

Prediction

Prediction

- **What is Prediction**
- **Simple methods for Prediction**
- Classification by decision tree induction
- Classification and regression evaluation

Prediction

- Goal: to predict the value of a given variable (named **target** or **objective** variable)
- If the target variable is:
 - Categorical: we have a **classification** problem;
 - Numeric: we have a **regression** problem.
- For each record on the dataset determines the **value of the class attribute**
- Constructs a model based on the **training set**; then, uses the model in **predicting new data**

Prediction

This is an example on classification

Attributes describing customers

Customer class

Examples to train the model

Known

New customers

unknown

Known

Classification

- **Typical Applications:**
 - Credit approval: classifies credit application as low risk, high risk, or average risk
 - Determine if a local access on a computer is legal or illegal
 - Target marketing (send or not a catalogue?)
 - Medical diagnosis
 - Text classification (spam, not spam)
 - Text recognition (Optical character recognition)
 - ...

Regression

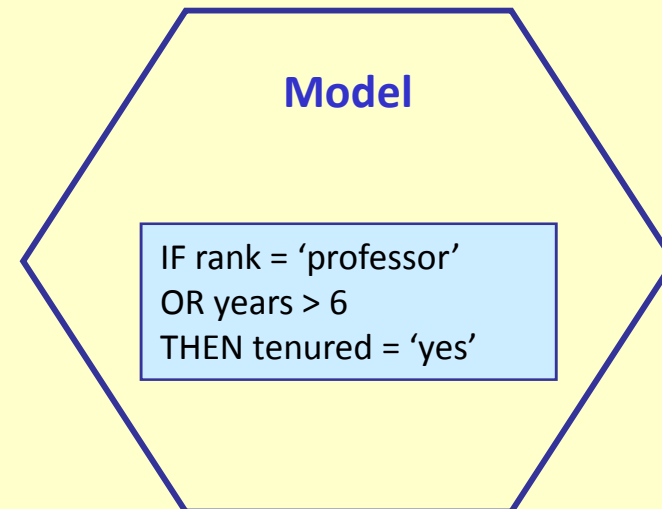
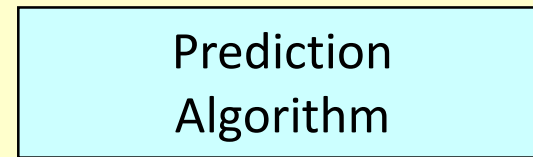
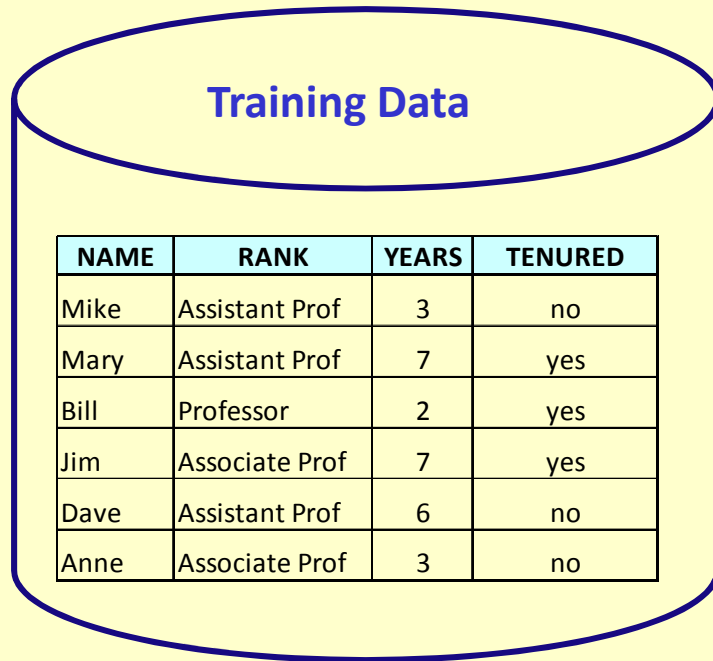
- Typical Applications:

- Stock market: predict the share price for the future
- Energy demanding in a dam
- Wind speed: eolic energy
- Travel time prediction: for the planning of transport companies
- Level of water in a river: for safety & prevention
- Tax income: public budget
- ...

A two step process

- **1. Model construction:**
 - Each **sample** (or record, or object, or instance, or example) is assumed to have a value for the **target attribute**
 - The set of samples is used for the model construction: **training set**
 - The model is represented as classification rules, **decision trees**, or mathematical formulae

