eleven | ENPC – Département IMI
Refresher on Supervised Learning

To the attention of IMI Department

February 13th, 2024

École des Ponts ParisTech

eleven
strategy • data • digital

Supervised Learning

Artificial intelligence can be organized around three families of machine learning models

	Approach	Model	Goal
Machine learning	**Supervised learning**	Regression	Learn a function that maps an input to an output based on example input-output pairs (labeled data)
		Classification	
	Unsupervised learning	Clustering	Identify and uncover previously undetected patterns with no preexisting labeled data
		Collaborative filtering	
	Reinforcement learning	Q-table based	Find the optimal solution based on: An environment A set of rules A playing agent
		Deep learning based	

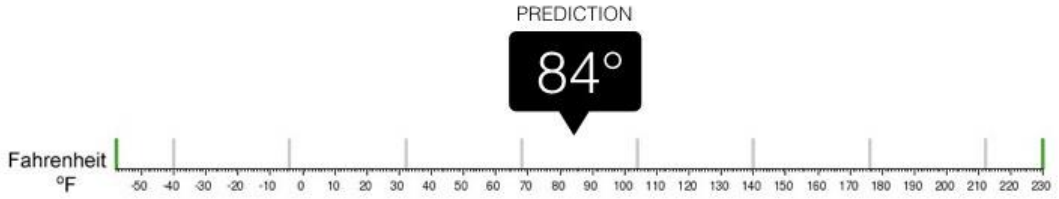
Not necessary for the hackathon

Regression models aim at modeling continuous variables whereas classification models aim at modeling categorical variables

Example of the difference between regression and classification

Regression

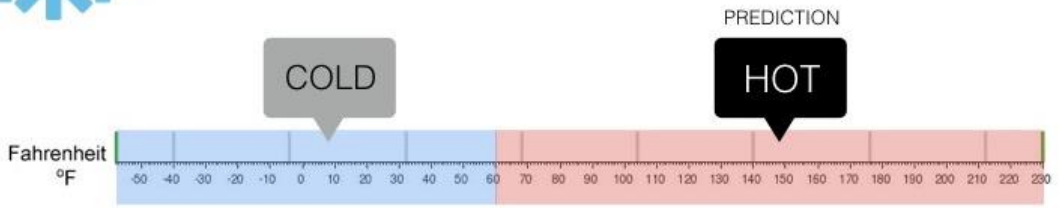
What is the temperature going to be tomorrow?



The output of the model is the temperature which is a continuous variable

Classification

Will it be Cold or Hot tomorrow?



The output of the model is a category: cold or hot

Machine learning models can be leveraged to fulfill different goals

Understand

- Understanding the behavior or a given process, e.g.,
  - The relation between temperature and a building infrastructure
  - The relation between the intensity of a press and the quality of the animal food
  - …

- If the goal is to understand a behavior, then the value is in the estimation of the parameters
  (the causal parameters)

Predict

- Predicting the output of a process given a new occurrence, e.g.,
  - What will the energy price be in one week?
  - What is this task's time to completion?
  - …

Optimize

- Optimizing a process given a certain amount of information and constraints, e.g.,
  - Generating an efficient sprinkler network based on the building's blueprint and the involved regulatory principles
  - …

- If the goal is to predict an output or optimize a process, then the value is in the goodness of the output estimation
- It would be enough to have a « good » estimation of the parameter to be able to make accurate predictions or efficient optimizations

…

A machine learning model refers to both a group of functions to solve a task, and the chosen function within that group

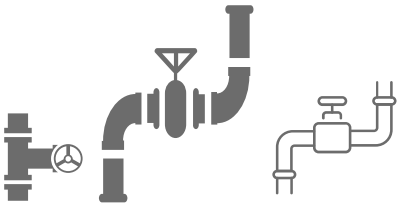
Illustration of machine learning models

Is there a cat in this picture?

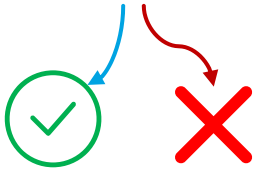
What could be an individual's revenue

Input space: Images

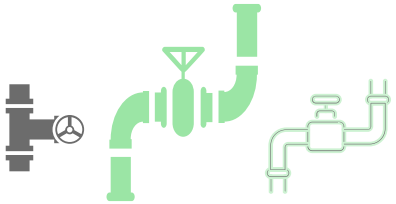
Model: Maps data from the input space into the output space



Output space: Yes or No



Input space: Tabular data



Model: Maps data from the input space into the output space

£ 35k

Output space: A predicted revenue

Machine Learning tasks consist in making a statistical model learn a mapping function from the inputs to the desired outputs from the data available

Formally, a model (e.g., Linear Regression) is a class of functions to consider for that task (e.g., all linear functions)

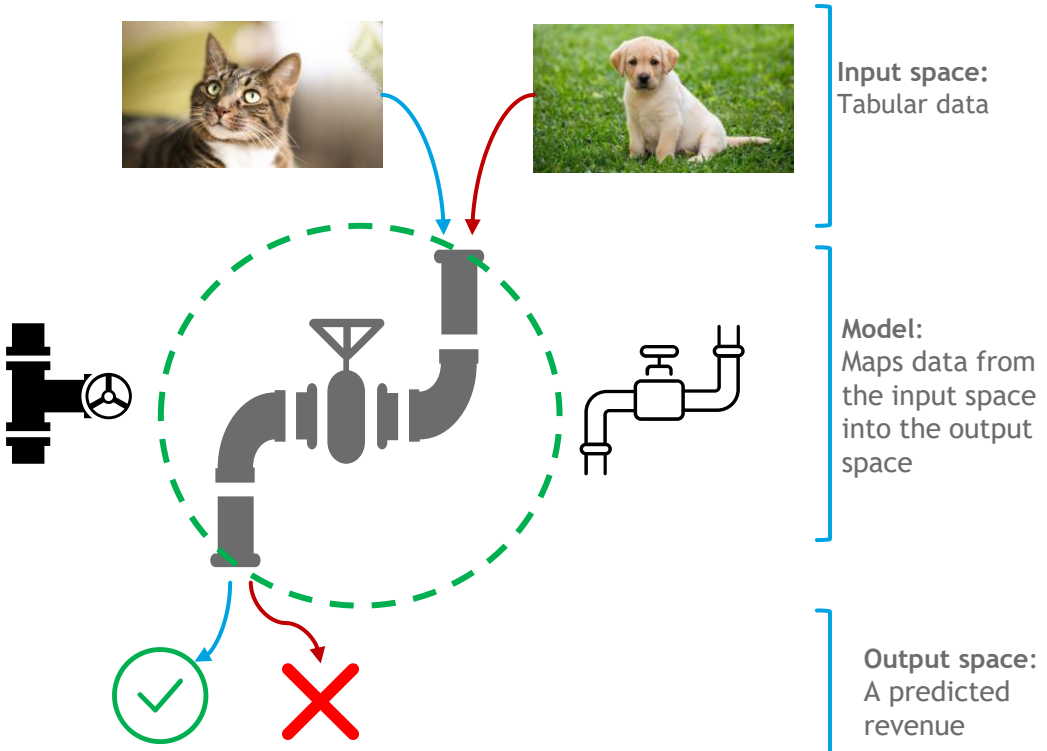
$$S = \{f_\theta, \theta \in \Theta\}$$

In other words, a group of mappings to choose from:

A machine learning model refers to both a group of functions to solve a task, and the chosen function within that group

Illustration of machine learning models

Is there a cat in this picture?



