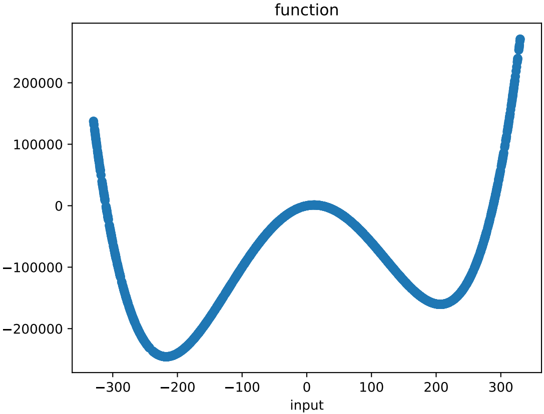
Machine learning I, supervised learning: problem statement



Machine learning (ML)

- (proposed definition of) **learning** : "Modification of a behavior, based on a life experiment"
- a **machine learning** system is programmed to learn in a **semi-automatic** way.

Classical programming vs ML

- Classical program : predict the total amount of money spent, based on the number of fruits bought and the price of each individual fruit (a summation is enough)
- ML program : predict which fruit a person buys, based on a database of customers and on some information about this person (e.g. buying log, age, profession).

Book

https://cazencott.info/index.php/pages/Introduction-au-Machine-Learning

Use cases of ML

ML is useful for problems :

- that we cannot solve direcly thanks to an explicit representation (such as the amount of money spent in the previous first example).
- that we can solve in practice, but without a complete understanding of the underlying mechanism (face recognition)
- that we could solve explicitely, but the computationnal ressources would be too heavy (molecular dynamics), so we **approximate** a solution with ML.

Denomination

- The name "machine learning" is rather deceitful !
- It is no more machine driven than any algorithm or coffee machine.
- The denomination "statistical learning" is sometimes used, and is more epxlicit.

Ingredients of machine learning (ML)

ML's main ingredients are

- statistics and probabilities : most ML objects can (should) be seen as outcomes of **random variables**.
- optimization : most (but not all) ML problems are formulated as an optimization problem.

Other tools come from :

- graph theory
- information theory
- statistical physics

Deep learning

- ML is a subset of AI.
- Deep learning is a subset of ML.

Learning paradigms

- **supervised learning** : learn to predict an output as a function of an input (predict the energy production of a wind farm based on sensors)
- **unsupervised learning** : learn information about the structure of data (density estimation, clustering, dimensionality reduction)
- **reinforcement learning** : learn to perform actions in order to maximize a reward (game player, alphago)

Why is there a hype around machine learning ?

Machine learning has received attention and funding because it has reached state-of-the-art efficiency on several problems, such as :

- computer vision
- spam classification
- machine translation
- speech recognition
- self-driving cars

Deep learning is involved in several of these setups.

ML revolution

- Technical progress in the computing and storage capacities
- Increase in amount of available data. According to IBM, 10^{18} bytes are created each day.
- Progress in algorithmic methods to analyze the data.

Example 1 : ImageNet

- A database of images (more than 15M), hand-annotated in order to indicate what objects are present in the image. More than 20000 categories of images.
- Contest : ImageNet Large Scale Visual Recognition Challenge.
- The best top 5 score (a measure of the classification error) went from 25% in 2011 to $\simeq$ 15.3% in 2012.

Example 1 : ImageNet

- A database of images (more than 15M), hand-annotated in order to indicate what objects are present in the image. More than 20000 categories of images.
- Contest : ImageNet Large Scale Visual Recognition Challenge.
- The best top 5 score (a measure of the classification error) went from 25% in 2011 to $\simeq$ 15.3% in 2012.
- The technology used was deep learning, exploiting GPUs (AlexNet).

`https://paperswithcode.com/sota/image-classification-on-imagenet`

Example 2 : AlphaGo

- In 2015 : beats a professionnal player. In 2017 : beats the world champion.
- Uses several technologies : among them **Deep reinforcement learning.**
- Improvements : AlphaGo Zero, trained without a database of played games. In 2017, AlphaZero beats AlphaGo Zero after 3 days of learning.

