From Basic Machine Learning models to Advanced Kernel Learning

Université Grenoble Alpes, Inria

General information

Teachers: Julien Mairal, Pierre Gaillard, Michael Arbel

Website: `https://kernel-learning.github.io/`

The class will last 36 hours.

Final grade: 50% final exam, 50% homework.

Pre-requesites

Linear algebra (matrix operations, linear systems)

Probability (e.g. notion of random variables, conditional expectation)

Basic coding skills in python for homeworks

Rule

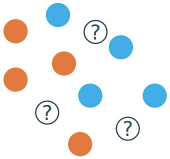
QUESTIONS if something is unclear

Supervised learning Basics

Supervised learning

Goal: Given some observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \dots, n$ of inputs/outputs, we want to predict a new output $y \in \mathcal{Y}$ given a new input $x \in \mathcal{X}$.

We usually refer to training data for $(x_i, y_i), i = 1, \dots, n$ and testing data for the unobserved data (x, y).



Examples:

- **Inputs** $x \in \mathcal{X}$: images, sounds, video, text, proteins, sequence of DNA bases, web pages, social network activity, sensors from industry, financial or meteorological,...
  In this class we will assume that $\mathcal{X}$ is a vector space (such as $\mathbb{R}^d$)
- **Outputs** $y \in \mathcal{Y}$: binary labels $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$, multiclass classification $\mathcal{Y} = \{1, \dots, K\}$, or regression $\mathcal{Y} = \mathbb{R}$.

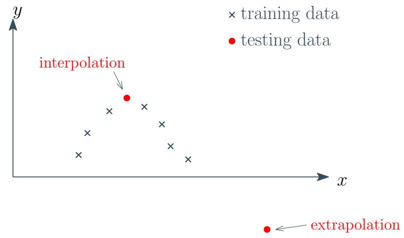
Why is supervised learning difficult? (1)

- The output y may not be a deterministic function of x. It is noisy. When $y \in \mathbb{R}$ we will often make the "additive noise" assumption:

$$y = f(x) + \varepsilon$$

  with zero-mean noise ε.

- The regression function f may be complex: non linear if $\mathcal{X}$ is a vector space or even hard to define otherwise.
- Only few $x's$ are observed:



May lead to overfitting.

Why is supervised learning difficult? (2)

- The input space $\mathcal{X}$ may be very large $\rightarrow$ computational or statistical issues. Curse of dimensionality.

- Difference between data at training time and testing time.

- The criterion of performance is not always well defined.

Machine learning algorithm

Assumption: In standard supervised learning analysis, $\{(x_i, y_i)\}_{1 \leqslant i \leqslant n} \in (\mathcal{X} \times \mathcal{Y})^n$ is the realization of i.i.d. random variables.

A machine learning algorithm $\mathcal{A}$ is a function that goes from a dataset in $(\mathcal{X} \times \mathcal{Y})^n$ to a function from $\mathcal{X}$ to $\mathcal{Y}$:

$$\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^n \quad \mapsto \quad \mathcal{Y}^{\mathcal{X}}$$

⚠ In some situation, the algorithm may be asked to provide a prediction in a "decision space" $\mathcal{Z}$ that may differ from the output space $\mathcal{Y}$.

Example: predict the probability of rain. In this case $\mathcal{Z} = [0, 1]$ but $\mathcal{Y} = \{-1, 1\}$.

Random design v.s. fixed design

Random design: both x and y are assumed random and sampled i.i.d.

Fixed design: $x_1, \ldots, x_n$ are assumed to be deterministic.
- either because they are indeed deterministic (regular grid of the input space)
- or because the analysis is performed conditionnally on them.

This difference will play an important role in the class on least-square regression.

Loss function

To evaluate the performance of the algorithm, one needs to choose a loss function depending on the problem we want to solve:

$$\ell : \mathcal{Y} \times \mathcal{Z} \mapsto \mathbb{R}$$

where $\ell(y, z)$ is the loss of predicting z while the true label is y.

In some communities, the loss is replaced with a utility (economics) or a reward (RL) to be maximized.