Training a model amounts to finding the optimal function with that class:

Find best $s \in S$

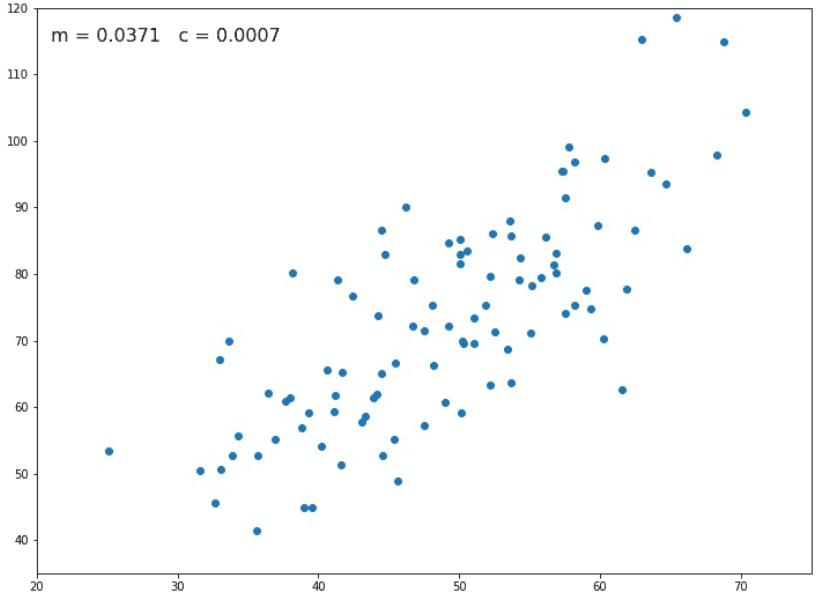
By extension, we call « model » both the ensemble of considered mappings (before training) and the one selected after the training process

- The goal of ML model training is to find the configuration of parameters' values of the model that most suit the observed data
- Once the model is "trained" it can then make proper inferences about a previously unknown observation (test set, real-life data)

Training or fitting a model amounts to finding the parameters making the model closer to the dataset with regards to a metric

Illustration of machine learning models

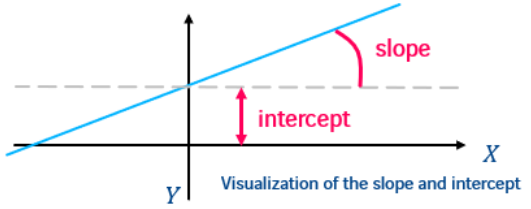
Example of linear regression



Linear regression example

Our objective is to find the line that "best represents" (fits) the available observations (the point cloud).

That line is defined by two parameters: the slope θ_1 and the intercept θ_2



- We must find the intercept and slope (the parameters) that best fit the points
- We need a metric (the Loss) in order to tell apart the different parameters. We choose the distance between each point and the line.
- We find the best parameters by minimizing this distance.

Supervised learning algorithms aim to understand the fundamental relationship between explanatory variables X and a variable to be predicted Y

TWO MAIN PARADIGMS

Estimation of the relationship Y = f(X)

- **The problem is** to estimate the general function based on the available data and assuming an a priori form of the function

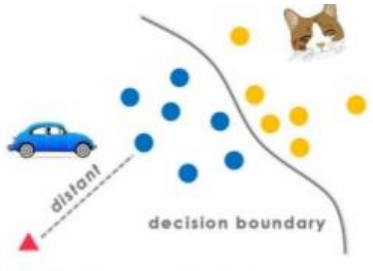
Approximation of the function dependent on the dataset

$$Y \approx \hat{f}_\tau(X_1, \ldots, X_n)$$

Target variable or events to be predicted

Explanatory variables or Features

- **The key question:** is the approximation good enough to generalise to new observations, beyond the available data?
- **Illustration:**



Estimation of the probability distribution P(X,Y)

- **The problem** is to estimate the joint distribution based on the available data (e.g. Bayesian learning)

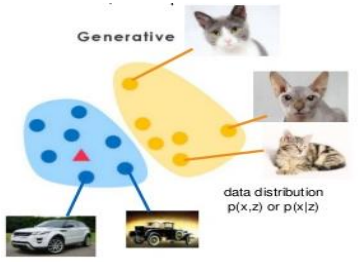
Approximation of the distribution dependent on the data set

$$\hat{P}_\tau(Y, X_1, \ldots, X_n)$$

Target variable or events to be predicted

Explanatory variables or Features

- **The key question:** is the approximation good enough to generalise to new observations, beyond the available data?
- **Illustration:**



Regression attempts to estimate the mapping function from the input variables to numerical or continuous output variables

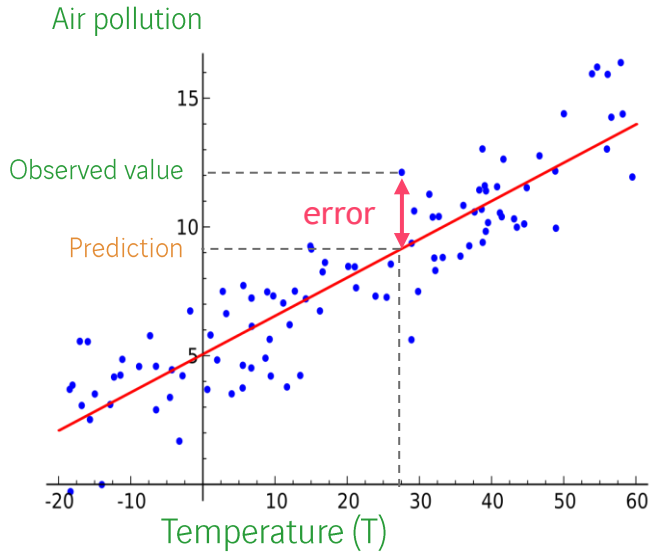
Description : Linear Regression

- Find the relation between input variables and continuous output variable
- Example : Understanding the relation between air pollution and temperature
- Class of model selected: linear regression

$$\hat{y} = f_\theta(x) = \theta_0 + \theta_1 x_1$$

- Parameter $\boldsymbol{\theta} = (\theta_0, \theta_1)$

Illustration



Metric

The error is the difference between model's prediction and the real value

$$error = (y - \hat{y})$$

A classical metric for regression is the Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}$$