Model construction

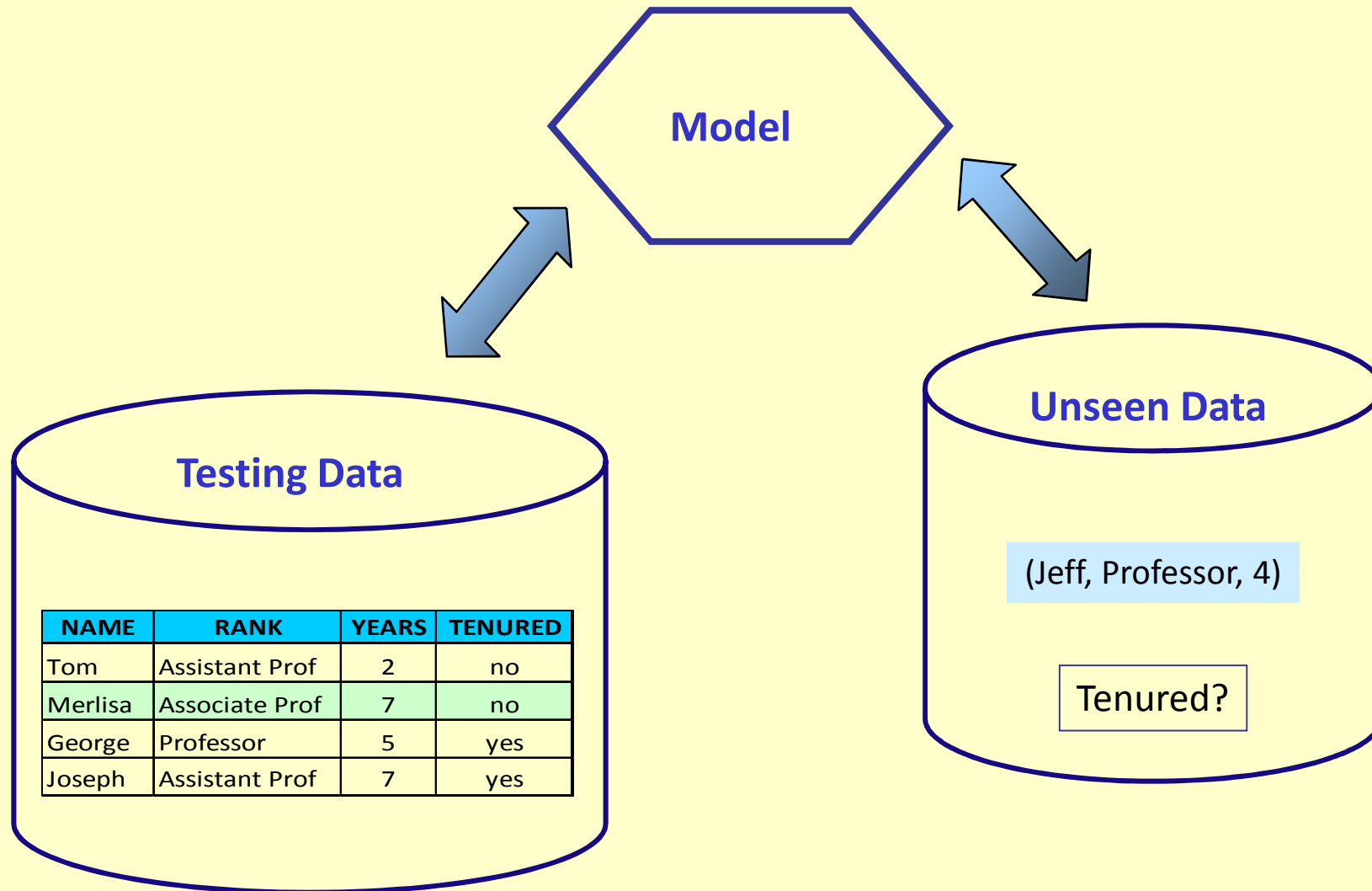


A two step process

■ 2. Model usage

- For predicting future or unknown objects
- Estimate accuracy of the model
 - The known target of test sample is compared with the classified result from the model
 - **For classification: accuracy rate** is the percentage of test set samples that are correctly classified by the model
 - **For regression: mean squared error** is the averaged squared error between each prediction and the corresponding target value
 - **Test set is independent of training set**, otherwise over-fitting will occur