Examples:

- binary classification: $\ell(y, z) = \mathbb{1}\{y \neq z\}$ and $\mathcal{Y} = \mathcal{Z} = \{-1, 1\}$
- multiclass classification: $\ell(y, z) = \mathbb{1}\{y \neq z\}$ and $\mathcal{Y} = \mathcal{Z} = \{1, \dots, k\}$
- regression: $\ell(y, z) = (y - z)^2$ and $\mathcal{Y} = \mathcal{Z} = \mathbb{R}$
- structured prediction, survival analysis, ranking,...

We will assume that the loss is given to us.

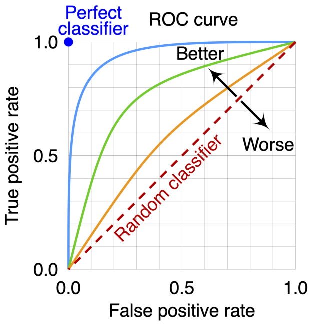
Loss function: multi-objective

The loss will be used to evaluate the final performance of the algorithm.

Often a single number may not be enough to charaterize the entire prediction behavior: multi-objective optimization

Examples:

- minimizing cost v.s. maximizing comfort while buying a car
- maximizing performance v.s. minimizing fuel consumption and emission of pollutants of a vehicle
- Minimizing cost v.s. flight duration while buying a ticket
- False positive v.s. false negative in classification



Expected risk

Given a loss function $\ell : \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}$, the expected risk (or generalization error or testing error) of a function $f : \mathcal{X} \to \mathcal{Z}$ is defined as

$$\mathcal{R}(f) := \mathbb{E}\big[\ell(y, f(x))\big]$$

⚠ **Pay attention to what is random:**

- the expectation is taken over the randomness of (x, y) and thus $\mathcal{R}$ depends on its distribution.
- if f is the output of an algorithm (i.e., that was learnt on data (x_i, y_i)), it is random because of the dependence on the training data and so is $\mathcal{R}(f)$
- if f is a deterministic function, $\mathcal{R}(f)$ is deterministic.
- the function $\mathcal{R}$ is itself deterministic
- if the algorithm provides random predictions, then the expectation is also taken over the randomness of the prediction.

Empirical risk

The empirical risk of a function $f : \mathcal{X} \to \mathcal{Z}$ is its average loss over the training data:

$$\widehat{\mathcal{R}}(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) \,.$$

$\widehat{\mathcal{R}}$ is a random function on functions because it depends on the training set.

⚠ It is often applied on random functions that also depend on the training data. Pay attention to the dependencies.

Examples

In (multi-class) classification,

- the expected risk of the zero-one loss is the probability of making a mistake on the testing data.
- the empirical risk is the proportion of mistakes on the training data

	Least square regression	**Classification**
$\mathcal{A} = \mathcal{Y}$	$\mathbb{R}$	$\{1, \dots, k\}$
$\ell(y, z)$	$(y - z)^2$	$\mathbb{1}_{y-z}$
$\mathcal{R}(f)$	$\mathbb{E}\big[(f(x) - y)^2\big]$	$\mathbb{P}(f(x) \neq y)$
$\widehat{\mathcal{R}}(f)$	$\frac{1}{n}\sum_{i=1}^n (f(x_i) - y_i)^2$	$\frac{1}{n}\sum_{i=1}^n \mathbb{1}_{y_i \neq f(x_i)}$

Bayes predictor and Bayes risk (1)

We would like to answer to the question:

— What is the best prediction function f if we know the distribution of (x, y)? —

Using the conditional expectation, we can rewrite the expected risk

$$\mathcal{R}(f) = \mathbb{E}\big[\ell(y, f(x))\big] = \mathbb{E}_x\Big[\mathbb{E}\big[\ell(y, f(x))\big|x\big]\Big] =: \mathbb{E}_x\Big[r(f(x)|x)\Big]$$

where $r(z|x) = \mathbb{E}[\ell(y, z)|x]$ is the conditional risk of any prediction $z \in \mathcal{Z}$ given the input $x \in \mathcal{X}$.

To minimize $\mathcal{R}(f)$, we can minimize $r(f(x)|x)$ for all $x \in \mathcal{X}$ independently.

