Chapter 3

Knowledge discovery from web data

PROFESSORS

Juan D. Velásquez
Víctor Rebolledo L.

Outline

1. The KDD Process
2. Data sources and cleaning
3. Data consolidation and information repositories
4. Data mining
5. **Tools for mining data**
6. Using data mining to extract knowledge
7. Validation of the extracted knowledge
8. Mining the web

Section 3.5

- Tools for Data Mining

Tools for Data Mining

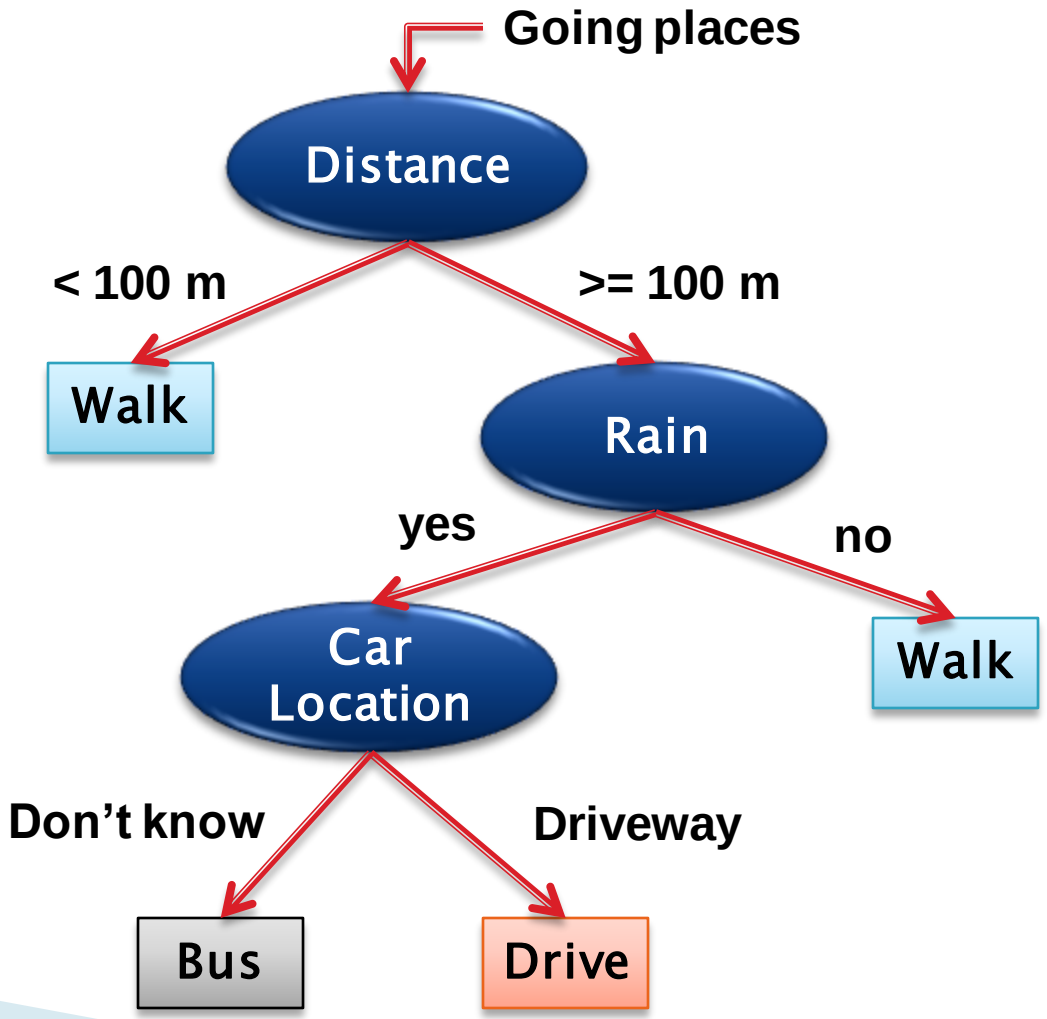
- Artificial Neural Networks
- Self-Organizing Feature Maps
- K-Means
- Decisions Trees
- Bayesian network
- K-Nearest Neighbor
- Support Vector Machines

Decisions trees

Decision Trees
(what they look like)

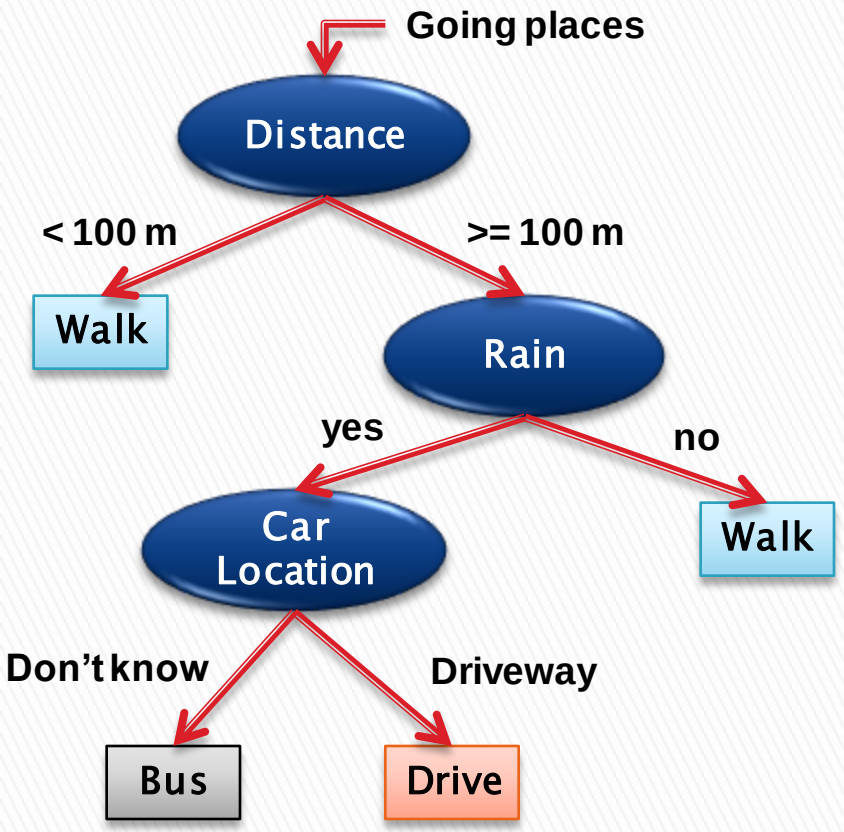
- Example:
  - **How to go to different places.**
    - By walk?
    - By car?
    - By bus?
  - **Parameters:**
    - wheather
    - travel distance, …
  - **The data:**
    - a survey over different person.

Decision Trees
(what they look like)



Programs that Construct Trees
(what goes in and what comes out)

Yes	45	…	…	Walk
No	125	…	…	Walk
Yes	190	…	…	Drive
Yes	…	…	…	…
Yes	…	…	…	…
No	…	…	…	…



Types of Trees

- **Classification Trees**:

  Assign the **non-numeric (or discrete)** target value to each record. For example, tomorrow's weather can be classified (predicted) as one of

  "sunny“

  "cloudy“

  or "rain".

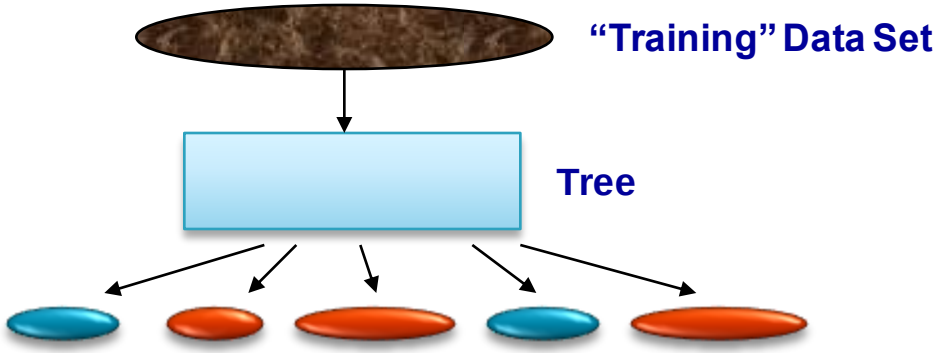
- **Regression Trees**:

  Estimate the **numeric** target value for each record. For example, tomorrow's maximum temperature would be 27° C.

Types of Trees …. (contd.)

- All of these trees have the **same basic structure**
- However, there are a **variety of algorithms for building tree-based models**
- Some of the most popular decision tree algorithms are:
  - ID3 (Quinlan, 1986)
  - CHAID
  - CART
  - C4.5
  - See5/C5.0, etc.
- CART and See5/C5.0 are **widely accepted as some of the best algorithms** for constructing tree-based models for data

Basic Tree Builder



- Trees are constructed by **repeatedly partitioning the training data set** into many **subsets**
- The aim is to **identify subsets** each of which contains **cases with one target value**
- In other words, we want to find subsets such that they are ***purer*** (have less *diversity*) than the original data set

Basic Tree Builder .... (contd.)

Measuring Purity

The term **entropy** (used in information theory) indicates the **impurity** of an arbitrary collection of cases (examples)

$$Entrophy(S) = -(p_{pos})\log_2(p_{pos}) - (p_{neg})\log_2(p_{neg})$$

For a Boolean function (target is up or down),

where,
S is a sample of training cases, (p_{pos}) is a proportion of positive cases (say up values), (p_{neg}) is a proportion of negative cases (say down values)

Entropy([15 up, 4 down]) = $-(15/19)\log_2(15/19) - (4/19)\log_2(4/19)$
= 0.742

Basic Tree Builder .... (contd.)

