Machine learning initiation

DATA SCIENCE

Nicolas Sutton-Charani (Euromov - DHM)

IMT Mines Alès
École Mines-Télécom

Terminology

Machine Learning, Data Science, Data Mining, Data Analysis, Statistical Learning, Knowledge Discovery in Databases, Pattern Discovery.

Data everywhere !

- Google : processes 24 peta bytes of data per day.
- Facebook : 10 million photos uploaded every hour.
- Youtube : 1 hour of video uploaded every second.
- Twitter : 400 million tweets per day.
- Astronomy : Satellite data is in hundreds of PB.
- ...
- "By 2020 the digital universe will reach 44 zettabytes..."

Source: The Digital Universe of Opportunities :
Rich Data and the Increasing Value of the Internet of Things, April 2014.

That's 44 trillion gigabytes !

Data types

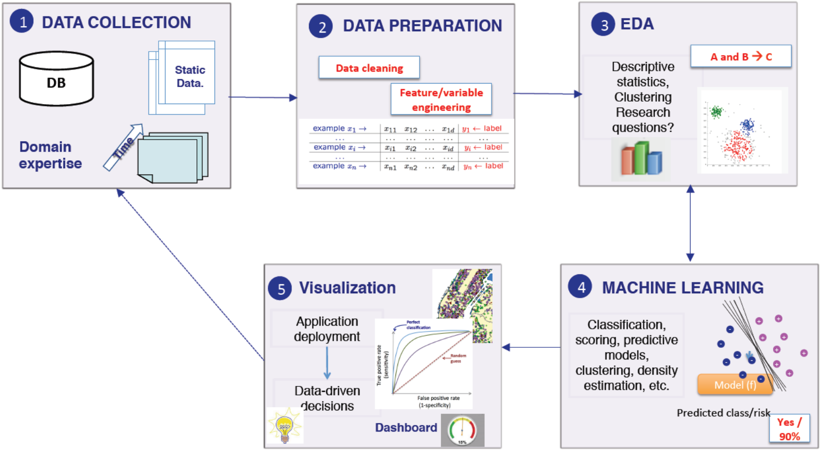
Data comes in different sizes and also flavors (types) :

- Texts
- Numbers
- Clickstreams
- Graphs
- Tables
- Images
- Transactions
- Videos
- Some or all of the above !

Smile, we are 'DATAFIED' !

- Wherever we go, we are "datafied".
- Smartphones are tracking our locations.
- We leave a data trail in our web browsing.
- Interaction in social networks.
- Privacy is an important issue in Data Science.

The Data Science process



Applications of ML

- We all use it on a daily basis.

  Examples :

Machine Learning

- Spam filtering
- Credit card fraud detection
- Digit recognition on checks, zip codes
- Detecting faces in images
- MRI image analysis
- Recommendation system
- Search engines
- Handwriting recognition
- Scene classification
- etc...

Interdisciplinary field

ML versus Statistics

Statistics :

- Hypothesis testing
- Experimental design
- Anova
- Linear regression
- Logistic regression
- GLM
- PCA

Machine Learning :

- Decision trees
- Rule induction
- Neural Networks
- SVMs
- Clustering method
- Association rules
- Feature selection
- Visualization
- Graphical models
- Genetic algorithm

Source: http ://statweb.stanford.edu/ jhf/ftp/dm-stat.pdf

Outline

Outline

Machine Learning definition

"How do we create computer programs that improve with experience ?"

Tom Mitchell

http ://videolectures.net/mlas06_mitchell_itm/

Machine Learning definition

"How do we create computer programs that improve with experience?"

Tom Mitchell

http ://videolectures.net/mlas06_mitchell_itm/

"A computer program is said to ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by M, improves with experience E."*

Tom Mitchell

Machine Learning 1997.

Supervised vs. Unsupervised

Training data : $(x, y) = (x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \Omega^1 \times \cdots \times \Omega^d$ and $y_i \in E$ the label

$$
\begin{array}{r|cccc|}
\text{example } x_1 \rightarrow & x_1^1 & x_1^2 & \ldots & x_1^d \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\text{example } x_i \rightarrow & x_i^1 & x_i^2 & \ldots & x_i^d \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\text{example } x_n \rightarrow & x_n^1 & x_n^2 & \ldots & x_n^d
\end{array}
$$

Supervised vs. Unsupervised

Training data : $(x, y) = (x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \Omega^1 \times \cdots \times \Omega^d$ and $y_i \in E$ the label

$$
\begin{array}{c|cccc|c}
\text{example } x_1 \rightarrow & x_1^1 & x_1^2 & \ldots & x_1^d & y_1 \leftarrow \text{label} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{example } x_i \rightarrow & x_i^1 & x_i^2 & \ldots & x_i^d & y_i \leftarrow \text{label} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{example } x_n \rightarrow & x_n^1 & x_n^2 & \ldots & x_n^d & y_n \leftarrow \text{label}
\end{array}
$$

Supervised vs. Unsupervised

Training data : $(x, y) = (x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \Omega^1 \times \cdots \times \Omega^d$ and $y_i \in E$ the label

$$
\begin{array}{r|cccc|l}
\text{example } x_1 \rightarrow & x_1^1 & x_1^2 & \ldots & x_1^d & y_1 \leftarrow \text{label} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{example } x_i \rightarrow & x_i^1 & x_i^2 & \ldots & x_i^d & y_i \leftarrow \text{label} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{example } x_n \rightarrow & x_n^1 & x_n^2 & \ldots & x_n^d & y_n \leftarrow \text{label}
\end{array}
$$

fruit	length	width	weight	label
fruit 1	165	38	172	Banana
fruit 2	218	39	230	Banana
fruit 3	76	80	145	Orange
fruit 4	145	35	150	Banana
fruit 5	90	88	160	Orange
...				
fruit n	...	...	...	...

Supervised vs. Unsupervised

fruit	length	width	weight	label
fruit 1	165	38	172	Banana
fruit 2	218	39	230	Banana
fruit 3	76	80	145	Orange
fruit 4	145	35	150	Banana
fruit 5	90	88	160	Orange
...				
fruit n	...	...	...	...

Unsupervised learning

Learning a model from **unlabeled** data.

Supervised learning

Learning a model from **labeled** data.

Supervised vs. Unsupervised

- we start with n examples described by d features
  $X = X^1, ..., X^d \in \Omega^1 \times \cdots \times \Omega^d$
  and eventually labelled with $Y \in E$ (training data).

- we suppose that the $(x_i, y_i) = (x_i^1, ..., x_i^d, y_i)$ are realisations of random variables $(X, Y) = (X^1, ..., X^d, Y)$.

- we try to compute/*learn* a function f that
  - characterise $X \rightarrow f(X) =$ profiling $\rightarrow$ unsupervised learning
  - maps X to $Y \rightarrow Y = f(X)$ $\rightarrow$ supervised learning

Unsupervised Learning

Training data : "examples" x.

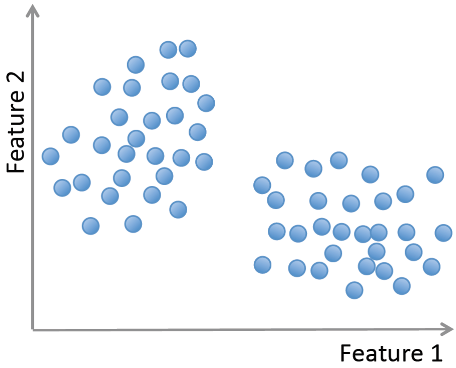
$$x_1, ..., x_n \text{ with } x_i \in \Omega^1 \times \cdots \times \Omega^d,$$

- Clustering/segmentation :

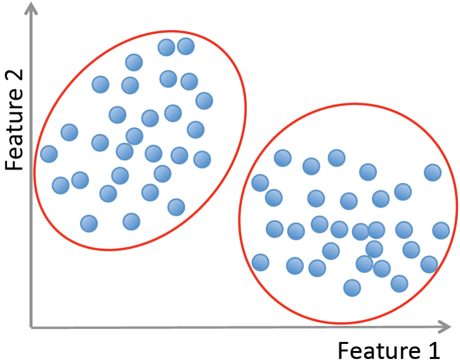
$$f : \Omega^1 \times \cdots \times \Omega^d \longrightarrow \{C_1, ..., C_k\} \text{ set of clusters}$$

Example : Find clusters in the population, fruits, species.

