

Fundamentals of machine learning

Courses 1 & 2: basics of machine learning

Titouan Vayer & Mathurin Massias

email: titouan.vayer@inria.fr, mathurin.massias@inria.fr,

January 15, 2025



ENS DE LYON

Full course outline

From theory ...

1. Basics of machine learning
2. Decision theory & statistical learning
3. (Penalized) Linear models
4. Dimension reduction
5. Kernels and support vector machines
6. Ensembles methods
7. Clustering, density estimation
8. Neural networks
9. Advanced neural networks
10. Density estimation

... to practice

We will use Python notebooks and scikit-learn



Some references

[Shai Shalev-Shwartz and Shai Ben-David \(2014\).](#)

Understanding Machine Learning - From Theory to Algorithms. [Cambridge University Press](#)

[Francis Bach \(2022\).](#) *Learning Theory from First Principles.*

[Trevor Hastie, Robert Tibshirani, and Jerome Friedman \(2001\).](#) *The Elements of Statistical Learning.* [Springer New York Inc.](#)

Evaluation

- ▶ 50 % final exam
- ▶ 50 % report (jupyter notebooks) on practical sessions
 - ▶ Each TD has practical lab: to do at home.
 - ▶ Report in notebook should be send after 4 TDs
 - ▶ Can be done in groups of ≈ 2 (max 3).

Python installations

- ▶ The practical sessions of the course will require to run jupyter notebooks.
- ▶ We recommend that you install python through the Anaconda distribution (python 3.7, 3.8 or 3.9 is preferable) available at <https://www.anaconda.com/products/distribution>

You should check that you are able to create and open a jupyter notebook, and inside, run the following imports:

```
1 import matplotlib
2 import numpy
3 import sklearn
4 import pytorch
5 import pandas
6 import scipy
```

If any of these packages is missing, it can be installed with 'conda install numpy', the command being run in a terminal or in Anaconda prompt for Windows user.

Basics of machine learning

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

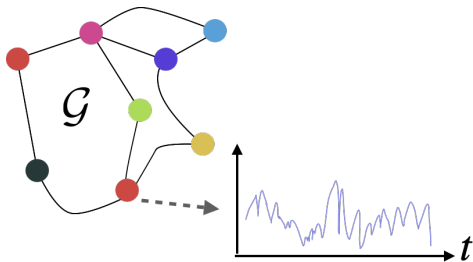
- No free-lunch theorem

Conclusion

What is machine learning ?

Some applications

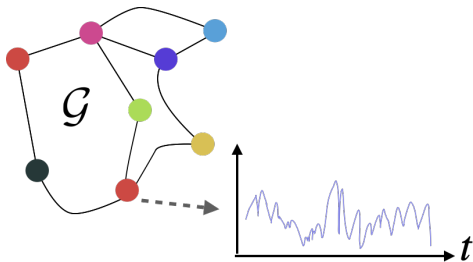
1. Energy networks, disease propagation



What is machine learning ?

Some applications

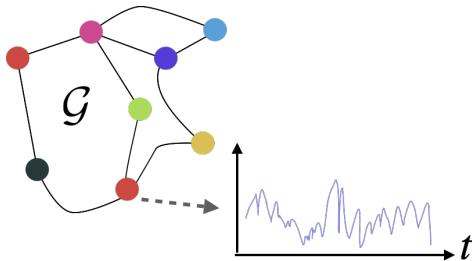
1. Energy networks, disease propagation
2. Image analysis (medical application, web)



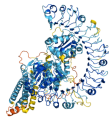
What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding [Jumper et al. 2021](#)



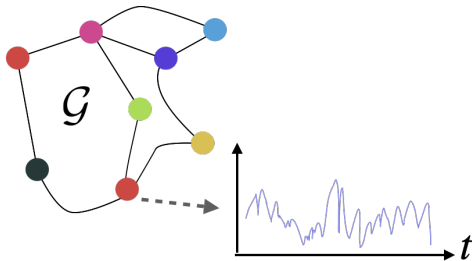
AAATGCG.... - - - - ->



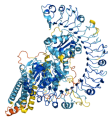
What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding [Jumper et al. 2021](#)
4. Generative models <https://stablediffusionweb.com/>



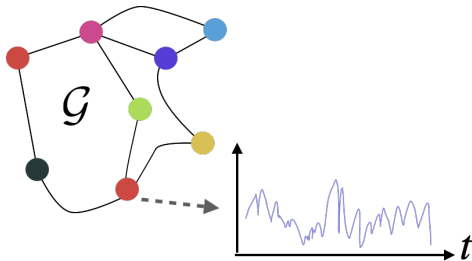
AAATGCG.... - - - - ->



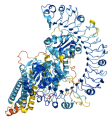
What is machine learning ?

Some applications

1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding [Jumper et al. 2021](#)
4. Generative models <https://stablediffusionweb.com/>
5. Natural language processing <https://chat.openai.com/chat>



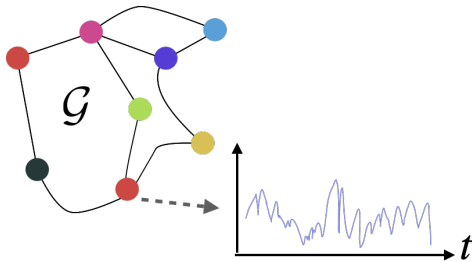
AAATGCG.... - - - - ->



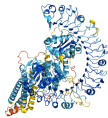
What is machine learning ?

Some applications

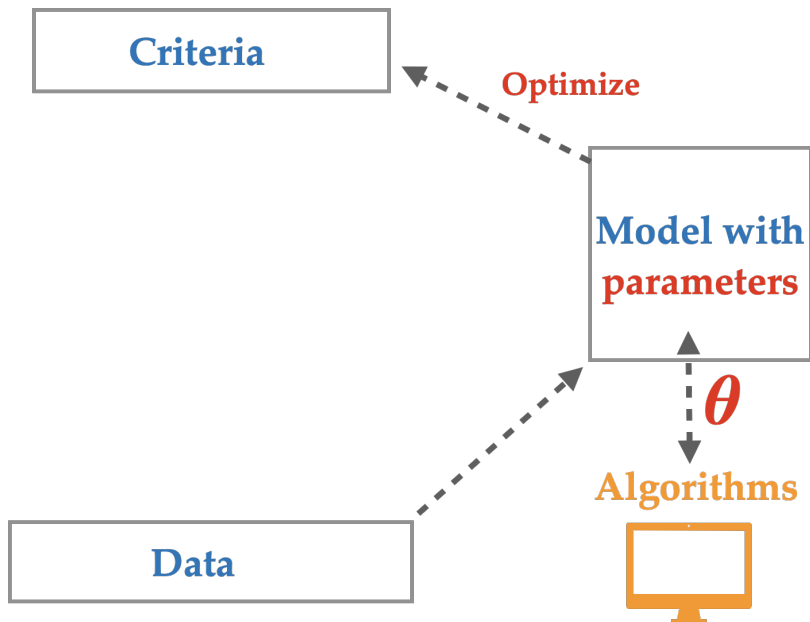
1. Energy networks, disease propagation
2. Image analysis (medical application, web)
3. Protein folding [Jumper et al. 2021](#)
4. Generative models <https://stablediffusionweb.com/>
5. Natural language processing <https://chat.openai.com/chat>
6. For art <https://www.youtube.com/watch?v=MwtVkPKx3RA>



AAATGCG.... - - - - ->



What is machine learning ?



What is machine learning ?

The objective of machine learning

Teach a machine to process automatically a some data in order to solve a given problem.

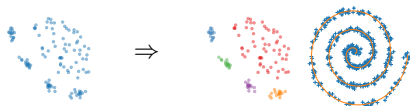
What is machine learning ?

The objective of machine learning

Teach a machine to process automatically a some data in order to solve a given problem.

Unsupervised learning: understanding the data

- ▶ Clustering & probability density estimation
- ▶ Dimensionality reduction



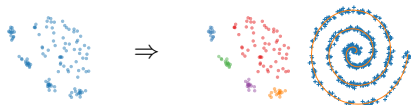
What is machine learning ?

The objective of machine learning

Teach a machine to process automatically a some data in order to solve a given problem.

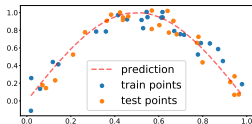
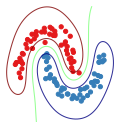
Unsupervised learning: understanding the data

- ▶ Clustering & probability density estimation
- ▶ Dimensionality reduction



Supervised learning: learning to predict

- ▶ Classification: classify points according to some labels
- ▶ Regression: predict real (vector) values



Some images and slides have been obtained by the courtesy of [Rémi Flamary](#)

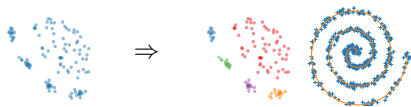
What is machine learning ?

The objective of machine learning

Teach a machine to process automatically a some data in order to solve a given problem.

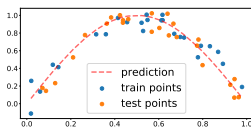
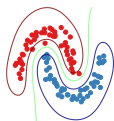
Unsupervised learning: understanding the data

- ▶ Clustering & probability density estimation
- ▶ Dimensionality reduction



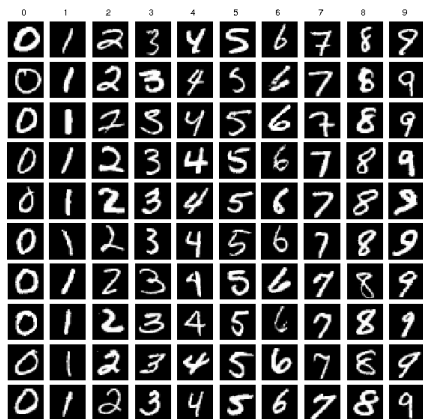
Supervised learning: learning to predict

- ▶ Classification: classify points according to some labels
- ▶ Regression: predict real (vector) values



Not covered: reinforcement learning *i.e.* train a machine to choose actions that maximize a reward (games, autonomous vehicles, control).