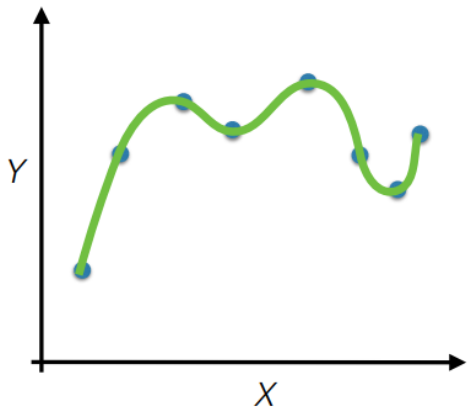
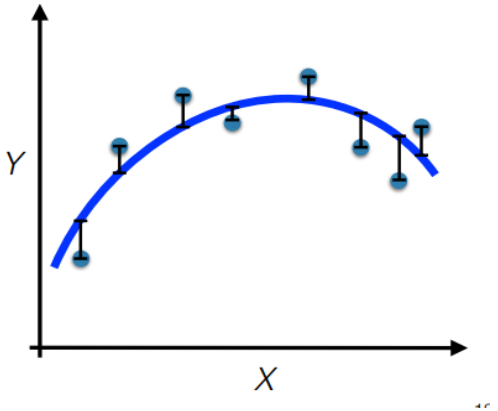
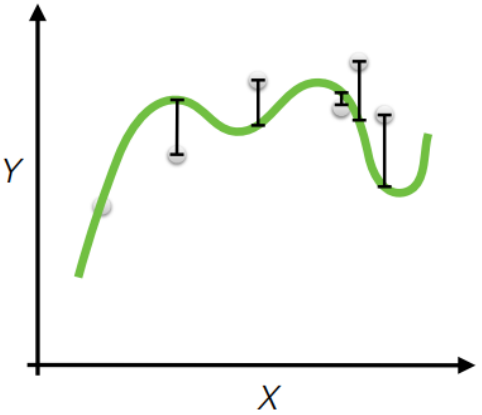
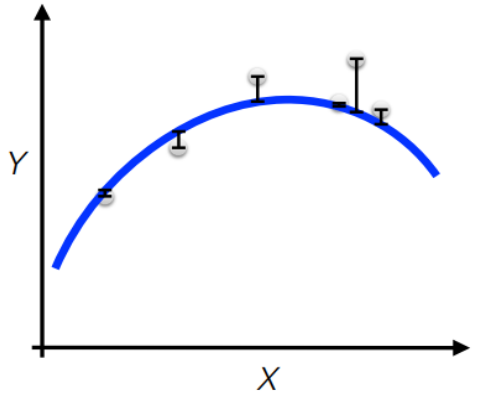
- N is the number of observations
- Yi the observed output values
- $\hat{Y}_i$ represents the predicted values

The goal of regression algorithms is to minimize the RMSE

How can we make sure that the parameters found are also optimal for new samples ?

Various sources of error are then possible: measurement errors, errors in estimating the function f, errors related to the choice of the regularity of the function

Example of overfitting

	Overfitting	Generalizable model
Training Data		
Validation Data		

Artificial neural networks are based on biological neurons because the human brain processes information efficiently

Biological neurons

- Basis constituent of the nervous system
- Information is propagated by electric signals through dendrites and axons
- A neuron is activated when the input signal is higher than a threshold
- A neuron is equivalent to a Heaviside step function

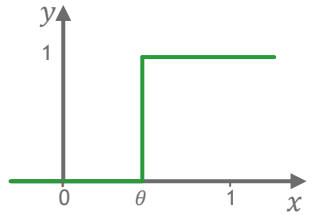


Illustration of Heaviside Function

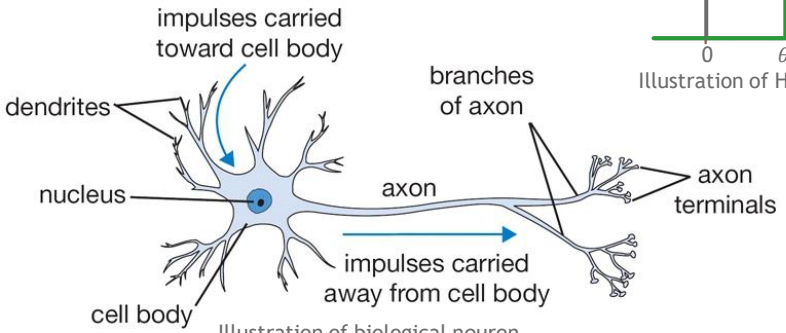
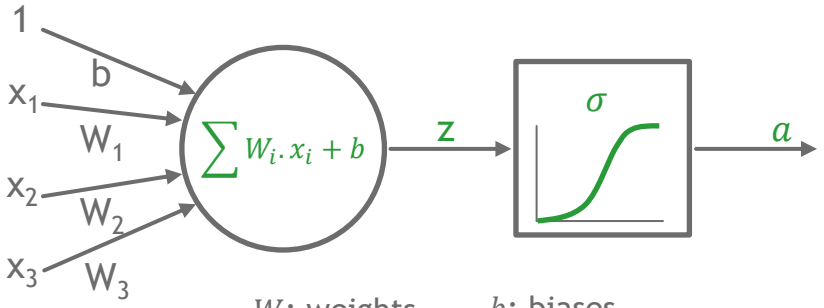


Illustration of biological neuron

Artificial neuron

- A simplified model of a biological neuron
- An artificial neuron is a linear function, on which an activation function is added
- The step function is not adapted due to its sensitivity to noise
- The common activation function used is the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



W: weights

b: biases

x: inputs

σ: activation function

Summary

- Each neuron processes a bit of information and passes it to its children
- Overall, the network processes raw information into general concepts

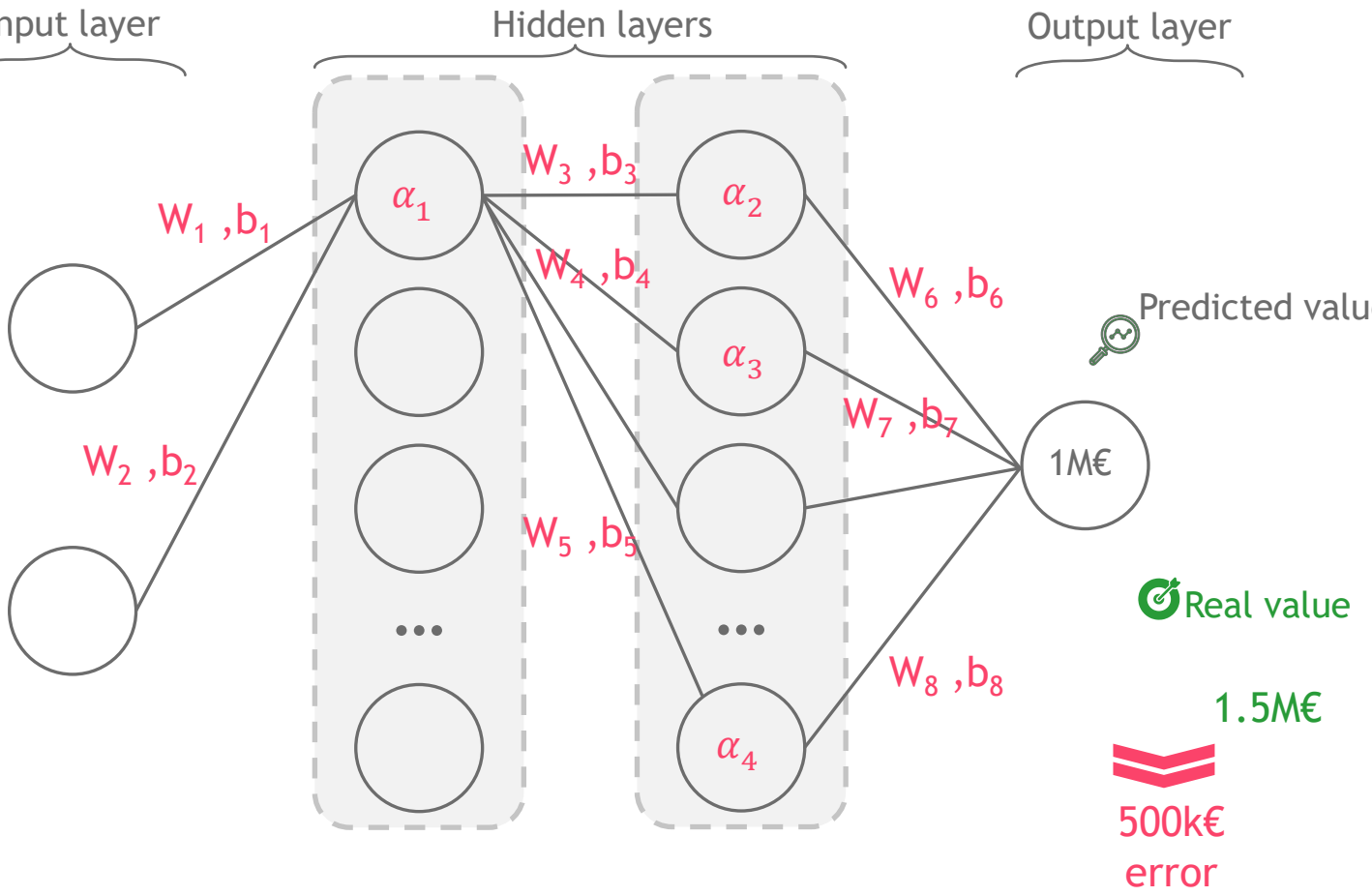
A neural network learn how to imitate a function from a training dataset

