MACHINE LEARNING

Course code:ACS014

IV. B. Tech VIII semester

Regulation: IARE R-16

BY

Mrs. G Sulakshana

Assistant Professors

Mr. A Praveen, Mrs. B Anupama

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

DUNDIGAL, HYDERABAD - 500 043

CO’s	Course outcomes
CO1	Understand the concept of learning and candidate elimination algorithms.
CO2	Understand the concept of perception and explore on forward and backward practices
CO3	Explore on basic statistics like variance, covariance and averages
CO4	Explore on Evolutionary learning techniques used in genetic algorithms
CO5	Explore on similarity concept and different distance measures

CLOs	Course Learning Outcome
CLO1	Understand the concept of learning and candidate elimination algorithms.
CLO2	Explore on different types of learning and explore On tree based learning.
CLO3	Understand the construction process of decision trees used for classification problem.
CLO4	Understand the concept of perception and explore on forward and backward practices.
CLO5	Illustrate on kernel concept and optimal separation used in support vector machines
CLO6	Explore on basic statistics like variance, covariance and averages
CLO7	Understand the concepts of Gaussian and bias- variance tradeoff

CLOs	Course Learning Outcome
CLO8	Understand the concepts of Bayes theorem and Bayes optimal classifiers
CLO9	Explore on Bayesian networks and approximate inference on markov models
CLO10	Explore on Evolutionary learning techniques used in genetic algorithms
CLO11	Illustrate the ensemble learning approaches used in bagging and boosting
CLO12	Explain the importance of principal component analysis and its applications
CLO13	Explore on similarity concept and different distance measures
CLO14	Understand the outlier concept and explain about data objects

CLOs	Course Learning Outcome
CLO15	Understand the hierarchical algorithms and explain CART
CLO16	Understand the partitioned algorithms and explain segmentation
CLO17	Explore on clustering large database and explain K-means clustering algorithm
CLO18	Understand the clustering with categorical Attributes and comparison with other data types.
CLO19	Understand the clustering large databases and explain clustering methods
CLO20	Describe clustering with categorical attributes and explain KNN

UNIT I
TYPES OF MACHINE LEARNING

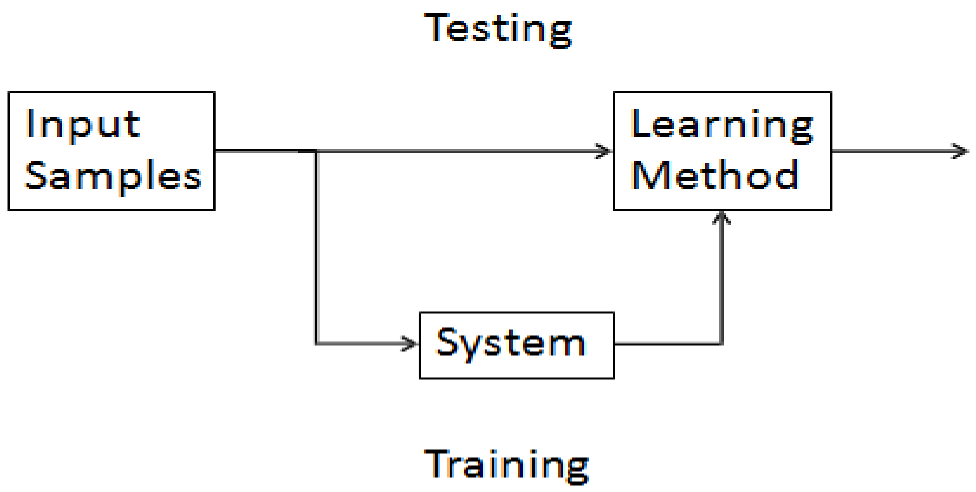
TYPES OF MACHINE LEARNING

- Concept learning: Introduction, version spaces and the candidate elimination algorithm; Learning with trees: Constructing decision trees, CART, classification example.

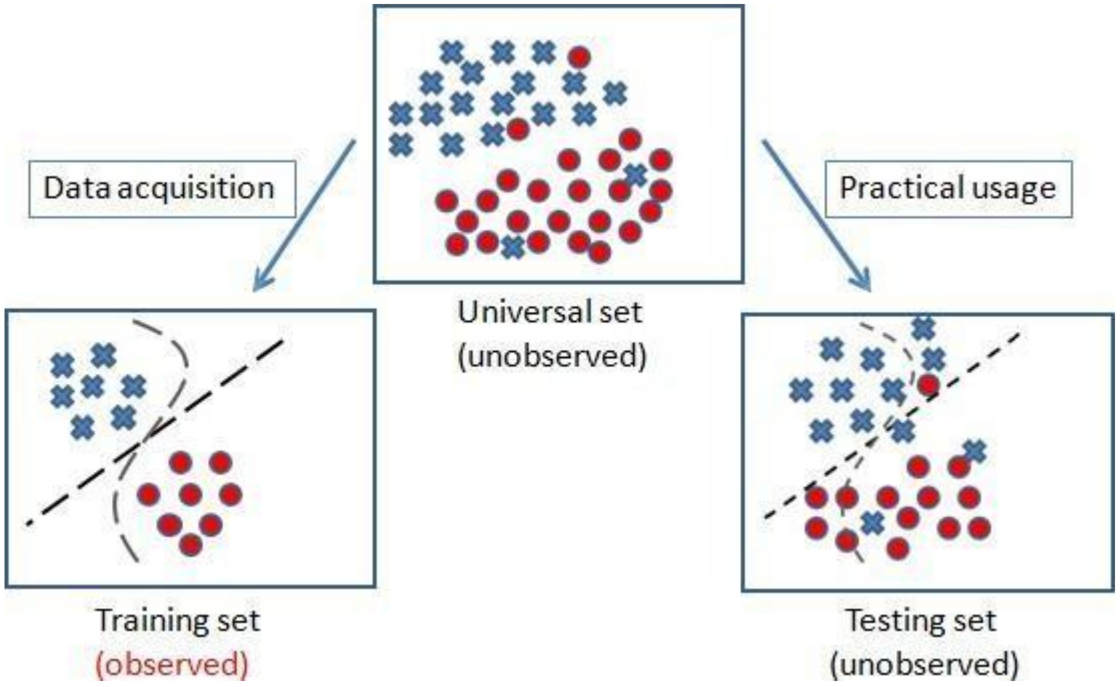
MACHINE LEARNING

- A branch of **artificial intelligence**,concerned with the design and development of algorithms.
- Allow computers to evolve behaviors based on empirical data.
- As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.

Learning system model

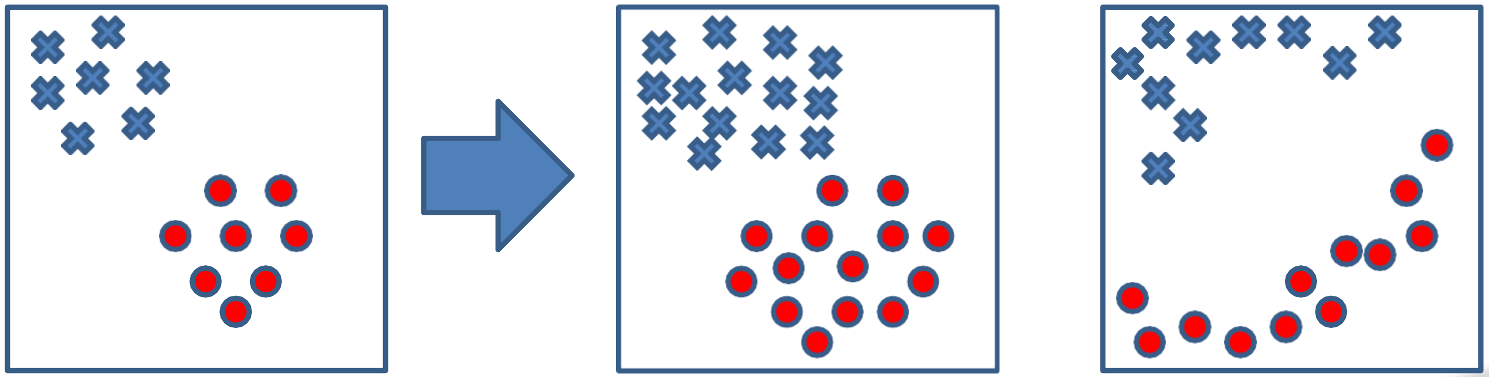


Training and testing



Training and testing

- Training is the process of making the system able tolearn.
- No free lunch rule:
  - Training set and testing set come from the same distribution
  - Need to make some assumptions orbias



Performance

- There are several factors affectingtheperformance:
  - Types of trainingprovided
  - The form and extent of any initial backgroundknowledge
  - The type of feedbackprovided
  - The learning algorithms used
- Two important factors:
  - Modeling
  - Optimization

Algorithms

- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples
- **Supervised learning**
  - Prediction
  - Classification (discrete labels), Regression

Algorithms

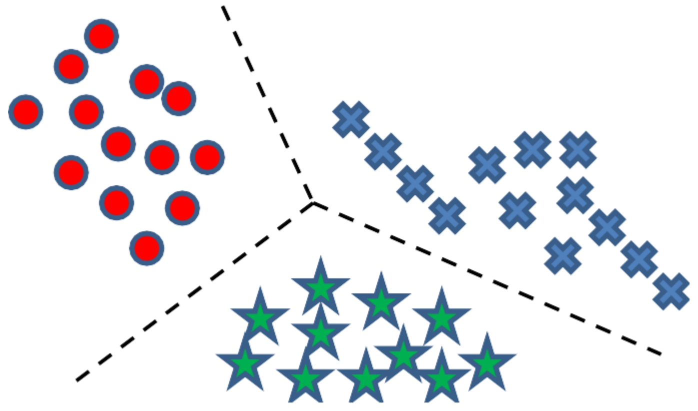
- Clustering
- Probability distribution estimation
- Finding association (infeatures)
- Dimension reduction

- **Semi-supervised learning**
- **Reinforcement learning**
  - Decision making (robot, chess machine)

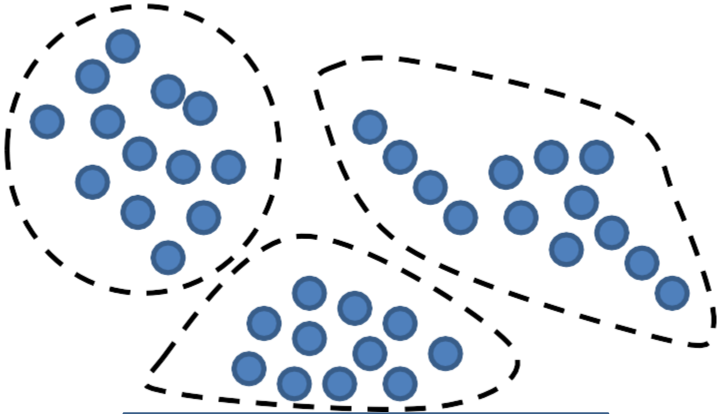
Learning techniques

- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
  - **Parametric** (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - **Non-parametric** (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - **Aggregation**
    - Bagging (bootstrap + aggregation), Adaboost, Randomforest