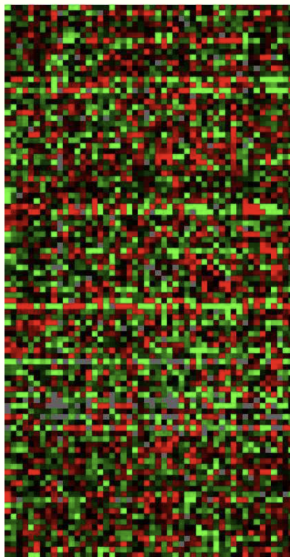
What is machine learning ?



Supervised classification examples

- ▶ e.g. to identify the numbers on images from a 16×16 gray level image (image classification)
- ▶ SPAM, fraud detection, disease classification ...

What is machine learning ?

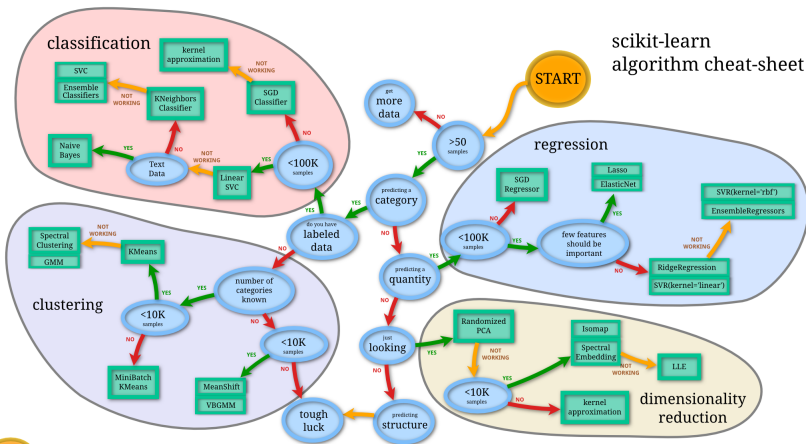


Clustering example

- ▶ Analyse n sequences (individuals) of d genetical responses
- ▶ Groups of similar samples ? Gene with similar expressions ?

Find your ML method

scikit-learn
algorithm cheat-sheet



https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion

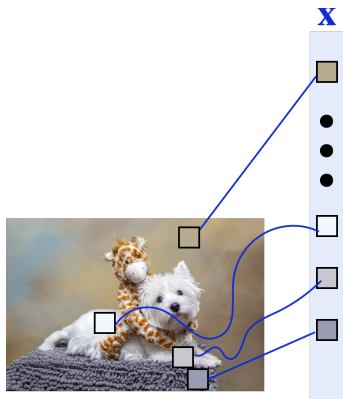
Store a data point

Vectorial representation

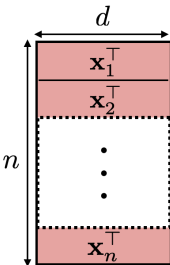
One “sample”, “data point”, “individual”:

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

- ▶ d is the dimension, x_i is the i th information i of \mathbf{x}
- ▶ Can describe information about an individual
- ▶ For an image \mathbf{x} : each pixel of an image
- ▶ Descriptors of a cell, word embedding ...



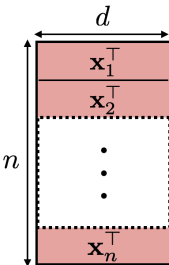
Unsupervised dataset

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$$



Unsupervised learning

- ▶ The dataset contains the samples $(\mathbf{x}_i)_{i=1}^n$ where n is the number of samples of size d .
- ▶ d and n define the dimensionality of the learning problem.
- ▶ Data stored as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ that contains the training samples as rows.

Unsupervised dataset

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$$


Unsupervised learning

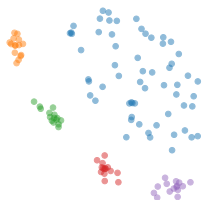
- ▶ The dataset contains the samples $(\mathbf{x}_i)_{i=1}^n$ where n is the number of samples of size d .
- ▶ d and n define the dimensionality of the learning problem.
- ▶ Data stored as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ that contains the training samples as rows.
- ▶  in ML vectors are sometimes described **in row instead of column**

Supervised dataset

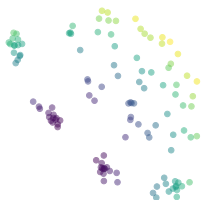
Samples + labels:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Classification



Regression



Supervised learning

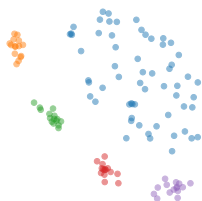
- ▶ The dataset contains the samples $(\mathbf{x}_i, y_i)_{i=1}^n$ where \mathbf{x}_i is the feature sample and $y_i \in \mathcal{Y}$ its label.
- ▶ The values to predict (label) can be concatenated in a vector $\mathbf{y} \in \mathcal{Y}^n$

Supervised dataset

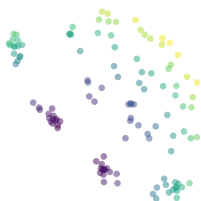
Samples + labels:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Classification



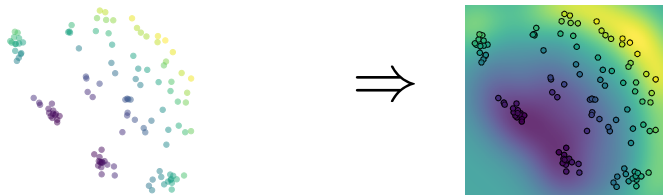
Regression



Supervised learning

- ▶ The dataset contains the samples $(\mathbf{x}_i, y_i)_{i=1}^n$ where \mathbf{x}_i is the feature sample and $y_i \in \mathcal{Y}$ its label.
- ▶ The values to predict (label) can be concatenated in a vector $\mathbf{y} \in \mathcal{Y}^n$
- ▶ Semi-supervised learning: few labeled points are available, but a large number of unlabeled points are given.

Regression

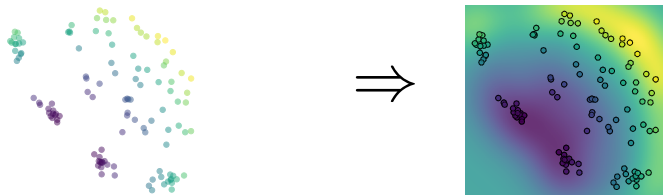


Objective

$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a continuous value ($\mathcal{Y} = \mathbb{R}$).
- ▶ Can be extended to multi-value prediction ($\mathcal{Y} = \mathbb{R}^p$).

Regression



Objective

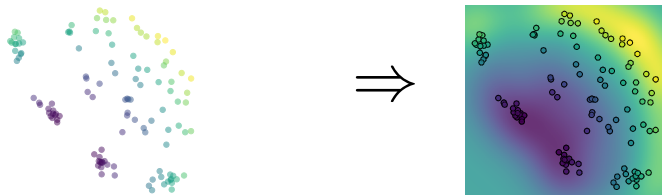
$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a continuous value ($\mathcal{Y} = \mathbb{R}$).
- ▶ Can be extended to multi-value prediction ($\mathcal{Y} = \mathbb{R}^p$).

Hyperparameters

- ▶ Type of function (linear, kernel, neural network).
- ▶ Performance measure.
- ▶ Regularization.

Regression



Objective

$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \mathbb{R}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a continuous value ($\mathcal{Y} = \mathbb{R}$).
- ▶ Can be extended to multi-value prediction ($\mathcal{Y} = \mathbb{R}^p$).

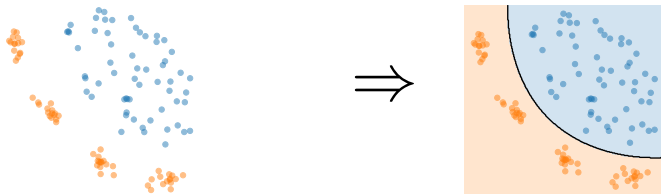
Hyperparameters

- ▶ Type of function (linear, kernel, neural network).
- ▶ Performance measure.
- ▶ Regularization.

Methods

- ▶ Least Square (LS).
- ▶ Ridge regression, Lasso.
- ▶ Kernel regression.
- ▶ Deep learning.

Binary classification

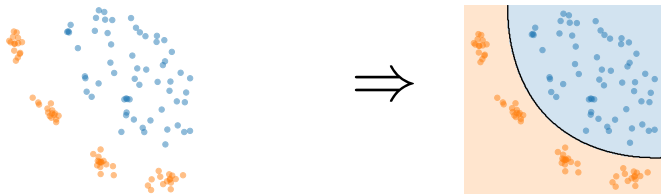


Objective

$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \{-1, 1\} \text{ or } \{0, 1\}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value.
- ▶ $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

Binary classification



Objective

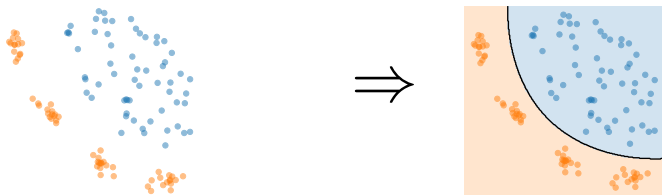
$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \{-1, 1\} \text{ or } \{0, 1\}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value.
- ▶ $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

Hyperparameters

- ▶ Type of function (linear, kernel, neural network).
- ▶ Performance measure.
- ▶ Regularization.

Binary classification



Objective

$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \{-1, 1\} \text{ or } \{0, 1\}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value.
- ▶ $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.

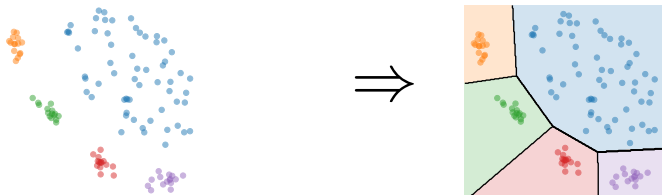
Hyperparameters

- ▶ Type of function (linear, kernel, neural network).
- ▶ Performance measure.
- ▶ Regularization.

Methods

- ▶ Bayesian classifier (LDA, QDA)
- ▶ Linear and kernel discrimination
- ▶ Decision trees, random forests.
- ▶ Deep learning.

