SOMACHINE

Machine Learning, Big Data, and Deep Learning in Astronomy

# Theoretical Foundations of ML: Classical Problems, Algorithms and Validation


**Salvador García**

Andalusian Research Institute of Data Science and Computational Intelligence (DaSCI)

Dpto. Ciencias de la Computación e I.A.

Universidad de Granada

salvagl@decsai.ugr.es

http://sci2s.ugr.es

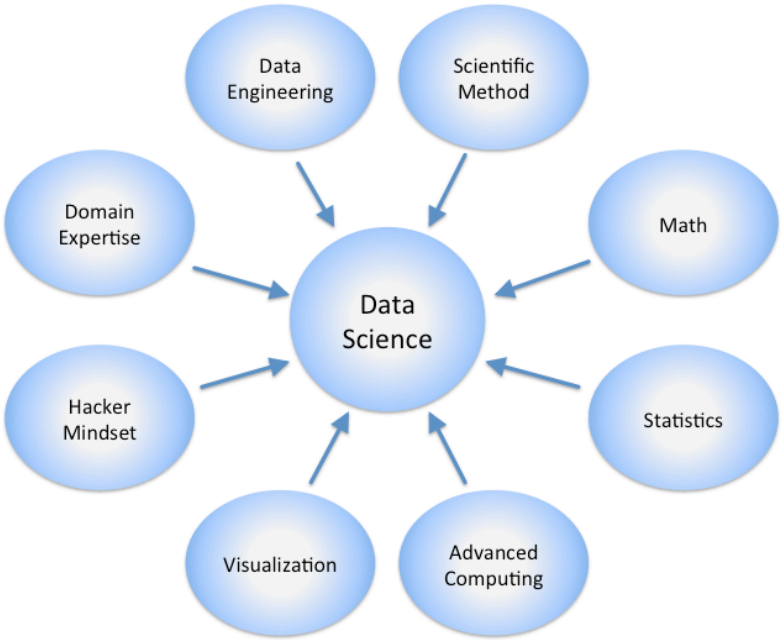
# Summary


- Basic concepts. Data Science, Data Mining, Big Data, Machine Learning (ML).
- Data Mining Process.
- ML Classical Tasks: Clasification, Regression, Clustering and Association.
- ML Algorithms: K-Nearest Neighbours, Decision Trees, Neural Networks, Support Vector Machines, K-Means and Ensembles.
- Evaluation and Validation

# Basic concepts


## Data Science

**Data Science is the field of knowledge that encompasses the skills associated with data knowledge extraction, including Big Data**



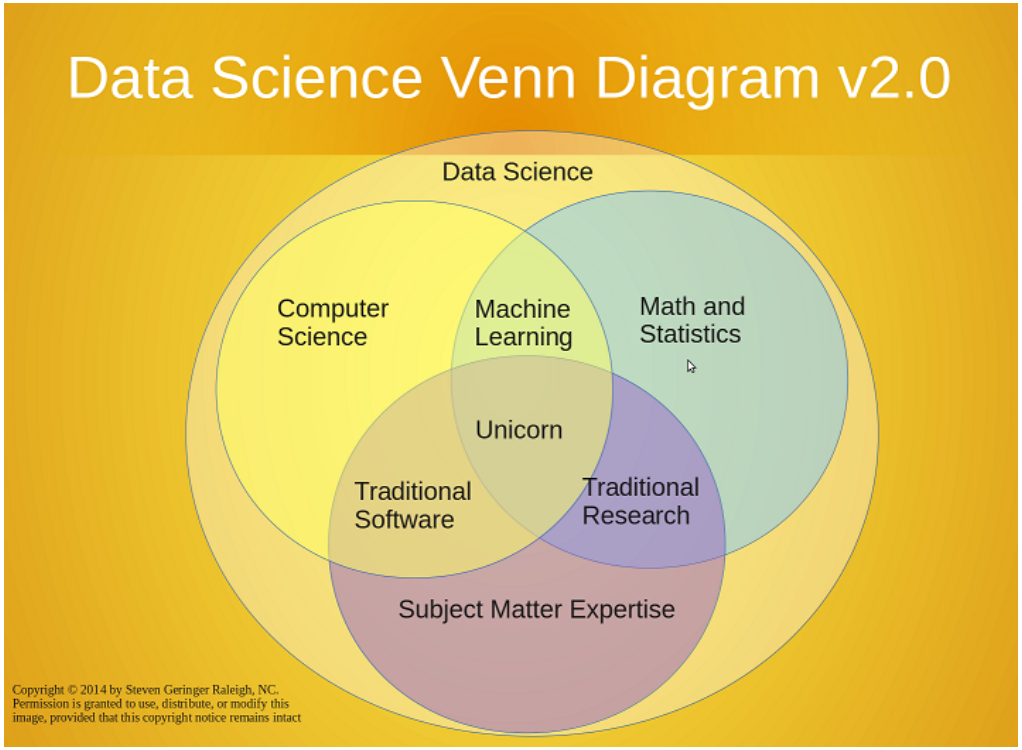

**Machine Learning** is a scientific discipline in the field of Artificial Intelligence that creates systems that learn automatically. Learning in this context means identifying complex patterns in data.

**The machine that actually learns** is an algorithm that examines the data and is able to predict future behaviour.

# Basic concepts

## What is a Data Scientist?

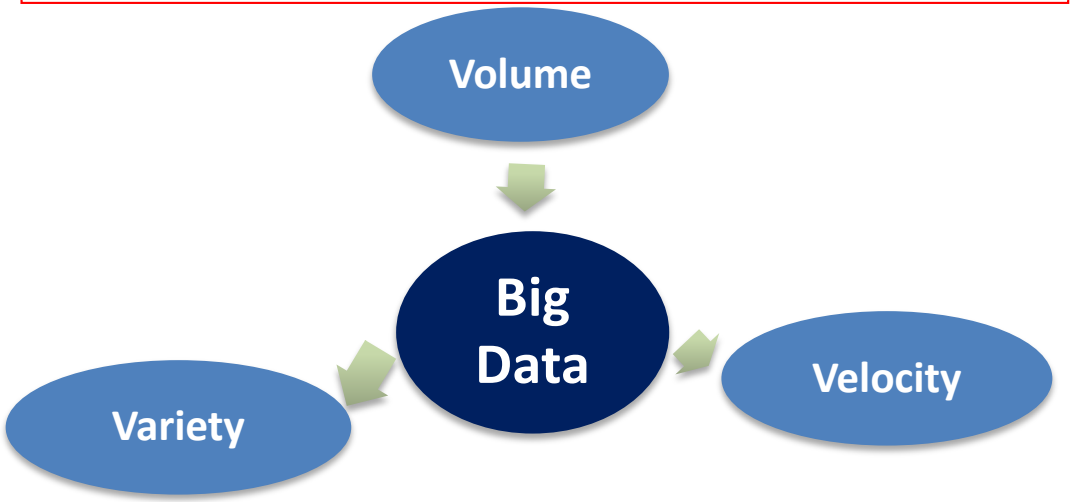
**A data scientist is a professional who must be proficient in mathematics and statistics, have a thorough knowledge of programming (and its many languages), computer science and analytics.**

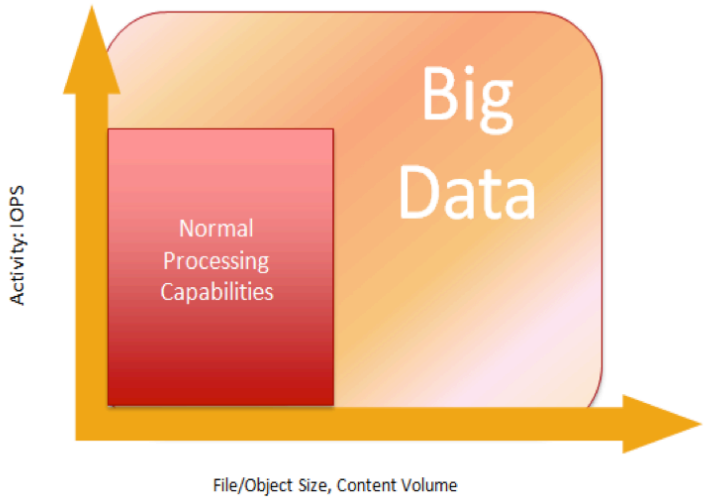


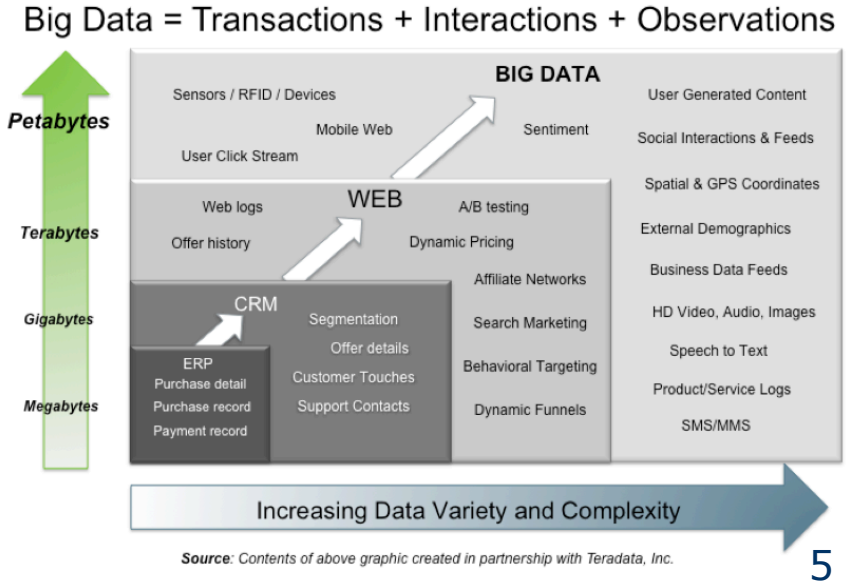
# Basic concepts

## Big Data

"***Big Data***" are data whose volume, diversity and complexity **require new architecture, techniques, algorithms and analysis** to manage and extract value and knowledge hidden in them ...




*Source: Contents of above graphic created in partnership with Teradata, Inc.*

# Basic concepts

## Data Mining vs. Machine Learning


**Data mining is a subset of business analytics and refers to exploring an existing large dataset to unearth previously unknown patterns, relationships and anomalies that are present in the data.**

**Machine learning is a subset of artificial intelligence (AI). With ML, computers analyse large data sets and then 'learn' patterns that will help it make predictions about new data sets. Apart from the initial programming and maybe some fine-tuning, the computer doesn't need human interaction to learn from the data.**

**Key Differences:**

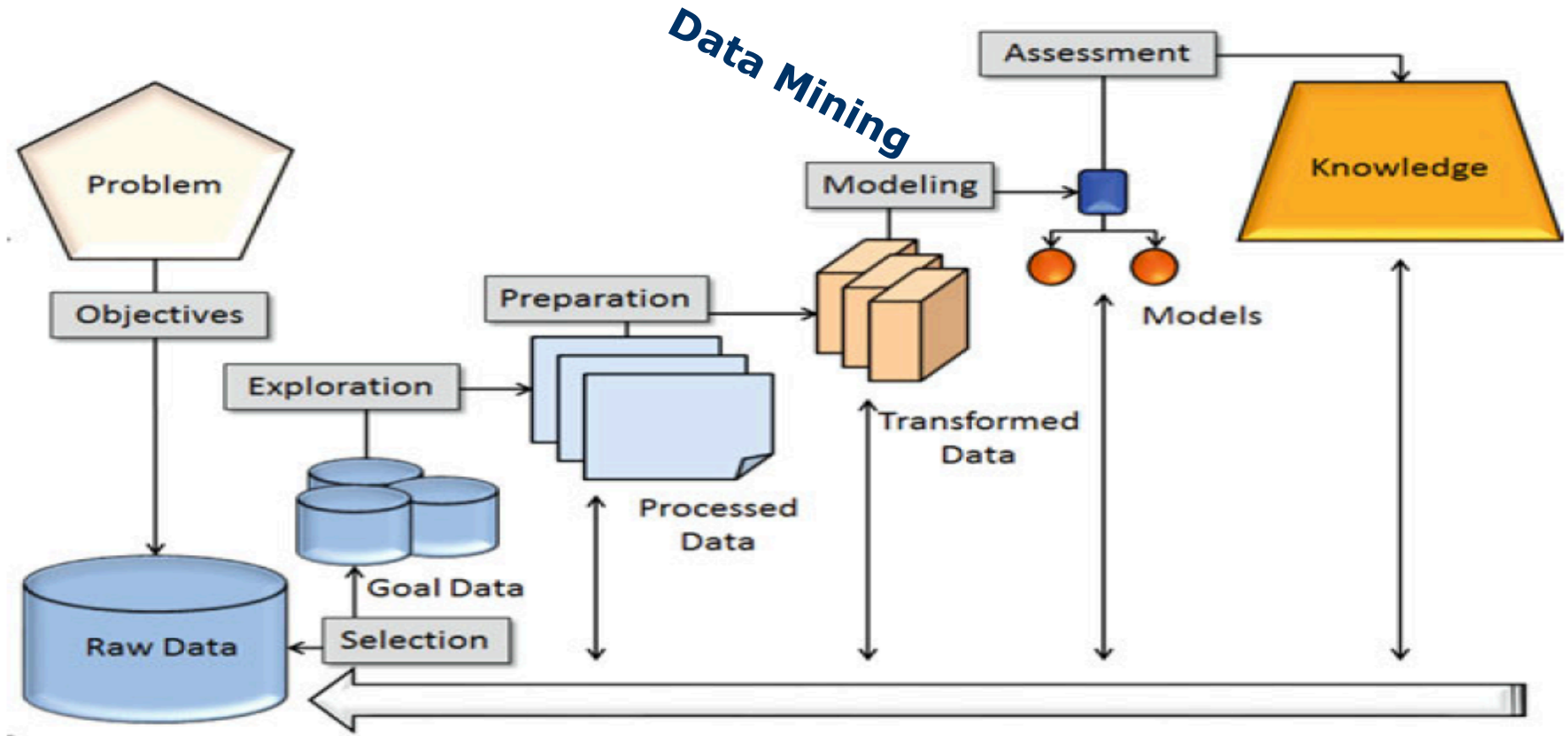
- **Patterns vs. models**
- **Unknown rules vs. predefined rules**
- **Semi-automatic vs. Automatic.**
- **Data warehouses vs. Training data.**

# Data Mining Process

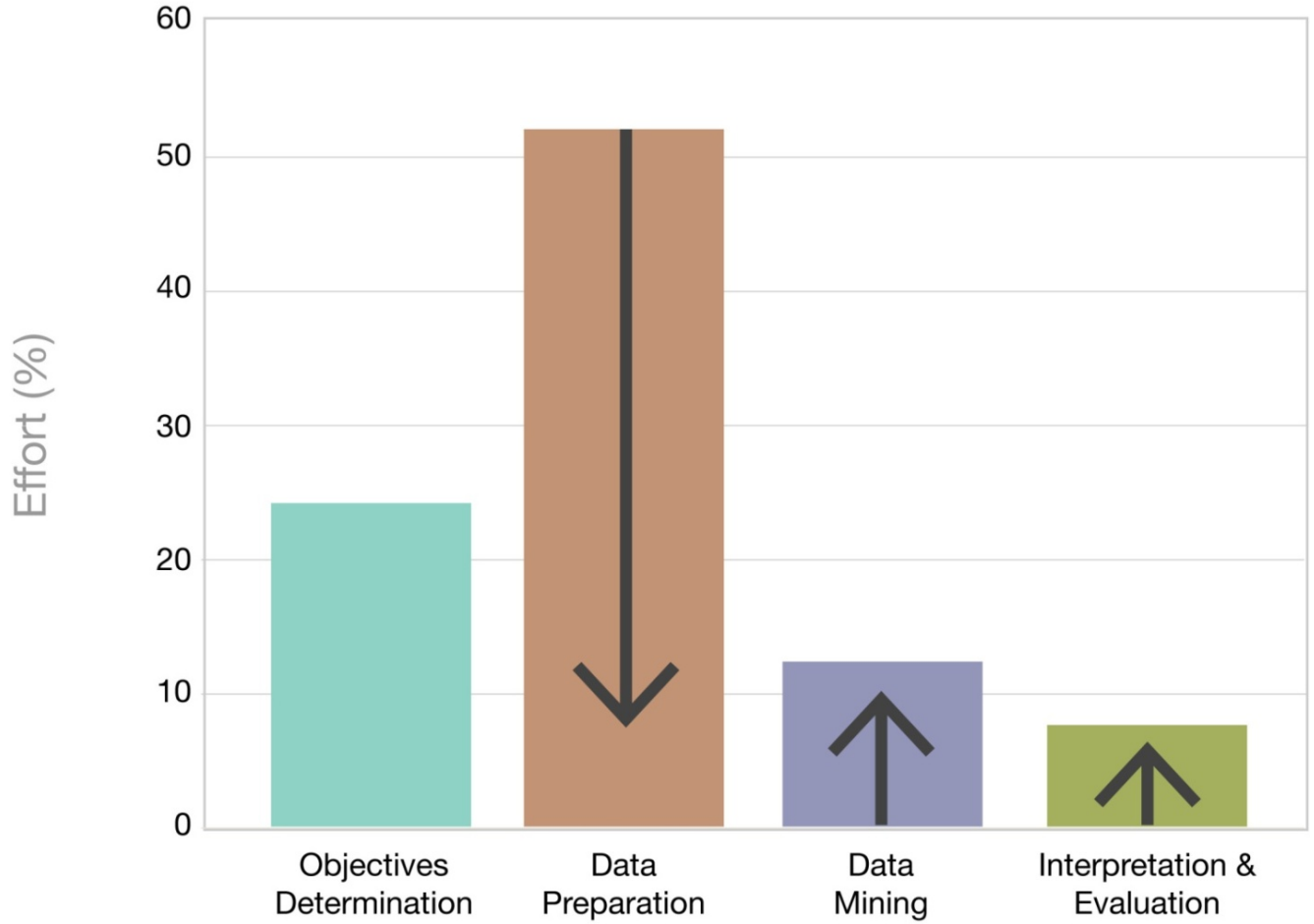
- KDD = *Knowledge Discovery from Databases*
- KDD is the complete process of knowledge extraction from databases
- The term was coined in 1989 to emphasize that knowledge is the end product of a data-driven discovery process
- Data Mining is only one step in the KDD process
- Informal association of Data Mining with KDD

# Data Mining Process

## Stages in a KDD Process



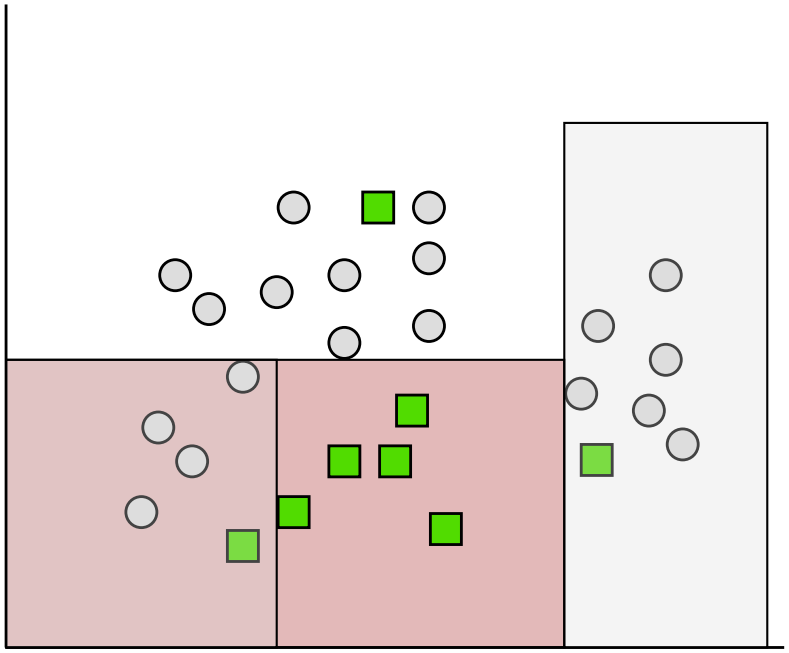
# Data Mining Process




**Estimated times in the analysis of a problem using data mining techniques**

# ML Classical Tasks

## Supervised vs. Unsupervised Learning



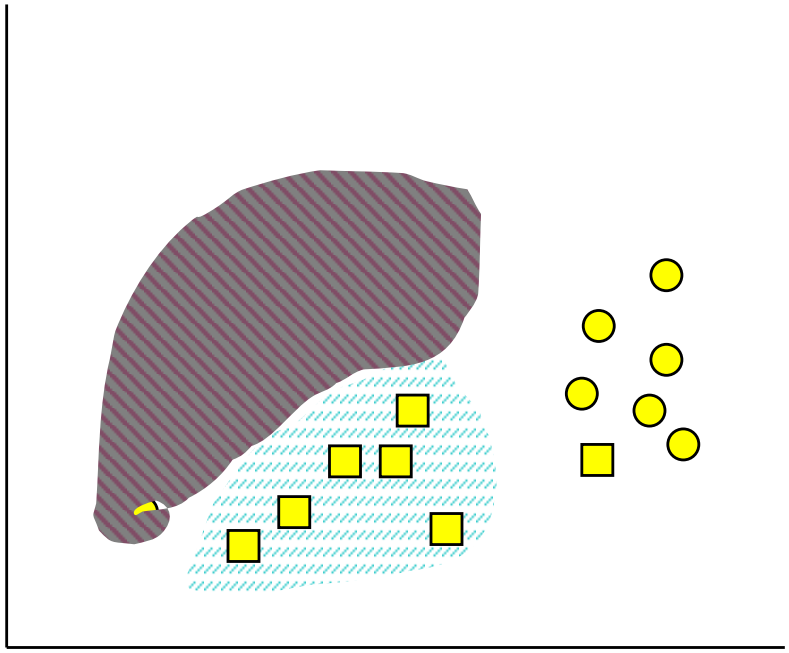
**Supervised learning:**

**Learn, from a set of pre-labelled instances a model to future predictions**

**(Example, classification: the class to which a new instance belongs)**

# ML Classical Tasks

## Supervised vs. Unsupervised Learning



**Unsupervised learning:**

**No a priori knowledge of the problem, no labeled instances, no supervision of the procedure.**

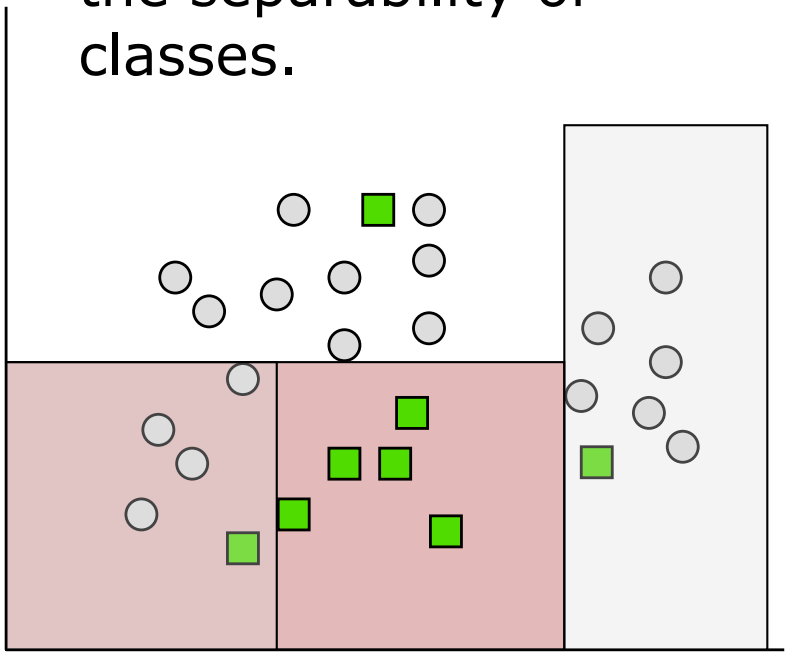
**(Example, clustering: Find a "natural" grouping of instances given a set of unlabeled instances)**

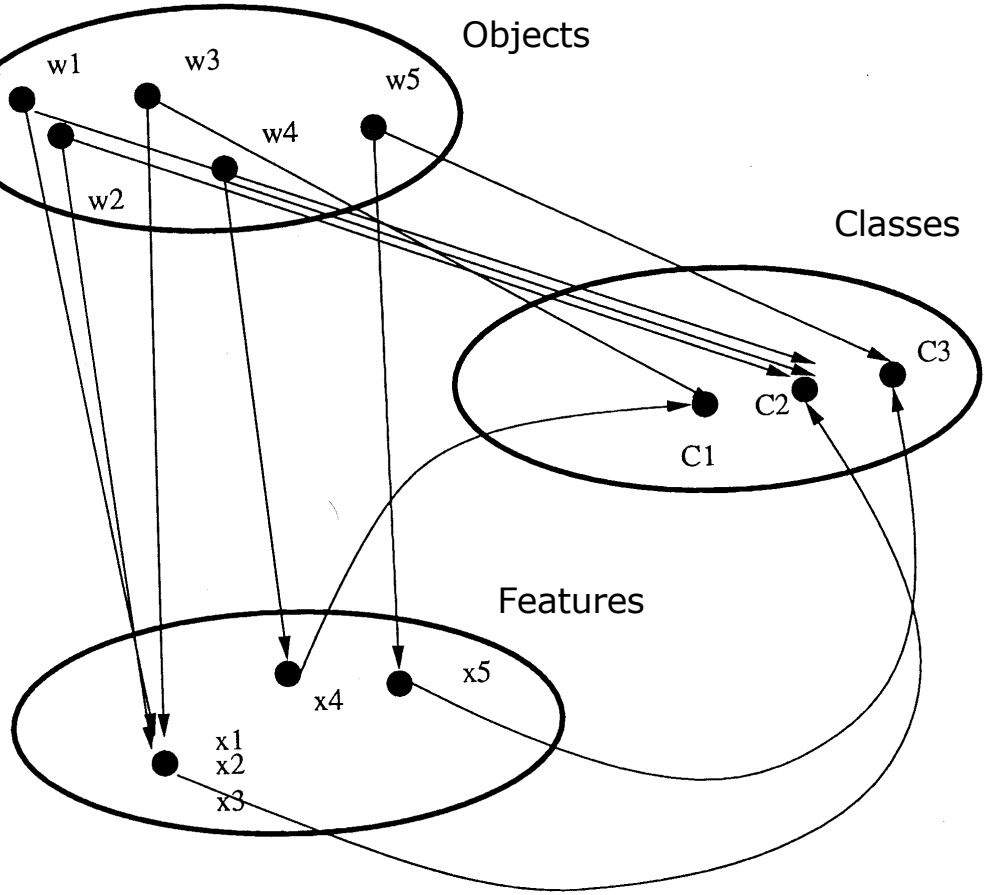
# ML Classical Tasks

## Clasification

- **Clasification**

  The fundamental problem of classification is directly related to the separability of classes.

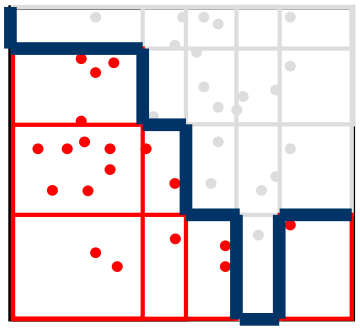


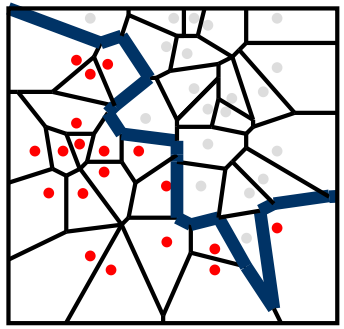


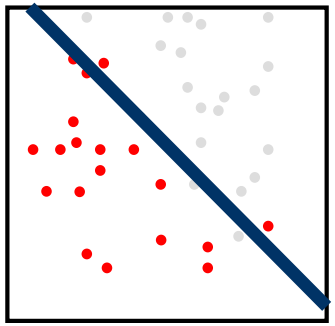
# ML Classical Tasks

## Definition of the classification problem

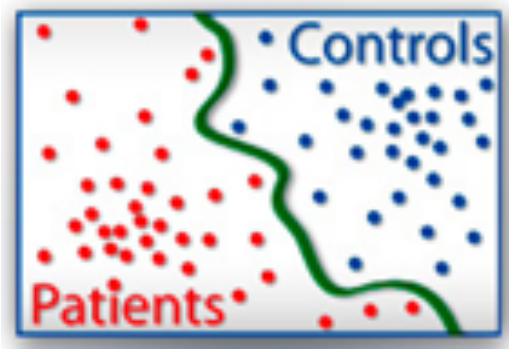
- A **classifier** can be a set of rules, a decision tree, a neural network, etc.







- Typical applications:
  - Credit approval,
    direct marketing, fraud detection,
    medical diagnosis...