- The entropy (impurity) is **0** if **all** members of S belong to the same class (up or down)

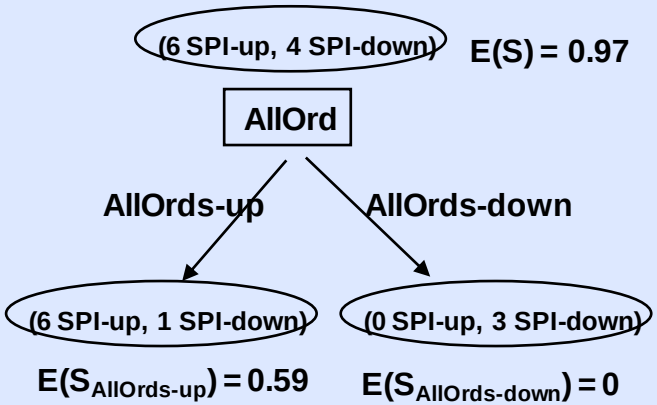
  - **The entropy (impurity) is 1 if S contains an equal number of positive (up) and negative (down) cases** 
  - **If S contains unequal numbers of positive and negative cases, the entropy is between 0 and 1**

Basic Tree Builder …. (contd.)

- **Information Gain** measures the expected **reduction in entropy** (impurity) when we partition S using an attribute A

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- **Gain(S, A) = Entropy(S) – weighted average of entropies (impurities) of Subsets**



Gain(S, AllOrd) = E(S) - (7/10)*E($S_{AllOrds-up}$)
- (3/10)*E($S_{AllOrds-down}$)
= 0.97 - (0.7)*(0.59)
- (0.3)*(0)
= 0.557

Basic Tree Builder …. (contd.)

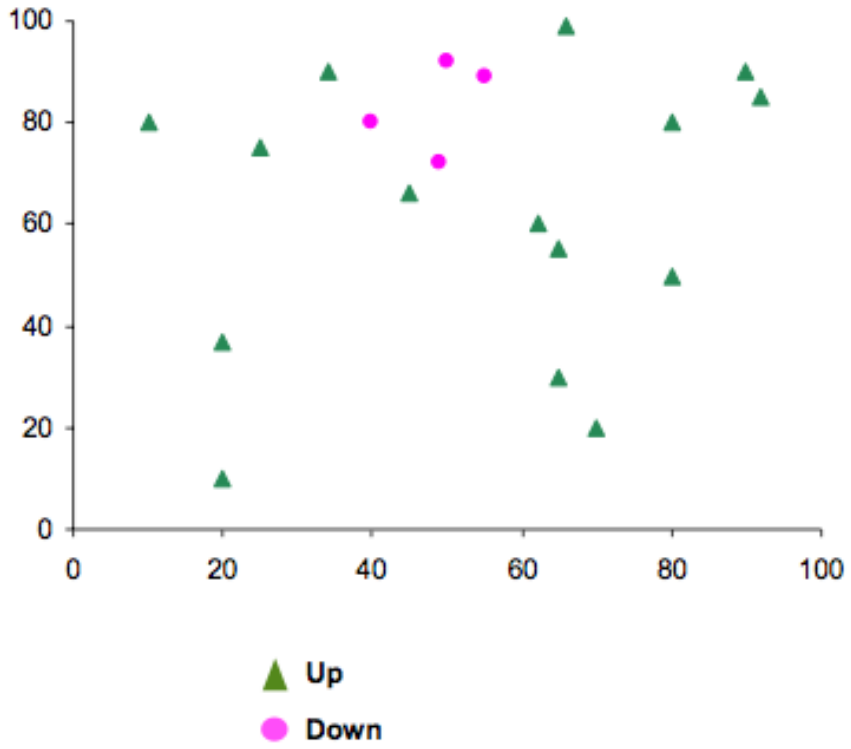
- **Calculate** the information gain for every attribute
- **Select** the attribute that produces **maximum information gain** and partition the data using that attribute
- **Terminate** if a "pure" node is reached or no attribute increases information gain

A Simple Example

We can visualise the following simple data set using a 2-dimensional graph on the right hand side

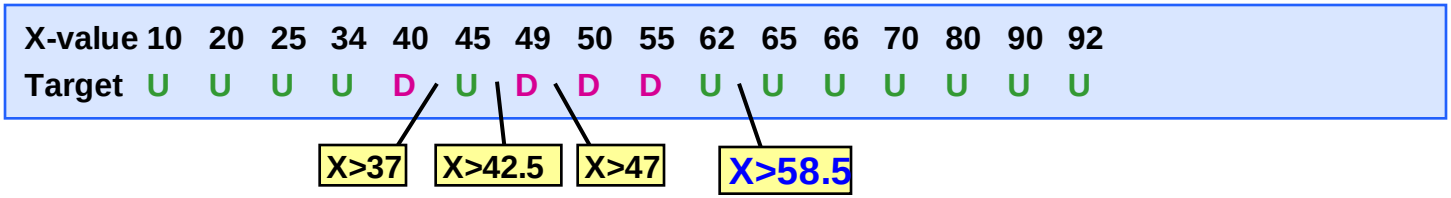
Training Data Set (S)

X-value	Y-value	Target
20	10	Up
70	20	Up
65	30	Up
20	37	Up
80	50	Up
65	55	Up
62	60	Up
45	66	Up
49	72	Down
25	75	Up
10	80	Up
40	80	Down
55	89	Down
90	90	Up
34	90	Up
50	92	Down



A Simple Example …(contd.)

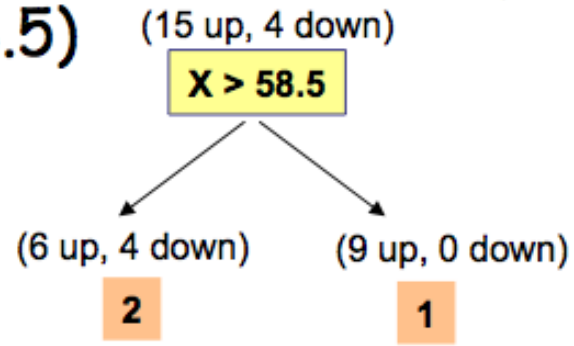
- Calculate the information gain for every possible split position of the attribute X
  - possible split positions for the attribute X

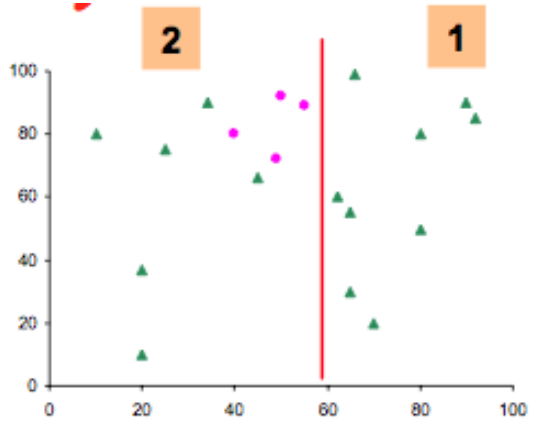


- Similarly, calculate the information gain for every possible split position of the attribute Y
- The **attribute with maximum information gain** (the split position with max information gain) is selected and the data set S is partitioned accordingly

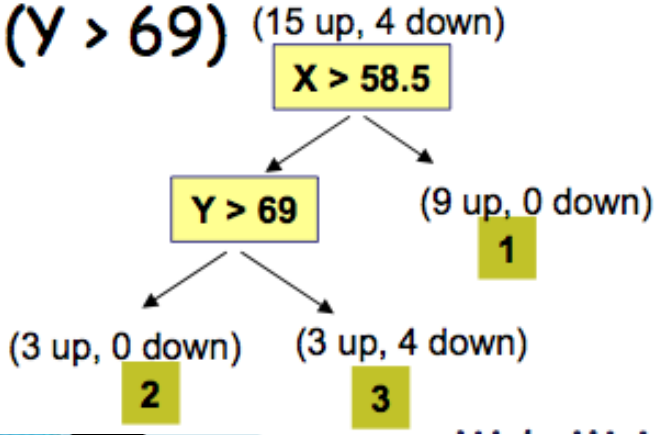
A Simple Example …(contd.)

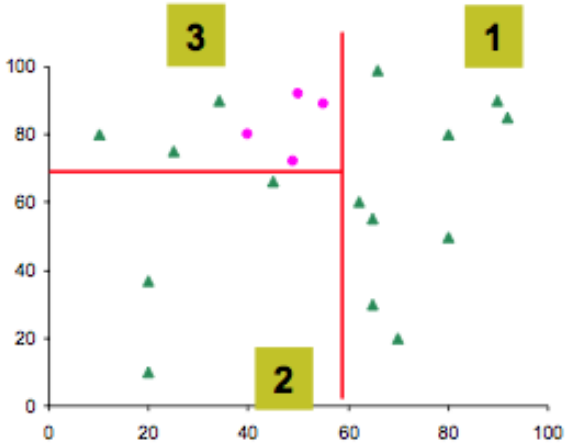
The tree after the **first split** for (X > 58.5)