Bayes predictor and Bayes risk (2)

Proposition

The expected risk is minimized at a Bayes predictor $f^* : \mathcal{X} \to \mathcal{Y}$ satisfying for all $x \in \mathcal{X}$

$$f^*(x) \in \operatorname*{arg\,min}_{z \in \mathcal{Y}} \mathbb{E}\big[\ell(y,z)|x\big] = \operatorname*{arg\,min}_{z \in \mathcal{Y}} r(z|x)\,.$$

The Bayes risk $\mathcal{R}^*$ is the risk of all Bayes predictors

$$\mathcal{R}^* := \mathbb{E}_x\Big[\inf_{z \in \mathcal{Y}} \mathbb{E}\big[\ell(y,z)|x\big]\Big]$$

The Bayes predictor is not unique but they all have the same Bayes risk. It is usually non zero.

Excess risk: $\mathcal{R}(f) - \mathcal{R}^*$. It is always non-negative.

The goal of supervised learning is to estimate the distribution $y|x$ to minimize the excess risk.

Example: Bayes risk for classification with zero-one loss

Consider binary classification $\mathcal{Y} = \{0, 1\}$ with zero-one loss $\ell(y, z) = \mathbb{1}\{y \neq z\}$.

A Bayes predictor is defined for all $x \in \mathcal{X}$ by

$$
\begin{aligned}
f^*(x) &\in \underset{z \in \{0,1\}}{\arg\min}\, \mathbb{E}\big[\mathbb{1}\{z \neq y\}|x\big] \\
&= \underset{z \in \{0,1\}}{\arg\min}\, \mathbb{P}\big(z \neq y|x\big) \\
&= \underset{z \in \{0,1\}}{\arg\max}\, \mathbb{P}\big(z = y|x\big) \\
&= \begin{cases} 1 & \text{if } \eta(x) > 1/2 \\ 0 & \text{if } \eta(x) < 1/2 \\ 0 \text{ or } 1 & \text{if } \eta(x) = 1/2 \end{cases} ,
\end{aligned}
$$

where $\eta(x) := \mathbb{P}(y = 1|x)$.

Then, the Bayes risk equals

$$
\mathcal{R}^* = \mathbb{E}\big[\mathbb{P}(f^*(x) \neq y|x)\big] = \mathbb{E}\big[\min\{\eta(x), 1 - \eta(x)\}\big] .
$$

Bayes risk for regression

Consider least-square regression with $\mathcal{Y} = \mathcal{Z} = \mathbb{R}$ and $\ell(y, z) = (y - z)^2$.

A Bayes predictor is given for all $x \in \mathcal{X}$ by

$$\begin{aligned} f^*(x) &\in \arg\min_{z \in \mathbb{R}} \mathbb{E}\left[(y - z)^2 | x\right] \\ &= \arg\min_{z \in \mathbb{R}} \left\{ \mathbb{E}\left[(y - \mathbb{E}[y|x])^2\right] + (z - \mathbb{E}[y|x])^2 \right\} \\ &= \mathbb{E}[y|x]. \end{aligned}$$

The Bayes risk is thus the conditional variance

$$\mathcal{R}^* = \mathbb{E}\left[(y - \mathbb{E}[y|x])^2\right].$$

Exercise (1)

Exercise (Bayes risk for non symmetric loss)

We consider binary classification with $\mathcal{Y} = \{-1, 1\}$ with the loss function

$$\ell(y,z) = \begin{cases} 0 & \text{if } y = z & \leftarrow \text{good prediction} \\ c_- & \text{if } y = -1 \text{ and } z = +1 & \leftarrow \text{false positive} \\ c_+ & \text{if } y = +1 \text{ and } z = -1 & \leftarrow \text{false negative} \end{cases}.$$

Compute the Bayes estimator at $x \in \mathcal{X}$ and the Bayes risk as a function of $\mathbb{E}[y|x]$.

"It's not the most sophisticated Spam blocker I've tried, but it's the only one that works!"