Multiclass classification



Objective

$$(\mathbf{x}_i, y_i)_{i=1}^n \Rightarrow f : \mathbb{R}^d \rightarrow \{1, \dots, K\}$$

- ▶ Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting an integer value ($\mathcal{Y} = \{1, \dots, K\}$).
- ▶ In practice K continuous score functions f_k are estimated and the prediction is

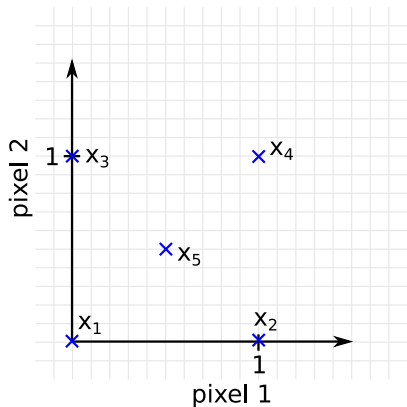
$$f(\mathbf{x}) = \arg \max_k f_k(\mathbf{x})$$

Two toy examples

Examples



Positions in 2D



Two toy examples

Examples



Labels

Class 1 ×

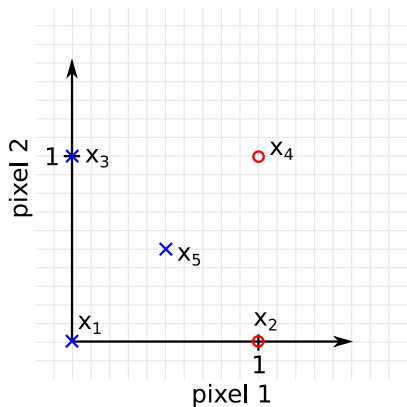
Class 2 ○

Class 1 ×

Class 2 ○

Class 1 ×

Positions in 2D



Two toy examples

Examples



Labels

Class 1 ×

Class 2 ○

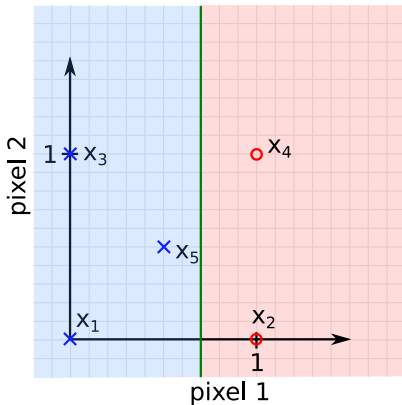
Class 1 ×

Class 2 ○

Class 1 ×

 Classifier

Positions in 2D



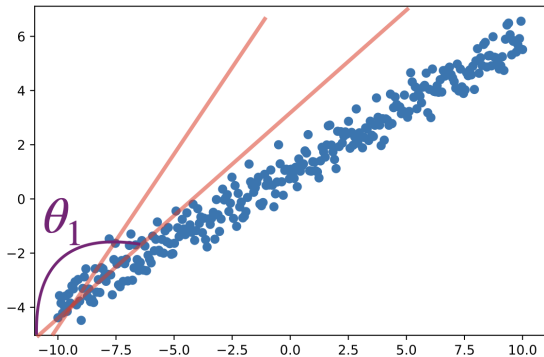
Two toy examples

Linear regression

In 1D: $x_j \in \mathbb{R}, y_j \in \mathbb{R}$. Goal: find the linear function $f : \mathbb{R} \rightarrow \mathbb{R}$ that “best predicts” y_j from x_j .

$$\mathbf{y} \approx \theta_1 \mathbf{X} + \theta_2$$

y_1	x_1
\vdots	\vdots
y_j	x_j
\vdots	\vdots
y_n	x_n



Two toy examples

Linear regression

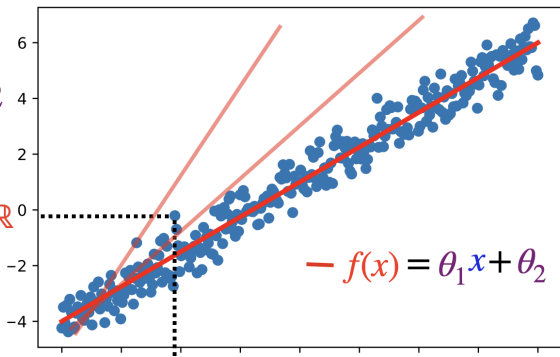
In 1D: $x_j \in \mathbb{R}, y_j \in \mathbb{R}$. Goal: find the linear function $f : \mathbb{R} \rightarrow \mathbb{R}$ that “best predicts” y_j from x_j .

$$\mathbf{y} \approx \theta_1 \mathbf{X} + \theta_2$$

y_1
⋮
 y_j
⋮
 y_n

x_1
⋮
 x_j
⋮
 x_n

$y_j \in \mathbb{R}$



Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

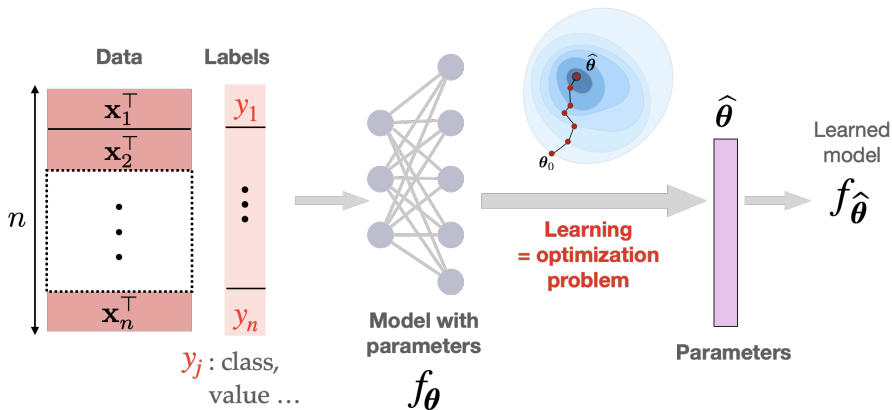
Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion

The big picture of (parametrized) ML

How to find this function ?



The goal in the **learning step** will be to find the parameters $\hat{\theta}$ (hence the function) that **minimizes a measure of error on the data**

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ so that:

$$\ell(\text{true value}, \text{predicted value}) = \text{how good is my prediction}$$

Regression problems

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ so that:

$$\ell(\text{true value}, \text{predicted value}) = \text{how good is my prediction}$$

Regression problems

- ▶ E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ so that:

$$\ell(\text{true value}, \text{predicted value}) = \text{how good is my prediction}$$

Regression problems

- ▶ E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)
- ▶ E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_2^2$ (square loss)

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ so that:

$$\ell(\text{true value}, \text{predicted value}) = \text{how good is my prediction}$$

Regression problems

- ▶ E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)
- ▶ E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_2^2$ (square loss)
- ▶ E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_q^q$ (ℓ_q loss)

Loss functions

Supervised case

A loss function is $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ so that:

$$\ell(\text{true value}, \text{predicted value}) = \text{how good is my prediction}$$

Regression problems

- ▶ E.g. $y_i \in \mathbb{R}$ $\ell(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$ (square loss)
- ▶ E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_2^2$ (square loss)
- ▶ E.g. $\mathbf{y}_i \in \mathbb{R}^p$ $\ell(y_i, f(\mathbf{x}_i)) = \|\mathbf{y}_i - f(\mathbf{x}_i)\|_q^q$ (ℓ_q loss)

Classification problems

- ▶ E.g. $y_i \in \{-1, 1\}$ $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i \neq f(\mathbf{x}_i)}$ (0/1 loss)

Loss functions

A focus on classification problems $\mathcal{Y} = \{-1, 1\}$

$$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) \text{ with } \Phi \text{ non-increasing.}$$

Loss functions

A focus on classification problems $\mathcal{Y} = \{-1, 1\}$

$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i))$ with Φ non-increasing.

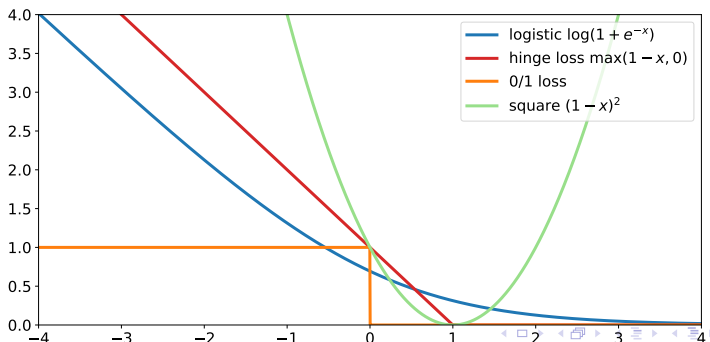
- ▶ $y_i f(\mathbf{x}_i)$ is **the margin** (on the board).
- ▶ $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ (0/1 loss)
- ▶ $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ (hinge loss: **SVM**)
- ▶ $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ (logistic loss)

Loss functions

A focus on classification problems $\mathcal{Y} = \{-1, 1\}$

$$\ell(y_i, f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) \text{ with } \Phi \text{ non-increasing.}$$

- ▶ $y_i f(\mathbf{x}_i)$ is the **margin** (on the board).
- ▶ $\ell(y_i, f(\mathbf{x}_i)) = \mathbf{1}_{y_i f(\mathbf{x}_i) \leq 0}$ (0/1 loss)
- ▶ $\ell(y_i, f(\mathbf{x}_i)) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ (hinge loss: **SVM**)
- ▶ $\ell(y_i, f(\mathbf{x}_i)) = \log(1 + e^{-y_i f(\mathbf{x}_i)})$ (logistic loss)



Loss functions

Cross-entropy

When f predicts a probability of belonging to a class.

- ▶ When $y_i \in \{0, 1\}$, $f : \mathbb{R}^d \rightarrow [0, 1]$

$$\ell(y_i, f(\mathbf{x}_i)) = -y_i \log f(\mathbf{x}_i) - (1 - y_i) \log (1 - f(\mathbf{x}_i)). \quad (1)$$

- ▶ When $y_i \in \{1, \dots, K\}$ and $f : \mathbb{R}^d \rightarrow [0, 1]^K$.

