1

Machine Learning IML

P. DE LOOR & C. BUCHE

2

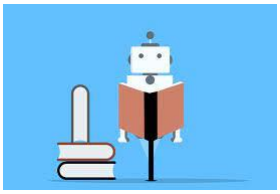
Plan

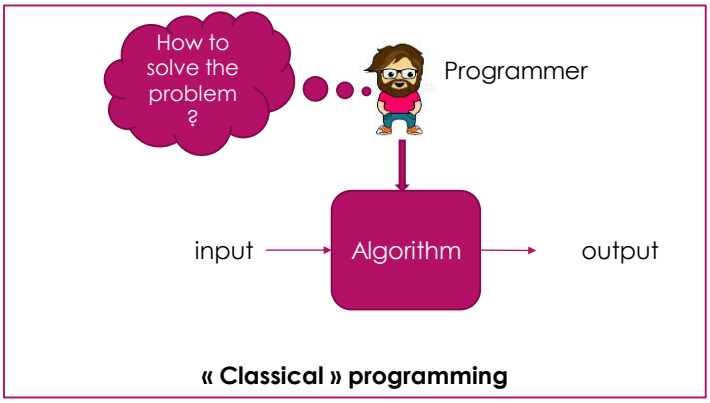
- Machine Learning
- Supervised Regression
  - Linear regression
  - Polynomial regression
- Supervised classification
  - Naive Bayes reasoning
  - Decision Tree
  - Random Forest
  - Logistic Regression
  - Support Vector Machine
  - KNN
- Unsupervised clustering
  - K-Means

3

Machine learning

- Teaching computers to learn to perform tasks from examples



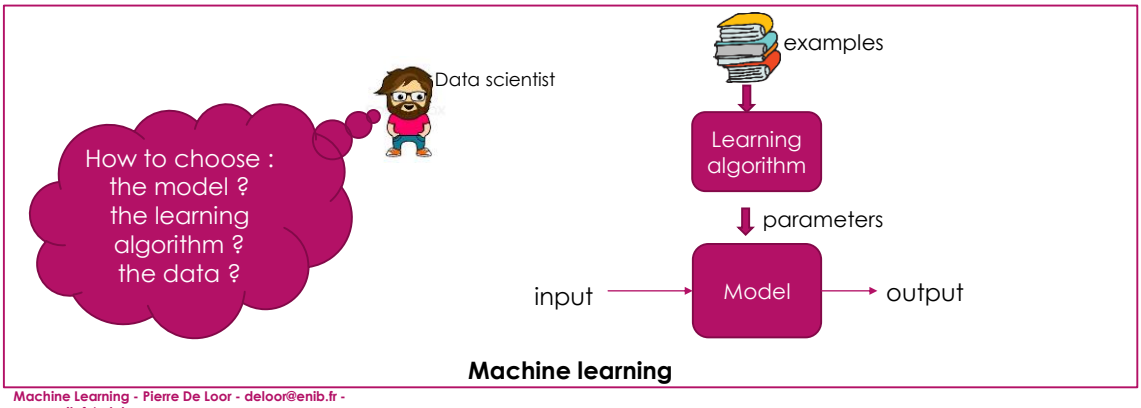


« Classical » programming

4

Machine learning

- Teaching computers to learn to perform tasks from examples



Machine learning

5

There are a lot of models and algorithms

- Depends of :
  - What is the task :
    - Classification
      - Is it a cancer ?
      - What did you say ?
    - Prediction
      - Which advertissement a shopper is most likely to click on ?
      - Which football team is going to win Super Bowl ?
    - Behavior
      - What action to do ?
  - What is « an example »
  - What are the links between inputs and outputs

6

There are a lot of models and algorithms

- Depends of :
  - What is the task :
  - What is « an example »
    - A couple of input/output ? (this image is a bird)
    - An experience that the program test ? (if I turn left, what become my perception ?)
    - An answer that the program ask to a human ? (I need to known the output that correspond to THIS input)
  - What are the links between inputs and outputs

7

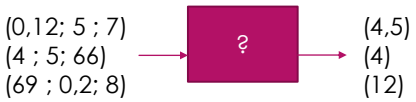
There are a lot of models and algorithms

- Depends of :
  - What is the task :
  - What is « an example »
  - What are the links between inputs and outputs
    - Causalities
    - Probabilities
    - Temporal Sequences
    - And often, there are errors and noise into the data

8

Supervised regression

- The task :
  - Finding the parameters of an equation able to fit to data and to predict new one
- What is an example :
  - Couple (input values / output values)
- Links between input and output
  - Supposed to be a mathematical continuous function (for example : a line for a linear regression)



9

Supervised Regression

$20, 000

$300, 000

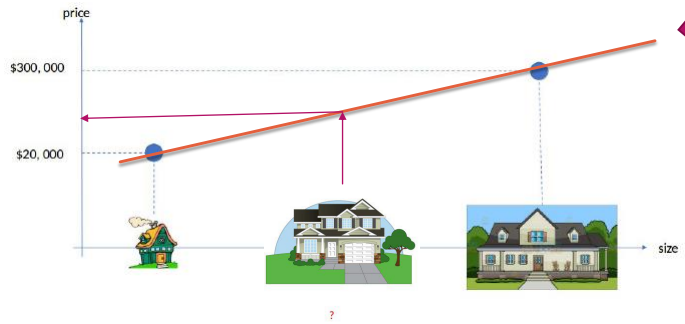
?

10

Supervised Regression

11

Supervised Regression

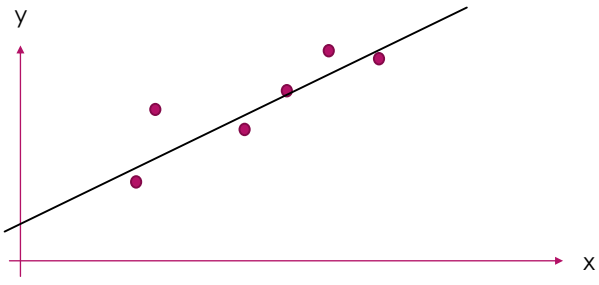


Linear regression

12

Linear Regression

- Finding the line that feet the data

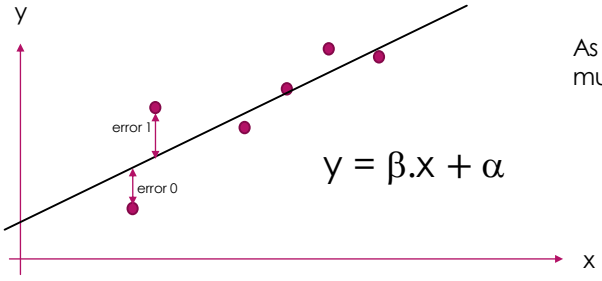


$$y = \beta.x + \alpha$$

13

Linear Regression

- The better line is such that errors are minimal



As errors can be positive or negative, we must minimize de sum of squared errors

Which are the values of β and α that minimize

$$\sum_{k=0}^{n}(error_k)^2$$

14

Linear Regression

- Analytic solution (moindre carrées)