Model usage

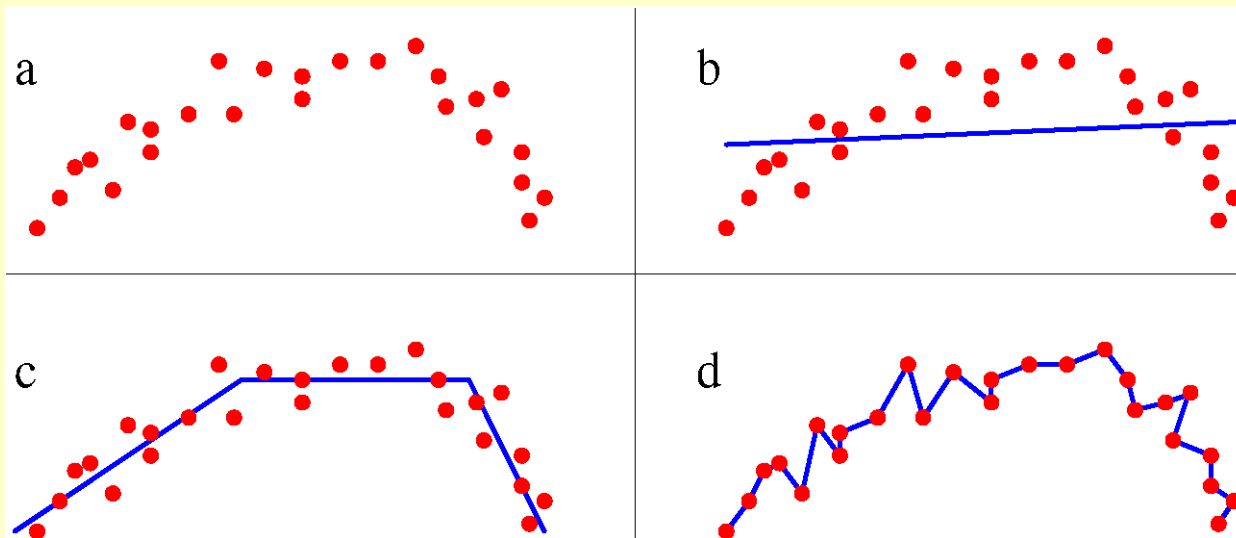


Dataset subsets

- **Training set** – used in model construction
- **Test set** – used in model validation
- **Pruning set** – used in model construction
 - (30% of training set)
- Train/test (70% / 30%)

Overfitting

If your goal is to choose a model in order to predict new values, which one would you choose: b? c? d?



Does the model that best fits the training data (the red dots) is the one that best predicts?

There is no such guarantee!

That is the reason why the model should be evaluated using samples that have not been used for training that model.

Taken from: <http://cg.postech.ac.kr/publications/images/YLee06b.png>

ALGORITHMS

Simplicity first

- Simple algorithms often work very well!
- There are many kinds of simple structure, e.g.:
 - One attribute does all the work
 - All attributes contribute equally & independently
 - A weighted linear combination might do
 - Instance-based: use a few prototypes
 - Use simple logical rules
- Success of method depends on the domain

1-R ALGORITHM

Inferring rudimentary rules

- 1R: learns a 1-level decision tree (we only discuss classification)
 - rules that classify an object on the basis of a single attribute
- Basic version
 - One branch for each value
 - Each branch assigns most frequent class
 - Error rate: proportion of instances that don't belong to the majority class of their corresponding branch
 - Choose attribute with lowest error rate

(assumes nominal attributes)

Pseudo-code for 1R

For each attribute,

For each value of the attribute, make a rule as follows:

count how often each class appears

find the most frequent class

make the rule assign that class to this attribute-value

Calculate the error rate of the rules

Choose the rules with the smallest error rate

- Note: “missing” is treated as a separate attribute value

Evaluating the weather attributes

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

* indicates a tie

oneR rule for this example

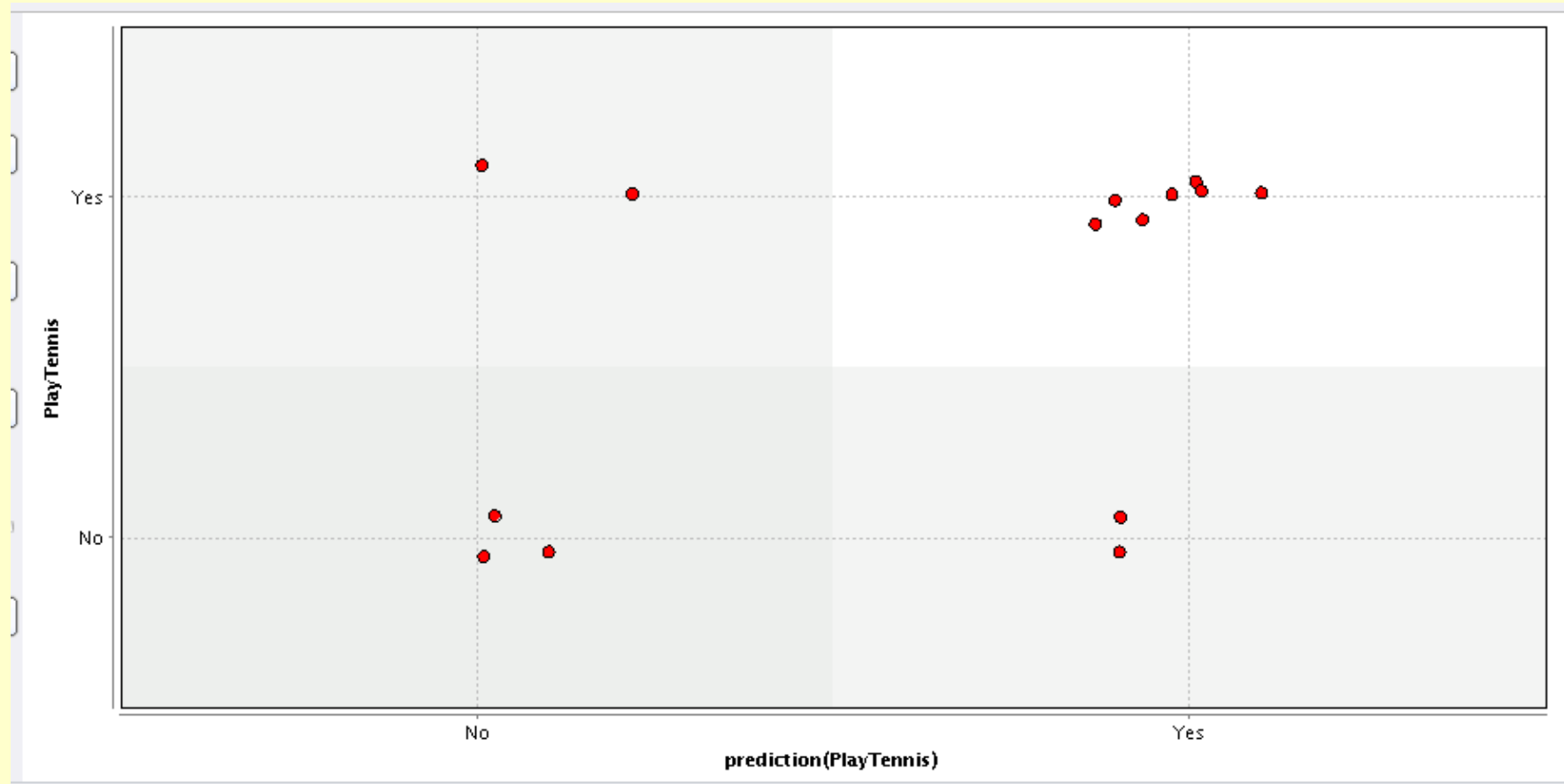
IF **overcast** THEN **Play**
 ELSE IF **sunny** THEN **Don't Play**
 ELSE IF **rain** THEN **Play**