- ▶ We do "one-hot encoding" of the labels $(y_i)_{i \in [n]} \rightarrow \mathbf{Y} \in \{0, 1\}^{n \times K}$.
with $f(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$.

$$\ell(y_i, f(\mathbf{x}_i)) = - \sum_{k=1}^K Y_{i,k} \log(f_k(\mathbf{x}_i)). \quad (2)$$

Sigmoid and softmax

- ▶ First case: in practice $f(\mathbf{x}) = \sigma(g(\mathbf{x}))$ where $g : \mathbb{R}^d \rightarrow \mathbb{R}$ and

$$\sigma(z) = \frac{\exp(z)}{1 + \exp(z)}.$$

- ▶ Second case: $f_j(\mathbf{x}) = \frac{\exp(g_j(\mathbf{x}))}{\sum_{k=1}^K \exp(g_k(\mathbf{x}))}$ where $g : \mathbb{R}^d \rightarrow \mathbb{R}^K$.

Empirical risk minimization

Minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) \quad (\text{ERM})$$

Empirical risk minimization

Minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) \quad (\text{ERM})$$

- ▶ It is called **empirical risk minimization (ERM)**
- ▶ Given the loss, finds the “best” f on the training data
- ▶ E.g. linear regression
- ▶ Same idea applies for unsupervised problem (reconstruction error)

Empirical risk minimization

Parametrized models

- ▶ In practice we do ERM with parametrized model $f = f_{\theta}$

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\theta}(\mathbf{x}_i)) \quad (\text{ERM})$$

Examples

- ▶ Most classical example: linear least-squares regression (course 3)

$$\frac{1}{n} \sum_{i=1}^n (y_i - \theta^{\top} \mathbf{x}_i)^2$$

- ▶ For classification $\mathcal{Y} = \{-1, +1\}$ (see courses 3/5)

Logistic regression:

$$\frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i \theta^{\top} \mathbf{x}_i))$$

Support vector machine:

$$\frac{1}{n} \sum_{i=1}^n \max\{0, 1 - y_i \theta^{\top} \mathbf{x}_i\}$$

Empirical risk minimization

Parametrized models

- ▶ In practice we do ERM with parametrized model $f = f_{\theta}$

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\theta}(\mathbf{x}_i)) \quad (\text{ERM})$$

Examples

- ▶ Most classical example: linear least-squares regression (course 3)

$$\frac{1}{n} \sum_{i=1}^n (y_i - \theta^{\top} \mathbf{x}_i)^2$$

- ▶ For classification $\mathcal{Y} = \{-1, +1\}$ (see courses 3/5)

Logistic regression:

$$\frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i \theta^{\top} \mathbf{x}_i))$$

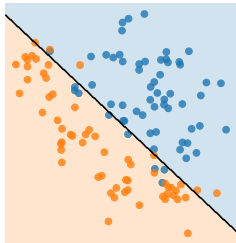
Support vector machine:

$$\frac{1}{n} \sum_{i=1}^n \max\{0, 1 - y_i \theta^{\top} \mathbf{x}_i\}$$

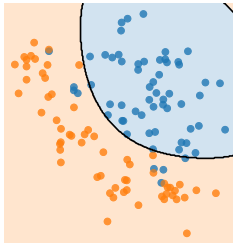
Once solved how do I know if my model is good ?

Underfitting and overfitting

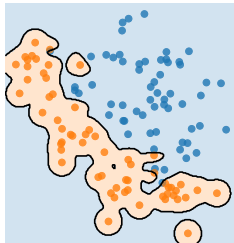
Acc. 0.89/0.89 train/test



Acc. 0.93/0.92 train/test

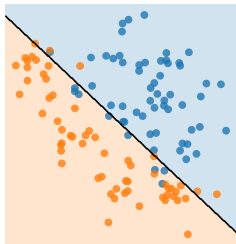


Acc. 0.98/0.88 train/test

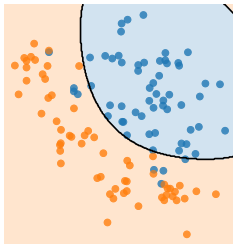


Underfitting and overfitting

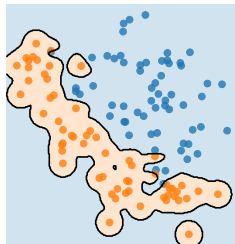
Acc. 0.89/0.89 train/test



Acc. 0.93/0.92 train/test



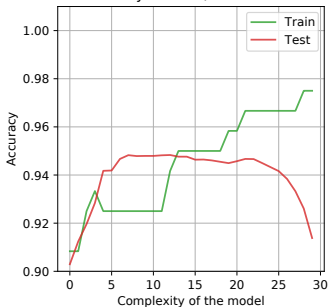
Acc. 0.98/0.88 train/test



Complexity of a model

- ▶ Under-fitting when the model is too simple.
- ▶ Over-fitting occurs when the model is too complex
- ▶ Training data performance is not a good proxy for testing performance.
- ▶ We want to predict well on new data!
- ▶ Parameter and model validation.

Accuracy on train/test datasets



Empirical risk minimization

Train by minimizing the train error

To find f the idea is to **minimize the averaged error** on the training samples:

$$\min_f \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i)) + \lambda \text{Reg}(f) \quad (\text{ERM})$$

- ▶ It is called **empirical risk minimization (ERM)**
- ▶ Given the loss, finds the “best” f on the training data
- ▶ Teacher/student analogy
- ▶ Same idea applies for unsupervised problem (minimizing reconstruction error)

... but we want generalization !

- ▶ We want f to be good outside the training samples
- ▶ Add regularization ! (limit the complexity of f)
- ▶ But another problem: **how to choose λ ?**

Parameters vs hyperparameters

Hyperparameters

- ▶ Are parameters that have to be **selected/chosen** to define a model.
- ▶ Used for the configuration of the model.
- ▶ They are **not learned** on the data !
- ▶ Examples (1): regularization strength λ , number of neighbors k in k -NN, number of trees, number of iterations for an algorithm...
- ▶ Example (2): but also all the data pipeline (normalization...)

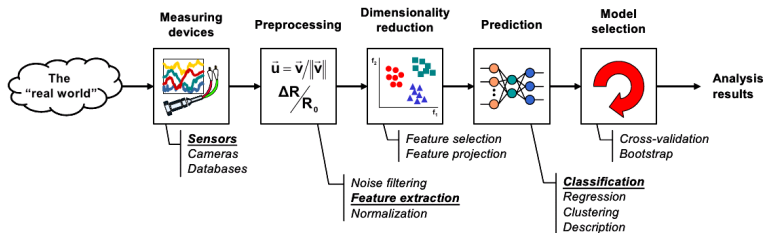
Parameters

- ▶ **Are learned** on the data.
- ▶ θ in linear regression, tree cuts, $\mathbf{U} \in \mathbb{R}^{d \times k}$ in dimension reduction (PCA), centroids in k -means, weights of the neural network...

In scikit-learn

```
1 from sklearn import Model
2 hyperparameters = ...
3 clf = Model(hyperparameters) # define the
  model
4 cv.fit(X ,y) #train the model
```


Machine learning in practice



- ▶ **Data acquisition** : sensor, databases, manual or automatic labeling
- ▶ **Pre-processing** : denoising, formatting, numerical conversion, normalization
- ▶ **Feature extraction** : manual when prior knowledge, feature selection dimensionality reduction
- ▶ **Model estimation** : classification, regression, clustering.
- ▶ **Validation** : model and parameter selection.
- ▶ **Analysis** : performance, uncertainty, interpretation of the model.

Features extraction, selection and model estimation can be done simultaneously (deep learning, sparse models).

Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !**

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion

Model selection and validation

Bias-complexity tradeoff

generalization error = estimation error + approximation error

Select a model that is not too complex but not too simple !

Model selection and validation

Bias-complexity tradeoff

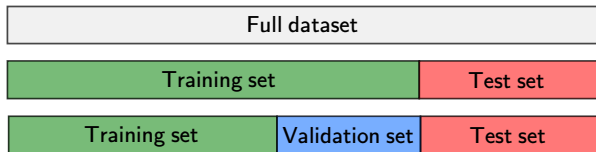
generalization error = estimation error + approximation error

Select a model that is not too complex but not too simple !

General principle

- ▶ Estimate the generalization error **on data not seen during training**
- ▶ Is a rough estimate of the test error
- ▶ Select the model with the lowest “approximate” test error

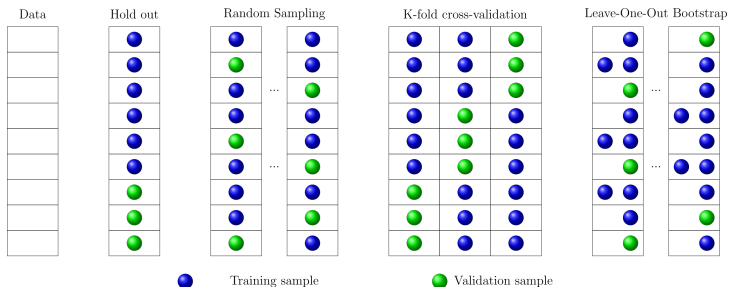
Splitting the data



Principle of Hold-Out cross-validation

- ▶ Split the training data in a training and validation sets (non overlapping).
- ▶ Train different models (different methods and/or hyperparameters) on the train data.
- ▶ Evaluate performance on the validation data and select the method/parameters with best performance.
- ▶ **Validation set acts as a proxy of test data**
- ▶ But only one split is a poor proxy !