Solution (1)

Let $z \in \{-1, 1\}$. Then, the expected loss is

$$\begin{aligned}\mathbb{E}[\ell(y,z)|x] &= c_- \mathbb{P}(y=-1|x)\mathbb{1}\{z=+1\} + c_+ \mathbb{P}(y=+1|x)\mathbb{1}\{z=-1\} \\ &= c_-(1-\mathbb{P}(y=1|x))\mathbb{1}\{z=+1\} + c_+ \mathbb{P}(y=+1|x)\mathbb{1}\{z=-1\}\end{aligned}$$

But,

$$\mathbb{E}[y|x] = \mathbb{P}(y=1|x) - \mathbb{P}(y=-1|x) = 2\mathbb{P}(y=1|x) - 1,$$

which yields

$$\mathbb{P}(y=1|x) = \frac{1+\mathbb{E}[y|x]}{2}.$$

Thus,

$$\mathbb{E}[\ell(y,z)|x] = \begin{cases} \frac{c_-}{2}\big(1-\mathbb{E}[y|x]\big) & \text{if } z=+1 \\ \frac{c_+}{2}\big(1+\mathbb{E}[y|x]\big) & \text{if } z=-1 \end{cases}$$

The Bayes risk is the minimum of these two values and the Bayes predictor the argmin.

Exercise (2)

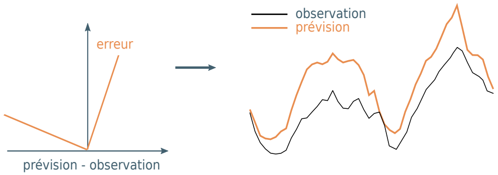
Exercise (Bayes risk for absolute loss)

1. *What is the Bayes predictor for regression with absolute loss*

$$\ell(y, z) = |y - z|\,?$$

2. *What is the Bayes predictor for the pinball loss*

$$\ell(y, z) = (\tau - \mathbb{1}\{y \leqslant z\})(y - z)\ ?$$



Solution (2)

Let $p(y|x)$ be the density of y given x. Then,

$$\begin{aligned}\mathbb{E}\left[\ell(y,z)|x\right] &= \mathbb{E}\left[|y-z||x\right] \\ &= \int_{-\infty}^{z}(z-y)p(y|x)dy + \int_{z}^{+\infty}(y-z)p(y|x)dy.\end{aligned}$$

Cancelling the derivative in z yields

$$\int_{-\infty}^{f^*(x)} p(y|x)dy - \int_{f^*(x)}^{\infty} p(y|x)dy = 0$$

which entails

$$\int_{-\infty}^{f^*(x)} p(y|x)dy = \frac{1}{2}.$$

The Bayes predictor $f^*(x) = \arg\min_{z\in\mathbb{R}} \mathbb{E}\left[|y-z|\,\middle|x\right]$ is the median of $y|x$.

Supervised learning theory (summary)

Some data $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is distributed according to a probability distribution P.

We observe training data $D_n := \big\{(x_1, y_1), \dots, (x_n, y_n)\big\}$.

We must form prediction into a decision set $\mathcal{A}$ by choosing a prediction function

$$f : \underbrace{\mathcal{X}}_{\text{observation}} \to \underbrace{\mathcal{A}}_{\text{decision}}$$

Our performance is measured by a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}$. We define the expected risk

$$\mathcal{R}(f) := \mathbb{E}\big[\ell\big(f(x), y\big)\big] \qquad = \quad \text{expected loss of } f$$

Goal: minimize $\mathcal{R}(f)$ by approaching the performance of the oracle $f^* = \arg\min_{f \in \mathcal{F}} \mathcal{R}(f)$

	Least square regression	**Classification**
$\mathcal{A} = \mathcal{Y}$	$\mathbb{R}$	$\{1, \dots, k\}$
$\ell(y, z)$	$(y - z)^2$	$\mathbb{1}_{z \neq y}$
$R(f)$	$\mathbb{E}\big[(f(x) - y)^2\big]$	$\mathbb{P}(f(x) \neq y)$
f^*	$\mathbb{E}[y \mid x]$	$\arg\max_k \mathbb{P}(y = k \mid x)$

Empirical risk minimization

Idea: estimate $\mathcal{R}(f)$ thanks to the training data with the empirical risk

$$\underbrace{\widehat{\mathcal{R}}(f) := \frac{1}{n}\sum_{i=1}^{n}\ell\big(f(x_i), y_i\big)}_{\text{average error on training data}} \quad \approx \quad \underbrace{\mathcal{R}(f) = \mathbb{E}\big[\ell\big(f(x), y\big)\big]}_{\text{expected error}}$$

We estimate $\widehat{f}_n$ by minimizing the empirical risk

$$\widehat{f}_n \in \arg\min_{f\in\mathcal{F}} \widehat{\mathcal{R}}(f)\,.$$

Many methods are based on empirical risk minimization: ordinary least square, logistic regression, Ridge, Lasso,. . .

