_1), \dots, (x_m, y_m))$  **sampled** from  $\mathcal{D}$  and  $f$

# Hypothesis class



- Since the space of functions is enormous, the learner usually **restricts** the space of candidate functions
- Hypothesis class  $\mathcal{H}$ : set of functions considered by the learner
- Each hypothesis  $h \in \mathcal{H}$  is a function from  $\mathcal{X}$  to  $\mathcal{Y}$ , i.e.  $h : \mathcal{X} \rightarrow \mathcal{Y}$
- Papaya example: set of rectangles in two dimensions

# Loss function

- Need a way to measure the **performance** of a hypothesis  $h \in \mathcal{H}$
- **Loss function**  $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ : measures the **error** between a prediction  $\hat{y}$  and the true label  $y$
- Loss is a (non-negative) **cost**: lower is better
- Many different loss functions exist
  - Classification or 0-1 loss:  $\ell(\hat{y}, y) = [\hat{y} \neq y]$
  - Squared loss:  $\ell(\hat{y}, y) = (\hat{y} - y)^2$

# True loss vs. training loss

- Loss can measure the **performance** of a hypothesis  $h \in \mathcal{H}$
- **True loss**  $L_{\mathcal{D},f}(h)$ , also known as **generalization error** or **risk**, measures the mistakes of  $h$  on the entire **domain set**  $\mathcal{X}$

$$L_{\mathcal{D},f}(h) = \mathbb{E}_{x \sim \mathcal{D}} \{\ell(h(x), f(x))\}$$

- **Training loss**  $L_S(h)$ , also known as **empirical risk**, measures the mistakes of  $h$  on the **training set**  $S = ((x_1, y_1), \dots, (x_m, y_m))$

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

# Learning algorithm



- A learning algorithm should output a hypothesis  $h \in \mathcal{H}$  that approximates  $f$  well
- We would like  $h$  to minimize the true loss  $L_{\mathcal{D},f}(h)$
- However,  $L_{\mathcal{D},f}(h)$  is **unobservable** since  $\mathcal{D}$  and  $f$  are unknown

# Empirical risk minimization

- Basic principle of supervised learning
- Given a hypothesis class  $\mathcal{H}$  and a training set  $S$ , return the hypothesis that **minimizes the empirical risk** (i.e. training loss)
- Formally,

$$h_S = \text{ERM}_{\mathcal{H}}(S) \in \arg \min_{h \in \mathcal{H}} L_S(h),$$

- Note that if  $f \in \mathcal{H}$ ,  $L_S(f) = 0$

# Supervised learning

To summarize, given a supervised learning problem, the learner chooses the following:

- A hypothesis class  $\mathcal{H}$  of candidate labelling functions
- A loss function  $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- An algorithm  $\mathcal{A}$  that minimizes the empirical risk

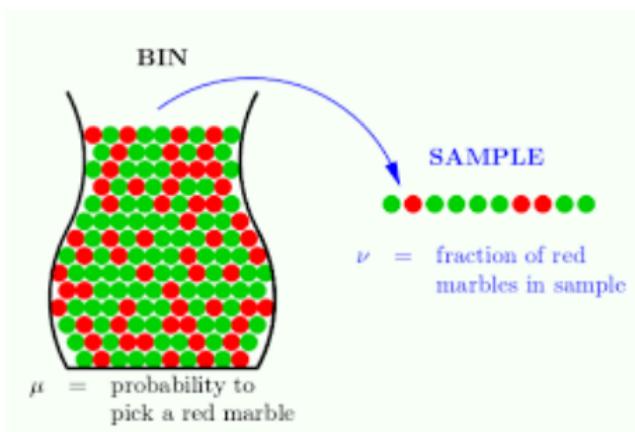
# Generalization properties

- Trivially, we have

$$L_{\mathcal{D},f}(h) = L_S(h) + (L_{\mathcal{D},f}(h) - L_S(h))$$

- For a fixed  $h \in \mathcal{H}$ , how well does  $L_S(h)$  approximate  $L_{\mathcal{D},f}(h)$ ?
- Depends on quality of the training set  $S = ((x_1, y_1), \dots, (x_m, y_m))$
- If we are **unlucky**,  $L_S(h)$  is not representative of  $L_{\mathcal{D},f}(h)$
- With **high probability**,  $L_S(h)$  approximates  $L_{\mathcal{D},f}(h)$  well