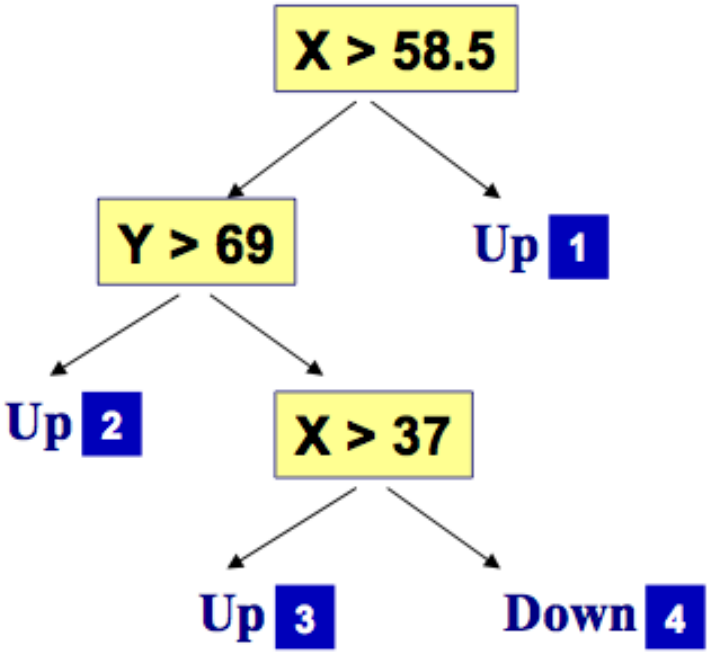


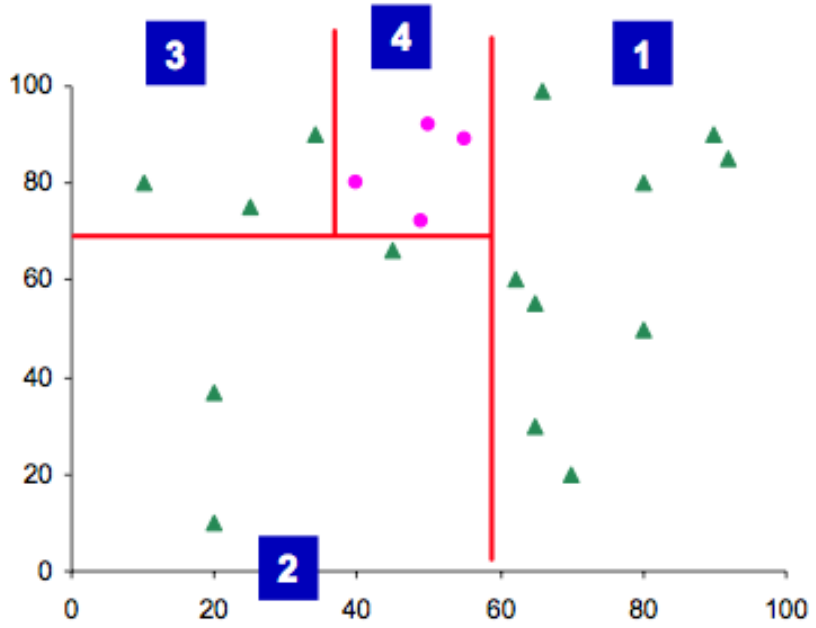
The tree after the **second split** for (Y > 69)





A Simple Example: .... The Final Tree





Rule-1: IF (X > 58.5) THEN Up
Rule-2: IF (X <= 58.5) AND (Y <= 69) THEN Up
Rule-3: IF *(X <= 58.5)* AND (Y>69) AND (X <= 37) THEN Up
Rule-4: IF (X <= 58.5) AND (Y>69) AND (X > 37) THEN Down

What If the Target is Numeric?

- Each leaf node stores either
  - mean value of cases in the leaf; or
  - a **multivariate linear model** for the cases at that leaf, and this model is used to predict the value
- Example,

```
IF (X <= 38.5) THEN LM1
IF (X >  38.5) AND (Y <= -14 )
     THEN LM2
IF (X >  38.5) AND (Y > -14 )
     AND (Y <= 1.5 ) THEN LM3
IF (X >  38.5) AND (Y > -14 )
     AND (Y > 1.5 ) THEN LM4
```

Models at the leaves:

LM1: A = 2.74 – 0.026Y

LM2: A = 30.7 – 0.362Y

LM3: A = –884 + 0.318Z – 0.253Y

LM4: A = 15.8 – 1.22Y

What If the Target is Numeric? …. (contd.)

- Splitting criterion used is **standard deviation** of the target values of cases in the node (as a **measure of the error**)
- The expected **error reduction** (standard deviation reduction) can be calculated using

$$\Delta error = sdev(S) - \sum_i \frac{|S_i|}{|S|} * sdev(S_i)$$

where S_i are sets that result from splitting

- The attribute which maximises the expected error reduction is selected

Issues for a Tree Builder

- Selecting a **splitting criterion.** Few different criteria are available:
  - Information Gain, Gini Index, Gain Ratio, etc.
- Number of splits allowed at each level of the tree (2 or 3 or ….)
- Depth / height of the tree
- Avoid **over-fitting** the training data
- Handle **missing values** for attributes
- Estimate error on new (unseen) data

Avoiding Over-Fitting

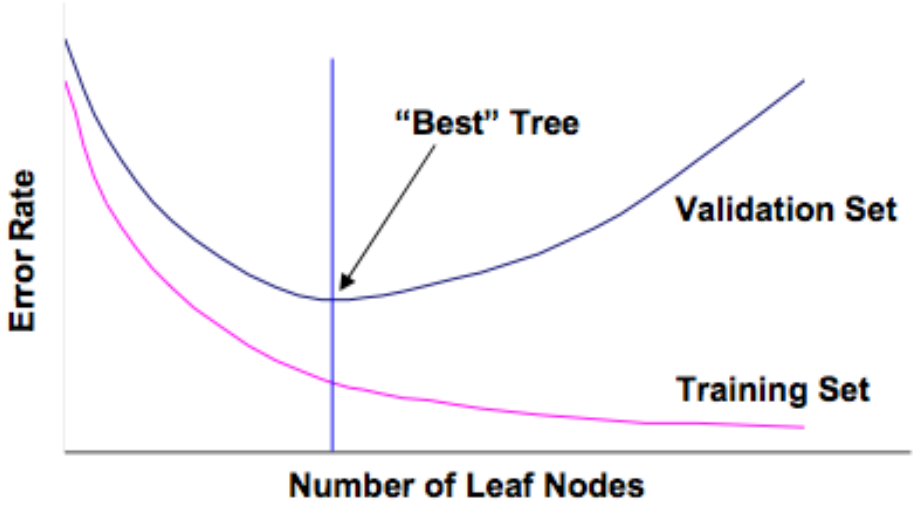
- Selecting the "best" tree:

Data Set is divided in to three sets:

- Training
- Validation

} used for model building

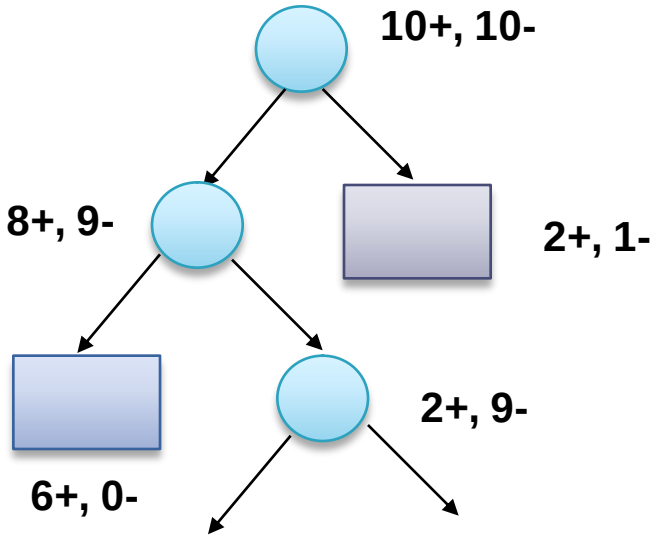
- Test



- **Occam's razor**: Prefer the simplest hypothesis (i.e., model) that fits the data

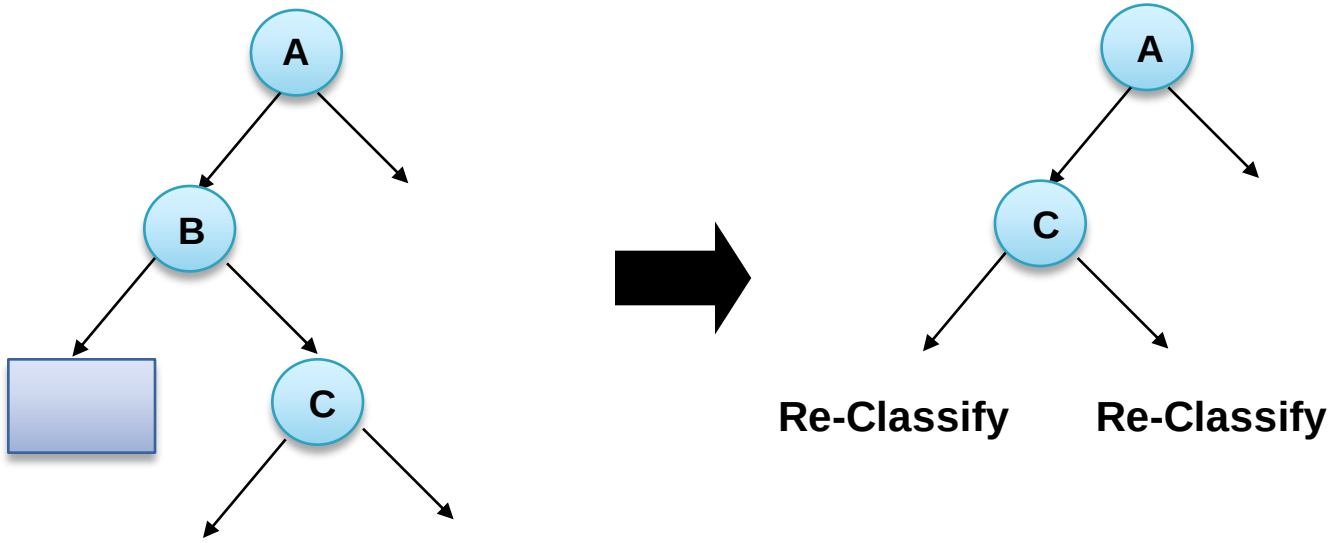
Over-fitting Avoidance by "Pruning"

- **"Forward" pruning**:
  - stop when number of cases in a node reach some pre-defined value (or use some statistical test like χ^2)



Pruning
…. (contd.)