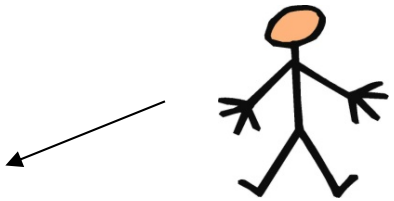


# ML Classical Tasks

## Definition of the classification problem

- An object is described by a set of characteristics (variables or attributes)

$$X \rightarrow \{X_1, X_2, \ldots, X_n\} \qquad e_i \rightarrow \{x_1, x_2, \ldots, x_n, c_k\}$$



- Age
- Astigmatism
- Tearing rate
- Myopia
- CLASIFICATION: Contact lens type

-Income
-Debts
-Assets…

-CLASIFICATION: Credit approval

- Objective of the classification task : Classify the objective within one of the class categories C={$c_1$,…,$c_k$}

$$f: X_1 \times X_2 \times \ldots \times X_n \rightarrow C$$

- The features or variables chosen depend on the classification problem

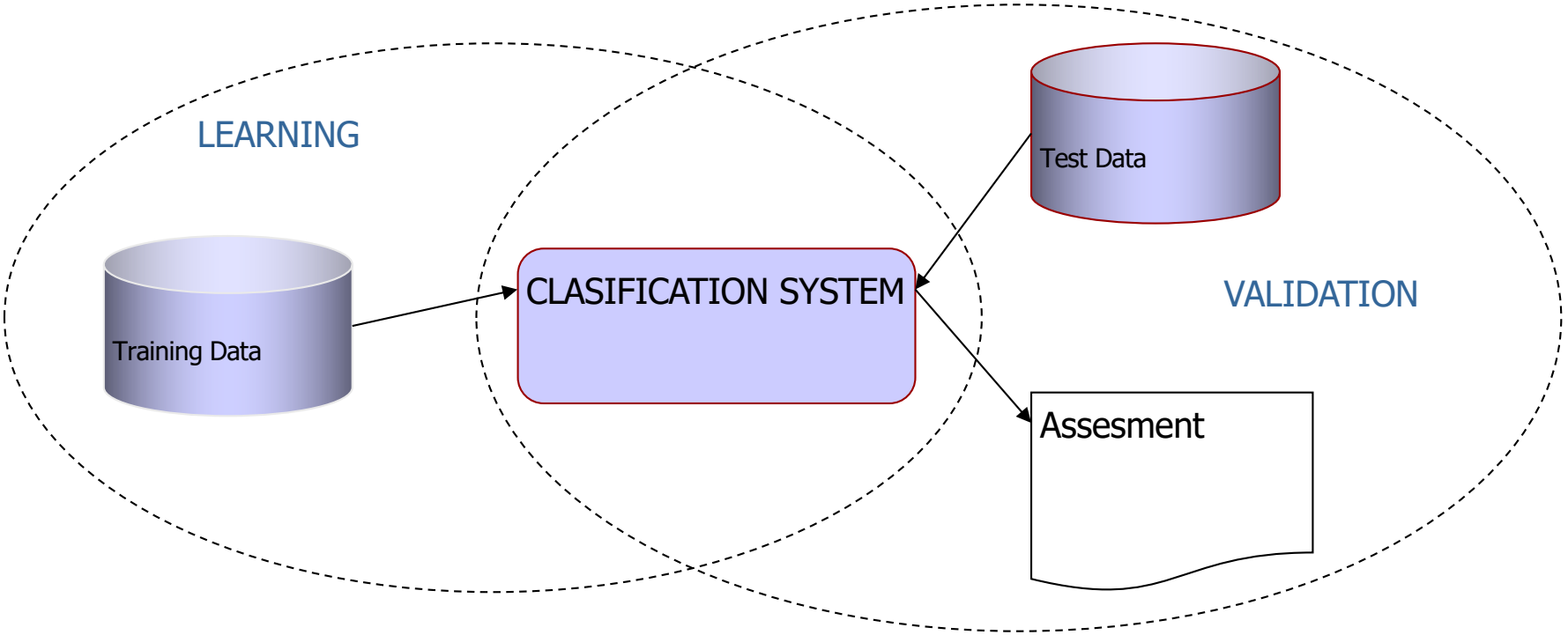
# ML Classical Tasks

## Stages in the classification process

1. Model construction describing the set of (default) classes: Learning Phase
   - Each example (instance) is known to belong to a class (class attribute label)
   - A set of examples is used for the construction of the model: *training set.*
   - The model obtained is represented as a set of classification rules, decision trees, mathematical formula, …
2. Use of the model: Validation
   - Model accuracy estimation:
   - A different set of examples is used from those used for the construction of the model : *test set*
     - If the test set was not independent from the training set, an over-adjustment process would occur (*overfitting*)
   - For each test example, the class determined by the model is compared with the real (known) class
   - The accuracy ratio is the percentage of test examples that the model correctly classifies

# ML Classical Tasks

## Stages in the classification process



# ML Classical Tasks

## Criteria for evaluating a classifier

- Accuracy(s, ε)
  - Sometimes the cost of incorrect classification must be considered
- Velocity
  - Time needed to build the model
  - Time needed to use the model
- Robustness: capacity to deal with unknown values
- Scalability: Increase in time needed (under construction and evaluation) with the size of the Data Base
- Interpretability: comprehensibility of the model obtained
- Complexity of the model : Size of the classification tree, number of rules, number of antecedents in the rules,…

# ML Classical Tasks

## Criteria for evaluating a classifier

- Confusion matrix

Given a classification problema with *m* classes, a confusión matrix is a matrix *m* x *m* in which an entry $c_{i,j}$ indicates the number of example assigned to the class $c_j$, being the correct class $c_i$

| | Prediction | | |
|---|---|---|---|
| | Short | Medium | Tall |
| Short | 0 | 4 | 0 |
| Medium | 0 | 5 | 3 |
| Tall | 0 | 1 | 2 |

*N=TP+TN+FP+FN*

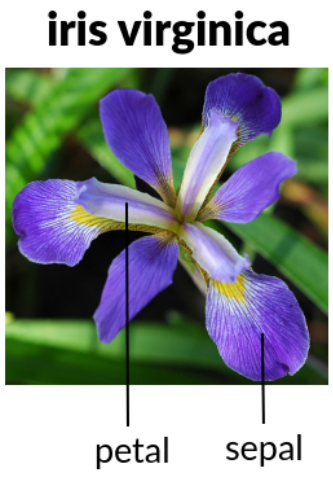
- Accuracy $s = \frac{TP + TN}{N}$
- Error Rate $\varepsilon = 1 - s$

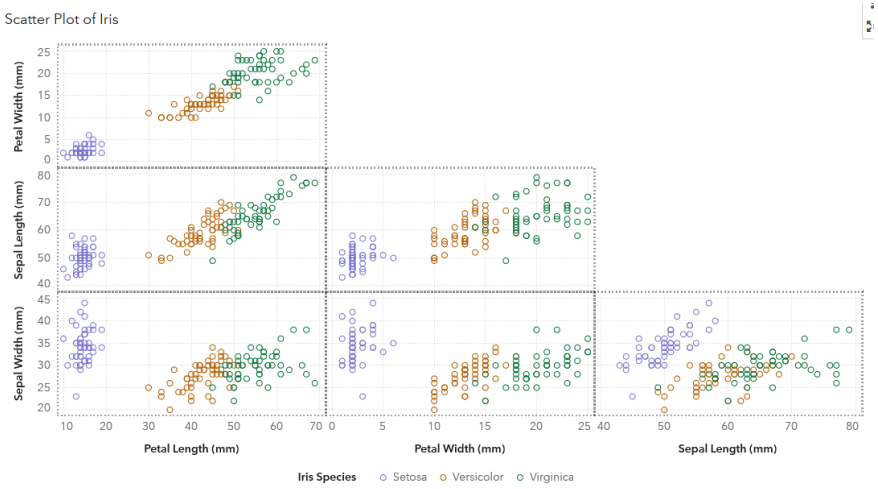
# ML Classical Tasks

## Clasification. Example

- **Example:** *Iris*
  - Three clases of iris flowers: *setosa, versicolor* y *virginica*.
  - Four attributes/features/variables: *length and width of petal and sepal* respectively.
  - 150 examples/samples/instances/patterns, 50 of each class/label.
  - Available in http://www.ics.uci.edu/~mlearn/MLRepository.html



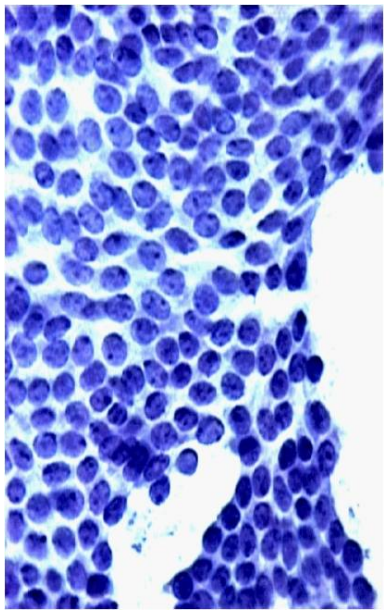


# ML Classical Tasks

## Clasification. Example

**Wisconsin Breast Cancer:**
**Predict malignant/benign**



| Attribute name | Description |
|---|---|
| RADIUS | *Mean of distances from center to points on the perimeter* |
| TEXTURE | *Standard deviation of grayscale values* |
| PERIMETER | *Perimeter of the mass* |
| AREA | *Area of the mass* |
| SMOOTHNESS | *Local variation in radius lengths* |
| COMPACTNESS | *Computed as: perimeter$^2$/area – 1.0* |
| CONCAVITY | *Severity of concave portions of the contour* |
| CONCAVE POINTS | *Number of concave portions of the contour* |
| SYMMETRY | *A measure of the nucleii's symmetry* |
| FRACTAL DIMENSION | *'Coastline approximation'* – 1.0 |
| DIAGNOSIS (Target) | *Diagnosis of cell sample: malignant or benign* |

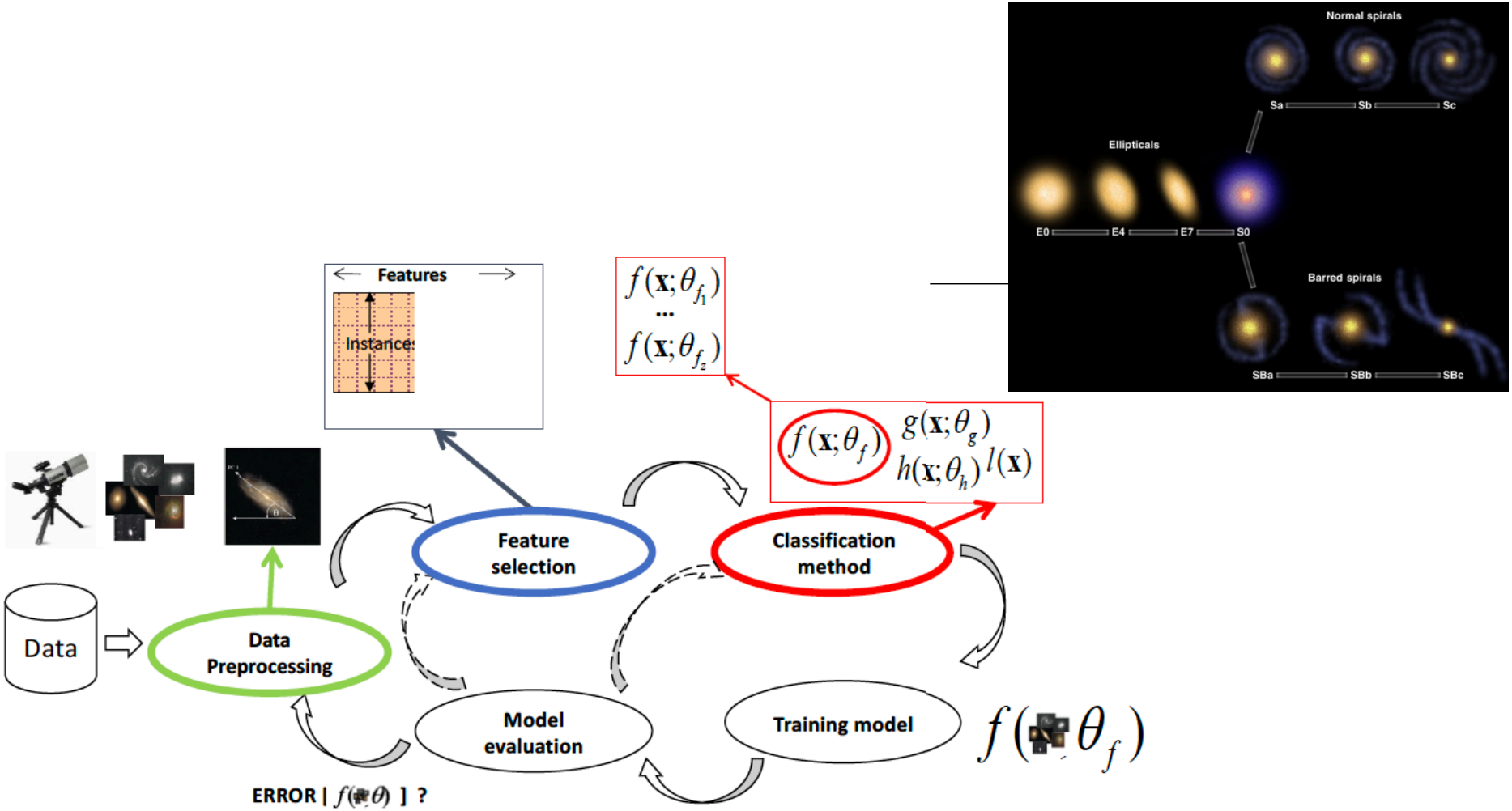
# ML Classical Tasks

## Clasification. Example

**Handwritting recognition.**
**Assign a digit from 0 to 9.**

# ML Classical Tasks

## Clasification. Typical Design Process

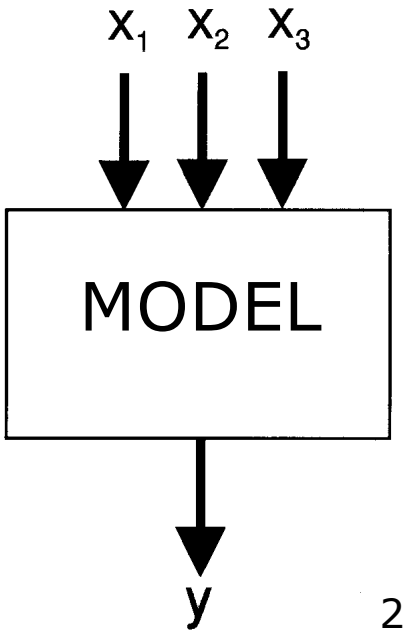


# ML Classical Tasks

## Regression

- Objective: to predict the numerical value for one variable from the values for others
- The definition of the problem is similar to that of the classification problem: we have predictor variables and a regression variable which in this case is numerical
- In regression most (and even all) predictor variables are numerical

The fundamental problem of prediction is in modelling the relationship between the state variables to obtain the value of the variable to be predicted.



# ML Classical Tasks

## Criteria for evaluating a regressor

- All the validation techniques studied in classification are valid for numerical prediction
- The difference is that now we must measure the error in another way
- We must measure the error made in approximating a set of values $\{v_1,\ldots,v_n\}$ by its esimation $\{v'_1,\ldots,v'_n\}$

Mean Squared Error (MSE)

$$MSE = \frac{\sum_{i=1}^{n}(v_i - v_i')^2}{n}$$

Standarized MSE (SMSE)

$$SMSE = \sqrt{\frac{\sum_{i=1}^{n}(v_i - v_i')^2}{n}}$$

Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{i=1}^{n}|v_i - v_i'|}{n}$$

Relative MAE (RMAE)

$$RMAE = \frac{\sum_{i=1}^{n}|v_i - v_i'|}{\sum_{i=1}^{n}|v_i - \bar{v}|}$$

Correlation coeficient

$$r_{vv'} = \frac{\sum_{i=1}^{n}(v_i - \bar{v})(v_i' - \bar{v}')}{(n-1)\sigma_v\sigma_{v'}}$$

# ML Classical Tasks

## Regression analysis

- Regression analysis is the most commonly used method to perform the numerical prediction task
- **Goal:** to estimate the output variable *(y)* as an equation operating with the remaining inputs variables $(x_1,\ldots,x_n)$
- The simplest model is the **lineal regression** which reduced as a only predictor variable has the form:

$$y = a + b \cdot x$$

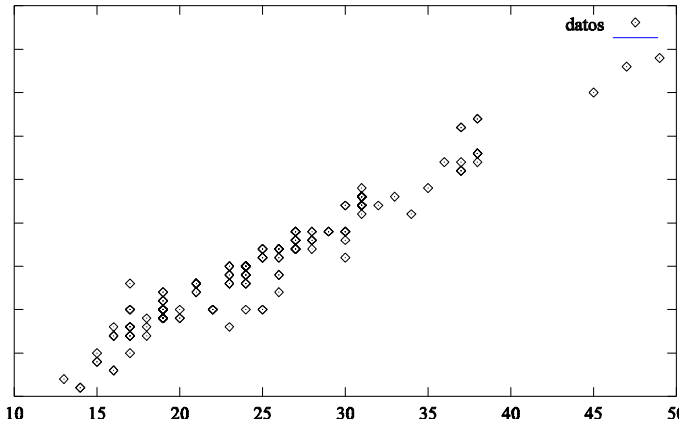
- These coefficients can easily be obtained using the method of least squares

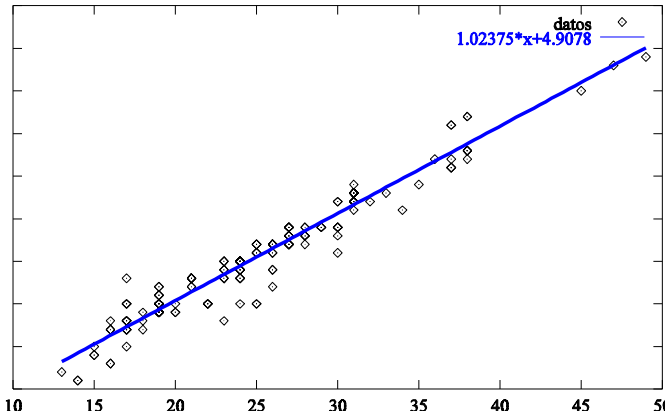
$$b = \frac{\sum_{i=1}^{s}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{s}(x_i - \bar{x})^2}$$

$$a = \bar{y} - b \cdot \bar{x}$$

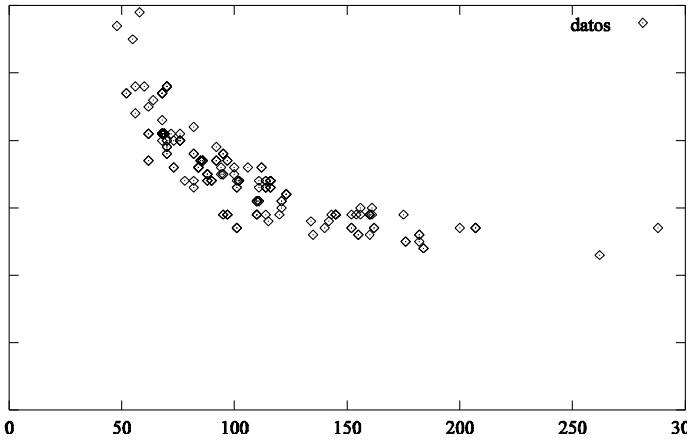
# ML Classical Tasks

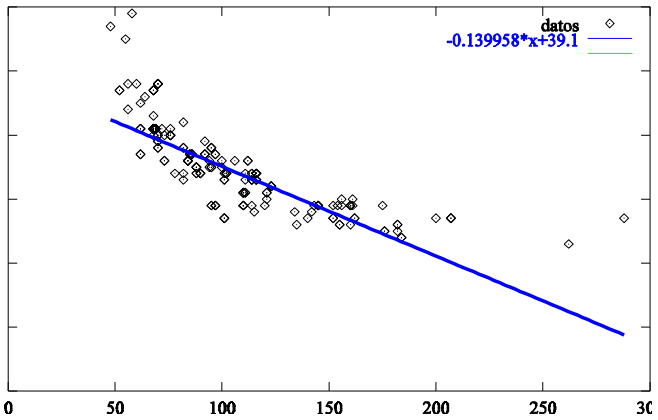
## Regression analysis




Quite appropriate



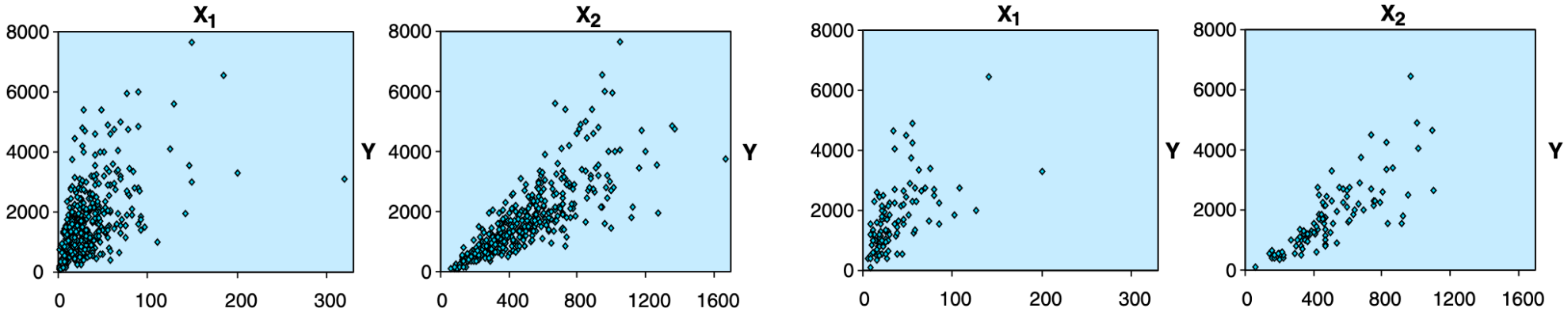




Not so appropriate

# ML Classical Tasks

## Regression. Example

- **Example:** *Electrical length data set*
  - Goal: Estimating the length of low voltage lines of a village.
  - Two attributes/features: inhabitants and distance (radius of the village).
  - 495 examples/samples/instances/patterns. Output is the length.
  - Available in http://www.keel.es/




(a) Training data

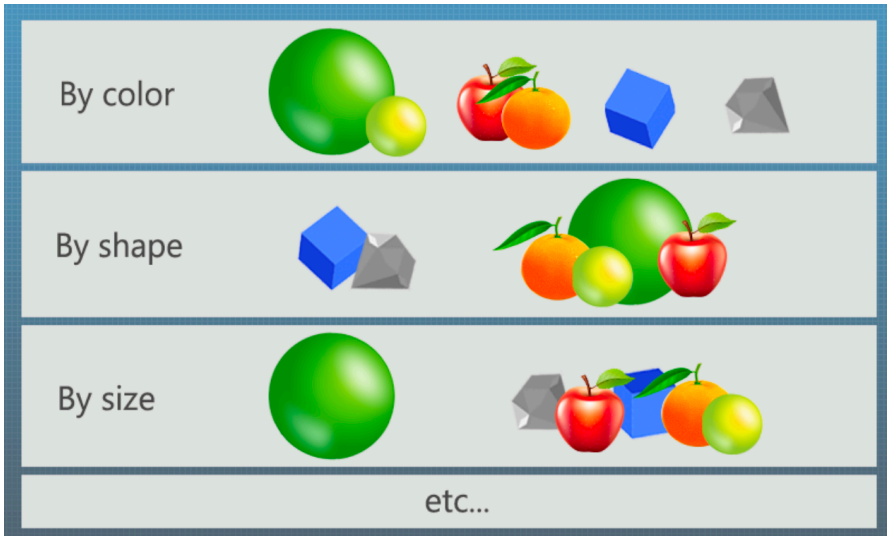
(b) Test data

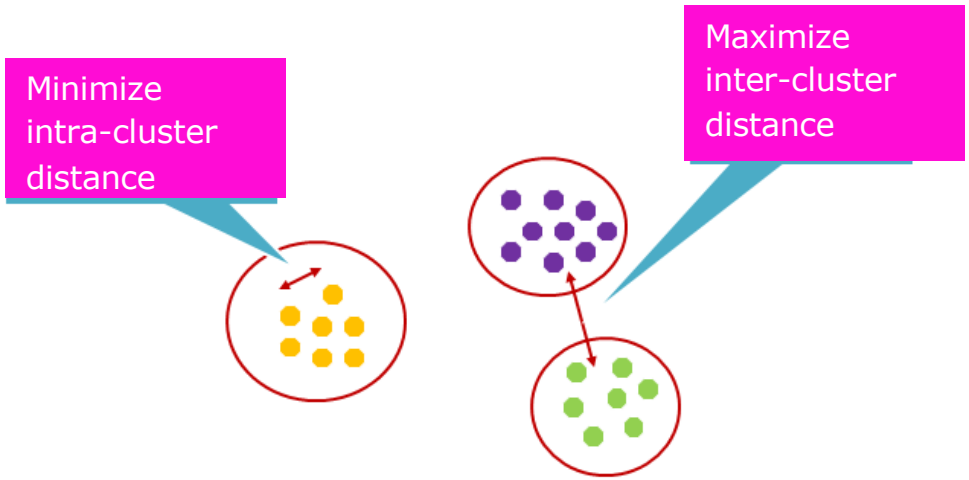
# ML Classical Tasks

## Clustering

**There are problems where we want to group the instances by creating clusters with similar characteristics**

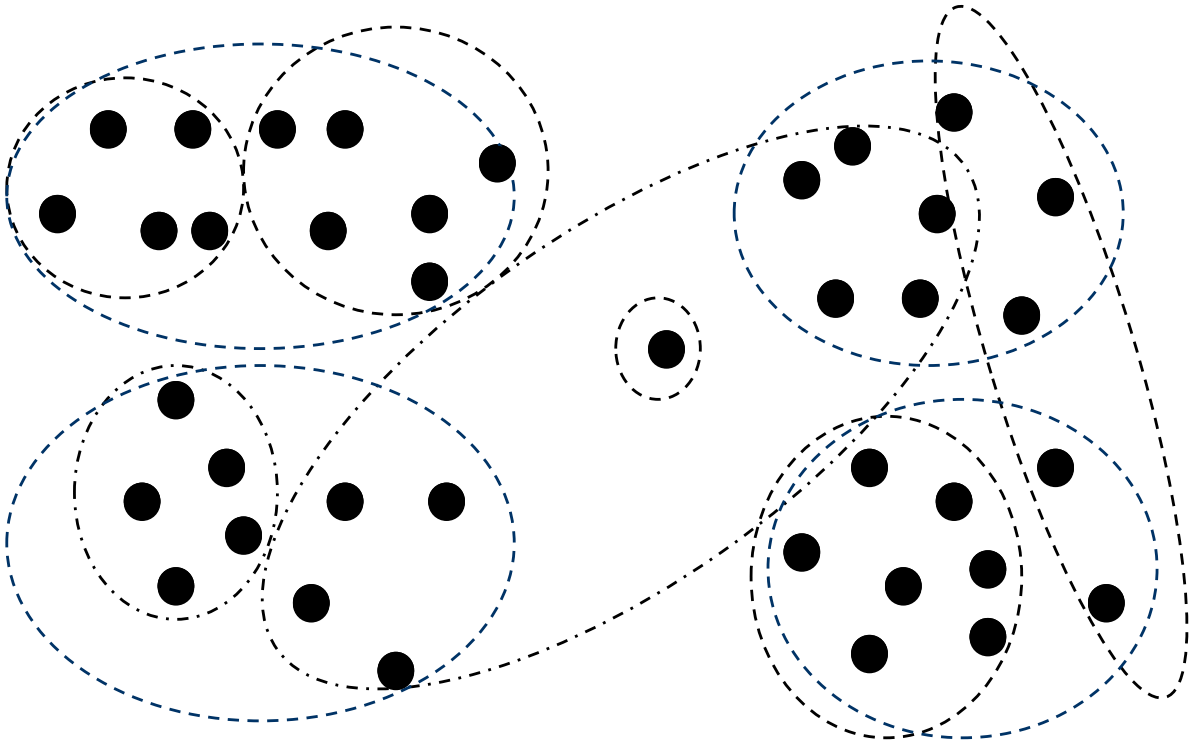
**E.g. Objects in a backpack**



# ML Classical Tasks

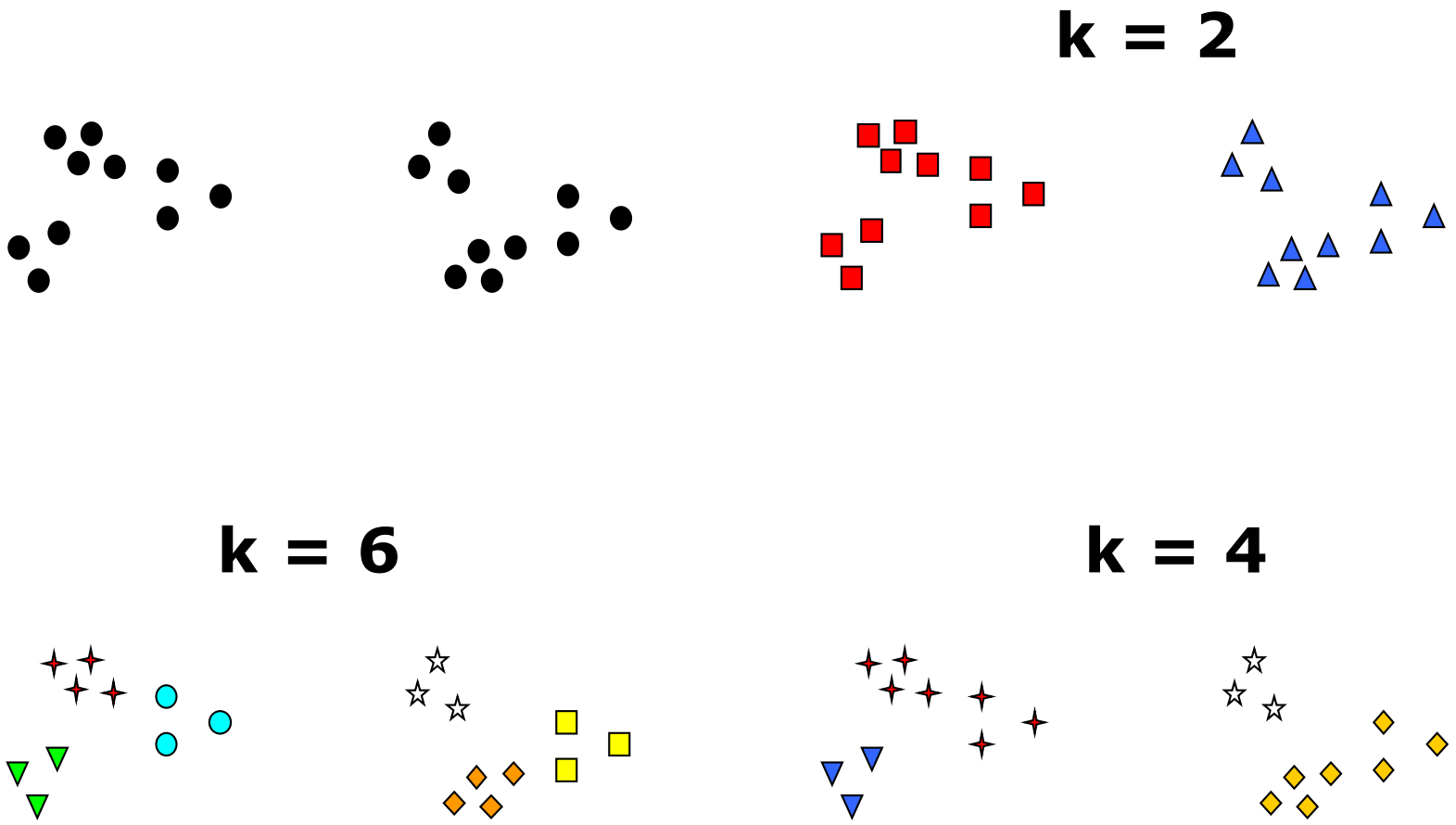
## Clustering. Levels



**Deciding on the number of clusters is one of the challenges in clustering**

# ML Classical Tasks

## Clustering. Levels



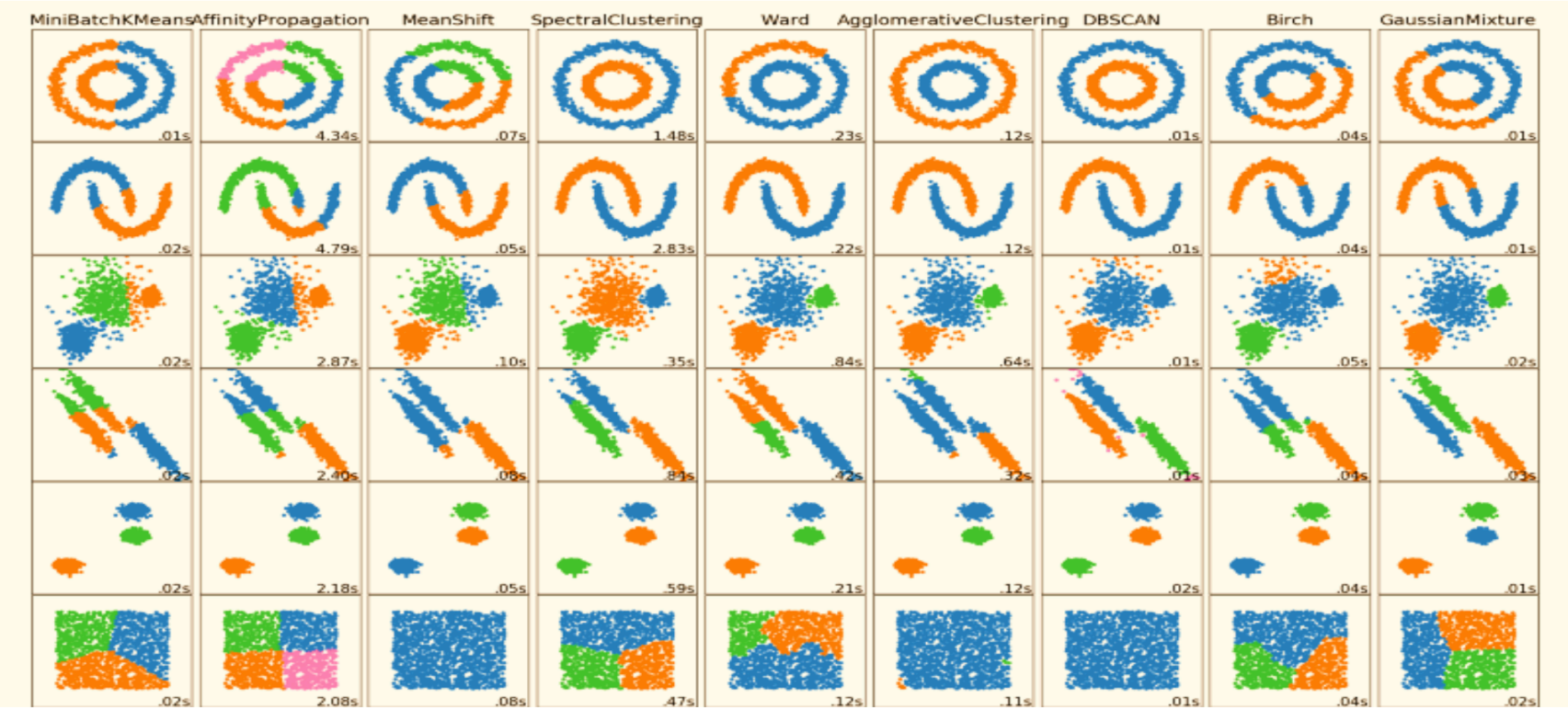

**Deciding on the number of clusters is one of the challenges in clustering**

# ML Classical Tasks

## Clustering. Algorithms.

**Goal**

Finding groupings so that objects in one group are similar to each other and different from objects in other groups [*clusters*].