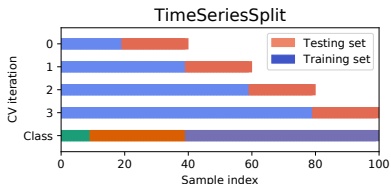
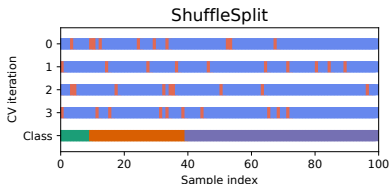
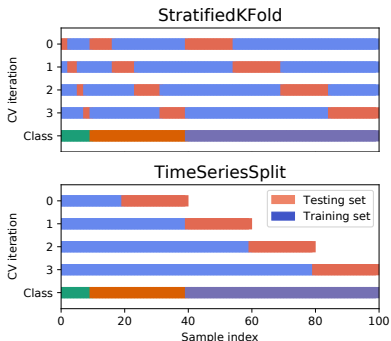
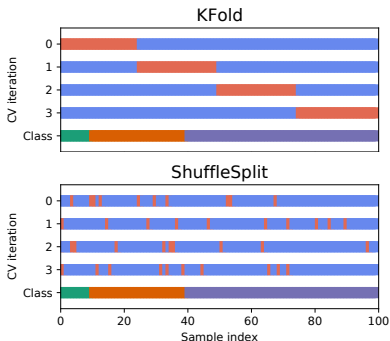
Different ways to split the data



Cross-validation Arlot and Celisse 2010

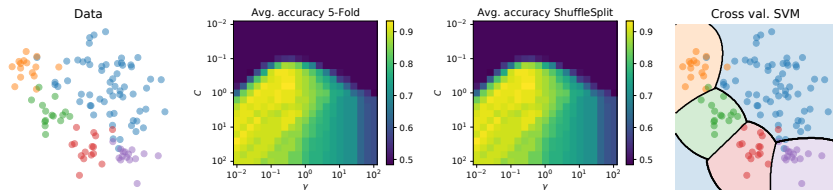
- ▶ The training data is split in non-overlapping training/validation sets.
- ▶ **Hold-Out** uses a unique split and computes the performance on the validation set.
- ▶ More robust cross-validation approaches actually investigate several splits of the data and compute the average performance:
 - ▶ **K-fold** (split in K sets and use one split as test for all k)
 - ▶ Random sampling (aka **Shuffle split**) draws several random splittings.
- ▶ Scikit-learn implementation : `sklearn.model_selection.cross_validate`

Data splitting with Scikit-learn



- ▶ Scikit-learn implements iterator classes for data split in `sklearn.model_selection`.
- ▶ KFold is the classical K-fold cross-validation.
- ▶ StratifiedKFold ensures a data split that preserves the proportion of classes.
- ▶ ShuffleSplit randomly selects a proportion of the samples for train/validation.

Validation with Scikit-learn



Principle

- ▶ GridSearchCV takes a model and a grid of hyperparameters as input and performs cross-validation.
- ▶ Number of splits and type of data splitting can be chosen.
- ▶ For large number of parameters complexity is exponential, RandomizedSearchCV can be more efficient.

Python code

```
1 from sklearn.svm import SVC
2 from sklearn.model_selection import
  GridSearchCV
3
4 ngrid=21
5 clf = SVC()
6 param_grid={'C':np.logspace(-2,2,ngrid),
7             'gamma':np.logspace(-2,2,ngrid),}
8
9 cv = GridSearchCV(clf,param_grid)
10
11 cv.fit(xn,y)
12
13 # recover best parameters and estimators
14 clf_opt = cv.best_estimator_
15 params_opt = cv.best_params_
```


Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion

Risk and empirical risk

Training data

- ▶ We have access to n r.v. $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \sim p$ and loss.
- ▶ Randomness is the key for a mathematical analysis.
- ▶ $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ is the **unknown** data distribution.

Risk and empirical risk

Training data

- ▶ We have access to n r.v. $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \sim p$ and loss.
- ▶ Randomness is the key for a mathematical analysis.
- ▶ $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ is the **unknown** data distribution.

The (expected) risk

- ▶ For a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ defined as:

$$\mathcal{R}_p(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p} [\ell(y, f(\mathbf{x}))]$$

- ▶ **Generalization error**: with the true (unknown) data distrib.

Risk and empirical risk

Training data

- ▶ We have access to n r.v. $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \sim p$ and loss.
- ▶ Randomness is the key for a mathematical analysis.
- ▶ $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ is the **unknown** data distribution.

The (expected) risk

- ▶ For a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ defined as:

$$\mathcal{R}_p(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p} [\ell(y, f(\mathbf{x}))]$$

- ▶ **Generalization error**: with the true (unknown) data distrib.

The empirical risk

- ▶ Averaged error on the empirical distribution of the data

$$\hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, f(\mathbf{x}_i))$$

- ▶ $\hat{\mathcal{R}}(f)$ is a **random quantity**: ML is statistics !

Risk and empirical risk

(one) Ultimate goal of ML

- ▶ Find a function f^* that minimizes the (expected) risk:

$$f^* \in \arg \min_{f: \mathcal{X} \rightarrow \mathcal{Y}} \mathcal{R}_p(f)$$

Risk and empirical risk

(one) Ultimate goal of ML

- ▶ Find a function f^* that minimizes the (expected) risk:

$$f^* \in \arg \min_{f: \mathcal{X} \rightarrow \mathcal{Y}} \mathcal{R}_p(f)$$

- ▶ Good news ! there is one (TD1): *the Bayes predictor*

$$f^*(\mathbf{x}_0) = \arg \min_{y' \in \mathcal{Y}} \mathbb{E}[\ell(y, y') | \mathbf{x} = \mathbf{x}_0] = \int_{\mathcal{Y}} \ell(y, y') d p(y | \mathbf{x}_0)$$

- ▶ Given a supervised learning problem, the Bayes risk $\mathcal{R}^* = \mathcal{R}_p(f^*)$ is the optimal performance.

Risk and empirical risk

(one) Ultimate goal of ML

- ▶ Find a function f^* that minimizes the (expected) risk:

$$f^* \in \arg \min_{f: \mathcal{X} \rightarrow \mathcal{Y}} \mathcal{R}_p(f)$$

- ▶ Good news ! there is one (TD1): *the Bayes predictor*

$$f^*(\mathbf{x}_0) = \arg \min_{y' \in \mathcal{Y}} \mathbb{E}[\ell(y, y') | \mathbf{x} = \mathbf{x}_0] = \int_{\mathcal{Y}} \ell(y, y') d p(y | \mathbf{x}_0)$$

- ▶ Given a supervised learning problem, the Bayes risk $\mathcal{R}^* = \mathcal{R}_p(f^*)$ is the optimal performance.

Excess risk

- ▶ We **cannot calculate** f^* but we would like to get closer to it
- ▶ For a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ the excess risk is:

$$\mathcal{R}(f) - \mathcal{R}^* \geq 0$$

Risk decomposition

How to obtain guarantees ?

- ▶ Practical machine learning ERM:

$$\hat{\theta} \in \arg \min_{\theta \in \Theta} \hat{\mathcal{R}}(f_{\theta}) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\theta}(\mathbf{x}_i))$$

- ▶ Question: what is the excess risk of $f_{\hat{\theta}}$?

$$\mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}^* = ?$$

Risk decomposition

How to obtain guarantees ?

- ▶ Practical machine learning ERM:

$$\hat{\theta} \in \arg \min_{\theta \in \Theta} \hat{\mathcal{R}}(f_{\theta}) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\theta}(\mathbf{x}_i))$$

- ▶ Question: what is the excess risk of $f_{\hat{\theta}}$?

$$\mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}^* = ?$$

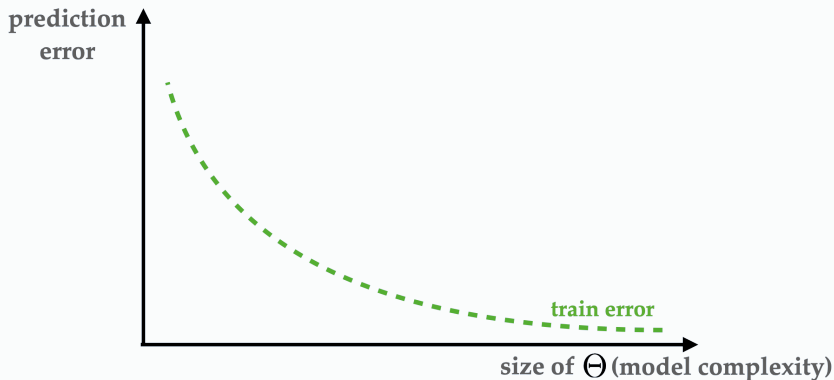
Bias-complexity tradeoff (see TD2)

$$\mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}^* = \text{estimation error} + \text{approximation error}$$

Risk decomposition

Bias-complexity tradeoff

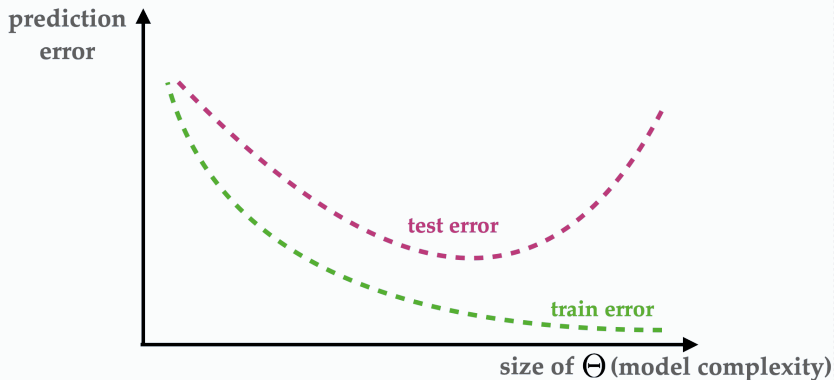
excess risk = estimation error + approximation error



Risk decomposition

Bias-complexity tradeoff

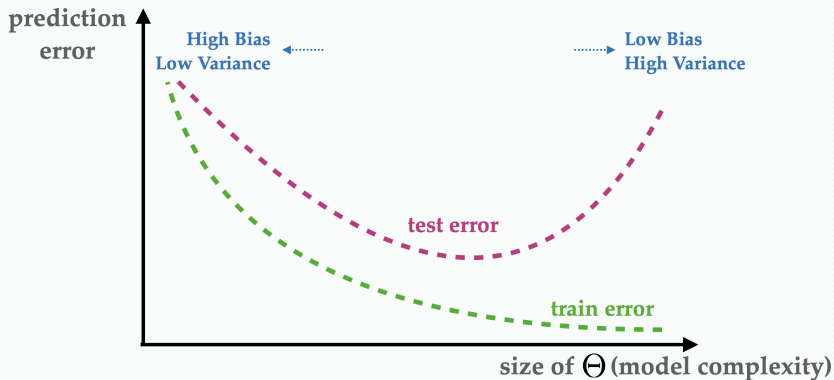
$$\text{excess risk} = \text{estimation error} + \text{approximation error}$$