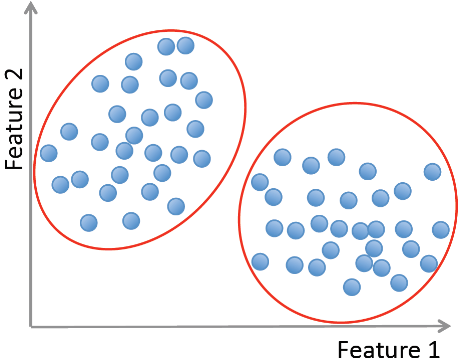
Unsupervised Learning



Unsupervised Learning



Unsupervised Learning



Methods : K-means, gaussian mixtures, hierarchical clustering, spectral clustering, etc.

Supervised learning

Training data : "examples" x with "labels" y.

$(x, y) = (x_1, y_1), \dots, (x_n, y_n)$ with $x_i \in \Omega^1 \times \cdots \times \Omega^d$ and $y_i \in E$

We want a model to map X and Y (from x and y).

- $E \subset \mathbb{R} \rightarrow$ regression
- $E = \{C_1, ..., C_k\}$ *categorical* set $\rightarrow$ classification

Supervised learning

Training data : "examples" x with "labels" y.

$$(x_1, y_1), \ldots, (x_n, y_n), \text{ with } x_i \in \Omega^1 \times \cdots \times \Omega^d \text{ and } y_i \in E \subset \mathbb{R}$$

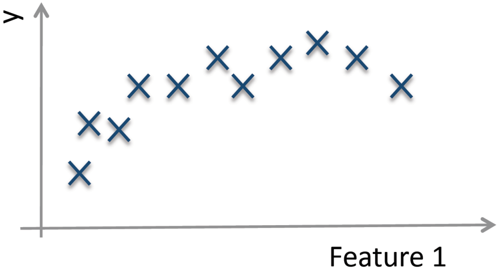
- Regression : y is a real value, $y \in \mathbb{R}$

  $$f : \Omega^1 \times \cdots \times \Omega^d \longrightarrow E \qquad \text{f is called a } \textbf{regressor}.$$

  Example : amount of credit, weight of fruit.

Supervised learning

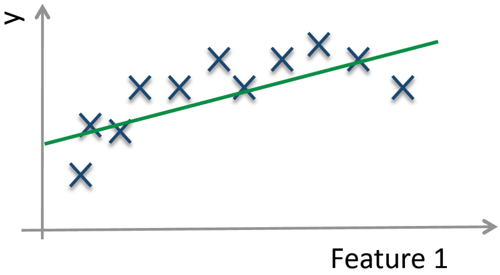
Regression :



Example : Income in function of age, weight of the fruit in function of its length.

Supervised learning

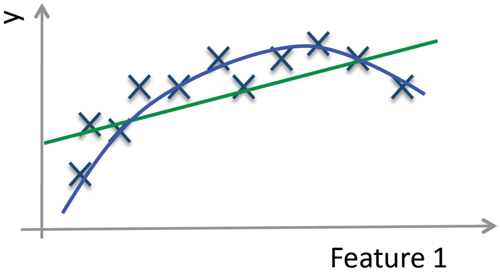
Regression :



Example : Income in function of age, weight of the fruit in function of its length.

Supervised learning

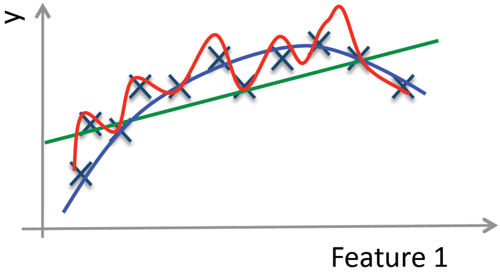
Regression :



Example : Income in function of age, weight of the fruit in function of its length.

Supervised learning

Regression :



Example : Income in function of age, weight of the fruit in function of its length.

Supervised learning

Training data : "examples" x with "labels" y.

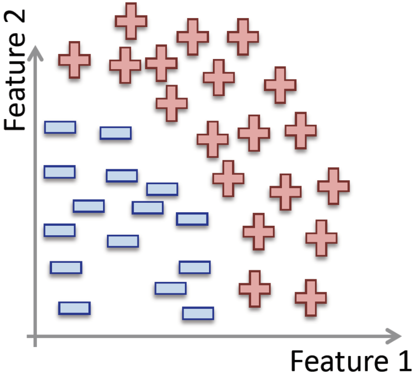
$$(x_1, y_1), \dots, (x_n, y_n), \text{ with } x_i \in \Omega^1 \times \cdots \times \Omega^d \text{ and } y_i \in E \not\subset \mathbb{R}$$

- Classification : y is discrete. To simplify (binary case), $y \in \{-1, +1\}$

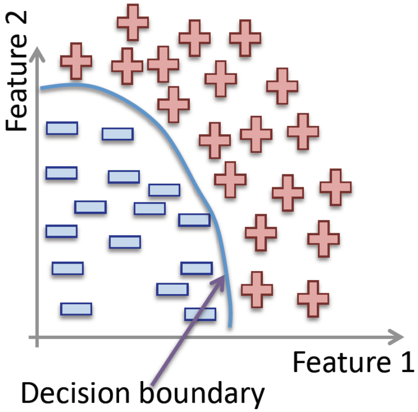
  $f : \mathbb{R}^d \longrightarrow \{-1, +1\}$ f is called a binary **classier**.

  Example : Approve credit yes/no, spam/ham, banana/orange.

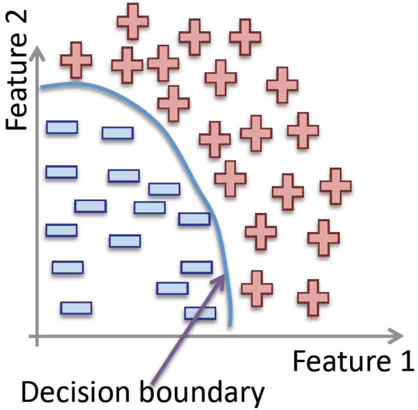
Supervised Learning



Supervised Learning



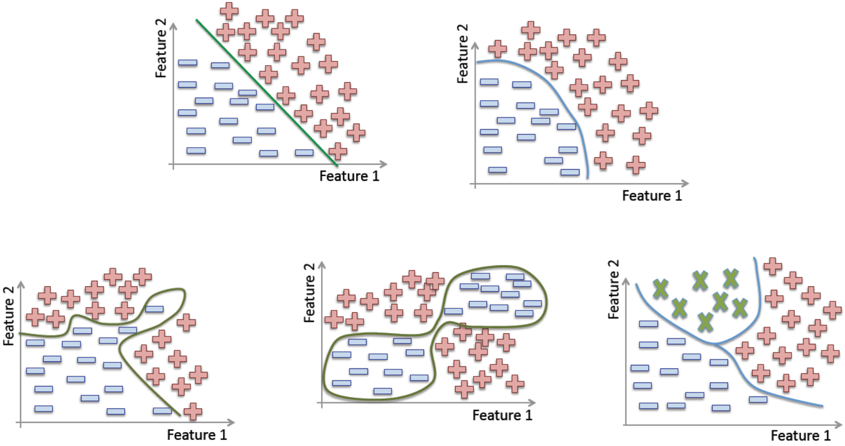
Supervised Learning



Methods : Support Vector Machines, neural networks, decision trees, K-nearest neighbors, naive Bayes, etc.

