Data Science (CDA)

Unit 3: Data Analysis

- José Hernández Orallo, DSIC, UPV, jorallo@dsic.upv.es

Content

- **Unit 3**: Data analysis
  - Predictive and descriptive tasks
  - Supervised techniques
  - Unsupervised techniques
  - Model evaluation

Content

- **Unit 3**: Data analysis
  - Predictive and descriptive tasks
  - Supervised techniques
  - Unsupervised techniques
  - Model evaluation

Predictive and descriptive tasks

- Once the data is integrated, clean, transformed and appropriately selected into a minable view, the "core" analytical stage (modelling) can start.
- The kind of knowledge (task) determines the possible techniques.
- Substages:
  - Define the task and type the variables.
  - Choose a technique/algorithm.
  - Gauge its parameters.
  - Apply/train the model.

Predictive and descriptive tasks

- Frequent itemsets, **associations** and functional dependencies
  - An association between two attributes happens when the frequency of two specific values to happen together is relatively high (also known as co-occurrence grouping or frequent itemset mining).

    Example: in a bookshop we analyse pairs of books that are purchased frequently by the same people.
  - A functional dependency (approximate or absolute) is a pattern in which it is established that one or more attributes determine the value of the other

Predictive and descriptive tasks

- **Correlations** and multi-variate analysis:
  - Correlations relate numerical variables in terms of their relations in magnitude (Pearson) or order (rank correlations, Spearman).

    Example: obesity and age are positively correlaed.

Both associations and correlations are usually included in what is usually called exploratory analysis.

Predictive and descriptive tasks

- **Clustering** / Segmentation:
  - Finds groups of individuals because they are "similar".
    - It's different from classification in that we do not know the classes in advance (or even its number).
    - The goal is to determine groups or clusters which are different from the other.

      Example: find types (clusters) of telephone calls, customers or credit card purchases.
  - It is also useful in preliminary domain exploration.
    - These groups are useful to design different policies for each group or to analyse one group in more detail.

Predictive and descriptive tasks

- **Classification**:
  - A classification can be seen as the clarification of a dependency, in which each dependent attribute can take a value from several classes, known in advance.

    Example: Which customers will respond to a given offer? In this example there are two classes: will respond and will not respond.

    - We have to determine the exact rules that classify a case as positive or negative from the others attributes.

Predictive and descriptive tasks

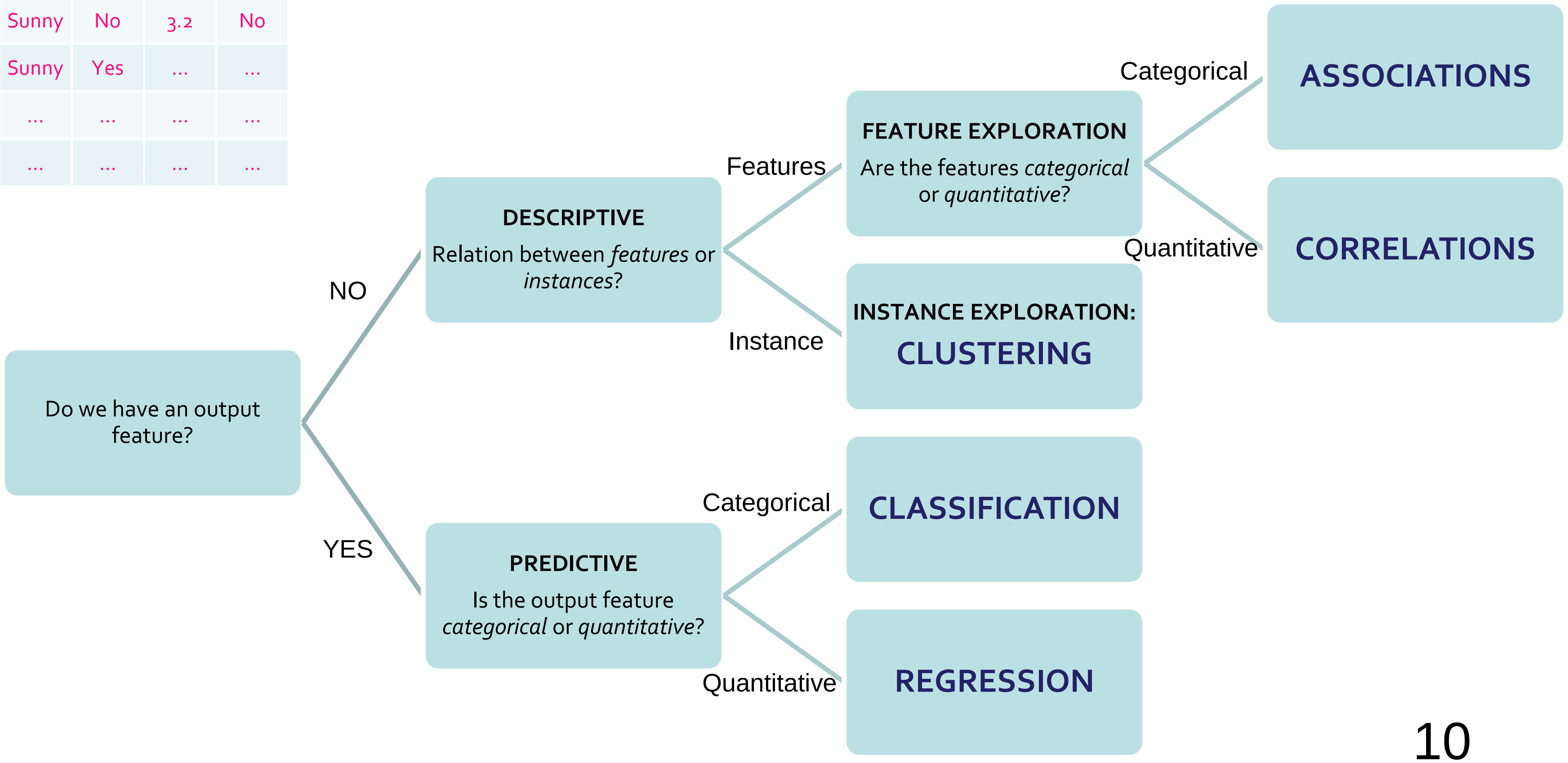
- Trends / **regression**:
  - The goal is to predict the values of a continuous variable from the evolution of another continuous variable (generally time) or from other continuous or nominal variables.