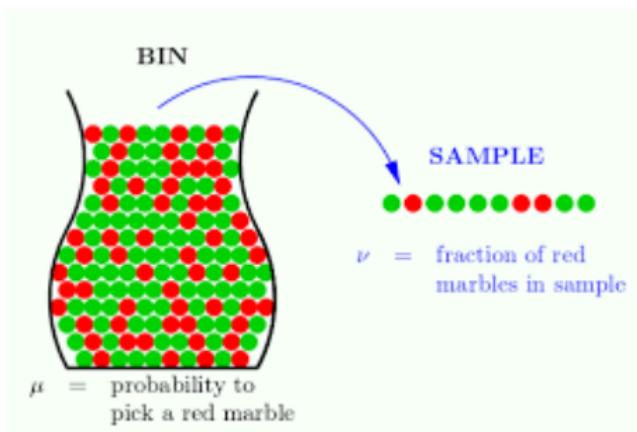
# Bin analogy



- Large bin with red and green marbles, proportion red/green =  $\mu$
- Randomly pick  $m$  marbles, record proportion  $\nu$  of red marbles
- Hoeffding's inequality (holds for any  $\epsilon$ ):

$$\mathbb{P} [|\nu - \mu| > \epsilon] \leq 2e^{-2m\epsilon^2}$$

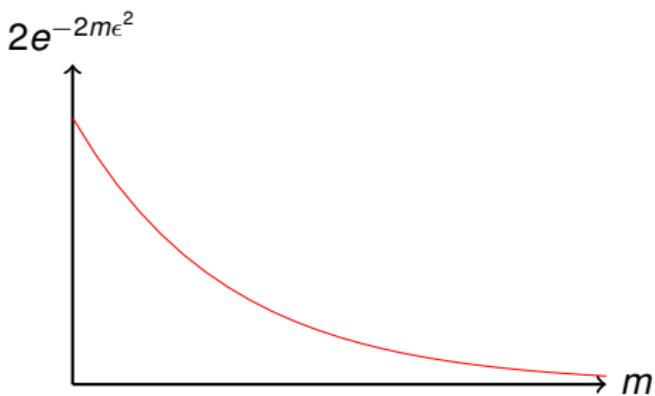
# Bin analogy



- Marbles: inputs in  $\mathcal{X}$ , red:  $h(x) \neq f(x)$ , green:  $h(x) = f(x)$
- Then  $L_{\mathcal{D},f}(h) = \mu$ ,  $L_S(h) = \nu$
- Hoeffding's inequality:

$$\mathbb{P} [|L_S(h) - L_{\mathcal{D},f}(h)| > \epsilon] \leq 2e^{-2m\epsilon^2}$$

# Growth of upper bound



- Probability of  $S$  being **bad** decreases as a function of  $m$

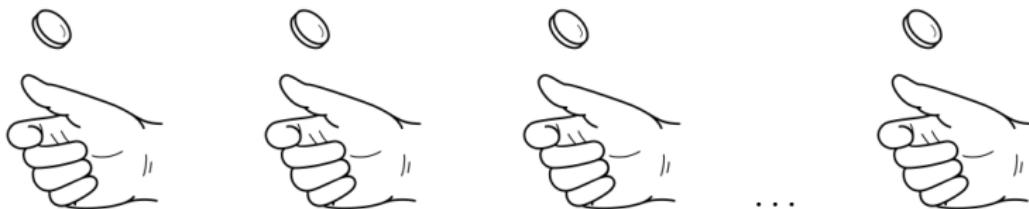
# Generalization

- Empirical risk minimization returns a hypothesis

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

- How well does  $L_S(h_S)$  approximate  $L_{\mathcal{D},f}(h_S)$ ?
- **Bias** due to the fact that we select  $h$  with **smallest  $L_S(h)$**

# Coin flipping



- Assume that  $K$  people each flip a coin 10 times
- In expectation, each person will get 5 heads and 5 tails
- However, probability of getting 10 heads (or 10 tails)  $\approx 0.1\%$
- Analogously, the  $h$  with smallest  $L_S(h)$  may have been **lucky**