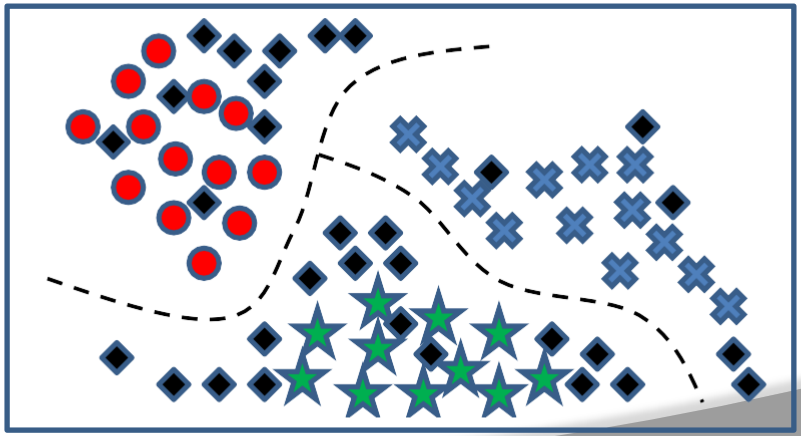
Algorithms



Supervised learning

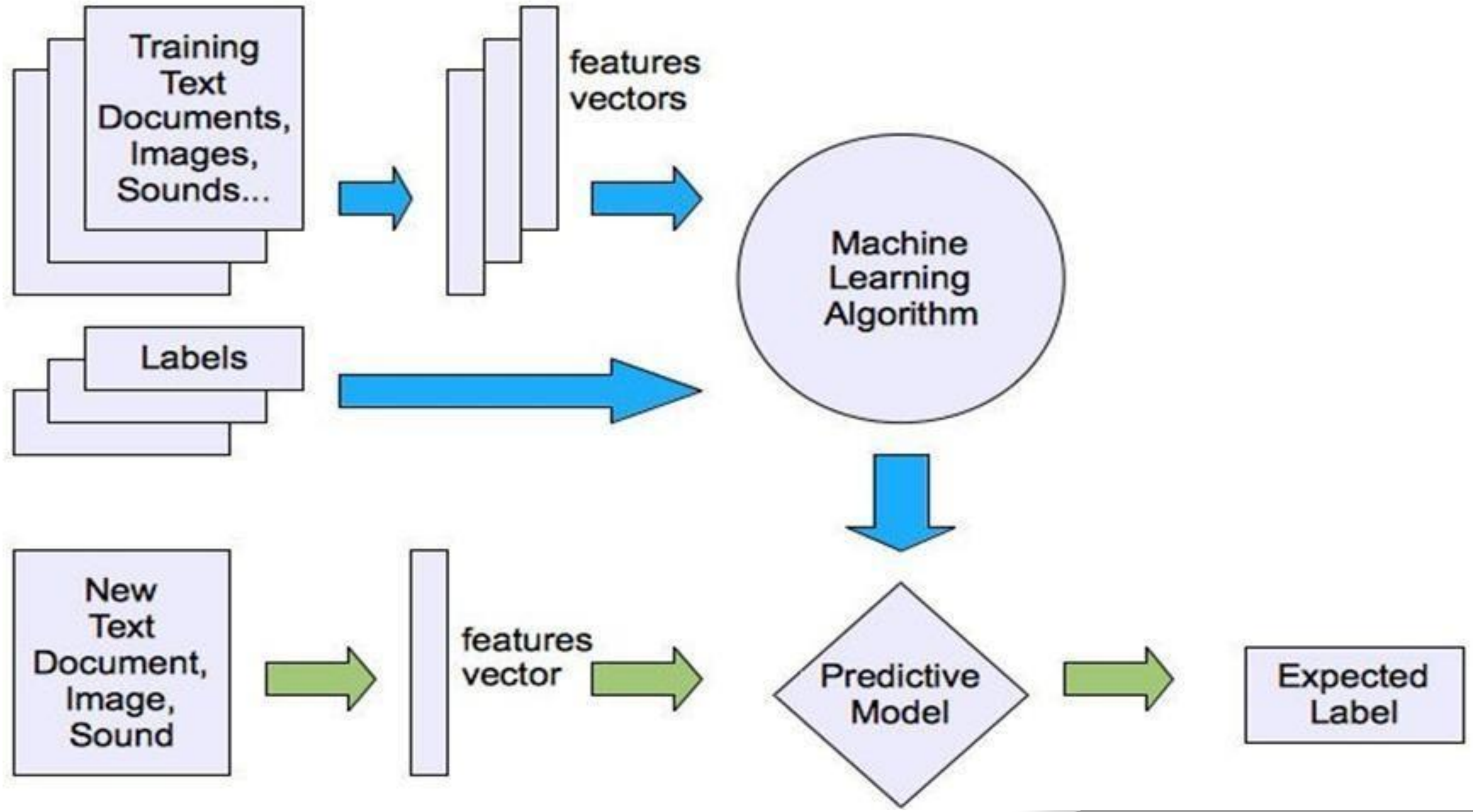


Unsupervised learning



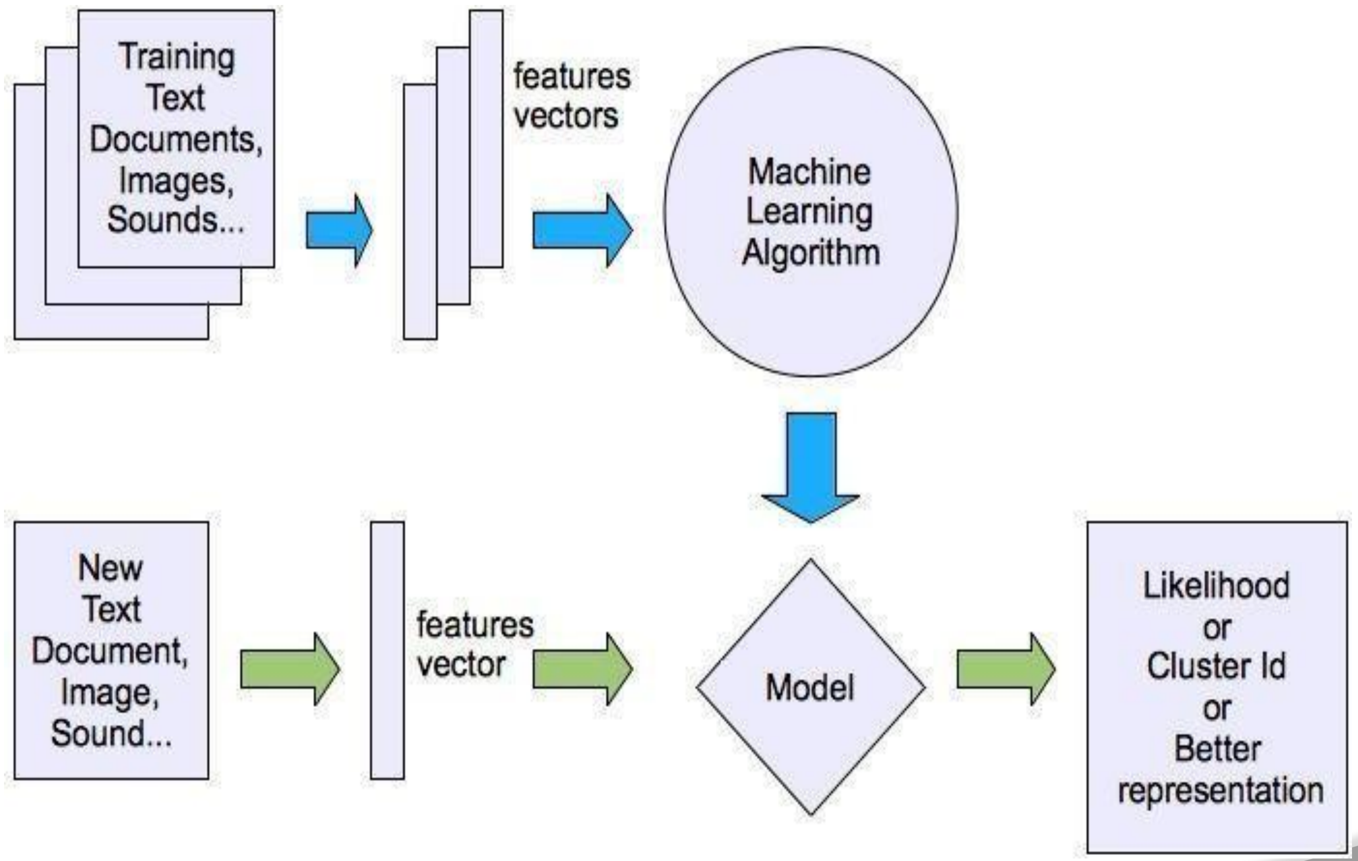
Machine learning structure

- **Supervised learning**

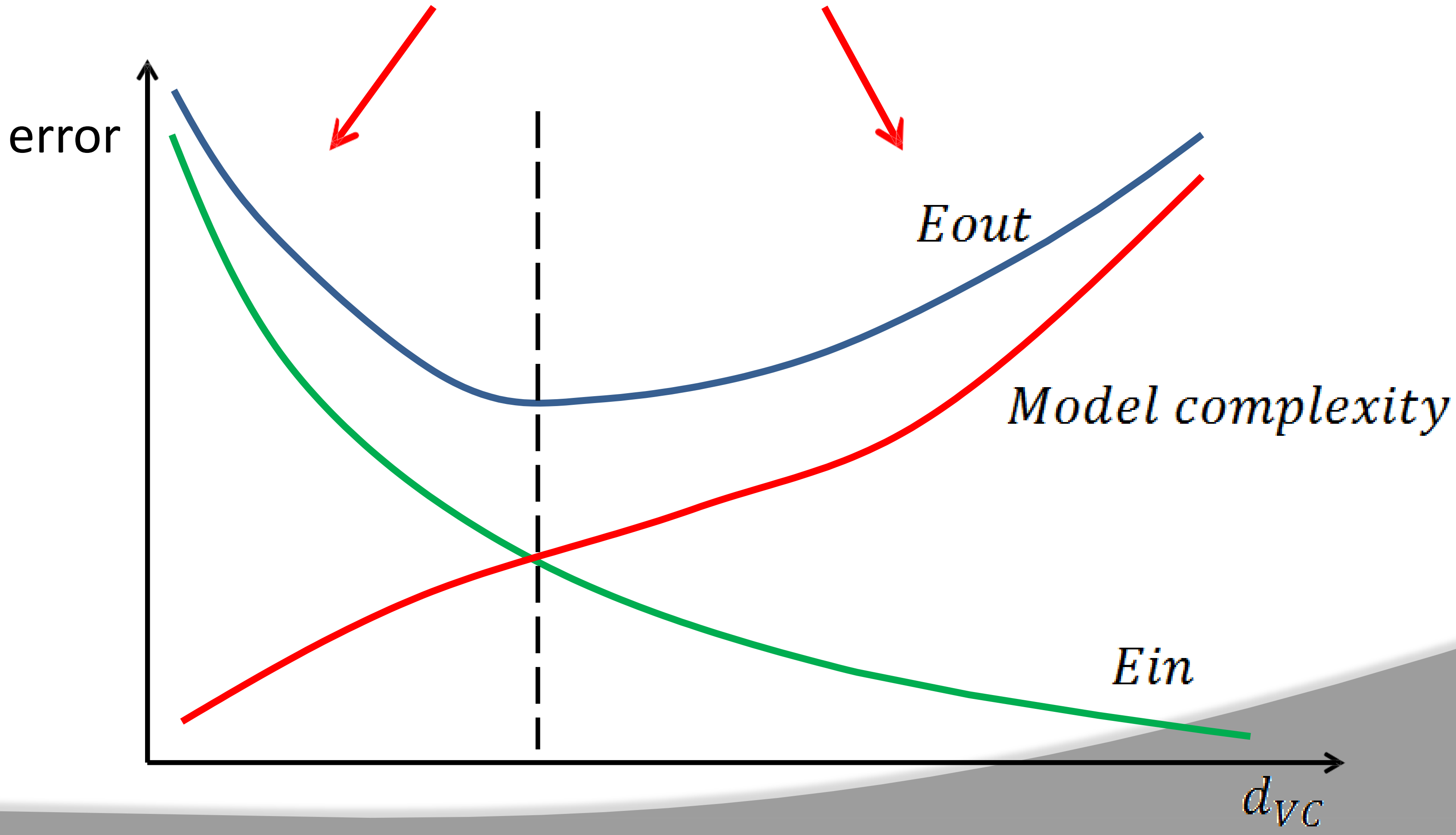


Training Text Documents, Images, Sounds...
features vectors
Labels
Machine Learning Algorithm
New Text Document, Image, Sound
features vector
Predictive Model
Expected Label

Machine learning structure


Training Text Documents, Images, Sounds...
features vectors
Machine Learning Algorithm
New Text Document, Image, Sound...
features vector
Model
Likelihood or Cluster Id or Better representation

Under-fitting VS. Over-fitting (fixed*N*)



Learning techniques

- Linear classifier
- Techniques:
  - Perceptron
  - Logistic regression
  - Support vector machine (SVM)
  - Ada-line
  - Multi-layer perceptron(MLP)

Learning in version space

Generalization operators in version space

•**Replace constants with variables**

color(ball, red)   color(X, red)

•**Remove literals from conjunctions**
shape(X, round) ⍰ size(X, small) ⍰ color(X, red) shape(X, round) ⍰ color(X, red)

•**Add disjunctions**

shape(X, round) ⍰ size(X, small) ⍰ color(X, red)

shape(X, round) ⍰ size(X, small) ⍰ (color(X, red) ⍰ color(X, blue))

•**Replace an class with the superclass in is-a relations**

is-a(tom, cat)   is-a(tom, animal)

Version Spaces

- A hypothesis h is **consistent** with a set of training examples D of target concept if and only if h(x)=c(x) for each training example <x,c(x)> in D.
  - Consistent(h,D) := $\forall$<x,c(x)>$\in$D h(x)=c(x)
- The **version space**, $VS_{H,D}$, with respect to hypothesis space H, and training set D, is the subset of all hypotheses in H consistent with all training examples:
  - $VS_{H,D} = \{h \in H \mid Consistent(h,D)\}$

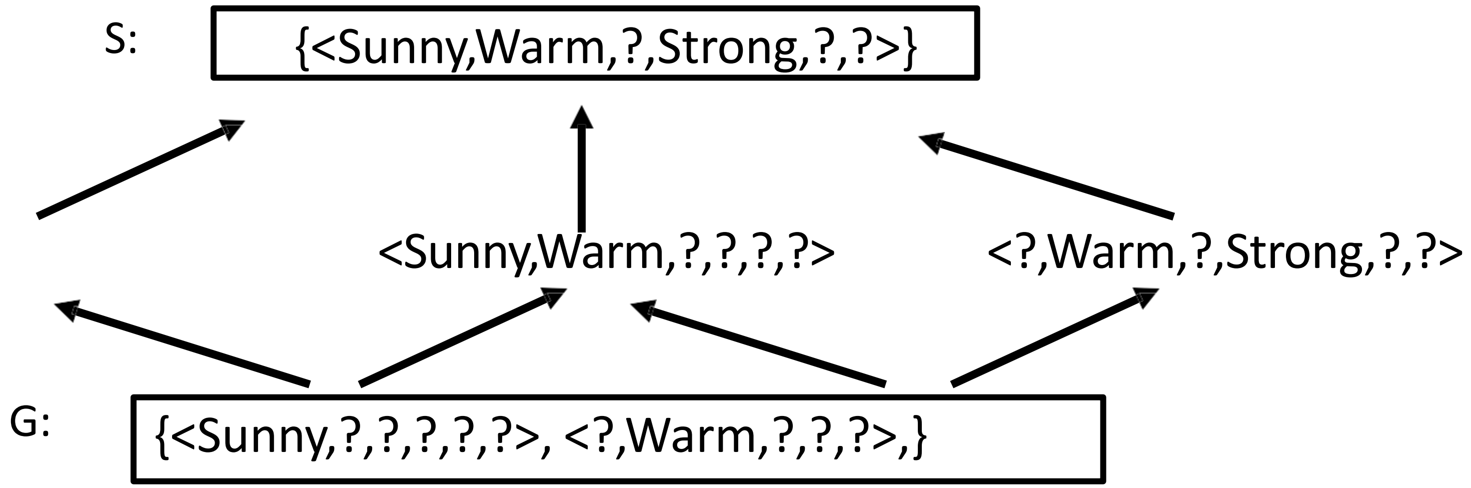
Elimination algorithms

- List-then-Eliminate
    - list every hypothesis in H
    - at each example, remove inconsistent hypo
    - output the list of surviving hypotheses
- something to start with
- ineffective
    - Candidate elimination
        - same principle
        - more compact representation
        - maintain most specific and most general elements of VS(H,D)

Representing VersionSpaces

- The **general boundary**, G, of version space $VS_{H,D}$ is the set of maximally general members.
- The **specific boundary**, S, of version space $VS_{H,D}$ is the set of maximally specific members.
- Every member of the version space lies between these boundaries $VS_{H,D}$ = {h ⍰ H| (⍰ s ⍰ S) (⍰ g ⍰ G) (g ⍰ h ⍰ s) where x ⍰ y means x is more general or equal than y

Example Version Space



x_1 = <Sunny Warm Normal Strong Warm Same>+
x_2 = <Sunny Warm High Strong Warm Same> +
x_3 = <Rainy Cold High Strong Warm Change> -
x_4 = <Sunny Warm High Strong Cool Change> +

Candidate EliminationAlgorithm

$G \leftarrow$ maximally general hypotheses in H specific
S $\leftarrow$ maximally hypotheses in H For each training example d=<x,c(x)>

If d is a positive example

Remove from G any hypothesis that is

inconsistent with d For each hypothesis s in S

that is not consistent with d

•remove s from S.

•Add to S all minimal generalizations h of s such that

- h consistent with d
- Some member of G is more general than h

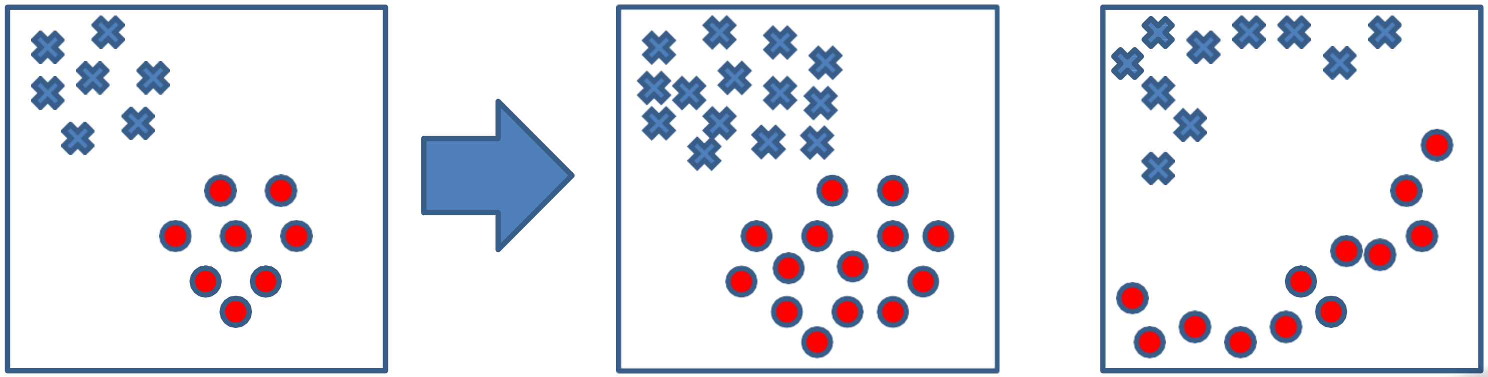
•Remove from S any hypothesis that is more general than another
hypothesis in S

Decision trees ID3 algorithm

- Inductive learning
- Learns concept descriptions fromexamples
- Examples (instances of concepts) are defined by attributesand classified inclasses
- Concepts are represented as a decision tree in which every level of the tree is associated to an attribute
- The leafs are labeled withconcepts

Training and testing

- Training is the process of making the system able to learn.
- No free lunch rule:
  - Training set and testing set come from the same distribution
  - Need to make some assumptionsorbias



CART

- Classification And Regression Trees
- Developed by Breiman, Friedman, Olshen, Stone in early 80's.
  - Introduced tree-based modeling into the statistical mainstream
  - Rigorous approach involving cross-validation to select the optimal tree
- One of many tree-based modeling techniques.
  - CART -- the classic
  - CHAID

– C5.0

  - Software package variants (SAS, S-Plus, R…)
  - Note: the "rpart" package in "R" is freely available

CART advantages

- Nonparametric (no probabilistic assumptions)
- Automatically performs variable selection
- Uses any combination of continuous/discrete variables
  - Very nice feature: ability to automatically bin massively categorical variables into a few categories.
    - zip code, business class, make/model…
- Discovers "interactions" among variables
  - Good for "rules" search
  - Hybrid GLM-CART models

Performance

- There are several factors affectingtheperformance:
  - Types of trainingprovided
  - The formand extent of any initial backgroundknowledge
  - The type of feedbackprovided
  - The learning algorithms used
- Two important factors:
  - Modeling
  - Optimization

CART Disadvantages

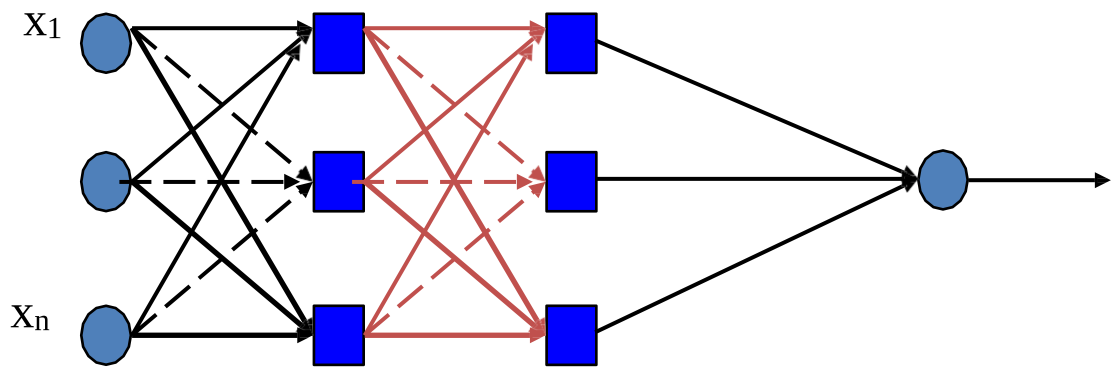
- The model is a *step function*, not a continuous score
    - So if a tree has 10 nodes, yhat can only take on 10 possible values.
    - MARS improves this.
- Might take a large tree to get good lift
    - But then hard to interpret
    - Data gets chopped thinner at each split
- Instability of model structure
    - Correlated variables random data fluctuations could result in entirely different trees.
- CART does a poor job of modeling *linear structure*

UNIT II
LINEAR DISCRIMINANTS

LINEAR DISCRIMINANTS

Perceptron (MLP): Going forwards, backwards, MLP in practices, deriving back; Propagation support vector Machines: Optimal separation, kernels.

Perceptron Learning Theorem



- We will introduce the MLP and the back propagation algorithm
- which is used to train it
- MLP used to describe any general feed forward (no recurrent network

connections)

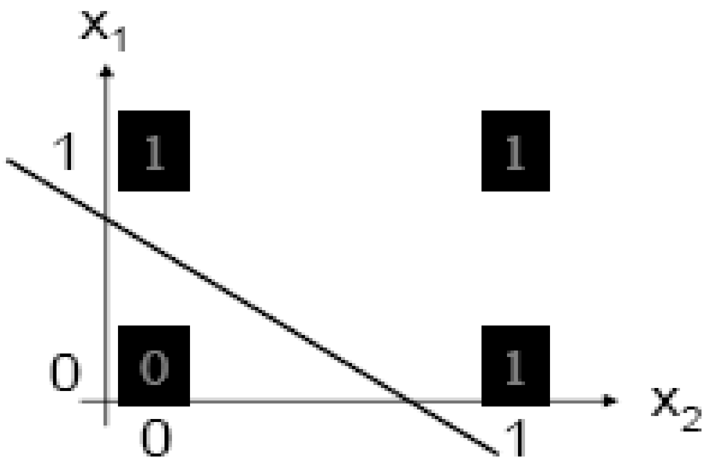
- However, we will concentrate on nets with units arranged in layers

Perceptron Learning Theorem

- *Recap*: A perceptron (threshold unit) can *learn* anything that it can
  represent (i.e. anything separable with a hyperplane)