    Example: we need to know the number of future customers or patients, the incomes, calls, earnings, costs, etc. from previous results (day, weeks, months or years before).

Predictive and descriptive tasks

- Taxonomy of tasks made easy:

x1	x2	x3	x4	x5
3.2	Rainy	Yes	4.5	Yes
-4.8	Sunny	No	3.2	No
2.1	Sunny	Yes	…	…
…	…	…	…	…
…	…	…	…	…



Predictive and descriptive tasks

- Other types of knowledge:
  - Link prediction: to predict connections between data items by suggesting that a link should exist.
    - Its is common in social networks and graphs.
  - Profiling: to characterise the typical behaviour of an individual, group, or population.
    - It is often used to establish behavioural norms for anomaly detection applications.
  - General rules: patterns which cannot be classified into the previous kinds: more complex patterns (dynamic, …).
  - Generative models.

Predictive and descriptive tasks

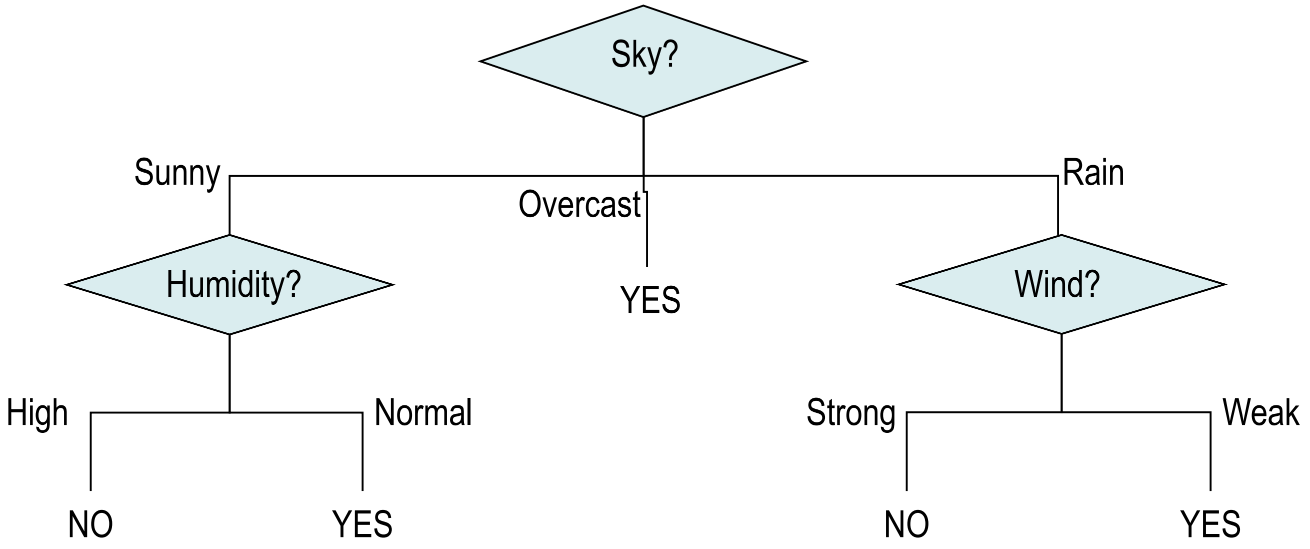
Predictive Model Example

- Will it be nice to play tennis this afternoon?
- Previous experiences:

Example	Sky	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Predictive and descriptive tasks

- We set the type "PlayTennis" as the class (output).
- We choose a **decision tree** and we gauge some parameters (e.g., pruning).
- We train the model and get this result:



- Now we can use to predict the output for a new instance:

(Sky= Sunny, Temperature = Hot, Humidity = High, Wind = Strong) is NO

Predictive and descriptive tasks

Descriptive Model Example

- We have the following table with employee data:

#	Salary	Marrd	Car	Chldrn	House	Union	offsick/Year	WorkYears	Gender
1	10000	Yes	No	0	Rent	No	7	15	M
2	20000	No	Yes	1	Rent	Yes	3	3	F
3	15000	Yes	Yes	2	Owner	Yes	5	10	M
4	30000	Yes	Yes	1	Rent	No	15	7	F
5	10000	Yes	Yes	0	Owner	Yes	1	6	M
6	40000	No	Yes	0	Rent	Yes	3	16	F
7	25000	No	No	0	Rent	Yes	0	8	M
8	20000	No	Yes	0	Owner	Yes	2	6	F
9	20000	Yes	Yes	3	Owner	No	7	5	M
10	30000	Yes	Yes	2	Owner	No	1	20	M
11	50000	No	No	0	Rent	No	2	12	F
12	8000	Yes	Yes	2	Owner	No	3	1	M
13	20000	No	No	0	Rent	No	27	5	F
14	10000	No	Yes	0	Rent	Yes	0	7	M
15	8000	No	Yes	0	Rent	No	3	2	M

- We want to obtain representative subgroups.