Which are the values of β and α that minimize

$$\sum_{k=0}^{n}(error_k)^2$$

$$\beta = \frac{cov(x,y)}{var(x)} = \frac{\sum\{(x_i-\bar{x})(y_i-\bar{y})\}}{\sum\{(x_i-\bar{x})^2\}} = r(x,y).\sqrt{\frac{var(y)}{var(x)}}$$

$$\alpha = \bar{y} - \beta\bar{x}$$

```
def least_squares_fit ( x , y ):
        beta = correlation ( x , y ) * standard_deviation ( y ) / standard_deviation ( x )
        alpha = mean ( y ) - beta * mean ( x )
        return alpha , beta
```

15

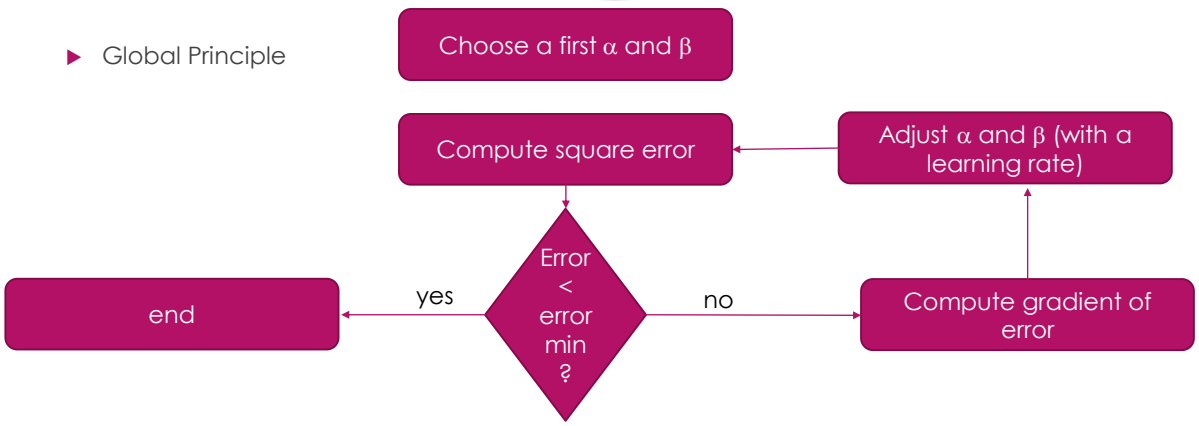
Problems with linear regression

- Memory cost if the size of the data is huge (matrix size)
- Computational cost
- Solution :
  - Gradient descent

16

Gradient descent

- Global Principle



17

Gradient of the sum of squarred errors

- Y is the vector of each yi values
- X is the vector of each xi values
- Sum of errors : $ERROR = Y - (\alpha + \beta.X)$
- Sum of squarred erros : $ERROR^2 = (Y - (\alpha + \beta.X))^2$
- $\frac{d(ERROR^2)}{d\alpha} = -2(Y - (\alpha + \beta.X))$
- $\frac{d(ERROR^2)}{d\beta} = -2X(Y - (\alpha + \beta.X))$

18

Gradient of the sum of squarred errors

- Computation of the gradients for each points of the data to obtain
  - $\frac{d(ERROR^2)}{d\alpha}$ and $\frac{d(ERROR^2)}{d\beta}$
- Multiplication by a learning rate « δ »
- Evolution of the value of β and α
  - $\beta = \beta + \delta\frac{d(ERROR^2)}{d\beta}$
  - $\alpha = \alpha + \delta\frac{d(ERROR^2)}{d\alpha}$
- Until the gradient become « very small »
- The learning rate is decreased progressively during the search

19

Gradient descent

- Code
  - Error computation

```
def error ( alpha , beta , x_i , y_i ):
        return y_i - predict ( alpha , beta , x_i )
```

  - With prediction computation

```
def predict(alpha,beta,x_i):
        return beta * x_i +alpha
```


20

Gradient descent

- Code
  - Squarred error

```
def squared_error ( x_i , y_i , theta ):
        alpha, beta = theta
        return error ( alpha , beta , x_i , y_i ) ** 2
```

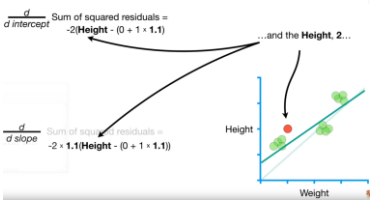
  - Gradient computation

```
def squared_error_gradient ( x_i , y_i , theta ):
        alpha, beta = theta
        return [ - 2 * error ( alpha , beta , x_i , y_i ), # alpha partial deriv
                 - 2 * error ( alpha , beta , x_i , y_i ) * x_i ] # beta partial deriv
```

21

Stochastic gradient descent

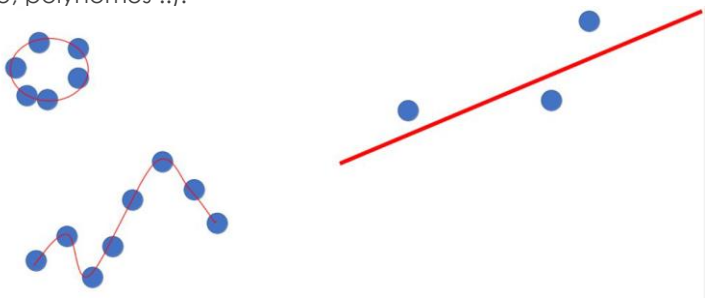
- If the number of data and the dimension of the data is high the time of computation becomes unpractible
- Stochastic gradient descent consists in randomly pick some samples (named batch) for each step of the computation of the parameters (α and β) and not all the data
- https://www.youtube.com/watch?v=vMh0zPT0tLI



22

Polynomial Regression

- Errors can be computed from an equation of a line or of whatever curve (circle, polynomes ..).



23

Supervised Classification

- The task
  - Finding the class of an input
- What is an example
  - A couple (inputs/class)
- What is the link between inputs and outputs
  - Depends of the model :
    - Naive Bayes : Probabilities between the inputs values and the class (independances between input variables)
    - Decition Tree : Hierachie of importances between input parameters to define a class
    - Logistic regression : Hypothesis an increasing function that define a class
    - KNN : Hypothesis of « clusters » defining by inputs values

24

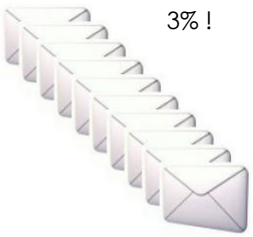
SPAM Detector

70% !