OR function

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

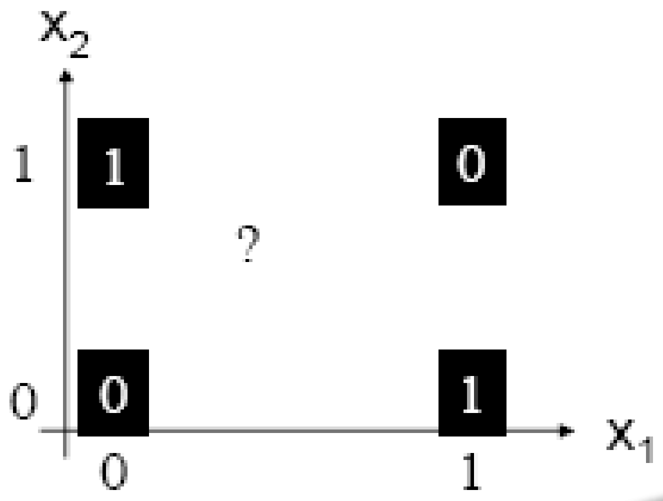


The Exclusive ORproblem

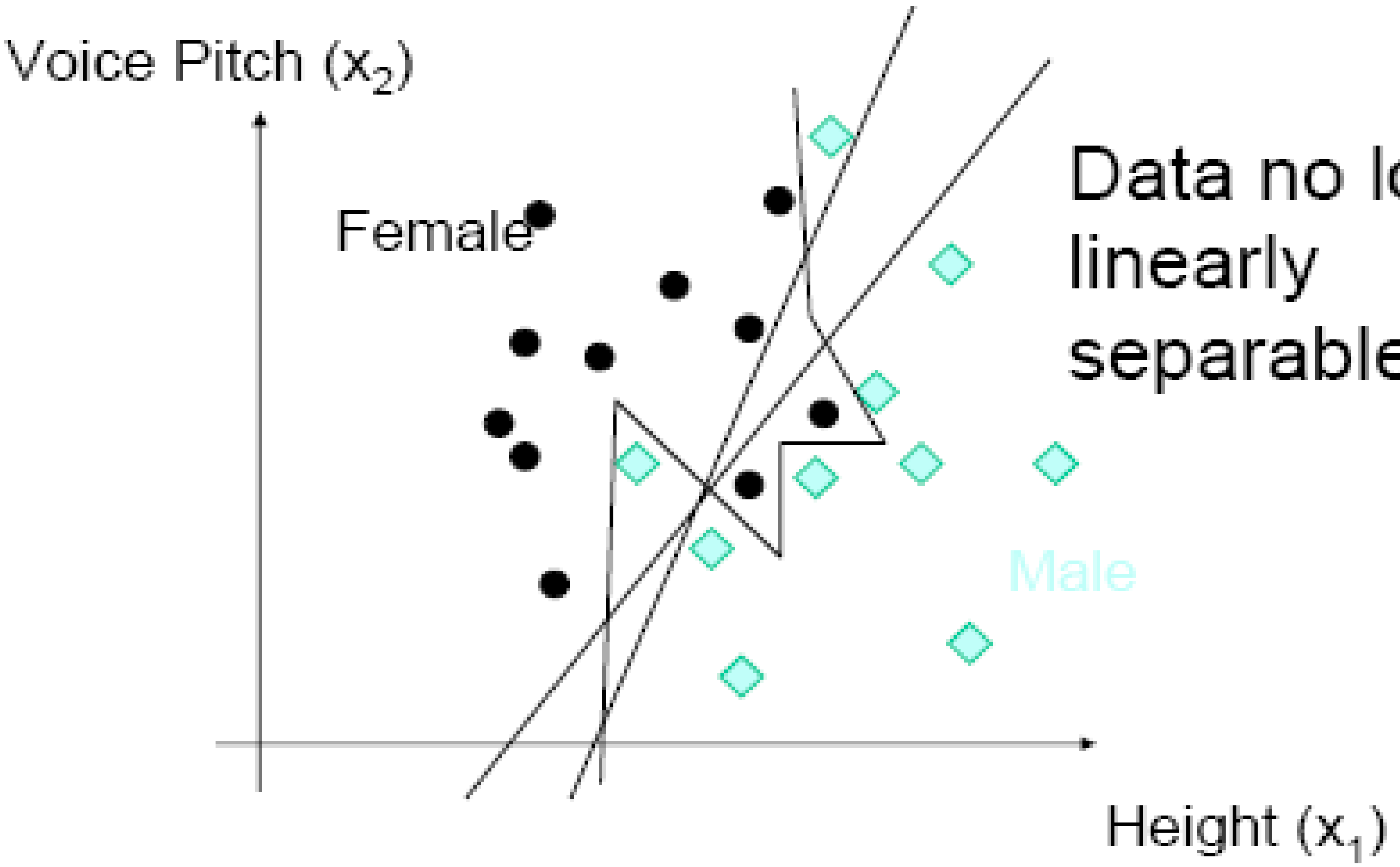
- A Perceptron cannot represent Exclusive OR since it is not linearly separable.

XOR function

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



The Exclusive ORproblem

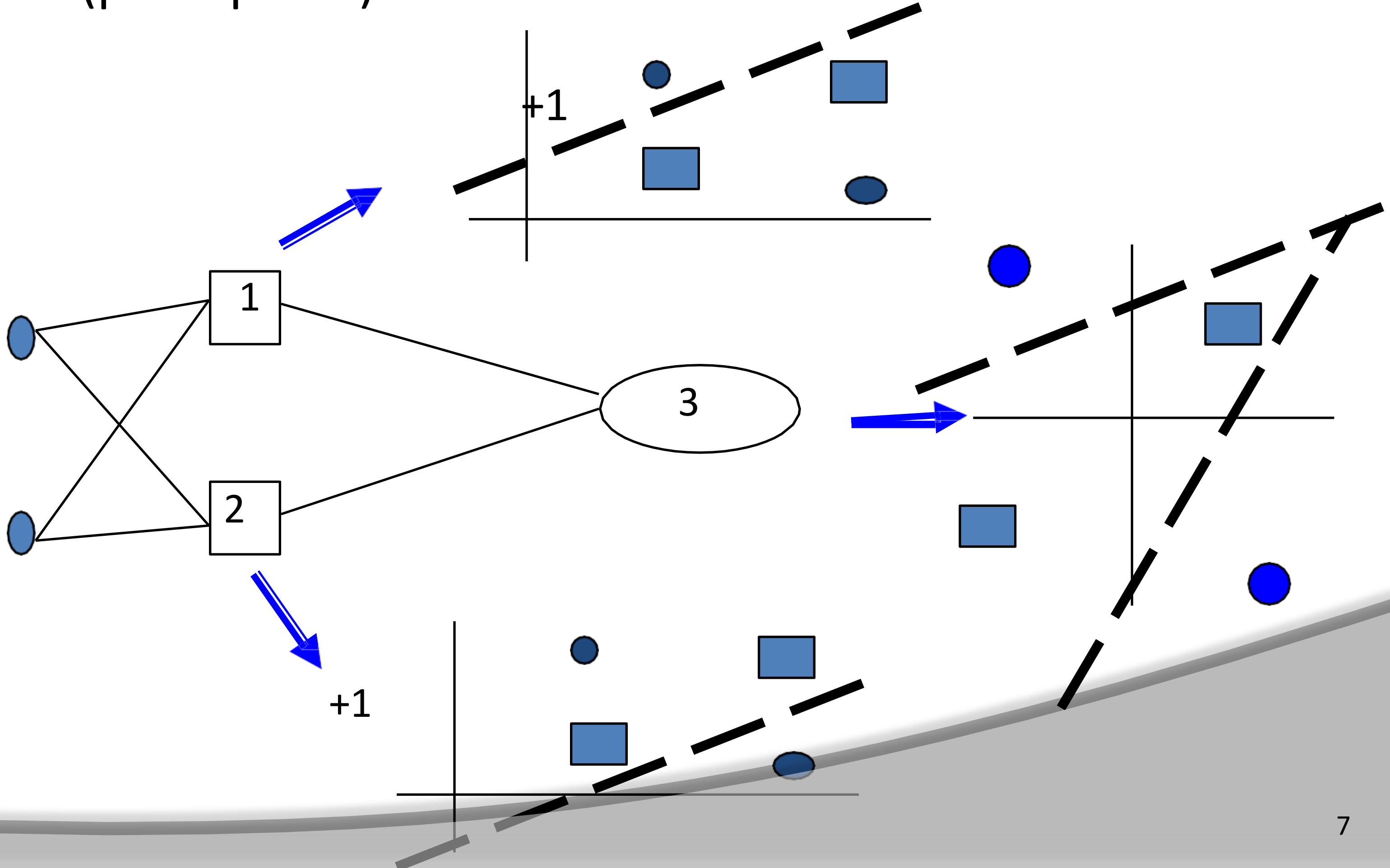


Data no longer linearly separable

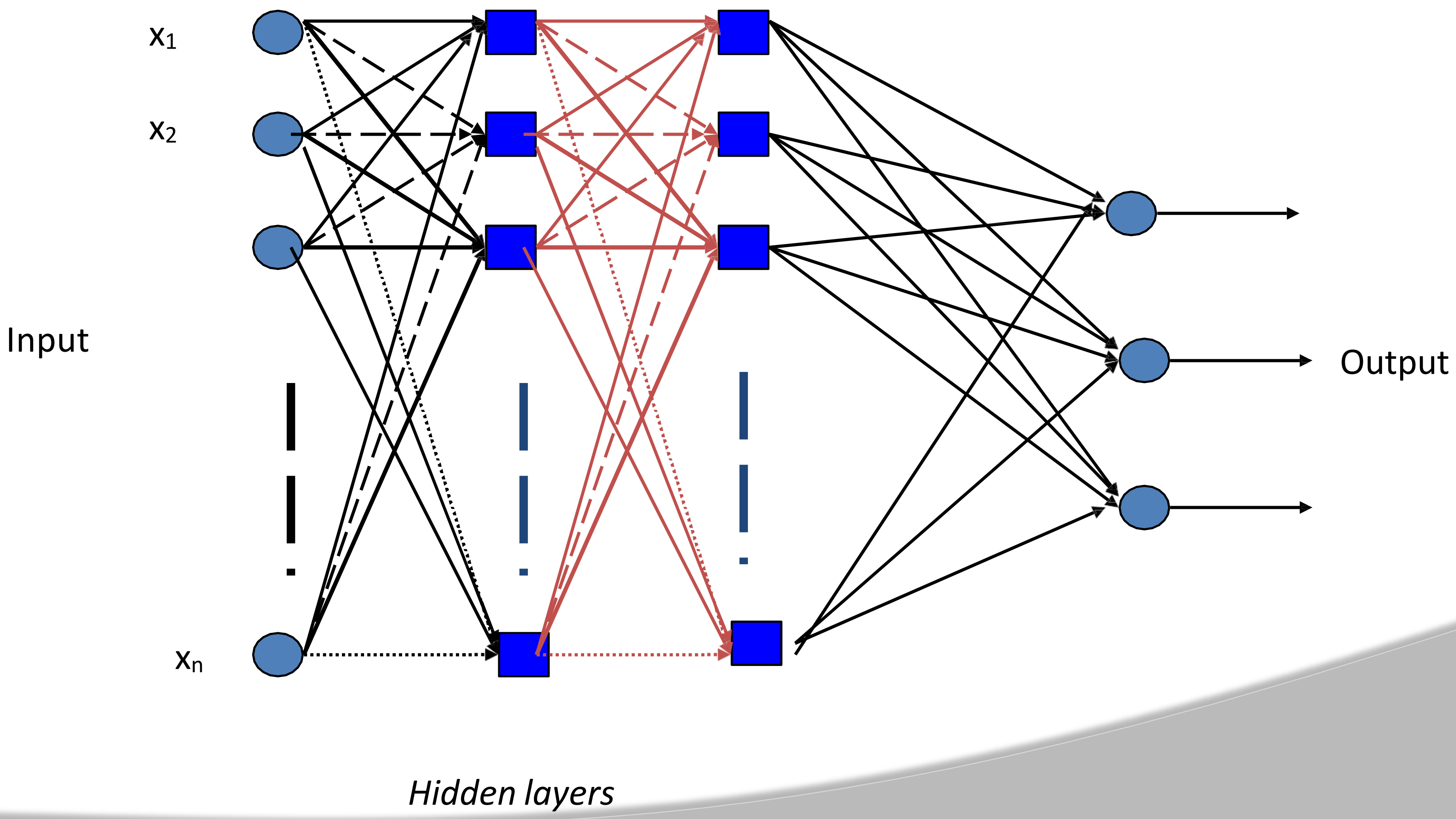
What is a good decision boundary ?

The Exclusive ORproblem

- Minsky & Papert (1969) offered solution to XOR problem by combining perceptron unit responses using a second layer of Units. Piecewise linear classification using an MLP with threshold (perceptron) units

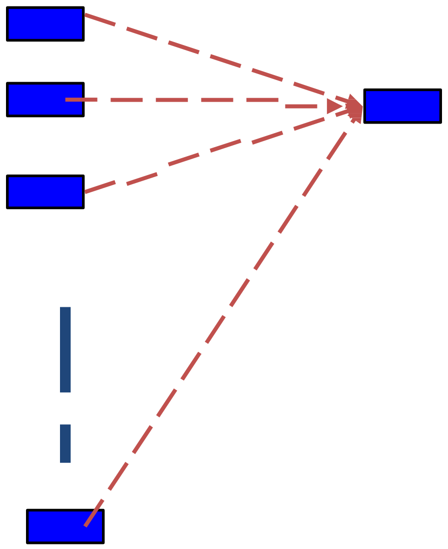


Three-layer networks



Properties of architecture

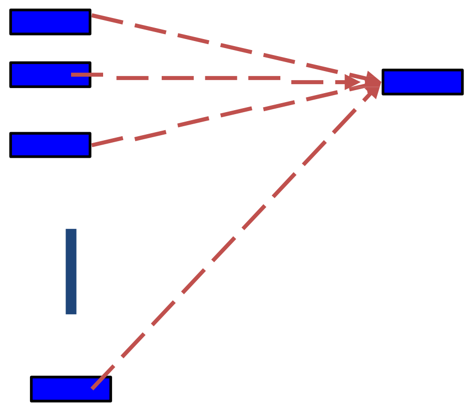
- No connections within alayer
- No direct connections between input and output layer.



$$y_i = f\left(\sum_{j=1}^{m} w_{ij} x_j + b_i\right)$$

Properties of architecture

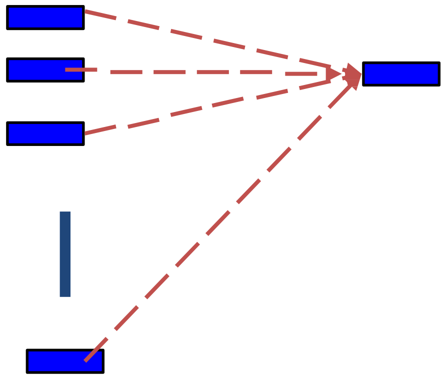
- No connections within a layer
- No direct connections between input and output layers
- Fully connected betweenlayers



$$y_i = f\left(\sum_{j=1}^{m} w_{ij} x_j + b_i\right)$$

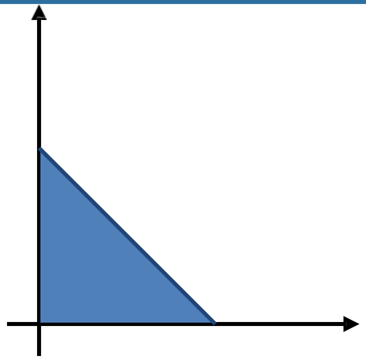
Properties of architecture

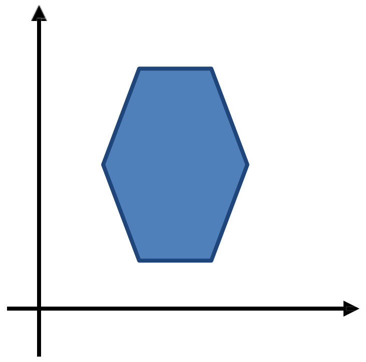
- No connections within a layer
- No direct connections between input and output layers
- Fully connected between layers
- Often more than 3 layers
- Number of output units need not equal number of input units
- Number of hidden units per layer can be more or less than input or output units

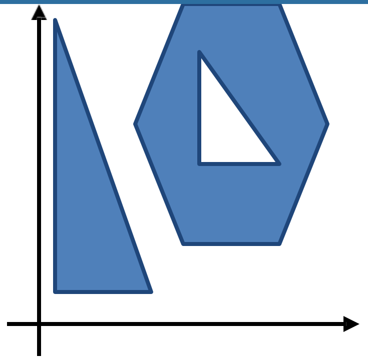


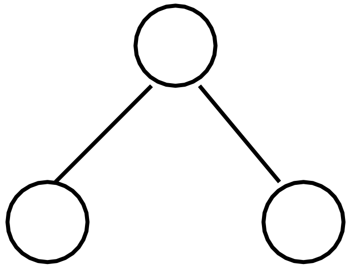
$$y_i = f\left(\sum_{j=1}^{m} w_{ij} x_j + b_i\right)$$

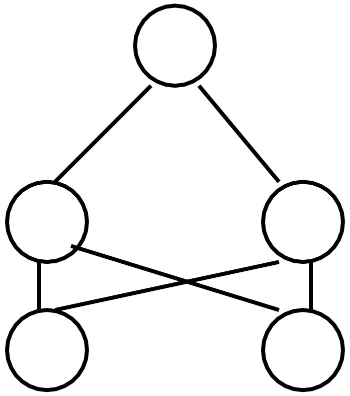
What do each of the layers do?