Predictive and descriptive tasks

- We don't indicate any output
- We apply the **k-means** algorithm to find clusters with k=3 (three most significant groups).
- We get this result:

cluster 1	cluster 2	cluster 3
5 examples	4 examples	6 examples
Salary : 226000 Married : No -> 0.8 Yes -> 0.2 Car : No -> 0.8 Yes -> 0.2 Children : 0 House : Rent -> 1.0 Union : No -> 0.8 Yes -> 0.2 Offsick/Year : 8 WorkYear : 8 Gender : M -> 0.6	Salary : 225000 Married : No -> 1.0 Car : Yes -> 1.0 Children : 0 House : Rent -> 0.75 Owner -> 0.25 Union : Yes -> 1.0 Offsick/Year : 2 WorkYear : 8 Gender : M -> 0.25 F -> 0.75	Salary : 188333 Married : Yes -> 1.0 Car : Yes -> 1.0 Children : 2 House : Rent -> 0.17 Owner -> 0.83 Union : No -> 0.67 Yes -> 0.33 Offsick/Year : 5 WorkYear : 8 Gender : M -> 0.83 F -> 0.17

Predictive and descriptive tasks

- Flexibility: many supervised techniques have been adapted to unsupervised problems (and vice versa).

TECHNIQUE	PREDICTIVE / SUPERVISED		DESCRIPTIVE / UNSUPERVISED		
	Classification	Regression	Clustering	Association rules	Other (factorial, correl, scatter)
Neural Networks	✓	✓	✓ *		
Decision Trees	✓ (c4.5)	✓ (CART)	✓		
Kohonen			✓		
Linear regression (local, global), exp..		✓			
Logistic Regression	✓				
Kmeans	✓ *		✓		
A Priori (associations)				✓	
factorial analysis, multivariate analysis					✓
CN2	✓				
K-NN	✓		✓		
RBF	✓				
Bayes Classifiers	✓	✓			

Content

- **Unit 3**: Data analysis
  - Predictive and descriptive tasks
  - Supervised techniques
  - Unsupervised techniques
  - Model evaluation

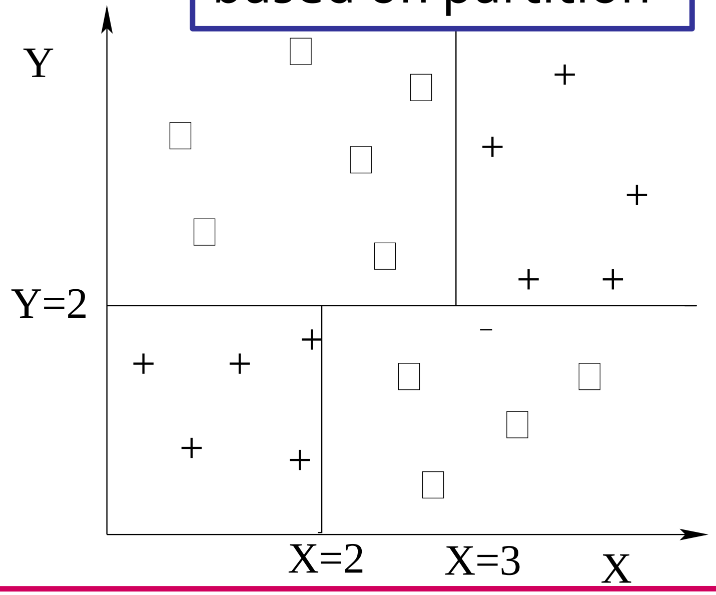
RULE INDUCTION

The model is represented as a set of rules:

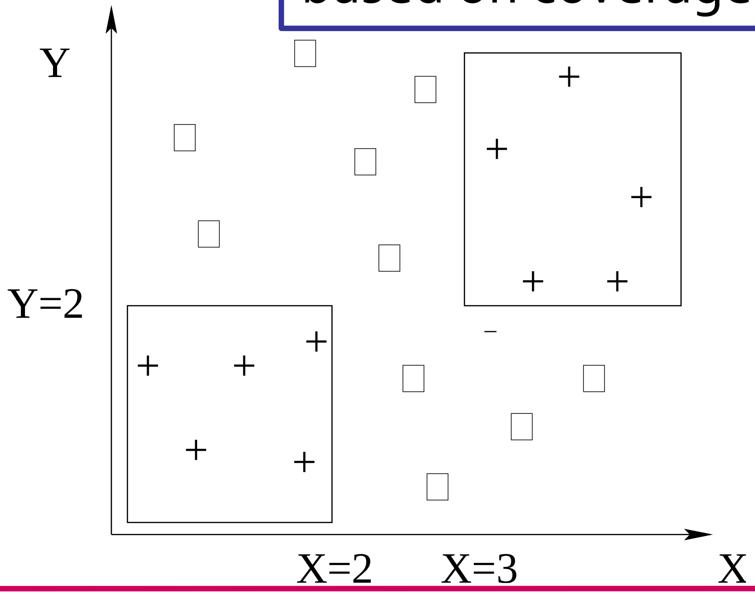
IF … THEN conditions

- Two approaches for rule induction: based on coverage and based on partition.

based on partition



based on coverage



Supervised Learning

- Rule Induction based on coverage
  - A simple approach

```
Repeat until a sufficient set of rules is obtained
    For each attribute A:
        For each value V of that attribute, create a rule:
            1.  count how often each class appears
            2.  find the most frequent class, c
            3.  make a rule "if A=V then Class=c"
        Calculate the error rate of this rule
    Pick the attribute whose rules produce the lowest error rate
```

Playtennis data:

Example	Sky	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Supervised Learning

RULE 1 (humidity)

IF humidity=‘high’

THEN playtenis=no

ERROR RATE = 3/7

RULE 1’ (humidity)

IF humidity=‘normal’

THEN playtenis=yes

ERROR RATE = 1/7

Supervised Learning

RULE 2 (sky)

```
IF sky=‘overcast’
       THEN playtenis=yes
        ELSE IF sky=‘rain’
                  THEN playtenis=yes
                  ELSE sky=‘sunny’
                            THEN playtenis=no
```

confusion matrix

```
   no   yes    <-- classified as
    3      2    | no
    2      7    | yes
```