Spam

Is the word « cheap » caracterizing a SPAM ?

What is the probability of a e-mail being a spam if it contains the word « cheap » ?



3% !

Non-Spam

25

Naives bayes reasoning

- Reminder !
- Prior probability : $p(C_k) = p(y = C_k) = \frac{\sum_{i=1}^{N} I(y_i = C_k)}{N}$

$\mathbf{I}$ =1 if y_i = Ck, 0 either

- Conditionnal probability :

$$p(x_1 = a_j | y = C_k) = \frac{\sum_{i=1}^{N} I(x_1 i = a_j, y_i = C_k)}{\sum_{i=1}^{N} I(y_i = C_k)}$$

26

Naives Bayes reasonning

- Example
  - Probability that a message is a SPAM
    - P(S) = 0,2 (prior probabiliy)
  - Probability that a message contains the word **cheap** is 0,23
    - P(X=X1)=0,23 (prior probabiliy)
  - Probability that a SPAM message contains the word **cheap** is 0,7
    - P(X=x1 | S) = 0,7 (conditional probability)
- **Bayes's Theorem :**
  - **P(S | X=x1) =(P(X=x1 | S)*P(S))/P(X=x1)**
  - P(S | X=x1) = 0,7*0,2/(0,23) = 0,60

With :
The message is SPAM : event S
The message contains the word **cheap** : X=x1

27

SPAM Detector

- The word **cheap** is a **feature** that allows for the prediction of a SPAM
- Other possible features :
  - Spelling mistake
  - Missing title
  - The word « Congratulation »
  - …

28

Warning 1 !

- The key to Naive Bayes is making the (big) assumtion that the presences (or absences) of each word are independent of one another.
- $\mathrm{P}(\mathrm{X1}..\mathrm{Xn}|\mathrm{Y}) = \prod_i P(Xi|Y)$

29

How to classify an observation x1..xi ?

- If we have different possible classes C1 … Ck
- We compute the probabily of the observation to belong to each class
- We take the class that have the higher probability

- $y = argmax\ p(y = C_k) \prod_x p(x|y = C_k)$

30

An example

Obs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X1	A	A	A	A	A	B	B	B	B	B	C	C	C	C	C
X2	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0

Prior Probability

P(Y=1) = 9/15 P(Y=0)=6/15

Conditional Probability

P(X1=A|Y=1)=2/9 P(X1=B|Y=1)=3/9 P(X1=C|Y=1)=4/9
P(X2=S|Y=1)=1/9 P(X2=M|Y=1)=4/9 P(X2=L|Y=1)=4/9
P(X1=A|Y=0)=3/6 P(X1=B|Y=0)=2/6 P(X1=C|Y=0)=1/6
P(X2=S|Y=0)=3/6 P(X2=M|Y=0)=2/6 P(X2=L|Y=0)=1/6

31

An example

- What is the class of the observation X(B,S) ?

$P(Y=1)P(X1=B|Y=1)P(X2=S|Y=1)=1/45$
$P(Y=0)P(X1=B|Y=0)P(X2=S|Y=0)=1/15$

32

Laplace Smoothing

- Laplace smoothing solve the problem encounter when the examples don't cover all combination and lead to prior or conditional probablities to « 0 »
- Adding a value λ (often 1 or less)
- Prior probability becomes

$$p_\lambda(C_k) = p_\lambda(Y = C_k) = \frac{\sum_{i=1}^{N} I(y_i = C_k) + \lambda}{N + K\lambda}$$

N denotes the number of examples

$\mathbf{I}$ =1 if y_i = Ck, 0 either

K denotes the bumber of values in Y

33

Laplace smoothing

- A posteriori probability becomes

$$p(x_1 = a_j | y = C_k) = \frac{\sum_{i=1}^{N} I(x_{1i} = a_j, y_i = C_k) + \lambda}{\sum_{i=1}^{N} I(y_i = C_k) + A\lambda}$$

A denotes the number of different values in aj

34

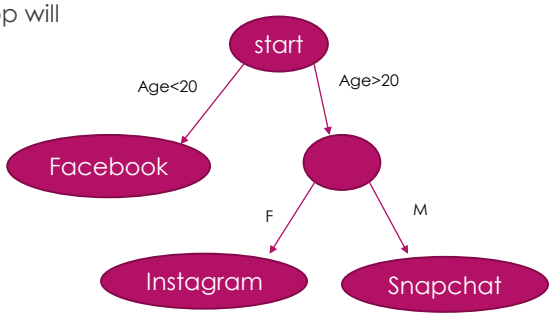
DEMO !

35

Decision Tree

- Which feature (Age or Gender) is more decisive to predict what app will the users downlad ?

Gender	Age	App
F	15	Facebook
F	25	Instagram
M	32	Snapchat
F	40	Instagram
M	12	Facebook
M	14	Facebook

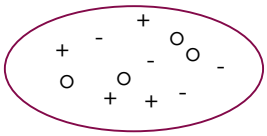


36

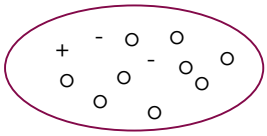
How to find the more relevant feature ?

- Entropy !!

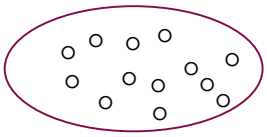
$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i * \log_2(p_i)$$



$$-\frac{4}{12}log_2\left(\frac{4}{12}\right) - \frac{4}{12}log_2\left(\frac{4}{12}\right) - \frac{4}{12}log_2\left(\frac{4}{12}\right) = 1{,}58$$