- **“Backward” pruning**:
  - replace nodes by leaves
- Sub tree lifting:



Computational Complexity

- **Assume**: N instances each with M attributes and tree depth **O(log N)**
- At each tree depth, we have to consider, for each attribute, the classification of all N instances. The overall cost for building the tree is thus: **O(MN log N)**
- When pruning, each node has to be considered for replacement by a leaf. The tree can have at most N leaves. For binary trees this would mean at most 2N-1 nodes i.e. O(N). For sub tree lifting, each node is considered for replacement: this is O(N). For each replacement an instance may have to be re-classified at each level of the tree. This is **O(N log N)** reclassifications. A reclassification requires at most **O(log N)** operations.
- **Total cost:** **O(MN log N) + O(N (log N)^2)**

Bayesian Networks

The Bayes framework

- Probability theory predict the **likelihood of different outcomes.**
- **Bayes theorem** give a formula for calculating a *conditional probability based on the reverse condition that sometime is easier*.
- **Conditional probability**:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

The Bayes framework (II)

- ***The Bayes Theorem***

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Expanding total probability for calculating $P(A=a_i|B)$:

$$P(A = a_i|B) = \frac{P(B|A = a_i)P(A = a_i)}{\sum_k P(B|A = a_k)P(A = a_k)}$$

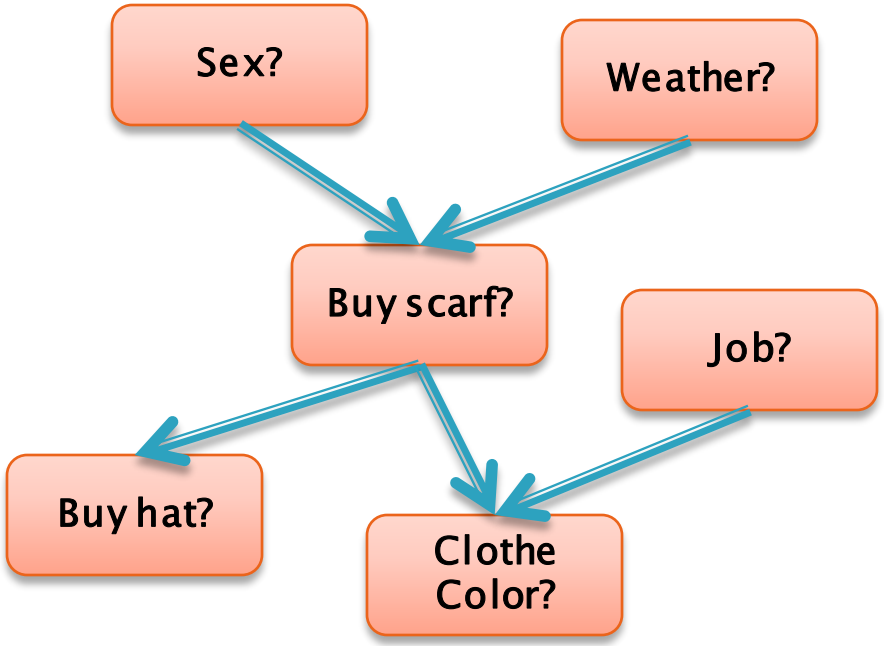
The Bayes framework (III)

- **Bayesian Network**
  - If **P(A|B)** if "**highly probable**" **is equivalent to** [**A=>B**] association Rule
  - A graph of relation [A=>B] is called "Bayesian Network"

- **Example**

If(sex=male & weather=rainy)
Then buy scarf is true with 80% prob.

P(buy scarf=yes|gender=male, weather=rainy)=0.8



Machine Learning (ML) Algorithms as Bayesian Learners

- So far, we have looked at **algorithm-independent methods** for *evaluating and comparing models.*
- ML algorithms can be seen as **algorithms that seek answers** to one or both of the following:
  - What is the **most probable hypothesis** (model) **given a set of 'training' data**?
  - What is **most probable classification of new data** instances?
- **Bayesian reasoning** provides the basis for answering these questions
  - Many choices made by practical ML algorithms can be better understood within Bayesian setting.

Looking at the Bayes Rule

Probability that hypothesis h holds given the observed training data D

Prior probability of a hypothesis h

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

Probability of observing data D if hypothesis h holds

Prior probability of observing the training data D

$$P(h|D) \propto P(D|h) \times P(h)$$

Maximum A Posteriori (MAP) Hypothesis

- The **most (maximally) probable hypothesis**, given the data is called the *maximum a posteriori (or MAP) hypothesis*:

$$h_{MAP} = argmax_h P(D|h) \times P(h)$$

Maximum Likelihood Hypothesis

- The term $P(D|h)$ is called the ***'likelihood'*** of the data, given the hypothesis. If **all hypotheses are equally likely**, then the MAP hypothesis reduces to the *maximum likelihood or ML hypothesis*:

$$h_{MAP} = h_{ML} = argmax_h P(D|h)$$

A Different View of MAP

$$h_{MAP} = argmin_h \; -\log_2 P(D|h) \; -\log_2 P(h)$$

bits required to encode D given h using an optimal code

bits required to encode h using an optimal code

The Minimum Description Length (MDL) Principle

1. Choose a scheme for encoding hypotheses (C1) and for encoding data given a hypothesis (C2)
2. The ***MDL hypothesis*** is:

$$h_{MDL} = argmin_h [L_{C_1}(h) + L_{C_2}(D|h)]$$

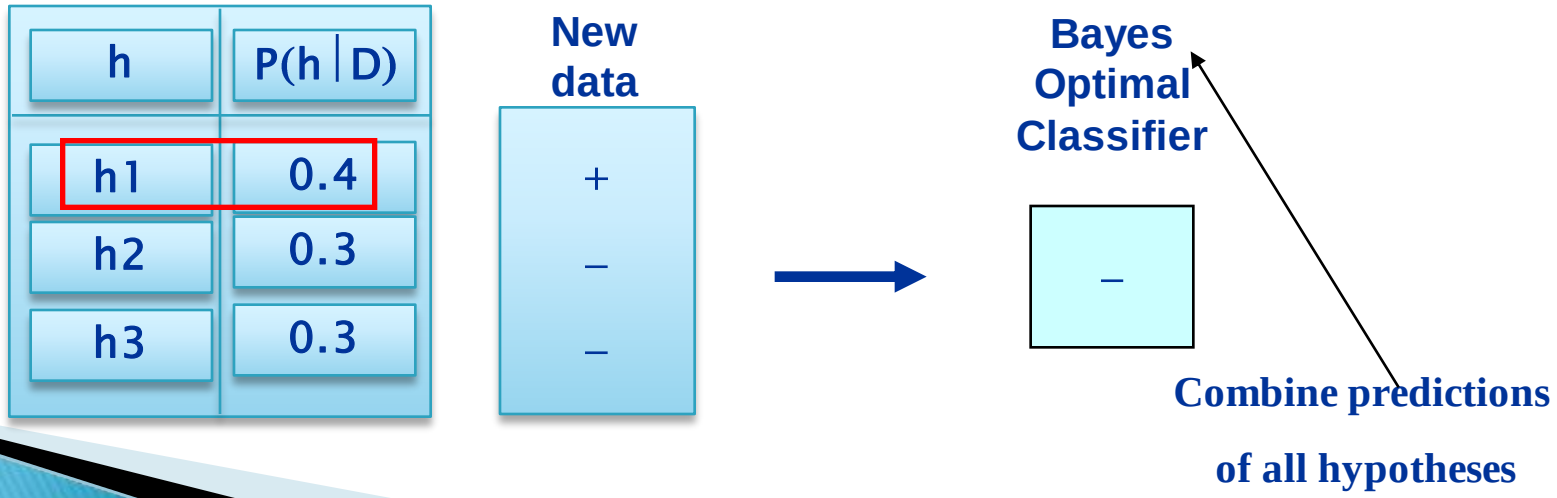
If C_1 and C_2 are optimal, then
$h_{MDL} = h_{MAP}$

Machine Learning Algorithms as MAP Learners

- Many choices made by particular ML algorithms translate to particular assumptions within the Bayesian framework for identifying $\mathbf{h_{MAP}}$
  - A NN that tries to find the **model that minimises MSE on the training set** is equivalent to a **Maximum Likelihood hypothesis** *under the assumption about noise in the training data*. Recall that this means that it is a **MAP hypothesis that assumes all hypotheses** (models) are **equally likely.**
  - A **Classification Tree** that tries to balance the size of the tree with the errors made on the training set can be seen as an **algorithm that is employing the MDL principle.** The result is some approximation to the **MAP hypothesis.**

Is the MAP Hypothesis Best?

- For a given problem (given the space of hypotheses and prior information),
  - *Will the MAP hypothesis result in the lowest error when predicting new data?*
- **Surprisingly**: **No!** The *'Bayes Optimal Classifier'* is one that classifies the new data instance by combining the predictions of *all* the hypotheses (not just the MAP hypothesis)



K-Nearest Neighbour

K-Nearest Neighbour Learning

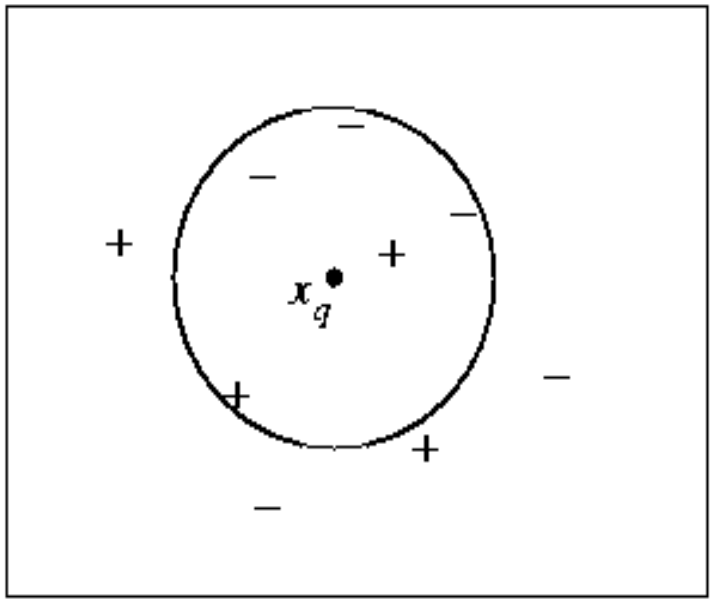
- 1-Nearest Neighbour:

  Given a query (test) instance x_q,
  - locate the **nearest training example** x_n and assign $f(x_q) = f(x_n)$
- K-Nearest Neighbour:

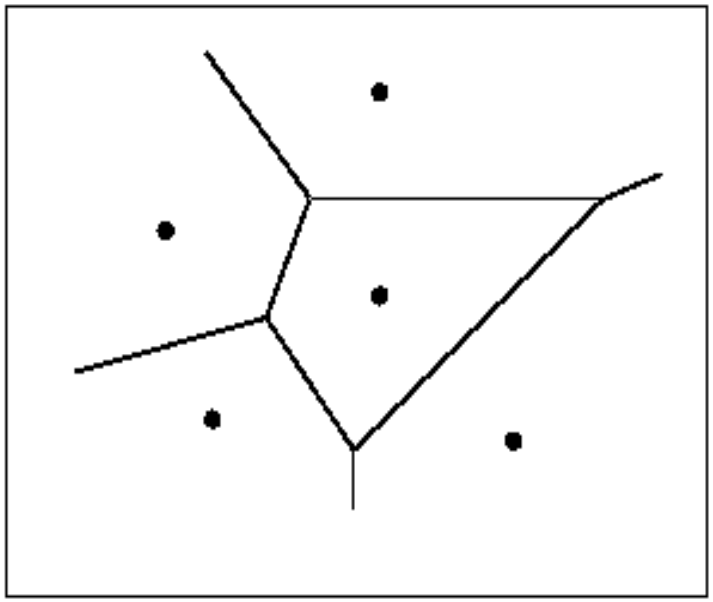
  Given a query (test) instance x_q,
  - locate the **k-nearest training examples**
  - if the **target function has discrete values**, then return the **most common value** of f among the k nearest training examples;
  - if **continuous-valued target function**, then return the **mean of the f values** of the k nearest training examples.

$$f(x_q) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

K-Nearest Neighbour – Example



5-NN classifies x_q as -, 1-NN as



Voronoi diagram

- If the target function is continuous-valued, kNN calculates the mean of the k nearest neighbours

How to measure distance between examples

- We need a **distance measure** in order to **know who are the neighbours**
- Assume that we have *M* attributes (features). Then one example point x_i is described by a feature vector $<x_{i1}, x_{i2}, \ldots, x_{iM}>$.
- The distance between two points x_i and x_j is usually defined in terms of **Euclidean distance**:

$$d(x_i, x_j) = \sqrt{\sum_{f=1}^{M} [x_{if} - x_{jf}]^2}$$

When use K-Nearest Neighbour algorithm?

- **Not more than say 20 attributes per instance**
- ***Advantages:***
  - Training is very **fast**
  - Can learn **complex target functions**
  - Do **not lose any information** contained in the training set
- ***Disadvantages:***
  - During classification, kNN has to calculate the **distances between the test case and *all* training examples**
  - Some attributes may be **irrelevant**

Version of kNN:
Distance-Weighted kNN

- Weight **nearer neighbours** more heavily:

$$f(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

where

$$w_i = \frac{1}{d(x_q, x_i)^2}$$

- This allows us to use *all training examples* instead of just k (**Sheppard's method**), and we call this a global learning method. If only the nearest examples are used, we have a local learning method.

Support Vector Machines

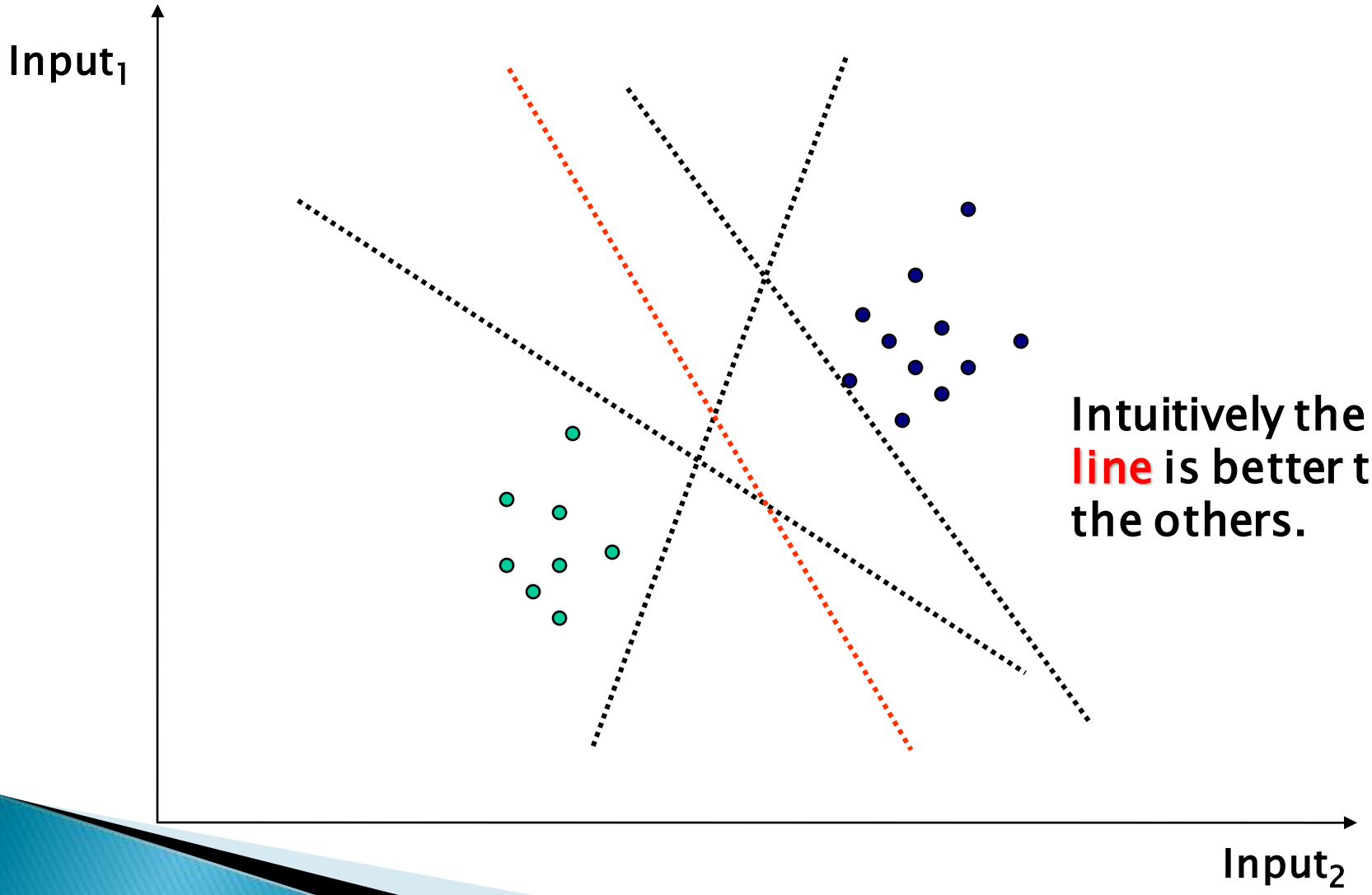
Support Vector Machines

- An **effective machine learning technique**; one the **most important recent discovery** in machine learning
- SVMs are based on Vapnik's work and were introduced in early '90s.
- Decision surface for classification is a **hyperplane** in the **feature** space (*a line in 2D case*).
- **Convert problems into linearly separable problems** by *transforming the input space into a higher dimensional feature space*.

Support Vector Machines: Basic ideas

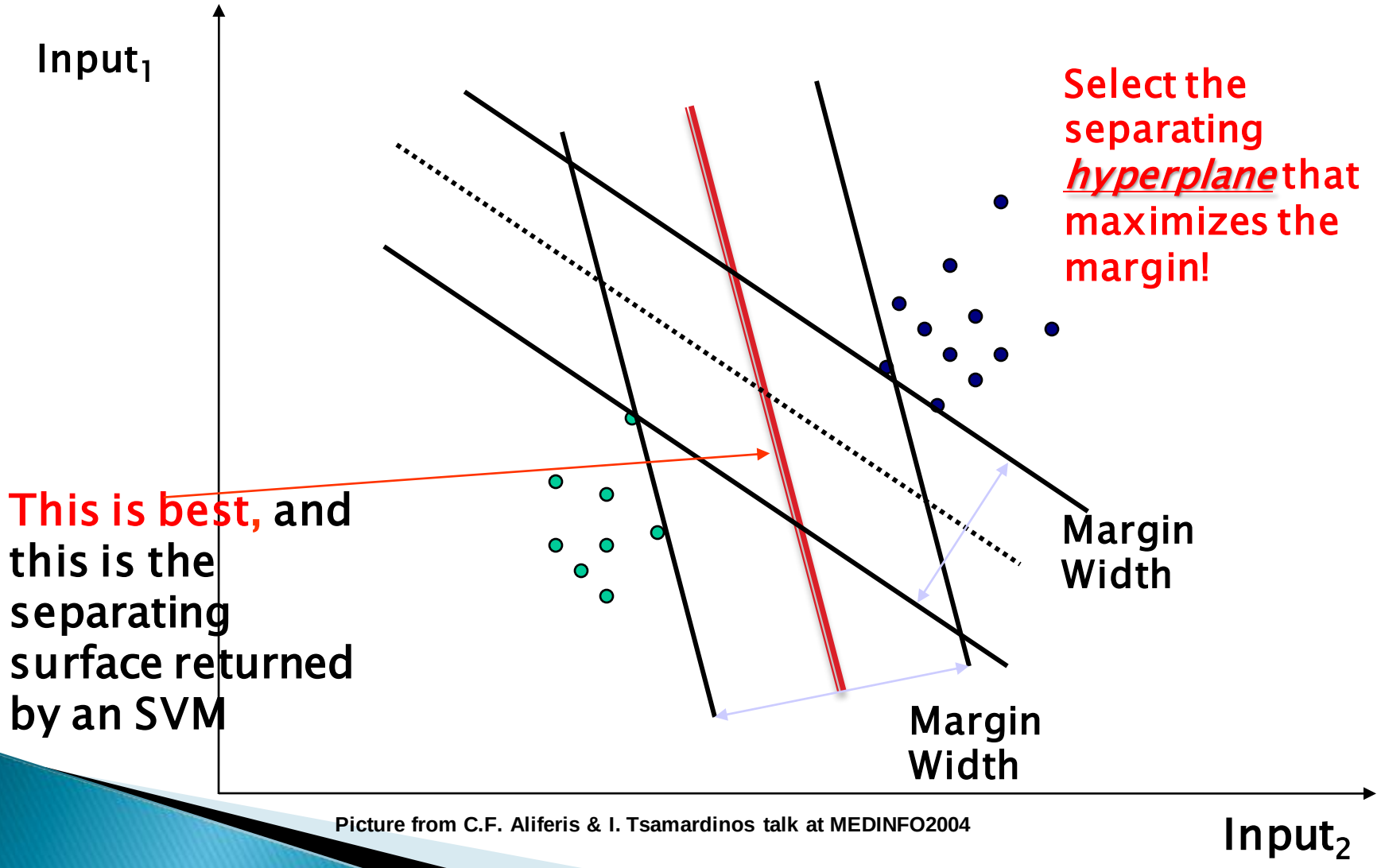
- **Map data onto a high dimensional space** via a ***kernel function*** where *data can be classified with linear decision surfaces*
- Find the **"optimal" hyperplane** that *maximizes the degree of separation between the two classes* (***maximizes the margin*** between the two classes)
- If data are not linearly separable in a given space ***find the hyperplane that maximizes the margin and also minimizes the number of misclassifications***