The target variable

oneR rule for this example

IF **overcast** THEN **Play**
ELSE IF **sunny** THEN **Don't Play**
ELSE IF **rain** THEN **Play**



Dealing with numeric attributes

- Discretize numeric attributes (*as seen in a previous class*)
- Divide each attribute's range into intervals
 - Sort instances according to attribute's values
 - Place breakpoints where the class changes (the majority class)
 - This minimizes the total error
- Example: *temperature* from weather data

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

The problem of overfitting

- This procedure is very sensitive to noise
 - One instance with an incorrect class label will probably produce a separate interval
 - High probability of overfitting
- Also: *time stamp* attribute will have zero errors
- Simple solution:
enforce minimum number of instances in majority class per interval

Discretization example

- Example (with $\text{min} = 3$):

64	65	68	69	70	71	72	72	75	75	80	81	83	85				
Yes	No	Yes	Yes	Yes		No	No	Yes	Yes	Yes	Yes		No	Yes	Yes	Yes	No

- Final result for temperature attribute

64	65	68	69	70	71	72	72	75	75	80	81	83	85	
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes		No	Yes	Yes	No

With overfitting avoidance

- Resulting rule set:

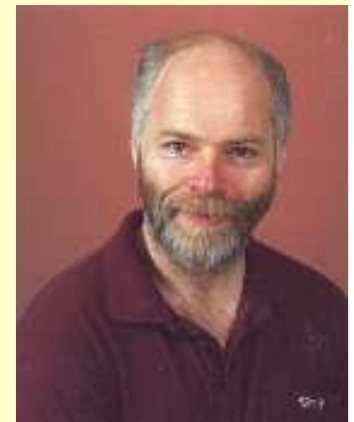
Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	$\leq 77.5 \rightarrow$ Yes	3/10	5/14
	$> 77.5 \rightarrow$ No*	2/4	
Humidity	$\leq 82.5 \rightarrow$ Yes	1/7	3/14
	> 82.5 and $\leq 95.5 \rightarrow$ No	2/6	
	$> 95.5 \rightarrow$ Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Discussion of 1R

- 1R was described in a paper by Holte (1993)
 - Contains an experimental evaluation on 16 datasets (using *cross-validation* so that results were representative of performance on future data)
 - Minimum number of instances was set to 6 after some experimentation
 - 1R's simple rules performed not much worse than much more complex decision trees
- Simplicity first pays off!

Very Simple Classification Rules Perform Well on Most Commonly Used Datasets

Robert C. Holte, Computer Science Department, University of Ottawa



BAYESIAN CLASSIFICATION

Bayesian (Statistical) modeling

- “Opposite” of 1R: use all the attributes
- **Two assumptions:** Attributes are
 - *equally important*
 - *statistically independent* (given the class value)
 - I.e., knowing the value of one attribute says nothing about the value of another (if the class is known)
- Independence assumption is almost never correct!
- But ... this scheme works well in practice

Bayesian Theorem

- Given training data D , posteriori probability of a hypothesis h , $P(h|D)$, follows the Bayes theorem

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)}$$

- The goal is to identify the hypothesis with the maximum posteriori probability
- **Practical difficulty:** require initial knowledge of many probabilities, significant computational cost

Bayesian Classification

We want to determine $P(C|X)$, that is, the probability that the record $X = \langle x_1, \dots, x_k \rangle$ is of class C .

e.g. $X = \langle \text{rain, hot, high, light} \rangle$ PlayTennis?