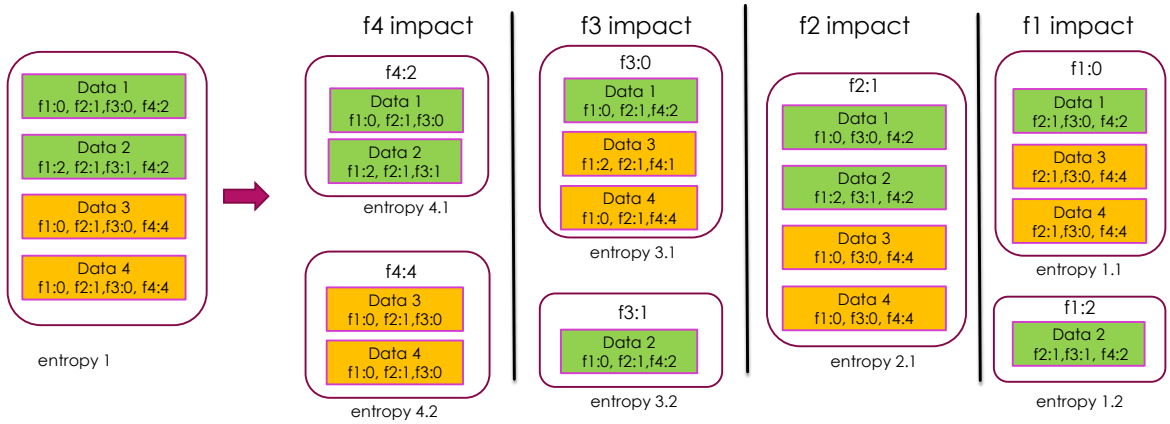
$$-\frac{9}{12}log_2\left(\frac{9}{12}\right) - \frac{2}{12}log_2\left(\frac{2}{12}\right) - \frac{3}{12}log_2\left(\frac{1}{12}\right) = 1{,}04$$



→ 0

37

Entropy comparaison



38

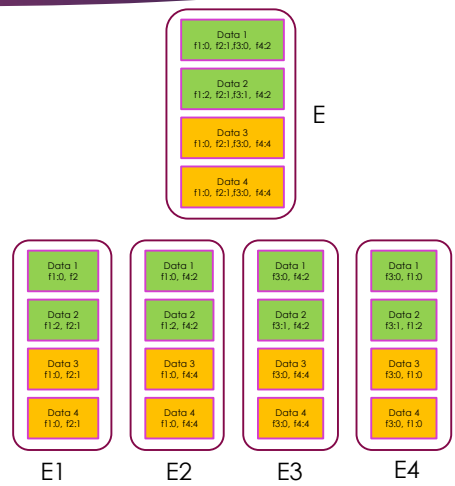
ID3 : Iterative Dichotomiser 3

- Calculate the Information Gain of each feature.
- Considering that all rows don't belong to the same class, split the dataset **S** into subsets using the feature for which the Information Gain is maximum.
- Make a decision tree node using the feature with the maximum Information gain.
- If all rows belong to the same class, make the current node as a leaf node with the class as its label.
- Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

39

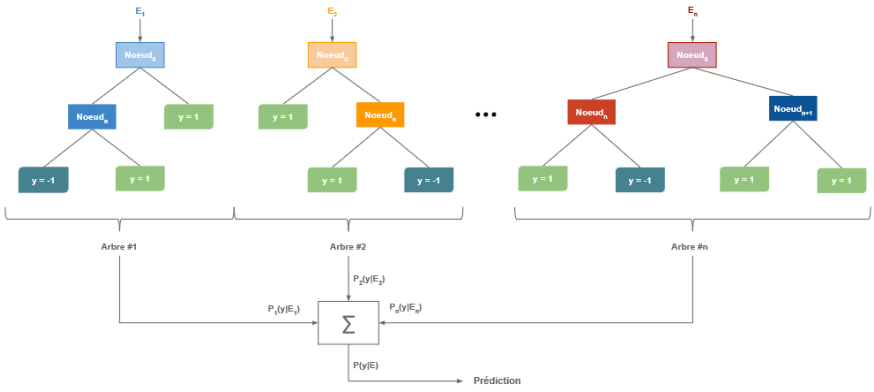
Random Forest

- Rather than making a unique Decision Tree (or whatever classifyers) – from all the features (which can be long and complex) : making many « small classyfiers » based only of a subset of randomly picked up features and process to a votation



40

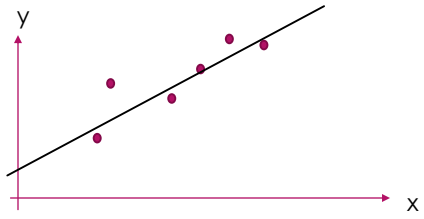
Random Forest



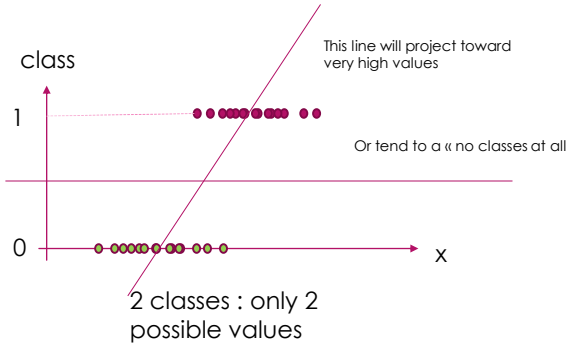
41

Logistic Regression

- A linear regression is adapted to find the value of a continuous variable



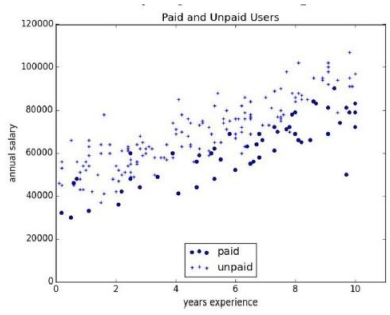
- But what about a classification ?



42

Logic Regression

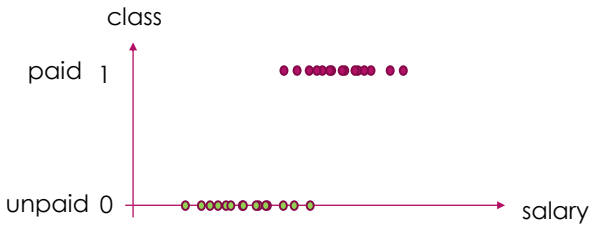
- Examples of real data (input with 2 dimensions, output with 2 classes)



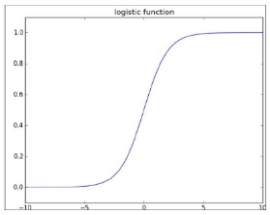
43

Logistic Regression

- For a lot of real cases, the dependency between input and output is on this form



- And the logistic funtion is :
- $\frac{1}{1+e^{-x}}$

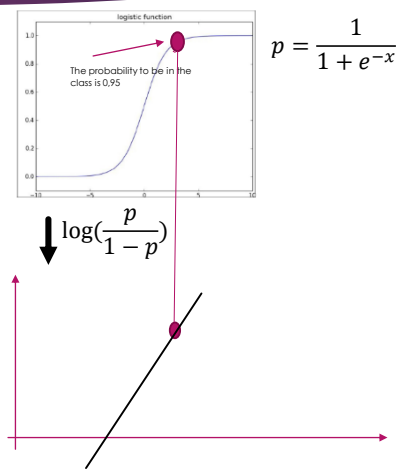


- The value of this function can represent the probability for the data to be in the « paid » class