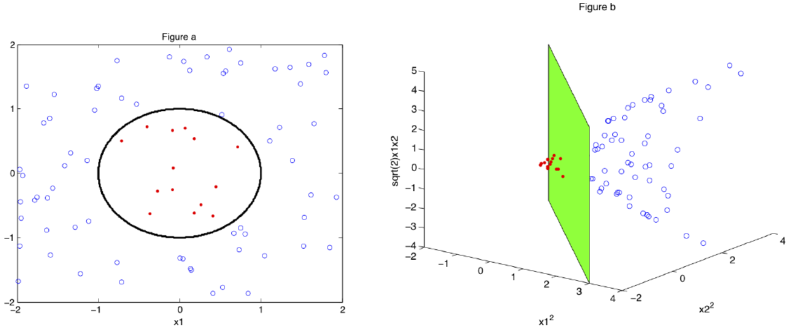
Supervised Learning

Classification :



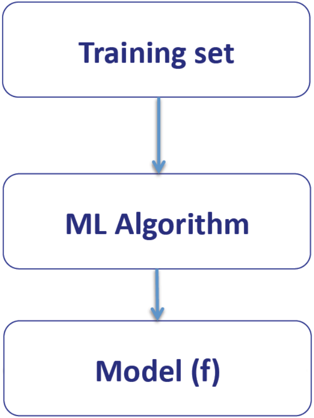
Supervised Learning

Non linear classification :

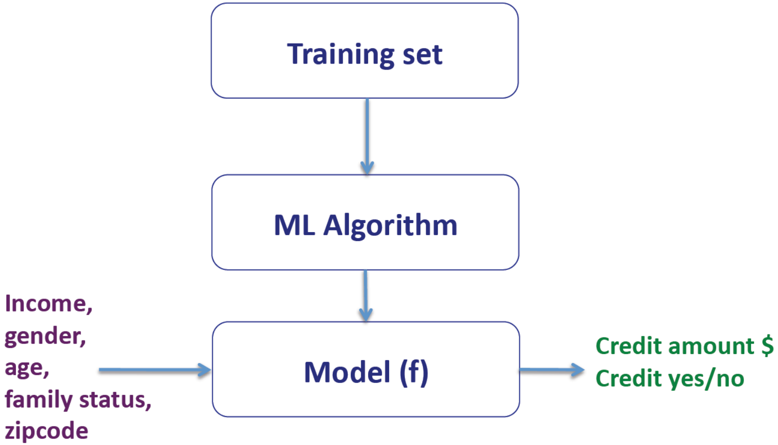


Outline

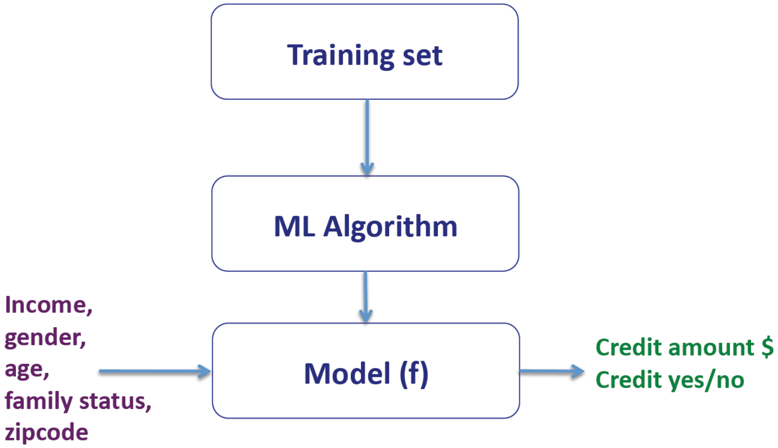
Training and Testing



Training and Testing



Training and Testing



Question : How can we be confident about f ?

Training and Testing

- Training set is a set of examples used for learning a model (e.g., a classification model).
- Test set is used to assess the performance of the final model and provide an estimation of the test error.

Note : Never use the test set in any way to further tune the parameters or revise the model.

K-fold Cross Validation

A method for estimating test error using training data.

Data: learning algorithm A and a dataset D

Result: test performance estimator $\hat{E}$

Step 1 : Randomly partition D into K equal-size subsets $D_1, ..., D_K$
Step 2 :
for $k = 1$ *to* K **do**
  Train A on all D_l, $l \in \{1, ..., K\}$ and $l \neq k$, and get f_k;
  Apply f_k to D_k and compute $Perf(f_k, D_k)$;
end
Step 3 : Average error over all folds :

$$\hat{E} = \frac{1}{K} Perf(f_k, D_k)$$

Confusion matrix

		Actual label	
		Positive	Negative
Predicted label	Positive	**True Positive (*TP*)**	**False Positive (*FP*)**
	Negative	**False Negative (*FN*)**	**True Negative (*TN*)**

Evaluation metrics

		Actual label	
		Positive	Negative
Predicted label	Positive	**True Positive** (TP)	**False Positive** (FP)
	Negative	**False Negative** (FN)	**True Negative** (TN)

Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	The proportion of correct predictions
Precision	$\frac{TP}{TP+FP}$	The proportion of positive predictions that were actually positive
Recall or *sensibility*	$\frac{TP}{TP+FN}$	The proportion of positive cases that were predicted positive
Specificity	$\frac{TN}{TN+FP}$	The proportion of negative cases that were predicted negative

Evaluation metrics

Once a supervised model f is learnt $\rightarrow$ prediction $\widehat{y_i} = f(x_i)$ of y_i

EVALUATION : comparison between the y_i and $\widehat{y_i}$ terms.

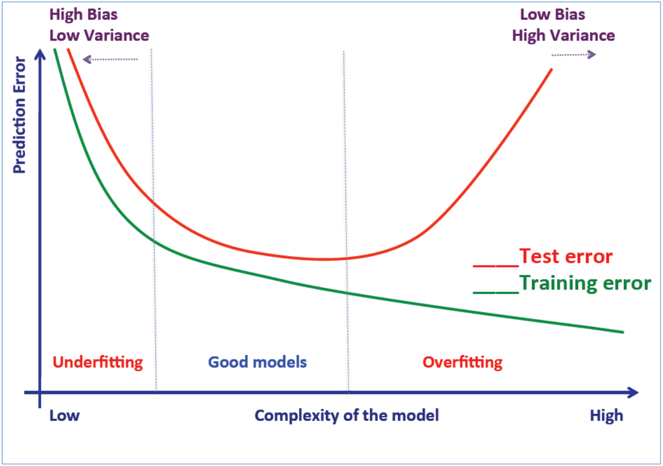
Regression

- **Root Mean Square Error** : $RMSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2$
- **Mean Absolute Error** : $MAE = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y_i} - y_i|$

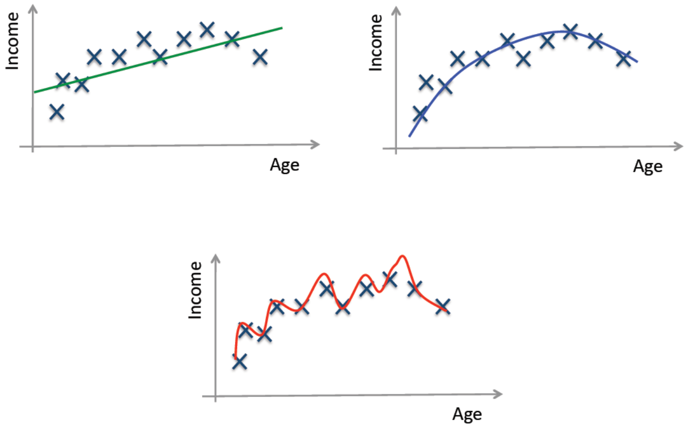
Classification

- **accuracy** : $acc = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{\{\widehat{y_i}=y_i\}}$
- **binary case** : precision, recall, F-measure, AUC, etc.

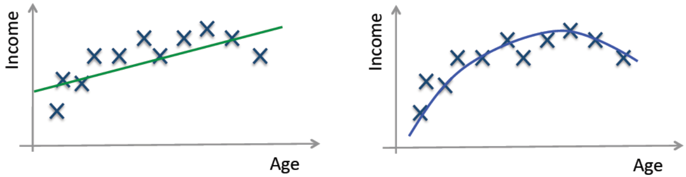
Structural Risk Minimization



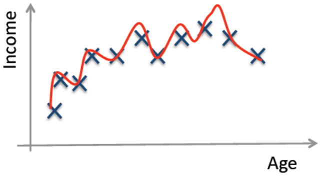
Training and Testing



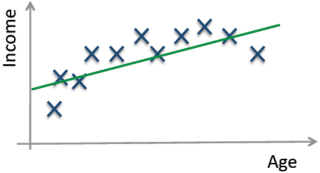
Training and Testing

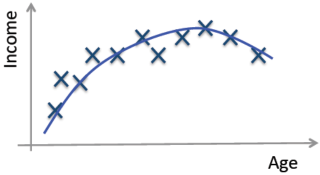


High bias (underfitting)

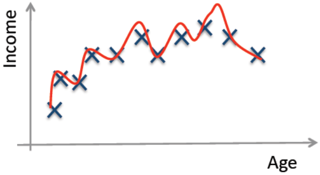


Training and Testing



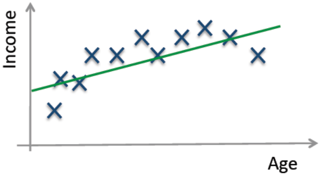


High bias (underfitting)

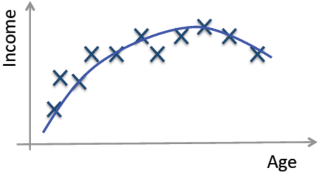


High variance (overfitting)

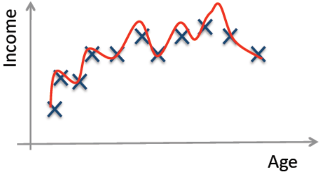
Training and Testing



High bias (underfitting)



Just right!