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

Bayesian Classification

- The classification problem may be formalized using **a-posteriori probabilities**
- $P(C|X)$ = probability that the record $X = \langle x_1, \dots, x_k \rangle$ is of class C.

e.g. $P(\text{class}=\text{N} \mid \text{outlook}=\text{sunny}, \text{windy}=\text{light}, \dots)$

- Thus: assign to sample **X** the class label **C** such that **$P(C|X)$ is maximal**

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative frequency of class C samples
- C such that $P(C|X)$ is maximum =
C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C) = P(x_1, \dots, x_k | C)$ is **unfeasible!**

Naive Bayesian Classification

- Naïve assumption: **attribute independence**

$$P(x_1, \dots, x_k | C) = P(x_1 | C) \cdot P(x_2 | C) \cdot \dots \cdot P(x_k | C)$$

- If i attribute is **categorical**:
 - $P(x_i | C)$ is estimated as the relative freq of samples having value x_i as i -th attribute in class C
- If i -th attribute is **continuous**:
 - $P(x_i | C)$ is estimated through a Gaussian density function
- Computationally easy in both cases

Naive Bayesian Classifier

- The **assumption** that attributes are conditionally independent greatly **reduces the computational cost**.
- Given a training set, we can compute the following probabilities

Example: $P(\text{sunny} | \text{Play})$

Outlook	P	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5

Play-tennis example: estimating P(C)

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

$$P(P) = 9/14$$

$$P(N) = 5/14$$

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temp.	Humidity	Wind	Play?
Sunny	Hot	High	Light	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Light	Yes
Rain	Mild	High	Light	Yes
Rain	Cool	Normal	Light	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Light	No
Sunny	Cool	Normal	Light	Yes
Rain	Mild	Normal	Light	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Light	Yes
Rain	Mild	High	Strong	No

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{strong} p) = 3/9$	$P(\text{strong} n) = 3/5$
$P(\text{light} p) = 6/9$	$P(\text{light} n) = 2/5$

Play-tennis example: classifying X

- An unseen sample

$X = \langle \text{rain, hot, high, light} \rangle$

$$\begin{aligned} P(\text{play} \mid X) &\Rightarrow P(X \mid \text{play}) \cdot P(\text{play}) = \\ &= P(\text{rain} \mid \text{play}) \cdot P(\text{hot} \mid \text{play}) \cdot \\ &\cdot P(\text{high} \mid \text{play}) \cdot P(\text{light} \mid \text{play}) \cdot P(p) = \\ &= \frac{3}{9} \cdot \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} = \\ &= 0.010582 \end{aligned}$$

outlook	
$P(\text{sunny} \mid p) = \frac{2}{9}$	$P(\text{sunny} \mid n) = \frac{3}{5}$
$P(\text{overcast} \mid p) = \frac{4}{9}$	$P(\text{overcast} \mid n) = 0$
$P(\text{rain} \mid p) = \frac{3}{9}$	$P(\text{rain} \mid n) = \frac{2}{5}$
temperature	
$P(\text{hot} \mid p) = \frac{2}{9}$	$P(\text{hot} \mid n) = \frac{2}{5}$
$P(\text{mild} \mid p) = \frac{4}{9}$	$P(\text{mild} \mid n) = \frac{2}{5}$
$P(\text{cool} \mid p) = \frac{3}{9}$	$P(\text{cool} \mid n) = \frac{1}{5}$
humidity	
$P(\text{high} \mid p) = \frac{3}{9}$	$P(\text{high} \mid n) = \frac{4}{5}$
$P(\text{normal} \mid p) = \frac{6}{9}$	$P(\text{normal} \mid n) = \frac{2}{5}$
windy	
$P(\text{strong} \mid p) = \frac{3}{9}$	$P(\text{strong} \mid n) = \frac{3}{5}$
$P(\text{light} \mid p) = \frac{6}{9}$	$P(\text{light} \mid n) = \frac{2}{5}$

Play-tennis example: classifying X

$$\begin{aligned} P(\text{don't play} \mid X) &\Rightarrow P(X \mid \text{don't play}) \cdot P(\text{don't play}) = \\ &= P(\text{rain} \mid \text{don't play}) \cdot P(\text{hot} \mid \text{don't play}) \cdot \\ &\quad \cdot P(\text{high} \mid \text{don't play}) \cdot P(\text{light} \mid \text{don't play}) \cdot \\ &\quad \cdot P(\text{don't play}) = \\ &= 2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286 > 0.010582 \end{aligned}$$

Sample **X** is classified in class **n** (don't play)

The “zero-frequency problem”

- What if an **attribute value doesn't occur** with every class value?
(e.g. “Humidity = high” for class “yes”)

- Probability will be zero! $\Pr[\textit{Humidity} = \textit{High} \mid \textit{yes}] = 0$
- *A posteriori* probability will also be zero!
(No matter how likely the other values are!)

$$\Pr[\textit{yes} \mid \langle \dots, \textit{hum} = \textit{high} \rangle] = 0$$