# ML Classical Tasks

## Discovery of Associations

- Discovery of association rules (frequently associated with Data Mining exclusively):
  - Search for frequent patterns, associations, correlations, or causal structures between sets of items or objects (data) from transactional, relational, and other databases
  - Searching for sequences or time patterns
  - Applications :
    - *Market Basket analysis*
    - catalogue design,...
    - What's in the basket? Jazz Books
    - What could be in the basket? The latest Jazz CD
    - How do you motivate the customer to buy the items they are likely to like?

# ML Classical Tasks

## *Market Basket Analysis*

**Shopping:** orange juice, bananas, dishwashing detergent, glass cleaner, soda, ...

Is it typical to buy soda and bananas? Is the brand of the soda important?

How does neighborhood demographics affect customer purchasing?


Where should detergents be placed to maximize your sales?

Does the purchase of window cleaner increase when dishwashing detergent and orange juice are bought together?

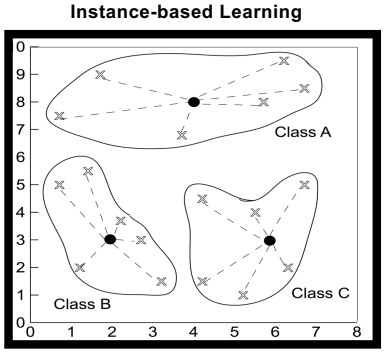
# ML Classical Tasks

- Classification [Predictive]
- Clustering [Descriptive]
- Association Rule Discovery [Descriptive]
- Sequential Pattern Discovery [Descriptive]
- Regression [Predictive]
- Deviation/Anomaly Detection [Predictive/Descriptive]
- Time Series [Predictive]
- Summarization [Descriptive]

# ML Algorithms

## Instance-based Learning

- They are based on learning by similarity
- They are part of the lazy learning paradigm, as opposed to the voracious one to which the previous paradigms belong
- Lazy: Computation is delayed as much as possible




- No model is built, the model is the BD or training set itself
- Work is done when a new case arrives for classification: the most similar cases are sought and the classification is constructed according to the class to which these cases belong

- The best known algorithms are based on the nearest neighbour rule

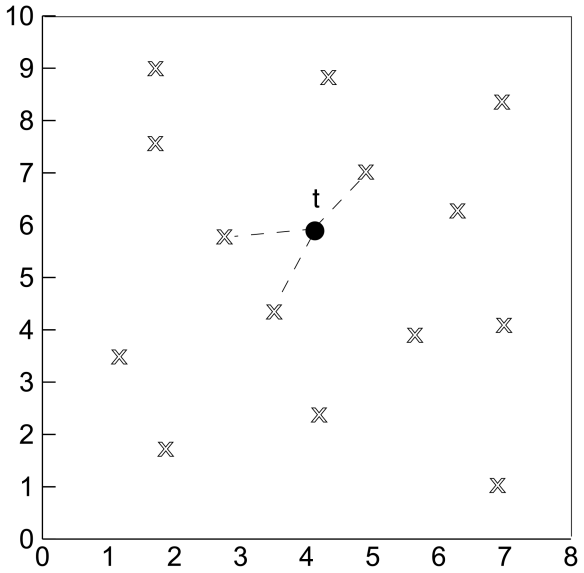
# ML Algorithms

## Instance-based Learning

*Nearest neighbour Rule* (1-NN)

- If we have *m* instances $\{e_1, \ldots, e_m\}$ in our data base, to clasify a new example e' the procedure is as follows:

1. $c_{min}$ = class ($e_1$)
2. $d_{min}$ = d ($e_1$, e')
3. For *i=2* to *m* do
   $d=d(e_i,e')$
   If ($d<d_{min}$)
   Then $c_{min}$ = class ($e_i$), $d_{min} = d$
4. Return $c_{min}$ as prediction of e'




- d(·,·) is a distance function
- In case of nominal/categorical variables, the *Hamming* distance could be used:

$$d_h(a,b) = \begin{cases} 0, & if \quad a = b \\ 1, & if \quad a \neq b \end{cases}$$

# ML Algorithms

## Instance-based Learning

### Distance functions for numerical variables

- Numerical variables are usually normalized to the interval [0,1]
- If $e_i^j$ is the value of the $j$ variable in $e_i$, namely $e_i=(e_i^1,\ldots,e_i^n)$ then some of the most used distances are
  - Euclidean

$$d_e(e_1,e_2)=\sqrt{\sum_{i=1}^{n}(e_1^i-e_2^i)}$$

  - Manhattan:

$$d_m(e_1,e_2)=\sum_{i=1}^{n}\left|e_1^i-e_2^i\right|$$

  - Minkowski

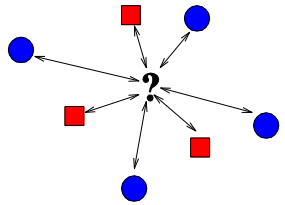
$$d_m^k(e_1,e_2)=\left(\sum_{i=1}^{n}\left|e_1^i-e_2^i\right|^k\right)^{1/k}$$

As we can observe, $d_m^1=d_m$ y $d_m^2=d_e$

# ML Algorithms

## Instance-based Learning

- The extension to the nearest neighbour rule is to consider the *k* nearest neighbours : k-NN (NN = 1-NN)
- Procedure: Give *e* as the example to clasify
  1. Select the k examples with $K = \{e_1,\ldots, e_k\}$ such there is no example $e'$ out of $K$ with $d(e,e')<d(e,e_i)$, $i=1,\ldots,k$
  2. Return the most repeated class in the set $\{class(e_1), \ldots, class(e_k)\}$ (majority class)
- For instance, if k=7 the next sample (?) will be clasified as ●

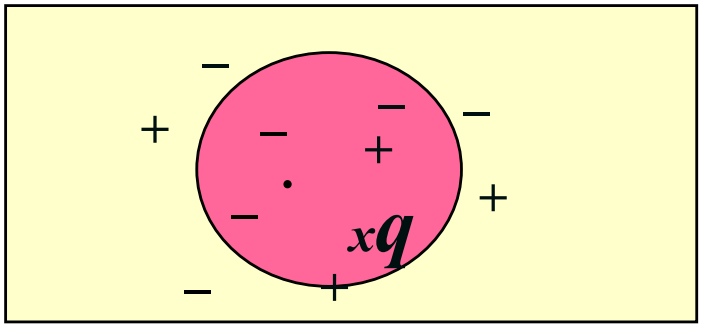


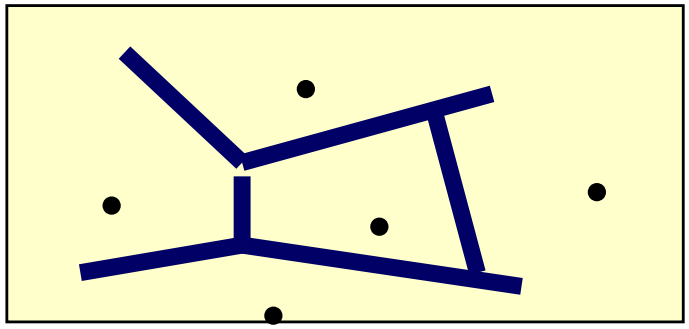

- It could be treated differently from k-neighbours, e.g., depending on the distance to the object to be sorted. In this way we would have:
  - Majority voting ●
  - Voting with distance-dependent weights □

# ML Algorithms

## Instance-based Learning

- The k-NN algorithm is robust to noise when moderate k values are used (k>1)
- It is quite effective, as it uses several local linear functions to approximate the target function
- It is valid for classification and for regression (in the siplest case, returning the average or the distance-weighted average)
- It is very inefficient in memory as all the DB has to be stored
- The distance between neighbours could be dominated by irrelevant variables
  - Pre-selection of characteristics: **Feature Selection**
- Its time complexity (to evaluate an example) is $O(dn^2)$ being O(d) the complexity of the distance used
  - One way to reduce this complexity is through the use of prototypes: **Prototype Selection**



# ML Algorithms

## Instance-based Learning

### 1-NN example

**Given the next data set with 4 instances, 3 attributes and 2 classes:**

**$x_1$: 0.4 0.8 0.2 positive**

**$x_2$: 0.2 0.7 0.9 positive**

**$x_3$: 0.9 0.8 0.9 negative**

**$x_4$: 0.8 0.1 0.0 negative**

**We calculate the distance of the example with all those of the set:**

$$d(x_1, y_1) = \sqrt{(0.4-0.7)^2 + (0.8-0.2)^2 + (0.2-0.1)^2} = 0.678$$

$$d(x_2, y_1) = \sqrt{(0.2-0.7)^2 + (0.7-0.2)^2 + (0.9-0.1)^2} = 1.068$$

$$d(x_3, y_1) = \sqrt{(0.9-0.7)^2 + (0.8-0.2)^2 + (0.9-0.1)^2} = 1.020$$

$$d(x_4, y_1) = \sqrt{(0.8-0.7)^2 + (0.1-0.2)^2 + (0.0-0.1)^2} = 0.173$$

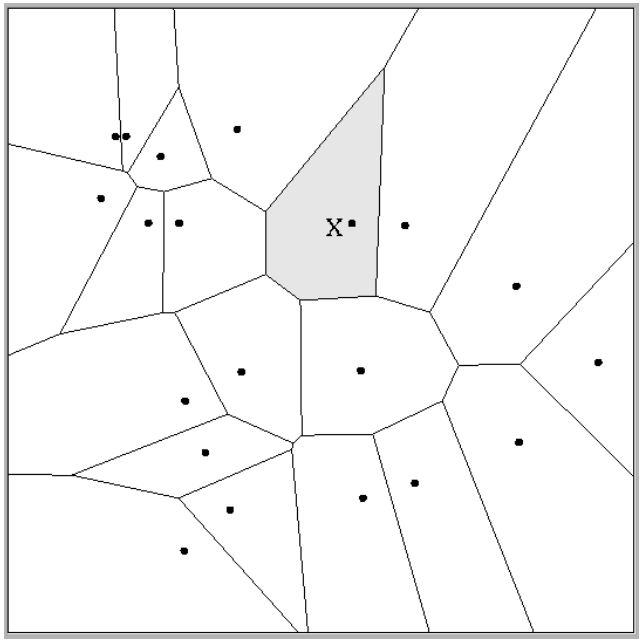
**Therefore, the example will be classified with respect to the negative class.**

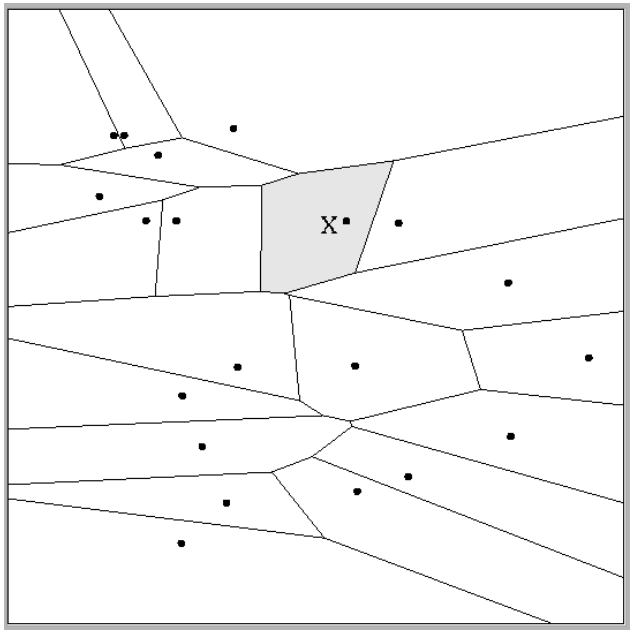
**IMPORTANT: Attributes must be standardized [0,1] so that they are not prioritized over others.**

# ML Algorithms

## Instance-based Learning

**Incorporation of weights in attributes**





$Dist(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$

$Dist(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (3x_{i2} - 3x_{j2})^2$

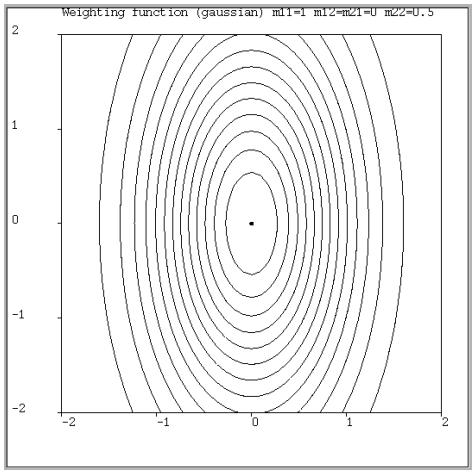
**Relative scales in distance metrics affect the shapes of regions**

# ML Algorithms

## Instance-based Learning

### Interesting Distance Metrics

**Scaled Euclidian ($L_2$)**

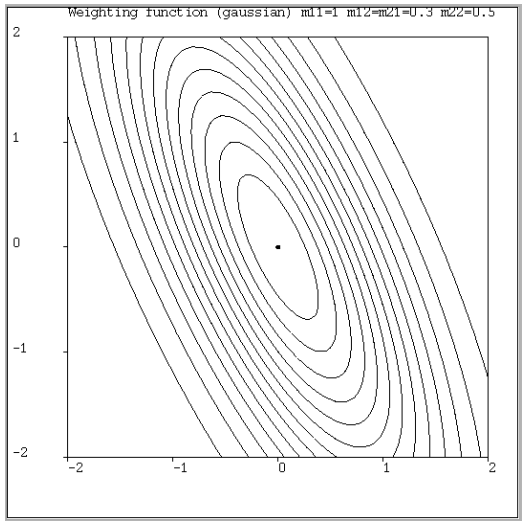



$$D(\mathrm{x},\mathrm{x}') = \sqrt{\sum_i \sigma_i^{\,2}\left(x_i - x'_i\right)^2}$$

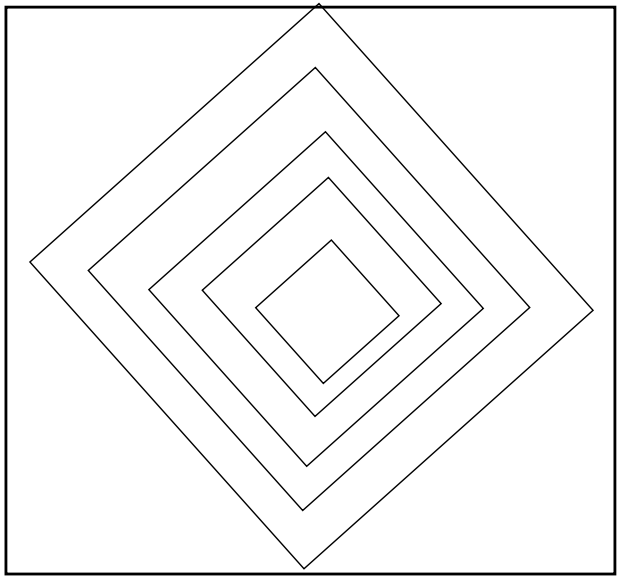
$$D(\mathrm{x},\mathrm{x}') = \sqrt{(\mathrm{x}-\mathrm{x}')^T \sum (\mathrm{x}-\mathrm{x}')}$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{bmatrix}$$

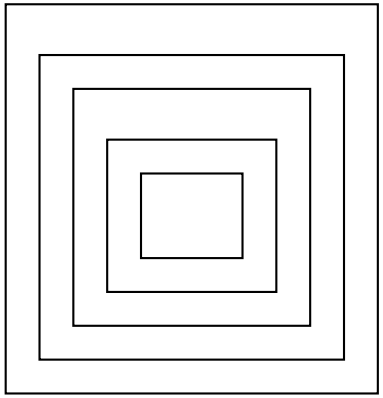
**Mahalanobis**




**$L_1$ norm (absolute)**



**$L_{infinity}$ *(max) norm***





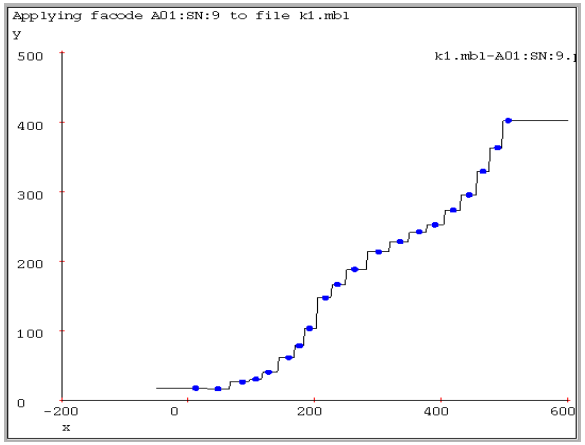
42

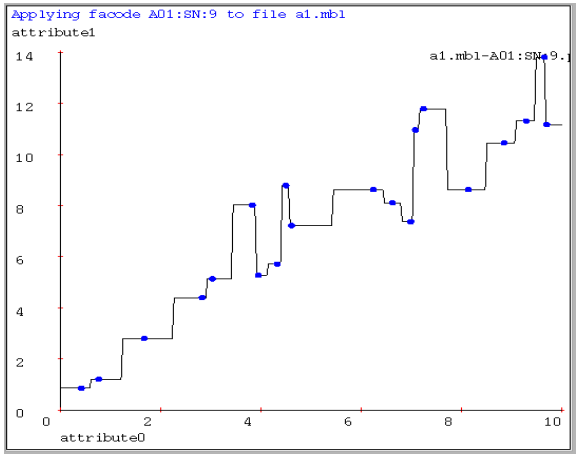
# ML Algorithms

## Instance-based Learning

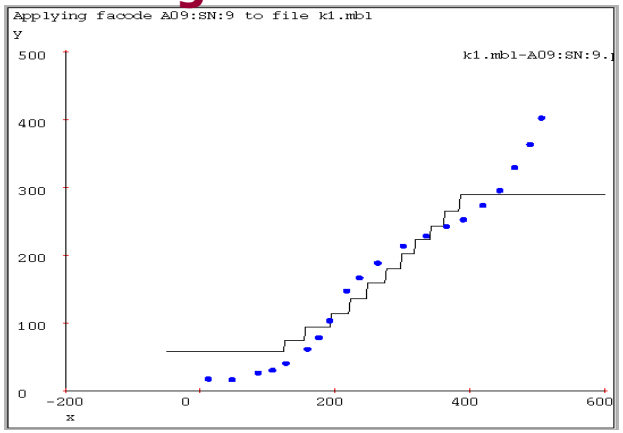
### K-NN in Regression

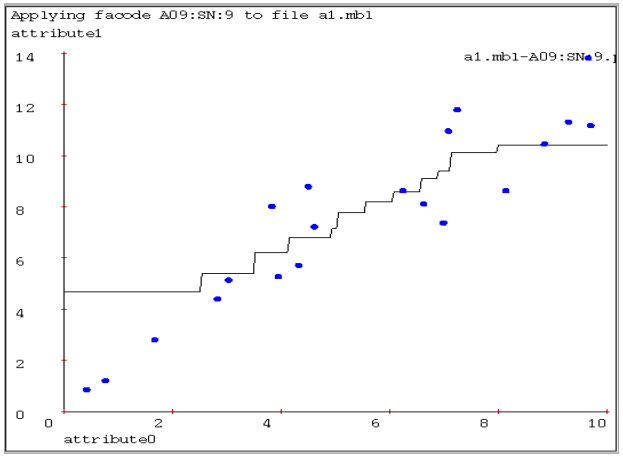
**1-NN**



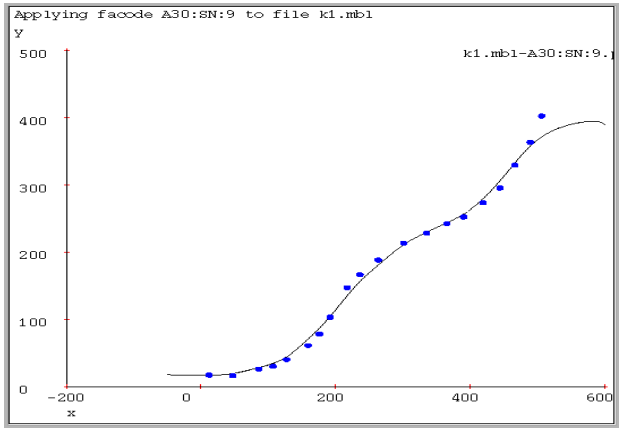



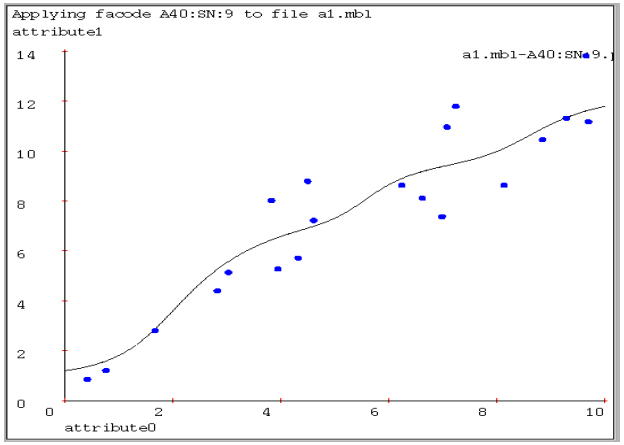

**9-NN: Predicts according to the neighborhood average**




**Kernel Regression**



# ML Algorithms

## Definition of Decision Trees

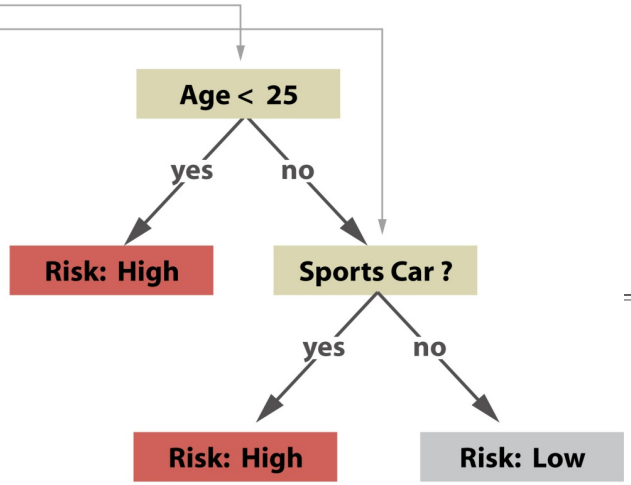
- A decision tree is a learner that according to a set of attributes allows to determine to which class or output the case under study belongs
- The structure of a decision tree is :
  - Each leaf is a category (class) or numeroc output of the variable that is the object of the prediction
  - Each node is a decision node that specifies a simple test (i.e. a comparison) to be performed
  - The descendants of each node are the possible results of the node test

# ML Algorithms

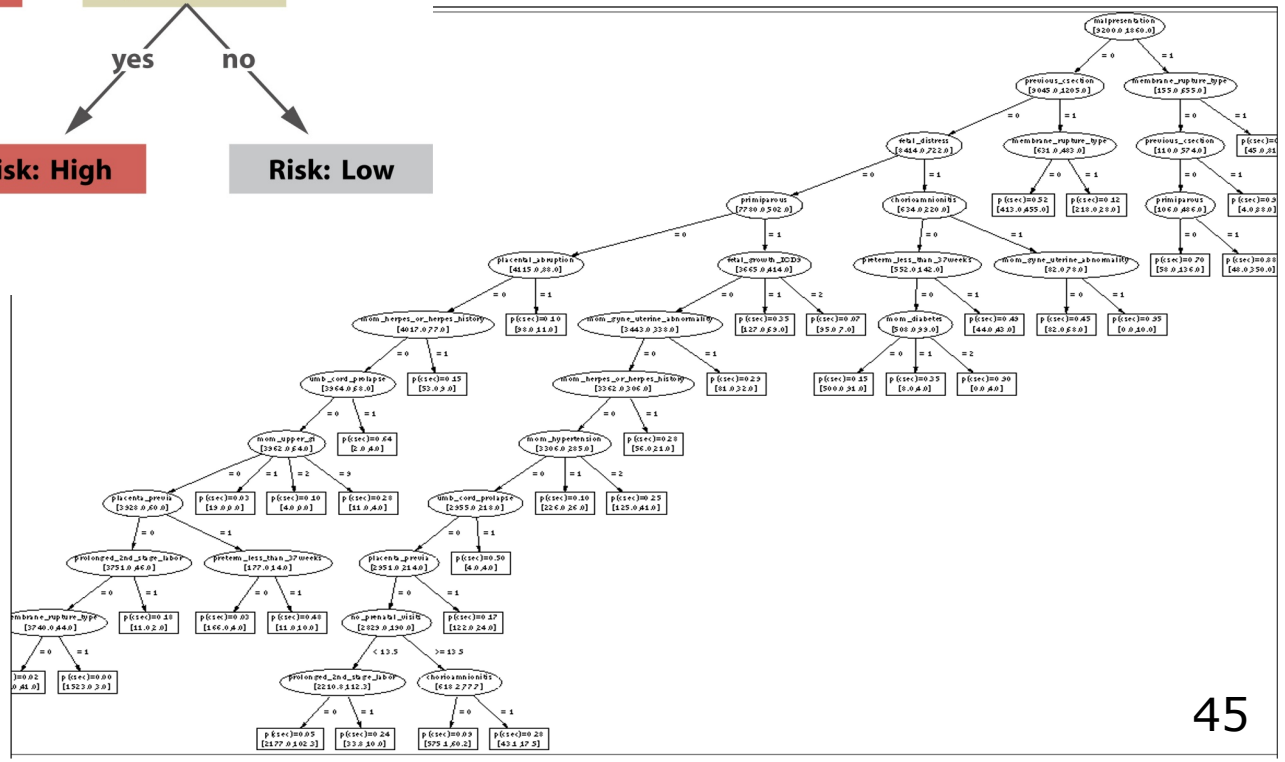
## Definition of Decision Trees

Insurance Risk Assessment

| Age | Car Type | Risk |
|---|---|---|
| 23 | family | High |
| 17 | sports | High |
| 43 | sports | High |
| 68 | family | Low |
| 32 | truck | Low |
| 20 | family | High |




**Real Decision Tree**



# ML Algorithms

## Building decision trees

- The process of generating a decision tree consists of two phases
  - Tree building
    - In the beginning all training examples are in the root node
    - The examples are recursively divided according to the selected attributes
  - Tree pruning
    - Identify and remove branches that describe noise or abnormal data
- Using the decision tree: Predicting an unknown example
  - Check the values of the attributes in the example through the decision tree

# ML Algorithms
## Feature Selection Criteria

There are different criteria for selecting the test variable X* at any given time. Some of the best known are :

- *InfoGain* (ID3)

$$X^* = \max_X (H(C) - H(C|X))$$

- *GainRatio* (C4.5)

$$X^* = \max_X \frac{H(C) - H(C|X)}{H(X)}$$

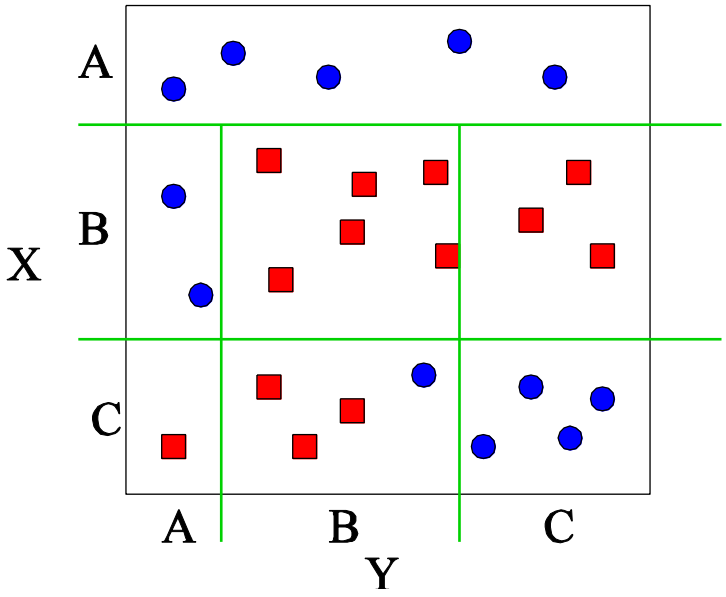
- GINI(CART/RPART)

$$X^* = \max_X (G(C) - G(C|X)) \quad con \quad G = 1 - \sum_{i=1}^{n} p_i^2$$

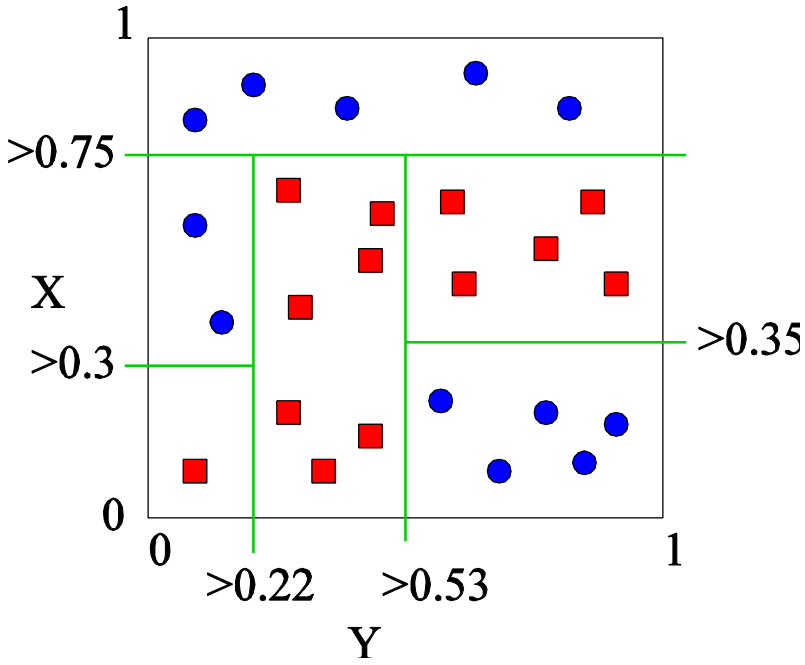
# ML Algorithms

## Partitioning the space in a decision tree

Trees partition the solution space comprehensively




Variables X and Y discrete




Variables X and Y continuous

# ML Algorithms

## Advantages and disadvantages of using decision trees

Advantages:

- Decision trees are easy to use and efficient
- The rules they generate are easy to interpret
- They scale better than other types of techniques
- They treat noise data well

Disadvantages:

- They do not easily handle continuous attributes
- They try to divide the domain of attributes into rectangular regions and not all problems are of that type
- Have difficulty working with missing values
- They may have problems of overfitting
- Thet don not detect correlations among attributes

# ML Algorithms
## A Decision tree Classifier: C4.5

### C4.5

*J.R. Quinlan. C4.5: Programs for Machine Learning. San Frnacisco: Morgan Kaufmann, 1993*

It improves ID3 (the pioneer decisiun tree) in the next aspects:

- Missing Values:
  - When the tree is constructed, the missing data is ignored (the decision tree is constructed by looking only at the records that have value for that attribute)
  - To classify an example with a missing value, it is predicted based on what is known about the attribute values for other records
- Numeric Data: It is divided into ranges based on the values found in the training set
- It proposes solutions for overfitting. Possibilities
  - pre-pruning: decide when to stop subdividing the tree
  - post-pruning: the tree is built and then pruned

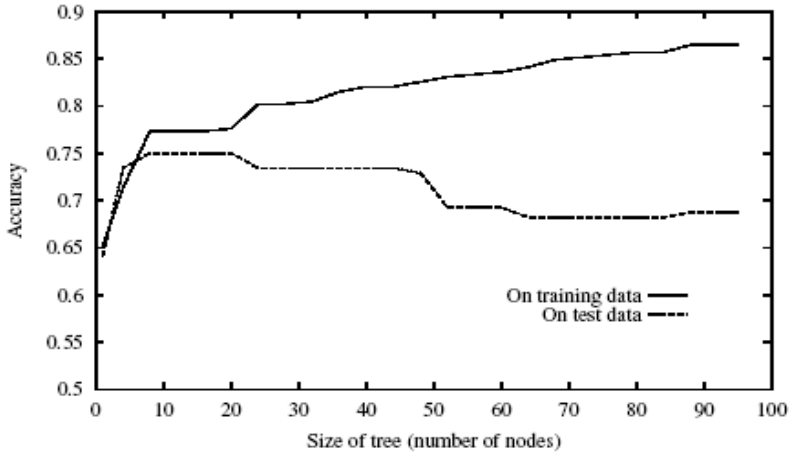
# ML Algorithms

## A Decision tree Classifier: C4.5

Pre-pruning:

• a node is not divided if there is little confidence in it (the class difference is not significant), or

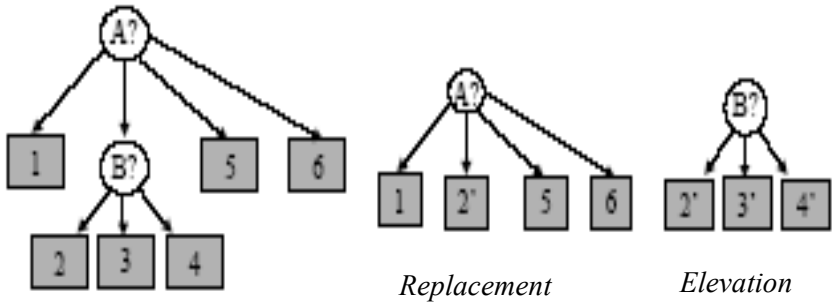
• it is validated with an independent test set and stops when the test set curve starts to go down (accuracy)




There are two strategies of post-pruning in C4.5

• Replacement of sub-trees: A sub-tree is replaced by a leaf if in doing so the error is similar to the original

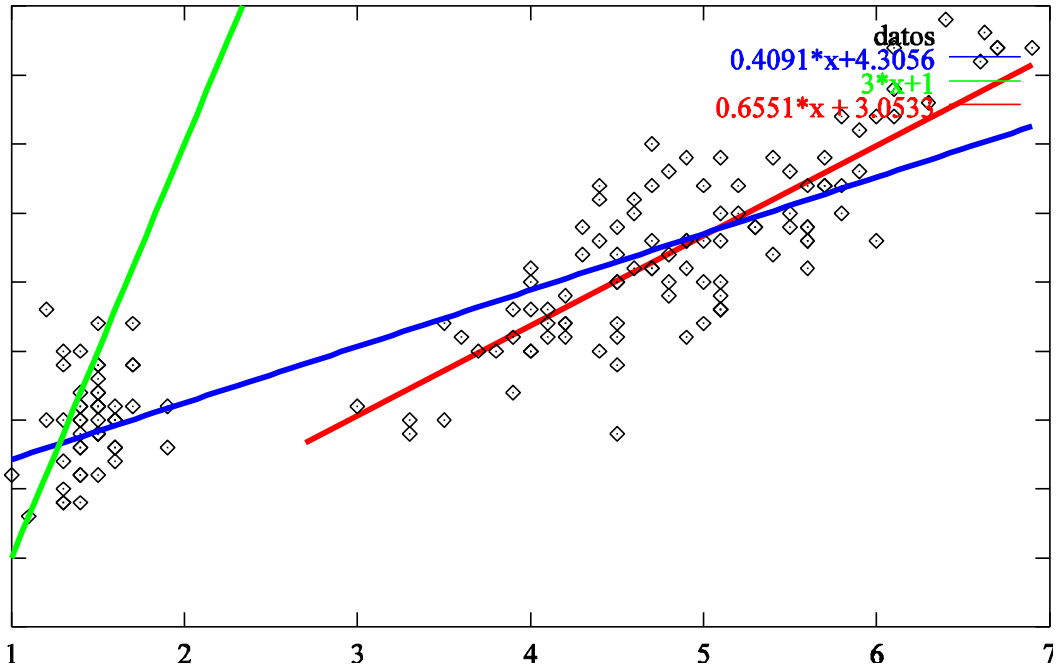
• Sub-tree elevation: Replaces a sub-tree with its most used sub-tree (a sub-tree moves from its location to a higher node in the tree)



# ML Algorithms

## M5: Model Trees

- They are regression trees in which the pruning is done to a greater extent and the leaves instead of a numerical value contain a local regression equation to that partition of space



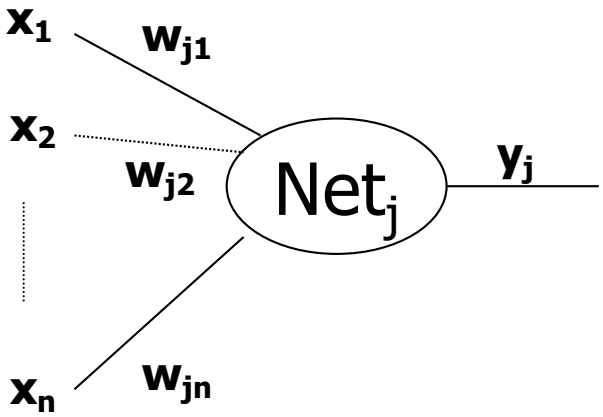

**MMAX <= 14000 :**
**| CACH <= 8.5 : LM1 (108/3.99%)**
**| CACH > 8.5 : LM2 (33/3.89%)**
**MMAX > 14000 :**
**| MMAX <= 22500 : LM3 (37/4.73%)**
**| MMAX > 22500 : LM4 (31/69.2%)**

**LM1: class = 15.9 - 0.00453MYCT + 0.00327MMAX**
**LM2: class = -0.609 + 0.004MMAX + 0.59CACH + 1.57CHMIN**
**LM3: class = 1.64 - 0.0266MYCT + 0.00485MMIN + 0.00346MMAX + 0.627CACH + 1.43CHMIN + 0.127CHMAX**
**LM4: class = -350 - 0.843MYCT + 0.0183MMAX + 1.62CACH**

**EMR 10.0402**
**EAR 11.4899 %**

# ML Algorithms

## Neural Networks: Artificial neurons




- The signals that reach the dendrites are represented as $x_1$, $x_2$, ..., $x_n$.
- The synaptic connections are represented by weights $w_{j1}$, $w_{j2}$, $w_{jn}$ that weight (multiply) the entries. If the weight between neurons j and i is :
  a) positive, it represents an excitatory synapse
  b) negative, represents an inhibitory synapse
  c) zero, no connection

The integrative action of the cell body (or internal activity of each cell) is presented by

$$Net_j = w_{j1} \cdot x_1 + w_{j2} \cdot x_2 + \ldots + w_{jn} \cdot x_n = \sum_{i=1}^{n} w_{ji} \cdot x_i$$

- The output of the neuron is represented by $y_j$. It is obtained through a function that is generally called output, transfer or activation function. This function depends on $Net_j$ and a parameter j which represents the activation threshold of the neuron

$$y_j = f(Net_j - \theta_j) = f\left(\sum w_{ji} \cdot x_i - \theta\right)$$

# ML Algorithms

## Types of transfer functions

- **Step function** or Haviside. It represents a neuron with only two states of activation: activated (1) and inhibited (0 ó -1)

$$y_j = H(Net_j - \theta_j) = \begin{cases} 1, & si\ Net_j >= \theta_j \\ -1, & si\ Net_j < \theta_j \end{cases}$$

- **Linear function**:

$$y_j = Net_j - \theta_j$$

- **Linear function in sections**:

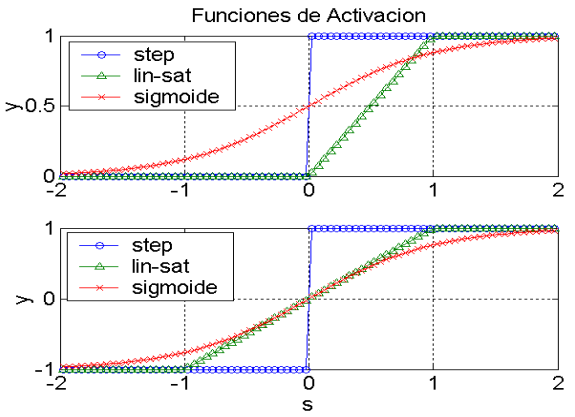
$$y_j = \begin{cases} 1, & si\ Net_j >= \theta_j + a \\ Net_j - \theta_j, & si\ |Net_j - \theta_j| < a \\ -1, & si\ Net_j < \theta_j - a \end{cases}$$

- **Sigmoidal function**:

$$y_j = \frac{1}{1+e^{-\lambda(Net_j-\theta_j)}} \qquad y_j = \frac{2}{1+e^{-\lambda(Net_j-\theta_j)}} - 1$$

- **Radial-basis function**:

$$y_j = e^{-\left(\frac{Net_j-\theta_j}{\sigma}\right)^2}$$



# ML Algorithms

## ¿What is a Neural Network?

An **artificial neuron network** is characterised by its :

- **Architecture**: Structure or pattern of connections between process units

- **Computational dynamics** which expresses the value taken by the processing units and which is based on **activation (or transfer) functions** that specify how the input signals from the processing unit are transformed into the output signal

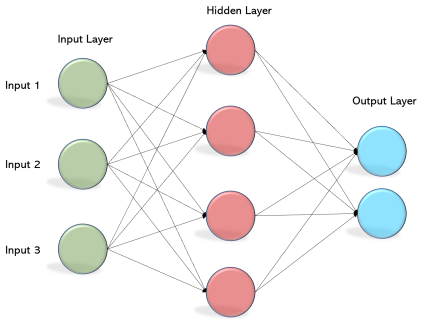
- **Training or Learning Algorithm**: Procedure for determining the weights of the connections. A very important feature of these networks is their **adaptive** nature, where **'learning by example'** replaces 'programming' in problem solving.

# ML Algorithms

## Multi-Layer Perceptron (MLP) in Classification

- Learning the structure of a neural network requires experience, although there are some guidelines.

- Inputs: For each numerical or binary/boolean variable an input neuron is set. For each nominal variable (with more than two states) an input neuron is put for each possible state.
- Outputs: If it is for clasification, a single output neuron is placed for each value of the class variable. For regression only a output neuron is needed.
- Hidden layers. You have to indicate how many layers and how many neurons have to be put in each layer.

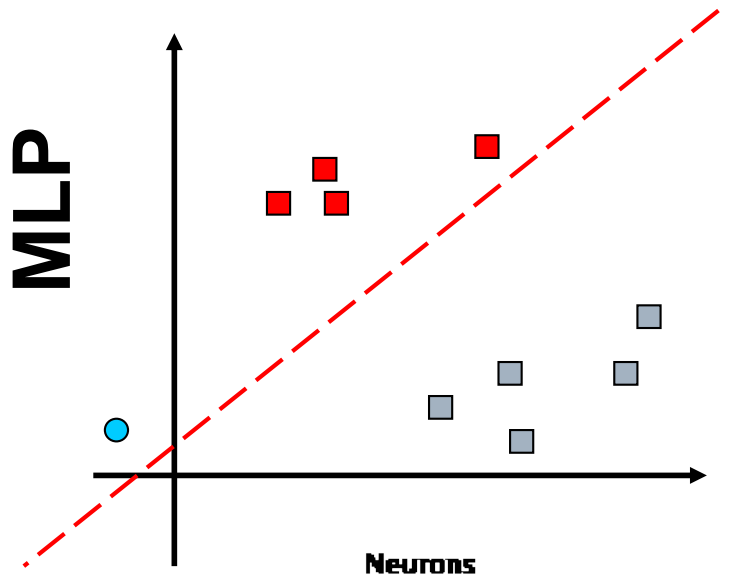


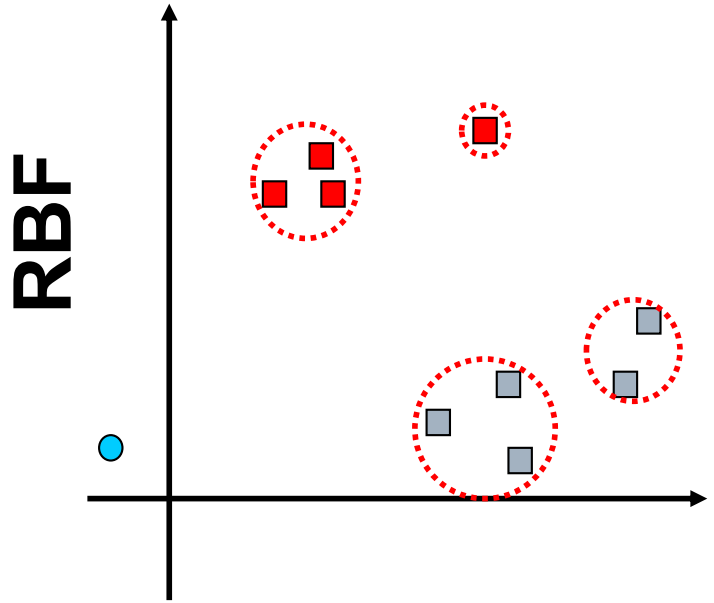

- When the neural network is trained, to predict an instance the values of the same are introduced corresponding to th variables of the input nodes:
  - The output of each output node indicates the probability that the instance. belongs to that class
  - The instance is assigned to the class with the highest probability.

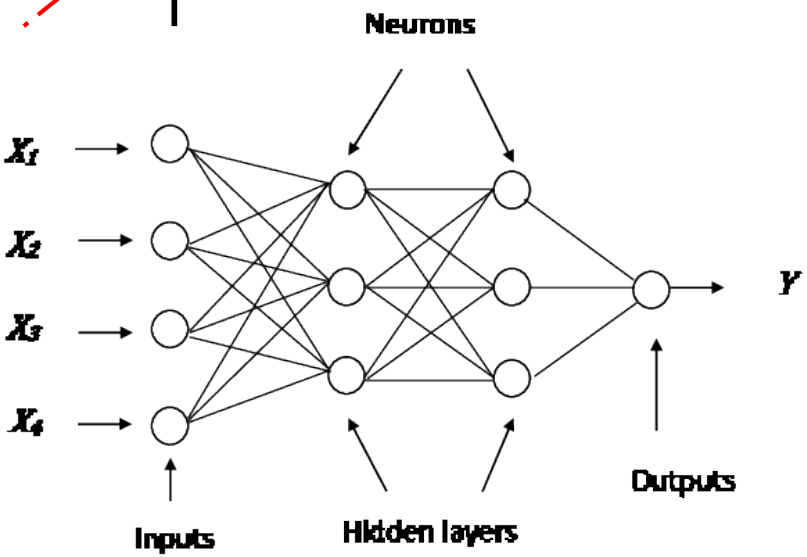
- All numerical variables are standardised [-1.1].
- Backprogation algorithm:
  - Based on the descending gradient technique
  - Does not allow backward connections (feedback) in the network

# ML Algorithms

## Radial-basis Functions Networks (RBFNs)




**Approximate function with linear combination of radially based functions**

$$f(x) = \sum w_j h_j(x)$$

$$h_j(x) = \exp( -(x - c_j)^2 / r_j^2 )$$

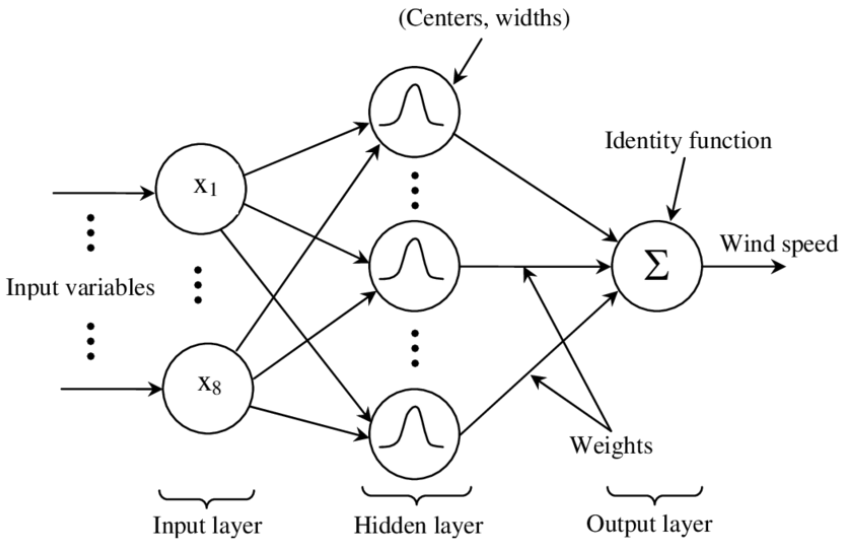
Where $c_j$ is center of a region,

$r_j$ is width of the receptive field

# ML Algorithms

## Radial-basis Functions Networks (RBFNs)

- Designing a neural network as a curve fitting problem
- Learning: Finding the surface in multidimensional space that best fits the training data
- Generalization: Use this multidimensional surface to **interpolate test data**
- Three Layers:
  - Input layer
    - Source nodes that connect the network to your environment
  - Hidden layer
    - Hidden units provide a set of basic functions
    - High dimensionality
  - Output layer
    - Linear combination of hidden functions

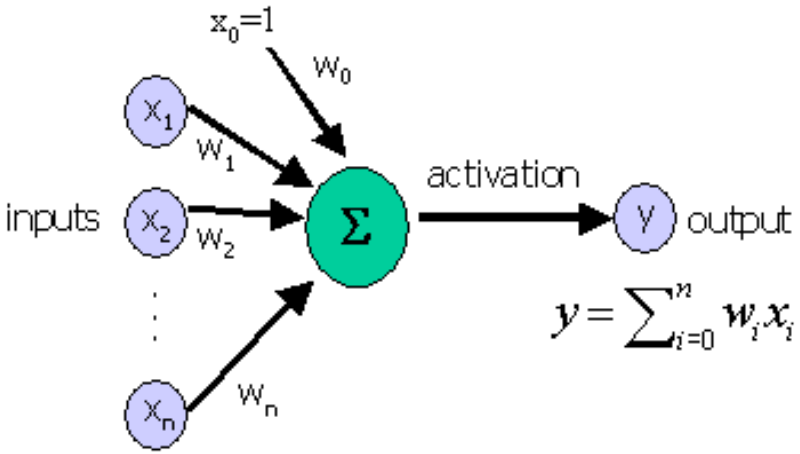


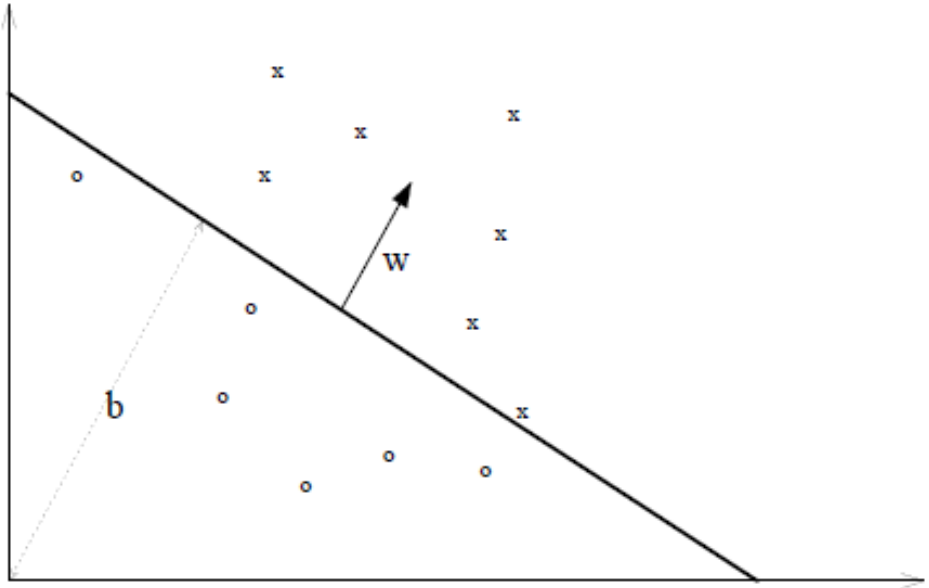
# ML Algorithms

## Support Vector Machines (SVMs)

- *Hyperplane: w* is the normal vector to the plane. *b* is the distance fro the origin as reference to the vector *w.*
- **Linear Learning Machines (LLMs):** The simplest case is to establish a hyperplane as a decision function in space

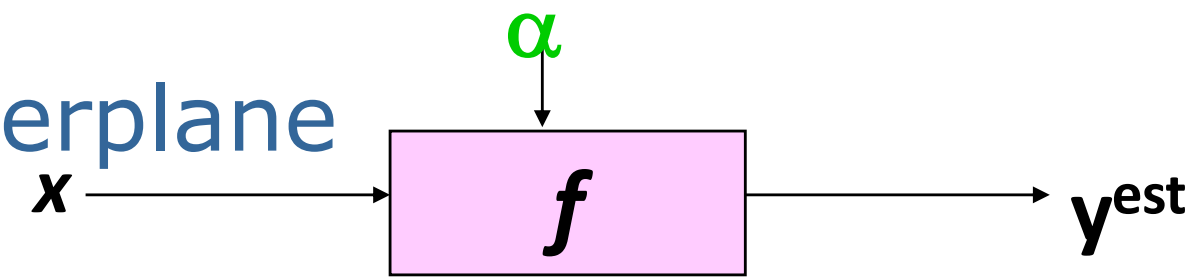
$$\langle w, x \rangle + b = 0$$



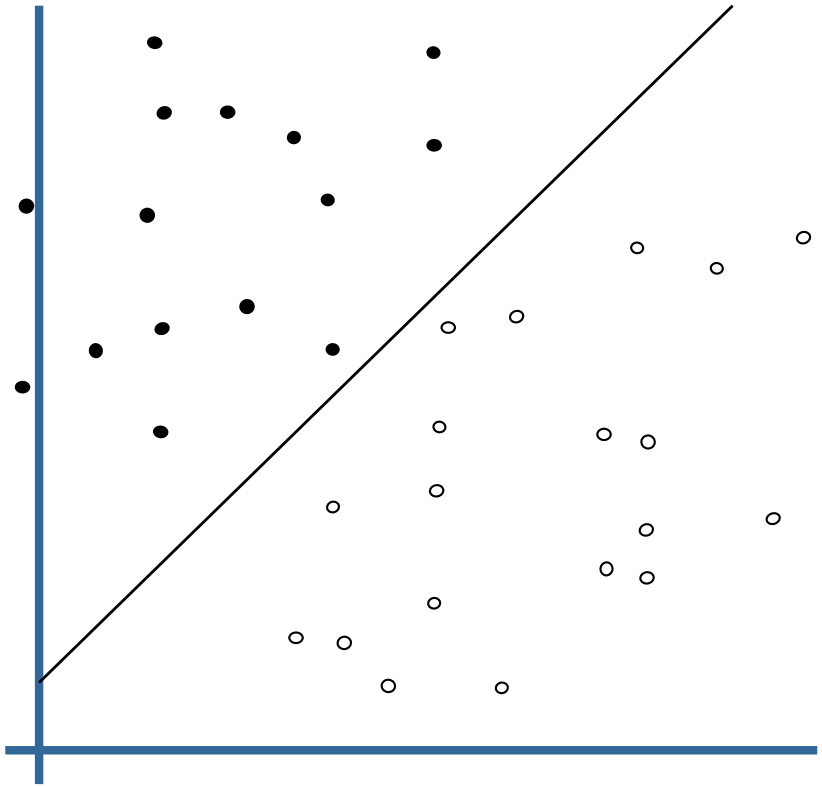

- A typical example is the perceptron!