Computation process: real estate price prediction example

Input features

400 m²

Paris

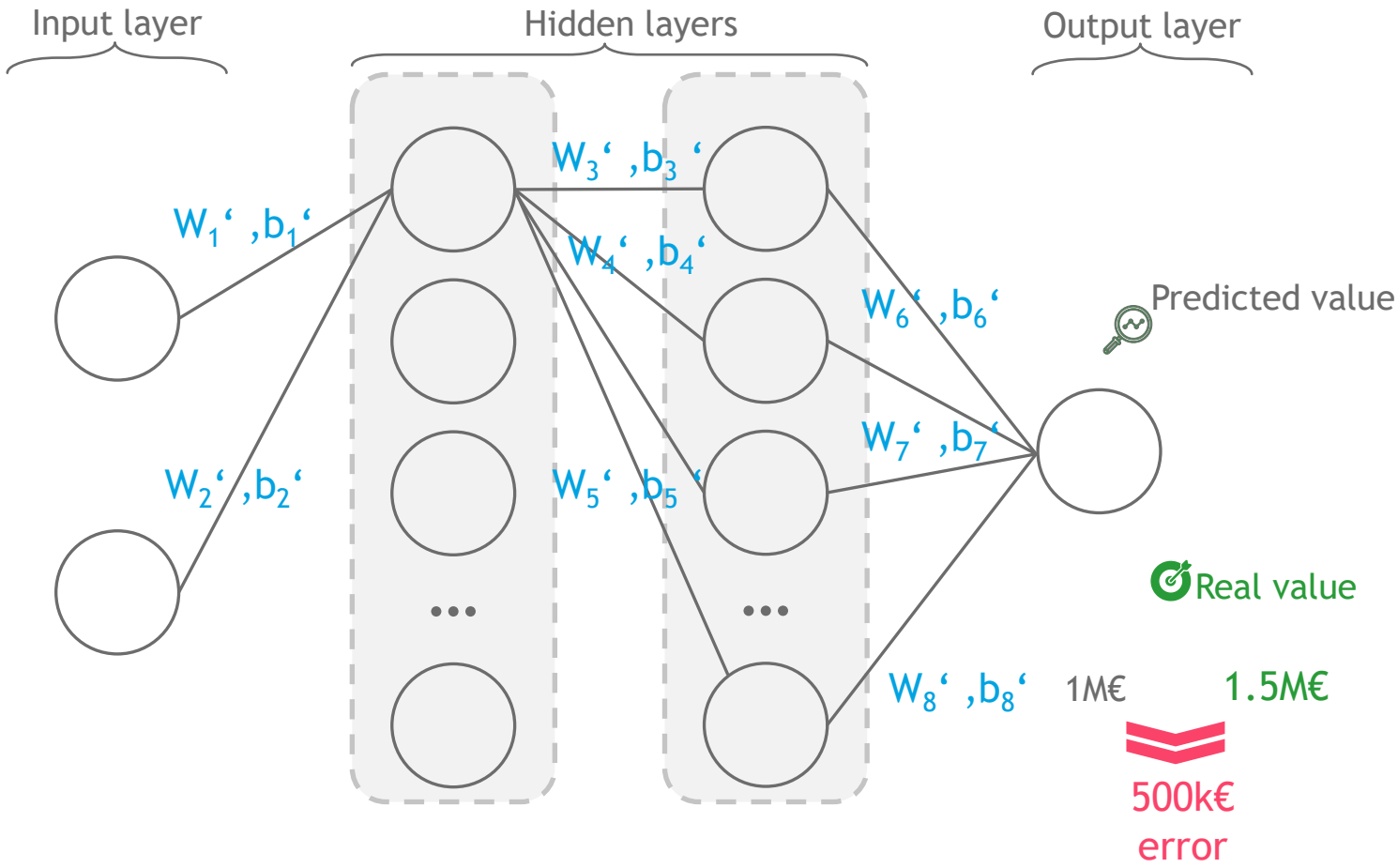


Note

- This process is called the forward propagation and allows to compute the loss function of the model
- How can we find the model's optimal parameters for this task ?

A neural network learn how to imitate a function from a training dataset

Computation process: real estate price prediction example



Note

- This process is called the backward propagation and allows to update the neural network parameters
- Forward and backward propagation are done iteratively until having a high-performance model, by doing a gradient descent to minimize loss

The gradient descent can be controlled by a hyperparameter of the network: the learning rate

Overview of the learning rate

- The learning rate (α) controls the speed at which the model learn
- It controls the update of the weights
- If the learning rate is high, the weights will be strongly modified at each epoch
- A technique to speed up the gradient descent is the learning rate decay, which consists to slowly reduce the learning rate over time : allows large weight changes at the start of the learning process, and more fine-tuning towards the end