- Remedy: add 1 to the count for every attribute value-class combination (*Laplace estimator*)
- Result: probabilities will never be zero!
(also: stabilizes probability estimates)

Modified probability estimates

- In some cases adding a constant different from 1 might be more appropriate
- Example: attribute *outlook* for class *yes*

$$\frac{2 + \mu/3}{9 + \mu}$$

Sunny

$$\frac{4 + \mu/3}{9 + \mu}$$

Overcast

$$\frac{3 + \mu/3}{9 + \mu}$$

Rainy

- Weights don't need to be equal (but they must sum to 1)

$$\frac{2 + \mu p_1}{9 + \mu}$$

$$\frac{4 + \mu p_2}{9 + \mu}$$

$$\frac{3 + \mu p_3}{9 + \mu}$$

Missing values

- **Training:** instance is not included in frequency count for attribute value-class combination
- **Classification:** attribute will be omitted from calculation

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{strong} p) = 3/9$	$P(\text{strong} n) = 3/5$
$P(\text{light} p) = 6/9$	$P(\text{light} n) = 2/5$

Example:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	Strong	?

Likelihood of "yes" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

Likelihood of "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

$P(\text{"yes"}) = 0.0238 / (0.0238 + 0.0343) = 41\%$

$P(\text{"no"}) = 0.0343 / (0.0238 + 0.0343) = 59\%$

Numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The *probability density function* for the normal distribution is defined by two parameters:

- *Sample mean* μ

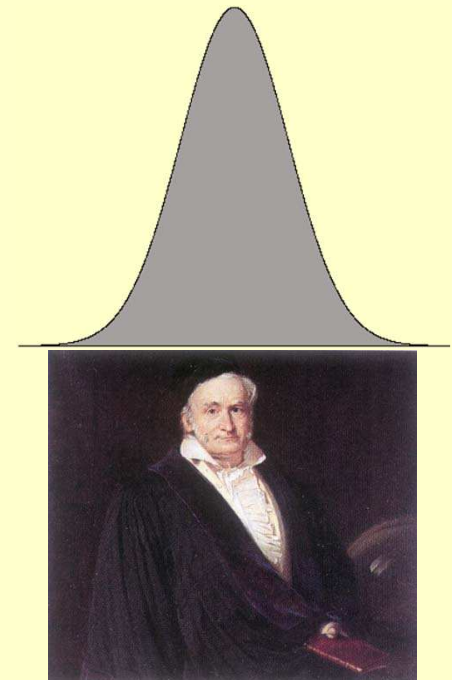
$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

- *Standard deviation* σ

$$\sigma = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2$$

- Then the density function $f(x)$ is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Karl Gauss, 1777-1855
great German mathematician

Probability densities

- Relationship between probability and density:

$$\Pr\left[c - \frac{\varepsilon}{2} < x < c + \frac{\varepsilon}{2}\right] \approx \varepsilon * f(c)$$

- But: this doesn't change calculation of *a posteriori* probabilities because ε cancels out
- Exact relationship:

$$\Pr[a \leq x \leq b] = \int_a^b f(t)dt$$

Statistics for weather data

Outlook	Temperature		Humidity		Windy		Play				
	Yes	No	Yes	No	Yes	No	Yes	No			
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, ...	85, ...	80, ...	95, ...					
Sunny	2/9	3/5	$\mu = 73$	$\mu = 75$	$\mu = 79$	$\mu = 86$	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	$\sigma = 6.2$	$\sigma = 7.9$	$\sigma = 10.2$	$\sigma = 9.7$	True	3/9	3/5		
Rainy	3/9	2/5									

- Example density value:

$$f(\text{temperature} = 66 \mid \text{yes}) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

Classifying a new day

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of "no" = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

$P(\text{"yes"}) = 0.000036 / (0.000036 + 0.000136) = 20.9\%$

$P(\text{"no"}) = 0.000136 / (0.000036 + 0.000136) = 79.1\%$

- Missing values during training are not included in calculation of mean and standard deviation

Naïve Bayes: discussion

- Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- Why? Because classification doesn't require accurate probability estimates *as long as maximum probability is assigned to correct class*
- However: adding too many redundant attributes will cause problems (e.g. identical attributes)
- Note also: many numeric attributes are not normally distributed

Naïve Bayes Extensions

- Improvements:
 - select best attributes (e.g. with greedy search)
 - often works as well or better with just a fraction of all attributes
- Bayesian Networks
 - combine Bayesian reasoning with causal relationships between attributes
 - is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG)