Risk decomposition

Choosing the right model: $\mathcal{F}$ is a set of models which needs to be properly chosen

$$\mathcal{R}(\widehat{f}_n) - \mathcal{R}^* = \underbrace{\min_{f\in\mathcal{F}} \mathcal{R}(f) - \mathcal{R}^*}_{\text{Approximation error}} \quad + \quad \underbrace{\mathcal{R}(\widehat{f}_n) - \min_{f\in\mathcal{F}} \mathcal{R}(f)}_{\text{Estimation error}}$$

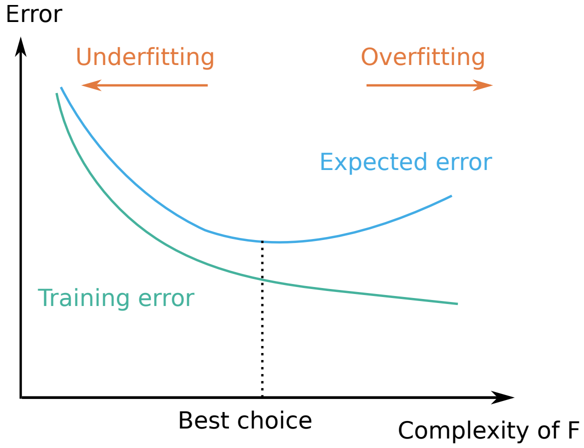
Approximation error

- Always non-negative.
- Depends on the class $\mathcal{F}$ but is independent of $\widehat{f}_n$ and n.
- Goes to zero when $\mathcal{F}$ grows and can approximate any measurable function.
- It is not random.

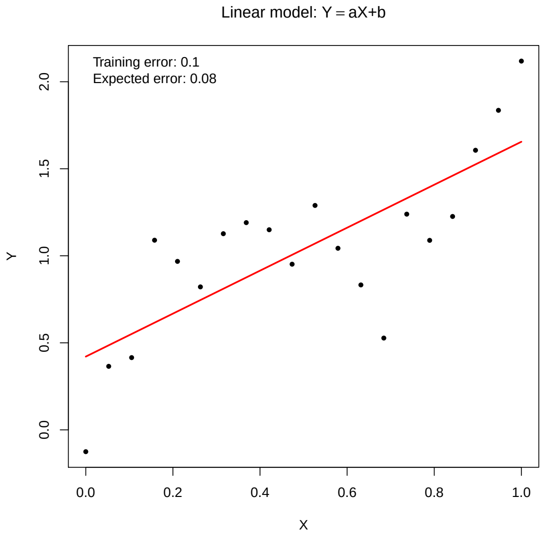
Estimation error

- Always non-negative.
- Depends on $\widehat{f}_n$ and thus on (x_i, y_i). Thus, it is random.
- Typically decreasing with n and increasing with the compexity of $\mathcal{F}$.