# Choose the hyperplane



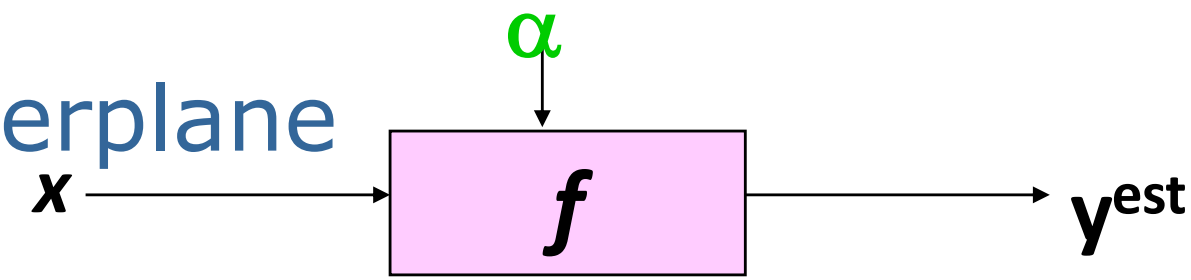

$f(x,w,b) = sign(w.\ x - b)$

- denotes +1
- denotes -1



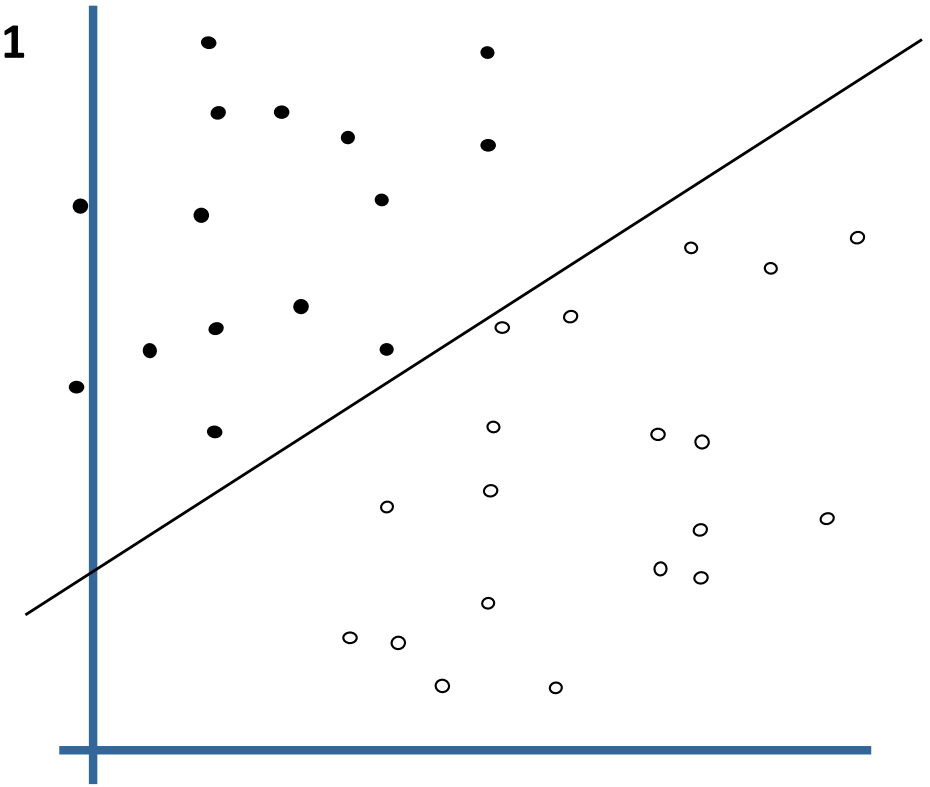
**How would you classify this data?**

# Choose the hyperplane



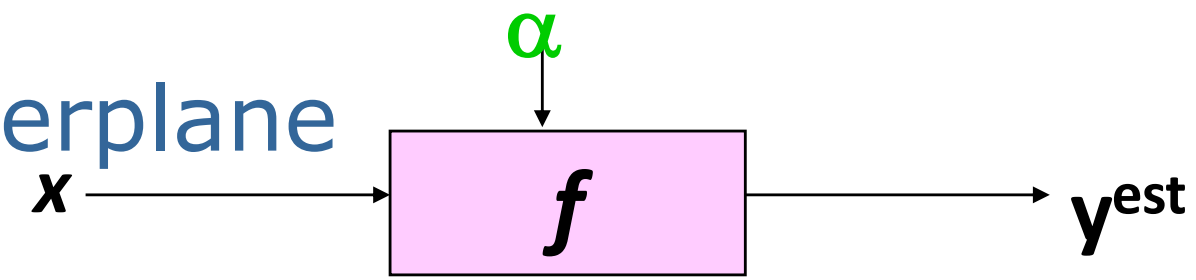

$f(x,w,b) = sign(w.\ x - b)$

- denotes +1
- denotes -1

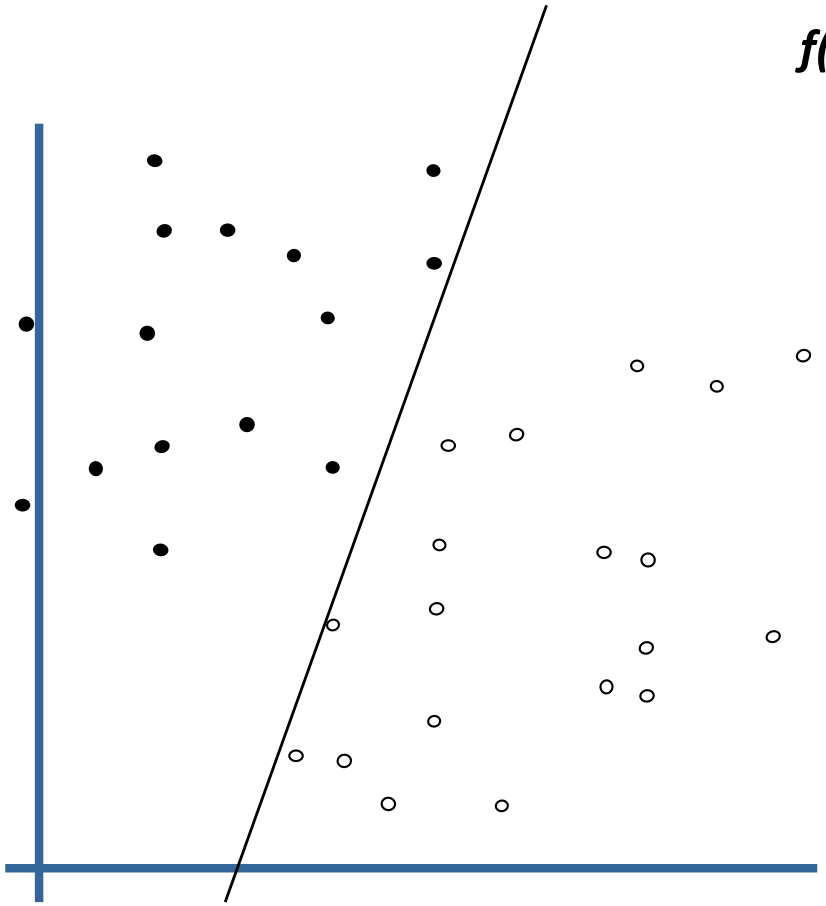


**How would you classify this data?**

# Choose the hyperplane



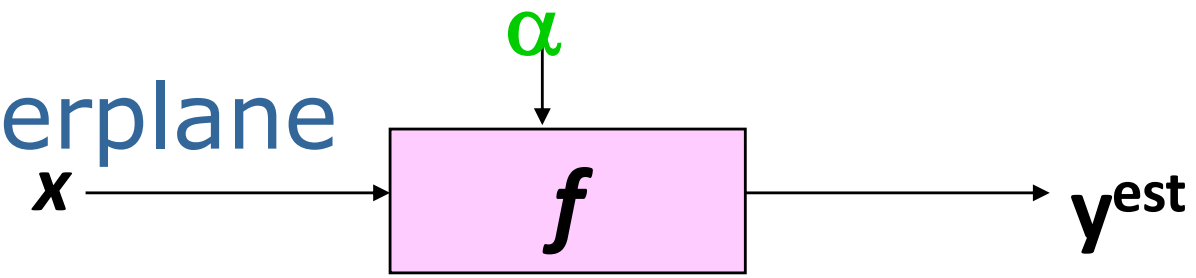

- denotes +1
- denotes -1

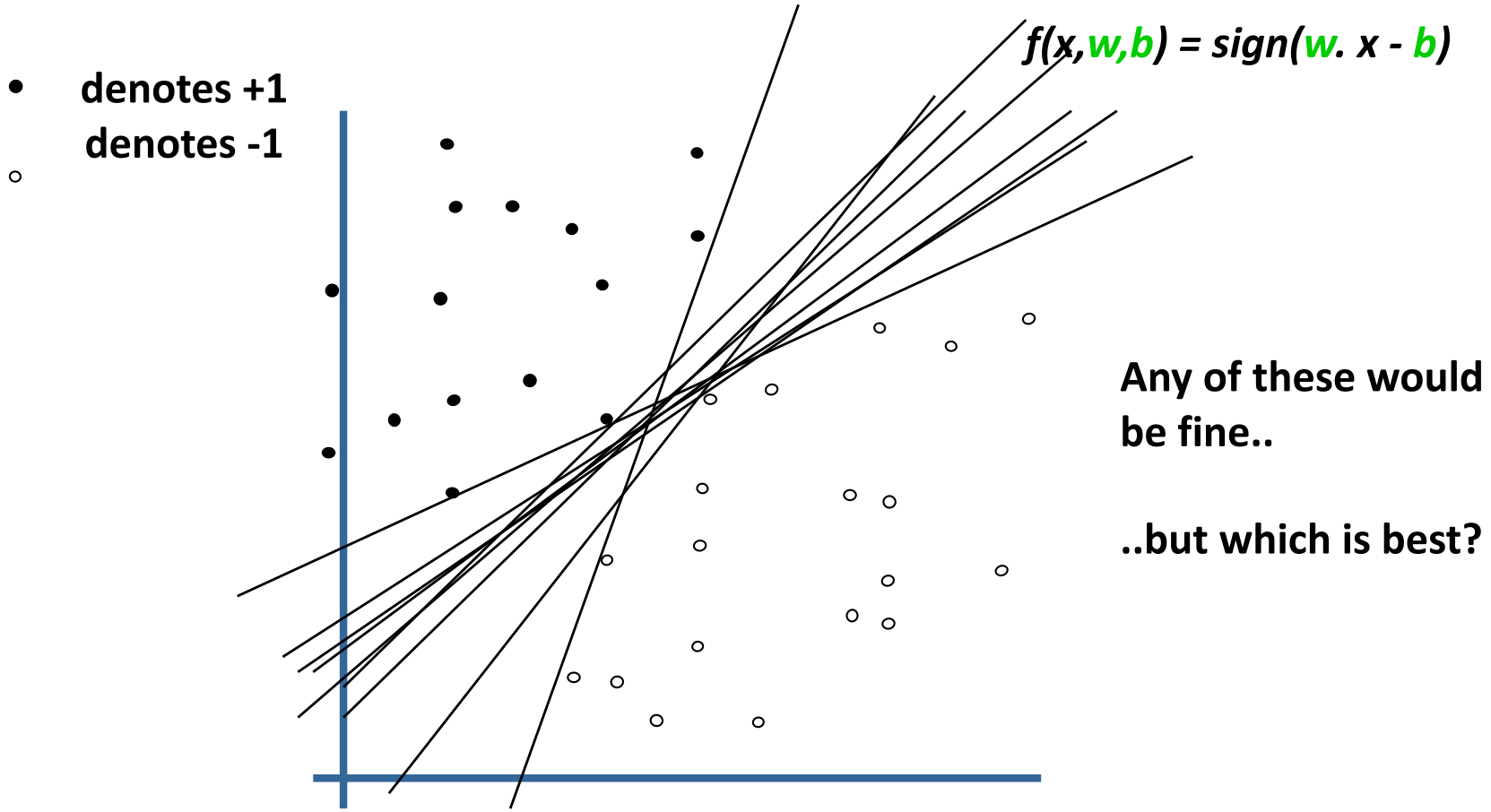


*f(x,w,b) = sign(w. x - b)*

**How would you classify this data?**

# Choose the hyperplane

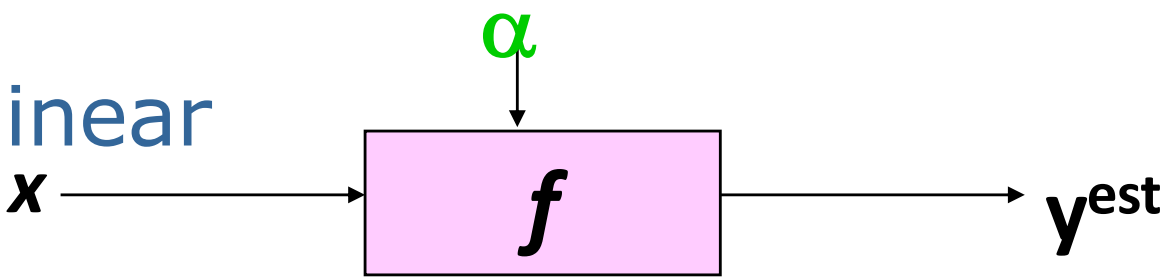


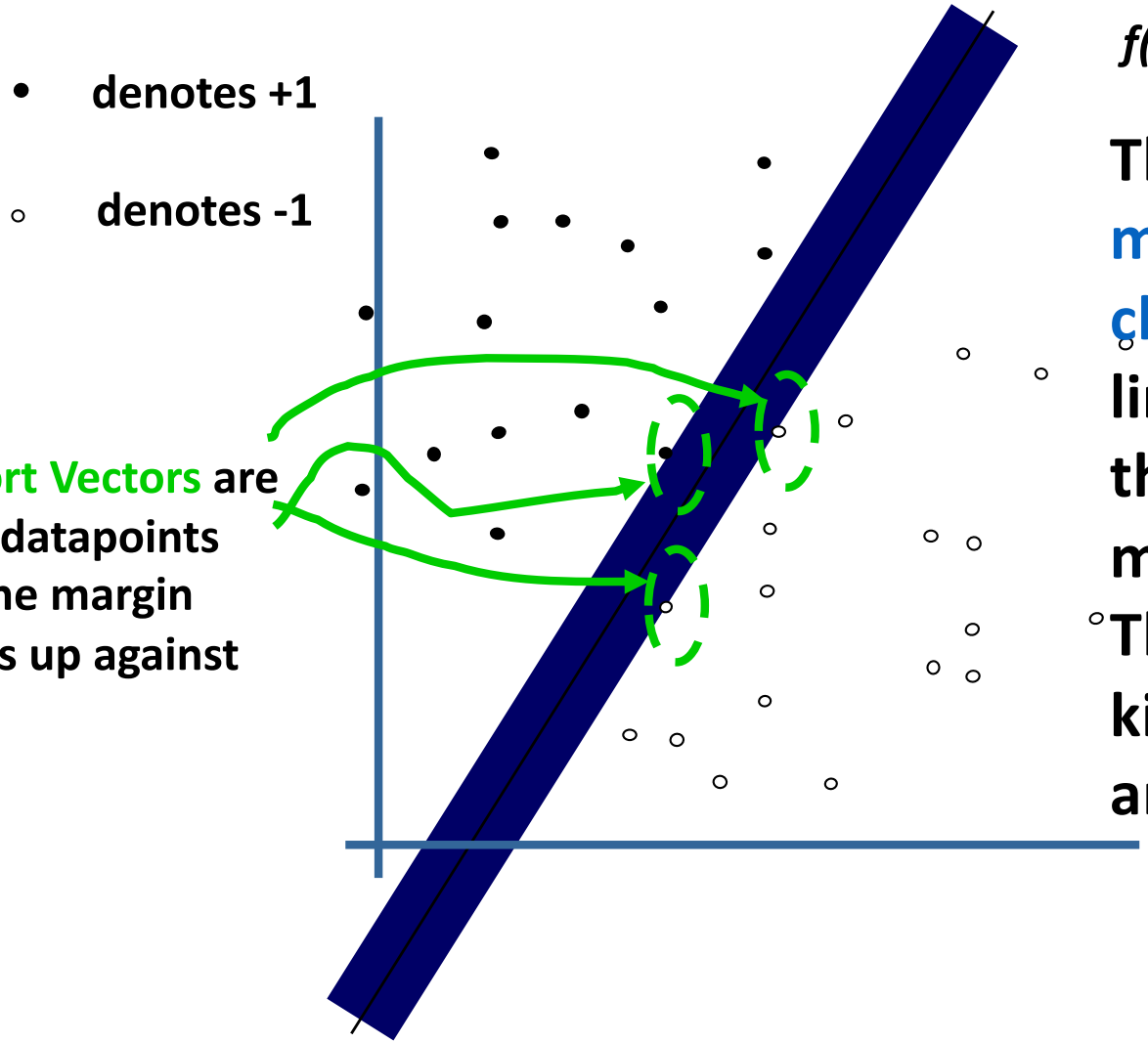




**Any of these would be fine..**

**..but which is best?**

# Maximum margin linear classifiers




$f(x,w,b) = sign(w.\ x - b)$

**Support Vectors** are those datapoints that the margin pushes up against

The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

**Linear SVM**

# ML Algorithms

## SVMs: Optimization

- The problem of finding the maximum margin plane is a conditioned optimization problem (Quadratic Programming or QP)
- The Lagrange theory (or Kuhn-Tucker theory) is often used.
- Lagrange:

$$\frac{1}{2}\langle w, w\rangle - \sum \alpha_i y_i \left[\left(\langle w, x_i\rangle + b\right) - 1\right]$$

$$\alpha \geq 0$$

- The goal is to find w y b so that α calculated for each example allow that thw value of <w,w>=||w|| is minimum

# ML Algorithms

## Complete solution: SMO

- SMO (Sequential Minimal Optimization) updates **TWO** weights in each step
- Can make descending gradient by satisfying linear constraints (J. Platt)
- LibSVM uses the SMO method with a heuristic selection of the two Lagrange weights/parameters to be optimised at each step
- SMO chooses the two multipliers α in an uninformed way
- LibSVM proposes to choose two multipliers in a more intelligent way to improve the convergence of the algorithm
- This algorithm is **C-SVM**
- LibSVM also includes **v-SVM**
  - The parameter **v** acts in inverse proportion to the parameter C
  - Larger **v** → smaller C and vice versa
  - Original formulation for **regression**

# ML Algorithms

## Dual Representation

- The decision function can be rewritten as follows:

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

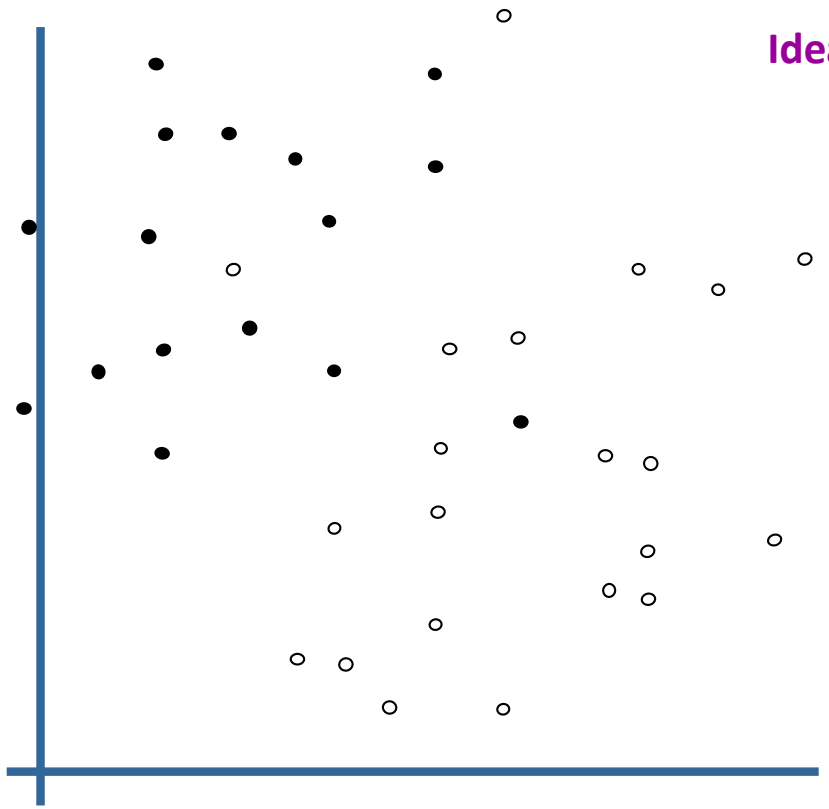
$$w = \sum \alpha_i y_i x_i$$

- It is very important because it allows us to define the decision function from the inputs
- SVMs are Linear Learning Machines (LLMs) represented in a dual form
- The input data only acts on scalar products:
  - In the decision function
  - In the training algorithm

# LLMs: Noise and non-separability

## Uh-oh!

• denotes +1
○ denotes -1



**This is going to be a problem!**
**What should we do?**

**Idea 1:**

**Find minimum *w.w*, while minimizing number of training set errors.**

**Problem: Two things to minimize makes for an ill-defined optimization**

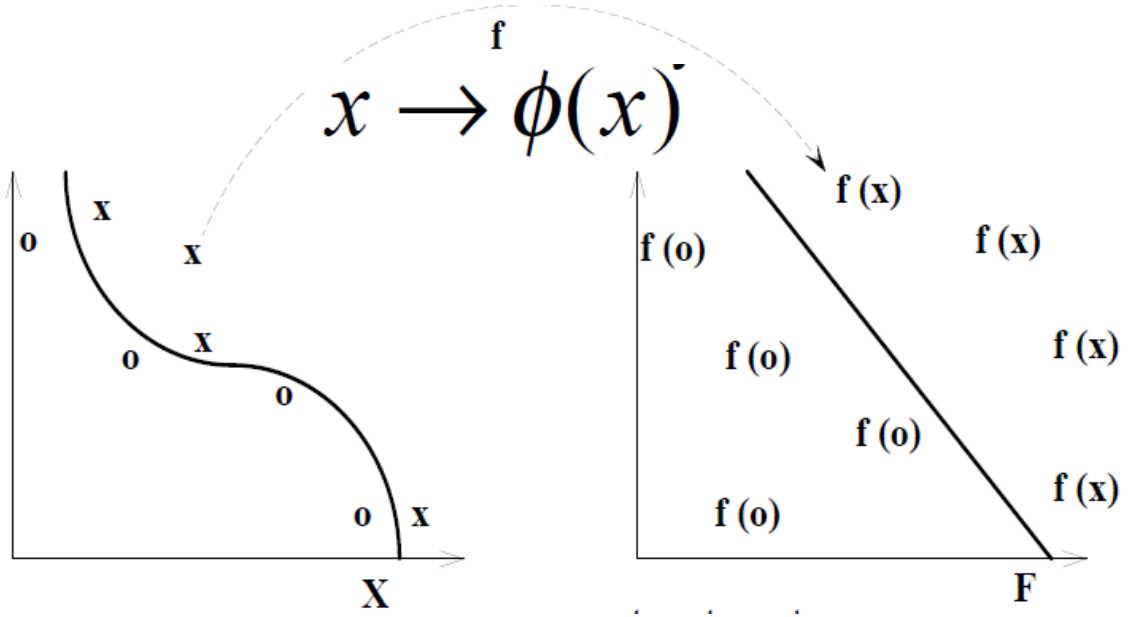
**Idea 2.0:**

**Minimize**
***w.w + C (distance of error points to their correct place)***

# ML Algorithms

## Kernel Functions

- Assign the data in a more complex space, with non-linear characteristics, and use a linear classifier in this space
- **¡Idea!** → we will move the data to another feature space where they are linearly separable



# ML Algorithms

## Kernel Functions

- Using the **dual representation**, examples only appeared in the scalar products

$$f(x) = \sum \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

- The dimensionality of the F space is not necessarily important. We don't even need to know the function Φ.
- Examples of kernels:

**Polinomial** $K(x,z) = \langle x,z \rangle^d$

**RBF** $K(x,z) = e^{-\|x-z\|^2 / 2\sigma}$

# ML Algorithms

## Effects of parameters in SVM

SVMs are **VERY SENSITIVE** to the parameters

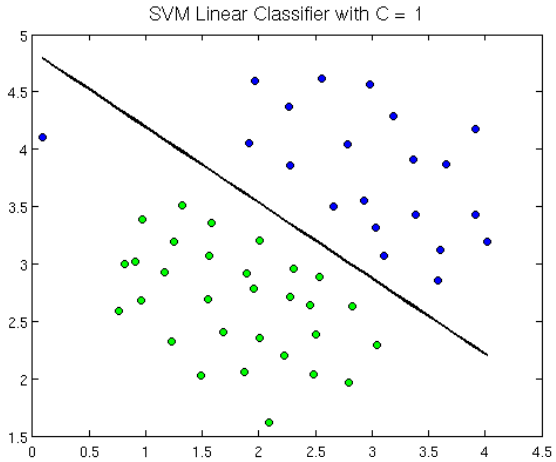
- The value of C changes the behaviour of the SVM:
  - A low value makes the SVM more permissive of outliers
  - A high value makes the SVM less permissive with strange examples
- High values of C → weights α other than zero for more examples (increases the number of support vectors) and lower average values (less significant support vectors)
- Why? Wrong classifications have a higher cost if C is higher ← **overfitting danger**

# ML Algorithms

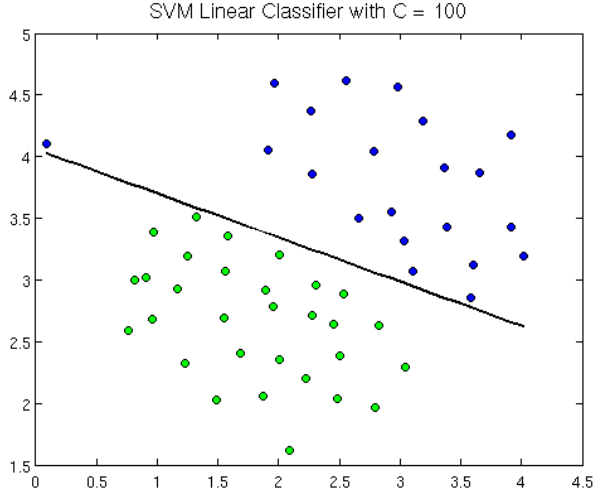
## SVM parameter adjusting: C

Where would you draw the hyper plane of separation?

- C = 1
  - It seems reasonable if we consider the outlier as noise




- C = 100
  - The separation does not seem so natural.
  - It can be over-adjusted.
  - Many support vectors involved



# ML Algorithms

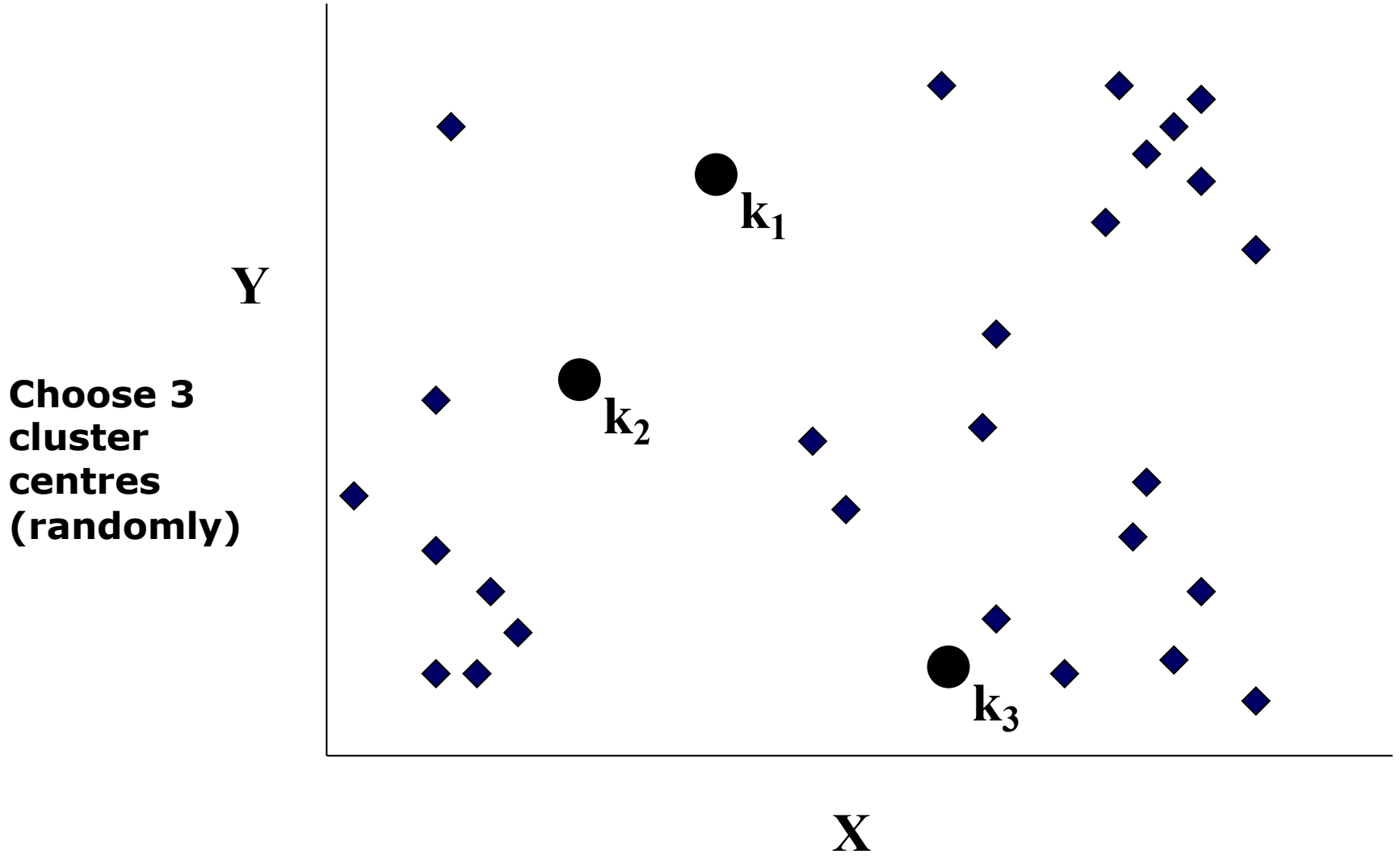
## Clustering: K-Means

- It needs as an input parameter the desired number of clusters

- It is an iterative algorithm in which instances are moved between clusters until the desired set of clusters is reached