(3+ 7) / (3+2+2+7) = 71% correct (and 29% incorrect)

Several rules (involving more than one attribute)

IF sky='sunny' AND humidity='high'

THEN playtennis=yes

IF sky='rain' AND temperature='hot'

THEN playtennis=yes

…

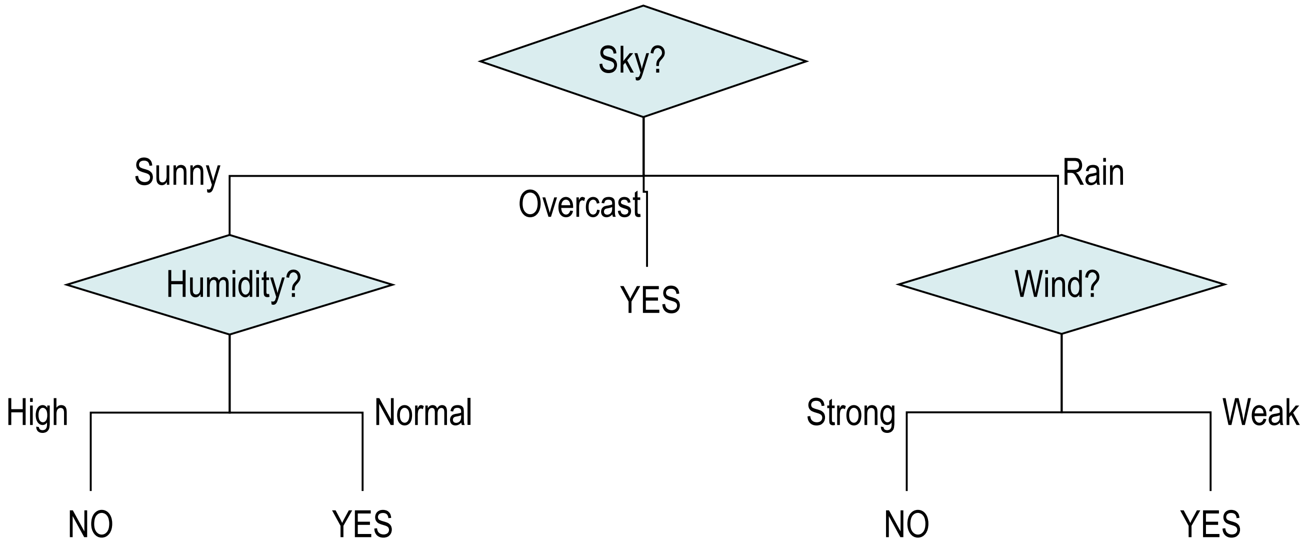
Sequential Covering

```
Initialize R to the empty set
for each class C {
  while D is nonempty {
      1. Construct one rule r that correctly classifies some instances
         in D that belong to class C and does not incorrectly classify
         any non-C instances
      2. Add rule r to ruleset R
      3. Remove from D all instances correctly classified by r
  }
}
return R
```

* from Bill Howe's course "Data Science"

RULE INDUCTION BASED ON PARTITION

Decision Trees (ID3 (Quinlan), C4.5 (Quinlan), CART).



Supervised Learning

- Each path from the root is a rule

```
IF Sky=Overcast
      THEN PlayTennis=yes
      ELSE IF Sky=Sunny
                THEN IF Humidity=High THEN PlayTennis=no
                                         ELSE PlayTennis= yes
                ELSE IF Wind=Strong THEN PlayTennis=no
                                       ELSE PlayTennis= yes
```

- A general algorithm for building decision trees following the divide-and-conquer strategy:

  1. Create the root node and assign D.
  2. If all data in D have the same label, stop the tree construction. Solution found.
  3. Choose a split condition according to a given splitting criterion.
  4. Create 2 subtrees and split D into 2 subsets (data that satisfy or not the splitting condition)
  5. Repeat for each subtree from step 2

- Which attribute do we choose at each level?
  - Depends on the splitting criterion different algorithms exist: (ID3, C4.5, CART)
    - **Entropy** measures homogeneity of examples.
    - **Information gain** measures the expected reduction in entropy when choosing attribute A to split D.

Supervised Learning

- Decision trees:
  - Advantages (especially if pruning is enabled)
    - easy to understand.
    - can be visualised graphically.
    - comprehensibility (model expressed as a set of rules).
    - robust against noise
  - Problems
    - top down, greedy search
    - depending on the splitting criterion and the shape (diagonal borders), partitions can be ad-hoc

Supervised Learning

- ***Naive Bayes Classifiers***

$$\arg\max_{c_i \in C} P(c_i \mid (x_1, x_2, \ldots, x_m)) \underset{Bayes}{=} \arg\max_{c_i \in C} \frac{P((x_1, x_2, \ldots, x_m) \mid c_i) \cdot P(c_i)}{P(x_1, x_2, \ldots, x_m)} =$$