44

Logistic Regression

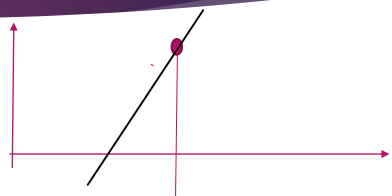
- Logistic regression consists of a regression on the parameters to fit with the data (with the value of the function considered as the probability p (for positive examples) and probability p-1 (for negatlive exampels) to belong to their class $\log(\frac{p}{1-p})$
- The Idea is to transform logistic function in a line (to make a linear regression) by a « log(odd) » function :
- $\log(\frac{p}{1-p})$



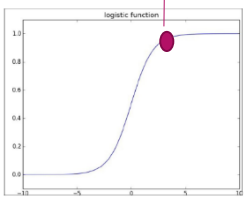
45

Logistic Regression

- **BUT** Mean square errors as criteria to optimize the regression is not possible because errors will tends to $\propto$
- The good criteria is the **Maximum Likelihood**
- The Likelihood is the product of all the probabilities of the examples computed with the logistic function

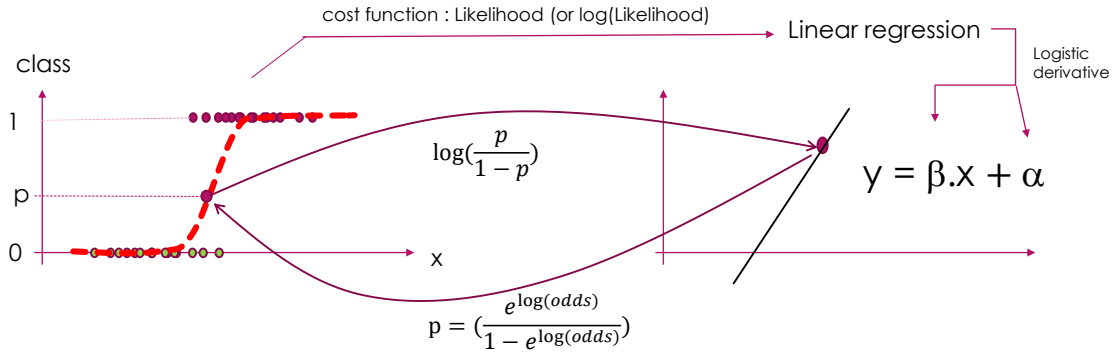


$$p = \left(\frac{e^{\log(odds)}}{1 - e^{\log(odds)}}\right)$$



46

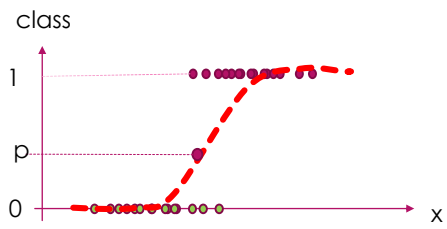
Logistic Regression

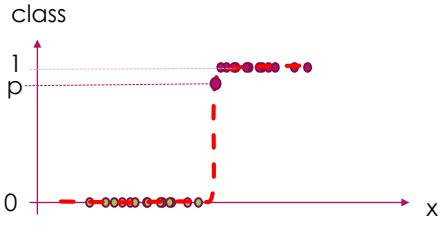


The more the probabilities are close to 1 (for positive case) and close to 0 (for negative case), the high is the likelihood

47

Logistic Regression





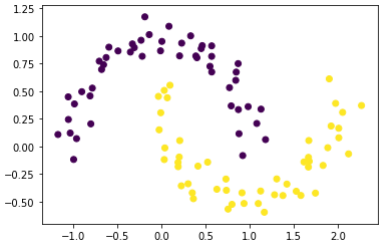
48

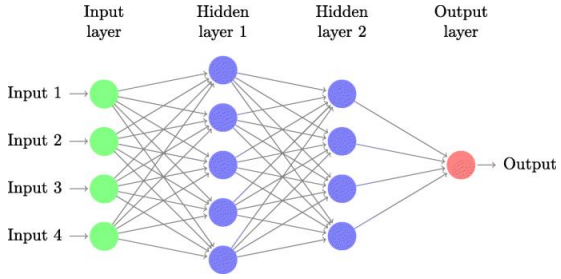
Logistic Regression

- Demo

Neural Networks

- What if the data are like that ?

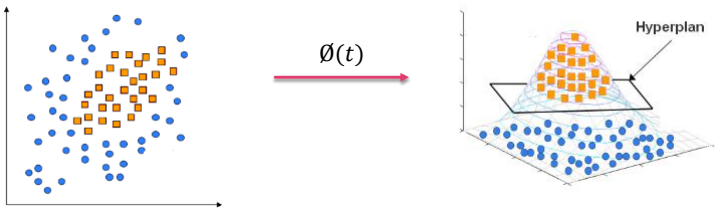




- Specific course on Deep Learning

Support Vector Machine

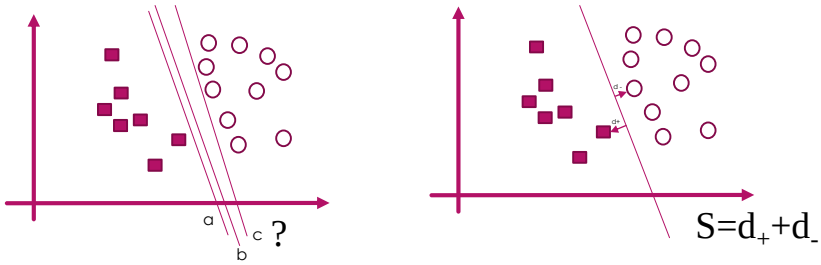
- Description of the data in a new space



51

Support Vector Machine

- Best linear separator
- Maximizing margin implies « the best » capacity to generalize the data



52

Support Vector Machine

- The « Kernel Trick »
- Mathematically, it possible to find the Maximum Marge without having to effectively compute the transformation $\emptyset(x)$ of each the points x ! (which can take a lot of time according to the complexity of $\emptyset$ and the number of data
- From mathematics (Lagrange formulation) it is possible to show that the maximum marge is a function of $\emptyset(x)^T.\emptyset(x)$ and it is possible to find a fonction $K = \emptyset(x)^T.\emptyset(x)$. K is a kernel
- There are different possible kernels and the difficulty is to chose a kernel that favor the best separation of the classes.
- SVM are powerful but the theory behind is difficult
- There are a lot of pyhon libraries for SVM that allows for their use whithout enter into mathematical concepts.