Which Separating Hyperplane to Use?



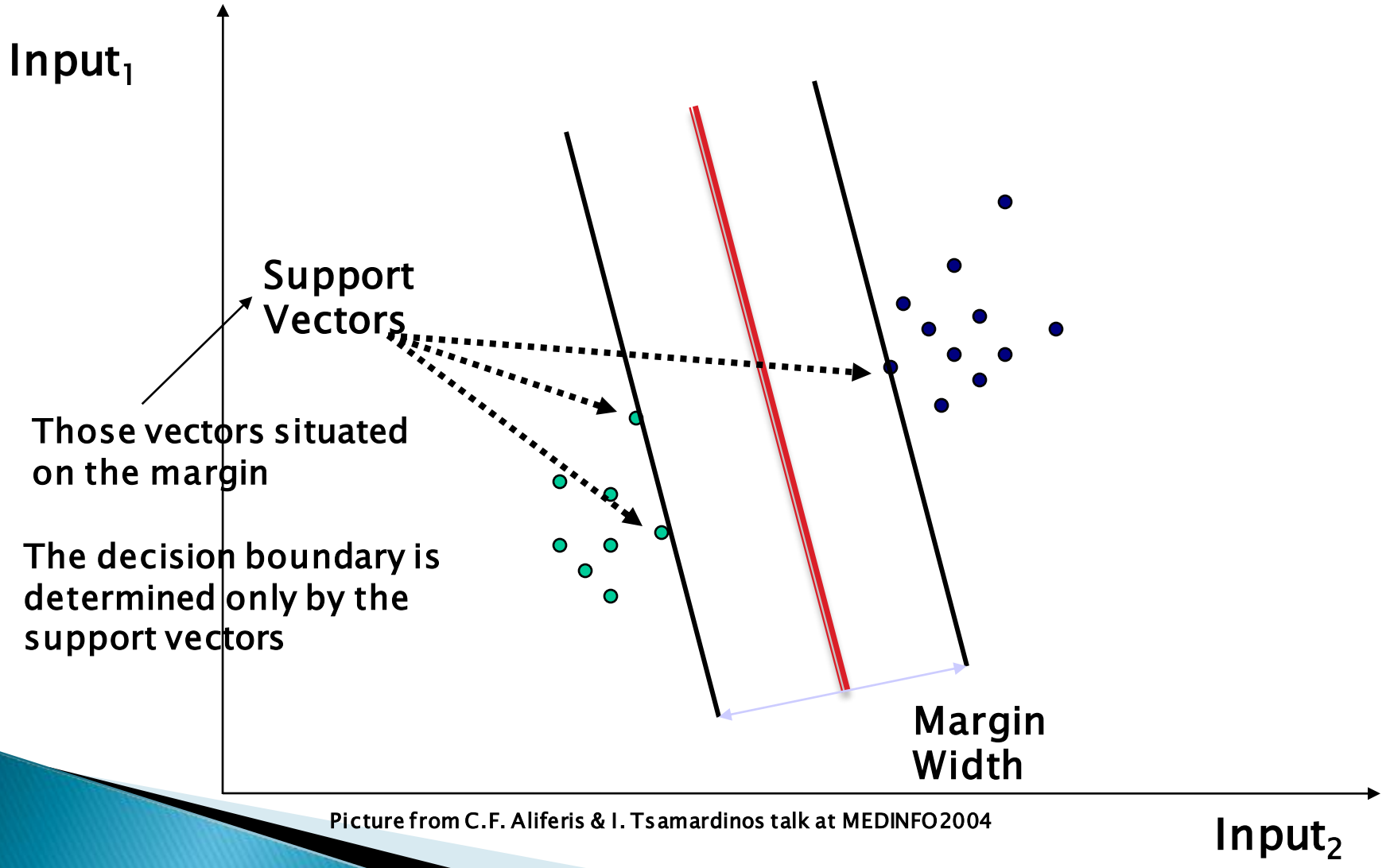
Intuitively the red line is better than the others.

Maximize the Margin



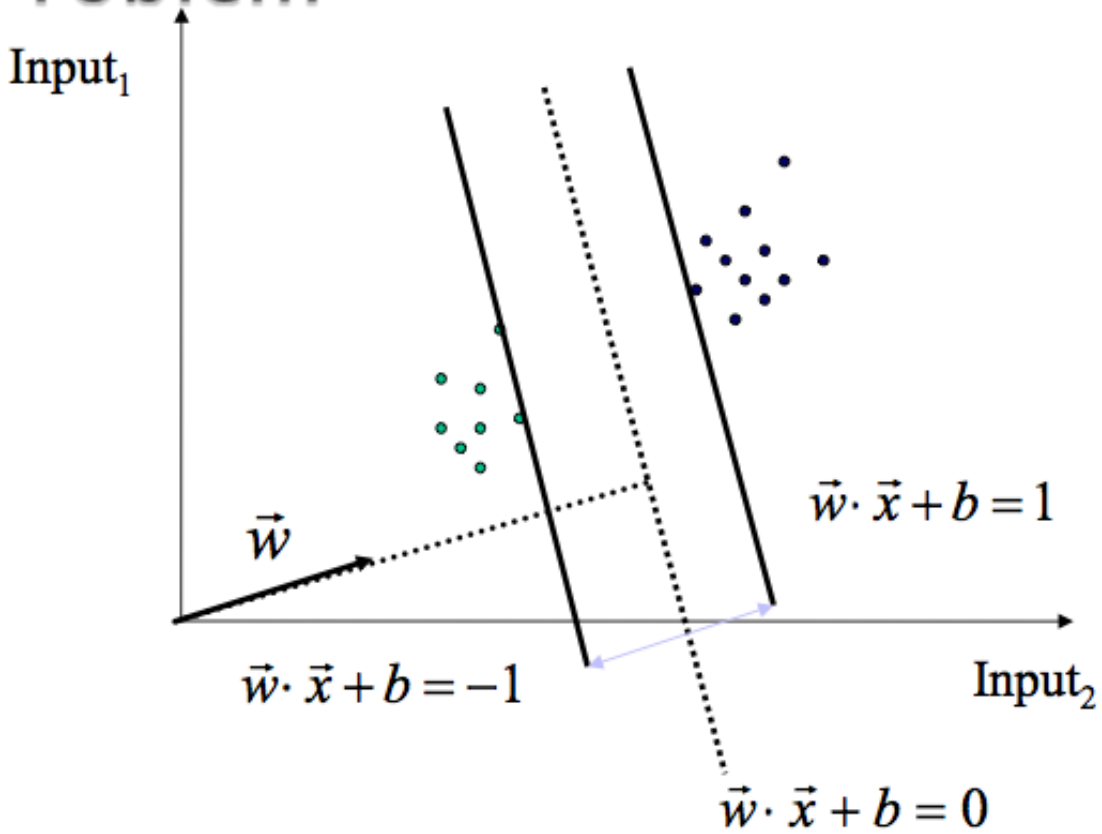
Picture from C.F. Aliferis & I. Tsamardinos talk at MEDINFO2004

What are Support Vectors?



Picture from C.F. Aliferis & I. Tsamardinos talk at MEDINFO2004

Formulation of the Optimization Problem



$$\max \frac{2}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq 1,\ \forall x \text{ of class 1}$$

$$(w\cdot x+b)\leq -1,\ \forall x \text{ of class 2}$$

The distance between the origin and the line wx+b=k is k/||w||, where k=1, or -1.

Then *margin* = 2/||w|| and we should maximize it.