Illustration of the learning rate

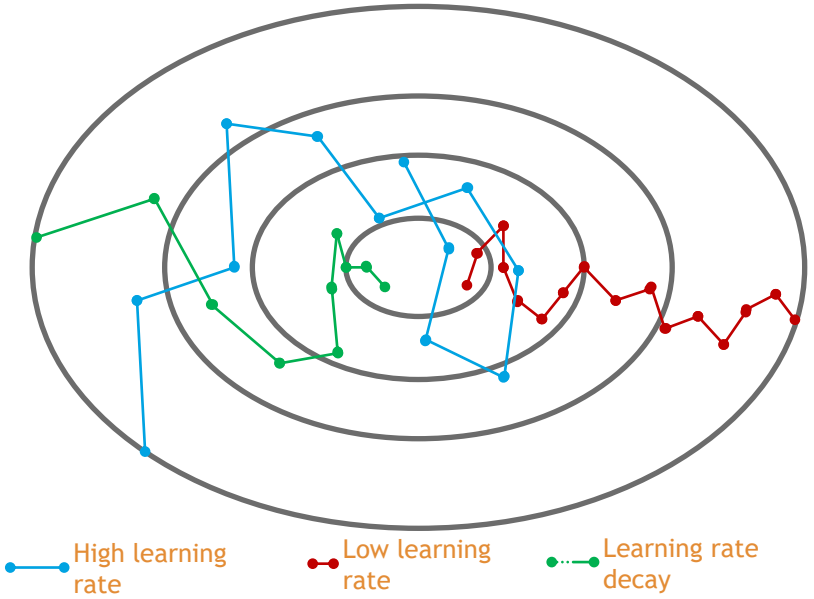


Illustration of the impact of the learning rate on the gradient descent

Note

- The learning rate is equivalent to the length of the step, and it may be the most important hyperparameter to configure our model, and is generally in $[\ 10^{-6}\ , 1]$
- Adaptive Learning Rates is a learning rate schedule adapting the LR to the performance of the model. Typically, once the performance of the model plateaus, LR can be decreased by a factor of 2

On Python, using an open-source machine learning framework like PyTorch can accelerate the path from research prototyping to production deployment

Deep Learning computation

PyTorch

- PyTorch is defined as an open-source machine learning library for Python
- It is designed for Deep Learning and used for applications such as NLP, Computer Vison

Main Features

- Easy Interface:
  - PyTorch offers easy to use API; hence it is considered to be very simple to operate and runs on Python
- Python usage:
  - Pytorch can leverage all the services and functionalities offered by the Python environment
- Computational graphs:
  - PyTorch provides a platform which offers dynamic computational graphs. Thus, a user can change them during runtime

Key advantages

- Easy to debug
- It includes many layers as Torch
- It includes lot of loss functions
- It can be considered as NumPy extension to GPUs
- It allows building networks whose structure is dependent on computation itself

Numerous other tools are useful on Machine Learning projects and can help you during the Hackathon

Other useful libraries

Pandas: handle dataframes and structured data easily

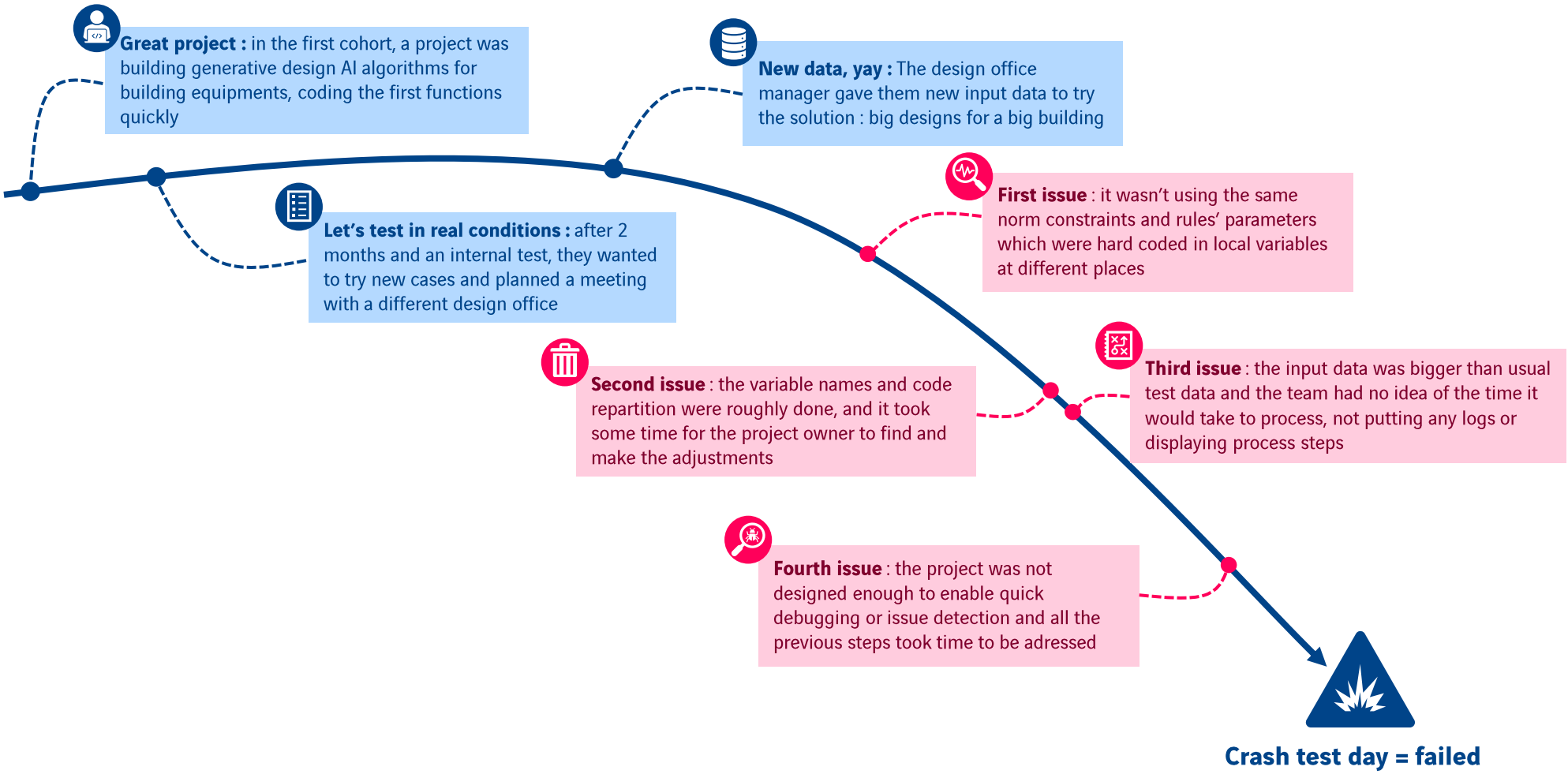
Scikit-Learn: train and evaluate shallow ML models

Jupyter: iterate quickly on your code, keeping your variables across time

GitHub: share your code across the team

Some coding best practices

WHY IS IT IMPORTANT TO WRITE CLEAN CODE?



WHY IS IT IMPORTANT TO WRITE CLEAN CODE?

Great project : in the first cohort, a project was building generative design AI algorithms for building equipments, coding the first functions quickly

New data, yay : The design office manager gave them new input data to try the solution : big designs for a big building

Let's test in real conditions : after 2 months and an internal test, they wanted to try new cases and planned a meeting with a different design office

First issue : it wasn't using the same norm constraints and rules' parameters which were hard coded in local variables at different places

This is when the coding issues started

This is when the coding issues were detected

Second issue : repartition were … some time for the project owner to find and make the adjustments

… issue : the input data was bigger than usual data and the team had no idea of the time it would take to process, not putting any logs or displaying process steps

Fourth issue : the project was not designed enough to enable quick debugging or issue detection and all the previous steps took time to be adressed

Coding best practices is the best return of investment you could ever dream of

Crash test day = failed

PEP8 IS A SET OF CONVENTIONS PROVIDING NAMING GUIDELINES FOR PYTHON

Type	Convention	Examples
Function	• **snake_case**: use **lowercase words, separate words by underscores** • Start with a **verb**	• function, write_name, delete_date, ...
Variable	• **snake_case**: use **lowercase words, separate words by underscores** • Use a **noun**	• first_name, last_name, month, ...
Class	• **CamelCase**: start each word with a **capital letter**, without separators • Use a **noun**	• MyClass, Player, ...
Method	• **snake_case**: Use lowercase words, separate words by underscores • Use **verb**	• class_method, set_player, get_age, ...
Constant	• Use **uppercase** words, **separate** words by **underscores** • Use a **noun**	• CONSTANT, INITIAL_DATE, ...
Module	• **snake_case:** use **lowercase** words, **separate** words by **underscores** • Be **short**, use a **noun**	• module.py, feaure_engineering.py, ...
Package	• Use short **lowercase words** • **Do not separate** words by underscores	• mypackage, ...