Risk decomposition

Bias-complexity tradeoff

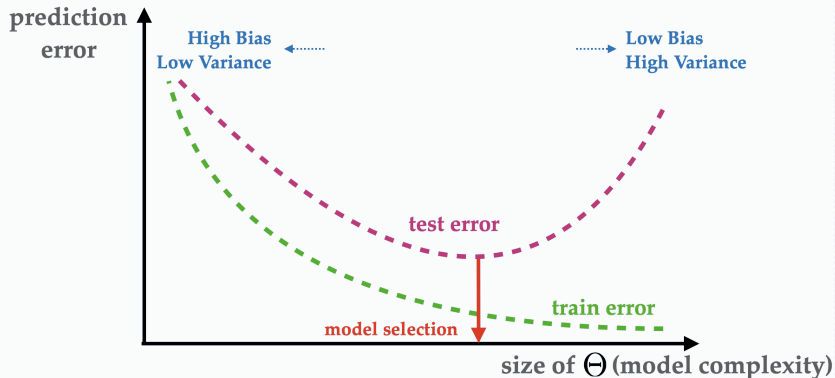
$$\text{excess risk} = \text{estimation error} + \text{approximation error}$$



Risk decomposition

Bias-complexity tradeoff

$$\text{excess risk} = \text{estimation error} + \text{approximation error}$$



Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion

Objective

Remember: minimize the risk = find the Bayes predictor.

- ▶ Regression with square loss: $f^*(\mathbf{x}_0) = \mathbb{E}[y|\mathbf{x} = \mathbf{x}_0]$
- ▶ Classification with 0/1 loss: $f^*(\mathbf{x}_0) = \arg \max_{y \in \mathcal{Y}} p(y' = y|\mathbf{x} = \mathbf{x}_0)$
- ▶ Generally: $f^*(\mathbf{x}_0) = \arg \min_{y' \in \mathcal{Y}} \mathbb{E}[\ell(y, y')|\mathbf{x} = \mathbf{x}_0]$

Problem: we don't know the conditional data distribution $p(y|\mathbf{x})$!

Objective

Remember: minimize the risk = find the Bayes predictor.

- ▶ Regression with square loss: $f^*(\mathbf{x}_0) = \mathbb{E}[y|\mathbf{x} = \mathbf{x}_0]$
- ▶ Classification with 0/1 loss: $f^*(\mathbf{x}_0) = \arg \max_{y \in \mathcal{Y}} p(y' = y|\mathbf{x} = \mathbf{x}_0)$
- ▶ Generally: $f^*(\mathbf{x}_0) = \arg \min_{y' \in \mathcal{Y}} \mathbb{E}[\ell(y, y')|\mathbf{x} = \mathbf{x}_0]$

Problem: we don't know the conditional data distribution $p(y|\mathbf{x})$!

Local averaging methods

Main idea: find an estimation $\hat{p}(y|\mathbf{x})$ of $p(y|\mathbf{x})$

- ▶ Regression: $\hat{f}(\mathbf{x}_0) = \int y d\hat{p}(y|\mathbf{x}_0)$
- ▶ Classification: $\hat{f}(\mathbf{x}_0) = \arg \max_{y \in \mathcal{Y}} \hat{p}(y' = y|\mathbf{x} = \mathbf{x}_0)$

Objective

Remember: minimize the risk = find the Bayes predictor.

- ▶ Regression with square loss: $f^*(\mathbf{x}_0) = \mathbb{E}[y|\mathbf{x} = \mathbf{x}_0]$
- ▶ Classification with 0/1 loss: $f^*(\mathbf{x}_0) = \arg \max_{y \in \mathcal{Y}} p(y' = y|\mathbf{x} = \mathbf{x}_0)$
- ▶ Generally: $f^*(\mathbf{x}_0) = \arg \min_{y' \in \mathcal{Y}} \mathbb{E}[\ell(y, y')|\mathbf{x} = \mathbf{x}_0]$

Problem: we don't know the conditional data distribution $p(y|\mathbf{x})$!

Local averaging methods

Main idea: find an estimation $\hat{p}(y|\mathbf{x})$ of $p(y|\mathbf{x})$

- ▶ Regression: $\hat{f}(\mathbf{x}_0) = \int y d\hat{p}(y|\mathbf{x}_0)$
- ▶ Classification: $\hat{f}(\mathbf{x}_0) = \arg \max_{y \in \mathcal{Y}} \hat{p}(y' = y|\mathbf{x} = \mathbf{x}_0)$

Benefits/inconvenients

- ▶ + **No optimization problem** & well understood [Györfi et al. 2002](#)
- ▶ ± Simple estimators (easy to find but simple)
- ▶ – Not very adaptive to the regularity of the function (curse of dim)

Approximate the conditional distribution

“Linear” estimator

The estimator of $p(y|\mathbf{x})$ has the form:

$$\hat{p}(y|\mathbf{x}) = \sum_{i=1}^n \omega_i(\mathbf{x}) \delta_{y_i}(y)$$

- ▶ $\omega_i(\mathbf{x})$ are weights, $\forall \mathbf{x}, \omega_i(\mathbf{x}) \geq 0$ and $\sum_{i=1}^n \omega_i(\mathbf{x}) = 1$
- ▶ $\omega_i : \mathcal{X} \rightarrow \mathbb{R}_+$ depends on the input data $(\mathbf{x}_j)_{j \in [n]}$ only.

Approximate the conditional distribution

“Linear” estimator

The estimator of $p(y|\mathbf{x})$ has the form:

$$\hat{p}(y|\mathbf{x}) = \sum_{i=1}^n \omega_i(\mathbf{x}) \delta_{y_i}(y)$$

- ▶ $\omega_i(\mathbf{x})$ are weights, $\forall \mathbf{x}, \omega_i(\mathbf{x}) \geq 0$ and $\sum_{i=1}^n \omega_i(\mathbf{x}) = 1$
- ▶ $\omega_i : \mathcal{X} \rightarrow \mathbb{R}_+$ depends on the input data $(\mathbf{x}_i)_{i \in [n]}$ only.

Examples

- ▶ Regression (square loss): $\hat{f}(\mathbf{x}_0) = ?$
- ▶ Classification (binary): $\hat{f}(\mathbf{x}_0) = ?$

Approximate the conditional distribution

“Linear” estimator

The estimator of $p(y|\mathbf{x})$ has the form:

$$\hat{p}(y|\mathbf{x}) = \sum_{i=1}^n \omega_i(\mathbf{x}) \delta_{y_i}(y)$$

- ▶ $\omega_i(\mathbf{x})$ are weights, $\forall \mathbf{x}, \omega_i(\mathbf{x}) \geq 0$ and $\sum_{i=1}^n \omega_i(\mathbf{x}) = 1$
- ▶ $\omega_i : \mathcal{X} \rightarrow \mathbb{R}_+$ depends on the input data $(\mathbf{x}_i)_{i \in [n]}$ only.

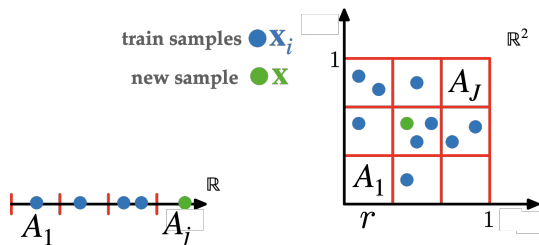
Examples

- ▶ Regression (square loss): $\hat{f}(\mathbf{x}_0) = \sum_i^n \omega_i(\mathbf{x}_0) y_i$ **local averaging**
- ▶ Classification (binary): $\hat{f}(\mathbf{x}_0) = \arg \max_{q \in \{+1, -1\}} \sum_{i=1}^n \omega_i(\mathbf{x}_0) \mathbf{1}_{y_i=q}$ **weighted majority vote**
- ▶ e.g. data agnostic: $\omega_i(\mathbf{x}) = \frac{1}{n}$, $\hat{f}(\mathbf{x}_0) = \arg \max_{q \in \{+1, -1\}} \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{y_i=q}$

Partition estimators

First basic idea: split your space

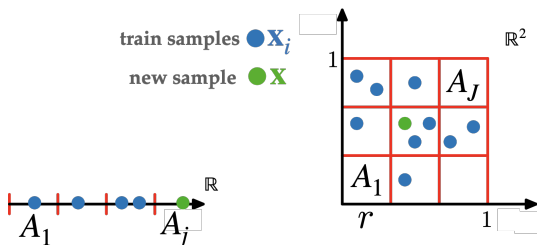
- ▶ $\mathcal{X} = \cup_{j \in J} A_j$ a partition of the input space ($j \neq j', A_j \cap A_{j'} = \emptyset$)



Partition estimators

First basic idea: split your space

- ▶ $\mathcal{X} = \cup_{j \in J} A_j$ a partition of the input space ($j \neq j', A_j \cap A_{j'} = \emptyset$)



- ▶ $A(\mathbf{x})$ is the unique partition corresponding to \mathbf{x}
- ▶ Weights are defined by:

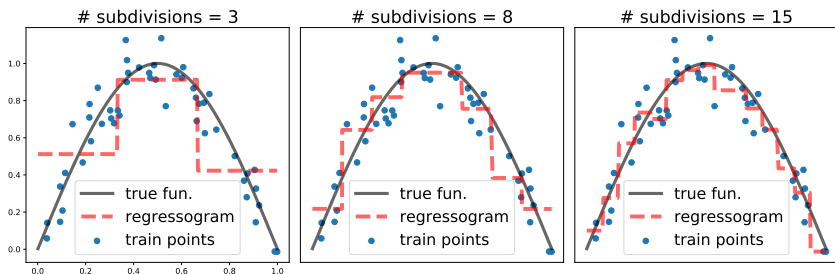
$$\forall i \in \{1, \dots, n\}, \omega_i(\mathbf{x}) = \frac{\mathbf{1}_{\mathbf{x}_i \in A(\mathbf{x})}}{\sum_{j=1}^n \mathbf{1}_{\mathbf{x}_j \in A(\mathbf{x})}}$$