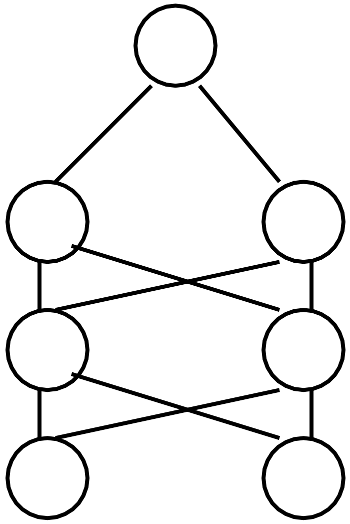












1st layer draws linear boundaries

2nd layer combines the boundaries

3rd layer can generate arbitrarily complex boundaries

Back propagation learning algorithm ‘BP’

•Solution to credit assignment problem in MLP. *Rumelhart, Hinton and Williams (1986) (*though actually invented earlier in a PhD thesis relating to economics)

•**BP has two phases**:

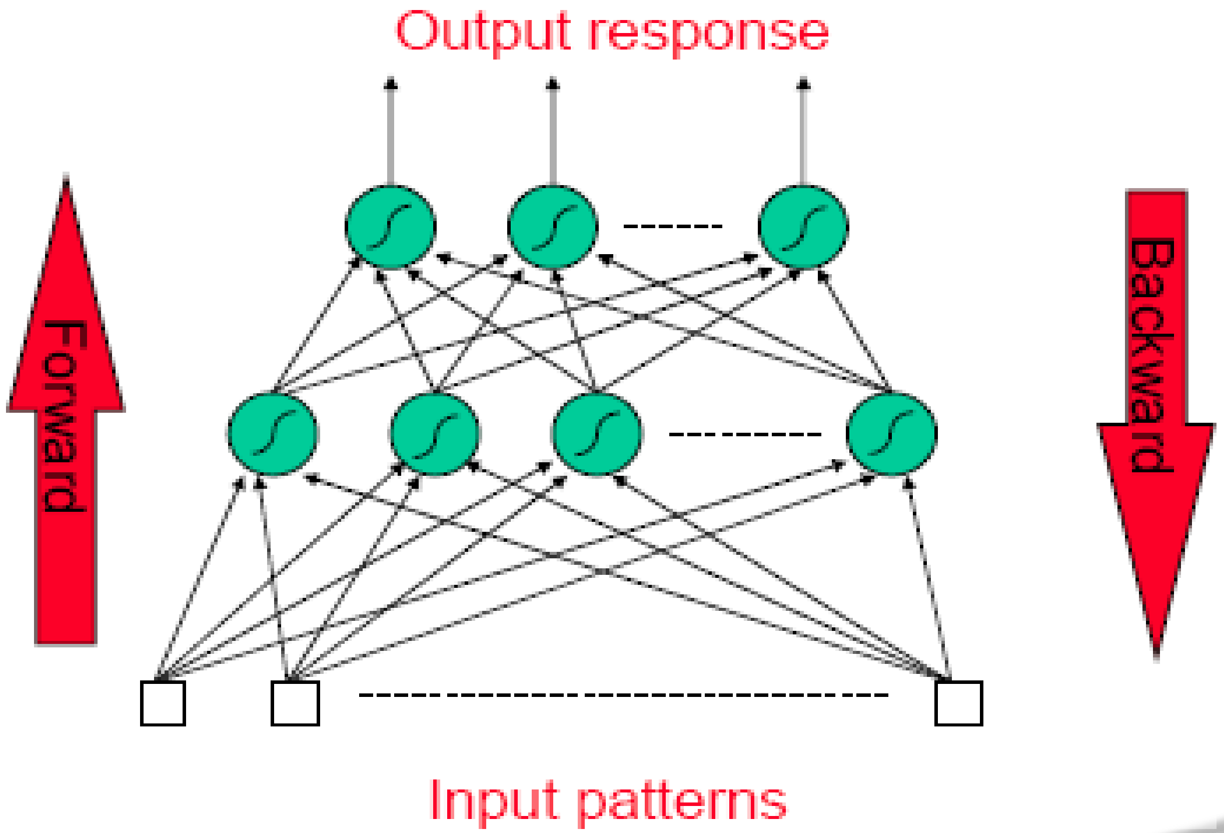
•Forward pass phase: computes ‘functional signal’, feed forward propagation of

•input pattern signals through network

•Backwardpass phase: computes‘error signal’, *propagates* the aerror

backwards through network starting at output units (where the error is the difference between actual and desired output values)

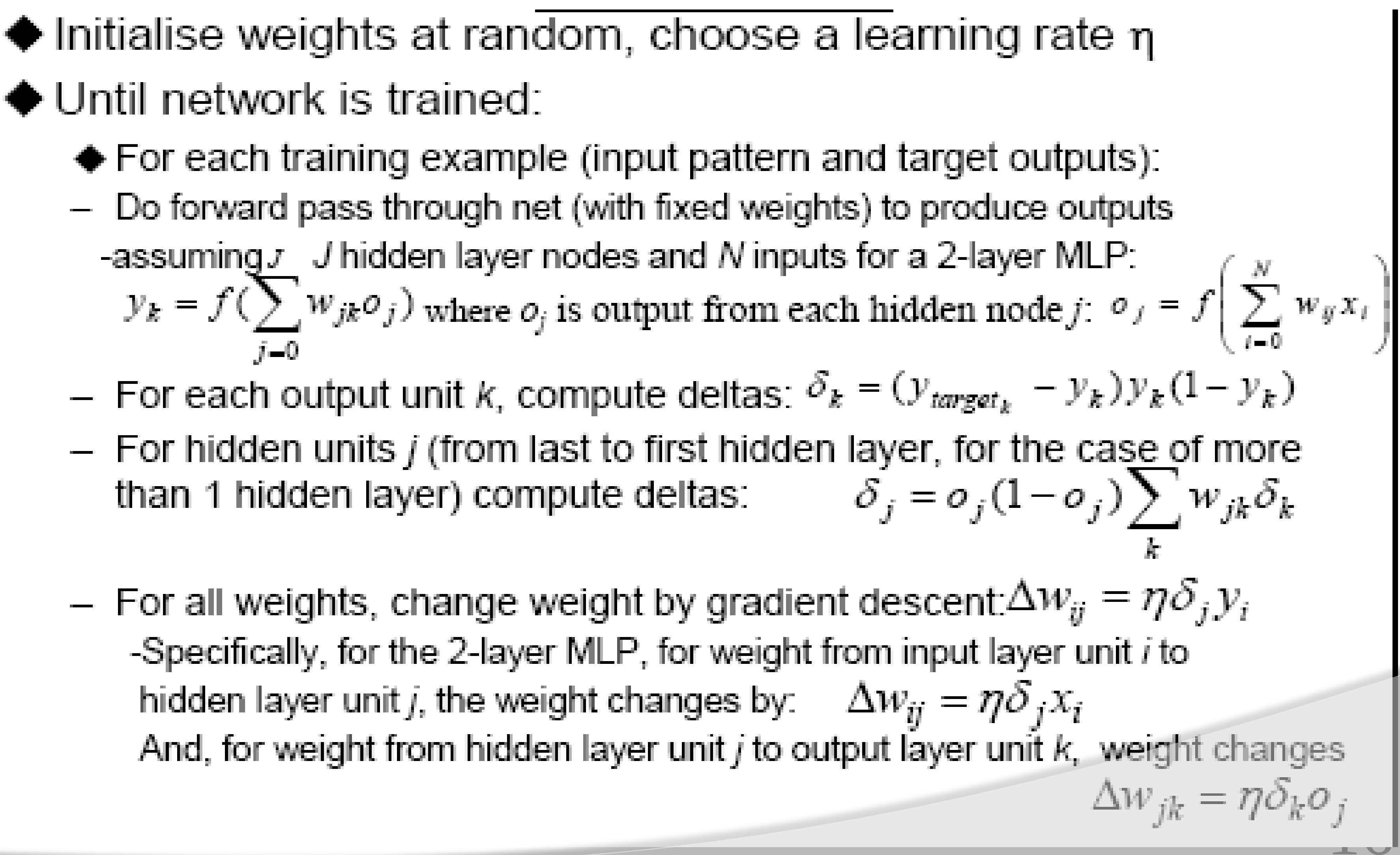
Conceptually: Forward Activity - Backward Error



Forward Propagation ofActivity

- Step 1: Initialize weights at random, choose a learning rate η
- Until network is trained:
- For each training example i.e. input pattern and target output(s)
- Step 2: Do forward pass through net (with fixed weights) to produce output(s)
  - i.e., in Forward Direction, layer by layer:
  - Inputs applied
  - Multiplied by weight
  - Summed 'Squashed' by sigmoid activation function
  - Output passed to each neuron in next layer
  - Repeat above until network output(s) produced

‘Back-prop’ algorithmsummary



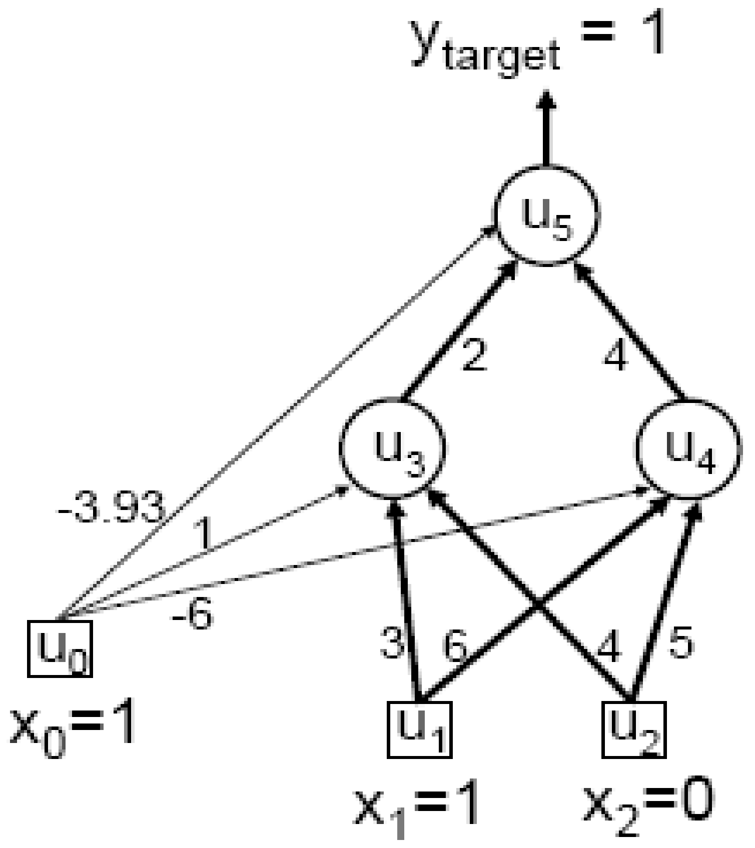
- Initialise weights at random, choose a learning rate η
- Until network is trained:
  - For each training example (input pattern and target outputs):
  - – Do forward pass through net (with fixed weights) to produce outputs
    -assuming J hidden layer nodes and N inputs for a 2-layer MLP:
    $y_k = f(\sum_{j=0}^{J} w_{jk} o_j)$ where o_j is output from each hidden node j: $o_j = f\left(\sum_{i=0}^{N} w_{ij} x_i\right)$
  - – For each output unit k, compute deltas: $\delta_k = (y_{target_k} - y_k) y_k (1 - y_k)$
  - – For hidden units j (from last to first hidden layer, for the case of more than 1 hidden layer) compute deltas: $\delta_j = o_j(1 - o_j) \sum_k w_{jk} \delta_k$
  - – For all weights, change weight by gradient descent: $\Delta w_{ij} = \eta \delta_j y_i$
    -Specifically, for the 2-layer MLP, for weight from input layer unit i to hidden layer unit j, the weight changes by: $\Delta w_{ij} = \eta \delta_j x_i$
    And, for weight from hidden layer unit j to output layer unit k, weight changes
    $\Delta w_{jk} = \eta \delta_k o_j$

‘Back-prop’ algorithmsummary

- Initialise weights at random, choose a learning rate η
- Until network is trained:
  - For each training example (input pattern and target outputs):
  - – Do forward pass through net (with fixed weights) to produce network outputs
  - – For each output unit *k*, compute deltas (local gradients):
  - – For hidden units *j* (from last to first hidden layer, for the case of more than 1 hidden layer) compute deltas (local gradients):
  - – For all weights, change weight by gradient descent (generalized Delta Rule): $\Delta w_{jk} = \eta \delta_k o_j$

    where o_j is the input to the neuron *k*, and η is the learning rate, and δ_k is local gradient for the neuron

MLP/BP: A worked example



Current state:

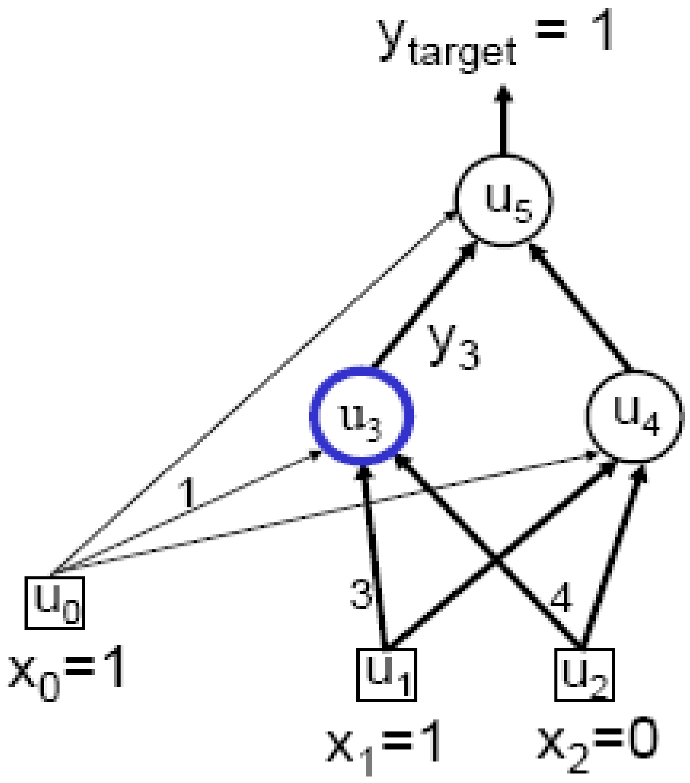
- Weights on arrows e.g. $w_{13} = 3$, $w_{35} = 2$, $w_{24} = 5$
- Bias weights, e.g.

bias for unit 4 (u_4) is $w_{04} = -6$

Training example (e.g. for logical OR problem):

- Input pattern is $x_1=1$, $x_2=0$
- Target output is $y_{target}=1$

Worked example: ForwardPass



Output for any neuron/unit j can be calculated from:

$$a_j = \sum_i w_{ij} x_i$$

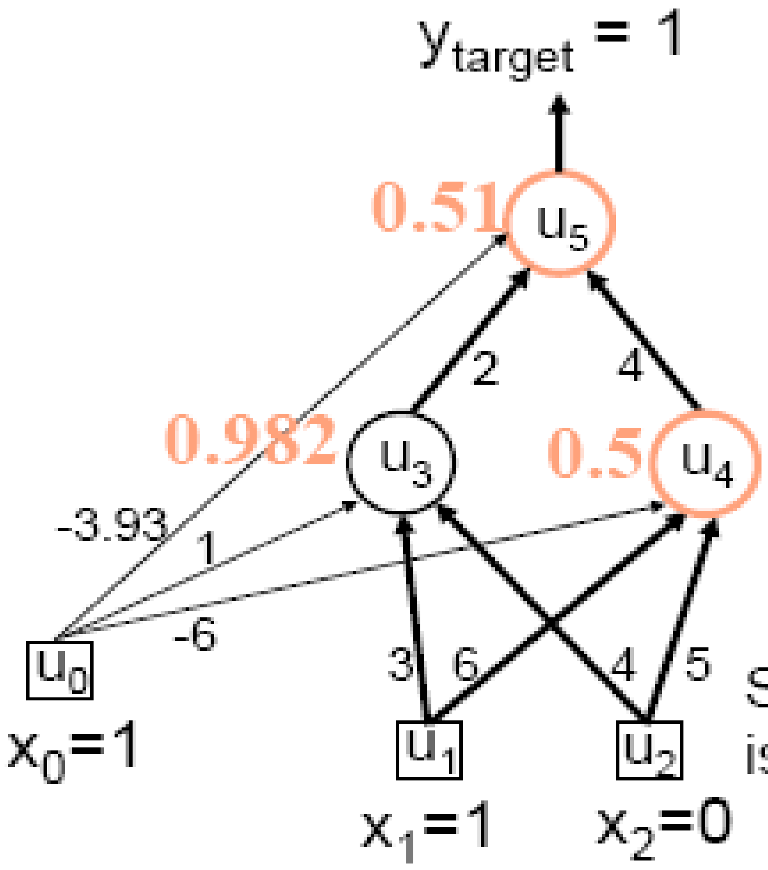
$$y_j = f(a_j) = \frac{1}{1+e^{-a_j}}$$

e.g Calculating output for Neuron/unit 3 in hidden layer:

$$a_3 = 1*1 + 3*1 + 4*0 = 4$$

$$y_3 = f(4) = \frac{1}{1+e^{-4}} = 0.982$$

Worked example: ForwardPass

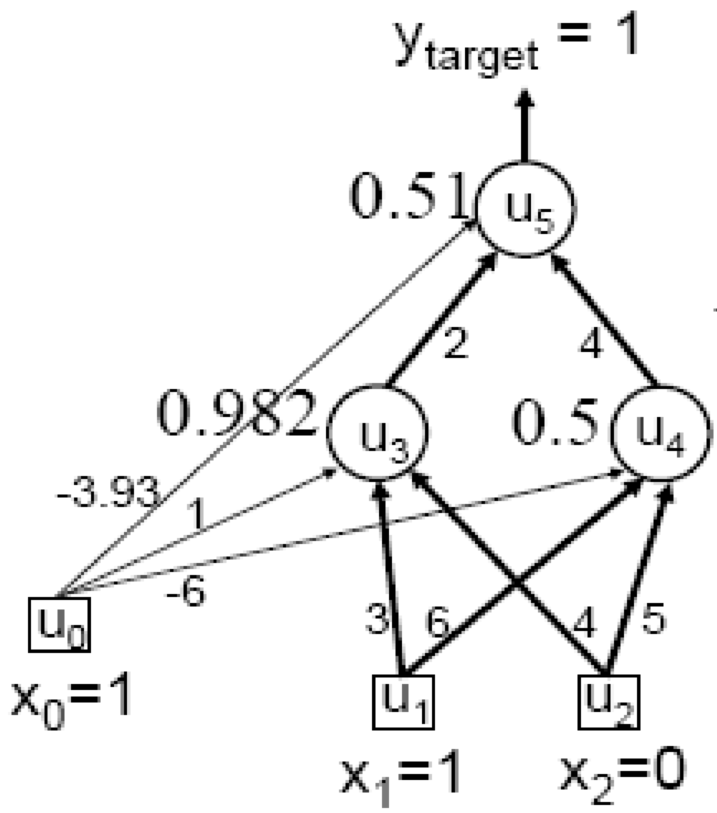


Unit	activation	output
	a_j	y_j
u_3	4.00	0.982
u_4	0.00	0.500
u_5	0.04	**0.510**

(network output)

So the error for this training example is: $(y_{target} - y_5) = (1 - 0.510) = 0.490$

Worked example: BackwardPass



Now compute delta values starting at the output:

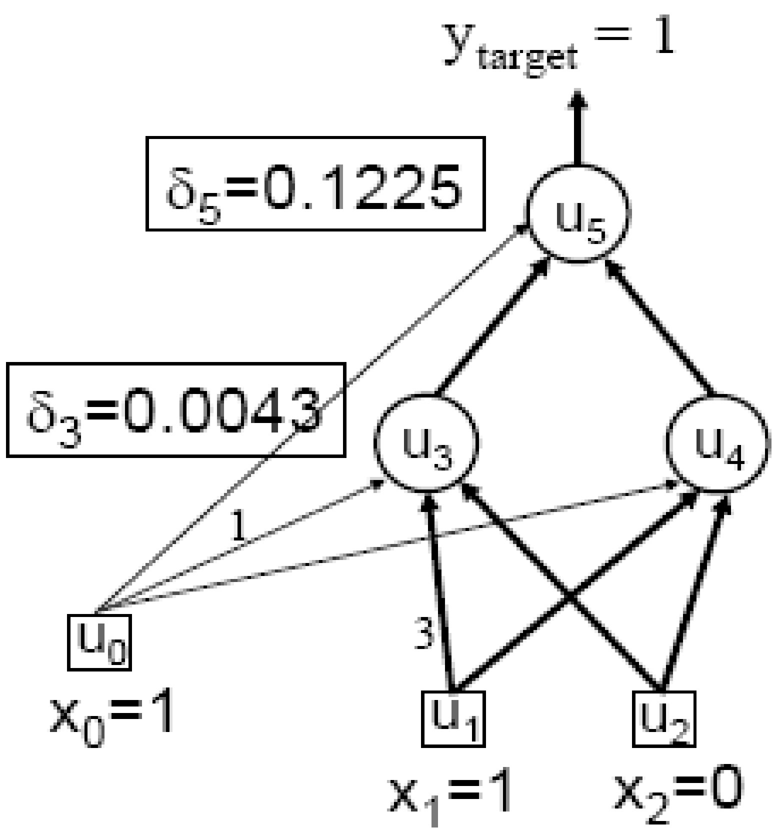
$$\delta_5 = y_5(1 - y_5)\,(y_{target} - y_5)$$
$$= 0.51(1 - 0.51) \times 0.49$$
$$= \mathbf{0.1225}$$