# Generalization

- For each hypothesis  $h \in \mathcal{H}$ , bound the probability of  $S$  being bad

$$2e^{-2m\epsilon^2} + 2e^{-2m\epsilon^2} + \dots + 2e^{-2m\epsilon^2}$$

- Union bound on Hoeffding's inequality yields

$$\mathbb{P} [|L_S(h_S) - L_{D,f}(h_S)| > \epsilon] \leq 2|\mathcal{H}|e^{-2m\epsilon^2}$$

# Noisy labels

- In practice, two identical inputs may have different labels
- As an alternative to the labelling function  $f$ , we can model this as a **conditional probability distribution**  $\mathbb{P}(y|x)$
- On input  $x$ , the label is  $y$  with probability  $\mathbb{P}(y|x)$
- Even more compactly, we can define a **joint probability distribution**  $\mathcal{D}$  on  $\mathcal{X} \times \mathcal{Y}$  and sample data points  $(x, y) \sim \mathcal{D}$

# Content

- 1 General information**
- 2 Introduction to Machine Learning**
- 3 Mathematical formulation of supervised learning**
- 4 Basic algorithms**
- 5 Exercises**

# Naïve empirical risk minimization

## Empirical risk minimization

- For each hypothesis  $h \in \mathcal{H}$ , compute the loss  $L_S(h)$
- Return hypothesis with smallest  $L_S(h)$

Iterates over **all hypotheses** in  $\mathcal{H}$  and **all data points** in  $S$ !

# Online learning



A blue wavy line graph showing a sequence of binary digits (0s and 1s) forming a curve. The digits are arranged in a zig-zag pattern along the curve, representing a sequence of data points.

- Process data points one at a time
- Update the candidate hypothesis after seeing each data point
- **Sequential**: order of the data points matter!
- For the moment, assume  $|\mathcal{Y}| = 2$ ,  $f \in \mathcal{H}$  and finite  $|\mathcal{H}|$

# Consistent learner

## Consistent learner

- Initialize  $\mathcal{H}_1 \leftarrow \mathcal{H}$
- For  $t = 1, 2, \dots$ 
  - 1 Get  $x_t$
  - 2 Pick some  $h \in \mathcal{H}_t$  and predict  $h(x_t)$
  - 3 Get  $y_t = f(x_t)$  and update  $\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : h(x_t) = y_t\}$

# Consistent learner

## Theorem

*The consistent learner will make at most  $|\mathcal{H}| - 1$  mistakes.*

# Halving learner

## Halving learner

- Initialize  $\mathcal{H}_1 \leftarrow \mathcal{H}$
- For  $t = 1, 2, \dots$ 
  - 1 Get  $x_t$
  - 2 Pick MAJORITY( $h(x_t) : h \in \mathcal{H}_t$ )
  - 3 Get  $y_t = f(x_t)$  and update  $\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : h(x_t) = y_t\}$

# Halving learner

## Theorem

*The halving learner will make at most  $\log_2(|\mathcal{H}|)$  mistakes.*

# Halving learner

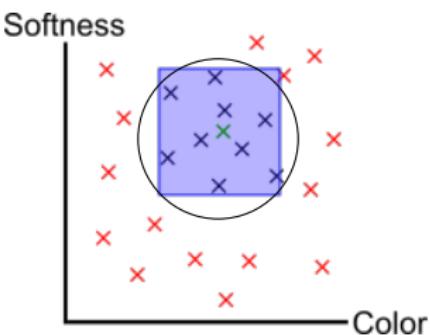
Advantages:

- Makes very few mistakes!
- No need to store a training set or compute a loss function

Disadvantages:

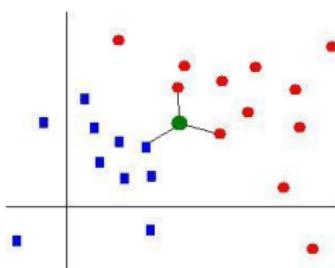
- Assumes  $f \in \mathcal{H}$  (else we may exclude best hypothesis)
- In each time step, iterates over all hypotheses in  $\mathcal{H}$ !