53

KNN : K Nearest Neighbors

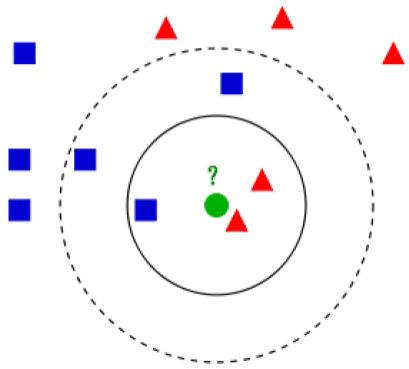
Example : predict how I'm going to vote in the next presidential election.

- If you know nothing else about me, one approach is to look at how my neighbors are planning to vote.
- Living in Seattle, my neighbors are planning to vote for the Democratic candidate, which suggests that Democratic candidate is a good guess for me as well.
- But you know more about me : my age, my income, how many kids I have ... To the extent my behavior is influenced by those things,
- Looking just at my neighbors who are close to me among all those dimensions seems likely to be an even better predictor than looking at all my neighbors.
- This is the idea behind nearest neighbors classification.

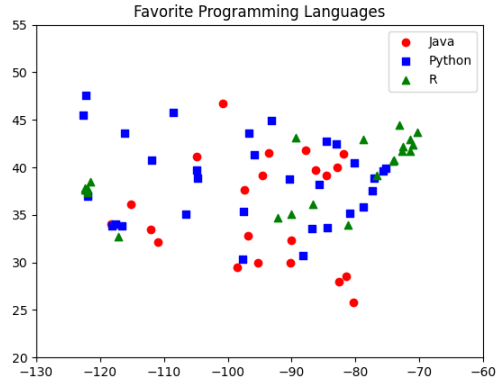
54

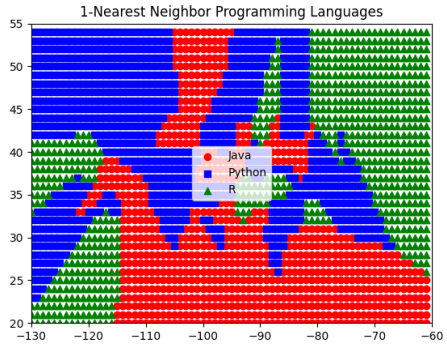
KNN: K Nearest Neighbors

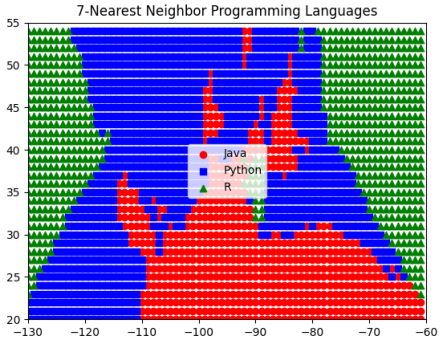
- Requirments
  - Some notion of distance
  - An assumption that points that are close to one another are similar
  - the prediction for each new point depends only on the handful of points closest to it.



knn







Comparison

- **Where Bayes Excels**
- Naive Bayes is a **linear classifier** while K-NN is not; It tends to be faster when applied to big data. In comparison, k-nn is usually slower for large amounts of data, because of the calculations required for each new step in the process. If speed is important, choose Naive Bayes over K-NN.
- In general, Naive Bayes is **highly accurate** when applied to big data. Don't discount K-NN when it comes to accuracy though; as the value of *k* in K-NN increases, the error rate decreases until it reaches that of the ideal Bayes (for $k \to \infty$).
- Naive Bayes offers you two hyperparameters to tune for smoothing: alpha and beta. A hyperparameter is a prior parameter that are tuned on the training set to optimize it. In comparison, K-NN only has one option for tuning: the "*k*", or number of neighbors.
- This method is not affected by the curse of dimensionality and l**arge feature sets**, while K-NN has problems with both.
- For tasks like **robotics** and **computer vision**, Bayes outperforms decision trees.

Source : datasciencecentral.com

57

Comparison

- **Where K-nn Excels**
- If having **conditional independence** will highly negative affect classification, you'll want to choose K-NN over Naive Bayes. Naive Bayes can suffer from the **zero probability problem;** when a particular attribute's conditional probability equals zero, Naive Bayes will completely fail to produce a valid prediction. This could be fixed using a Laplacian estimator, but K-NN could end up being the easier choice.
- Naive Bayes will only work if the **decision boundary** is linear, elliptic, or parabolic. Otherwise, choose K-NN.
- Naive Bayes requires that you known the underlying **probability distributions** for categories. The algorithm compares all other classifiers against this ideal. Therefore, unless you know the probabilities, use of the ideal Bayes is unrealistic. In comparison, K-NN doesn't require that you know anything about the underlying probability distributions.
- K-NN doesn't require any **training**—you just load the dataset and off it runs. On the other hand, Naive Bayes does require training.
- K-NN (and Naive Bayes) outperform decision trees when it comes to **rare occurrences**. For example, if you're classifying types of cancer in the general population, many cancers are quite rare. A decision tree will almost certainty prune those important classes out of your model. If you have any rare occurrences, avoid using decision trees.

Source : datasciencecentral.com

58

Comparison

- **Where Decision trees Excels**
- Of the three methods, decision trees are the **easiest to explain and understand**. Most people understand hierarchical trees, and the availability of a clear diagram can help you to communicate your results. Conversely, the underlying mathematics behind Bayes Theorem can be very challenging to understand for the layperson. K-NN meets somewhere in the middle; Theoretically, you could reduce the K-NN process to an intuitive graphic, even if the underlying mechanism is probably beyond a layperson's level of understanding.
- Decision trees have **easy to use features** to identify the most significant dimensions, handle missing values, and deal with outliers.
- Although **over-fitting** is a major problem with decision trees, the issue could (at least, in theory) be avoided by using boosted trees or random forests. In many situations, boosting or random forests can result in trees outperforming either Bayes or K-NN. The downside to those add-ons are that they add a layer of complexity to the task and detract from the major advantage of the method, which is its simplicity.
- More branches on a tree lead to more of a chance of over-fitting. Therefore, decision trees work best for a **small number of classes.**
- Unlike Bayes and K-NN, decision trees can work directly from a **table of data,** without any prior design work.
- If you don't know your classifiers, a decision tree will **choose those classifiers** for you from a data table. Naive Bayes requires you to know your classifiers in advance.

Source : datasciencecentral.com

59

Unsupervised Clustering

DISCOVERING THE CLASSES

60

No labels

- But some similarities

61

Examples

- A data set showing where millionaires live probably has clusters in places like Beverly Hills and Manhattan.
- A data set showing how many hours people work each week probably has a cluster around 40.
- A data set of demographics of registered voters likely forms a variety of clusters (e.g., "soccer moms", "bored retirees" ...)
  - the clusters won't label themselves. You'll have to do that by looking at the data underlying each one.