High variance (overfitting)

Avoid overfitting

In general, use simple models !

- **Reduce the number** of features manually or do feature selection.
- Do a **model selection**.
- Do a **cross-validation** to estimate the test error.

Outline

Unsupervised Learning

Training data : "examples" x.

$$x_1, \dots, x_n \text{ with } x_i \in \Omega^1 \times \dots \times \Omega^d$$

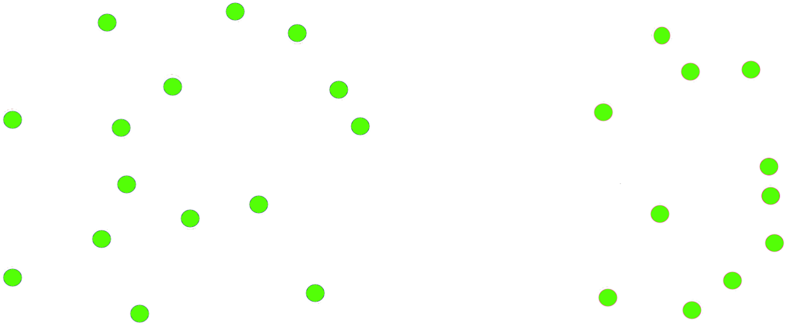
- Clustering/segmentation :

$$f : \Omega^1 \times \dots \times \Omega^d \longrightarrow \{C_1, ..., C_k\} \text{ set of clusters}$$

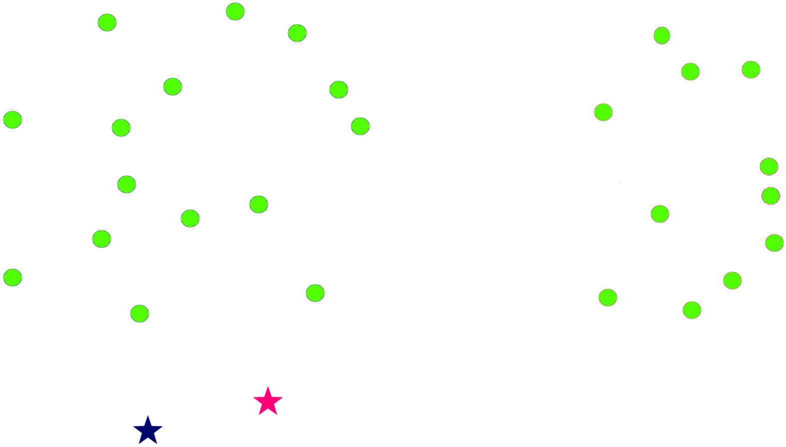
Example : Find clusters in the population, fruits, species.

Outline

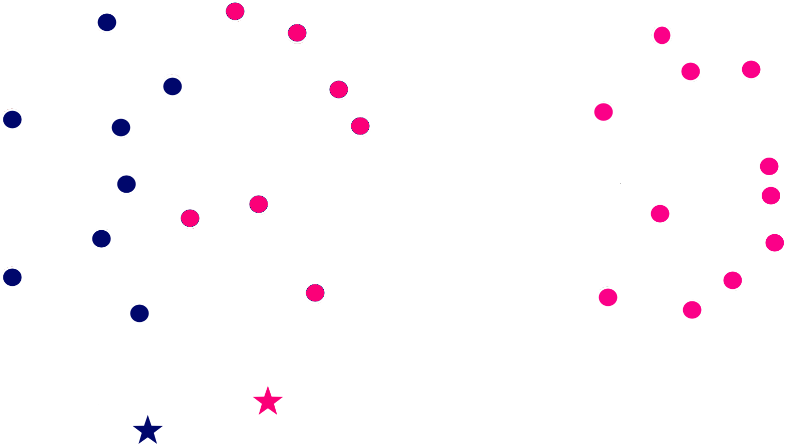
K-Means : example



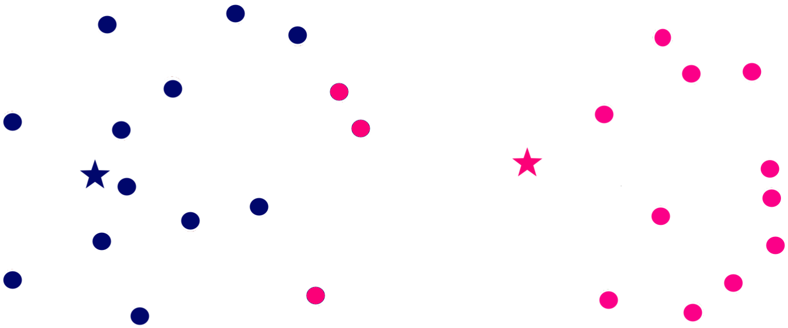
K-Means : example



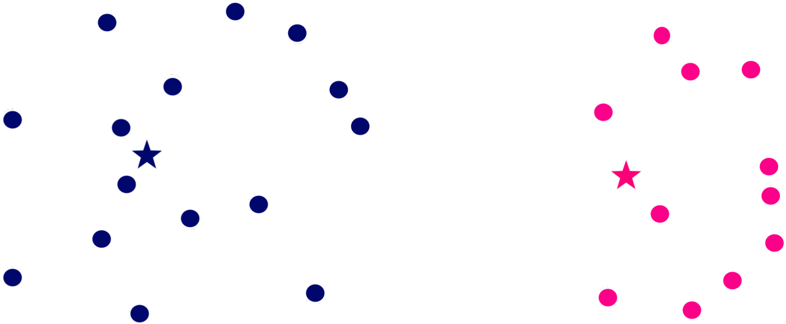
K-Means : example



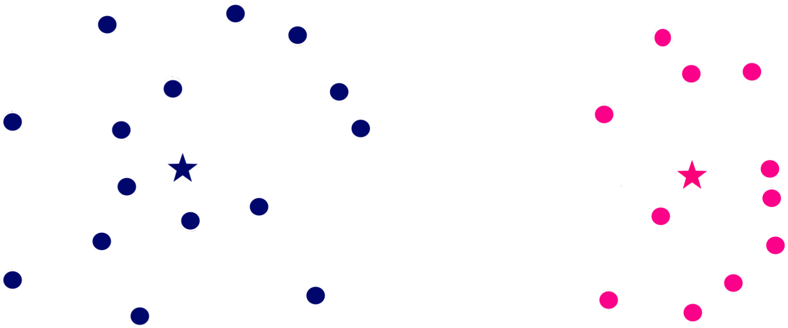
K-Means : example



K-Means : example



K-Means : example



Clustering : K-Means

- **Goal :** Assign each example $(x_1, ..., x_n)$ to one of the K clusters $\{C_1, ..., C_K\}$.

Clustering : K-Means

- **Goal :** Assign each example $(x_1, ..., x_n)$ to one of the K clusters $\{C_1, ..., C_K\}$.
- Centroid μ_j is the mean of all examples in the j^{th} cluster.

Clustering : K-Means

- **Goal :** Assign each example $(x_1, ..., x_n)$ to one of the K clusters $\{C_1, ..., C_K\}$.
- Centroid μ_j is the mean of all examples in the j^{th} cluster.
- **Minimize :**

$$J = \sum_{j=1}^{K} \sum_{x_i \in C_j} \| x_i - \mu_j \|^2$$

Clustering : K-Means

Data: without labels
Result: set of clusters $\{C_1, ..., C_K\}$ and data assignement to them
Initialize randomly $\mu_1, ..., \mu_K$;
while *convergence* not reached* **do**
Assign each point x_i to the cluster with the closest μ_j;
Calculate the new centers of each cluster as follows :

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j}$$

end
convergence* = no change in the clusters OR maximum number of iterations reached;

Algorithm 1: K-means

K-Means : pros and cons

- Easy to implement

  BUT...