- ▶ $\forall \mathbf{x} \in A_k, \omega_i(\mathbf{x}) = 1/\#\{\text{train samples in } A_k\}$ if $\mathbf{x} \in A_k$ else 0.
- ▶ If no train samples in A_k then $\forall i \in \llbracket n \rrbracket, \omega_i(\mathbf{x}) = 1/n$

Partition estimators

Regressogram

- ▶ Let $x_i \sim \text{Unif}([0, 1])$ with $n = 50$
- ▶ True function $f(x) = \sin(\pi x)$ and we observe $y_i = f(x_i) + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$
- ▶ Regression with partition estimator $\hat{f}(\mathbf{x}) = \sum_i^n \omega_i(\mathbf{x}) y_i$
- ▶ Local average of the outputs y_i in each region/partition
- ▶ Known as “regressogram”: piecewise affine estimator



Nearest neighbors

Birds of a feather flock together

- ▶ Given a new input \mathbf{x} K NN predictor looks at the K nearest points in the dataset $(\mathbf{x}_{i_1(\mathbf{x})}, y_{i_1(\mathbf{x})}), \dots, (\mathbf{x}_{i_K(\mathbf{x})}, y_{i_K(\mathbf{x})})$
- ▶ Prediction for \mathbf{x} = majority vote / averaging on K neighbors

Nearest neighbors

Birds of a feather flock together

- ▶ Given a new input \mathbf{x} KNN predictor looks at the K nearest points in the dataset $(\mathbf{x}_{i_1(\mathbf{x})}, y_{i_1(\mathbf{x})}), \dots, (\mathbf{x}_{i_K(\mathbf{x})}, y_{i_K(\mathbf{x})})$
- ▶ Prediction for \mathbf{x} = majority vote / averaging on K neighbors
- ▶ e.g. regression $\hat{f}(\mathbf{x}) = \frac{1}{K} \sum_{i \in \{i_1(\mathbf{x}), \dots, i_K(\mathbf{x})\}} y_i = \frac{1}{K} \sum_{i \in \mathcal{N}^K(\mathbf{x})} y_i$

Nearest neighbors

Birds of a feather flock together

- ▶ Given a new input \mathbf{x} KNN predictor looks at the K nearest points in the dataset $(\mathbf{x}_{i_1(\mathbf{x})}, y_{i_1(\mathbf{x})}), \dots, (\mathbf{x}_{i_K(\mathbf{x})}, y_{i_K(\mathbf{x})})$
- ▶ Prediction for \mathbf{x} = majority vote / averaging on K neighbors
- ▶ e.g. regression $\hat{f}(\mathbf{x}) = \frac{1}{K} \sum_{i \in \{i_1(\mathbf{x}), \dots, i_K(\mathbf{x})\}} y_i = \frac{1}{K} \sum_{i \in \mathcal{N}^K(\mathbf{x})} y_i$
- ▶ Corresponds to $\omega_i(\mathbf{x}) = \frac{1}{K}$ if $i \in \mathcal{N}^K(\mathbf{x})$ else 0

Nearest neighbors

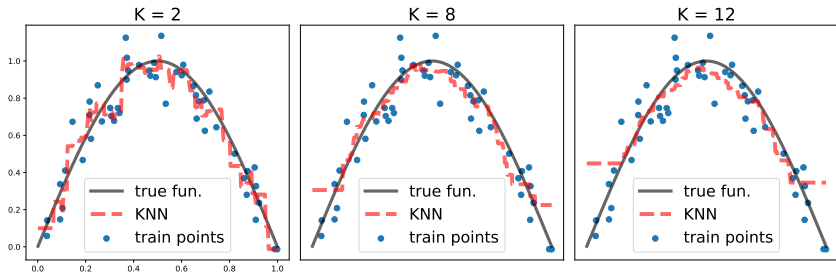
Birds of a feather flock together

- ▶ Given a new input \mathbf{x} KNN predictor looks at the K nearest points in the dataset $(\mathbf{x}_{i_1(\mathbf{x})}, y_{i_1(\mathbf{x})}), \dots, (\mathbf{x}_{i_K(\mathbf{x})}, y_{i_K(\mathbf{x})})$
- ▶ Prediction for \mathbf{x} = majority vote / averaging on K neighbors
- ▶ e.g. regression $\hat{f}(\mathbf{x}) = \frac{1}{K} \sum_{i \in \{i_1(\mathbf{x}), \dots, i_K(\mathbf{x})\}} y_i = \frac{1}{K} \sum_{i \in \mathcal{N}^K(\mathbf{x})} y_i$
- ▶ Corresponds to $\omega_i(\mathbf{x}) = \frac{1}{K}$ if $i \in \mathcal{N}^K(\mathbf{x})$ else 0
- ▶ K needs to be validated (cross-validation) + naive complexity $\mathcal{O}(nd)$ per test point \mathbf{x}

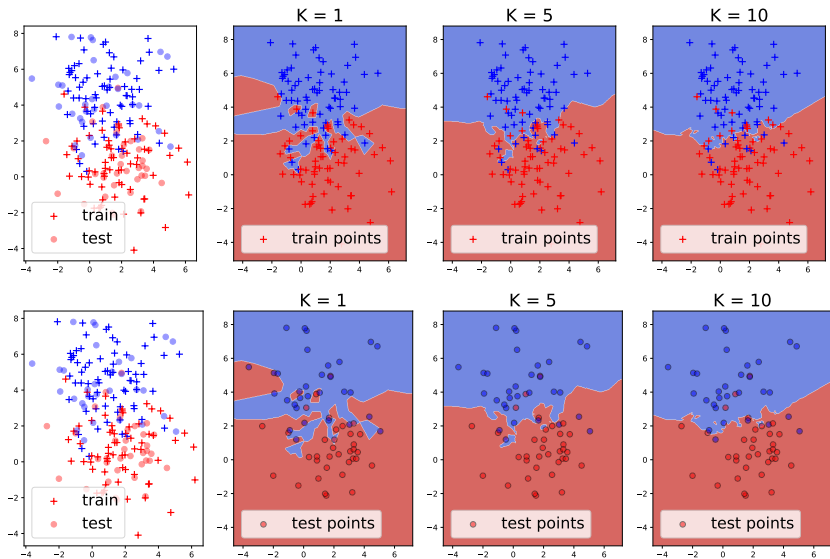
Nearest neighbors

Birds of a feather flock together

- ▶ Given a new input \mathbf{x} K NN predictor looks at the K nearest points in the dataset $(\mathbf{x}_{i_1(\mathbf{x})}, y_{i_1(\mathbf{x})}), \dots, (\mathbf{x}_{i_K(\mathbf{x})}, y_{i_K(\mathbf{x})})$
- ▶ Prediction for \mathbf{x} = majority vote / averaging on K neighbors
- ▶ e.g. regression $\hat{f}(\mathbf{x}) = \frac{1}{K} \sum_{i \in \{i_1(\mathbf{x}), \dots, i_K(\mathbf{x})\}} y_i = \frac{1}{K} \sum_{i \in \mathcal{N}^K(\mathbf{x})} y_i$
- ▶ Corresponds to $\omega_i(\mathbf{x}) = \frac{1}{K}$ if $i \in \mathcal{N}^K(\mathbf{x})$ else 0
- ▶ K needs to be validated (cross-validation) + naive complexity $\mathcal{O}(nd)$ per test point \mathbf{x}



Nearest neighbors



Nadaraya-Watson estimator

“Kernel” function

- ▶ $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ a pointwise non-negative function
- ▶ κ measures **similarity between points**


Nadaraya-Watson estimator

“Kernel” function

- ▶ $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ a pointwise non-negative function
- ▶ κ measures **similarity between points**
- ▶ In practice when $\mathcal{X} \subseteq \mathbb{R}^d$, $k(\mathbf{x}, \mathbf{x}') = q_h(\mathbf{x} - \mathbf{x}') = h^{-d}q(\frac{1}{h}(\mathbf{x} - \mathbf{x}'))$
- ▶ $q : \mathbb{R}^d \rightarrow \mathbb{R}_+$ that has large values around 0, $h > 0$ the **bandwidth**


Nadaraya-Watson estimator

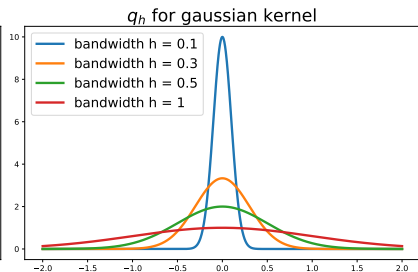
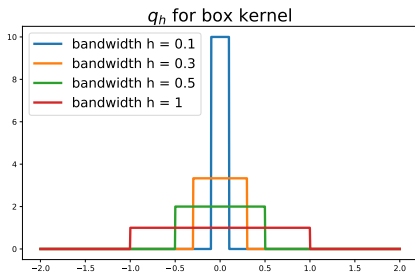
“Kernel” function

- ▶ $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ a pointwise non-negative function
- ▶ κ measures **similarity between points**
- ▶ In practice when $\mathcal{X} \subseteq \mathbb{R}^d$, $k(\mathbf{x}, \mathbf{x}') = q_h(\mathbf{x} - \mathbf{x}') = h^{-d}q(\frac{1}{h}(\mathbf{x} - \mathbf{x}'))$
- ▶ $q : \mathbb{R}^d \rightarrow \mathbb{R}_+$ that has large values around 0, $h > 0$ the **bandwidth**
- ▶ e.g. box kernel $q(\mathbf{x}) = \mathbf{1}_{\|\mathbf{x}\|_2 \leq 1}$, gaussian $q(\mathbf{x}) = \exp(-\|\mathbf{x}\|_2/2)$
- ▶  same name but not the same as kernels in SVM !

Nadaraya-Watson estimator

“Kernel” function

- ▶ $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ a pointwise non-negative function
- ▶ κ measures **similarity between points**
- ▶ In practice when $\mathcal{X} \subseteq \mathbb{R}^d$, $k(\mathbf{x}, \mathbf{x}') = q_h(\mathbf{x} - \mathbf{x}') = h^{-d}q(\frac{1}{h}(\mathbf{x} - \mathbf{x}'))$
- ▶ $q : \mathbb{R}^d \rightarrow \mathbb{R}_+$ that has large values around 0, $h > 0$ the **bandwidth**
- ▶ e.g. box kernel $q(\mathbf{x}) = \mathbf{1}_{\|\mathbf{x}\|_2 \leq 1}$, gaussian $q(\mathbf{x}) = \exp(-\|\mathbf{x}\|_2/2)$
- ▶  same name but not the same as kernels in SVM !