62

Distances between data

- K-Means supposes that it is able to compute a distance between data
- Generally the Euclidean distance between attributs of data is used

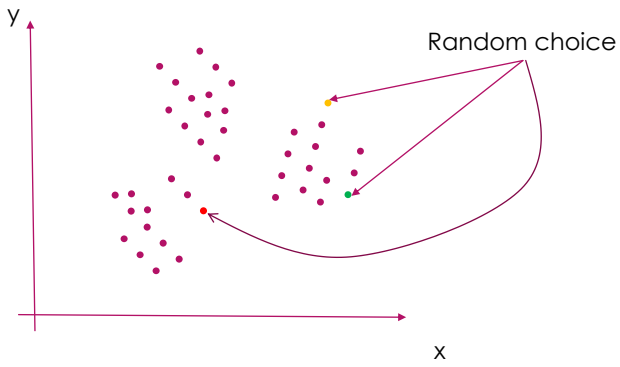
$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{n} (x_{1n} - x_{2n})^2}$$

- But there are a lot of other propositions according to the nature of the data (Manhattan, Minkowski, Chebychev, Canberra, Hamming, Mahalanobis, Pearson correlation distance, Eisen cosine correlation distance …)

K-means

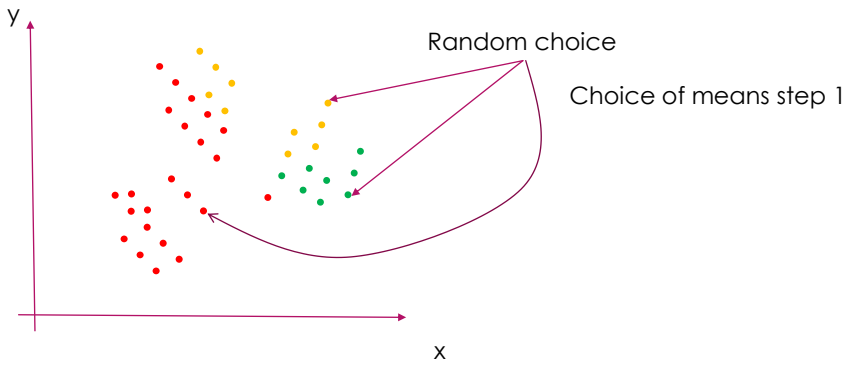
- 1. Start with a set of k-means, which are points in d-dimensional space choose randomly.
- 2. Assign each point to the mean to which it is closest.
- 3. If no point's assignment has changed, stop and keep the clusters.
- 4. If some point's assignment has changed, recompute the means and return to step 2.