Example 3 : AlphaFold

- Goal : to predict the spatial configuration of proteins, from their DNA sequence.
- Achieves a breakthgough performance on the CASP challenge :
  - 2018 : more than 50% GDT (Global distance test), whereas it was $\leq 40\%$ before then.
  - 2020 : $92,4\%$ GDT. At a $\geq 90\%$ score, the method is considered competitive with experimental methods..
- `https://alphafold.ebi.ac.uk/`
- Also based on Deep learning.

Audio engineering

https://www.sonible.com/smarteq3/
https://www.youtube.com/watch?v=ZGetnk222YU&t=494s

Supervised learning

Predict an output from an input.

- discrete output : classification
- continuous output : regression

We learn from a dataset D_n of n **labeled examples** ;

$$D_n = \{(x_i, y_i), i \in [1, n]\} \tag{1}$$

Supervised Learning : example of regression

- x : age and height of a person. Here x, is a **vector** containing 2 **features**.
- y : record on a 100 meters track

Regression : raw inputs

```
nv (node)
2.928290359468087800e+01 1.819015686789989559e+02
2.804812241859411515e+01 1.808859323205479654e+02
1.812029840482162513e+01 1.614667413796299229e+02
2.565190120950552100e+01 1.844006827739839309e+02
3.054629307666820637e+01 1.739981941657430866e+02
2.600748351639588662e+01 1.811637151224612410e+02
1.507418892029618895e+01 1.868432490529098970e+02
3.042766519165544281e+01 1.789727990680546270e+02
3.658212742350856672e+01 1.785726961288677046e+02
2.353156221187539643e+01 1.780926290043428253e+02
2.830054102263598281e+01 1.618048901441496241e+02
2.327509936340274166e+01 1.908102611218433537e+02
3.044525330771638849e+01 1.736836766109009886e+02
2.488014451964265206e+01 1.794912044737207282e+02
2.226526635755104166e+01 1.788433969242797446e+02
1.556357798027287487e+01 1.909257627614161379e+02
1.233905436641630615e+01 1.769331920900993964e+02
2.430793641280866879e+01 1.712111558087560752e+02
3.881928396681708193e+01 1.794596597181927962e+02
1.982792868143741671e+01 1.845118060871293721e+02
3.118524101800569781e+01 1.636248014144912872e+02
2.005698608291917395e+01 1.803034695205161881e+02
2.700274799779473511e+01 1.792502475568811064e+02
2.506227027010324448e+01 1.762586026486706032e+02
1.907277569274998186e+01 1.826660104477087145e+02
2.528781623067084183e+01 1.883758877752532612e+02
3.232888827510984697e+01 1.767863031548815229e+02
2.209651833311454183e+01 1.746617964376699206e+02
3.087017061896954573e+01 1.780176276478124180e+02
2.220465169371339087e+01 1.927466639363435377e+02
2.347947811151988162e+01 1.889400651201518144e+02
2.185924634187936633e+01 1.729108385149218350e+02
1.808735582805482878e+01 1.719105999524763604e+02
1.844754334541114105e+01 1.694675852134845400e+02
2.817410966313331855e+01 1.715017351672584027e+02
2.523219998240179862e+01 1.798536540705599407e+02
3.109896511888699422e+01 1.824544193833422696e+02
3.185907680322553404e+01 1.759604448214206229e+02
2.969165277770446210e+01 1.875971336737983961e+02
2.787730255078729158e+01 1.757878208874049051e+02
2.591225877027716606e+01 1.601041967482410087e+02
2.498572495922823578e+01 1.721740687961011247e+02
2.290736984087772043e+01 1.749157042606373409e+02
3.372473127822453876e+01 1.906407728744829058e+02
2.607195861993280062e+01 1.727834180024526347e+02
2.600320546710192460e+01 2.021277060723097918e+02
2.491918989168637566e+01 1.768863549264243886e+02
3.377987890024456163e+01 1.726478999441613098e+02
1.650983933678821103e+01 1.782204896060512169e+02
1.908717367205185766e+01 1.817607063871927551e+02
2.629780828398855164e+01 1.801754036532959162e+02
NORMAL  master  github/code/examples/X.txt
:NERDTreeToggle
```

Input data.