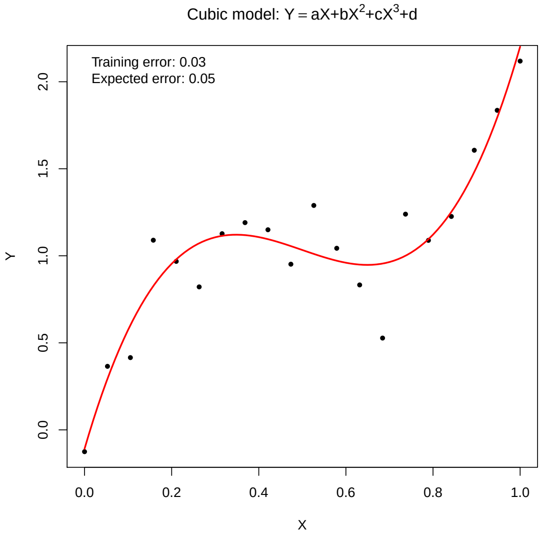
Overfitting



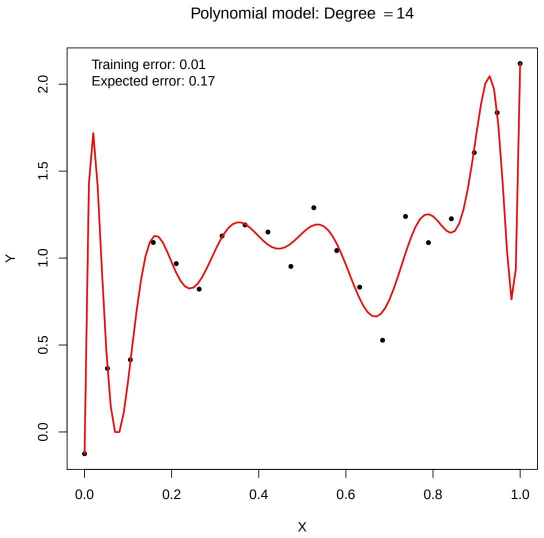
Overfitting: example in regression



Overfitting: example in regression



Overfitting: example in regression

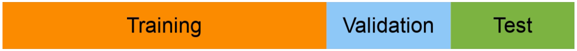


Practical performance evaluation

Usually the data is split into three parts:

- training set: to train algorithms and estimate parameters
- validation set: to estimate hyper-parameters
- testing set: to evaluate the final performance

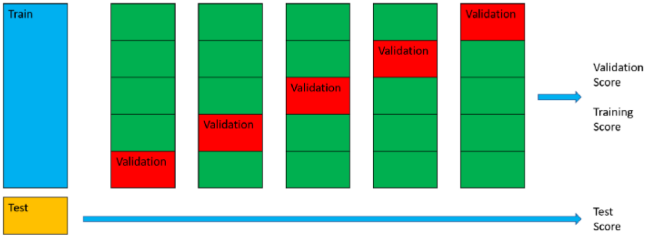
⚠ The test set can only be used once at the very end!



Practical performance evaluation: Cross-validation

Cross-validation:

- randomly break data into K groups
- for each group, use it as a test set and train the data on the $(K-1)$ other groups



We choose the parameter with the smallest average error on the test sets.

👍 only $1/K$ of the data lost for training

👎 K times more expensive

In practice: choose $K \approx 10$.

Measures of performance: Expected error

We recall that an algorithm $\mathcal{A}$ is a function that takes a dataset $D_n = \{(x_i, y_i)\}_{1 \leqslant i \leqslant n} \in (\mathcal{X} \times \mathcal{Y})^n$ and return a function from $\mathcal{X}$ to $\mathcal{Z}$. Its performance is measured by its expected risk:

$$\mathcal{R}(\mathcal{A}(D_n)) = \mathbb{E}\Big[\ell(y, \mathcal{A}(D_n)(x)) \Big| D_n\Big],$$

where the expectation is taken over (x, y).

Measures of performance: Expected error

We recall that an algorithm $\mathcal{A}$ is a function that takes a dataset $D_n = \{(x_i, y_i)\}_{1 \leqslant i \leqslant n} \in (\mathcal{X} \times \mathcal{Y})^n$ and return a function from $\mathcal{X}$ to $\mathcal{Z}$. Its performance is measured by its expected risk:

$$\mathcal{R}(\mathcal{A}(D_n)) = \mathbb{E}\Big[\ell(y, \mathcal{A}(D_n)(x)) \Big| D_n\Big],$$

where the expectation is taken over (x, y).

⚠ The risk of an algorithm $\mathcal{R}(\mathcal{A}(D_n))$ is random because it depends on D_n, which is itself random.

Measures of performance: Expected error

We recall that an algorithm $\mathcal{A}$ is a function that takes a dataset $D_n = \{(x_i, y_i)\}_{1 \leqslant i \leqslant n} \in (\mathcal{X} \times \mathcal{Y})^n$ and return a function from $\mathcal{X}$ to $\mathcal{Z}$. Its performance is measured by its expected risk:

$$\mathcal{R}(\mathcal{A}(D_n)) = \mathbb{E}\Big[\ell(y, \mathcal{A}(D_n)(x)) \Big| D_n\Big],$$

where the expectation is taken over (x, y).

⚠ The risk of an algorithm $\mathcal{R}(\mathcal{A}(D_n))$ is random because it depends on D_n, which is itself random.