# ML Algorithms

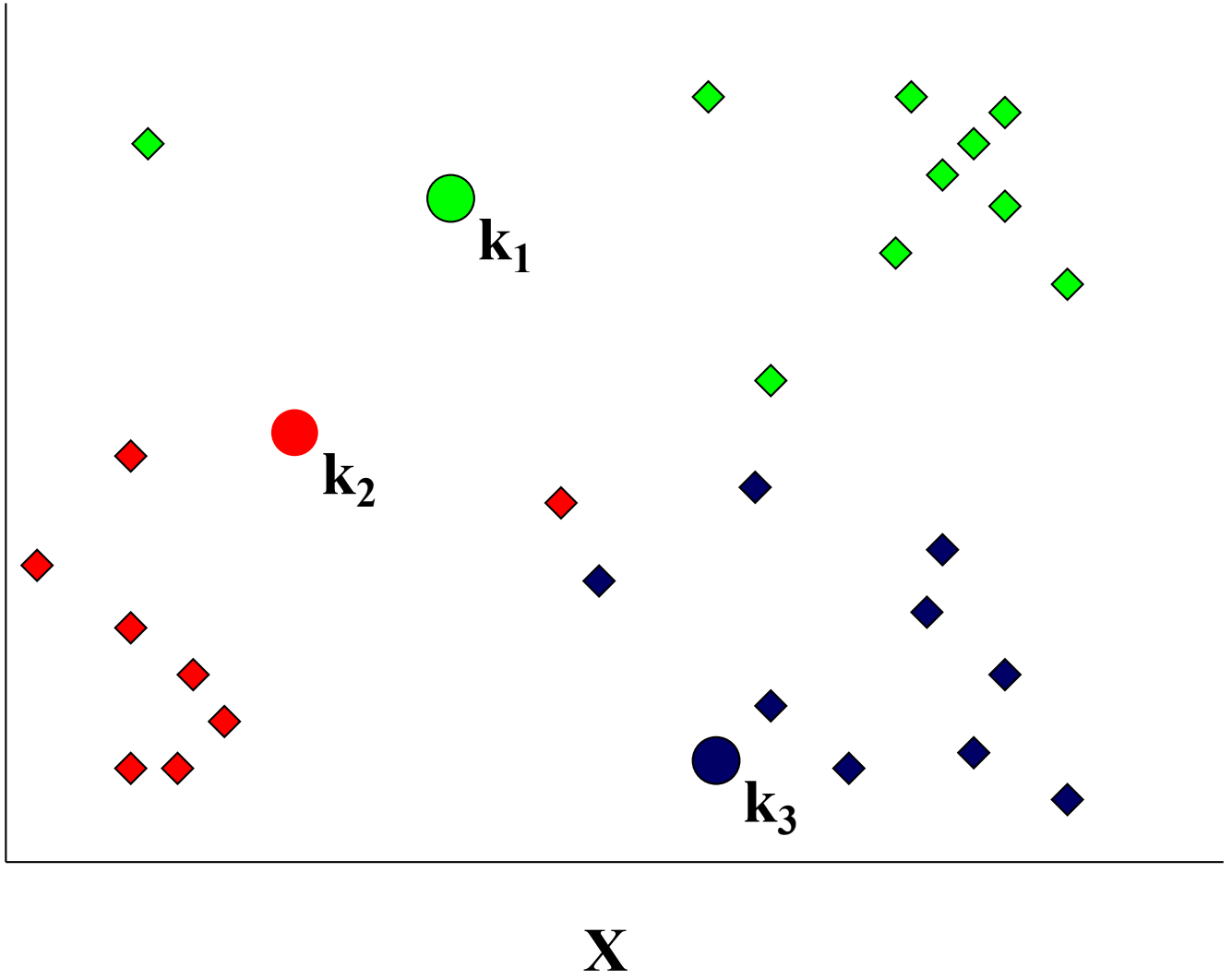
## K-Means example



# ML Algorithms

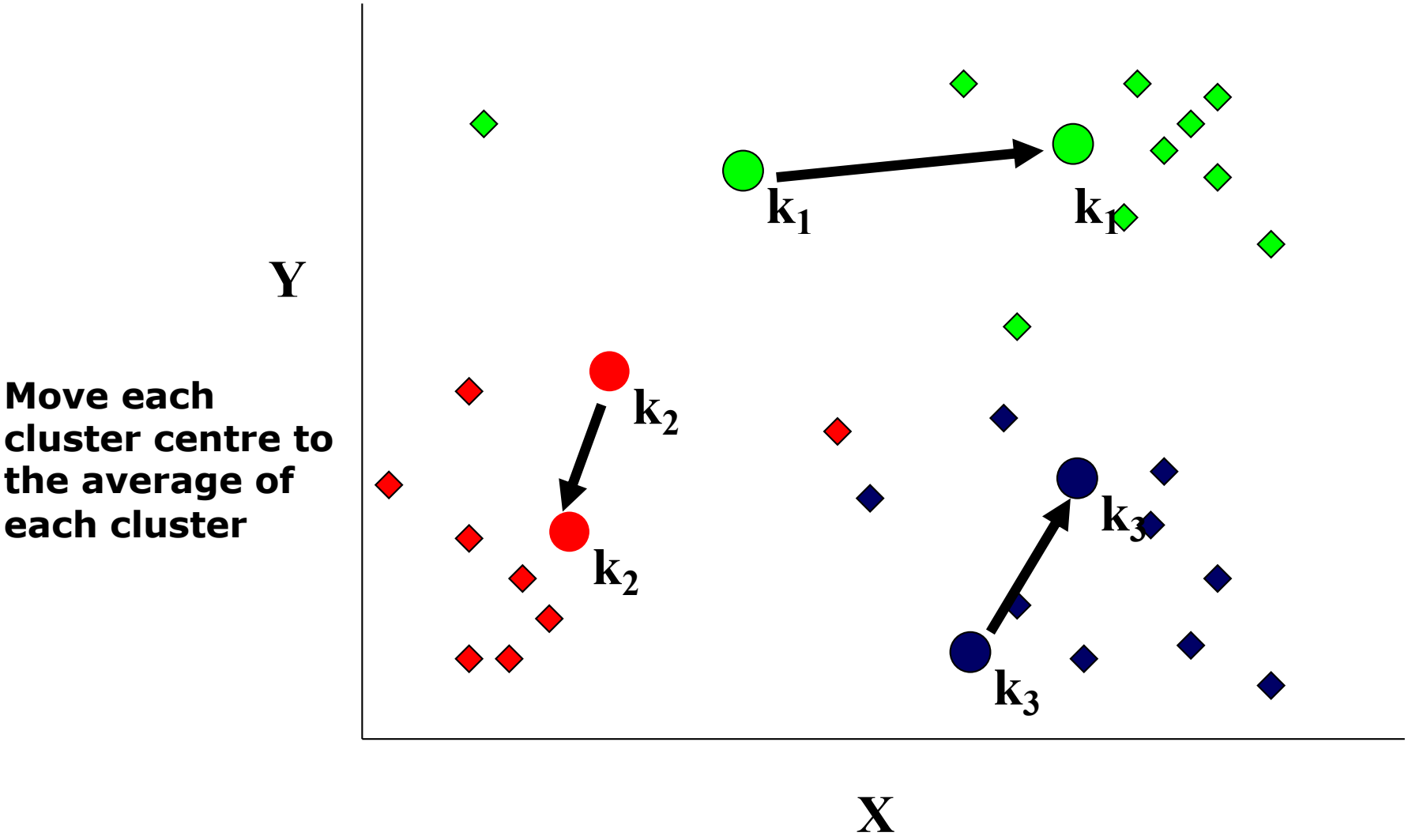
## K-Means example

**Assign each point to the nearest cluster centre**



# ML Algorithms

## K-Means example

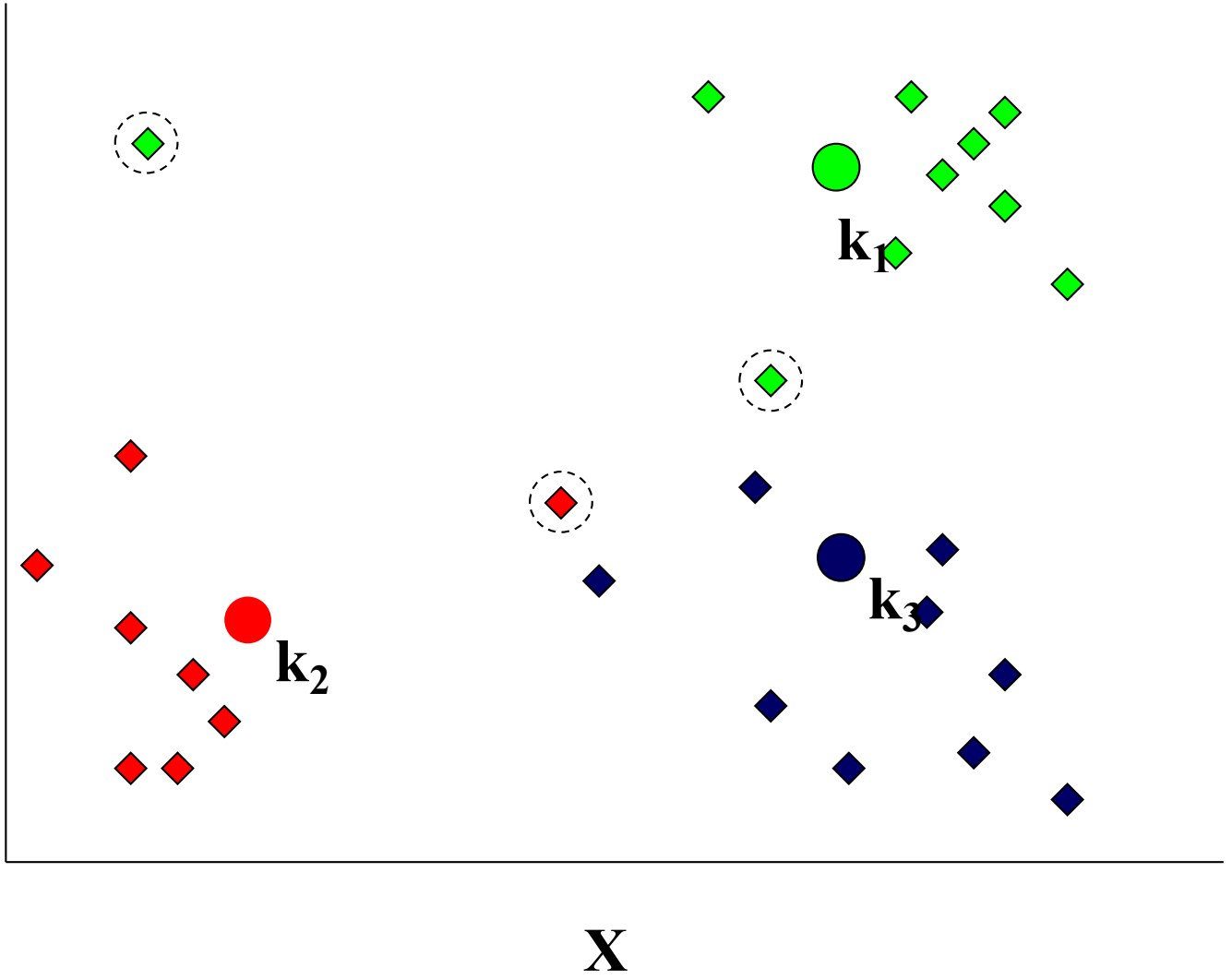


# ML Algorithms

## K-Means example

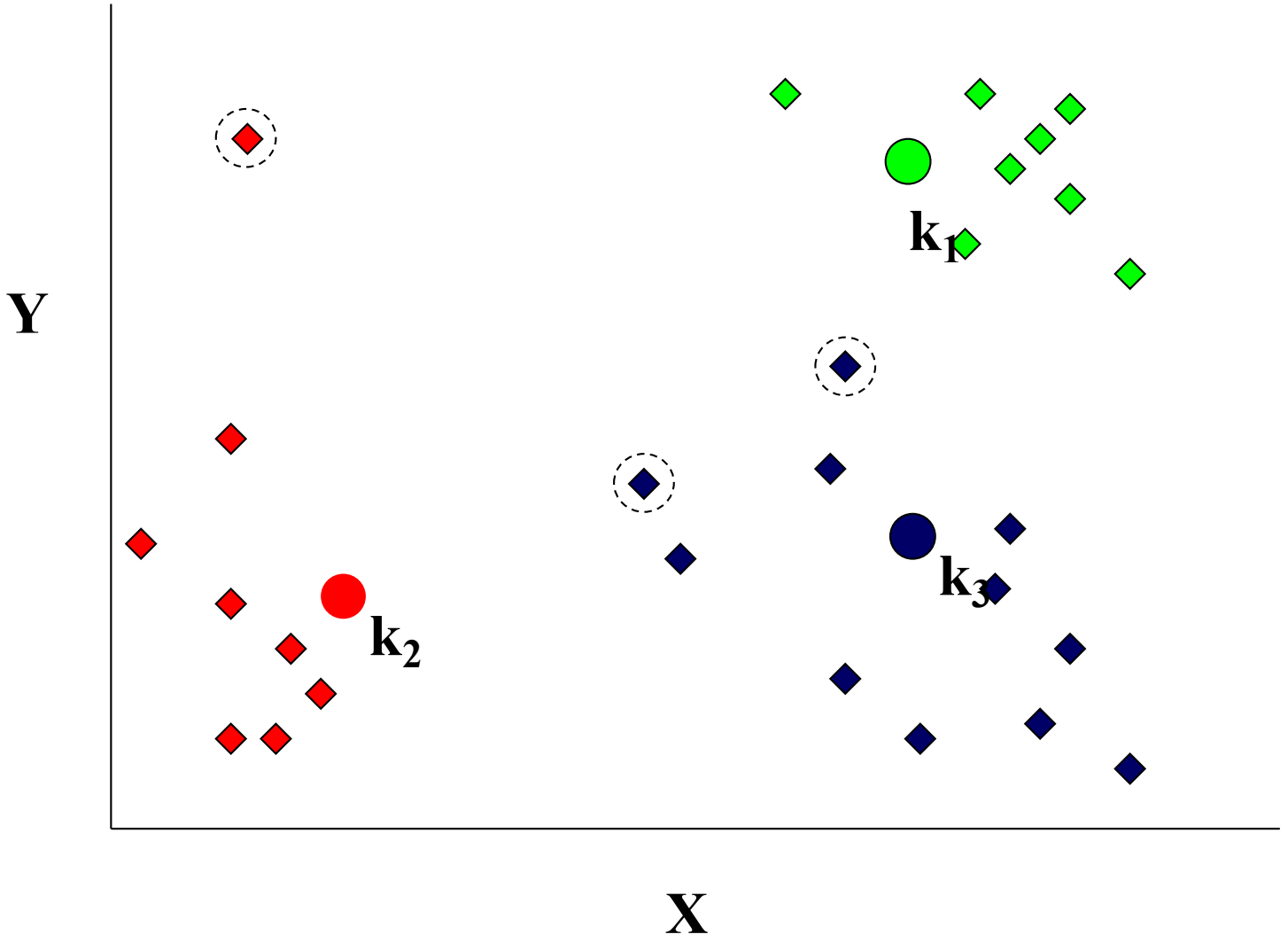
**Reassign the points closest to different cluster centres**

**What points are reassigned?**



# ML Algorithms

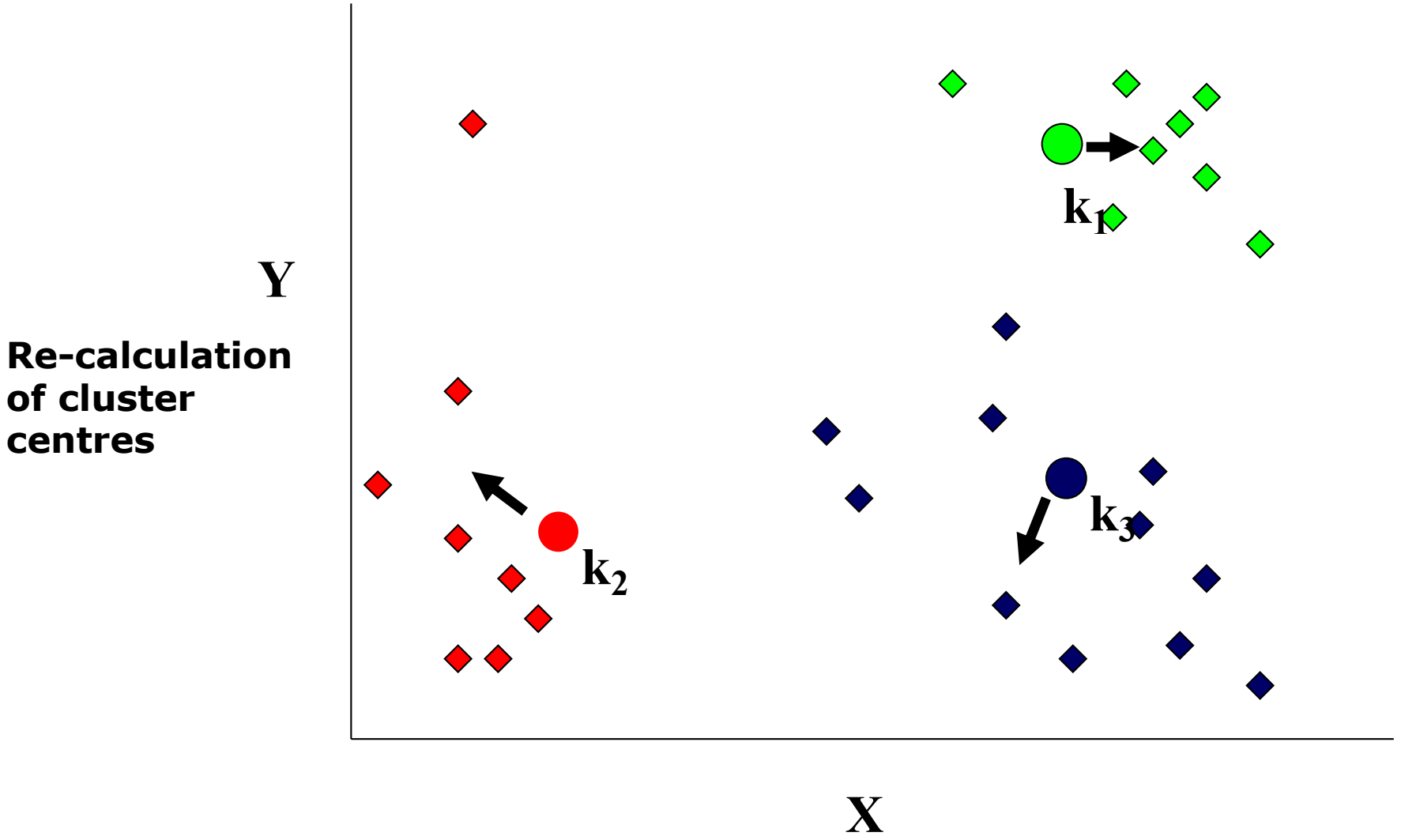
## K-Means example




**The three points with movement**

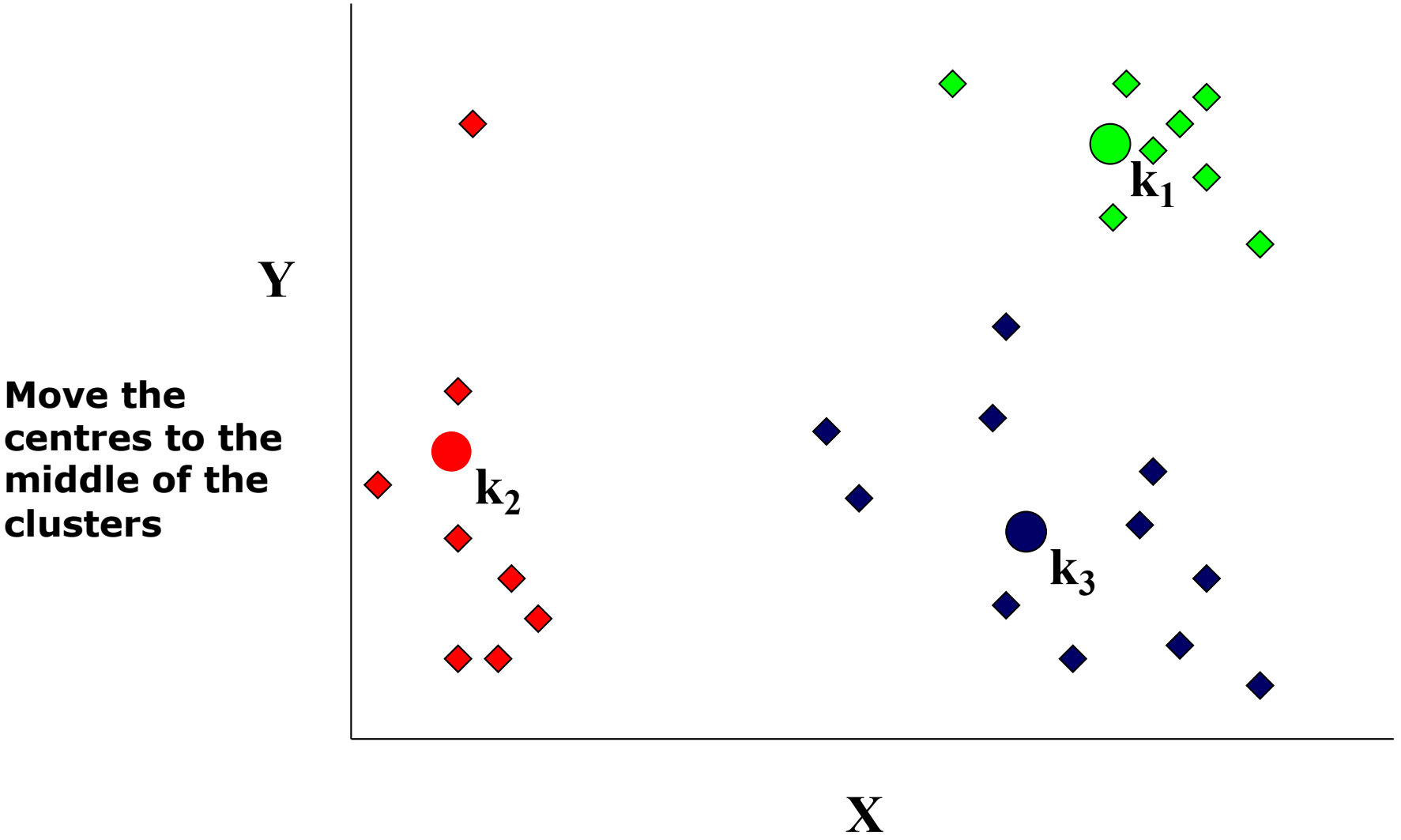
# ML Algorithms

## K-Means example



# ML Algorithms

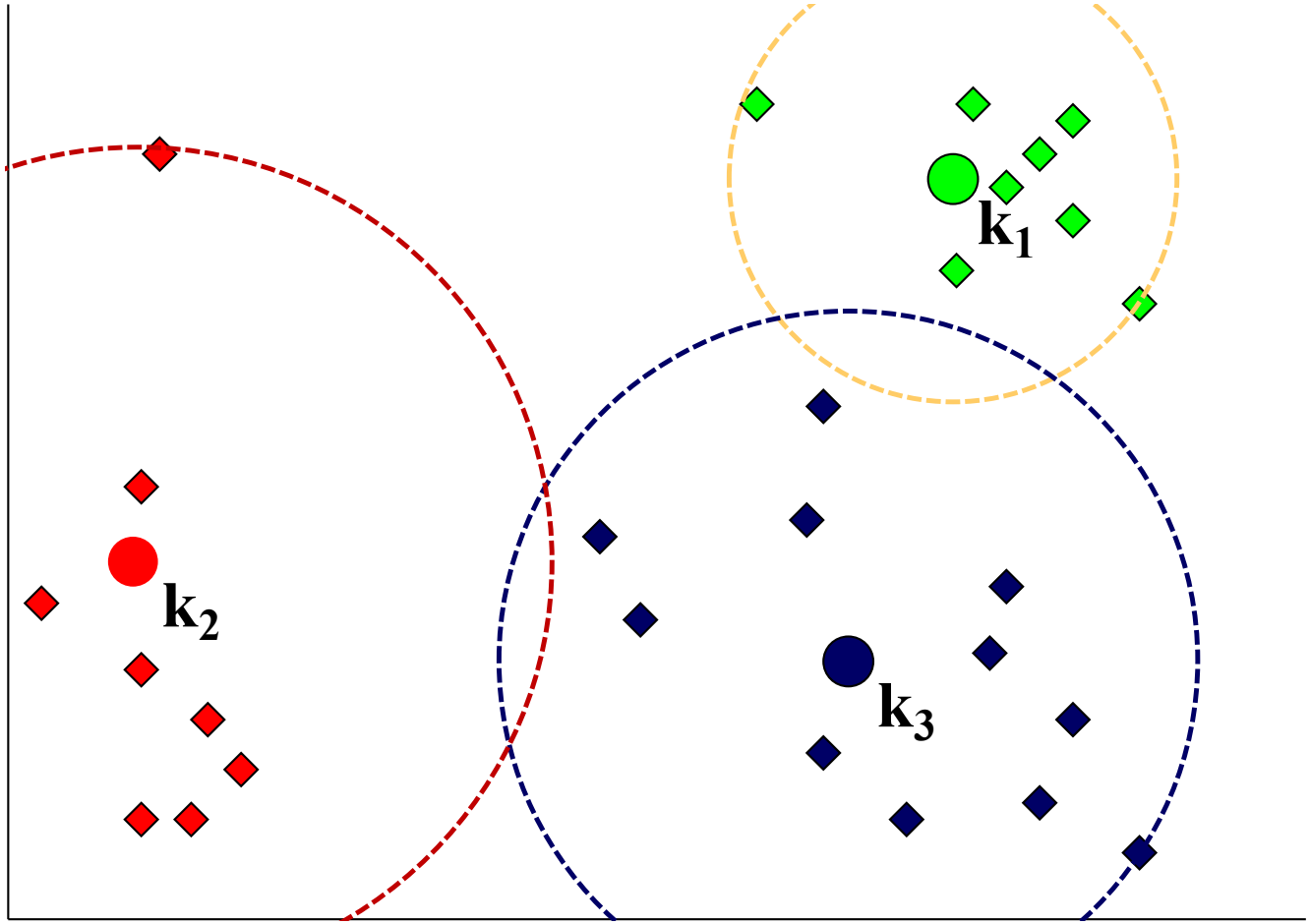
## K-Means example



# ML Algorithms

## K-Means example

**After moving the centres, all the points still belong to the same clusters, so the process ends**

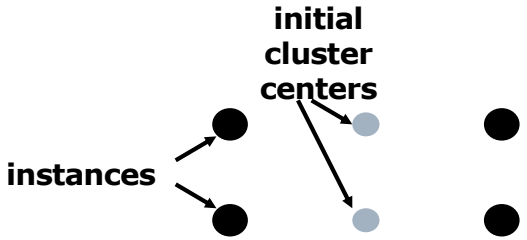


# ML Algorithms

## Some comments on K-Means

Advantages

- *Relatively efficient*: *O*(*tkn*), where *n* is # object, *k* is # *clusters*, and *t* *is* # iterations. Usually, *k*, *t* << *n*.
- It often ends up in a local optimal, depending on the initial choice of cluster centres.




  - Reset the seeds
  - Use more powerful search techniques such as genetic algorithms or stochastic cooling

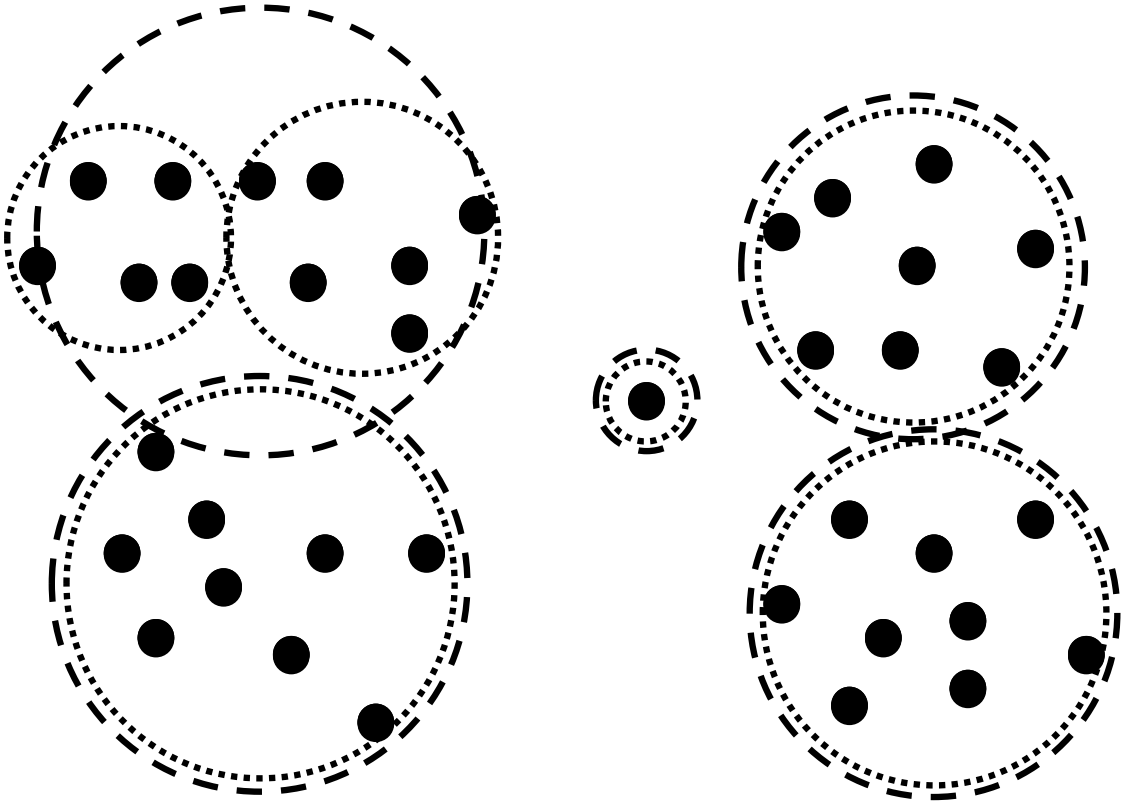
Disadvantages

- It is only applicable when the concept of average is definable. What to do with nominal data?
- Need to set the number of clusters in advance (k)
- Weak in the face of noisy data and/or outliers
- Only suitable for convex clusters (spherical...)

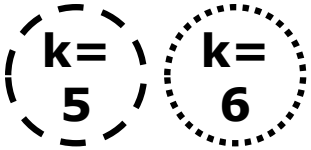
# ML Algorithms

## Some comments on K-Means

- Need to set the number of clusters in advance (k)



**¿choice of k?**

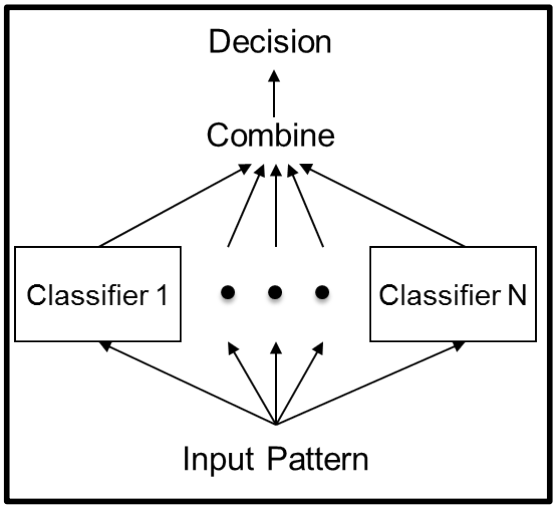



**One option is to iterate with different k-values and choose the best solution based on some measure of performance**

# ML Algorithms

## Rationale for Ensemble Learning

- No Free Lunch theorem: There is no algorithm that is always the most accurate
- Generate a group of base-learners which when combined have higher accuracy
- **Diversity** and **Accuracy** among classifiers are the key points for the success of ensembles.
- Different learners use different
  - Algorithms / Parameters
  - Training sets / Subproblems



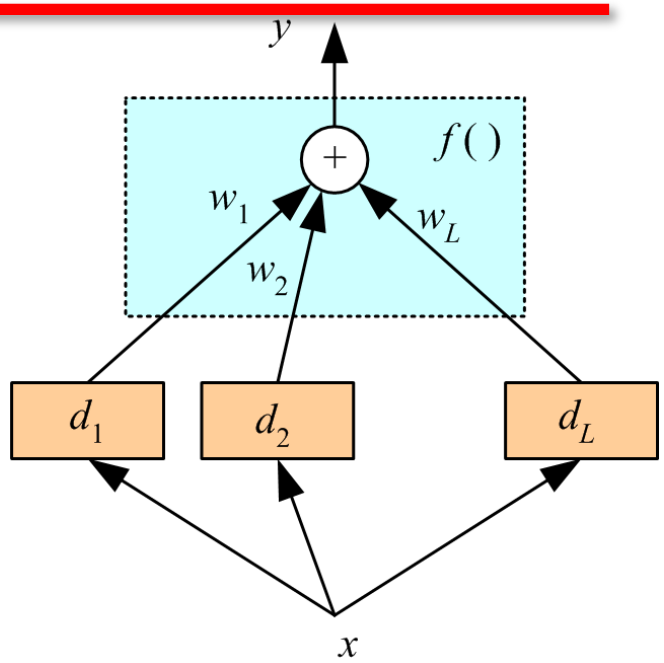
# ML Algorithms

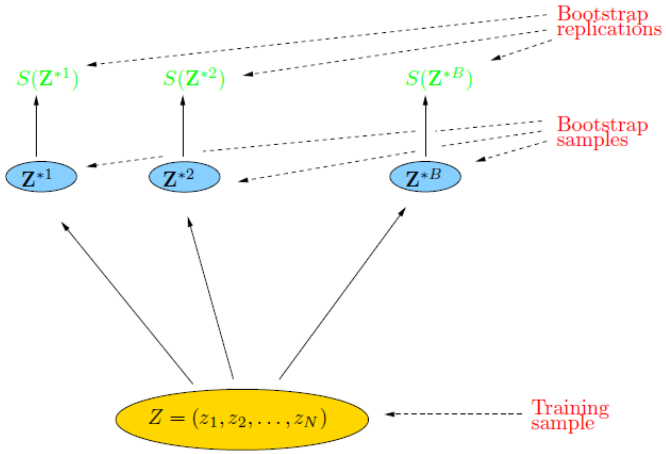
## Voting & Bootstrap

- Linear combination

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$

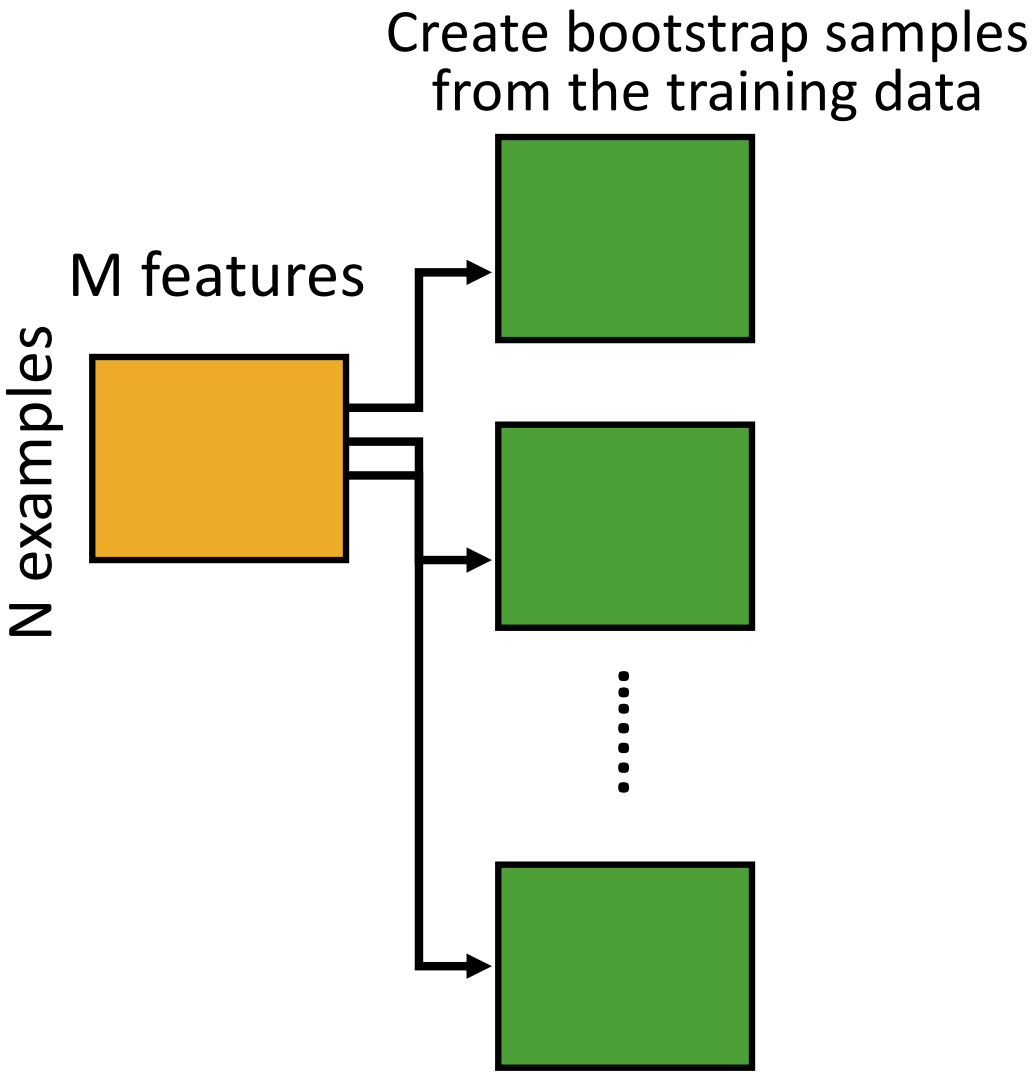



Bootstrap: randomly draw datasets *with replacement* from the training data, each sample *the same size as the original training set*