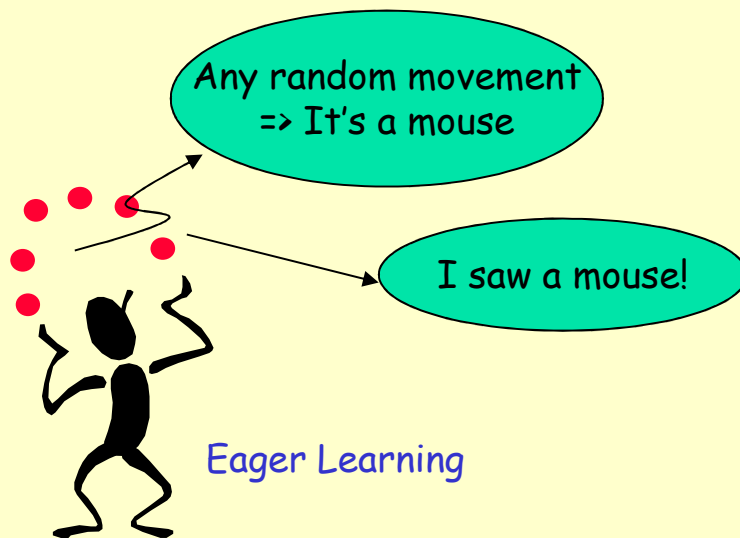
Summary

- OneR – uses rules based on just one attribute
- Naïve Bayes – uses all attributes and Bayes rules to estimate probability of the class given an instance.
- Simple methods frequently work well, but ...
 - Complex methods can be better (as we will see)

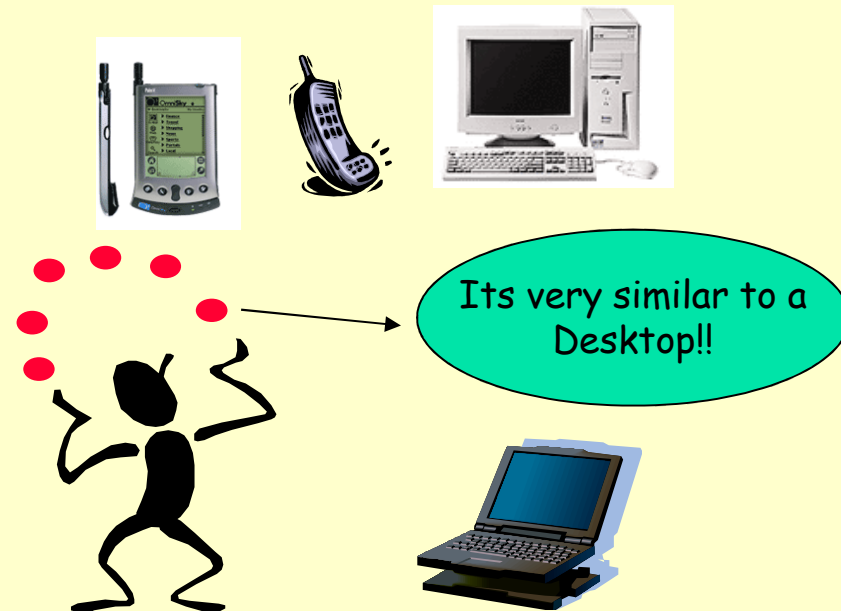
K-NEAREST NEIGHBOR CLASSIFIER

Different Learning Methods

- Instance-based learning:
 - Learning=storing all training instances
 - Classification=assigning target function to a new instance
 - Delay the processing until a new instance must be classified
 - Referred to as “Lazy” learning



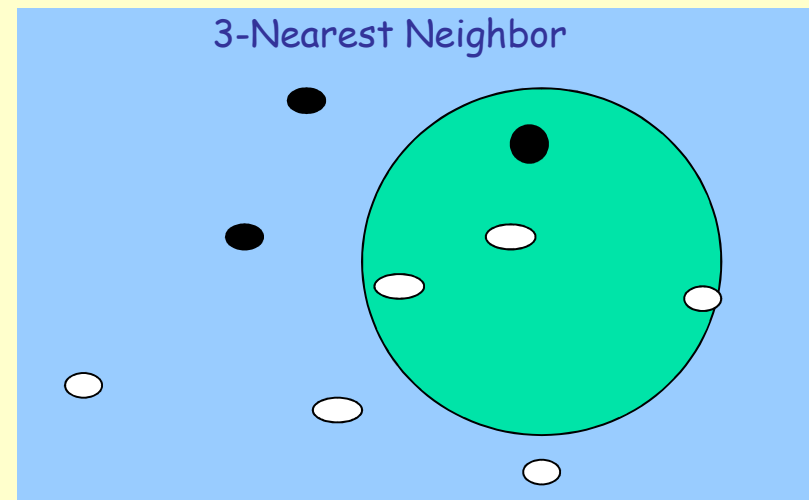
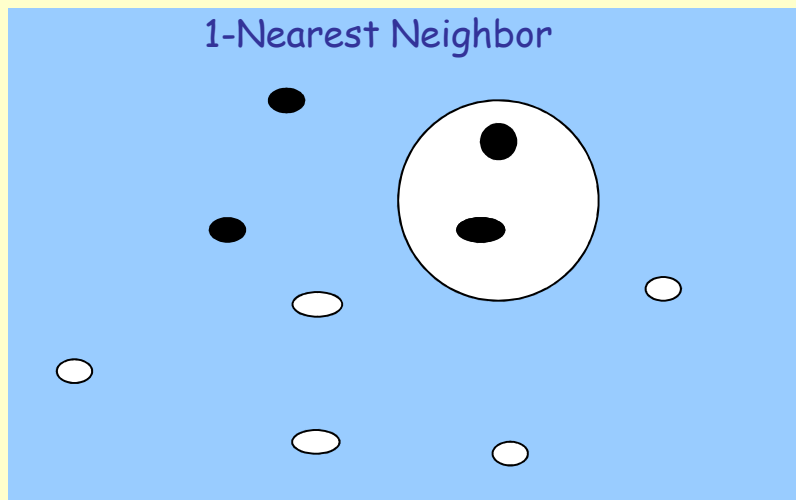
Instance-based Learning