Formulation of the Optimization Problem

- If class 1 is "1" and class 2 is "−1" we have:

$$\begin{aligned}(w * x_i + b) &\geq \quad 1, \quad \forall x_i \; with \; y_i = \quad 1 \\ (w * x_i + b) &\leq -1, \quad \forall x_i \; with \; y_i = -1\end{aligned}$$

$$y_i(w * x_i + b) \geq 1, \forall x_i$$

- So the problem becomes:

$$\max_{y_i(w*x_i+b)\geq 1, \forall x_i} \frac{2}{\|w\|}$$

or

$$\min_{y_i(w*x_i+b)\geq 1, \forall x_i} \frac{1}{2}\|w\|^2$$

Linear, Hard-Margin SVM Formulation

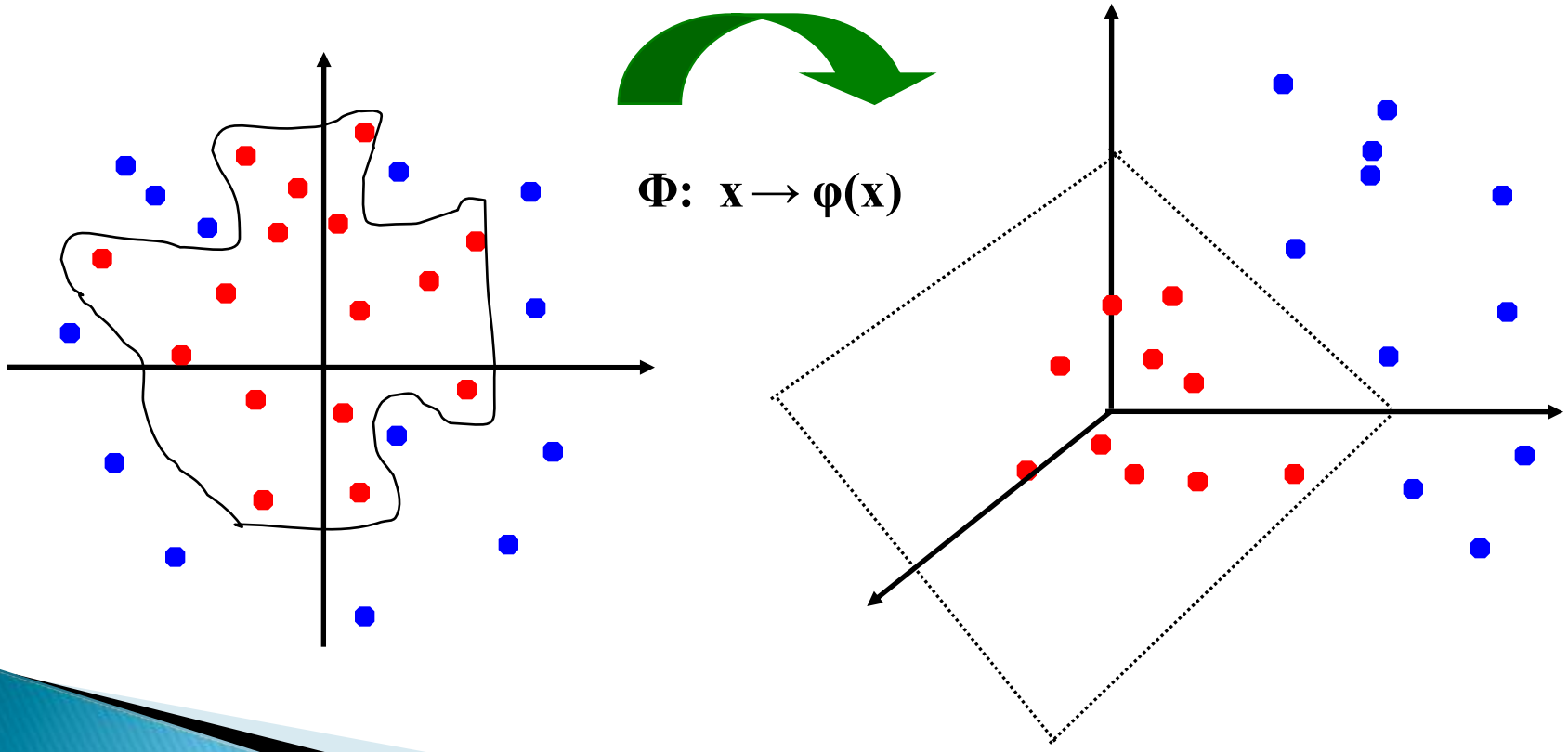
- Find *w, b* that solves

$$\min_{y_i(w*x_i+b)\geq 1, \forall x_i} \frac{1}{2}\|w\|^2$$

- **Problem is convex** so, there is a **unique global minimum value** (when feasible)
- There is also a **unique minimizer**, i.e. *weight and b value that provides the minimum*
- ***What if the data is not linearly separable?***
- **Answer**: We may *allow some examples to fall within the margin but penalize them* (using a so called **soft margin**); or see next slide.

How to solve non-linearly separable problems?

- **Basic idea:** the original input space is mapped to some **higher-dimensional feature space** (using a kernel function) *where the training set is linearly separable*



Why do SVMs work?

- The feature space is often very high dimensional. Why don't we have the curse of dimensionality?
- *A classifier in a high-dimensional space has **many parameters and is hard to estimate**.*

- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the **flexibility of a classifier**

From C.F. Aliferis & I. Tsamardinos talk at MEDINFO2004

Why do SVMs work?

- The flexibility of a classifier should not be characterized by the number of parameters, but by the flexibility (capacity) of a classifier, formalized by the "VC-dimension" of a classifier.
- The **higher the VC-dimension**, the **more flexible** a classifier is
- In practice, **the VC-dimension may be difficult to be computed exactly**

Structural Risk Minimization (SRM) problem

- SRM means **we should find a classifier that minimizes the sum of training error**
  - Called empirical risk
- as well as a term that is a **function of the flexibility of the classifier**
  - Called model complexity

Choosing the Kernel Function

- **It is the most tricky part of using SVM**. It is *equivalent to choosing the number of hidden nodes for a neural network*.
- In practice, **start with a low degree polynomial kernel function** or **RBF kernel function with a reasonable width**
- Note that **SVM with RBF kernel is closely related to RBF neural networks**, where the centers of the radial basis functions are automatically chosen for SVM

Advantages of SVMs

- Training is **relatively easy** – **No local optima**
  - (very good compared to neural networks)
- **It scales relatively well to high dimensional data**
- The **tradeoff** *between classifier complexity and error* (empirical risk) can be **controlled explicitly**
- But we need to choose a **"good" kernel function!!!** **This may not be so easy.**

Section 3.5

- Using data mining to extract knowledge

Using data mining to extract knowledge

- ¿How the use the **prediction** for knowledge?
- ¿How to find the **rules implied** by the **generated models**?
- ¿How to **model human behavior**?

Rules: Knowledge for humans

- Human usually understand better **knowledge expressed in the form of "Rules"**
- **EEUU legislation** doesn't **allows credit assignations** based in **black box predictors**.
- **Decision Trees** and **Bayesian network** **throws directly a set of rules.**
- **ANN are black box predictors** but recent discovery allows us to extract rules from them
  - V. Palade, S. Bumbaru, M.G. Negoita (1998). "A method for compiling neural networks into fuzzy rules using genetic algorithms and hierarchical approach", Proceedings of the 2nd IEEE International Conference on Knowledge-Based Intelligent Electronic Systems- KES1998, vol. 2 pp. 353-358, Adelaide - Australia, 1998.

Web user in the new economy

- **Association rules and ANNs**:
  - **predicting next web page to be visited** in a session path.
- **Virtual Shopping**
  - prediction are useful for
    - **displaying desired information**
    - direct **hyperlink to similar product** preferred by others
    - etc.

User behavioural

- **Clustering techniques**
  - Segment into *group of web users*.
- **Marketing**
  - *opinion poll* or *market sample survey*
    - market segmentation
- We could **predict the behavior of users** in a **same cluster**.

Section 3.8

- Mining the Web

Data mining techniques

- Association rules
- Clustering
- Classification
- Prediction
- Probabilistic models
- Regression

What happen with web data?

"The web is a huge collection

of heterogeneous, unlabeled, distributed, time variant,

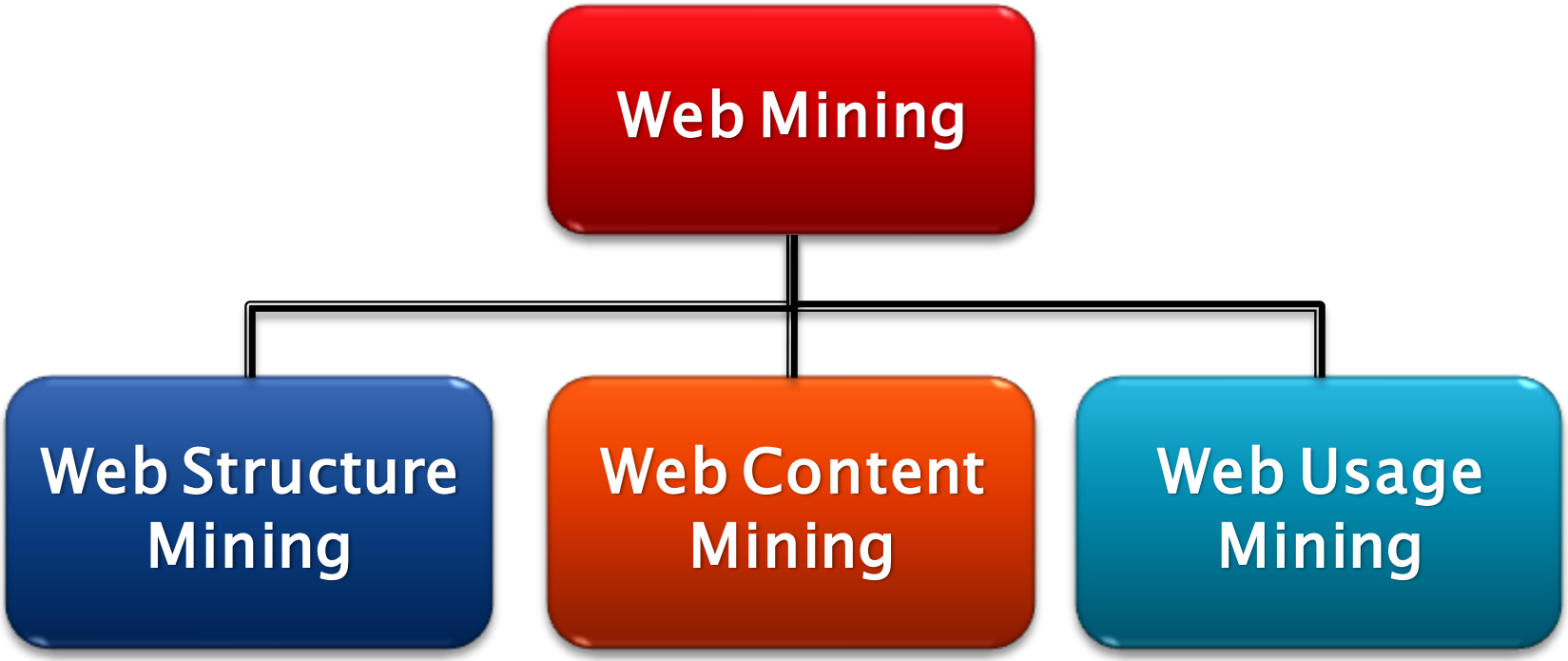
semi-structured and high dimensional data"