# ML Algorithms

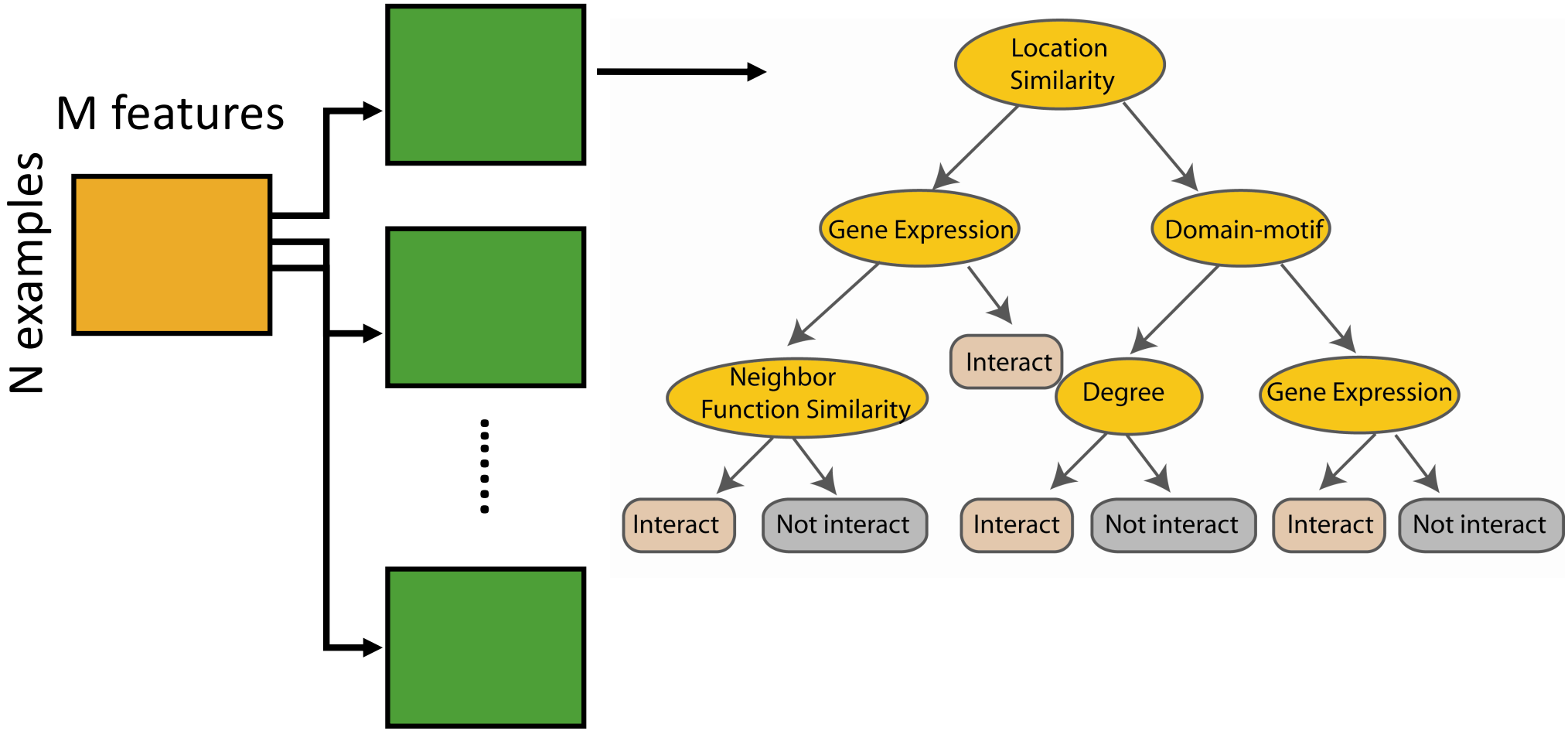
## Bagging



# ML Algorithms

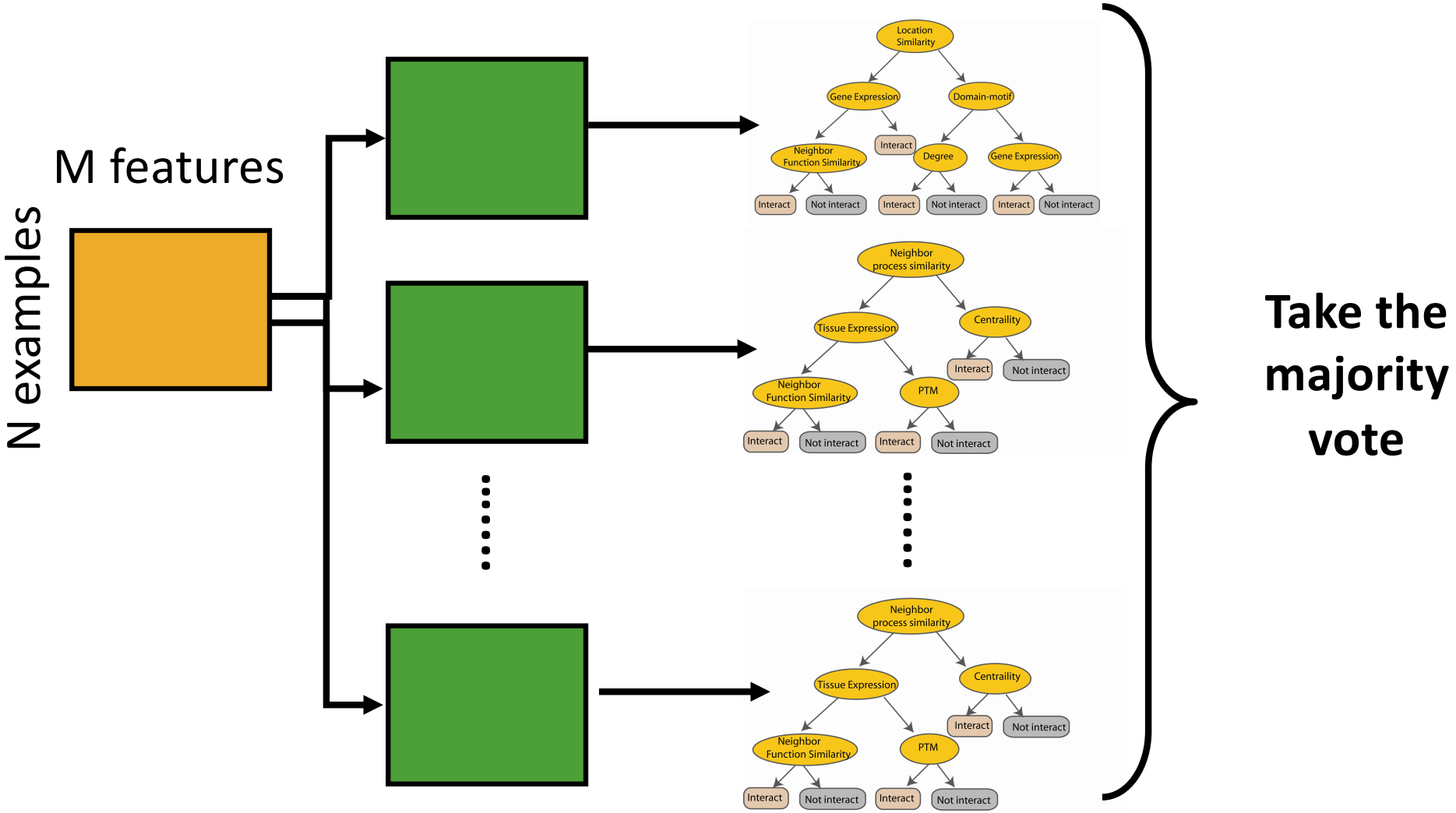
## Random Forest Classifier

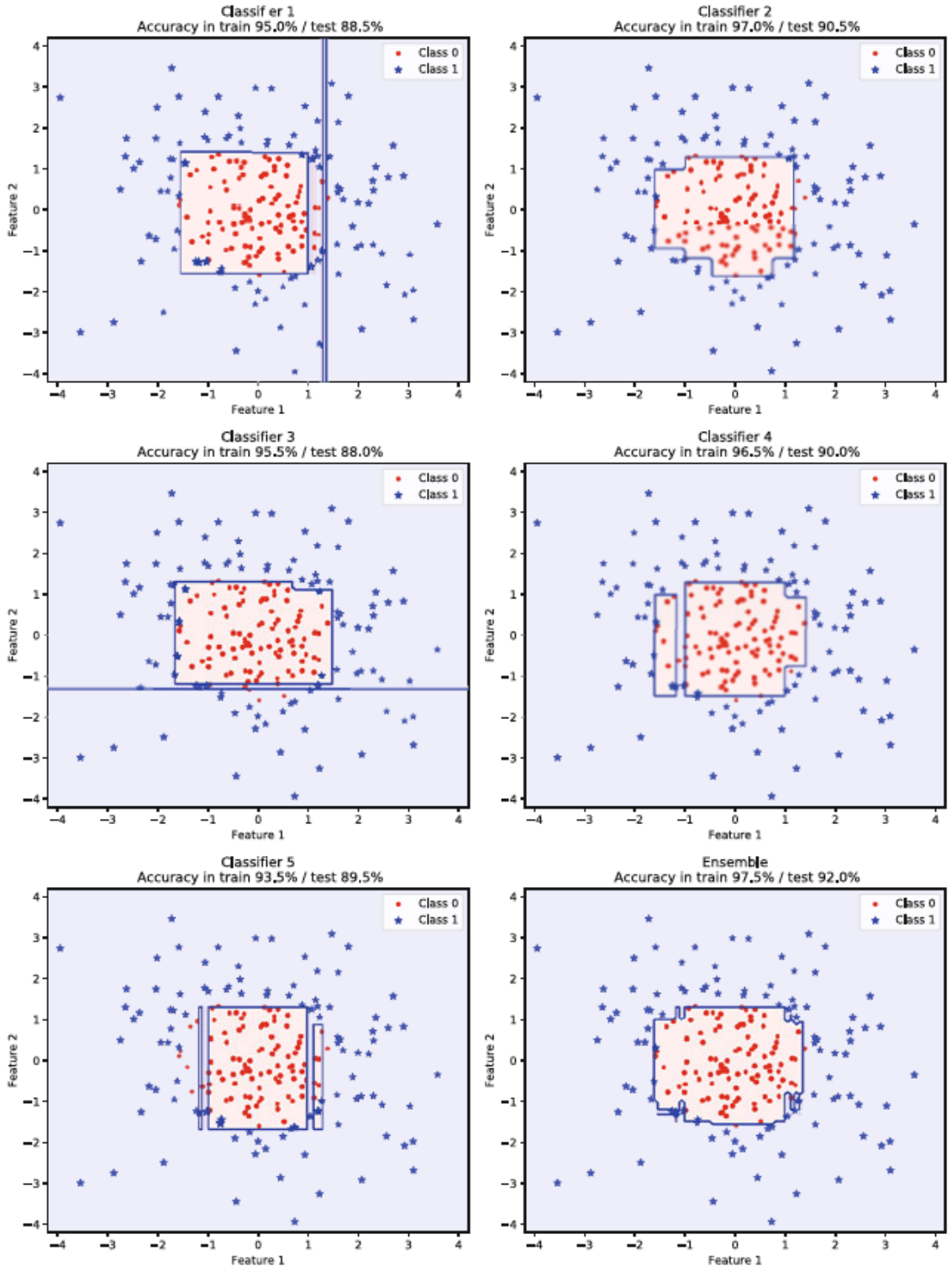
Construct a decision tree



# ML Algorithms

## Random Forest Classifier




Classif er 1
Accuracy in train 95.0% / test 88.5%
Classifier 2
Accuracy in train 97.0% / test 90.5%
Classifier 3
Accuracy in train 95.5% / test 88.0%
Classifier 4
Accuracy in train 96.5% / test 90.0%
Classifier 5
Accuracy in train 93.5% / test 89.5%
Ensemble
Accuracy in train 97.5% / test 92.0%
Class 0
Class 1
Feature 1
Feature 2

# ML Algorithms

## Boosting

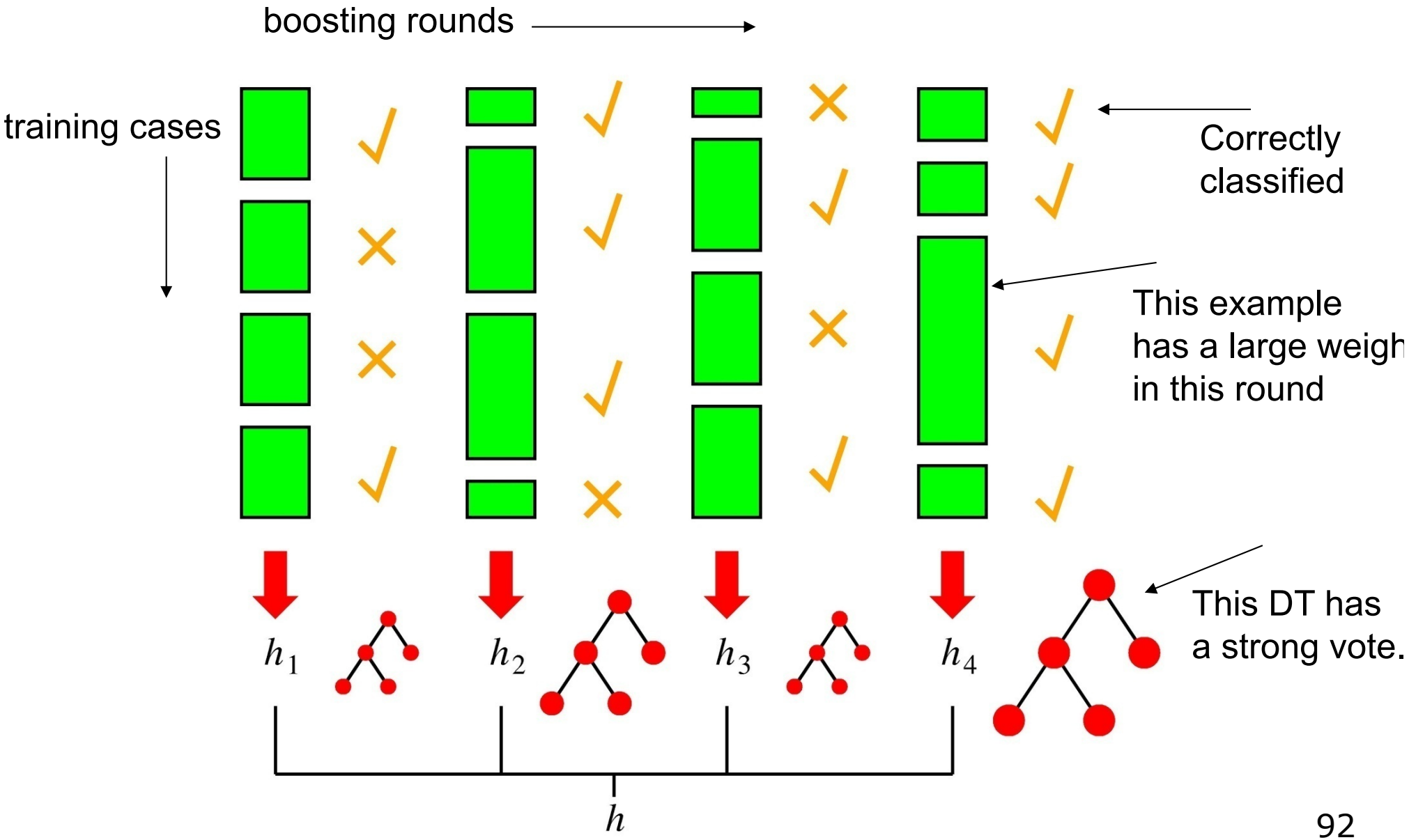
- Train classifiers (e.g. decision trees) in a sequence.
- A new classifier should focus on those cases which were incorrectly classified in the last round.
- Combine the classifiers by letting them vote on the final prediction (like bagging).
- Each classifier is “weak” but the ensemble is “strong.”
- AdaBoost is a specific boosting method.

# ML Algorithms
## Boosting Intuition

- We adaptively weight each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them)
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that obtain low training error rate have high weight.

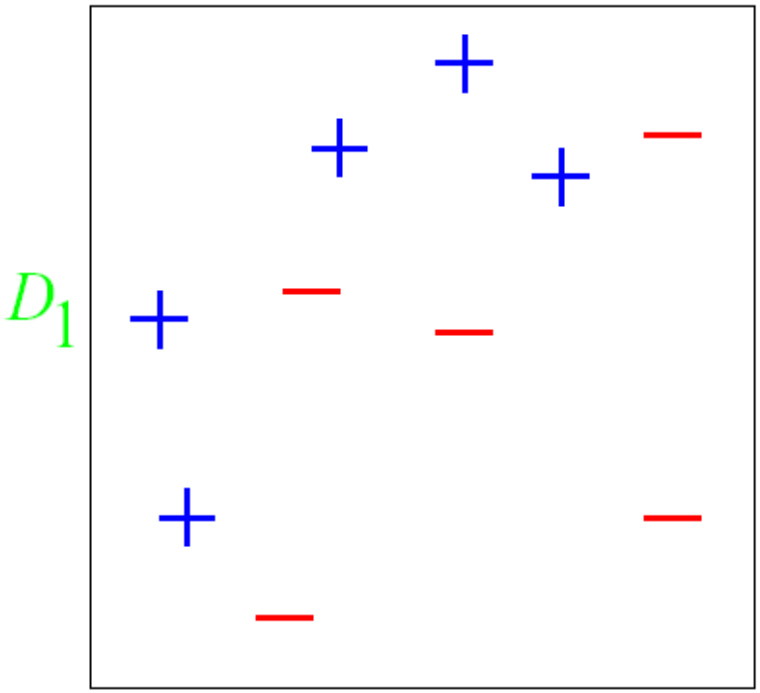
# ML Algorithms

## Boosting in a Picture



# ML Algorithms

## Boosting in animation




Original training set: equal weights to all training samples

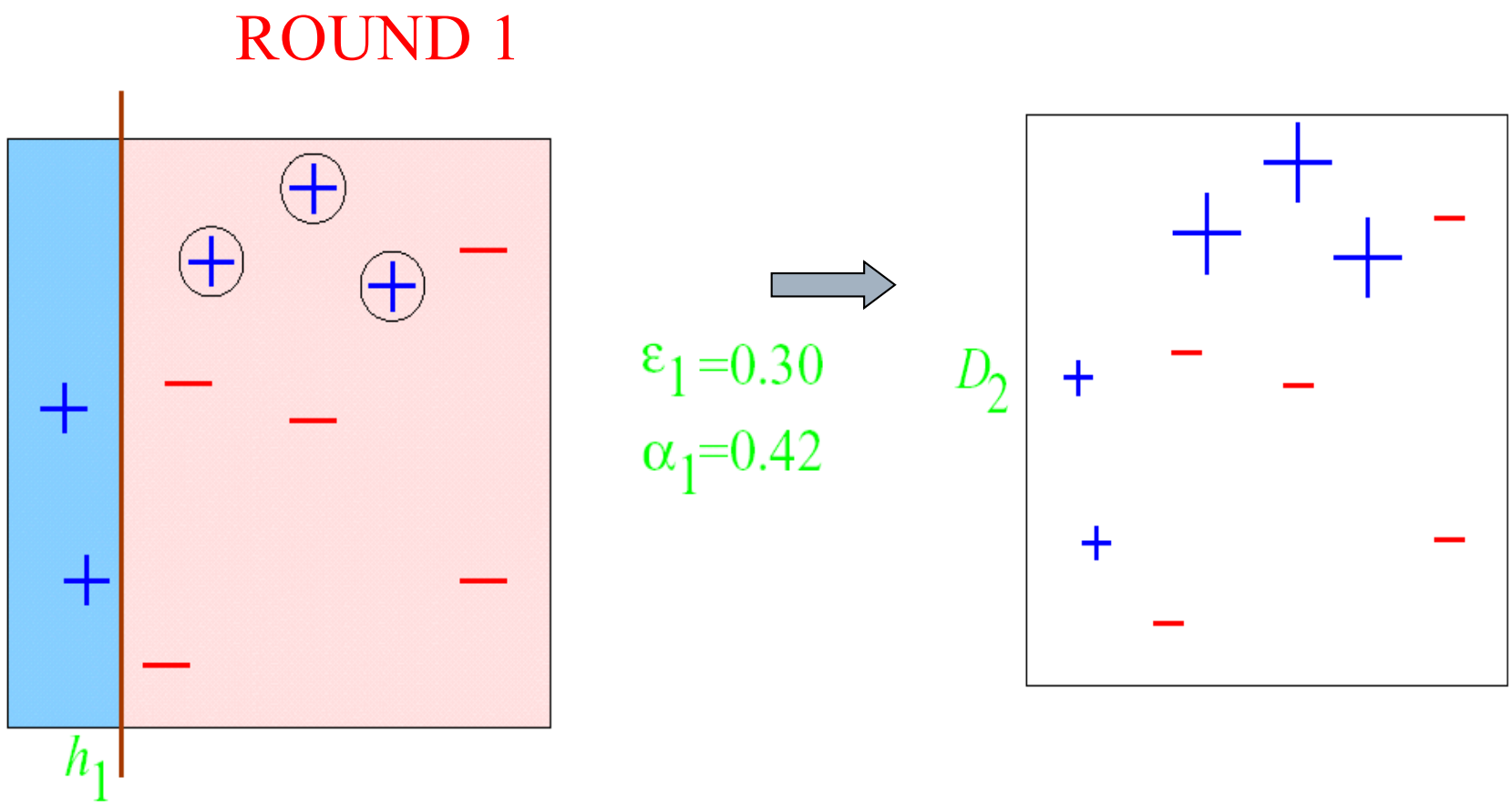
Taken from "**A Tutorial on Boosting**" **by** Yoav Freund and Rob Schapire

# ML Algorithms

## AdaBoost example

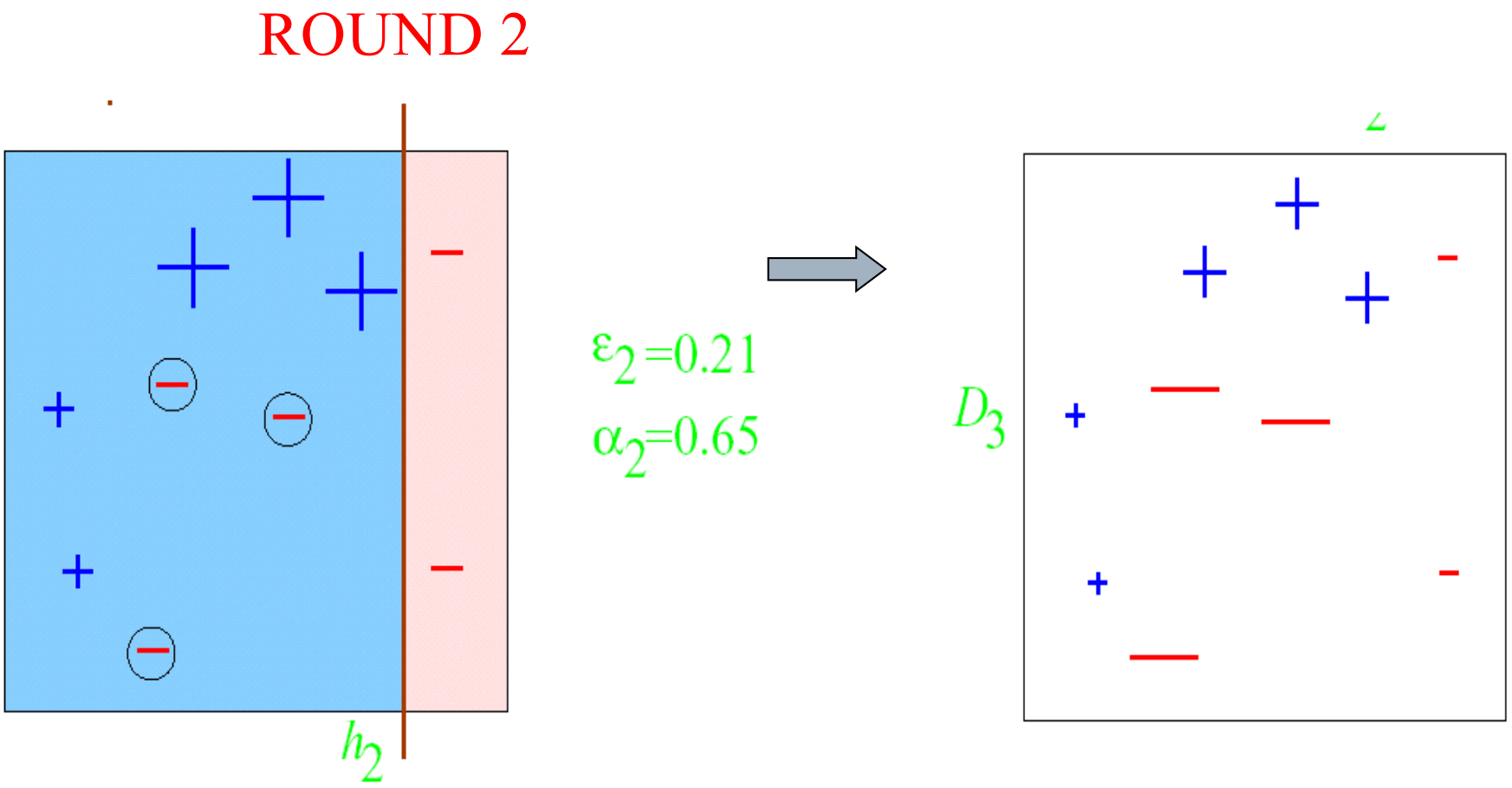
ε = error rate of classifier

α = weight of classifier



# ML Algorithms

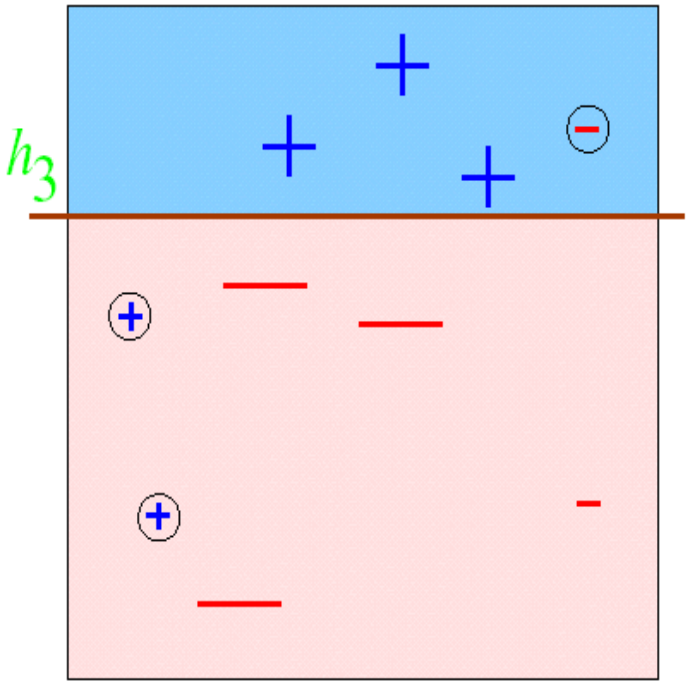
## AdaBoost example



# ML Algorithms

## AdaBoost example

ROUND 3



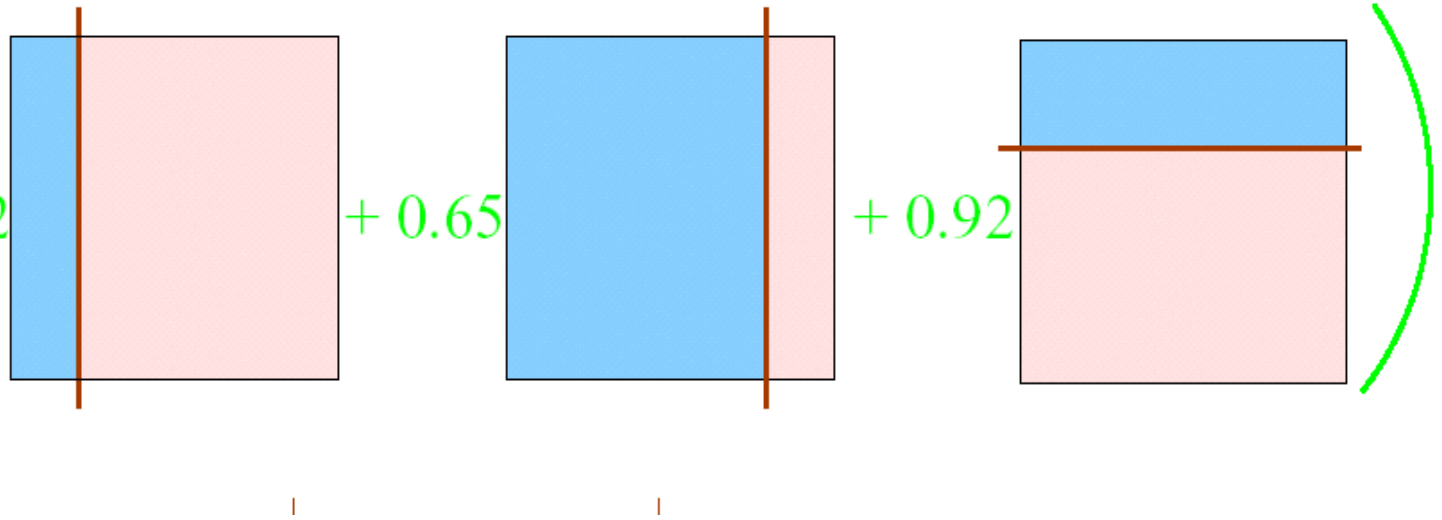

$\varepsilon_3 = 0.14$

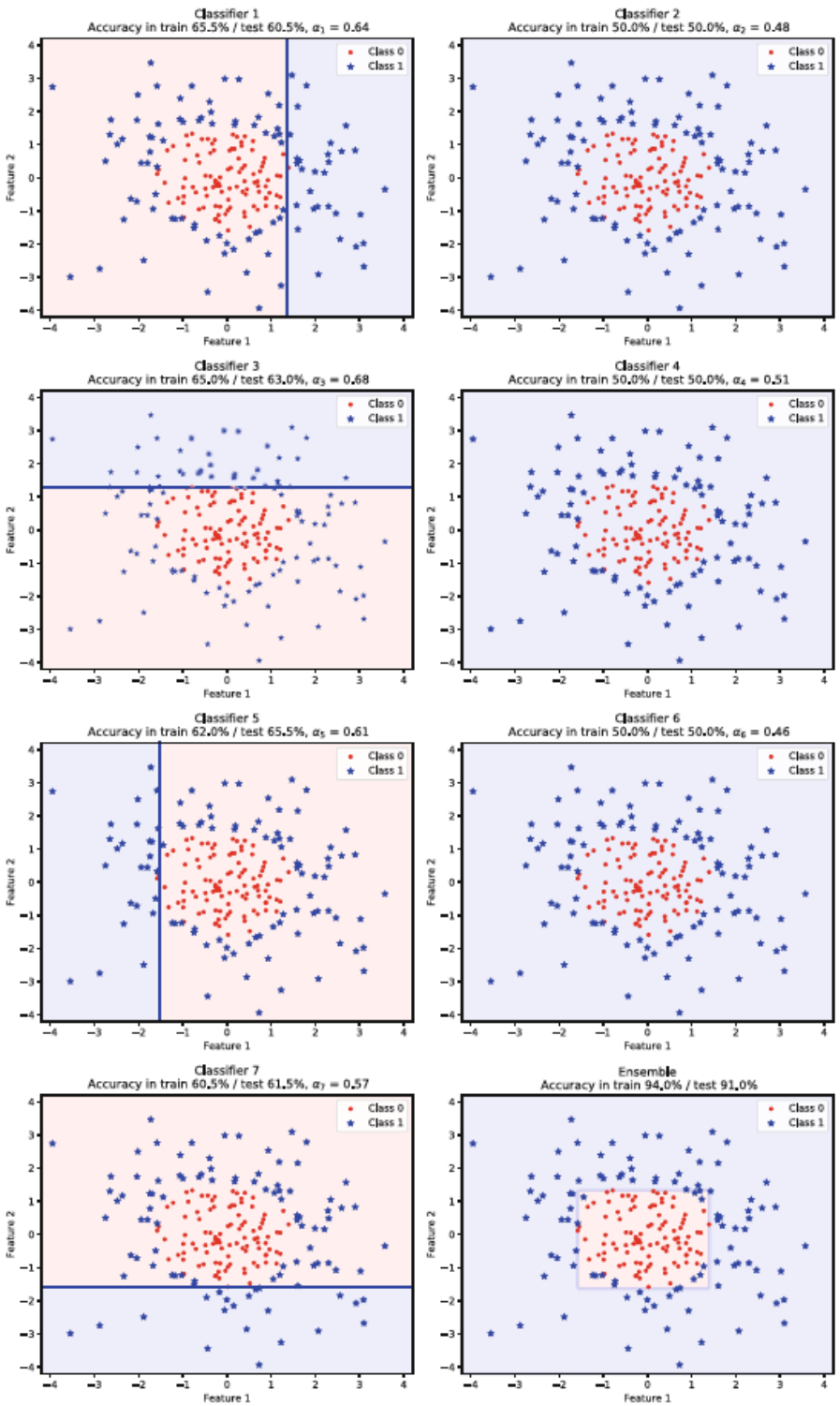
$\alpha_3 = 0.92$

# ML Algorithms

## AdaBoost example

$$H_{\text{final}} = \text{sign}\left(0.42 + 0.65 + 0.92\right)$$




Classifier 1
Accuracy in train 65.5% / test 60.5%, $\alpha_1 = 0.64$
Classifier 2
Accuracy in train 50.0% / test 50.0%, $\alpha_2 = 0.48$
Classifier 3
Accuracy in train 65.0% / test 63.0%, $\alpha_3 = 0.68$
Classifier 4
Accuracy in train 50.0% / test 50.0%, $\alpha_4 = 0.51$
Classifier 5
Accuracy in train 62.0% / test 65.5%, $\alpha_5 = 0.61$
Classifier 6
Accuracy in train 50.0% / test 50.0%, $\alpha_6 = 0.46$
Classifier 7
Accuracy in train 60.5% / test 61.5%, $\alpha_7 = 0.57$
Ensemble
Accuracy in train 94.0% / test 91.0%
Class 0
Class 1
Feature 1
Feature 2