Regression : raw ouptuts

```
1.618446082230513028e+01
1.186648764508175802e+01
1.464778565654695086e+01
1.634498113765004490e+01
1.745913980223237516e+01
1.166095693440647096e+01
1.520870379572047604e+01
1.406742858858115319e+01
1.288786198880906753e+01
1.455705638818248104e+01
1.659472779771285644e+01
1.545656976520241521e+01
1.400667994864794963e+01
1.316859968706612349e+01
1.850569481499744739e+01
1.522353555851539397e+01
1.357865817589575563e+01
1.575478790439513865e+01
1.151989857308148757e+01
1.366059581731965622e+01
1.474667733562292504e+01
1.420670108820832667e+01
1.695499948932924639e+01
1.351075264826661382e+01
1.421153509268070181e+01
1.554265426599405409e+01
1.490759453066251261e+01
1.607137159898928758e+01
1.435965829372504032e+01
1.493929537374976135e+01
1.286585663123390688e+01
1.653170786752944110e+01
1.524065698484768028e+01
1.421366775637391555e+01
1.431635706484540194e+01
1.216972043271089277e+01
1.333268127294045513e+01
1.628442909496436641e+01
1.491744545525917331e+01
1.392289095842998847e+01
1.237963466793929257e+01
1.454636430040251938e+01
1.591321928754086912e+01
1.517261623645199720e+01
1.417429098481692940e+01
1.430763524632289752e+01
1.426651768644284068e+01
1.485261785696788195e+01
1.417034608280185104e+01
9.800000000000000711e+00
1.413277600956676672e+01
NORMAL  master  github/code/examples/time.txt
:NERDTreeToggle
```

Output data.

Precision

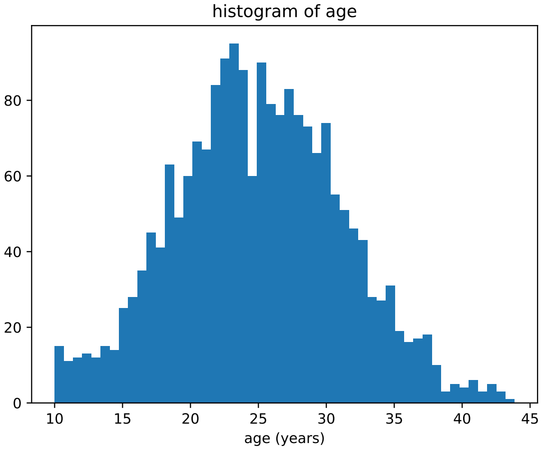
Exercice 1 :

What order of magnitude of precision do we expect to obtain with this example ?

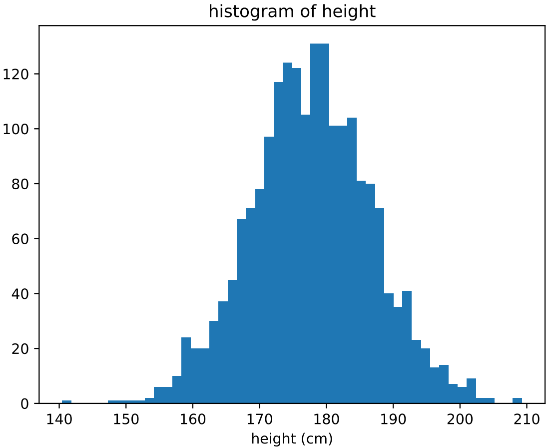
Precision

Exercice 1 :
What order of magnitude of precision do we expect to obtain with this example ?
In a machine learning problem, there is (almost) always a statistical error associated to a prediction. Our goal is to minimize it !

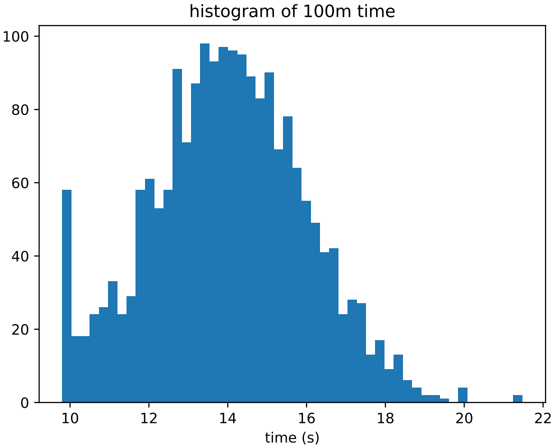
Histograms



Histograms


histogram of height
0
20
40
60
80
100
120
140
150
160
170
180
190
200
210
height (cm)

Histograms



histogram of 100m time
100
80
60
40
20
0
10
12
14
16
18
20
22
time (s)

Important question

We assume that there exists a relationship between the inputs x and the output y, given by some function f.

$$y = f(x) \tag{2}$$

f is most of the time a function that contains some **randomness**.