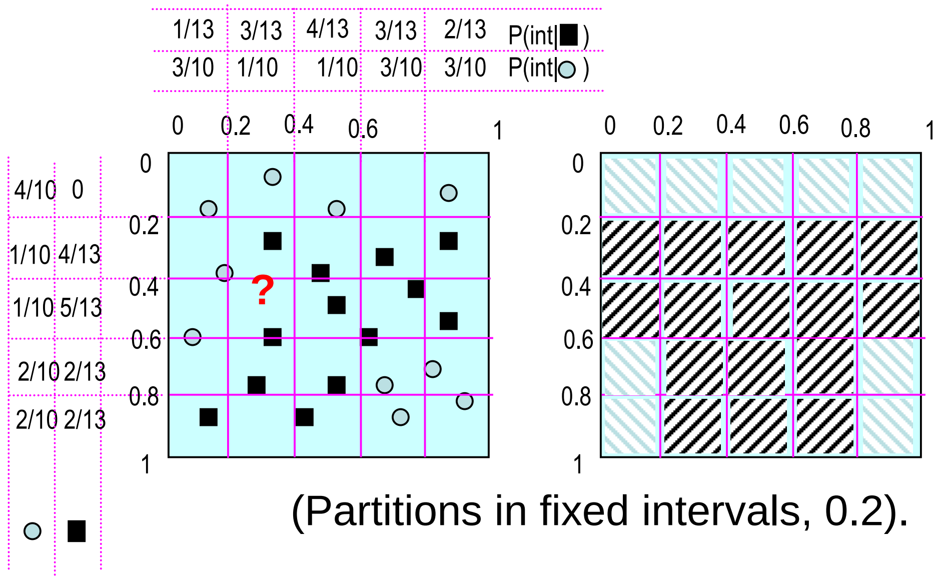
$$= \arg\max_{c_i \in C} P((x_1, x_2, \ldots, x_m) \mid c_i) \cdot P(c_i)$$

  - Assuming that the attributes are independent

$$V_{NB} = \arg\max_{c_i \in C} P(c_i) \prod_i P(x_i \mid c_i)$$

Numerical attributes can be discretised or density functions used.

Example



(Partitions in fixed intervals, 0.2).

P(○)=10/23= 0.435
P(■)=13/23= 0.565

P(○|?) = P(○) · P(0.2<x<=0.4|○) · P(0.4<y<=0.6|○)=
= 0.435 · 1/10 · 1/10 = 0.004

P(■|?) = P(■) · P(0.2<x<=0.4|■) · P(0.4<y<=0.6|■)=
= 0.565 · 3/13 · 5/13 = 0.05

Supervised Learning

- More frequently used with nominal/discrete variables. E.g. playtennis:
- We want to classify a new instance:

(Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong)

$$V_{NB} = \arg\max_{c_i \in \{yes,no\}} P(c_i)\prod_j P(x_j \mid c_i) =$$

$$= \arg\max_{c_i \in \{yes,no\}} P(c_i)\cdot P(Outlook = sunny \mid c_i)\cdot P(Temperature = cool \mid c_i)$$

$$\cdot P(Humidity = high \mid c_i)\cdot P(Wind = strong \mid c_i)$$

- Estimating the 10 necessary probabilities:

  P(Playtennis=yes)=9/14=.64, P(Playtennis=no)=5/14=.36

  P(Wind=strong|Playtennis=yes)=3/9=.33 P(Wind=strong|Playtennis=no)=3/5=.60

  ...

- We have that:

  P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)=0.0053

  P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no)=0.206

Supervised Learning

- Naive Bayes Classifiers. *m*-estimate.
  - With few data, there might be some conditional probability equal to 0.
  - To avoid that, we use an m-estimate of the probability:

$$\frac{n_c + mp}{n + m}$$

where *p* is a prior estimate of the probability of each attribute (generally, *1/k* for *k* attributes), *n* is the number of times the considered class happened in the training data, n_c is the number of samples (of that class) that have a certain value for the considered attribute, *m* that says how confident we are of our prior estimate *p*, as measured in number of samples (generally it is a small number, from 1 to 10).

Case-based Reasoning (CBR) and k-NN (Nearest Neighbours)

- Based on the idea that similar examples will have similar labels.
- How can similarity be measured?
  - distance is the opposite to similarity
- Methods based on similarity/distance store and use the examples already seen to calculate for a new example *s* the similarity/distance between *s* and the stored examples.

Supervised Learning

- Distance can be calculated of different ways:
  - ***Euclidian distance:*** $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - ***Manhattan distance:*** $\sum_{i=1}^{n}|x_i - y_i|$
  - ***Cosine distance:***

  Continuous values (a normalisation between 0 and 1 is recommended)

  - ***Distances based on differences:***
    if x=y then D=0 else D=1
  - ***Edit distance***

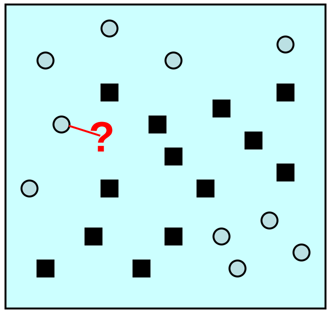
  Discrete values

  - ***Other specific distances***

k-NN (Nearest Neighbour)

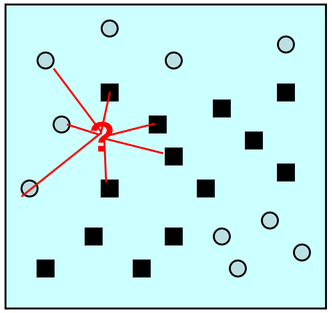
Interactive example:
http://vision.stanford.edu/teaching/cs231n-demos/knn/

(Poliedric or Voronoi)



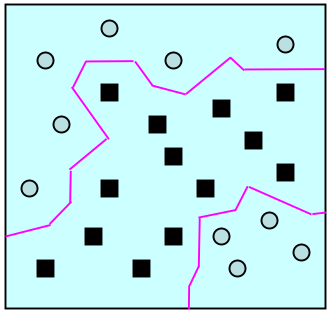
Classifies circle

1-nearest neighbor



Classifies square

7-nearest neighbor



1-nearest neighbor PARTITION

- Whenever we have a new point to classify:
  - assigning the label that is most frequent among the k training samples nearest to it.