  - Need to know K
  - Suffer from the curse of dimensionality
  - Lack of theoretical foundation

K-Means : question

How to evaluate your model ?

- Not trivial (as compared to counting the number of errors in classification).
- **Internal evaluation :** using same data. high intra-cluster similarity and low inter-cluster similarity.
- **External evaluation :** use of ground truth of external data.

Outline

Hierarchical clustering

- Hierarchical clustering is a widely used data analysis tool.

Hierarchical clustering

- Hierarchical clustering is a widely used data analysis tool.
- The idea is to build a binary tree of the data that successively merges similar groups of points

Hierarchical clustering

- Hierarchical clustering is a widely used data analysis tool.
- The idea is to build a binary tree of the data that successively merges similar groups of points
- Visualizing this tree provides a useful summary of the data

Hierarchical clusering vs K-means

- Recall that k-means requires
  - A number of clusters K
  - An initial assignment of data to clusters
  - A distance measure between data d(xn, xm)
- Hierarchical clustering only requires a measure of similarity between groups of data points.

Clustering : hierarchical clustering

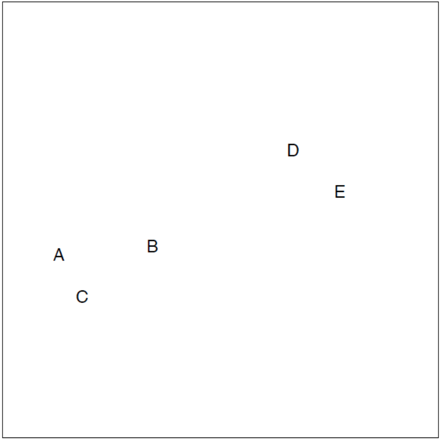
Data: without labels
Result: data tree (taxonomy)

Place each data point into its own singleton group;

while *all the data are not merged into a single cluster* **do**
    merge the two closest groups;
end

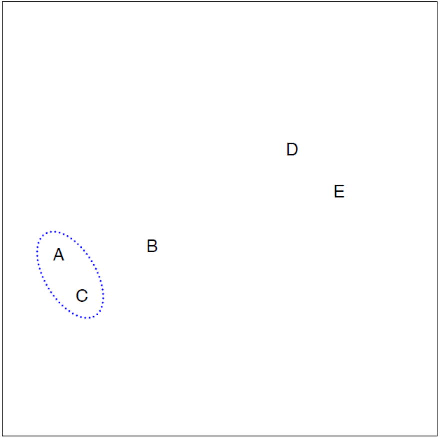
Algorithm 2: Agglomerative clustering

Hierarchical clustering



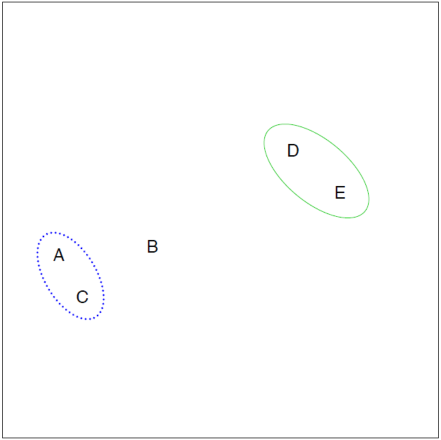
Each point starts as its own cluster.

Hierarchical clustering



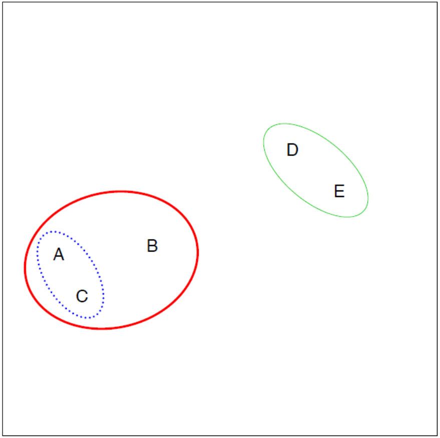
We merge the two clusters (points) that are closet to each other.

Hierarchical clustering



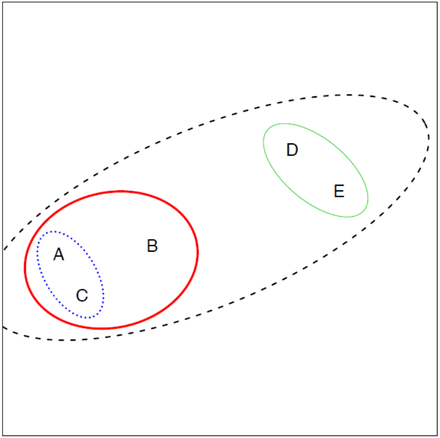
Then we merge the next two closest clusters.

Hierarchical clustering



Then the next two closest clusters. . .

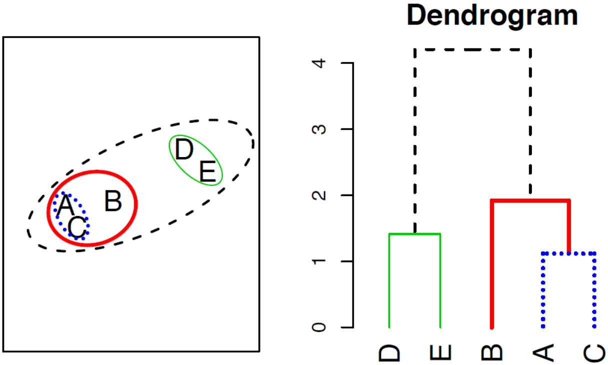
Hierarchical clustering



Until at last all of the points are all in a single cluster.

Hierarchical clustering

To visualise the results, we can look at the resulting **dendrogram**.



y-axis on dendrogram is (proportional to) the distance between the clusters that got merged at that step.

Outline

Classification

Given : Training data :
$(x_1, y_1), \ldots, (x_n, y_n)$, with $x_i \in \Omega^1 \times \cdots \times \Omega^d$ and $y_i \in E$ is discrete (categorical/qualitative)..

Example $E = \{-1, +1\}$, $E = \{0, 1\}$.

Task : Learn a classification function :

$$f : \Omega^1 \times \cdots \times \Omega^d \longrightarrow E$$

Linear Classification : A classification model is said to be linear if it is represented by a linear function f (linear hyperplane)

Classification : example

Credit default/not default ! Which customers will default on their credit card debt ?

Balance	Default
300	no
2000	no
500	yes
⋮	⋮

Outline

Classification

- We can't predict Credit Card Default with any certainty. Suppose we want to predict how likely is a customer to default. We must compute a probability between 0 and 1 that a customer will default.

- It makes sense and would be suitable and practical.

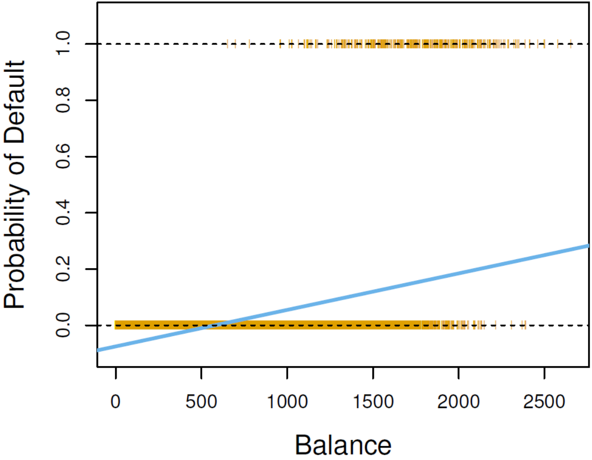
- In this case, the output is real (regression) but is bounded (classification).

$$P(y|x) = P(\text{default} = \text{yes} \mid \text{balance})$$

Classification

- Can we use linear regression ?
- Yes. However...
  - Works only for Binary classification (2 classes). Won't work for Multiclass classification e.g.
    $E = \{\text{green, blue, brown}\}$
    $E = \{\text{stroke, heart attack, drug overdose}\}$
  - If we use linear regression, some of the predictions will be outside of [0,1].
  - Model can be poor. Example.