Then for hidden units:

$$\delta_4 = y_4(1 - y_4)\, w_{45}\, \delta_5$$
$$= 0.5(1 - 0.5) \times 4 \times 0.1225$$
$$= \mathbf{0.1225}$$

$$\delta_3 = y_3(1 - y_3)\, w_{35}\, \delta_5$$
$$= 0.982(1\text{-}0.982) \times 2 \times 0.1225$$
$$= \mathbf{0.0043}$$

Worked example: Update Weights Using Generalized Delta Rule (BP)

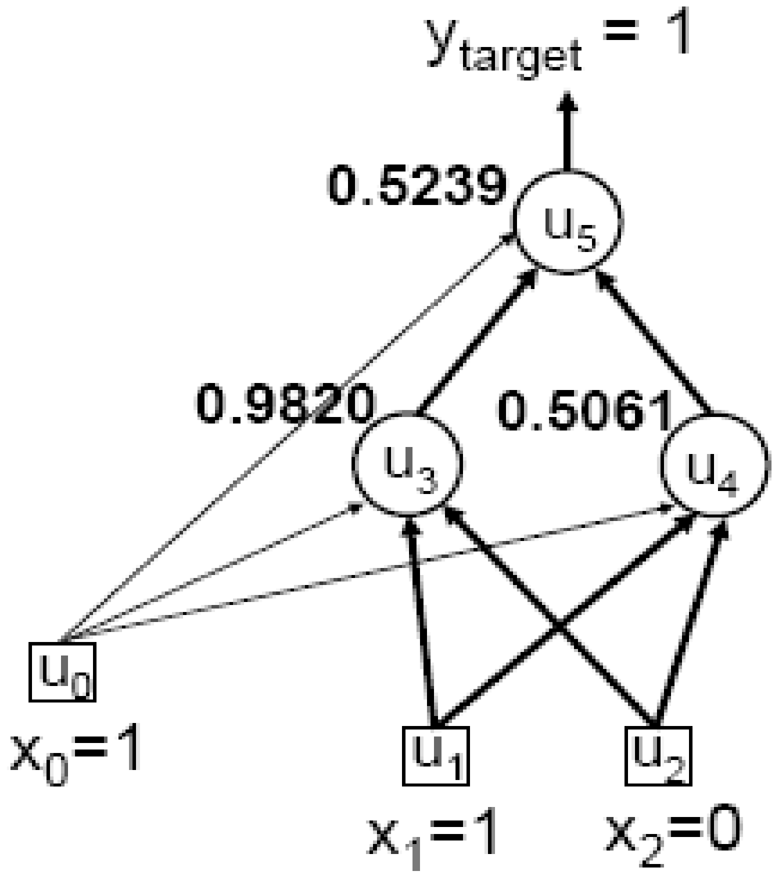


- Set learning rate η = 0.1
  Change weights by:
  $$\Delta w_{ij} = \eta \delta_j y_i$$
- e.g.bias weight on u_3:
  Δw_{03} = $\eta\delta_3 x_0$
  =0.1*0.0043*1
  = 0.0004
  So, new w_{03} ⌫
  w_{03}(old)+Δw_{03}
  =1+0.0004=1.0004
- and likewise:
  w_{13} ⌫ 3 + 0.0004
  = 3.0004

Similarly for the all weights wij

i	j	w_{ij}	δ_j	y_i	Updated w_{ij}
0	3	**1**	0.0043	1.0	**1.0004**
1	3	**3**	0.0043	1.0	**3.0004**
2	3	**4**	0.0043	0.0	**4.0000**
0	4	**-6**	0.1225	1.0	**-5.9878**
1	4	**6**	0.1225	1.0	**6.0123**
2	4	**5**	0.1225	0.0	**5.0000**
0	5	**-3.92**	0.1225	1.0	**-3.9078**
3	5	**2**	0.1225	0.9820	**2.0120**
4	5	**4**	0.1225	0.5	**4.0061**

Similarly for the all weights wij



On next forward pass:

The new activations are:

$y_3 = f(4.0008) = 0.9820$

$y_4 = f(0.0245) = 0.5061$

$y_5 = f(0.0955) = \mathbf{0.5239}$

Thus the new error

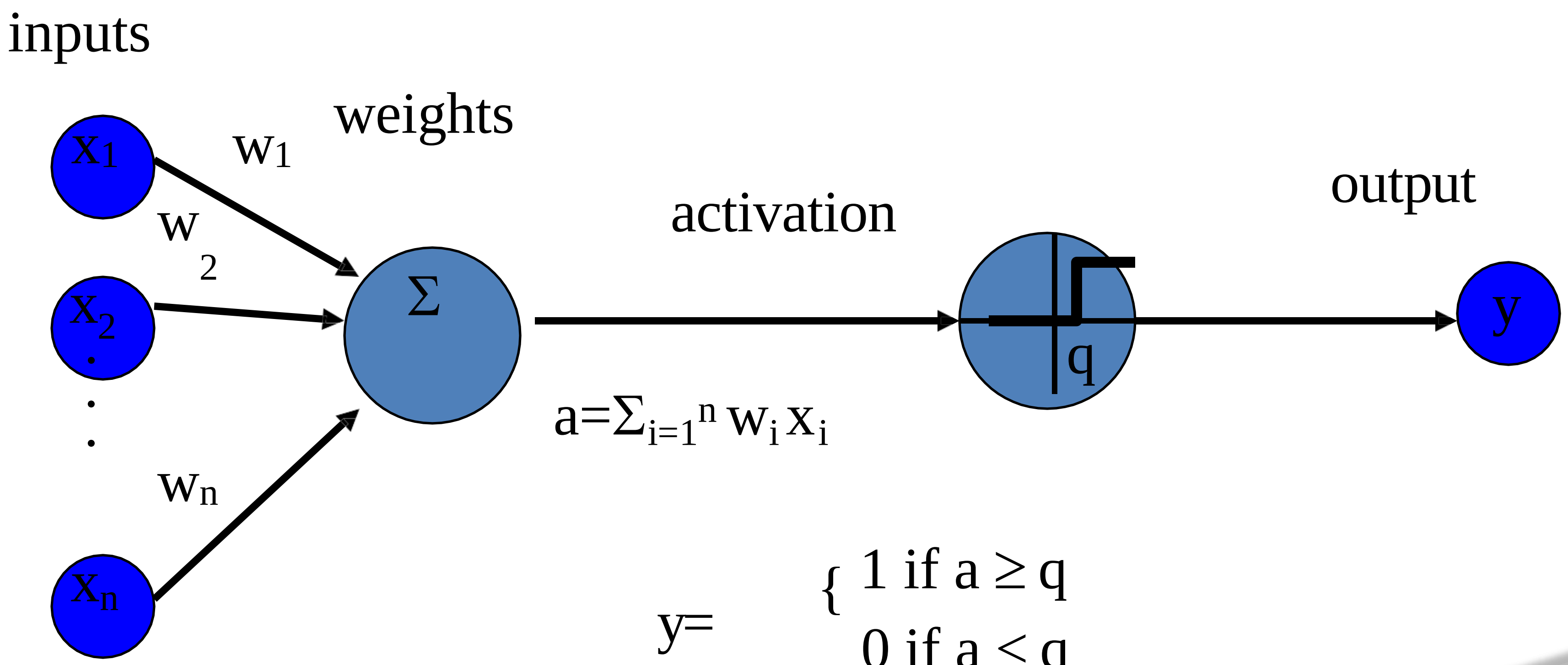
$(y_{target} - y_5) = (1 - 0.5239) = 0.476$

has been reduced by **0.014**

(from **0.490** to **0.476**)

Ref: "Neural Network Learning & Expert Systems" by Stephen Gallant

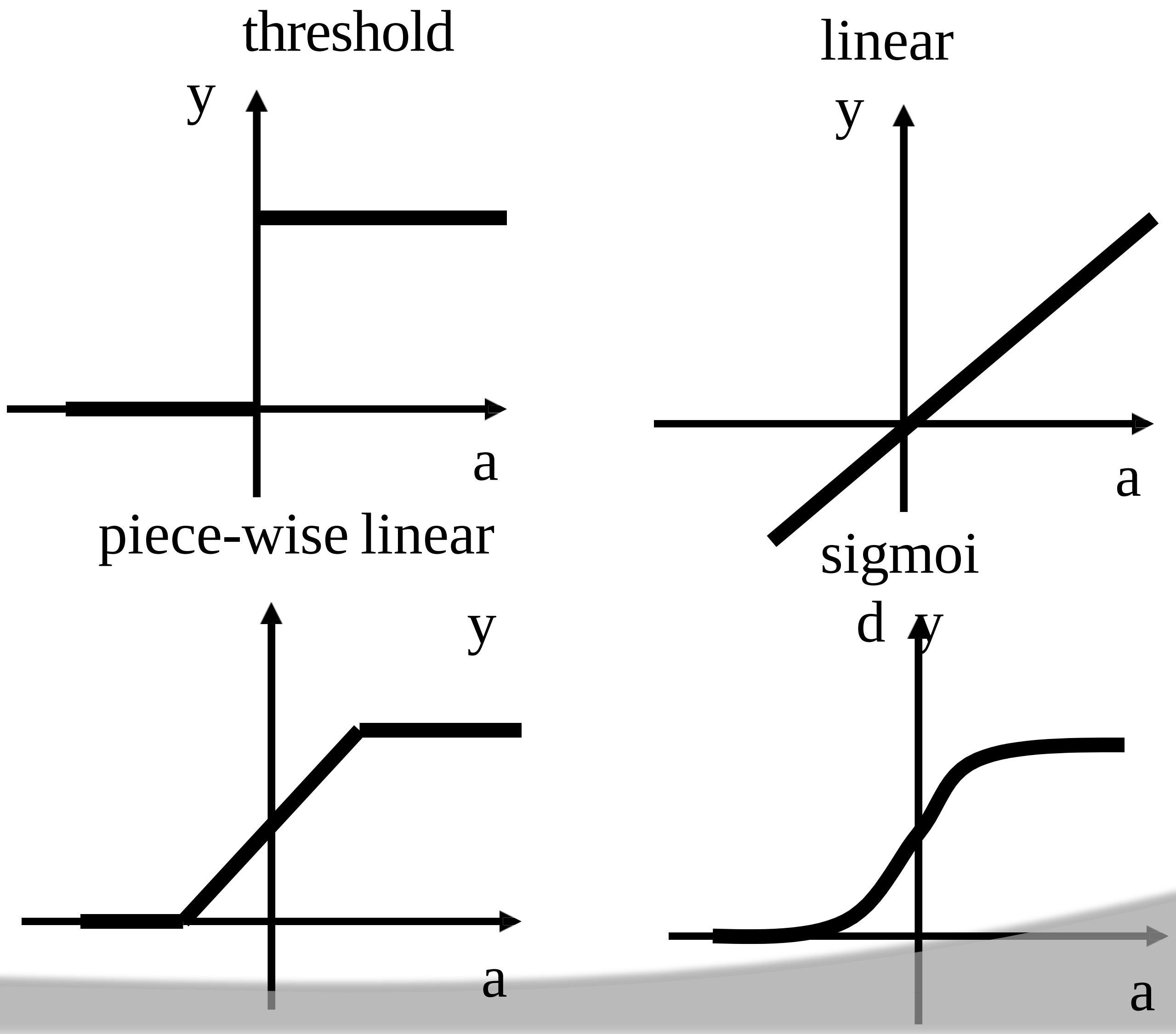
Threshold Logic Unit (TLU)