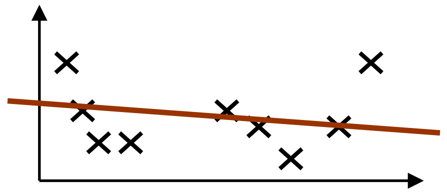
Supervised Learning

- A weighted version of k-NN (Nearest Neighbour): giving more weight to the nearest examples.

$$Attraction(c_j, x_q) = |x_i \in c_j| \cdot krnl_i \quad \text{where} \quad krnl_i = \frac{1}{d(x_q, x_i)^2}$$

  - assign the label c_j with greatest attraction value to the new example x_q .

Linear Regression



- The coefficients of a linear function *f* are estimated

$$\hat{f}(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

- For more than two dimensions it can be solved through *gradient descent*
  - It tries to minimise the cost function J: (m = |D|)

$$J(\theta) = \frac{1}{2m} \sum_{x \in D} \left(\hat{f}(x) - f(x) \right)^2$$

  - and the parameters are adjusted repeatedly:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

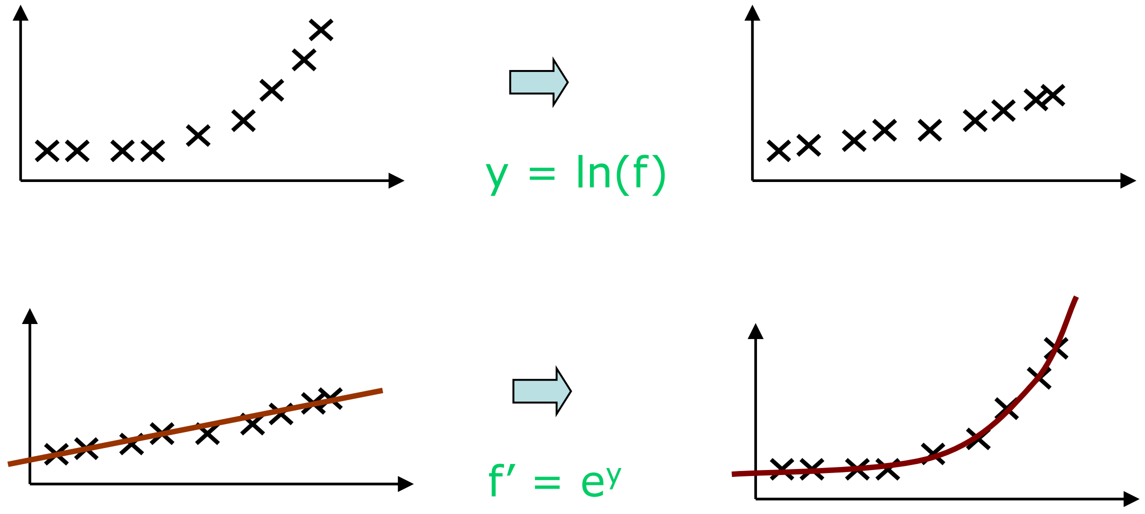
where the derivative is

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{x \in D} \left(\hat{f}(x) - f(x) \right) \cdot x_j$$

Supervised Learning

- ***Nonlinear Regression***
  - the hypothesis (model) is a nonlinear combination of the parameters and the input variables
  - Some functions, such as the exponential or logarithmic functions, can be transformed so that they are linear (and standard linear regression can be performed):

$$y = \ln(f)$$
$$y = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$
$$f = e^y$$



Supervised Learning

- ***Logistic Regression***
  - We can use regression and thresholds for classification tasks:

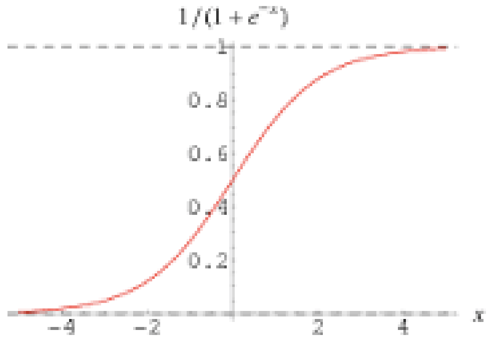
    Ej: Convert p in {neg, pos} into p={0,1} and use threshold classifier at 0.5:

    if $f(x) \geq 0.5$ then "y=pos"

    if $f(x) < 0.5$ then "y=neg"
  - Another option is to apply a sigmoid function (e.g., logistic function) with values between $-\infty$ and ∞: $f = \ln(p/(1-p))$

$$f(x) = g(q^T x)$$

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



Pick and Mix – Supercharging

- Add more variables (dimensions) by combining the existing ones.
    - For instance, if we have three variables x_1, x_2 and x_3, we can add
        - $x_4 = x_1 \cdot x_2$
        - $x_5 = x_3^2$
        - $x_6 = x_1^{x2}$

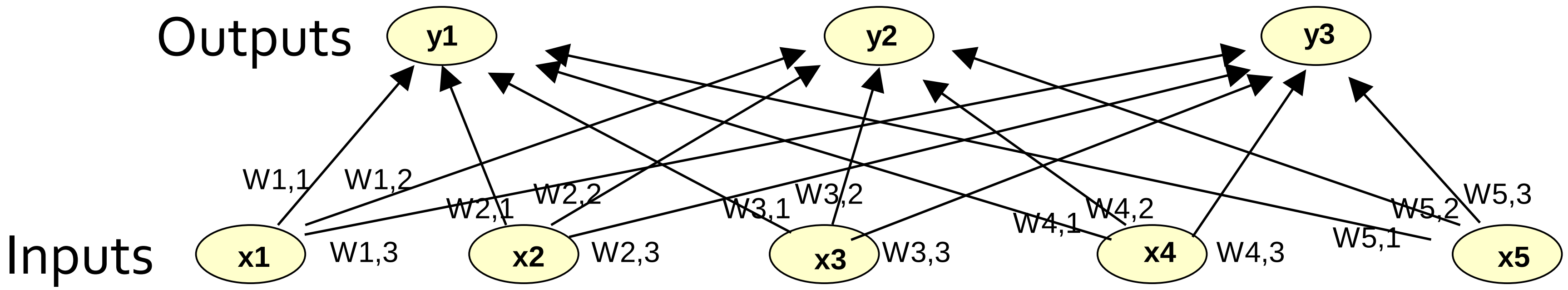
      and learn a linear function from the six variables.