Classification : example



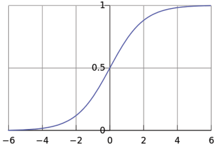
Classification

$$y = f(x) = \beta_0 + \beta_1 x$$

$$\text{Default} = \beta_0 + \beta_1 \times \text{Balance}$$

We want $0 \leq f(x) \leq 1$; $f(x) = P(y = 1|x)$

We use the sigmoid function :



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

$$g(z) \xrightarrow[z \to -\infty]{} 0 \quad \text{and} \quad g(z) \xrightarrow[z \to +\infty]{} 1$$

Logistic Regression

$$g(\beta_0 + \beta_1 x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

$$\boxed{\text{New } f(x) = g(\beta_0 + \beta_1 x)}$$

In general :

$$f(x) = g\left(\sum_{j=1}^{d} \beta_j x_j\right)$$

In other words, cast the output to bring the linear function quantity between 0 and 1.

Note : One can use other S-shaped functions.

Outline

K-nearest neighbours

- Not every ML method builds a model !
- Main idea of KNN : Uses the similarity between examples.
- Assumption : Two similar examples should have same labels.
- Numerical features : $\Omega^1 \times \cdots \times \Omega^d \subset \mathbb{R}^d$.
- Assumes all examples (instances) are points in the d dimensional space $\Omega^1 \times \cdots \times \Omega^d$.
- KNN uses the standard Euclidian distance (*usually*) to define nearest neighbours.

  Given two examples x_{i_1} and x_{i_2}, $d(x_{i_1}, x_{i_2}) = \sqrt{\sum_{j=1}^{d}(x_{i_1}^j - x_{i_2}^j)^2}$

K-nearest neighbours

Training algorithm :

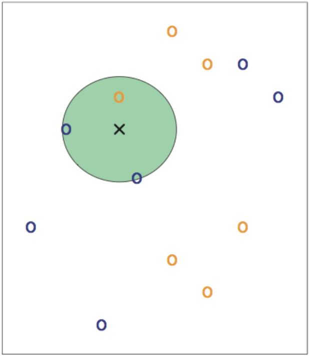
Add each training example (x, y) to the dataset $\mathcal{D}$.
$x \in \Omega^1 \times \cdots \times \Omega^d y \in \{-1, +1\}$.

Classification algorithm :

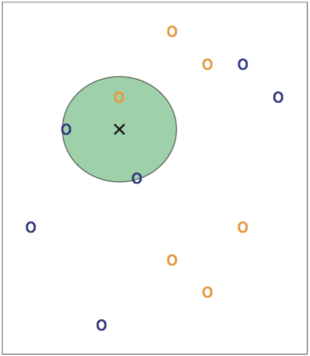
Given an example x_q to be classified.
Suppose $N_K(x_q)$ is the set of the K-nearest neighbours of x_q,

$$\hat{y}_q = sign\left(\sum_{x_i \in N_K(x_q)} y_i\right)$$

K-nearest neighbours

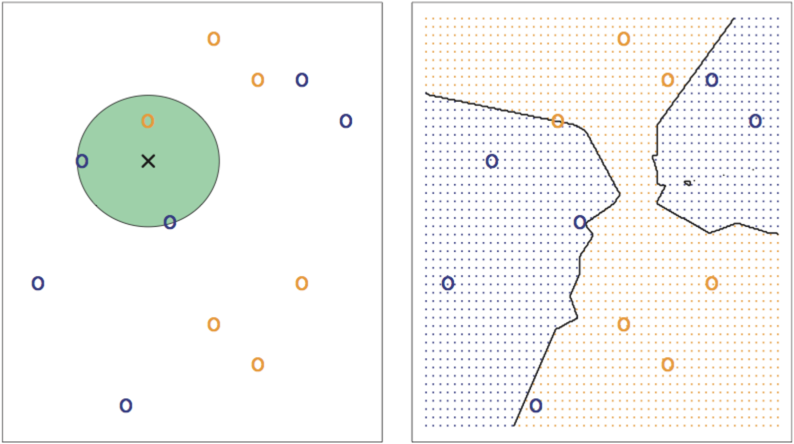


K-nearest neighbours



Question : Draw an approximate decision boundary for $K = 3$?

K-nearest neighbours



K-nearest neighbours

Question : What are the pros and cons of KNN ?

Pros :

+ Simple to implement.
+ Works well in practice.
+ Does not require to build a model, make assumptions, tune parameters.
+ Can be extended easily with news examples.

Cons :

- Requires large space to store the entire training dataset.
- Slow ! Given n examples and d features. The method takes $\mathcal{O}(n \times d)$ to run.
- Suffers from the curse of dimensionality.

Outline

Perceptron

- Belongs to Neural Networks class of algorithms (algorithms that try to mimic how the brain functions).

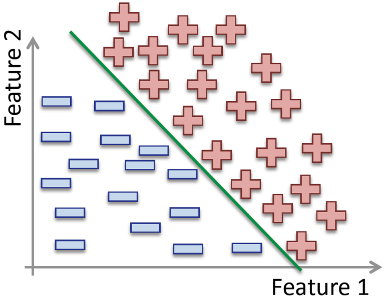
- The first algorithm used was the Perceptron (Rosenblatt 1959).

- Worked extremely well to recognize :
  - handwritten characters (LeCun et al. 1989)
  - spoken words (Lang et al. 1990)
  - faces (Cottrel 1990)

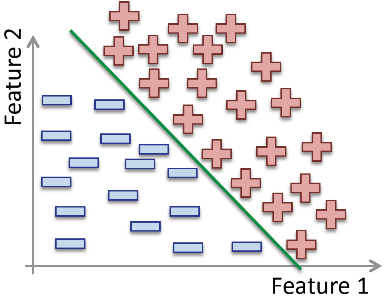
- NN were popular in the 90's but then lost some of its popularity.

- Now NN back with deep learning.

Perceptron

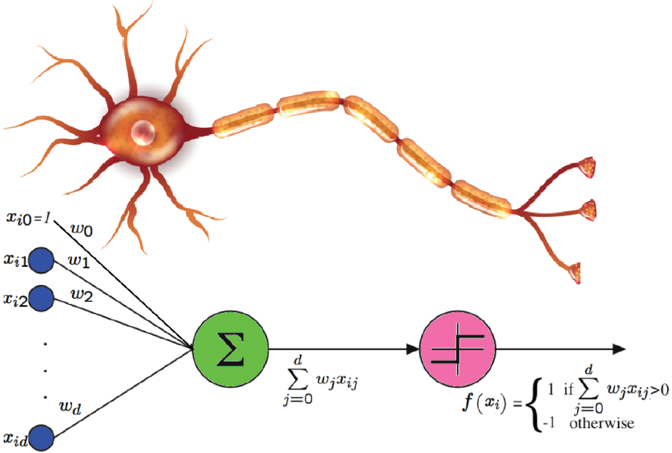


Perceptron



- Linear classification method.
- Simplest classification method.
- Simplest neural network.
- For perfectly separated data.

Perceptron



Given n examples and d features, $f(x_i) = sign\left(\sum_{j=0}^{d} w_j x_j\right)$

Perceptron

- Works perfectly if data is linearly separable. If not, it will not converge.
- Idea : Start with a random hyperplane and adjust it using your training data.
- Iterative method.

Perceptron

Data: A set of examples, $(x_1, y_1), ..., (x_n, y_n)$

Result: A perceptron defined by $(w_0, w_1, ..., w_d)$

Initialize the weights w_j to 0 $\forall j \in \{1, ..., d\}$;

```
while convergence not reached do
    update all w_j :
    for j ∈ {1, ..., d} do
        for i ∈ {1, ..., n} do
            if y_i f(x_i) ≤ 0 #i.e. error then
                w_j := w_j + y_i x_i
            end
        end
    end
end
```

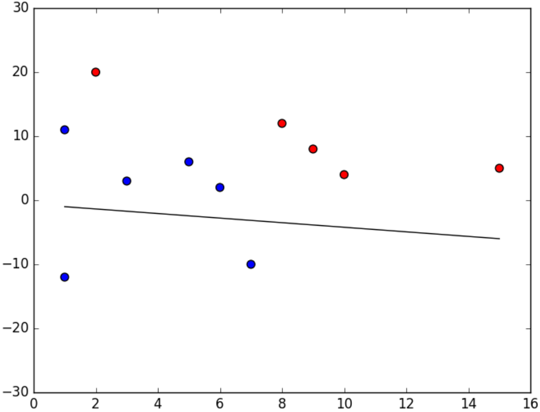
Algorithm 3: Perceptron

Perceptron

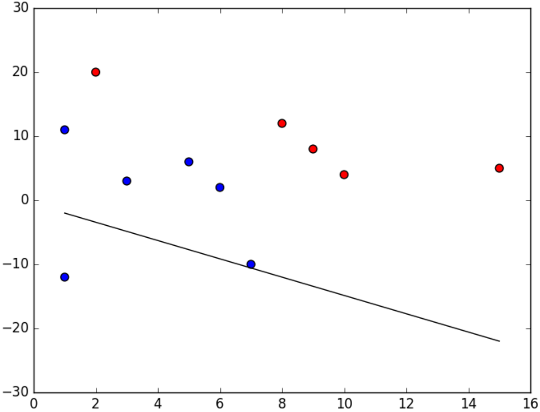
Some observations :

- The weights $w_1, \ldots, w_d$ determine the slope of the decision boundary.
- w_0 determines the offset of the decision boundary (can be noted b).
- weights adjustment corresponds to :
  - Mistake on positive : add x to weight vector.
  - Mistake on negative : substract x from weight vector.
  - Some other variants of the algorithm add or subtract 1.
- Convergence happens when the weights do not change anymore (difference between the last two weight vectors is $< \epsilon$).