OTHER GUIDELINES TO NAME VARIABLES

- Use descriptive names to make it clear what the object represents and use pronounceable names
- Use English language
- Avoid starting a name with Python specifics words like list, from, dict, ...
- Never use l, O, or I single letter names as these can be mistaken for 1 and 0
- And much more to be found here: Pep8

Illustration

✖

```python
def transform_float(values, nb_deciles=10):
    sortedValues = sorted(values)
    aa = [sortedValues[int(k)] for k in np.linspace(0, len(values) - 1, nb_deciles, endpoint=False)]
    xx = np.digitize(values, aa)
    return (xx - xx.min()).astype('str')
```

✔

```python
def get_percentile_values(values, number_of_quantiles=10):
    sorted_values = sorted(values)
    quantile_indices = [sorted_values[int(k)] for k in np.linspace(0, len(values) - 1,
                                                                   number_of_quantiles,
                                                                   endpoint=False)]
    quantiles = np.digitize(values, quantile_indices)
    return (quantiles - quantiles.min()).astype('str')
```

PEP8 ALSO PROVIDES A SET OF RULES FOR CODE LAYOUT

Type	Convention	Examples
Line length	• Limit all lines to a maximum of 79 characters	`# This sentence is 79 characters long - you shouldn't write longer code lines.`

Indentation	• To keep lines under 79 characters, it can be necessary to break the codes with line continuation
	• Use 4 spaces per indentation level
	• Align the indented block either with the opening delimiter (1) or use a hanging incident (2) where every line is indented except the first one
	• Use '\' to separate an expression into two lines
		1
		`def function(arg_one, arg_two,`
		`arg_three, arg_four):`
		`return arg_one`
		2
		`def function(`
		`arg_one, arg_two,`
		`arg_three, arg_four):`
		`return arg_one`
		`var = function(`
		`arg_one, arg_two,`
		`arg_three, arg_four)`

Closing brace	• Line-up closing brace with the first non-whitespace character of the previous line or with the first character of the first line	`my_list = [`
		`1, 2, 3,`
		`4, 5, 6,`
		`]`
		or
		`my_list = [`
		`1, 2, 3,`
		`4, 5, 6,`
		`]`

White space	• Add a single whitespace before and after assignment operators (=, +=,...), comparisons (==, !=,...) and Booleans (and, or, ...)
	• Use a whitespace after a ',' or ';'
	• Do not use a whitespace after '(' or '[' or '{'
		`x = 5`
		`y = 6`
		`print(x, y)`
		`def double(x):`
		`return x * 2`
		`list[3]`

FOR THE AI PROGRAM WE WILL USE A TOOL TO CHECK THE CODE SYNTAX AND LAYOUT CALLED FLAKE8

Quick introduction to linting and flake8

Lint

- Linting is the process of checking the code syntax and provide instructions on how to clean it
- A linter performs lint tasks on a given code or script
- The A.I. program's linter is Flake8

Flake8

- Linter which checks for pep8, pyflakes and circular complexity

Set up

- Flake only works with the python version used to install it
- e.g., python37 –m pip install flake8 will make flake 8 only work with python37

- Helps prevent bugs
- Writing better code
- Saves time in peer reviewer

- Example:

```
python3 -m flake8 path/to/scripts --statistics
```

- `python3`: Python version used to install flake8
- `path/to/scripts`: Path to a folder containing many scripts or to a .py script
- `--statistics`: Get statistics on the detected errors

Output example

```
20    E122 continuation line missing indentation or outdented
1     E124 closing bracket does not match visual indentation
3     E125 continuation line with same indent as next logical line
```

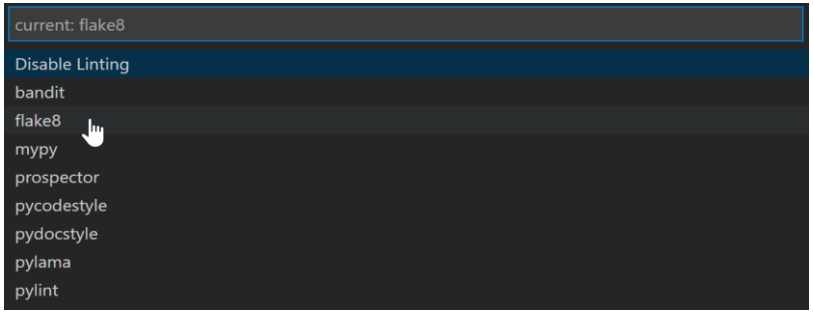
TUTORIAL: ONCE INSTALLED, FLAKE8 CAN BE DIRECTLY USED ON VSC

Setup on VSC

- Make sure Flake8 is installed
- Open the command palette with CTRL + SHIFT + P
- Open the category "Python: Select Linter" and choose Flake8
- Enable linting

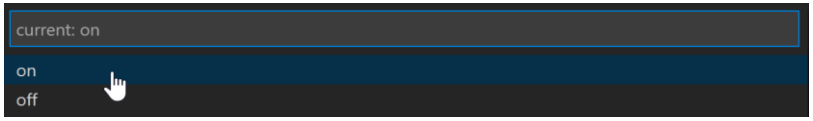
1

2



3

4



Illustration

```
34    for tweet in tweepy.Cursor(api.search,q="#airfrance",
```

twitter_scraping.py 2 of 69 problems

```
missing whitespace after ',' flake8(E231)
```

❌ 1 line is responsible for 10 linting problems !!

```
64    df = pd.read_csv('airfrance.csv', names =['Date','Account','Text','Retweeted','Favorited','Locat
```

twitter_scraping.py 4 of 69 problems

```
unexpected spaces around keyword / parameter equals flake8(E251)
```

twitter_scraping.py 6 of 69 problems

```
missing whitespace after ',' flake8(E231)
```

twitter_scraping.py 10 of 69 problems

```
line too long (128 > 79 characters) flake8(E501)
```

✅ The code is now PEP8 compliant

```
df = pd.read_csv('airfrance.csv', names=['Date', 'Account', 'Text',
                                         'Retweeted', 'Favorited', 'Location',
                                         'Timezone', 'Coordinates'
                                         ]
                 )
```