S.K. Pal 2002

Mining the web

- Web mining techniques are the **application of data mining** theory in order to **discovery patterns from web data.**
- Web mining must consider three important steps:
  1. Pre-processing
  2. Pattern discovery
  3. Pattern analysis

Web Mining Taxonomy [Jooshi00,Lu03]



Web Structure Mining (WSM)

- It deals with the **mining of the web hyperlink structure** (*inter document structure*).
- A **website is represented** by a **graph of its links**, within the site or between sites.
- Facts like the **popularity of a web page** can be studied, for instance, if a page is referred by a lot of other pages in the web.
- The web link structure allows to develop a **notion of hyperlinked communities**.
- It can be used by search engines, like GOOGLE or ALTAVISTA, in order to get the **set of pages more cited for a particular subject**.

WSM (2)

- To discover the **link structure of the hyperlinks** at the **inter-document level** to generate **structural summary about the Website and Web page**.
  - ***Direction 1***: based on the hyperlinks, categorizing the Web pages and generated information.
  - ***Direction 2***: discovering the structure of Web document itself.
  - ***Direction 3***: discovering the nature of the hierarchy or network of hyperlinks in the Website of a particular domain.

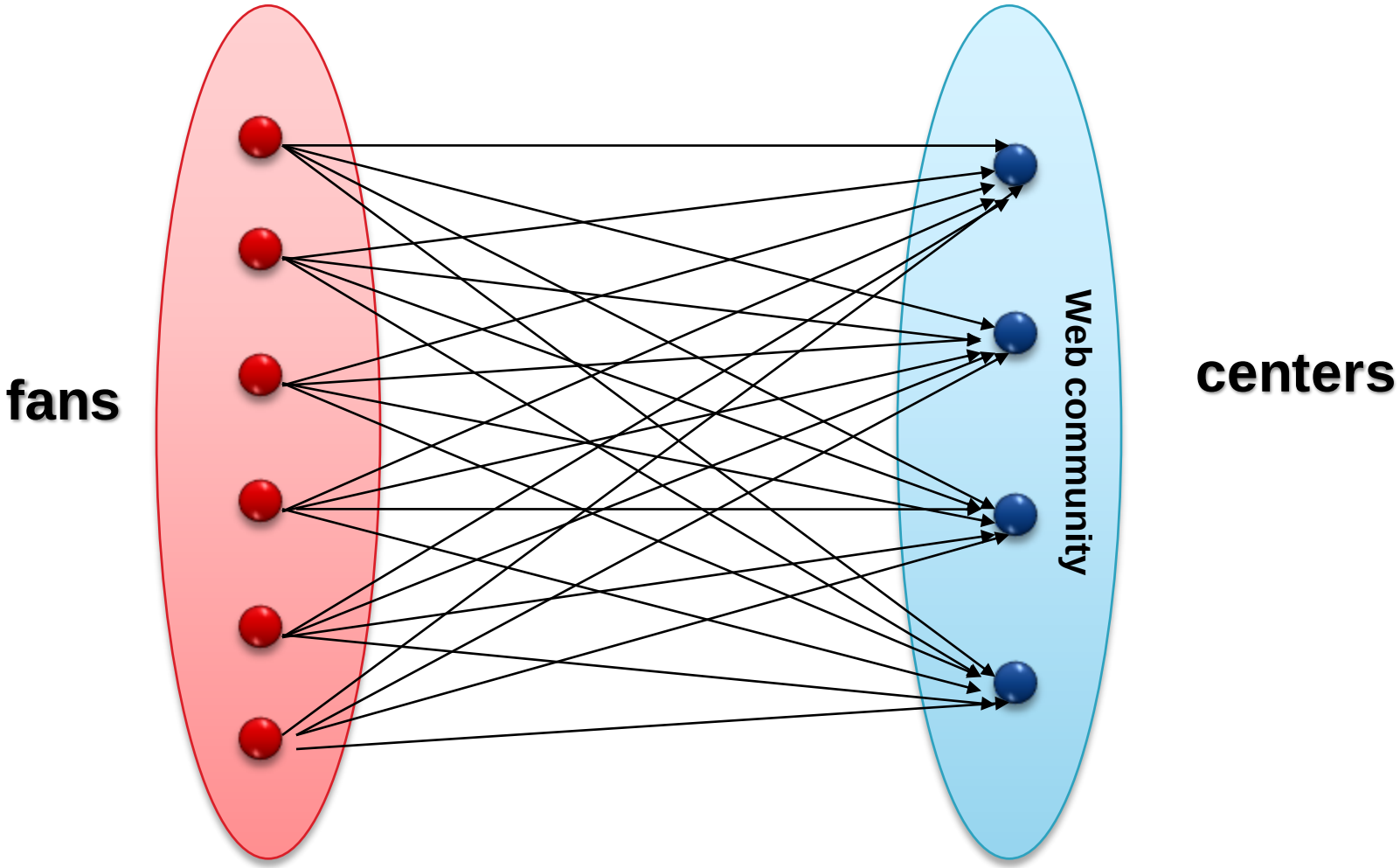
WSM (3)

- Finding **authoritative web pages**
  - Retrieving pages that are not only relevant, but also of **high quality**, or **authoritative on the topic**
- Hyperlinks can infer the **notion of authority**
  - The Web consists not only of pages, but also of hyperlinks pointing from one page to another
  - These hyperlinks contain an enormous amount of **latent human annotation**
  - A hyperlink pointing to another web page, this can be considered as the **author's endorsement of the other page**

WSM (4)

- Web pages **categorization**
  - (Chakrabarti, et al., 1998)
- Discovering **micro communities** on the Web
  - Example:
    - Clever system (Chakrabarti, et al., 1999)
    - Google (Brin and Page, 1998)
- **Schema Discovery** in Semi-structured Environment

WSM: Example



Web Community Centers: many web pages go there

Web Content Mining (WCM)

- The goal is to find **useful information from the web content**.
- In this sense, *WCM is similar to Information Retrieval (IR)*.
- However, *web content is not only free text*, other objects like *pictures*, *sound* and *movies* belong also to the content.
- There are two main areas in WCM :
  - the **mining of document contents** (web page content mining)
  - **the improvement of content search** in tools like search engines (search result mining)

WCM (2)

- **Web content**
  - *text, image, audio, video, metadata and Hyperlinks*
- **Information Retrieval View (Structured + Semi-Structured)**
  - Assist / Improve information finding
  - Filtering Information to users on user profiles
- **Database View**
  - Model Data on the Web
  - Integrate them for more **sophisticated queries**

WCM (3)

- Developing **Web query systems**
  - WebOQL
  - XML-QL

- Mining **multimedia data**
  - Mining **image** from satellite
    - (Fayyad, et al. 1996)
  - Mining **image to identify small volcanoes on Venus**
    - (Smyth, et al 1996)

Issues in Web Content Mining

- Developing intelligent tools for IR
  - Finding **keywords** and **key phrases**
  - Discovering **grammatical rules** and collocations
  - **Hypertext classification/categorization**
  - Extracting **key phrases** from text documents
  - Learning **extraction models/rules**
  - **Hierarchical clustering**
  - Predicting **(words) relationship**

Web page in Vector Space Model

- Each web page can be consider as a document text with tags.
- Applying filters, the web page is transformed to feature vectors
- Let P = {p_1, …,p_Q} be the set of *Q* pages in a web site.
- The i-th page is represented by

$$wp^i = (wp_1^i, ..., wp_R^i) \in WP$$

with *R* the amount of words after a stop word and stemming process.

Web Usage Mining (WUM)

- Also known as **Web log mining**
  - mining techniques to **discover interesting usage patterns** from the secondary data derived from the **interactions of the users while surfing the Web**

WUM – Considerations (Pierrakos03)

- WUM applies traditional data mining methods in order to **analyze usage data**.
- The **sessionization process** is necessary to **correct the problems** detected in the data.
- The goal is to discover **patterns in usage data** applying different kinds of data mining techniques.
- Applications of WUM can be grouped in two main categories:
  - **User modelling in adaptive interfaces**, known as personalization.
  - **User navigation patterns**, in order to improve the *web site structure*.

WUM: Applications

- Target potential customers for electronic commerce
- Enhance the quality and delivery of Internet information services to the end user
- Improve Web server system performance
- Identify potential prime advertisement locations
- Facilitates personalization/adaptive sites
- Improve site design
- Fraud/intrusion detection
- Predict user's actions (allows pre-fetching)

Summary

- KDD process
  - historic data +ETL + Data mining.
- Data mining:
  - Clustering
  - Association rules
- Tools:
  - ANNs, Trees, Rules, Bayes N.