- **Objective :** find a good **estimation** $\tilde{f}$, of f.
- $\tilde{f}$ maps an input to an output, deterministically. For the input x_i, we predict $\tilde{f}(x_i)$.
- In order to measure the quality of $\tilde{f}$, we use **loss functions**.

Example loss function for a regression problem

- The loss function should be a measure of the discrepancy between our prediction and the correct label.
- For an individual sample, **a** discrepancy is the least-square loss

$$(\tilde{f}(x_i) - y_i)^2 \tag{3}$$

Loss function

- Taking into account the whole dataset, the **loss function** writes :

$$\sum_{i=1}^{n} (\tilde{f}(x_i) - y_i)^2 \tag{4}$$

- Several other loss functions are possible :

$$\sum_{i=1}^{n} |\tilde{f}(x_i) - y_i| \tag{5}$$

Loss function

- The loss function is a **real number** measuring the relevance of a **collection** of parameters ($\tilde{f}$ is defined by these parameters.)
- The number of parameters depends on the situation, and varies between 1 (e.g. for a simple linear model) and millions (e.g. for some deep neural networks).

Important question II

- To what subset of functions does $\tilde{f}$ belong ?

More supervised learning examples : I

Predict the winning team of an NBA game at half-time.

- Dataset : 15 years of games (comments, text) : approximately 17000 games.
- The dataset is preprocessed to have as an input a time-series : each time contains the score **and** 10 technical features (rebounds, etc.). So for each time the dimension is 11. Each game is a matrix of size 1440×11, reorganized as a line vector.
- Output : Receiving team wins or looses (classification)
- Evaluation metric : classification error ("0-1" loss).

Example II

Predict the quantity of oil in a rock.

- Input : tomographic image of a rock.
- Output : material of the rock, average presence of residual oil in the rock (regression).

Example III

Detect issues in wind farms.

- Input : sensors on the wind turbine (wind direction, air temperature, electric tension, rotation speed, component temperature, etc.) as a time series. Each step represents 10 minutes (several years).
- Output : Power generated by the turbine (regression)
- Evaluation metric : MAE (mean absolute error).

Difficulties of machine learning

- hard optimization problem
- overfitting / statistical garantees
- curse of dimensionality

Optimization problem

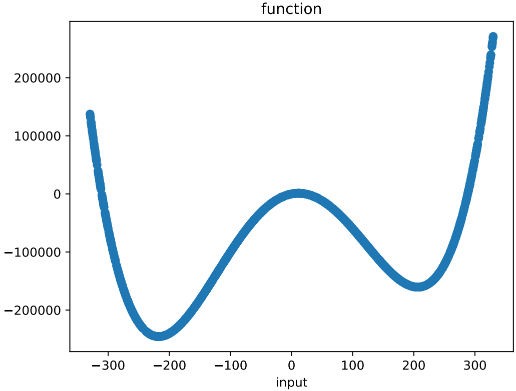
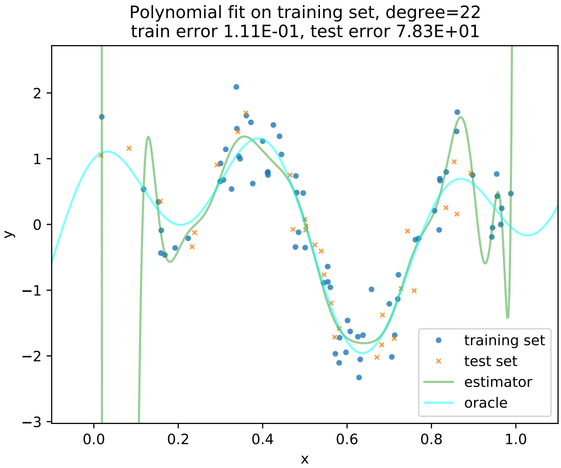


Figure – Loss function

Overfitting



Curse of dimensionality

Two numbers are important in machine learning :

- n : number of samples
- d : dimension (number of features) of a unique sample

Both can be large and prohibitive for some algorithms.

Curse of dimensionality

- n is large when the dataset has many samples.
- d is large if each sample has many features :
  - image
  - DNA sequence
  - text
  - audio/video file