General Nonlinear regression

- Adaptative regression and time series.

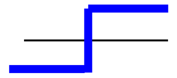
Supervised Learning

Perceptron Learning



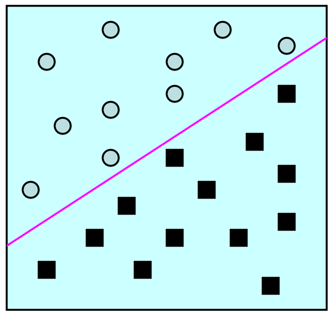
Computes a linear function and then use a threshold:

$$y'_j = \sum_{i=1}^{n} w_{i,j} \cdot x_i$$

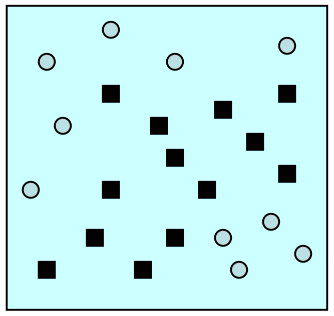


$$output_j = \operatorname{sgn}(y'_j)$$

$$\operatorname{sgn}(x) = \begin{Bmatrix} 1 \; if \; x > 0 \\ 0 \; otherwise \end{Bmatrix}$$



LINEAR PARTITION POSSIBLE



LINEAR PARTITION IMPOSSIBLE

Supervised Learning

- Gradient Descent (basic):
  - Error (Least-Mean Squares) of the m examples

$$E(\vec{w}) = \frac{1}{2}\sum_{k\in 1..m}(e_k)^2 = \frac{1}{2}\sum_{k\in 1..m}(y_k - y'_k)^2$$

  - The aim is to reduce the error step by step. The gradient is the derivate for each vector component

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum\nolimits_{k\in 1..m}(y_k - y'_k)^2 = \frac{1}{2}\sum\nolimits_{k\in 1..m} 2(y_k - y'_k)\frac{\partial}{\partial w_i}(y_k - y'_k) =$$

$$= \sum\nolimits_{k\in 1..m}(y_k - y'_k)\frac{\partial}{\partial w_i}(y_k - \vec{w}\cdot\vec{x}_k) = \sum\nolimits_{k\in 1..m}(y_k - y'_k)(-\vec{x}_{i,k})$$

giving

$$\boxed{\Delta w_i = \sum_{k:1..m}(y_k - y'_k)x_{i,k} = \sum_{k:1..m} x_{i,k}\cdot e_k}$$

Supervised Learning

- Gradient Descent (with momentum: backpropagation):

1. Initialise each $w_{i,j}$ to some small random value (between 0 and 1).

2. Until termination condition (there are no more examples or the average error reaches a threshold) is met, do:
    - initialise each Δw_i to zero
    - for each example (x_k, y_k) in the training dataset (or batch) do
        - compute the output y' and the error $(e_k = y_k - y'_k)$
        - we calculate the weight momentums following the Least-Mean Squares (LMS), with a learning rate (α):

$$Dw_i = Dw_i - a \cdot (x_{i,k} \cdot e_k)$$

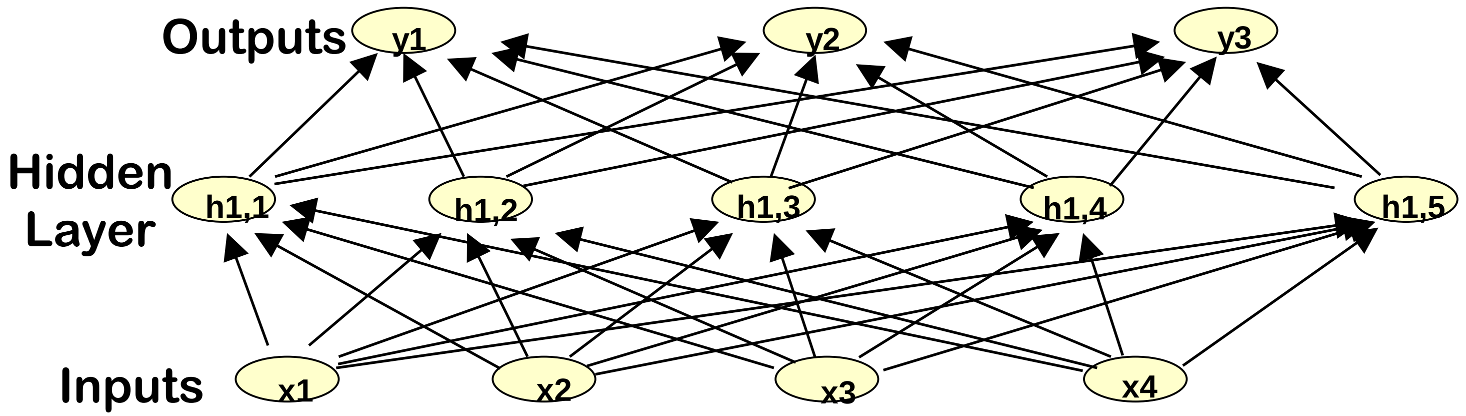
    - for each linear unit weight do

$$w_i = w_i + Dw_i$$

.

Multilayer Perceptron (Artificial Neural Networks, ANN).

- The one-layer perceptron is not able to learn even the most simplest functions.
- We add new internal layers.