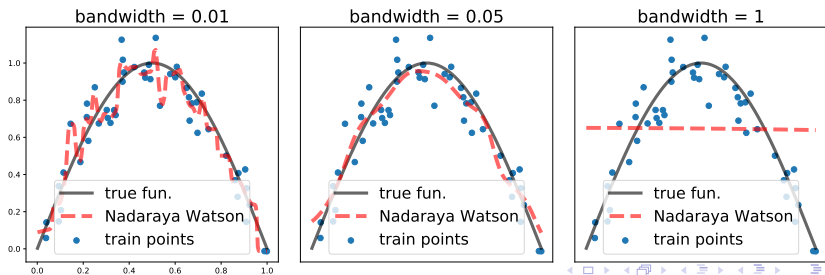
Nadaraya-Watson estimator

The Nadaraya-Watson estimator

- ▶ The weights are defined by:

$$\omega_i(\mathbf{x}) = \frac{\kappa(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^n \kappa(\mathbf{x}, \mathbf{x}_j)}$$

- ▶ $\omega_i(\mathbf{x})$ is close to 1 when \mathbf{x} is similar to \mathbf{x}_i
- ▶ e.g. regression $\hat{f}(\mathbf{x}) = \sum_{i=1}^n \frac{\kappa(\mathbf{x}, \mathbf{x}_i) y_i}{\sum_{j=1}^n \kappa(\mathbf{x}, \mathbf{x}_j)}$, smoothness depends on h
- ▶ Complexity of calculating $\hat{f}(\mathbf{x})$ usually in $\mathcal{O}(nd)$



Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

The answer is statistical

- ▶ Can I bound (w.h.p. or in expectation)

$$\text{excess risk} = \text{estimation error} + \text{approximation error} \leq h(n)$$

- ▶ Ideally the function $h(n) \xrightarrow{n \rightarrow +\infty} 0$ fast: it is called the *learning rate*

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

The answer is statistical

- ▶ Can I bound (w.h.p. or in expectation)

$$\text{excess risk} = \text{estimation error} + \text{approximation error} \leq h(n)$$

- ▶ Ideally the function $h(n) \xrightarrow{n \rightarrow +\infty} 0$ fast: it is called the *learning rate*

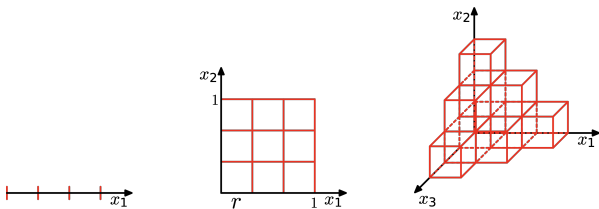
Curse of dimensionality

- ▶ If the best function f^* is only Lipschitz-continuous:

$$\mathbb{E}[\text{excess risk}] \lesssim n^{-1/d}$$

- ▶ This rate is **unavoidable** without further knowledge on f^*
- ▶ Slow rates: exponentially many samples are needed !

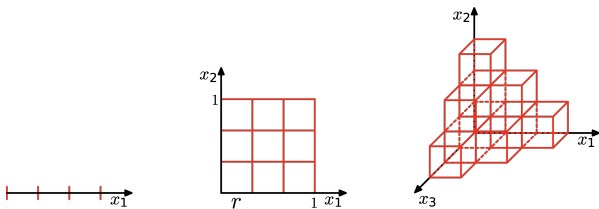
Curse of dimensionality



Think about interpolation: goal find $f : [0, 1]^d \rightarrow \mathbb{R}$

- ▶ You have n values of this function $f(x_i)$ at *sampled locations* x_i
- ▶ To find f you want to interpolate between the $f(x_i)$'s
- ▶ If f is not regular, a good approximation requires precise covering of $[0, 1]^d$ (small meshes)

Curse of dimensionality



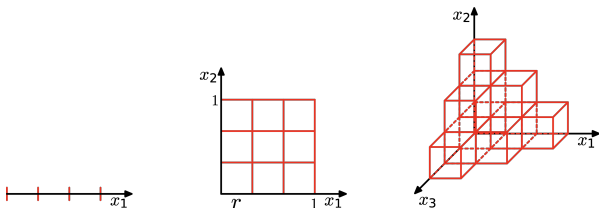
Think about interpolation: goal find $f : [0, 1]^d \rightarrow \mathbb{R}$

- ▶ You have n values of this function $f(x_i)$ at *sampled locations* x_i
- ▶ To find f you want to interpolate between the $f(x_i)$'s
- ▶ If f is not regular, a good approximation requires precise covering of $[0, 1]^d$ (small meshes)

Problem: points are isolated in high dimension

- ▶ Vol. of a hypercube with edge length $r < 1$ is r^d : quickly $\downarrow 0$ as $d \rightarrow \infty$
- ▶ To compensate you need a number of sample which **grows exponentially with d**

Curse of dimensionality



Main ideas

- ▶ Without prior on f^* the required number of samples n to estimate f^* is **exponential in d** .
- ▶ In high dim it is easy to overfit a model.
- ▶ The notion of nearest neighbors vanishes in high dim.
- ▶ Three remedies: use simple models (linear), dimensionality reduction or prior on f^*

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

The answer is statistical

- ▶ Can I bound (w.h.p. or in expectation)

$$\text{excess risk} = \text{estimation error} + \text{approximation error} \leq h(n)$$

- ▶ Curse of dimensionality (see rates in Györfi et al. 2002; Bach 2022)

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

The answer is statistical

- ▶ Can I bound (w.h.p. or in expectation)

$$\text{excess risk} = \text{estimation error} + \text{approximation error} \leq h(n)$$

- ▶ Curse of dimensionality (see rates in Györfi et al. 2002; Bach 2022)

A new hope: prior knowledge on the Bayes predictor

- ▶ If the best function f^* is smooth (bounded s -th order derivatives):

$$\mathbb{E}[\text{excess risk}] \lesssim n^{-s/d}$$

- ▶ Smoothness helps ! But also if data is low-dimensional.
- ▶ Even better: linear $\mathbb{E}[\text{excess risk}] = \frac{\sigma^2 d}{n}$, finite hypothesis space: **on the board !**

One algorithm to rule them all ?

After all, is it possible to learn all tasks with one algorithm ?

One algorithm to rule them all ?

After all, is it possible to learn all tasks with one algorithm ?

- ▶ $D_p(n) = \{(x_i, y_i)\}_{i \in [n]}$ where $(x_i, y_i) \sim p$ *i.i.d.*
- ▶ Very abstract way: an algorithm \mathcal{A} is a mapping from $D_p(n)$ to a function from \mathcal{X} to \mathcal{Y}
- ▶ Goal: find \mathcal{A} such that $\mathcal{R}_p(\mathcal{A}(D_p(n))) - \mathcal{R}_p^*$ is small

One algorithm to rule them all ?

After all, is it possible to learn all tasks with one algorithm ?

- ▶ $D_p(n) = \{(x_i, y_i)\}_{i \in [n]}$ where $(x_i, y_i) \sim p$ i.i.d.
- ▶ Very abstract way: an algorithm \mathcal{A} is a mapping from $D_p(n)$ to a function from \mathcal{X} to \mathcal{Y}
- ▶ Goal: find \mathcal{A} such that $\mathcal{R}_p(\mathcal{A}(D_p(n))) - \mathcal{R}_p^*$ is small

No free lunch theorem

Binary classif. with 0 – 1 loss, with \mathcal{X} infinite. Let \mathcal{P} = set of all prob. distributions on $\mathcal{X} \times \{0, 1\}$.

$\forall n > 0, \forall \mathcal{A}$:

$$\sup_{p \in \mathcal{P}} \mathbb{E}[\mathcal{R}_p(\mathcal{A}(D_p(n)))] - \mathcal{R}_p^* \geq 1/2$$

One algorithm to rule them all ?

After all, is it possible to learn all tasks with one algorithm ?

- ▶ $D_p(n) = \{(x_i, y_i)\}_{i \in [n]}$ where $(x_i, y_i) \sim p$ i.i.d.
- ▶ Very abstract way: an algorithm \mathcal{A} is a mapping from $D_p(n)$ to a function from \mathcal{X} to \mathcal{Y}
- ▶ Goal: find \mathcal{A} such that $\mathcal{R}_p(\mathcal{A}(D_p(n))) - \mathcal{R}_p^*$ is small

No free lunch theorem

Binary classif. with 0 – 1 loss, with \mathcal{X} infinite. Let \mathcal{P} = set of all prob. distributions on $\mathcal{X} \times \{0, 1\}$.

$\forall n > 0, \forall \mathcal{A}$:

$$\sup_{p \in \mathcal{P}} \mathbb{E}[\mathcal{R}_p(\mathcal{A}(D_p(n)))] - \mathcal{R}_p^* \geq 1/2$$

For any \mathcal{A} , for a fixed n , there is a data distribution that makes the algorithm useless (same as the chance level)

Plan

What is machine learning ?

Data in machine learning

From training data to prediction

- Loss functions

- Empirical risk minimization

- Underfitting/overfitting

Model selection and validation

- Split your dataset !

A glimpse of decision theory & statistical learning

- Risk and empirical risk

- Risk decomposition

First models: local averaging methods

Learning rates and curse of dimensionality

- No free-lunch theorem

Conclusion


Other problems


► Fairness:




- Interpretability
- Ecological problems
- And many more ...


References I

 Arlot, Sylvain and Alain Celisse (2010). “A survey of cross-validation procedures for model selection”. In: *Statistics Surveys* 4.

 Bach, Francis (2022). *Learning Theory from First Principles*.

 Györfi, L. et al. (2002). *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics. Springer New York.

 Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*. Springer New York Inc.

 Jumper, John et al. (2021). “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873, pp. 583–589.

 Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press.