Example with 2 attributs



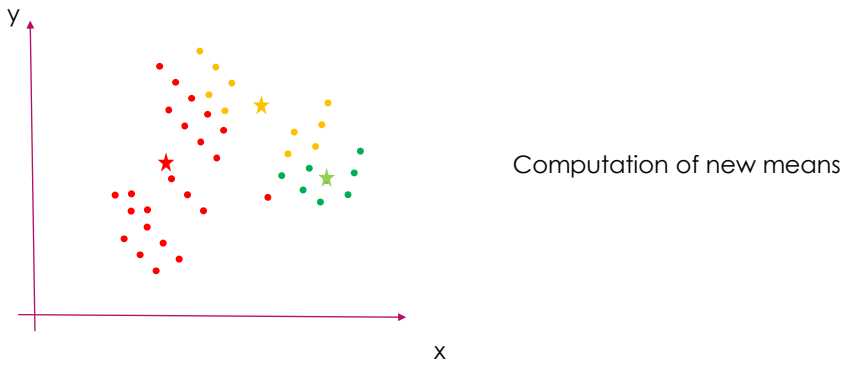
65

Example with 2 attributs



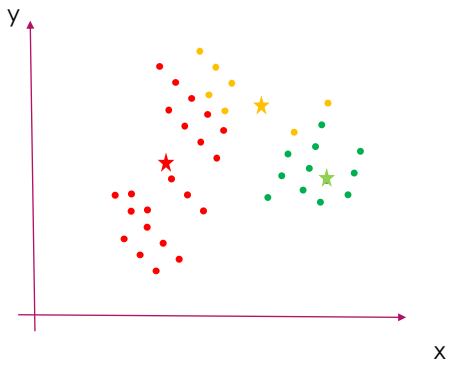
66

Example with 2 attributs



67

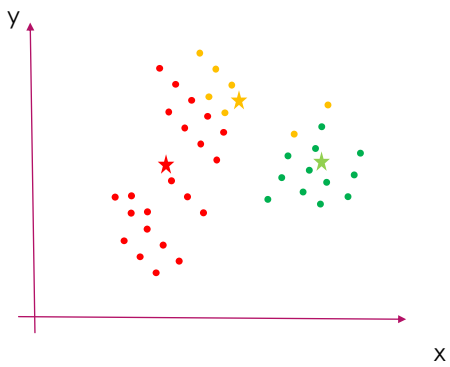
Example with 2 attributs



Choice of means step 2

68

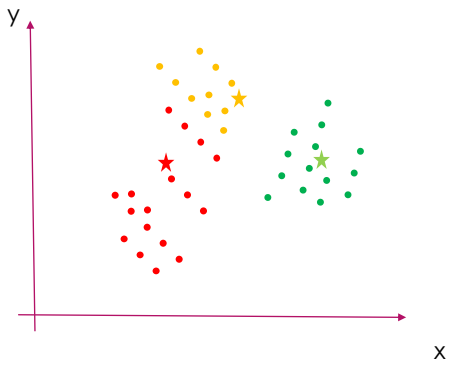
Example with 2 attributs



Computation of new means

69

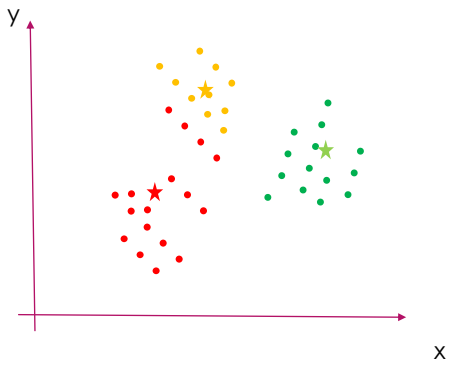
Example with 2 attributs



Choice of new means step 3

70

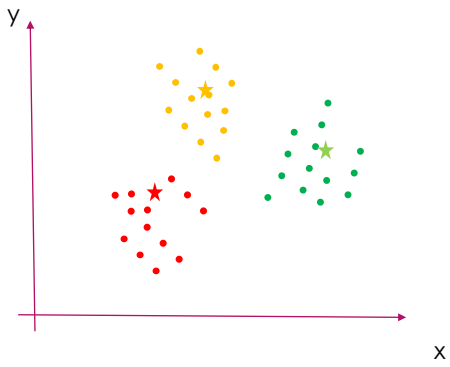
Example with 2 attributs



Computation of new means

71

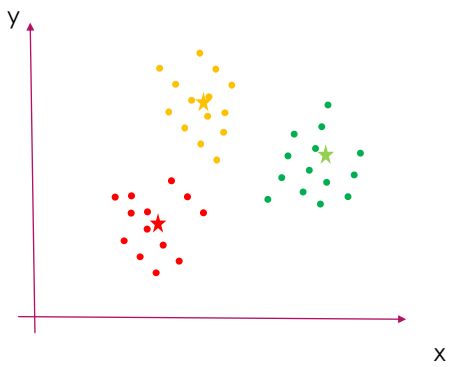
Example with 2 attributs



Choice of means, step 4

72

Example with 2 attributs



Computation of new means

K_means sur des couleurs

K=2

K=3

K=6

Hierarchical clustering

- « grow » clusters from the bottom up
- 1. Make each input its own cluster of one
- 2. As long as there are multiple clusters remaining, find the two closest clusters and merge them
- 3, At the end, we'll have on giant cluster containing all the inputs. If we keep track of the merge order, we can recreate any number of clusters by unmerging. For example, il we want three clusters, we can just undo the last two merges.

Distance

Name	Egg-laying	Scales	Poisonous	Cold-blooded	Legs nb	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes
Frog	True	False	True	True	4	No
Salmon	True	True	False	True	0	No
Python	True	True	False	True	0	Yes

- Choice of features representation = 4 binary and 1 integer
- Boa = (0,1,0,1,0)
- Frog = (1,0,1,0,4)

Euclidean distance

- Alligator is closer to a snake than a frog

	rattlesnake	boa	frog	Alligator
rattlesnake		1.4	1.7	1.4
boa	1.4		2.2	1.4
frog	1.7	2.2		1.7
Alligator	1.4	1.4	1.7	

About distances

77

Distance Measure	Equation	Time complexity	Advantages	Disadvantages	Applications
Euclidean Distance	$d_{euc} = \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}$	O(n)	Very common, easy to compute and works well with datasets with compact or isolated clusters [27,31].	Sensitive to outliers [27,31].	*K*-means algorithm, Fuzzy *c*-means algorithm [38].
Average Distance	$d_{ave} = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2\right)^{\frac{1}{2}}$	O(n)	Better than Euclidean distance [35] at handling outliers.	Variables contribute independently to the measure of distance. Redundant values could dominate the similarity between data points [37].	*K*-means algorithm
Weighted Euclidean	$d_{we} = \left(\sum_{i=1}^{n}w_i(x_i - y_i)^2\right)^{\frac{1}{2}}$	O(n)	The weight matrix allows to increase the effect of more important data points than less important one [37].	Same as Average Distance.	Fuzzy *c*-means algorithm [38]
Chord	$d_{chord} = \left(2 - 2\frac{\sum_{i=1}^{n}x_i y_i}{\|x\|_2\|y\|_2}\right)^{\frac{1}{2}}$	O(3n)	Can work with un-normalized data [27].	It is not invariant to linear transformation [33].	Ecological resemblance detection [35].
Mahalanobis	$d_{mah} = \sqrt{(x-y)S^{-1}(x-y)^T}$	O(3n)	Mahalanobis is a data-driven measure that can ease the distance distortion caused by a linear combination of attributes [35].	It can be expensive in terms of computation [33]	Hyperellipsoidal clustering algorithm [30].
Cosine Measure	$Cosine(x,y) = \frac{\sum_{i=1}^{n}x_i y_i}{\|x\|_2\|y\|_2}$	O(3n)	Independent of vector length and invariant to rotation [33].	It is not invariant to linear transformation [33].	Mostly used in document similarity applications [28,33].
Manhattan	$d_{man} = \sum_{i=1}^{n}(x_i - y_i)$	O(n)	Is common and like other Minkowski-driven distances it works well with datasets with compact or isolated clusters [27].	Sensitive to the outliers. [27,31]	*K*-means algorithm
Mean Character Difference	$d_{MCD} = \frac{1}{n}\sum_{i=1}^{n}\|x_i - y_i\|$	O(n)	*Results in accurate outcomes using the K-medoids algorithm.	*Low accuracy for high-dimensional datasets using K-means.	Partitioning and hierarchical clustering algorithms.
Index of Association	$d_{IOA} = \frac{1}{n}\sum_{i=1}^{n}\left\|x_i\sum_{j=1}^{n}y_j - y_i\sum_{j=1}^{n}x_j\right\|$	O(3n)	-	*Low accuracy using K-means and K-medoids algorithms.	Partitioning and hierarchical clustering algorithms.
Canberra Metric	$d_{canb} = \sum_{i=1}^{n}\frac{\|x_i - y_i\|}{(x_i + y_i)}$	O(n)	*Results in accurate outcomes for high-dimensional datasets using the K-medoids algorithm.	-	Partitioning and hierarchical clustering algorithms.
Czekanowski Coefficient	$d_{czekan} = 1 - \frac{2\sum_{i=1}^{n}\min(x_i, y_i)}{\sum_{i=1}^{n}(x_i + y_i)}$	O(2n)	*Results in accurate outcomes for medium-dimensional datasets using the K-means algorithm.	-	Partitioning and hierarchical clustering algorithms.
Coefficient of Divergence	$d_{canb} = \left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - y_i}{x_i + y_i}\right)^2\right)^{\frac{1}{2}}$	O(n)	*Results in accurate outcomes using the K-means algorithm.	-	Partitioning and hierarchical clustering algorithms.
Pearson coefficient	$Pearson(x,y) = \frac{\sum_{i=1}^{n}(x_i - x_0)(y_i - y_0)}{\sqrt{\sum_{i=1}^{n}(x_i - x_0)^2}\sqrt{\sum_{i=1}^{n}(y_i - y_0)^2}}$	O(2n)	*Results in accurate outcomes using the hierarchical single-link algorithm for high dimensional datasets.	-	Partitioning and hierarchical clustering algorithms.

*Points marked by asterisk are compiled based on this article's experimental results.

doi:10.1371/journal.pone.0144059.t001

Ressources

78

- Videos :
  - Logistic regression : https://www.youtube.com/watch?v=yIYKR4sgzI8
  - Maximum Like : https://www.youtube.com/watch?v=BfKanl1aSG0
  - Kmeans explanation : https://www.youtube.com/watch?v=4b5d3muPQmA
  - Knn : https://www.youtube.com/watch?v=HVXime0nQeI