CODE DOCUMENTATION IS ESSENTIAL TO UNDERSTAND CODE YOU WROTE IN THE PAST OR TO SHARE YOUR CODE WITH OTHERS

Code documentation

- Start a comment with "# "
- Explain WHY decisions are taken and not HOW
- Include information in the variable as much as possible
- DRY: Don't Repeat Yourself
- Use codetags (see PEP350) for tasks like TODO, TODOC, ...

```
# A list of cities to keep
list = ['Paris', 'London']
```

```
cities_to_keep = ['Paris', 'London']
```

Potential risks and inefficiencies

- Over-commenting code
- Lack of comments
- Update the code but not the comments
- Misleading/controversy comment
- Obvious comments
- Comments just because you feel obliged
- Funny comments

- Clean code is a good substitute to many comments
- Comments should be mostly used to explain main ideas or specific details

Python variables

Variables are the corner stones of any python code

Primitive types

int	76	0 zero	-46
float	9.23	0.0	-1.5
bool	True	False	

Container types

str	“hello world”	
set	{1, 5, 9}	set( )
list	[1, 5, 9]	list( )
dict	{1:”one”,2:”two”}	dict( )

Container indexing - for lists, strings...

negative index	-5	-4	-3	-2	-1	
positive index	0	1	2	3	4	
lst =	10	20	30	40	50	
positive slice	0	1	2	3	4	5
negative slice	-5	-4	-3	-2	-1	

Individual access to **items** via lst[index]

- lst[0] → 10 (first one)
- lst[-1] → 50 (last one)
- lst[1] → 20

Modify with assignment lst[4] =45

Access to **sub-sequence** via lst[start slice : end slice : step]

- lst[: -1] → [10, 20, 30, 40]
- lst[1 : -1] → [20, 30, 40]
- lst[: : -1] → [50, 40, 30, 20, 10]
- lst[: : -2] → [50, 30, 10]
- lst[1: 3] → [20, 30]
- lst[:] → [10, 20, 30, 40, 50]

Index **from** 0 (here from 0 to 4)

Items **count** len(lst) → 5

Modify with assignment lst[1 : 3] = [15, 25]

Print

```
a = 2 b = 3
print(a, “ * “, b,” = “, a*b)
print(“{ } * { } = { }”.format(a, b, a*b))
print(f“{a} * {b} = {a*b}”)
```

→ 2 * 3 = 6

Concatenation

“con” + “cat” → “concat”

1 + 2 → 3

1 + 2.5 → 3.5

1 + “cat” ×

[2, 3] + [4, 5] → [2, 3, 4, 5]

{2, 3} + {4, 5} ×

{2, 3}.add(4) → {2, 3, 4}

{1:”one”} + {2:”two”} ×

Conversion

int(“15”) → 15

int(15.56) → 15

float(5) → 5.0

bool(x) False for empty x, None, or False x; True for other x

type(expression)

int(“hello”) ×

str(15) → “15”

set([2,3]) → {2, 3}

list(“abc”) → [‘a’, ‘b’, ‘c’]

dict([(1,”one”),(2,”two”)]) → {1:”one”, 2:”two”}

Additional remarks

Python is a dynamically typed language, which means the type of variable don’t have to be declared

Loops and conditions

Loops and conditions lead to more complex program

Boolean logic

==	is equal
!=	is different
>=	is superior
<=	is inferior
>	is strictly superior
<	is strictly inferior
a and b	both a or b or both simultaneously
a or b	

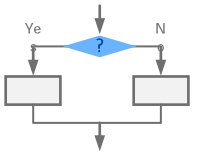
Ensure that a and b are booleans

Conditional statement

Statement block executed only ***if*** a condition is true

```
if logical condition :
    statements
        block
```

Can go with several ***elif***, and only one final ***else***. Only the bloc of the first true condition is executed




```
if age <= 18:
    State = "Kid"
elif age > 65:
    State = "Retired"
else:
    State = "Active"
```

Integer sequence

range([start,] end [, step])

start default 0, end not included in sequence, step default 1

range(5) → 0 1 2 3 4

range(3, 8) → 3 4 5 6 7

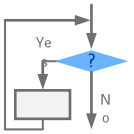
range(5, 12, 3) → 5 8 11

range(20, 5, -5) → 20 15 10

range(len(seq)) → sequence of index of values in seq

Conditional loop statement

Statement block executed **as long as** condition is true




```
while logical condition :
    statements
        block
```

Beware of infinite loop

```
s = 0
i = 1
while i <= 100 :
    s = s + i**2
    i = i + 1
print("sum: ", s)
```

initializations **before** the loop

condition with at least one variable value changing in the loop

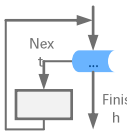
Make condition variable change !

Loop control

break	immediate exit
continue	next iteration

Iterative loop statement

Statement block executed **for each** item of a container or iterator




```
for var in sequence :
    statements
        block
```

```
numbers = [11, 15, 4, 2, 57]
```

Go over sequence's **values**

```
for number in numbers :
```

Go over sequence's **index**

```
for idx in range(len(numbers)) :
```

Good habit: don't modify loop variable

Iterative loop tricks

Go simultaneously over sequence's **index** and **values**:

```
for idx, val in enumerate(lst) :
```

Go simultaneously over two sequences' **values**:

```
for val1, val2 in zip(lst1, lst2) :
```

Additional remarks

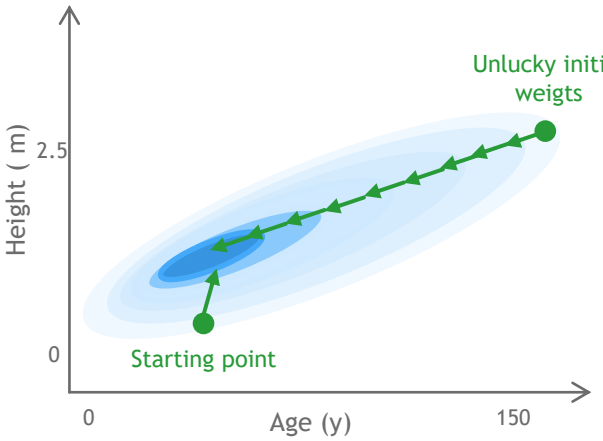
"For" loops are used for iterating through a sequence like a list, a tuple, a string or a dictionary

Appendix

Normalisation throughout the neural network helps ease the optimization process

Why should layers be normalized ?

Use case of cancer prediction based on Height and Age

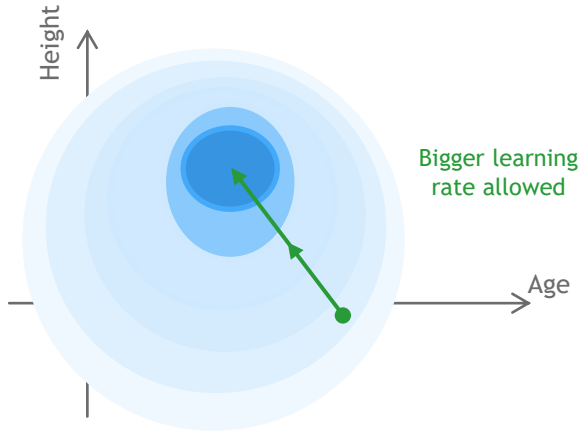


Loss function of the cancer prediction model before normalization

- Need of a small learning rate to reach minimum

Slow computation

Importance of initial weights



Loss function of the cancer prediction model after normalization

Fast computation

Less importance of initial weights

Tips

- Use BatchNorm1d to normalize a layer with PyTorch
- Weight initialization can be “random” by default. Otherwise, it can depend on the type of activation function used : for “Sigmoid” , a xavier initialization method is usually used

“Xavier” initialization comes from a researcher within Google DeepMind , named Xavier Glorot.