Expected error

$$\mathbb{E}\Big[\mathcal{R}(\mathcal{A}(D_n))\Big] = \mathbb{E}\Big[\ell(y, \mathcal{A}(D_n)(x))\Big],$$

where the expectation is taken over (x, y) and D_n.

Measures of performance: Expected error

We recall that an algorithm $\mathcal{A}$ is a function that takes a dataset $D_n = \{(x_i, y_i)\}_{1 \leqslant i \leqslant n} \in (\mathcal{X} \times \mathcal{Y})^n$ and return a function from $\mathcal{X}$ to $\mathcal{Z}$. Its performance is measured by its expected risk:

$$\mathcal{R}(\mathcal{A}(D_n)) = \mathbb{E}\left[\ell(y, \mathcal{A}(D_n)(x)) \middle| D_n\right],$$

where the expectation is taken over (x, y).

⚠ The risk of an algorithm $\mathcal{R}(\mathcal{A}(D_n))$ is random because it depends on D_n, which is itself random.

Expected error

$$\mathbb{E}\left[\mathcal{R}(\mathcal{A}(D_n))\right] = \mathbb{E}\left[\ell(y, \mathcal{A}(D_n)(x))\right],$$

where the expectation is taken over (x, y) and D_n.

Consistency in expectation: $\mathbb{E}\left[\mathcal{R}(\mathcal{A}(D_n))\right] \to \mathcal{R}^*$ as $n \to \infty$.

Measures of performance: Probably Approximately Correct (PAC)

PAC Learning

For some $\delta \in (0, 1)$ and $\varepsilon > 0$

$$\mathbb{P}\Big(\mathcal{R}(\mathcal{A}(D_n)) - \mathcal{R}^* \leqslant \varepsilon\Big) \geqslant 1 - \delta$$

The goal is to find ε as small as possible as a function of δ.

Universal Consistency

Universal consistency

An algorithm is called universally consistent if it is consistent in expectation

$$\mathbb{E}\big[\mathcal{R}(D_n)\big] \to \mathcal{R}^*$$

for all distributions of (x, y) and (x_i, y_i).

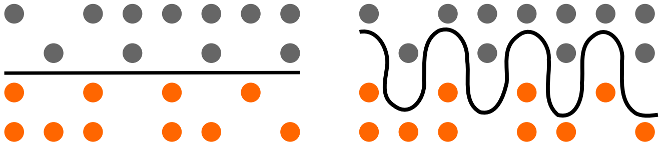
We will see some methods that are universally consistent in this class.

No free lunch theorem

There exists no universal learning algorithm that would outperform all other algorithms on all possible tasks:

- without assumptions, learning can be arbitrarily slow
- the optimal learning algorithm is always task-dependent
- for every algorithm, one can find task on which is performs well and task for which it perfoms poorly.

Which method is the best on the following example?

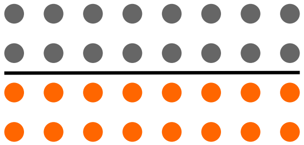


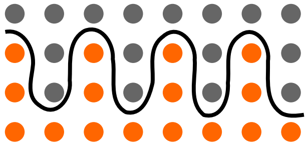
No free lunch theorem

There exists no universal learning algorithm that would outperform all other algorithms on all possible tasks:

- without assumptions, learning can be arbitrarily slow
- the optimal learning algorithm is always task-dependent
- for every algorithm, one can find task on which is performs well and task for which it perfoms poorly.

Which method is the best on the following example? It depends on the distribution!





Next class

We will study linear regression which dates back to Gauss and Legendre around 1800!
It minimizes the empirical risk

$$\frac{1}{n}\sum_{i=1}^{n}\big(y_i - \langle \theta, x_i\rangle\big)^2$$

But still very important nowadays in particular to understand kernel methods.

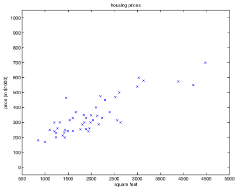
DATA

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

$$(x_1, y_1), \ldots, (x_n, y_n)$$

Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

example taken from Coursera



References

F. Bach. "Learning Theory from First Principles". 2022.

L. Bottou, F. E. Curtis, and J. Nocedal. "Optimization methods for large-scale machine learning". In: *arXiv preprint arXiv:1606.04838* (2016).

T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.