The k -Nearest Neighbor

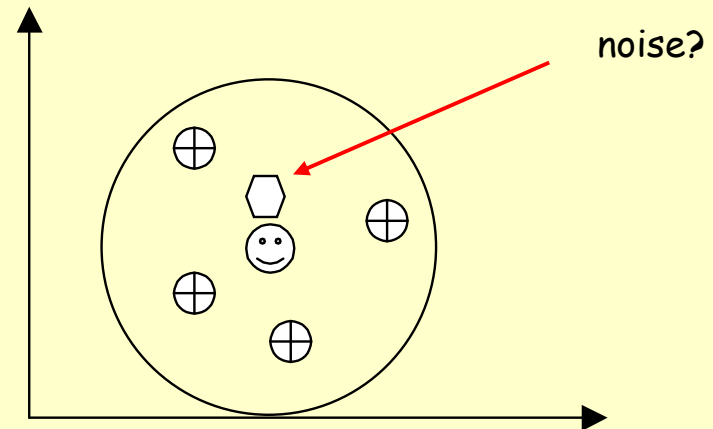
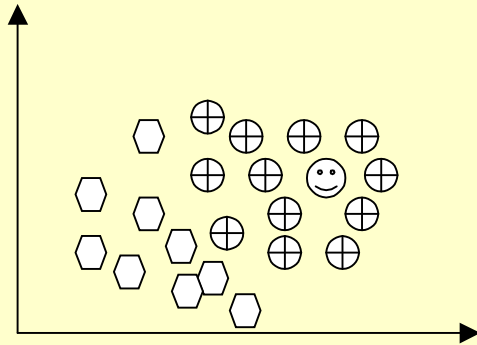
- Nearest neighbor classifiers are based on learning by analogy
- All instances correspond to points in the n -Dimensional space
- The nearest neighbors are defined in terms of Euclidean distance (need to normalize attributes)

$$d(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$



Example

Simple 2-D case, each instance described only by two values (x, y co-ordinates). The class is either \oplus or \hexagon

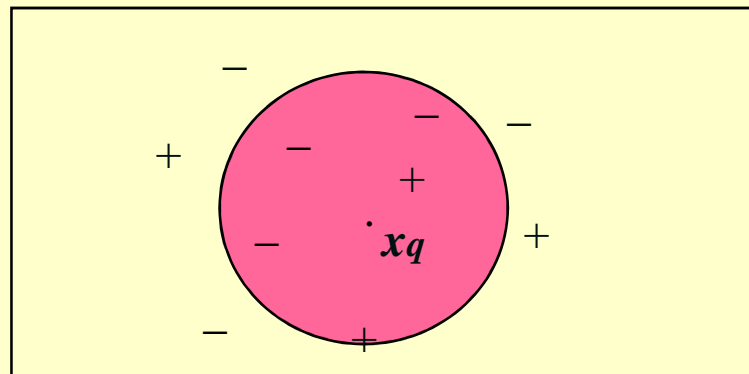


Need to consider :

- 1) Similarity (how to calculate distance)
- 2) Number (and weight) of similar (near) instances

The k -Nearest Neighbor

- The target function could be discrete- or real-valued.
- For discrete-valued, the k -NN returns the most common value among the k training examples nearest to x_q .
 - $k=1$ leads to an unstable classifier
 - k large means that the points may not be very close



Discussion on the k -NN Algorithm

- The k -NN algorithm for continuous-valued target functions
 - Calculate the mean values of the k nearest neighbors.
- Distance-weighted nearest neighbor algorithm
 - Weight the contribution of each of the k neighbors according to their distance to the query point x_q
 - giving greater weight to closer neighbors
 - Similarly, for real-valued target functions
- Robust to noisy data by averaging k -nearest neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

Advantages	Disadvantages
Training is very fast	Slow at query time
Learn complex target functions	Easily fooled by irrelevant attributes

Remarks on Lazy vs. Eager Learning

- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences
 - Lazy method may consider query instance when deciding how to generalize beyond the training data D
 - Eager method cannot since they have already chosen global approximation when seeing the query
- Efficiency: Lazy - less time training but more time predicting
- Accuracy
 - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
 - Eager: must commit to a single hypothesis that covers the entire instance space

References

- Jiawei Han and Micheline Kamber, “Data Mining: Concepts and Techniques”, 2000
- Ian H. Witten, Eibe Frank, “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”, 1999
- Tom M. Mitchell, “Machine Learning”, 1997



Thank you !!!