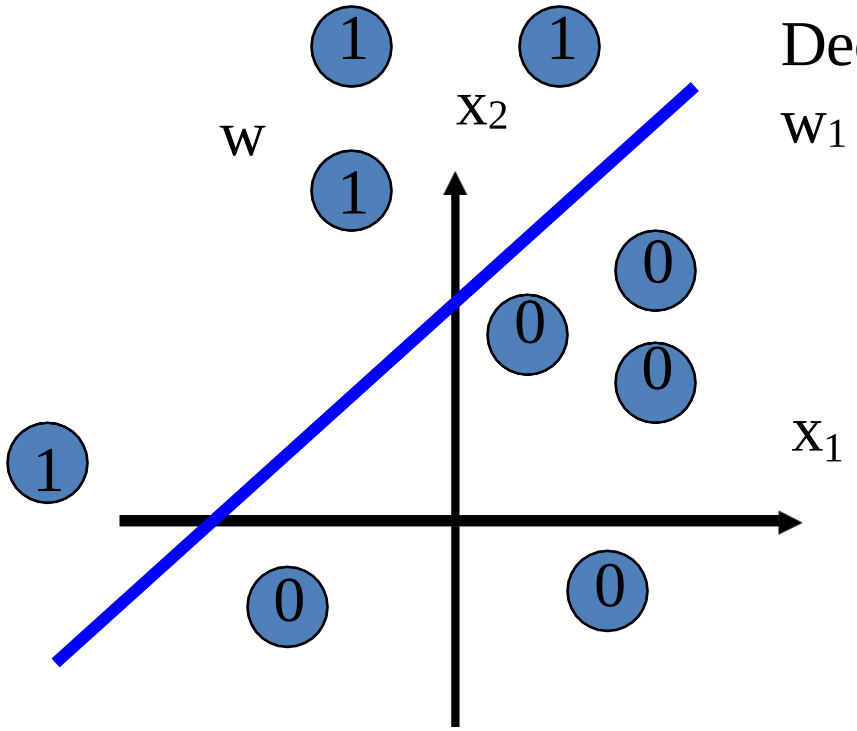
$$a=\Sigma_{i=1}^{n} w_i x_i$$

$$y= \begin{cases} 1 \text{ if } a \geq q \\ 0 \text{ if } a < q \end{cases}$$

Activation Functions



Decision Surface of a TLU

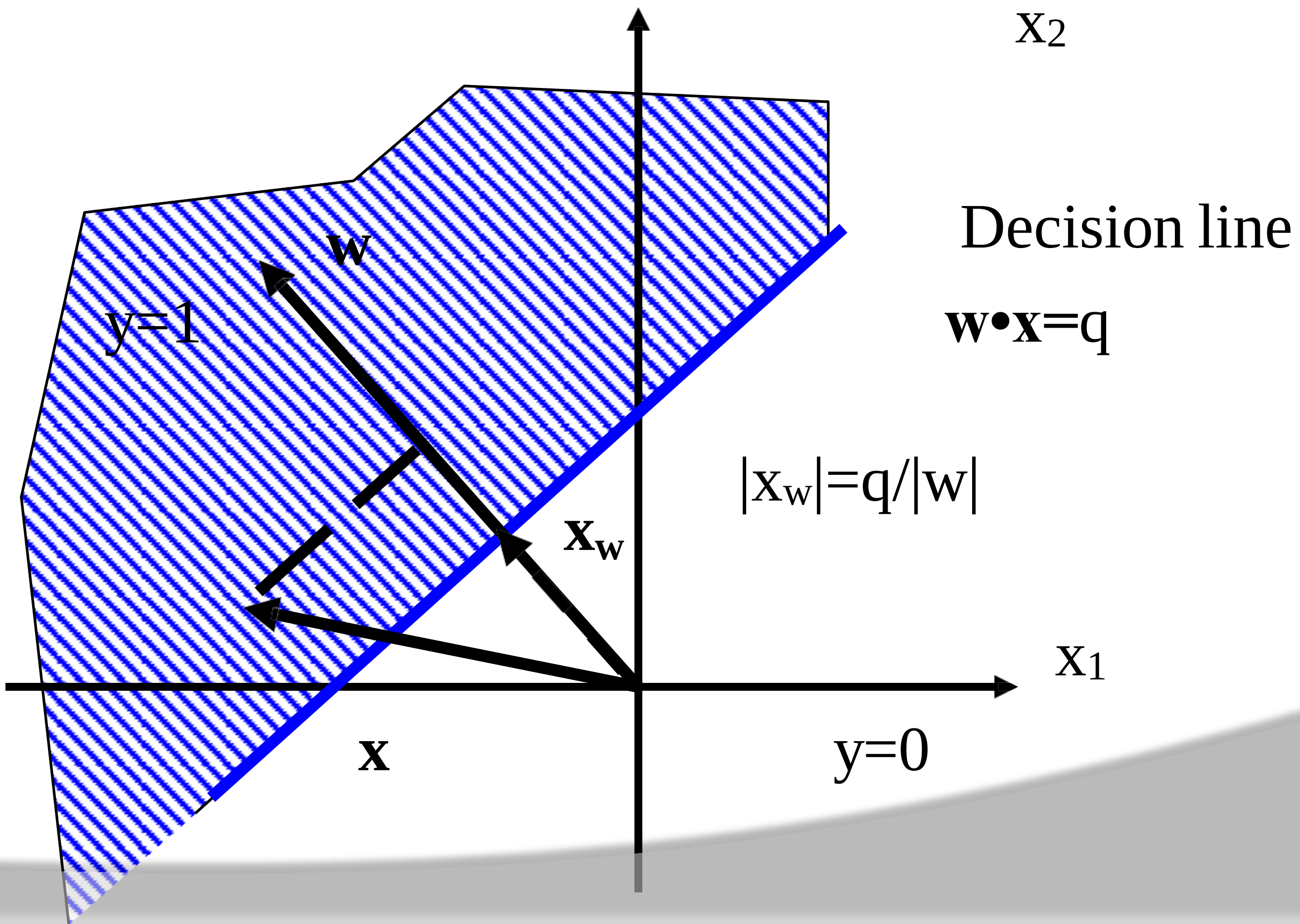


Decision line

$w_1 x_1 + w_2 x_2 = q$

Geometric Interpretation

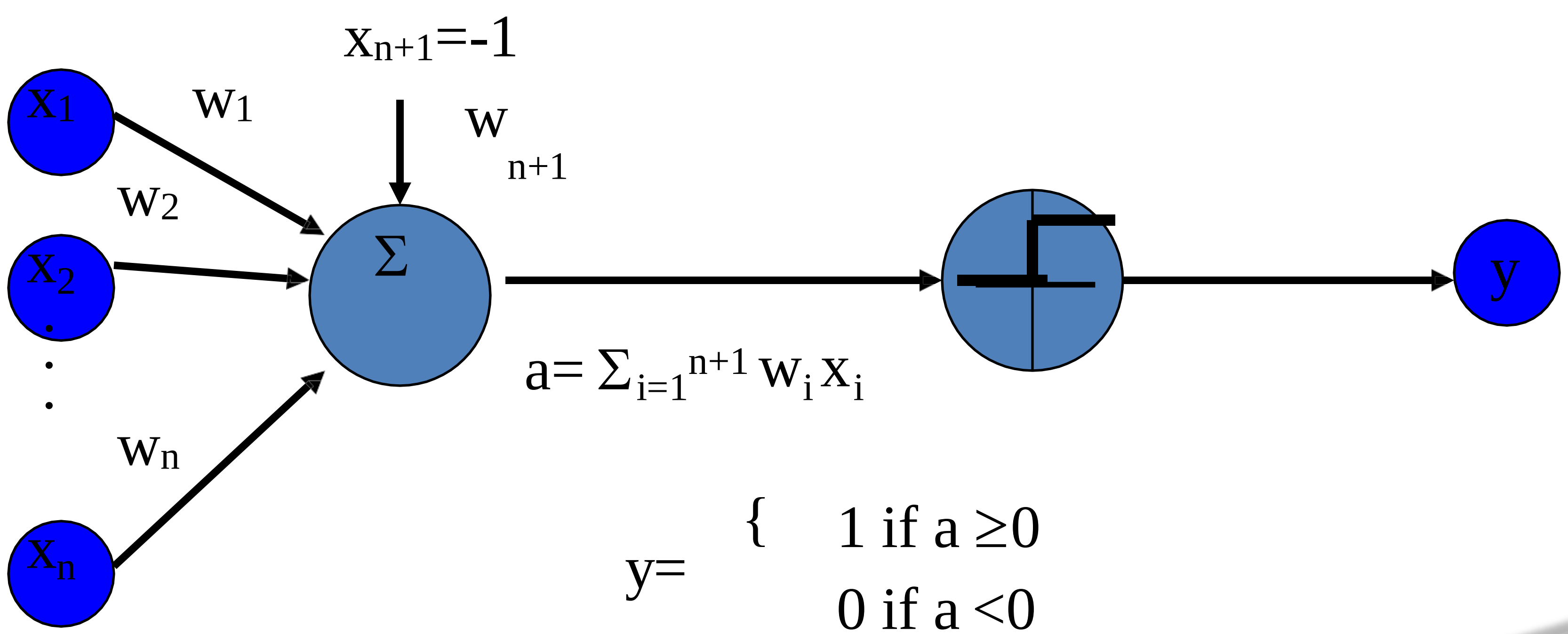
The relation **w•x**=q defines the decision line



Geometric Interpretation

- In n dimensions the relation **w•x**=q defines a n-1 dimensional hyper-plane, which is perpendicular to the weight vector **w**.
- On one side of the hyper-plane (**w•x**>q) all patterns are classified by the TLU as “1”, while those that get classified as “0” lie on the other side of the hyper-plane.
- If patterns can be not separated by a hyper-plane then they cannot be correctly classified with a TLU.

Threshold as Weight



$a= \Sigma_{i=1}^{n+1} w_i x_i$

$$y = \begin{cases} 1 \text{ if } a \geq 0 \\ 0 \text{ if } a < 0 \end{cases}$$

Training ANNs

- Training set S of examples {**x**,**t**}
  - **x** is an input vector and
  - **t** the desired target vector
  - Example: Logical And

    S = {(0,0),0}, {(0,1),0}, {(1,0),0}, {(1,1),1}
- Iterative process
  - Present a training example x , compute network output y , compare output y with target t, adjust weights and thresholds
- Learning rule
  - Specifies how to change the weights w and thresholds q of the network as a function of the inputs x, output y and target t.

Perceptron Learning Rule

- **w'**=**w** + a (t-y) **x**

  Or in components
- $w'_i = w_i + Dw_i = w_i + a\ (t\text{-}y)$ (i=1..n+1)

  x_i With $w_{n+1} = q$ and x_{n+1}=-1
- The parameter a is called the *learning rate*. It determines the magnitude of weight updates Dw_i .
- If the output is correct (t=y) the weights are not changed (Dw_i =0).
- If the output is incorrect (t ? y) the weights w_i are changed such that the output of the TLU for the new weights w'_i is *closer/further* to the input x_i.

Perceptron Training Algorithm

```
Repeat
for each training vector pair ($\mathbf{x}$,t)
evaluate the output y when $\mathbf{x}$ is the input if y            t then
form a new weight vector $\mathbf{w'}$ according to $\mathbf{w'}=\mathbf{w} + a\,(t-y)\,\mathbf{x}$
else
do nothing  end if
end for
Until y=t for all training vector pairs
```

Perceptron Training Algorithm

The algorithm converges to the correct classification

–if the training data is linearly separable

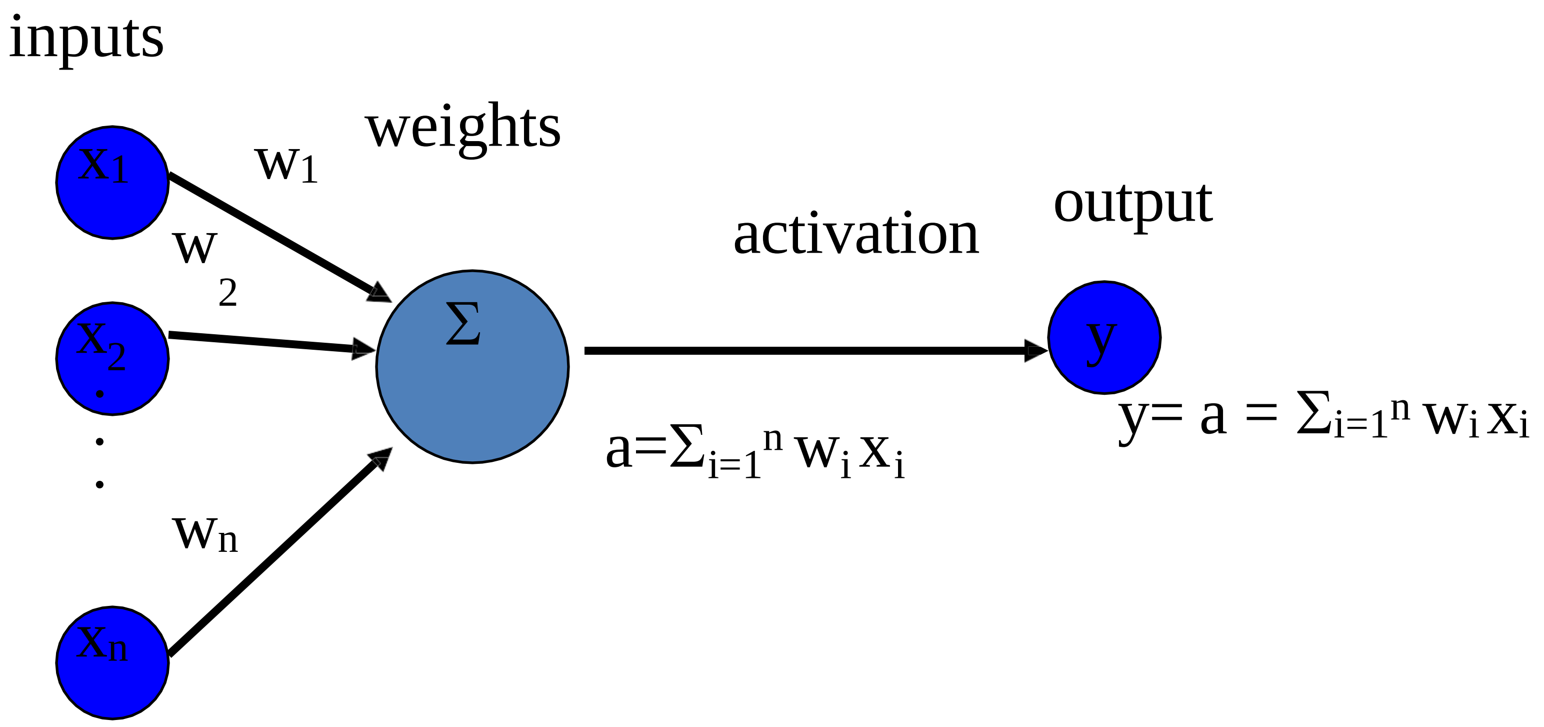
–and⍰ is sufficiently small

•If two classes of vectors X_1 and X_2 are linearly separable, the application of the perceptron training algorithm will eventually result in a weight vector $\mathbf{w_0}$, such that $\mathbf{w_0}$ defines a TLU whose decision hyper-plane separates X_1 and X_2 (Rosenblatt 1962).

•Solution $\mathbf{w_0}$ is not unique, since if $\mathbf{w_0}\,\mathbf{x} = 0$ defines a hyper- plane, so does

$\mathbf{w'_0} = k\,\mathbf{w_0}$.

Perceptron Training Algorithm



$a=\Sigma_{i=1}^{n} w_i x_i$

$y = a = \Sigma_{i=1}^{n} w_i x_i$

Perceptron Training Algorithm

- Consider linear unit without threshold and continuous output o (not just–1,1)
  - $o=w_0 + w_1 x_1 + \ldots + w_n x_n$
- Train the w_i's such that they minimize the squared error
  - $E[w_1,\ldots,w_n] = ½ \; _{d\ D}(t_d-o_d)^2$

  where D is the set of training examples

Gradient Descent

D={<(1,1),1>,<(-1,-1),1>,
<(1,-1),-1>,<(-1,1),-1>}

Gradient:

$\nabla E[w]=[\partial E/\partial w_0, \ldots \partial E/\partial w_n]$

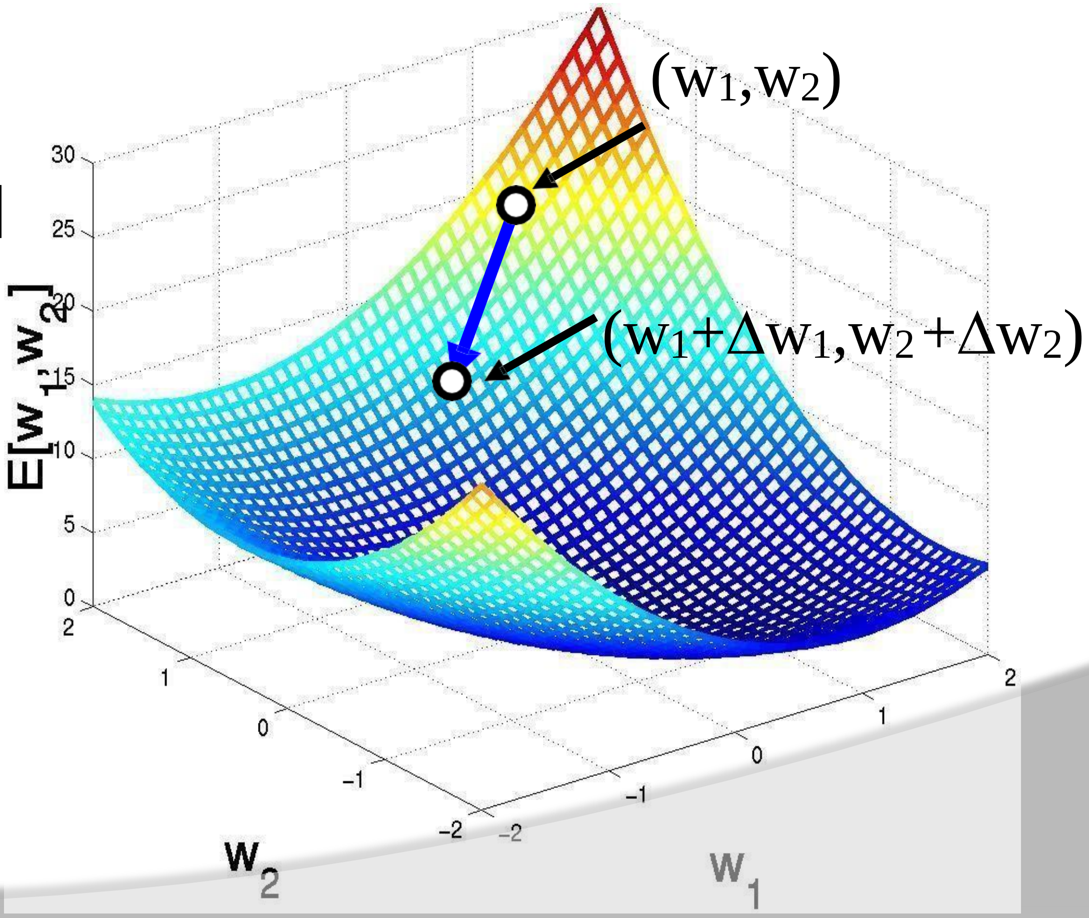
$\Delta w=-\eta \nabla E[w]$

$\Delta w_i=-\eta\, \partial E/\partial w_i$

$=\partial/\partial w_i\, 1/2\Sigma_d(t_d-o_d)^2$

$= \partial/\partial w_i\, 1/2\Sigma_d(t_d-\Sigma_i\, w_i\, x_i)^2$

$= \Sigma_d(t_d- o_d)(-x_i)$



Gradient Descent

Kernels

•The goal is a certain transformation $\mathbf{x}_i \rightarrow \mathbf{\Phi}(\mathbf{x}_i)$, such that problem becomes linearly separable (can be high- dimensional)

•Kernel: Function that is depict able as inner product of **Φ***s*:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{\Phi}(\mathbf{x}_1)\mathbf{\Phi}(\mathbf{x}_2)^T$$

- **Φ** does not have to be explicitlyknown

$$f(\mathbf{x}) = \sum_i t_i K(\mathbf{x}_i, \mathbf{x}) + w_0$$

Gradient Descent

Example: polynomial kernel

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}\mathbf{z}^T)^2$$

- 2 dimensions:

$$(\mathbf{x}\mathbf{z}^T)^2 = (x_1 z_1 + x_2 z_2)^2 =$$
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 =$$
$$= \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2\right)\left(z_1^2, z_2^2, \sqrt{2} z_1 z_2\right)^T =$$
$$= \mathbf{\Phi}(\mathbf{x})\mathbf{\Phi}(\mathbf{z})$$

•Kernel is indeed an inner product of vectors after transformation („preproces ing“)

Gradient Descent

The effect of the kernel trick

- Use of the kernel, e.g:

$$f(\mathbf{x}) = \sum_i t_i K(\mathbf{x}_i, \mathbf{x}) + w_0 = \sum_i t_i (\mathbf{x}_i^T \mathbf{x})^5 + w_0$$

- 16x16-dimensional vectors (e.g. pixel images), 5th degree polynomial: dimension = 10^{10}
  - Inner product of two 10000000000-dim. vectors
- Calculation is done in low-dimensional space:
  - Inner Product of two 256-dim. vectors
  - To the power of 5

Gradient Descent

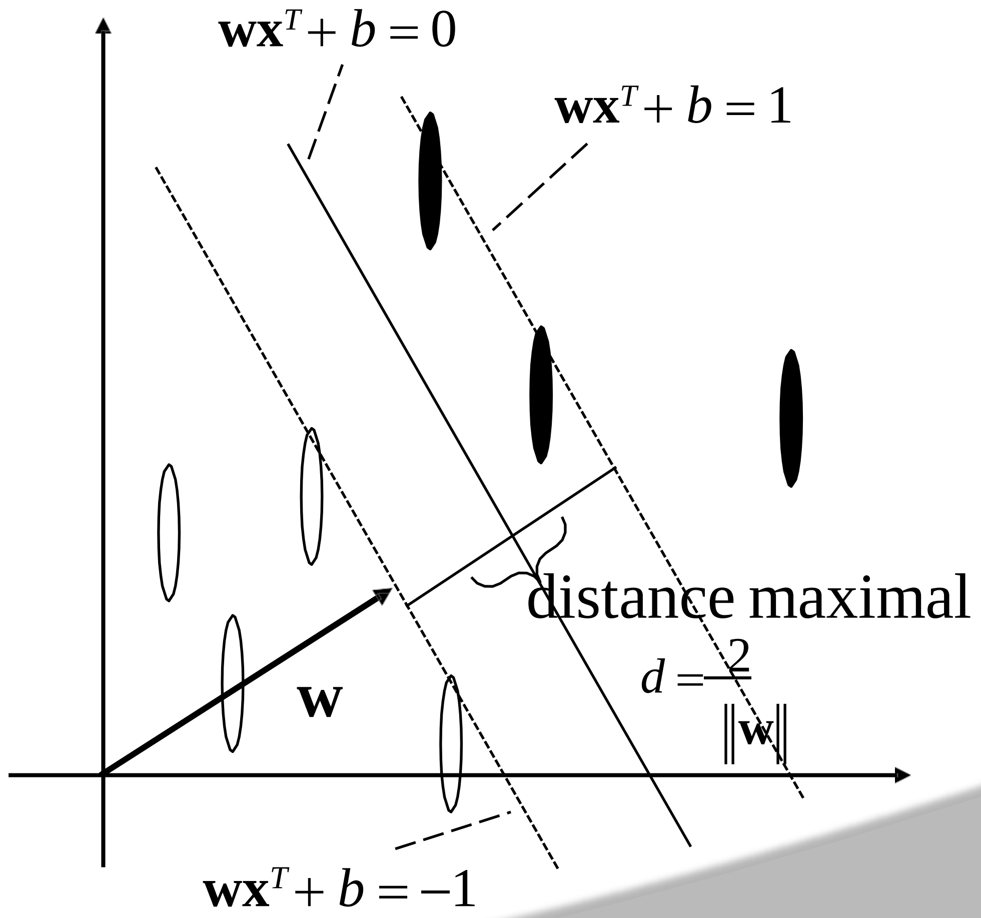
•High dimensional space:
Over fitting easily possible

- Solution: Search for decision border (hyper plabe) with largest distance to closest points

- Optimization:

Minimize $\frac{\|\mathbf{w}\|^2}{2}$

(Maximize $d = \frac{2}{\|\mathbf{w}\|}$)


$\mathbf{w}\mathbf{x}^T + b = 0$
$\mathbf{w}\mathbf{x}^T + b = 1$
distance maximal
$d = \frac{2}{\|\mathbf{w}\|}$
$\mathbf{w}$
$\mathbf{w}\mathbf{x}^T + b = -1$

Optimization of large margin classifier

- Quadratic optimization problem, Lagrange multiplier approach, leads to:

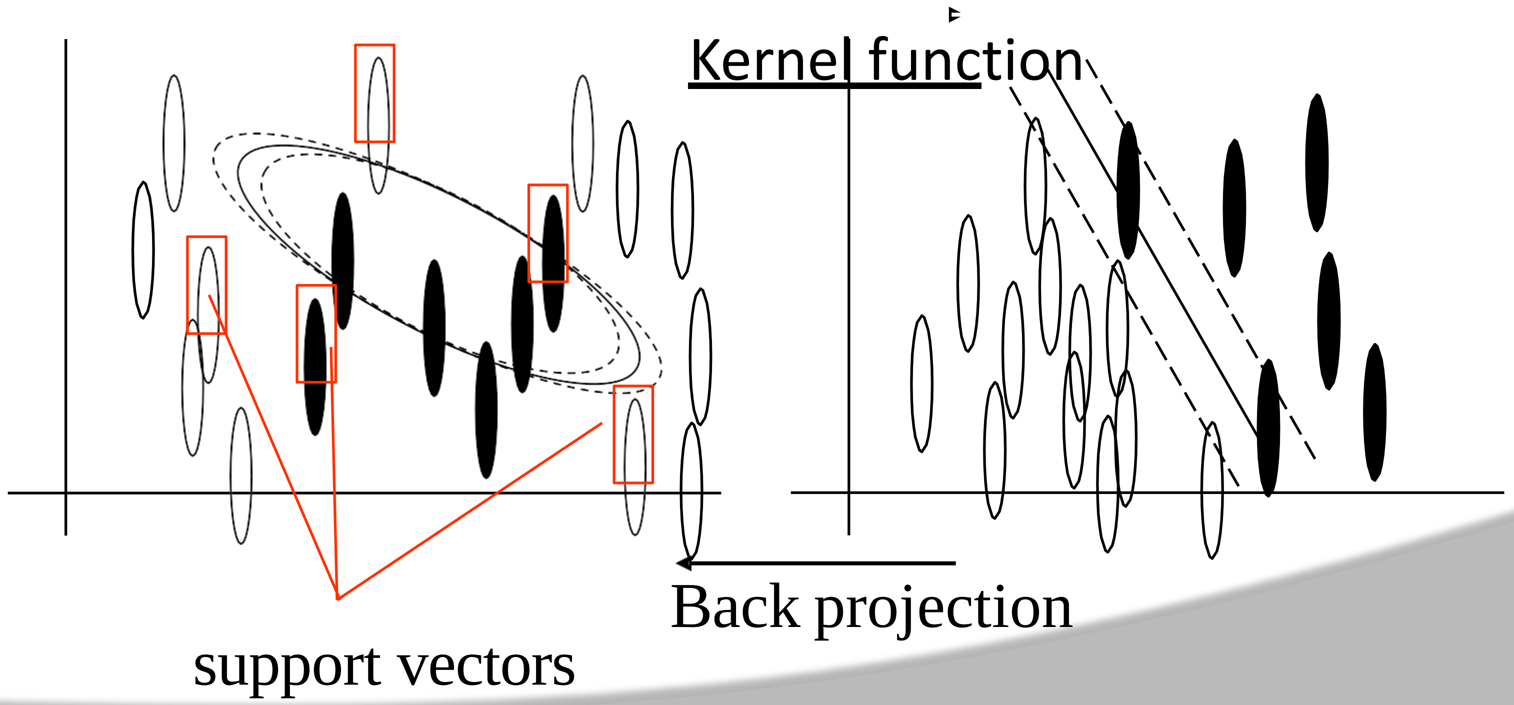
$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \underbrace{\mathbf{x}_i \mathbf{x}_j^T} \rightarrow \min$$

- „Dual“ form
- Important: Data is again denoted in terms of inner products
- Kernel trick can be usedagain

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \rightarrow \min$$

Optimization of large margin classifier

- Support-Vectors: Points at the margin (closest todecision border
- Determine the solution, all other points couldbeomitted



Optimization of large margin classifier

- All of the computations that we need to do to find the maximum- margin separator can be expressed in terms of scalar products between pairs of data points (in the high-dimensional feature space).
- These scalar products are the only part of the computation that depends on the dimensionality of the high-dimensional space.
  - So if we had a fast way to do the scalar products we would not have to pay a price for solving the learning problem in the high-D space.
- The kernel trick is just a magic way of doing scalar products a whole lot faster than is usually possible.
  - It relies on choosing a way of mapping to the high-dimensional feature space that allows fast scalar products.

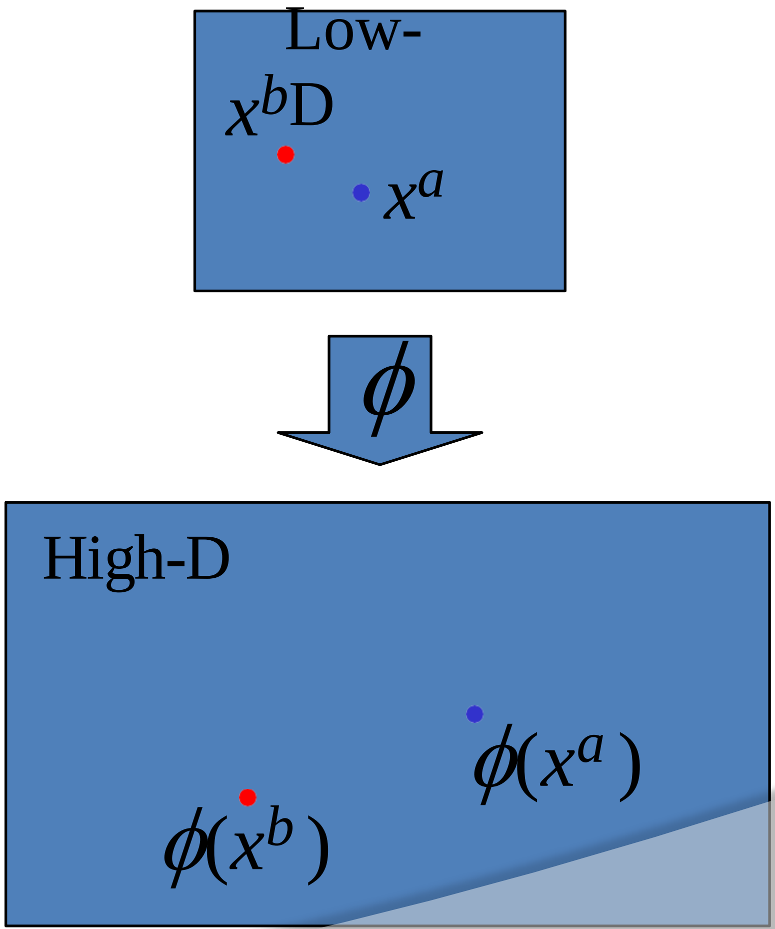
Optimization of large margin classifier

low-D space to a high-D space, there is a simple operation on two vectors in the low-D space that can be used to compute the scalar product of their two images in the high-D space.

$$K(x^a, x^b) = \phi(x^a) . \phi(x^b)$$

Letting the kernel do the work

doing the scalar product in the obvious way



Optimization of large margin classifier

- If we choose a mapping to a high-D space for which the kernel trick works, we do not have to pay a computational price for the high-dimensionality when we find the best hyper-plane.
  - We cannot express the hyperplane by using its normal vector in the high-dimensional space because this vector would have a huge number of components.
  - Luckily, we can express it in terms of the support vectors.
- But what about the test data. We cannot compute the scalar product because its in the high-D space. **w** .?(**x**)

Dealing with the test data

- We need to decide which side of the separating hyperplane a test point lies on and this requires us to compute a scalar product.
- We can express this scalar product as a weighted average of scalar products with the stored support vectors

– This could still be slow if there are a lot of support vectors .

Dealing with the test data

- The final classification rule is quite simple:

The set of support vectors

- All the cleverness goes into selecting the support vectors that maximize the margin and computing the weight to use on each support vector.

⬆

- We also need to choose a good kernel function and we may need to choose a lambda for dealing with non-separable cases.

Dealing with the test data

- Support Vector Machines work very well in practice.
  - The user must choose the kernel function and its parameters, but the rest is automatic.
  - The test performance is very good.
- They can be expensive in time and space for big datasets
  - The computation of the maximum-margin hyper-plane depends on the square of the number of training cases.
  - We need to store all the support vectors.
- SVM's are very good if you have no idea about what structure to impose on the task.
- The kernel trick can also be used to do PCA in a much higher- dimensional space, thus giving a non-linear version of PCA in the original space.

Dealing with the test data

•Neural networks are powerful machine learners for numerical features, initially inspired by neurophysiology

•Nonlinearity through interplay of simpler learners (perceptions)

•Statistical/probabilistic framework most appropriate

•Learning = Maximum Likelihood, minimizing error function with efficient gradient-based method (e.g. conjugant gradient)

•Power comes with downsides (over fitting) -> careful validation necessary

•Support vector machines are interesting alternatives, simplify learning problem through „Kernel trick“.

UNIT III
BASIC STATISTICS

BASIC STATISTICS

- Averages, variance and covariance, the Gaussian; The bias-variance tradeoff Bayesian learning: Introduction, Bayes theorem, Bayes optimal classifier, naïve Bayes classifier. Graphical models: Bayesian networks, approximate inference, making Bayesian networks, hidden Markov models, the forward algorithm.

Mean and Variance

•Two properties of a random variable that are often of interest are its expected value (also called its mean value) and its variance. The expected value is the average of the values taken on by repeatedly sampling the random variable.

•Consider a random variable *Y* that takes on the possible values yl, . . . *yn.* The expected value of *Y , E [Y] ,*is
For example, if *Y* takes on the value 1 with probability .7 and the value 2 with probability .3, then its expected value is (1 .0.7 + 2.0.3 = 1.3).

•In case the random variable *Y* is governed by a Binomial distribution, then it can be shown that In case the random variable Y is governed by a Binomial distribution, then the variance and standard deviation.

Estimators, Bias, and Variance

- Statisticians call errors(h) an estimator for the true error errorv(h).
- We define the estimation bias to be the difference between the expected value of the estimator and the true value of the parameter.
- If the estimation bias is zero, we say that Y is an unbiased estimator for p
- Is errors (h) an unbiased estimator for errorv(h)? Yes, because for a Bi-nominal distribution the expected value of r is equal to np

Bayesian

- Bayesian learning methods are relevant to our study of machine learning for two different reasons.
- First, Bayesian learning algorithms that calculate explicit probabilities for hypotheses,
- such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems

Bayesian

- The second reason that Bayesian methods are important to our studyof machine learning
- They provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities

BAYES THEOREM

- Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability,
- The probabilities of observing various data given the hypothesis, and the observed data itself.
- As one might intuitively expect, *P(h ID)* increases with *P(h)* and with *P(D1h)* according to Bayes theorem. It is also reasonable to see that *P(hl D)* decreases as *P (D)* increases,
- Because the more probable it is that *D* will be observed independent of *h ,* the less evidence *D* provides in support of *h.*

MAP

- In many learning scenarios, the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis *h* E H given the observed data *D* (or at least one of the maximally probable if there are several).
- Any such maximally probable hypothesis is called a *maximum* a *posteriori* (MAP) hypothesis.
- We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.
- More precisely, we will say that *MAP* is a MAP hypothesis provided

Bayes Theorem and Concept Learning

- What is the relationship between Bayes theorem and the problem of concept learning? Since Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data.
- we can use it as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable. This section considers such a brute-force Bayesian concept learning algorithm, then compares it to concept learning algorithms
- As we shall see, one interesting result of this comparison is that under certain conditions several algorithms discussed in earlier chapters output the same hypotheses as this brute-force Bayesian algorithm, despite the fact that they do not explicitly manipulate probabilities and are considerably moreefficient.

Brute-Force Bayes Concept Learning

- Consider the concept learning problem first introduced in Chapter 2. In particular, assume the learner considers some finite hypothesis space H defined over the instance space X, in which the task is to learn some target concept *c* : X + {0,1}.
- As usual, we assume that the learner is given some sequence of training examples *((x ~d,l). . . (xm,d m))*where *xi* is some instance from X and where *di* is the target value of *xi* (i.e., *di* = *c(xi)) .*
- To simplify the discussion in this section, we assume the sequence of instances *(xl . . .xm)*is held fixed, so that the training data D can be written simply as the sequence of target values D = *(dl . . .d m) .*
- It can be shown that this simplification does not alter the main conclusions of this section.

BRUTE-FORCEMAP Learning algorithm

- For each hypothesis *h* in H, calculate the posterior probability
- Output the hypothesis *hMAP*with the highest posterior probability
- This algorithm may require significant computation, because it applies Bayes theorem to each hypothesis in H to calculate *P(hJD) .*
- While this may prove impractical for large hypothesis spaces, the algorithm is still of interest because it provides a standard against which we may judge the performance of other concept learning algorithms.

MAP Hypotheses and Consistent Learners

- The above analysis shows that in the given setting, every hypothesis consistent with *D* is a MAP hypothesis.
- This statement translates directly into an interesting statement about a general class of learners that we might call *consistent learners.*
- We will say that a learning algorithm is a *consistent learner* provided it outputs a hypothesis that commits zero errors over the training examples.
- Given the above analysis, we can conclude that *every consistent learner outputs a MAP hypothesis,*
- *If we assume a uniform prior probability distribution over H (i.e., P(hi) = P(hj) for all i, j), and ifwe assume deterministic, noisefree training data (i.e., P(D Ih) = 1 i f D and h are consistent, and 0 otherwise).*

Bayesian Belief Networks

- A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities.
- In contrast to the naive Bayes classifier, which assumes that *all* the variables are conditionally independent given the value of the target variable, Bayesian belief networks allow stating conditional independence assumptions that apply to *subsets* of the variables.
- Thus, Bayesian belief networks provide an intermediate approach that is less constraining than the global assumption of conditional independence made by the naive Bayes classifier, but more tractable than avoiding conditional independence assumptions altogether.

Inference

- We might wish to use a Bayesian network to infer the value of some target variable (e.g., *ForestFire)* given the observed values of the other variables.
- Of course, given that we are dealing with random variables it will not generally be correct to assign the target variable a single determined value.
- What we really wish to infer is the probability distribution for the target variable, which specifies the probability that it will take on each of its possible values given the observed values of the other variables.
- This inference step can be straightforward if values for all of the other variables in the network are known exactly.
- In the more general case we may wish to infer the probability distribution for some variable given observed values for only a subset of the other variables

Learning Bayesian Belief Networks

- First, the network structure might be given in advance, or it might have to be inferred from the training data. Second, all the network variables might be directly observable in each training example, or some might be unobservable.
- In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward.
- We simply estimate the conditional probability table entries just as we would for a naive Bayes classifier.

Gradient Ascent Training of Bayesian Networks

- The gradient ascent rule given by Russell et al. (1995) maximizes P(D1h) by following the gradient of In P(D Ih) with respect to the parameters that define the conditional probability tables of the Bayesian network.
- Let *wi;k* denote a single entry in one of the conditional probability tables. In particular, let *wijk* denote the conditional probability that the network variable *Yi* will take on the value *yi,* given that its immediate parents *Ui* take on the values given by *uik.*
- For example, if *wijk* is the top right entry in the conditional probability table in Figure 6.3, then *Yi* is the variable *Campjire, Ui* is the tuple of its parents *(Stomz,BusTourGroup), yij = True,* and *uik = (False,False).*
- The gradient of In P(D1h) is given by the derivatives for each of the *toijk.*

Learning the Structure of Bayesian Networks

- Cooper and Herskovits (1992) present a Bayesian scoring metric for choosing among alternative networks.
- They also present a heuristic search algorithm called K2 for learning network structure when the data is fully observable.
- Like most algorithms for learning the structure of Bayesian networks, K2 performs a greedy search that trades off network complexity for accuracy over the training data.
- In one experiment K2 was given a set of 3,000 training examples generated at random from a manually constructed Bayesian network containing 37 nodes and 46 arcs.
- This particular network described potential anesthesia problems in a hospital operating room.

EM Algorithm

- The EM algorithm can be used even for variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known.
- The EM algorithm has been used to train Bayesian belief networks as well as radial basis function networks discussed .
- The EM algorithm is also the basis for many unsupervised clustering algorithms (e.g., Cheeseman et al. 1988), and it is the basis for the widely used Baum-Welch forward-backward algorithm for learning Partially Observable Markov Models (Rabiner 1989).

Estimating Means of k Gaussians

- First, one of the k Normal distributions is selected at random.
- Second, a single random instance *xi* is generated according to this selected distribution.
- This process is repeated to generate a set of data points.we consider the special case where the selection of the single Normal distribution at each step is based on choosing each with uniform probability, where each of the k Normal distributions has the same variance a2,and where a2 is known.
- The learning task is to output a hypothesis h = (FI, . . . *pk)* that describes the means of each of the k distributions.

UNIT IV
EVOLUTIONARY LEARNING

EVOLUTIONARY LEARNING

- Genetic Algorithms, genetic operators; Genetic programming; Ensemble learning: Boosting, bagging; Dimensionality reduction: Linear discriminate analysis, principal component analysis (JAX-RPC).

GENETIC ALGORITHMS

•The problem addressed by GAS is to search a space of candidate hypotheses to identify the best hypothesis.

•In GAS the "best hypothesis" is defined as the one that optimizes a predefined numerical measure for the problem at hand, called the hypothesis *Jitness.*

•For example, if the learning task is the problem of approximating an unknown function given training examples of its input and output, then fitness could be defined as the accuracy of the hypothesis over this training data.

•If the task is to learn a strategy for playing chess, fitness could be defined as the number of games won by the individual when playing against other individuals in the current population.

GENETIC ALGORITHMS

- Although different implementations of genetic algorithms vary in their de-tails, they typically share the following structure: The algorithm operates by itera-tively updating a pool of hypotheses, called the population.
- On each iteration, all members of the population are evaluated according to the fitness function.
- A new population is then generated by probabilistically selecting the most fit individuals from the current population.
- Some of these selected individuals are carried forward into the next generation population intact.
- Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation.