# Algorithmic properties



- We cannot always assume  $f \in \mathcal{H}$  and  $|\mathcal{H}|$  small (or even finite)
- Important that algorithms are computationally **efficient**
- Even minimizing the empirical risk is often expensive (NP-hard)
- Usually algorithms are **specialized** to the hypothesis class  $\mathcal{H}$  and loss function  $\ell$

# $k$ nearest neighbors



- Assume that there exists a **distance function**  $\rho : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
- Given  $S = ((x_1, y_1), \dots, (x_m, y_m))$  and  $x \in \mathcal{X}$ , let  $\pi_1(x), \dots, \pi_m(x)$  be a **permutation** of  $1, \dots, m$  according to their distance from  $x$
- $k$  nearest neighbors: make prediction based on the  $k$  nearest neighbors in  $S$

# $k$ nearest neighbors

## $k$ nearest neighbors

- Get input  $x$
- Return MAJORITY( $y_{\pi_i(x)} : i \leq k$ )

No need for a hypothesis class or a loss function!

# $k$ nearest neighbors

## Theorem

$$\mathbb{E}_{S \sim \mathcal{D}, f} [L_{\mathcal{D}, f}(h_S)] \leq 2L_{\mathcal{D}, f}(h^*) + 4c\sqrt{d}m^{-\frac{1}{d+1}}$$

$h^*$  is the **optimal** hypothesis

Constant  $c$  bounds how much  $f$  can **change** as a function of distance

# Content

- 1 General information**
- 2 Introduction to Machine Learning**
- 3 Mathematical formulation of supervised learning**
- 4 Basic algorithms**
- 5 Exercises**

# Consistent learner

Prove that the consistent learner makes at most  $|\mathcal{H}| - 1$  mistakes

## Consistent learner

- Initialize  $\mathcal{H}_1 \leftarrow \mathcal{H}$
- For  $t = 1, 2, \dots$ 
  - 1 Get  $x_t$
  - 2 Pick some  $h \in \mathcal{H}_t$  and predict  $h(x_t)$
  - 3 Get  $y_t = f(x_t)$  and update  $\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : h(x_t) = y_t\}$

# Consistent learner

- Assume the algorithm makes a mistake in round  $t$

# Consistent learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypothesis  $h$  used for prediction will **not** be part of  $\mathcal{H}_{t+1}$

# Consistent learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypothesis  $h$  used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t| - 1$

# Consistent learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypothesis  $h$  used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t| - 1$
- After  $k$  mistakes,  $|\mathcal{H}_t| \leq |\mathcal{H}| - k$

# Consistent learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypothesis  $h$  used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t| - 1$
- After  $k$  mistakes,  $|\mathcal{H}_t| \leq |\mathcal{H}| - k$
- Since  $f \in \mathcal{H}$  and  $f$  never makes mistakes,  $|\mathcal{H}_t| \geq 1$  for each  $t$

# Consistent learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypothesis  $h$  used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t| - 1$
- After  $k$  mistakes,  $|\mathcal{H}_t| \leq |\mathcal{H}| - k$
- Since  $f \in \mathcal{H}$  and  $f$  never makes mistakes,  $|\mathcal{H}_t| \geq 1$  for each  $t$
- Hence  $|\mathcal{H}| - k \geq |\mathcal{H}_t| \geq 1$ , which is equivalent to  $k \leq |\mathcal{H}| - 1$

# Halving learner

Prove that the halving learner makes at most  $\log_2(|\mathcal{H}|)$  mistakes

## Halving learner

- Initialize  $\mathcal{H}_1 \leftarrow \mathcal{H}$
- For  $t = 1, 2, \dots$ 
  - 1 Get  $x_t$
  - 2 Pick MAJORITY( $h(x_t) : h \in \mathcal{H}_t$ )
  - 3 Get  $y_t = f(x_t)$  and update  $\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : h(x_t) = y_t\}$

# Halving learner

- Assume the algorithm makes a mistake in round  $t$

# Halving learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypotheses used for prediction will **not** be part of  $\mathcal{H}_{t+1}$

# Halving learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypotheses used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t|/2$

# Halving learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypotheses used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t|/2$
- After  $k$  mistakes,  $|\mathcal{H}_t| \leq |\mathcal{H}|/2^k$

# Halving learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypotheses used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t|/2$
- After  $k$  mistakes,  $|\mathcal{H}_t| \leq |\mathcal{H}|/2^k$
- Since  $|\mathcal{H}_t| \geq 1$  for all  $t$ , we have  $|\mathcal{H}|/2^k \geq |\mathcal{H}_t| \geq 1$