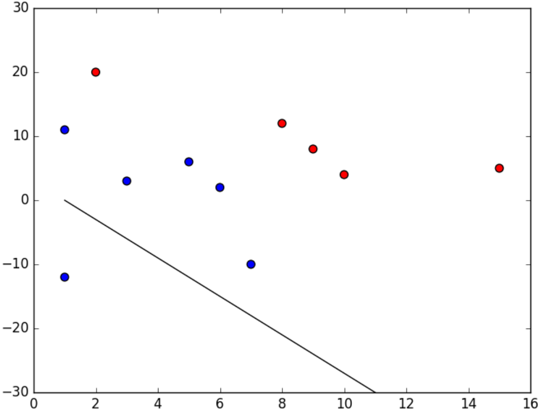
Perceptron



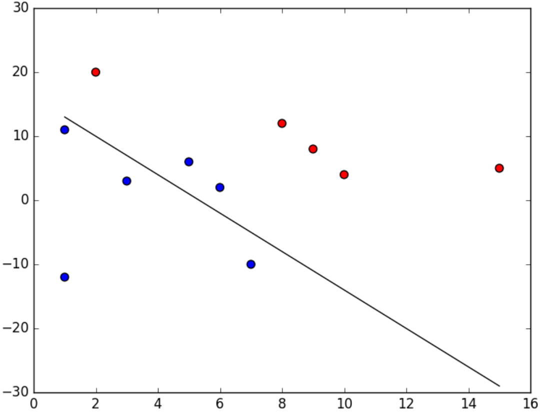
Perceptron



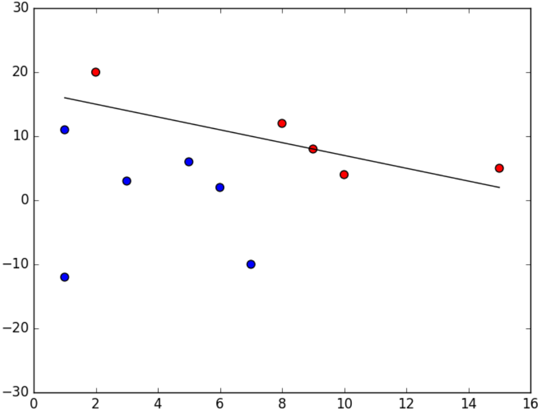
Perceptron



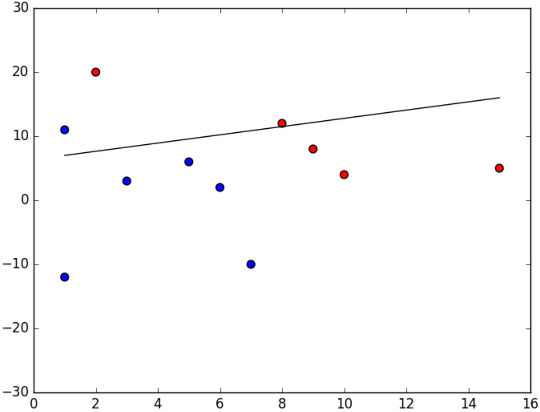
Perceptron



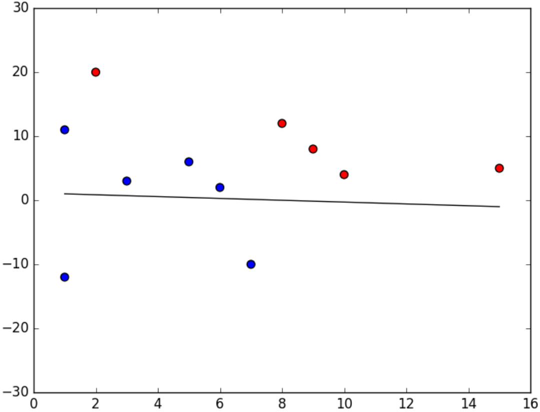
Perceptron



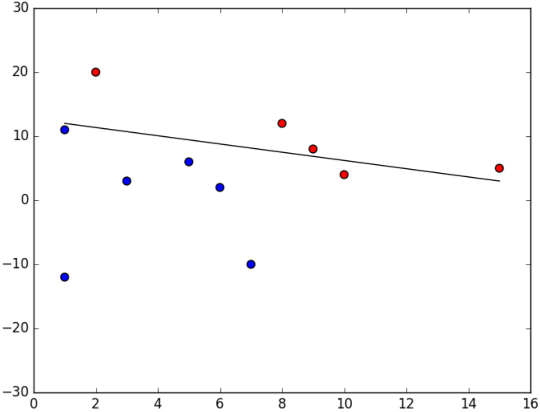
Perceptron



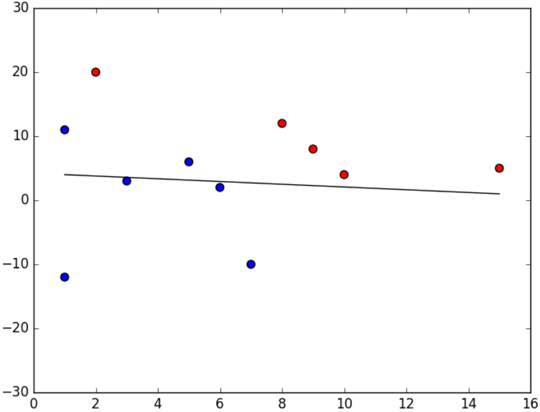
Perceptron



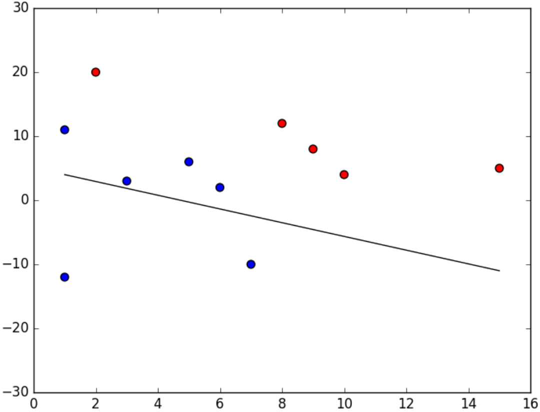
Perceptron



Perceptron

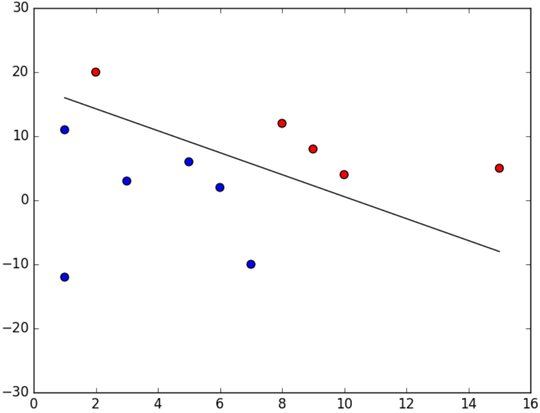


Perceptron



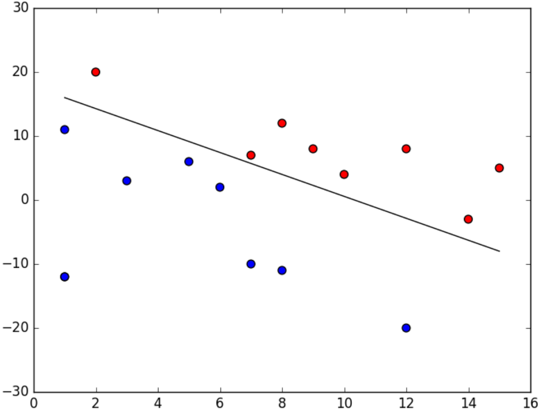
Perceptron

Finally converged !