# Evaluation and Validation

## Metrics

**Confusion Matrix**

| | | Prediction | |
|---|---|---|---|
| | | $C_P$ | $C_N$ |
| Actual class | $C_P$ | **TP**: True positive | **FN**: False negative |
| | $C_N$ | **FP**: False positive | **TN**: True negative |

**accuracy = (TP+TN)/(TP+TN+FP+FN)**

**Limitations of accuracy:**

Let's assume a problem with 2 classes:

- 9990 examples of class 1
- 10 examples of class 2

If the classification model always says that the examples are class 1, its accuracy is

9990/10000 = 99.9%

Totally misleading, as we will never detect any example of class 2

# Evaluation and Validation

## Metrics

**Alternative: Cost Matrix**

| C(i\|j) | | Prediction | |
|---|---|---|---|
| | | **$C_P$** | **$C_N$** |
| **Actual Class** | **$C_P$** | **C(P\|P)** | **C(N\|P)** |
| | **$C_P$** | **C(P\|N)** | **C(N\|N)** |

The cost of classification will be proportional to the accuracy of the classifier only if

$$\forall i,j: i \neq j \quad C(i|j) = C(j|i)$$

$$C(i|i) = C(j|j)$$

**cost-sensitive metrics**

| | | Prediction | |
|---|---|---|---|
| | | **$C_P$** | **$C_N$** |
| **Actual Class** | **$C_P$** | **TP**: True positive | **FN**: False negative |
| | **$C_N$** | **FP**: False positive | **TN**: True negative |

**precision = TP/(TP+FP)**

**True positive recognition rate**

**recall = sensitivity = TP/P = TP/(TP+FN)**

**True negative recognition rate**

**specificity = TN/N = TN/(TN+FP)**

**F-measure**

**F = 2*precision*recall / (precision+recall)**

# Evaluation and Validation

## Metrics

| | | Prediction | |
|---|---|---|---|
| | | $C_P$ | $C_N$ |
| Real | $C_P$ | TP | FN |
| | $C_N$ | FP | TN |

**Accuracy**

| | | Prediction | |
|---|---|---|---|
| | | $C_P$ | $C_N$ |
| Real | $C_P$ | TP | FN |
| | $C_N$ | FP | TN |

**Recall**

| | | Prediction | |
|---|---|---|---|
| | | $C_P$ | $C_N$ |
| Real | $C_P$ | TP | FN |
| | $C_N$ | FP | TN |

**Precision**

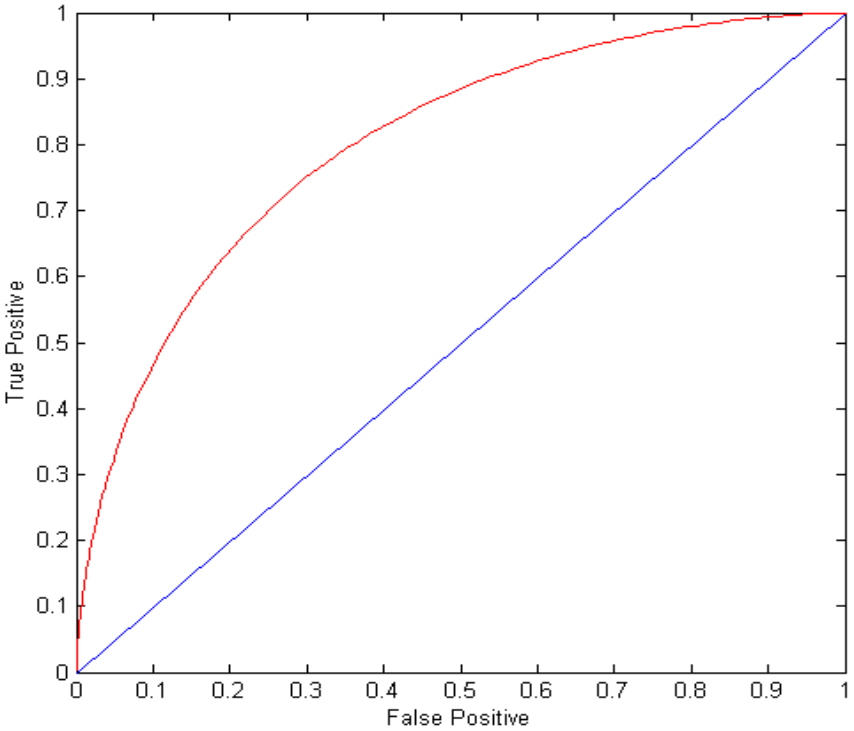
| | | Prediction | |
|---|---|---|---|
| | | $C_P$ | $C_N$ |
| Real | $C_P$ | TP | FN |
| | $C_N$ | FP | TN |

**F-measure**

# Evaluation and Validation

## Metrics

**ROC Curves**
**(Receiver Operating Characteristics)**



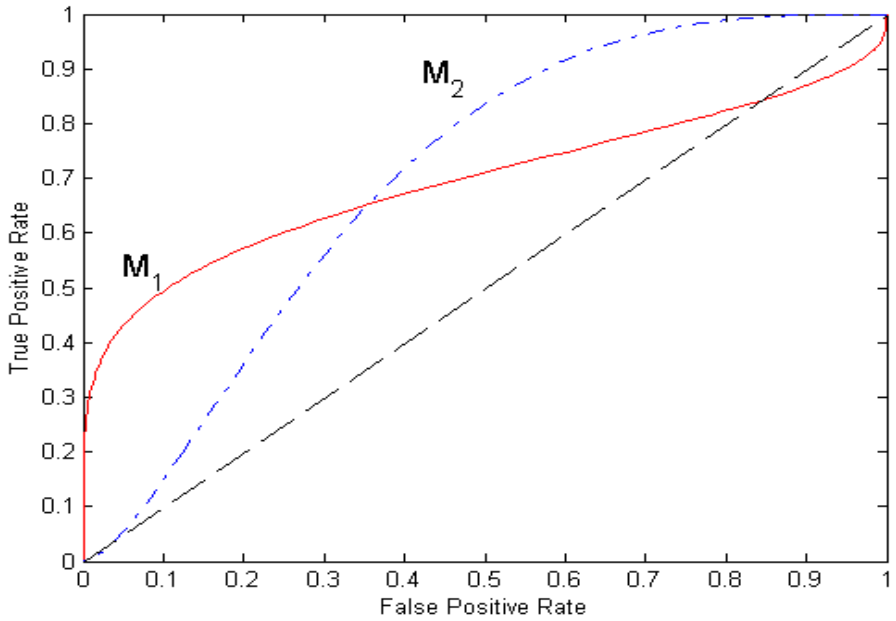

- Developed in the 1950s to analyse signals with noise: characterising the trade-off between right and wrong alarms.
- They allow us to visually compare different classification models.
- The area under the curve is a measure of the accuracy of the classifier:
  - The closer we are to the diagonal (area close to 0.5), the less accurate the model will be.
  - A "perfect" model will have area 1.

**Vertical axis:**
**"true positive rate" TPR = TP/(TP+FN)**
**Horizontal axis:**
**"false positive rate" FPR = FP/(FP+TN)**

# Evaluation and Validation

## Metrics

**ROC Curves**




**Neither model is consistently better than the other: M1 is better for low FPR, M2 for high FPR.**

# Evaluation and Validation

## Metrics

**ROC Curves**

**How to build the ROC curve?**

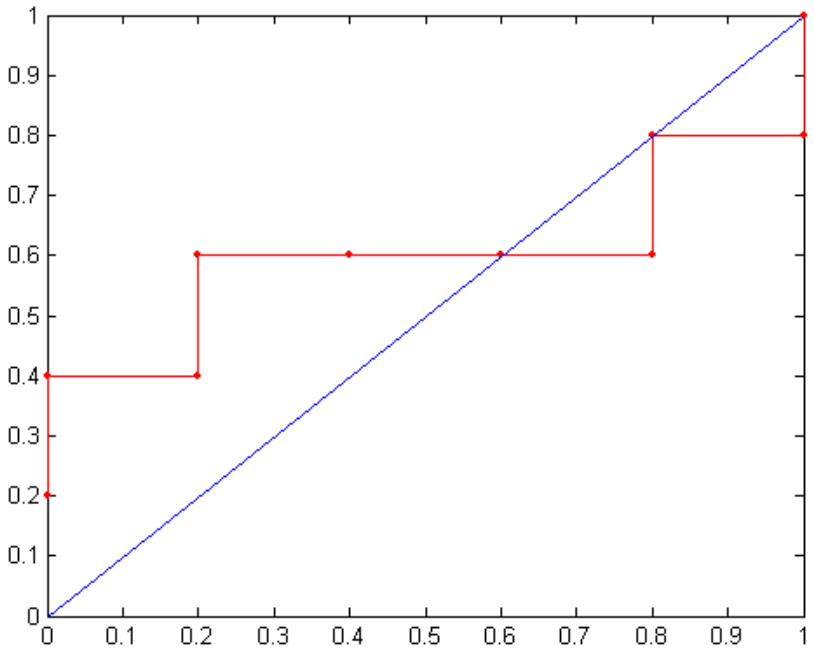
- **A classifier is used that predicts the probability of an E example belonging to the positive class P(+|E)**
- **The examples are arranged in decreasing order of estimated value P(+|E)**
- **A threshold is applied for each different value of P(+|E), where the number of TP, FP, TN y FN are counted.**

$$TPR = TP/(TP+FN)$$
$$FPR = FP/(FP+TN)$$

# Evaluation and Validation

## Metrics

**ROC Curves**




| Class | + | - | + | - | - | - | + | - | + | + | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.25 | 0.43 | 0.53 | 0.76 | 0.85 | 0.85 | 0.85 | 0.87 | 0.93 | 0.95 | 1.00 |
| TP | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 0 |
| FP | 5 | 5 | 4 | 4 | 3 | 2 | 1 | 1 | 0 | 0 | 0 |
| TN | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 4 | 5 | 5 | 5 |
| FN | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 5 |
| TPR | 1 | 0.8 | 0.8 | 0.6 | 0.6 | 0.6 | 0.6 | 0.4 | 0.4 | 0.2 | 0 |
| FPR | 1 | 1 | 0.8 | 0.8 | 0.6 | 0.4 | 0.2 | 0.2 | 0 | 0 | 0 |

| Example | P(+\|E) | Class |
|---|---|---|
| 1 | 0.95 | + |
| 2 | 0.93 | + |
| 3 | 0.87 | - |
| 4 | 0.85 | - |
| 5 | 0.85 | - |
| 6 | 0.85 | + |
| 7 | 0.76 | - |
| 8 | 0.53 | + |
| 9 | 0.43 | - |
| 10 | 0.25 | + |

# Evaluation and Validation

## Validation in Classification and Regression

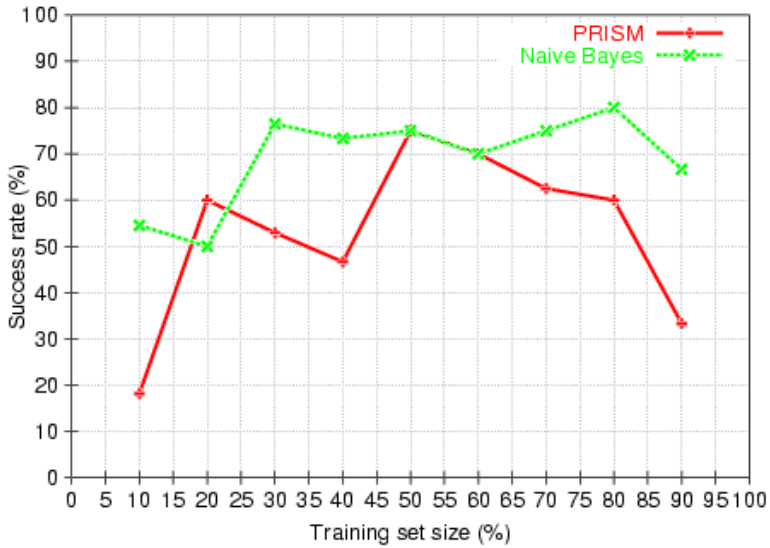
- The aim of the validation methods is to make an honest estimate of the goodness of the built learning algorithm
- Using the success rate on the training set as a good thing is unrealistic
  - The error obtained is often too optimistic because the model will be over-adjusted to the data used during the learning process
- There are various techniques for validating learning algorithms, including
  - *Hold-out*
  - *Cross-validation*

# Evaluation and Validation

## Validation in Classification and Regression

### Hold-out

- It consists of dividing the DB into two independent sets: training (TR) and test (TS)
- The size of TR is usually greater than TS: (2/3, 1/3, 4/5, 1/5,…)
- TR elements are usually obtained by sampling without replacement of the initial DB. The TS is made up of the elements not included in the TR
- Usually used in large data sets




Test set (%) + Training set (%) = 100%

# Evaluation and Validation

## Validation in Classification and Regression

### Cross-validation

- It consists of:
  1. Dividi the DB into *k* subsets (*folds*), $\{S_1,\ldots,S_k\}$ of equal size
  2. Learn *k* models by using a different TR in each of them. Check with the corresponding TS

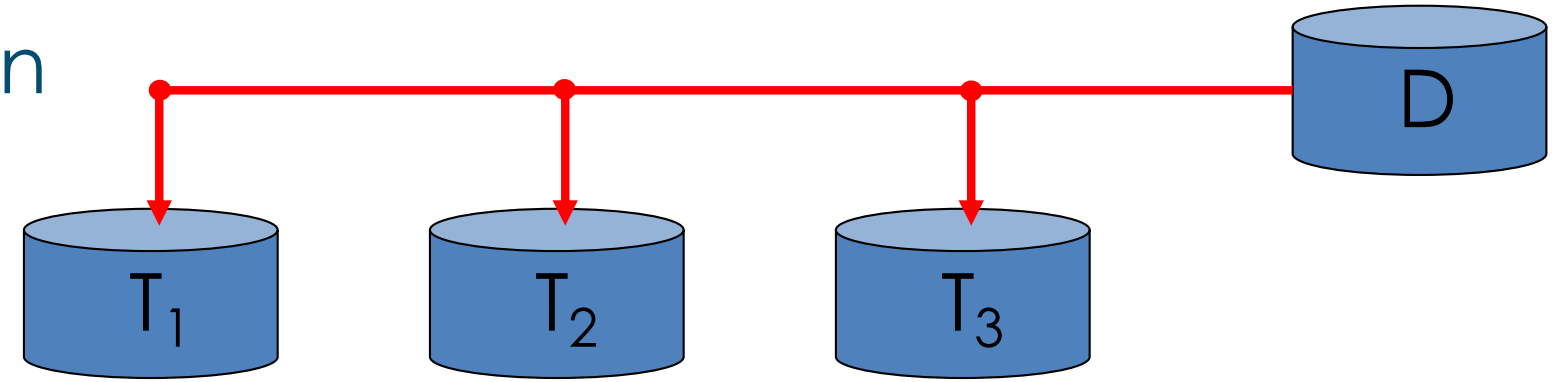
$$TR = S_1 \cup \ldots \cup S_{i-1} \cup S_{i+1} \cup \ldots S_k$$

$$TS = S_i$$

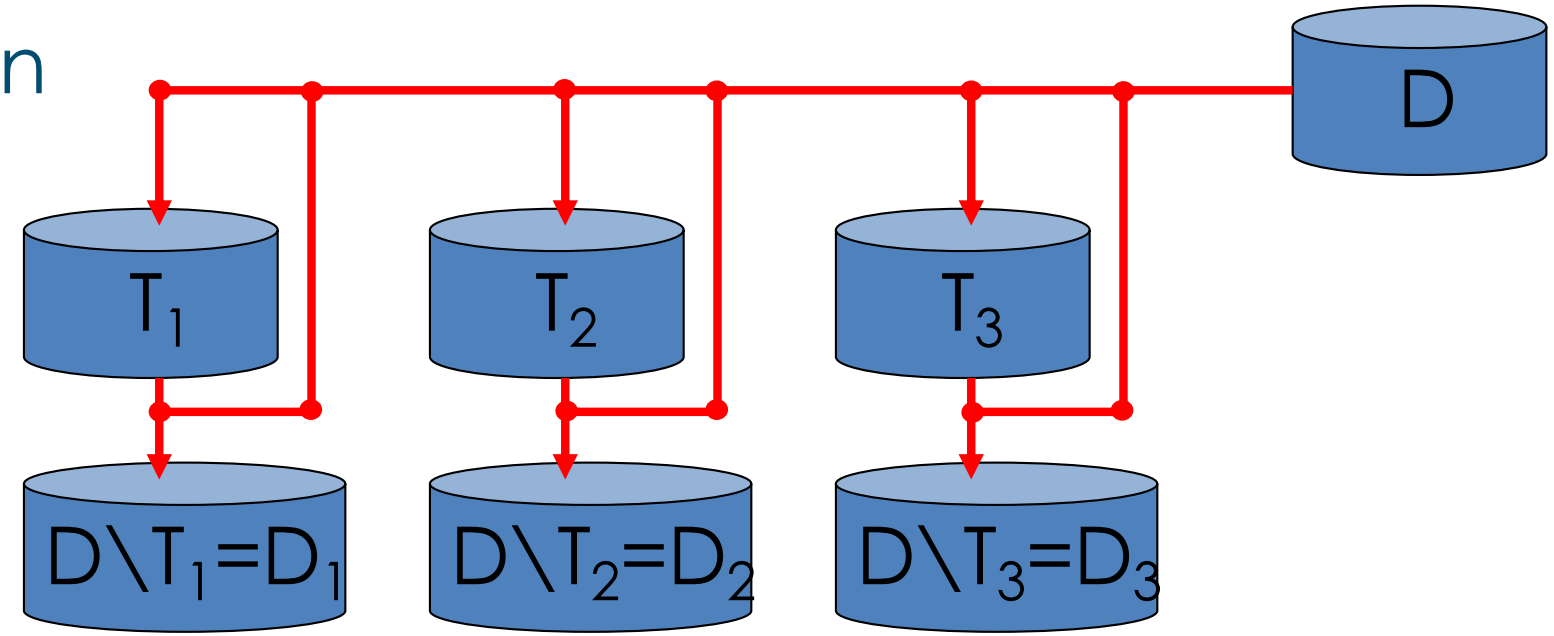
  3. Return as the error rate the mean obtained in the *k* iterations

- Stratified cross validation (in classification): Subsets are stratified according to the class variable
- Typical values of k=5,10
- Usually used in moderate sized databases

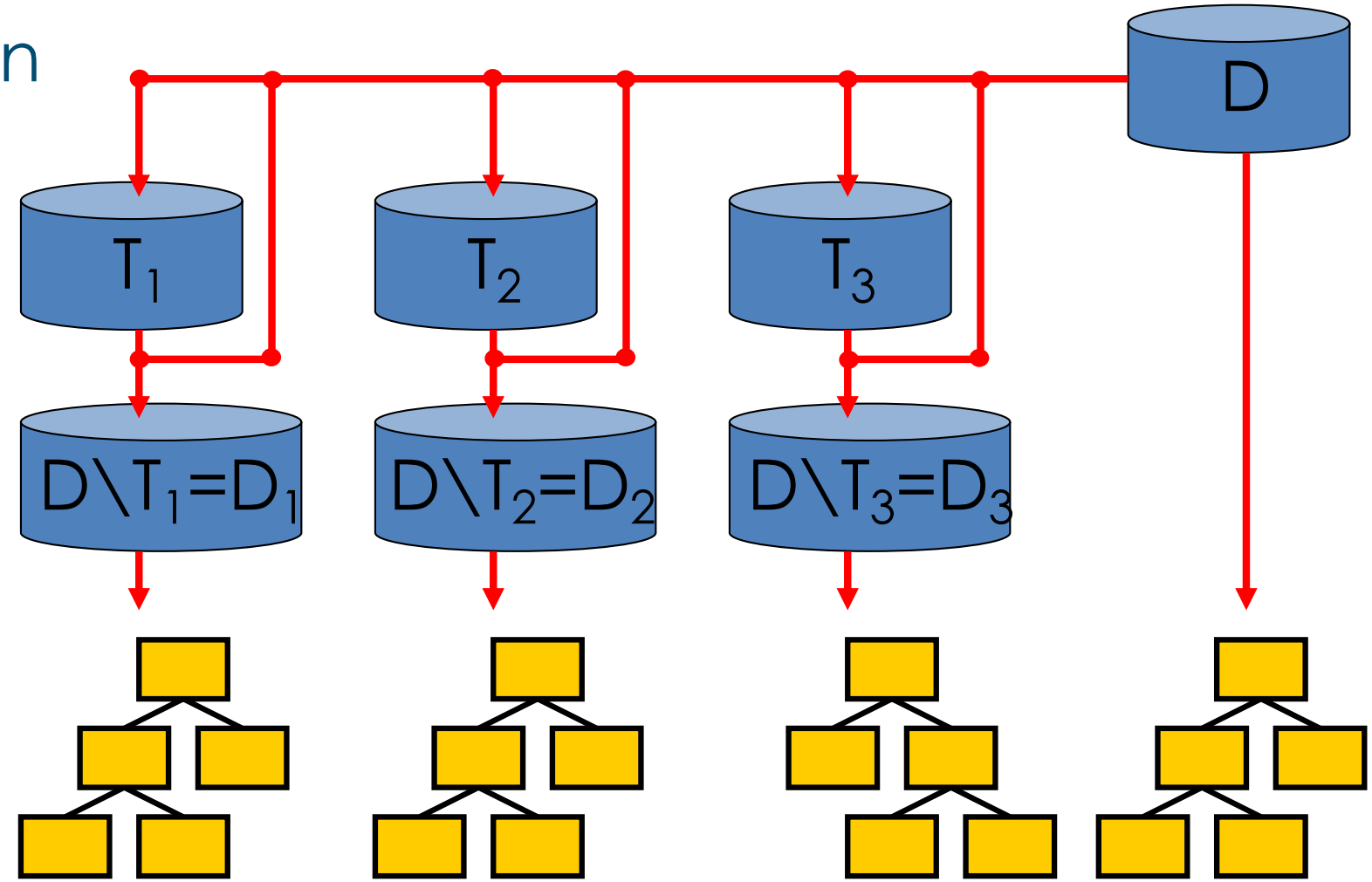
- Partition



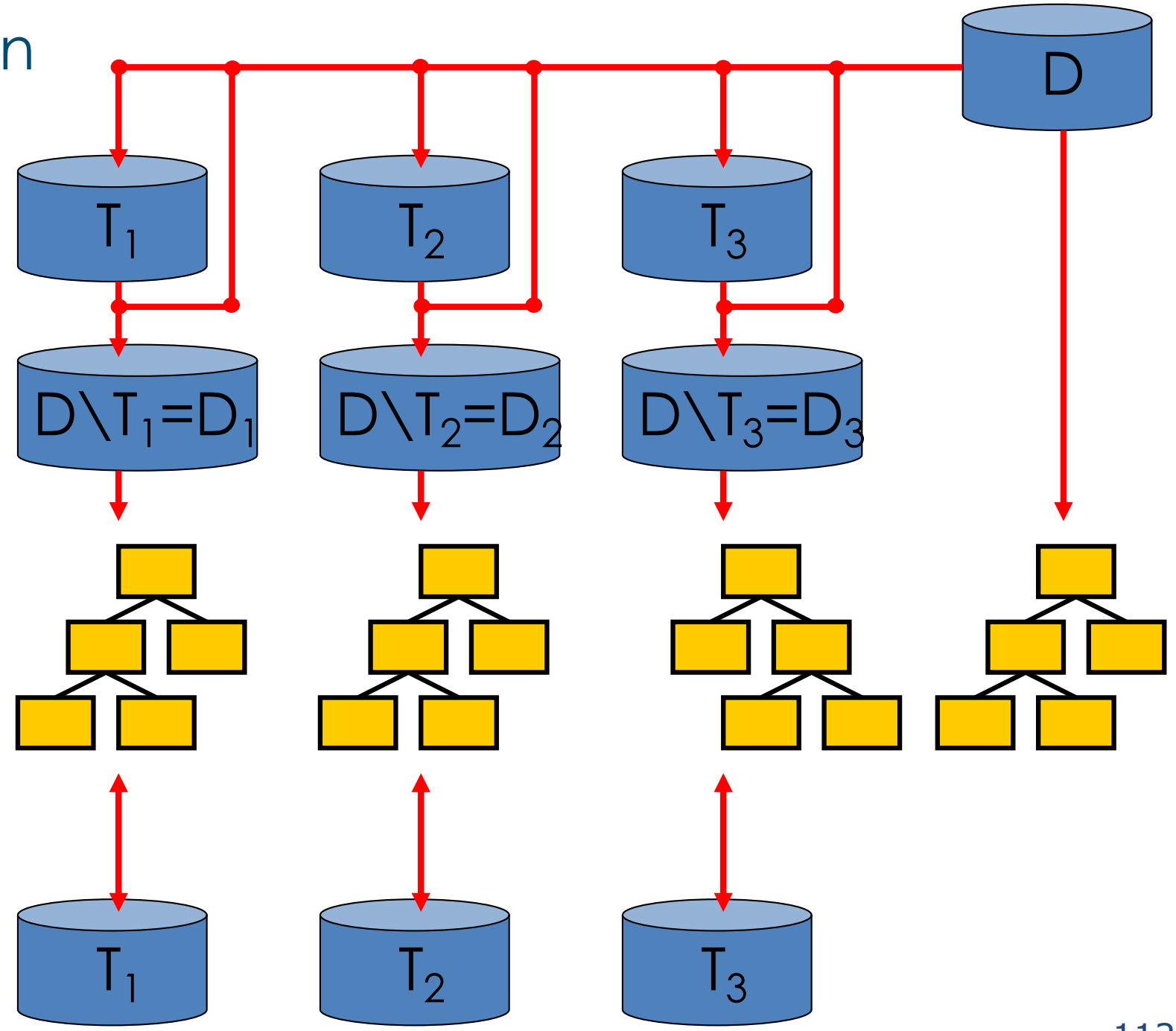
- Partition
- Train



- Partition
- Train

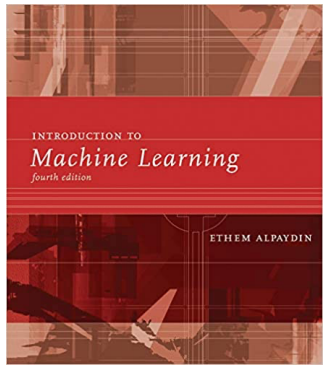


- Partition
- Train
- Test

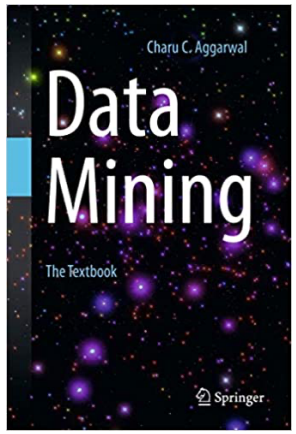


# Interesting Readings

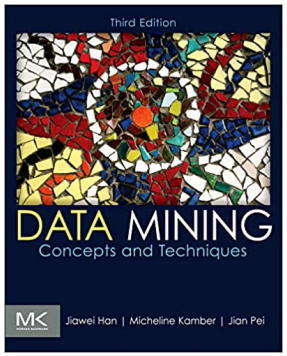
- E. Alpaydin. Introduction to Machine Learning, fourth edition (Adaptive Computation and Machine Learning series). MIT Press, 2020.




- C.C. Aggarwal. Data Mining: The Textbook. Springer, 2015.

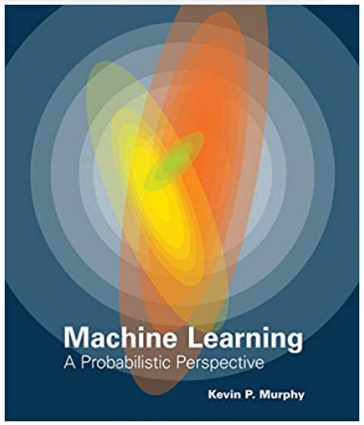



- J. Han, M. Kamber, J. Pei. Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann, 2011.

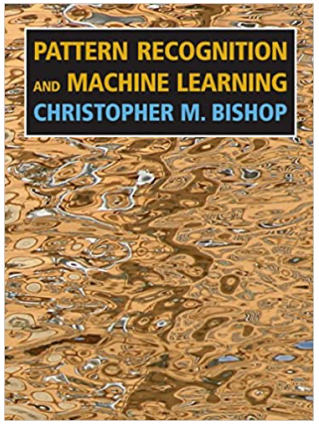


# Interesting Readings

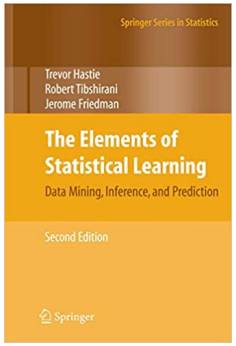
- K.P. Murphy. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). MIT Press, 2012.




- C.M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, 2006.




- T. Hastie, R. Tibshirani, J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics). Springer, 2016.



SOMACHINE

Machine Learning, Big Data, and Deep Learning in Astronomy

# Theoretical Foundations of ML: Classical Problems, Algorithms and Validation


**Salvador García**

Andalusian Research Institute of Data Science and Computational Intelligence (DaSCI)

Dpto. Ciencias de la Computación e I.A.

Universidad de Granada

salvagl@decsai.ugr.es

http://sci2s.ugr.es