# Halving learner

- Assume the algorithm makes a mistake in round  $t$
- Then the hypotheses used for prediction will **not** be part of  $\mathcal{H}_{t+1}$
- It follows that  $|\mathcal{H}_{t+1}| \leq |\mathcal{H}_t|/2$
- After  $k$  mistakes,  $|\mathcal{H}_t| \leq |\mathcal{H}|/2^k$
- Since  $|\mathcal{H}_t| \geq 1$  for all  $t$ , we have  $|\mathcal{H}|/2^k \geq |\mathcal{H}_t| \geq 1$
- Solving for  $k$  yields  $k \leq \log_2(|\mathcal{H}|)$

# Memorizing predictor

Consider the following predictor (assuming  $\mathcal{Y} = \{0, 1\}$ )

Note that this predictor fails on **any** input  $x$  not in  $S$  such that  $f(x) = 1$

## Memorizing predictor

- Get  $x$
- Return  $y_i$  if there exists  $(x_i, y_i)$  in  $S$  such that  $x = x_i$ , else 0

Show that there exists a polynomial  $p_S$  that determines the memorizing predictor by defining  $h(x) = 1$  whenever  $p_S(x) \geq 0$

# Memorizing predictor

Make  $p_S$  equal to 0 in each data point  $(x_i, y_i)$  s.t.  $y_i = 1$ , else  $< 0$

# Memorizing predictor

Make  $p_S$  equal to 0 in each data point  $(x_i, y_i)$  s.t.  $y_i = 1$ , else  $< 0$

For a single data point  $(x_1, y_1)$  such that  $y_1 = 1$ ,

$$p_S(x) = -\|x - x_1\|^2$$

# Memorizing predictor

Make  $p_S$  equal to 0 in each data point  $(x_i, y_i)$  s.t.  $y_i = 1$ , else  $< 0$

For a single data point  $(x_1, y_1)$  such that  $y_1 = 1$ ,

$$p_S(x) = -\|x - x_1\|^2$$

For multiple data points,

$$p_S(x) = - \prod_{i \in [m]: y_i=1} \|x - x_i\|^2$$

# Expected training loss

Fix some hypothesis  $h \in \mathcal{H}$ . Show that the expected value of  $L_S(h)$  equals  $L_{\mathcal{D},f}(h)$ , i.e. that

$$\mathbb{E}_{S \sim \mathcal{D}}[L_S(h)] = L_{\mathcal{D},f}(h)$$

# Expected training loss

We can write the expected training loss as

# Expected training loss

We can write the expected training loss as

$$\mathbb{E}_{S \sim \mathcal{D}}[L_S(h)] = \mathbb{E}_{S \sim \mathcal{D}} \left\{ \frac{1}{m} \sum_{i=1}^m \ell(h(x_i), f(x_i)) \right\}$$

# Expected training loss

We can write the expected training loss as

$$\left\{ \begin{aligned} \mathbb{E}_{S \sim \mathcal{D}}[L_S(h)] &= \mathbb{E}_{S \sim \mathcal{D}} \left\{ \frac{1}{m} \sum_{i=1}^m \ell(h(x_i), f(x_i)) \right\} \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{x_i \sim \mathcal{D}} \{ \ell(h(x_i), f(x_i)) \} \end{aligned} \right.$$

# Expected training loss

We can write the expected training loss as

$$\left\{ \begin{aligned} \mathbb{E}_{S \sim \mathcal{D}}[L_S(h)] &= \mathbb{E}_{S \sim \mathcal{D}} \left\{ \frac{1}{m} \sum_{i=1}^m \ell(h(x_i), f(x_i)) \right\} \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{x_i \sim \mathcal{D}} \{ \ell(h(x_i), f(x_i)) \} \\ &= \frac{1}{m} \sum_{i=1}^m L_{\mathcal{D}, f}(h) \end{aligned} \right.$$

# Expected training loss

We can write the expected training loss as

$$\left\{ \begin{aligned} \mathbb{E}_{S \sim \mathcal{D}}[L_S(h)] &= \mathbb{E}_{S \sim \mathcal{D}} \left\{ \frac{1}{m} \sum_{i=1}^m \ell(h(x_i), f(x_i)) \right\} \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{x_i \sim \mathcal{D}} \{ \ell(h(x_i), f(x_i)) \} \\ &= \frac{1}{m} \sum_{i=1}^m L_{\mathcal{D},f}(h) \\ &= \frac{m}{m} L_{\mathcal{D},f}(h) = L_{\mathcal{D},f}(h) \end{aligned} \right.$$