Perceptron

With some test data :



Perceptron expressiveness

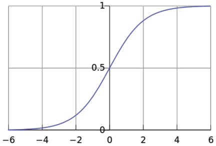
- Consider the perceptron with the *activation* function.
- Idea : Iterative method that starts with a random hyperplane and adjust it using your training data.
- It can represent Boolean functions such as AND, OR, NOT but not the XOR function.
- It produces a linear separator in the input space.

From perceptron to MLP

- The perceptron works perfectly if data are linearly separable. If not, it will not converge.
- Neural networks use the ability of the perceptrons to represent elementary functions and combine them in a network of layers of elementary questions.
- However, a cascade of linear functions is still linear,
- and we want networks that represent highly non-linear functions.

From perceptron to MLP

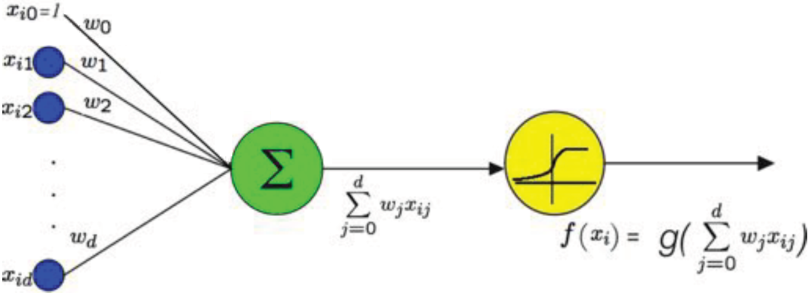
- Also, perceptron used an **activation function**, which is undifferentiable and not suitable for gradient descent (non-derivable) in case data is not linearly separable.
- We want a function whose input is a linear function of the data and whhose output is differentiable according to the data.
- One possibility is to use the sigmoid function :



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

$$g(z) \xrightarrow[z \to -\infty]{} 0 \quad \text{and} \quad g(z) \xrightarrow[z \to +\infty]{} 1$$

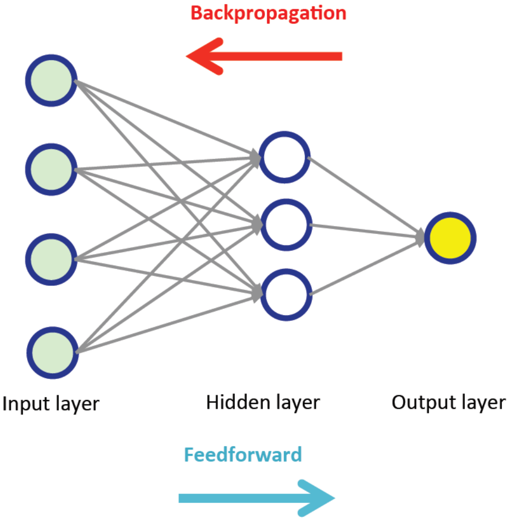
From perceptron to MLP



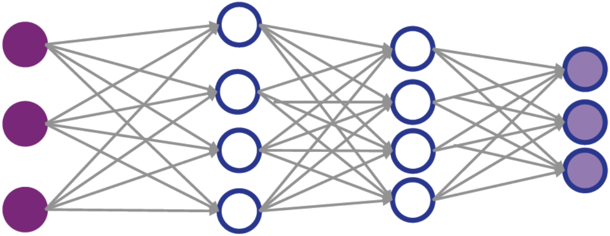
Given n examples and d features, for an example x_i (the i^{th} line in the matrix of examples) :

$$f(x_i) = \frac{1}{1 + exp\left(-\sum_{j=0}^{d} w_j x_{ij}\right)}$$

Feedforward-Backpropagation



Multi class case etc.



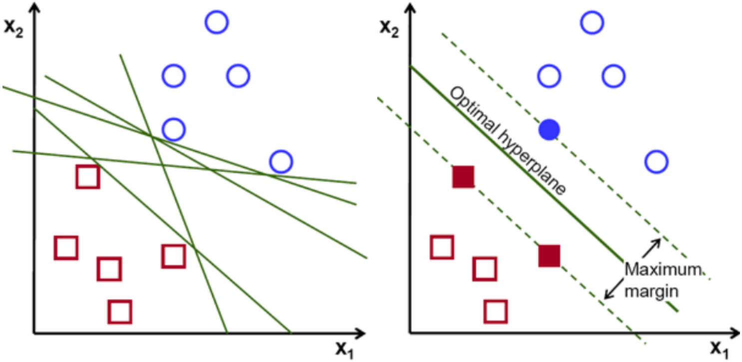
- Nowadays, networks with more than two layers, a.k.a. deep networks, have proven to be very effective in many domains.
- Examples of deep networks : restricted Boltzman machines, convolutional NN, auto encoders, etc.

Outline

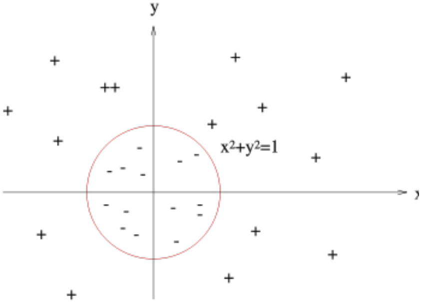
Support Vector Machine

- generalisation of linear classifiers
- end of 1990's : Vladimir Vapnik
- 2 key ideas :
  - maximal margin
  - kernels
- very popular

Margins



Kernel trick

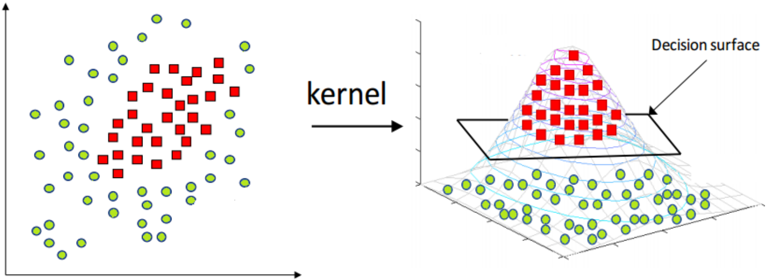


- research of the optimal hyperplane :

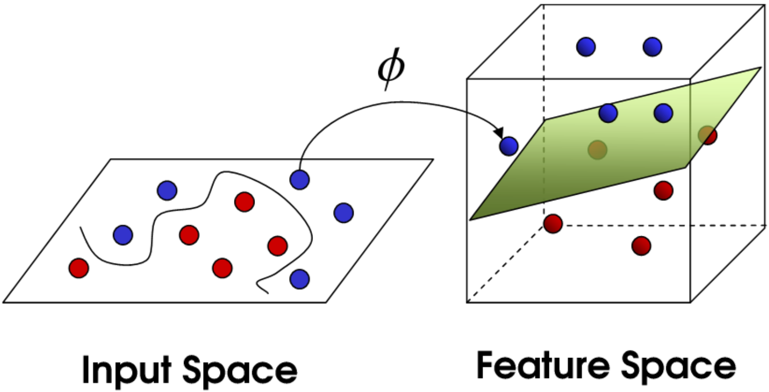
$$h(x) = w^T x + w_0 \rightarrow h(x) = w^T \phi(x) + w_0$$

- reduced → **high** dimension space → computation costs
- **kernels** : $K(x_i, x_j) = \phi(x_i)^T . \phi(x_j)$

Kernel trick



Kernel trick



Sources

- https://courses.edx.org/asset-v1:ColumbiaX+CSMM.101x+1T2017+type@asset+block@AI_edx_ml_5.1intro.pdf
- The elements of statistical learning. Data mining, inference, and prediction. 10th Edition 2009. T. Hastie, R. Tibshirani, J. Friedman.
- Machine Learning 1997. Tom Mitchell.
- https://supportivy.com/selection-de-fonctionnalites-et-noyaux-pier-paolo-ippolito-medium/
- https://datascience.stackexchange.com/questions/17536/kernel-trick-explanation/17537
- https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d