Various techniques exist to regularize a neural network and avoid overfitting

How to prevent Neural Network from overfitting ?

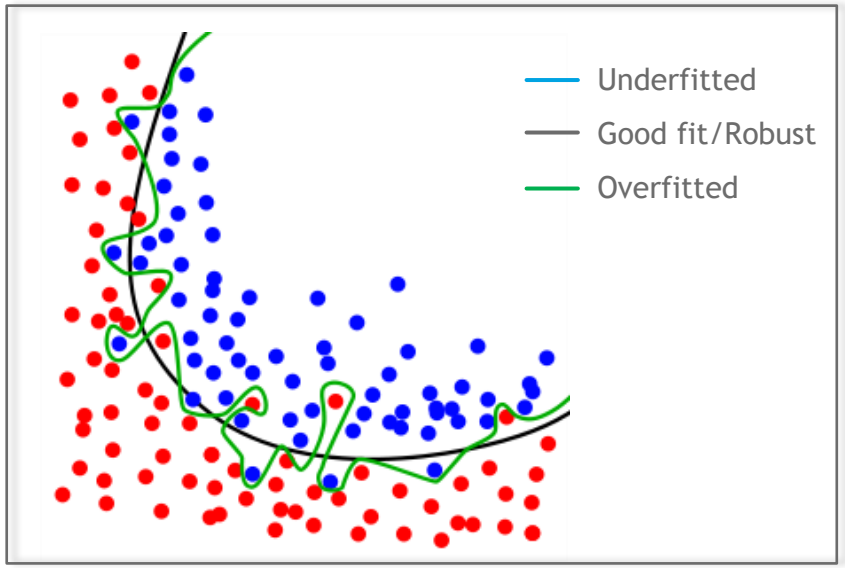


Illustration of a regression problem and overfitting possibilities

1. Data augmentation
2. Weights regularization
3. Early Stopping
4. Dropout

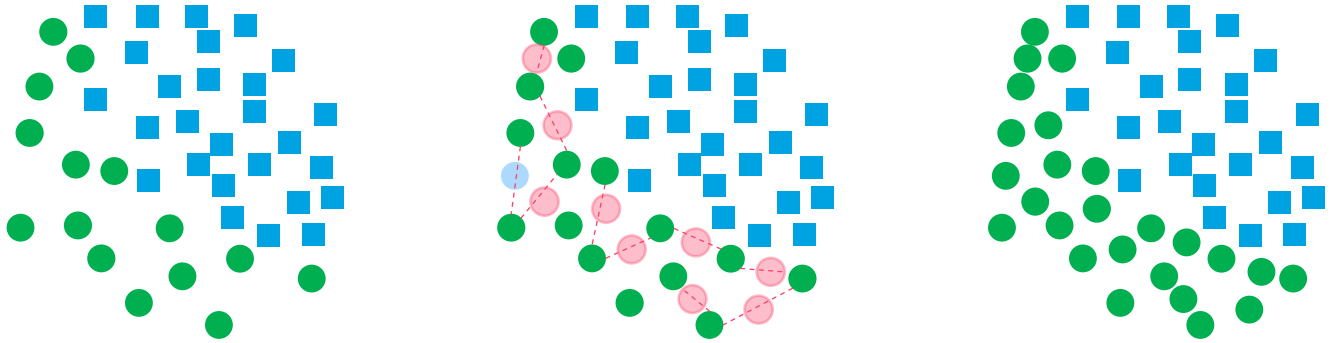
Various techniques exist to regularize a neural network and avoid overfitting

How to prevent Neural Network from overfitting ?

1. Data augmentation
2. Weights regularization
3. Early Stopping
4. Dropout

Illustration

SMOTE : Syntetic Minority oversampling technique



- By increasing the size of data and diversity, data augmentation generalize better, hence preventing overfitting

Various techniques exist to regularize a neural network and avoid overfitting

How to prevent Neural Network from overfitting ?

1 Data augmentation

2 Weights regularization

3 Early Stopping

4 Dropout

Illustration

1 Lasso regularization

New objective function:

$$\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 + \lambda_1 \overbrace{\sum_{j=0}^{n}|\beta_j|}^{\text{L1 norm}}$$

2 Ridge regularization

New objective function:

$$\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 + \lambda_2 \overbrace{\sum_{j=0}^{n}\beta_j^2}^{\text{L2 norm}}$$

ElasticNet data regularization is the combination of both Lasso and Ridge regularization

```
torch.optim.SGD(net.parameters(), lr=0.2, weight_decay=1e-2)
```

Various techniques exist to regularize a neural network and avoid overfitting

How to prevent Neural Network from overfitting ?

1 Data augmentation

2 Weights regularization

3 Early Stopping

4 Dropout

Illustration

- Early stopping stops the training when the validation loss is increasing
- If the validation loss is higher than the current lowest validation lost for a certain number of epochs the training is stopped
- The number of epochs we wait is known as the patience

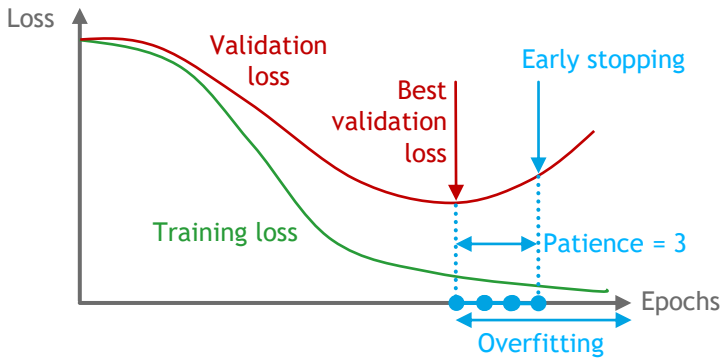


Illustration of early stopping technique

Various techniques exist to regularize a neural network and avoid overfitting

How to prevent Neural Network from overfitting ?

1 Data augmentation

2 Weights regularization

3 Early Stopping

4 Dropout

Illustration

- Not training certain weights at a given update step, leading to greater generalization over the whole network
- The classic formulation of this is dropout where neuron activations are “dropped out” (set to zero) with probability p

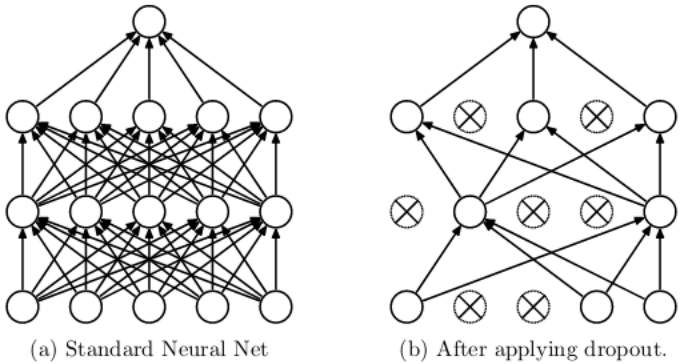


Illustration of dropout technique

```
torch.nn.Dropout(p=0.5)
```