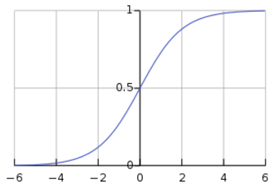


Supervised Learning

- To extend the gradient descent for producing nonlinear functions we can use a sigmoid function (this is the most popular)

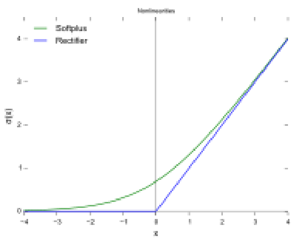
LOGISTIC

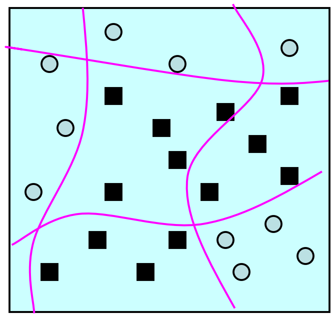
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



RELU/Softplus

$$\sigma(x) = \max(0, x)$$

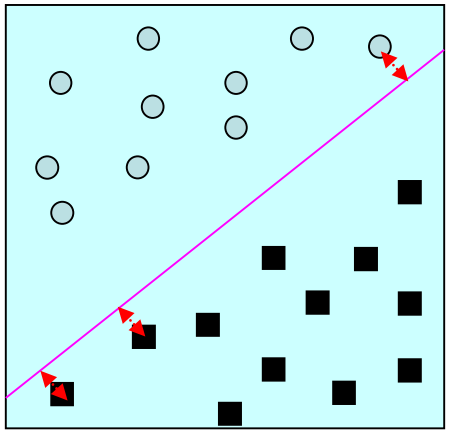




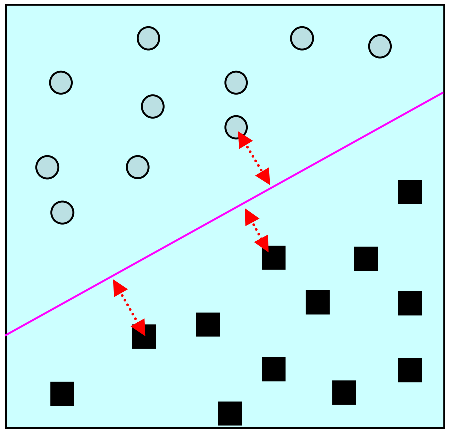
NON-LINEAR MULTIPLE PARTITION IS POSSIBLE WITH 4 INTERNAL UNITS

Support Vector Machines (SVM) / Kernel methods

- The basis is a very simple classifier.
  - The typical classifier is just the line (in more dimensions, a hyperplane) which splits the two classes more neatly in such a way that the three nearest examples to the boundary (the three support vectors) are as far as possible.



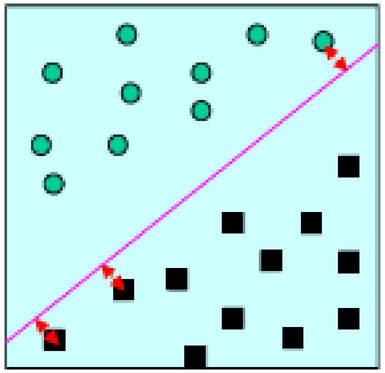
Data is perfectly split, but the three nearest examples are very near to the border line.



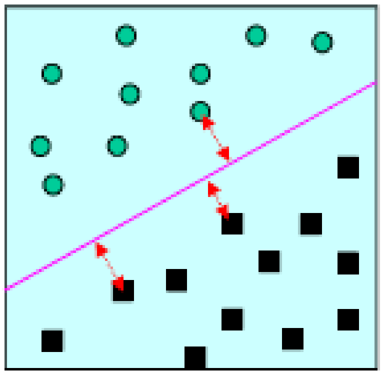
Data is perfectly split, but now the three nearest examples are much farther.

Support Vector Machines (SVM) / Kernel methods

- This linear discriminant is very efficient (even for hundreds of dimensions/attributes), since only a few examples are considered (many of them far away are just not considered).
- The aim is to maximise the distance of the nearest examples (the support vectors) to the boundary.



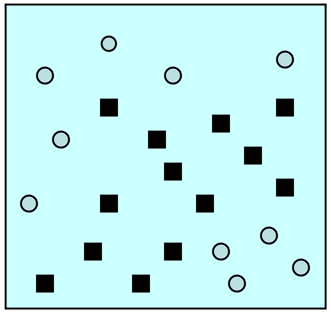
the classes are well separated but the three support vectors are very close to the boundary



the classes are also well separated and the three support vectors are as far as possible from the boundary.

Supervised Learning

- What happens if the data is not linearly separable?



- A kernel function is applied in order to increase the number of dimensions, which usually implies that now the data becomes linearly separable.

Method comparison:

- k-NN:
  - Easy to use.
  - Efficient if the number of examples is not very high.
  - The value *k* can fixed for many applications.
  - The partition is very expressive (complex boundaries).
  - Only intelligible visually (2D or 3D).
  - Robust to noise but not to non-relevant attributes (distances increases, known as the "the curse of dimensionality")

- Neural Networks (multilayer):
  - The number of layers and elements for each layer are difficult to adjust.
  - Appropriate for discrete or *continuous* outputs.
  - Low intelligibility.
  - Very sensitive to outliers (anomalous data).
  - Many examples needed.
  - Require many examples (better the more examples)

- Naive Bayes:
  - Very easy to use.
  - Very efficient (even with many variables).
  - THERE IS NO MODEL.
  - Robust to noise.

- Decision Trees:

  (e.g., C4.5):
  - Very easy to use.
  - Admit discrete and continuous attributes.
  - The output must be finite and discrete (although there are regression decision trees)
  - Noise tolerant, to non-relevant attributes and *missing attribute values*.
  - High intelligibility.