GENETIC ALGORITHMS

- Fitness: A function that assigns an evaluation score, given a hypothesis.
- Fitnessdhreshold: A threshold specifying the termination criterion.
- The number of hypotheses to be included in the population.
- The fraction of the population to be replaced by Crossover at each step The mutation rate.
- Initialize population: P c Generate p hypotheses at random
- Evaluate: For each h in P , compute Fitness(h)' While [max Fitness(h)]< Fitnessdhreshold do h

GENETIC ALGORITHMS

- *Select:* F'robabilistically select (1 - *r) p* members of *P* to add to *Ps.* The probability Pr(hi) of selecting hypothesis *hi* from *P* is given by
- *Crossover:* Probabilistically select pairs of hypotheses from *P ,* according to *&(hi)* given above. For each pair, *(h l ,h2),*produce two offspring by applying the Crossover operator. Add all offspring to *P,.*
- *Mutate:* Choose *m* percent of the members of *P,* with uniform probability. For each, invert one randomly selected bit in its representation.
- *Update: P* to *P,.*
- *5. Evaluate:* for each *h* in *P ,* compute *Fitness(h)*
- Return the hypothesis from *P* that has the highest fitness.

Representing Hypotheses

- Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover.
- The hypotheses represented by these bit strings can be quite complex.
- For example, sets of if-then rules can easily be represented in this way, by choosing an encoding of rules that allocates specific substrings for each rule precondition and postcondition.
- Examples of such rule representations in GA systems are described by Holland (1986); Grefenstette (1988); and DeJong et al. (1993).

Representing Hypotheses

- To see how if-then rules can be encoded by bit strings, .first consider how we might use a bit string to describe a constraint on the value of a single attribute.
- To pick an example, consider the attribute *Outlook,* which can take on any of the three values *Sunny, Overcast,* or *Rain.* One obvious way to represent a constraint on *Outlook* is to use a bit string of length three, in which each bit position corresponds to one of its three possible values.
- Placing a 1 in some position indicates that the attribute is allowed to take on the corresponding value.
- For example, the string 010 represents the constraint that *Outlook* must take on the second of these values, , or *Outlook* = *Overcast.* Similarly, the string 011 represents the more general constraint that allows two possible values, or *(Outlook* = *Overcast v Rain).*

Genetic Operators

- The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population.
- Typical GA operators for manipulating bit string hypotheses. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are *crossover* and *mutation.*
- The *crossover operator* produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which parent contributes the bit for position i is determined by an additional string called the *crossover mask*

Genetic Operators

- In *two-point crossover,* offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Put another way, the crossover mask is a string beginning with *no* zeros, followed by a contiguous string of *nl* ones, followed by the necessary number of zeros to complete the string.
- Each time the two-point crossover operator is applied, a mask is generated by randomly choosing the integers *no* and *nl.* For instance, the offspring are created using a mask for which *no* = 2 and *n* 1 = 5. Again, the two offspring are created by switching the roles played by the two parents.
- *Uniform crossover* combines bits sampled uniformly from the two parents. In this case the crossover mask is generated as a random bit string with each bit chosen at random and independent of the others.

Genetic Operators

- In addition to recombination operators that produce offspring by combining parts of two parents, a second type of operator produces offspring from a single parent.
- In particular, the *mutation* operator produces small random changes to the bit string by choosing a single bit at random, then changing its value.
- Mutation is often performed after crossover has been applied as in our prototypical algorithm

Fitness Function and Selection

- The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population.
- If the task is to learn classification rules, then the fitness function typically has a component that scores the classification accuracy of the rule over a set of provided training examples.
- Often other criteria may be included as well, such as the complexity or generality of the rule. More generally, when the bit-string hypothesis is interpreted as a complex procedure (e.g., when the bit string represents a collection of if-then rules that will be chained together to control a robotic device).
- the fitness function may measure the overall performance of the resulting procedure rather than performance of individual rules.

GENETIC PROGRAMMING

- Genetic programming (GP) is a form of evolutionary computation in which the individuals in the evolving population are computer programs rather than bit strings.
- Koza (1992) describes the basic genetic programming approach and presents a broad range of simple programs that can be successfully learned by GP.

Representing Programs

- Programs manipulated by a GP are typically represented by trees corresponding to the parse tree of the program. Each function call is represented by a node in the tree, and the arguments to the function are given by its descendant nodes.
- For example, this tree representation for the function sin(x) + *J-.* To apply genetic programming to a particular domain, the user must define the primitive functions to be considered (e.g., sin, cos, *J,*+, -, exponential~),as well as the terminals (e.g., x, *y ,* constants such as 2).
- The genetic programming algorithm then uses an evolutionary search to explore the vast space of programs that can be described using these primitives.

Representing Programs

- As in a genetic algorithm, the prototypical genetic programming algorithm maintains a population of individuals (in this case, program trees).
- On each iteration, it produces a new generation of individuals using selection, crossover, and mutation.
- The fitness of a given individual program in the population is typically determined by executing the program on a set of training data.
- Crossover operations are performed by replacing a randomly chosen subtree of one parent program by a subtree from the other parent program.

Remarks on Genetic Programming

- As illustrated in the above example, genetic programming extends genetic algorithms to the evolution of complete computer programs.
- Despite the huge size of the hypothesis space it must search, genetic programming has been demonstrated to produce intriguing results in a number of applications.
- A comparison of GP to other methods for searching through the space of computer programs, such as hillclimbing and simulated annealing, is given by O'Reilly and Oppacher (1994).
- While the above example of GP search is fairly simple, Koza et al. (1996) summarize the use of a GP in several more complex tasks such as designing electronic filter circuits and classifying segments of protein molecules. The filter circuit design problem provides an example of a considerably more complex problem.

Models Of Evolution And Learning

- In many natural systems, individual organisms learn to adapt significantly during their lifetime.
- At the same time, biological and social processes allow their species to adapt over a time frame of many generations.
- One interesting question regarding evolutionary systems is "What is the relationship between learning during the lifetime of a single individual, and the longer time frame species-level learning afforded by evolution?'

Lamarckian Evolution

- Lamarck was a scientist who, in the late nineteenth century, proposed that evolution over many generations was directly influenced by the experiences of individual organisms during their lifetime.
- In particular, he proposed that experiences of a single organism directly affected the genetic makeup of their offspring: If an individual learned during its lifetime to avoid some toxic food, it could pass this trait on genetically to its offspring, which therefore would not need to learn the trait.
- This is an attractive conjecture, because it would presumably allow for more efficient evolutionary progress than a generate-and-test process (like that of GAS and GPs) that ignores the experience gained during an individual's lifetime.

Lamarckian Evolution

- Despite the attractiveness of this theory, current scientific evidence overwhelmingly contradicts Lamarck's model.
- The currently accepted view is that the genetic makeup of an individual is, in fact, unaffected by the lifetime experience of one's biological parents.
- Despite this apparent biological fact, recent computer studies have shown that Lamarckian processes can sometimes improve the effectiveness of computerized genetic algorithms (see Grefenstette 1991; Ackley and Littman 1994; and Hart and Belew 1995).

Baldwin Effect

- Although Lamarckian evolution is not an accepted model of biological evolution, other mechanisms have been suggested by which individual learning can alter the course of evolution.
- One such mechanism is called the Baldwin effect, after J. M. Baldwin (1896), who first suggested the idea. The Baldwin effect is based on the following observations.
- If a species is evolving in a changing environment, there will be evolutionary pressure to favor individuals with the capability to learn during their lifetime.
- For example, if a new predator appears in the environment, then individuals capable of learning to avoid the predator will be more successful than individuals who cannot learn.

Baldwin Effect

- In effect, the ability to learn allows an individual to perform a small local search during its lifetime to maximize its fitness. In contrast, nonlearning individuals whose fitness is fully determined by their genetic makeup will operate at a relative disadvantage.
- Those individuals who are able to learn many traits will rely less strongly on their genetic code to "hard-wire" traits. As a result, these individuals can support a more diverse gene pool, relying on individual learning to overcome the "missing" or "not quite optimized" traits in the genetic code.

Parallelizing Genetic Algorithms

- GAS are naturally suited to parallel implementation, and a number of approaches to parallelization have been explored.
- *Coarse grain* approaches to parallelization subdivide the population into somewhat distinct groups of individuals, called *demes.*
- Each deme is assigned to a different computational node, and a standard GA search is performed at each node.
- Communication and cross-fertilization between demes occurs on a less frequent basis than within demes.

Parallelizing Genetic Algorithms

- Transfer between demes occurs by a *migration* process, in which individuals from one deme are copied or transferred to other demes.
- This process is modeled after the kind of cross-fertilization that might occur between physically separated subpopulations of biological species.
- One benefit of such approaches is that it reduces the crowding problem often encountered in nonparallel GAS, in which the system falls into a local optimum due to the early appearance of a genotype that comes to dominate the entire population.
- Examples of coarse-grained parallel GAS are described by Tanese (1989) and by Cohoon et al. (1987).

First-Order Rule

- First, they are designed to learn sets of first-order rules that contain variables. This is significant because first-order rules are much more expressive than propositional rules.
- Second, the algorithms discussed here use sequential covering algorithms that learn one rule at a time to incrementally grow the final set of rules.
- As an example of first-order rule sets, consider the following two rules that jointly describe the target concept *Ancestor.*
- Here we use the predicate *Parent(x,* y) to indicate that y is the mother or father of *x,* and the predicate *Ancestor(x, y)* to indicate that *y* is an ancestor of *x* related by an arbitrary number of family generations.

Sequential Covering Algorithms

- Here we consider a family of algorithms for learning rule sets based on the strategy of learning one rule, removing the data it covers, then iterating this process.
- Such algorithms are called *sequential covering* algorithms. To elaborate, imagine we have a subroutine LEARN-ONE-RULE that accepts a set of positive and negative training examples as input, then outputs a single rule that covers many of the positive examples and few of the negative examples.
- We require that this rule have high accuracy, but not necessarily high coverage. By high accuracy, we mean the predictions it makes should be correct.
- By accepting low coverage, we mean it need not make predictions for every training example.

Sequential Covering Algorithms

- Given this LEARN-ONE-RULE subroutine for learning a single rule, one obvious approach to learning a set of rules is to invoke LEARN-ONE-RULE on all the available training examples, remove any positive examples covered by the rule it learns, then invoke it again to learn a second rule based on the remaining training examples.
- This procedure can be iterated as many times as desired to learn a disjunctive set of rules that together cover any desired fraction of the positive examples.
- This is called a *sequential covering* algorithm because it sequentially learns a set of rules that together cover the full set of positive examples.

Specific Beam Search

- One effective approach to implementing LEARN-ONE-RULE is to organize the hypothesis space search in the same general fashion as the ID3 algorithm, but to follow only the most promising branch in the tree at each step.
- As illustrated in the search tree , the search begins by considering the most general rule precondition possible (the empty test that matches every instance), then greedily adding the attribute test that most improves rule performance measured over the training examples.
- Once this test has been added, the process is repeated by greedily adding a second attribute test, and so on.

Specific Beam Search

- Like ID3, this process grows the hypothesis by greedily adding new attribute tests until the hypothesis reaches an acceptable level of performance.
- Unlike ID3, this implementation of LEARN-ONE-RULE follows only a single descendant at each search step-the attribute-value pair yielding the best performance-rather than growing a subtree that covers all possible values of the selected attribute.
- This approach to implementing LEARN-ONE-RULE performs a general-to-specific search through the space of possible rules in search of a rule with high accuracy, though perhaps incomplete coverage of the data.

UNIT V
CLUSTERING

CLUSTERING

- Similarity and distance measures, outliers, hierarchical methods, partitional algorithms, clustering large databases, clustering with categorical attributes, comparison.

CLUSTERING

- Cluster analysis groups objects based on their similarity and has wide applications
- Measure of similarity can be computed for various types of data
- Clustering algorithms can be categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- Outlier detection and analysis are very useful for fraud detection, etc. and can be performed by statistical, distance-based or deviation-based approaches
- There are still lots of research issues on cluster analysis

Problems and Challenges

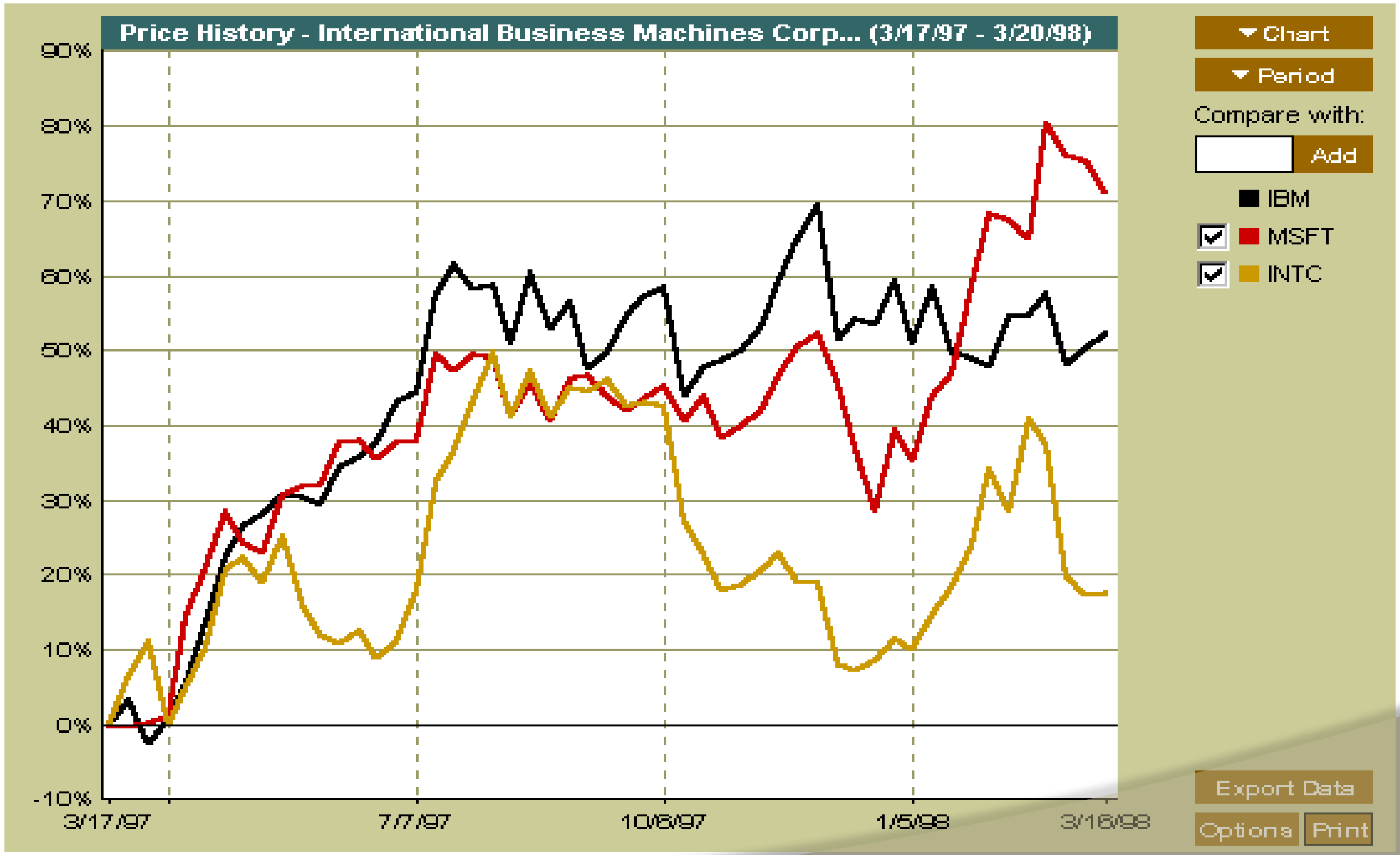
- Considerable progress has been made in scalable clustering methods
  - Partitioning: k-means, k-medoids, CLARANS
  - Hierarchical: BIRCH, ROCK, CHAMELEON
- Current clustering techniques do not address all the requirements adequately, still an active area of research

Mining Time-Series and Sequence Data

- Time-series database
  - Consists of sequences of values or events changing withtime
  - Data is recorded at regular intervals
  - Characteristic time-seriescomponents
    - Trend, cycle, seasonal, irregular
- Applications
  - Financial: stock price, inflation
  - Biomedical: blood pressure
  - Meteorological: precipitation

Mining Time-Series and Sequence Data

Time-series plot



Mining Time-Series and Sequence Data: Trend analysis

•A time series can be illustrated as a time-series graph which describes a point moving with the passage of time

•Categories of Time-Series Movements

- •Long-term or trend movements (trend curve)
- •Cyclic movements or cycle variations, e.g., business cycles Seasonal movements or seasonal variations
  - •i.e, almost identical patterns that a time series appears to follow during corresponding months of successive years.
- •Irregular or random movements

Estimation of Trend Curve

- The freehand method

  Fit the curve by looking at the graph

  Costly and barely reliable for large-scaled data mining
- The least-square method

  Find the curve minimizing the sum of the squares of the deviation of points on the curve from the corresponding data points
- The moving-average method

  Eliminate cyclic, seasonal and irregular patterns

  Sensitive to outliers

Discovery of Trend in Time-Series

Estimation of irregular variations

By adjusting the data for trend, seasonal and cyclic variations With the systematic analysis of the trend, cyclic, seasonal, and irregular components, it is possible to make long- or short-term predictions with reasonable quality

Similarity Search in Time-Series Analysis

- Normal database query finds exact match
- Similarity search finds data sequences that differ only slightly from the given query sequence
- Two categories of similarity queries
  - Whole matching: find a sequence that is similar to the query sequence
  - Subsequence matching: find all pairs of similar sequences
- Typical Applications
  - Financial market
  - Market basket data analysis
  - Scientific databases
  - Medical diagnosis

Data transformation

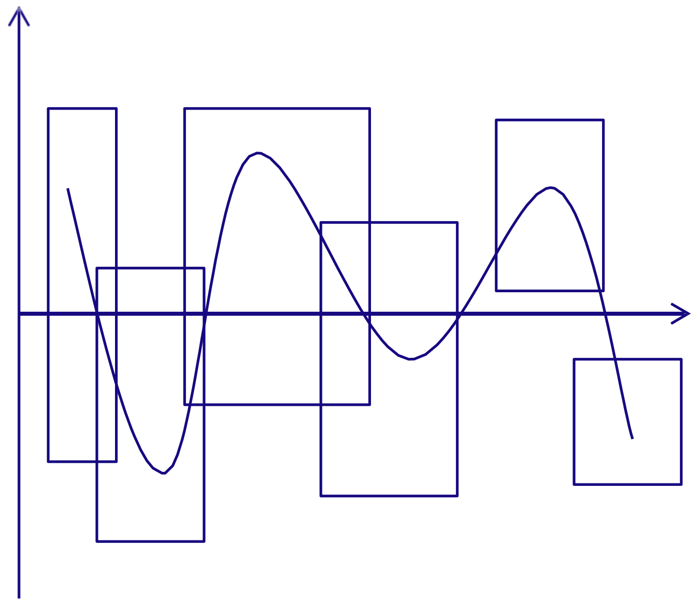
- Many techniques for signal analysis require the data to be in the frequency domain
- Usually data-independent transformations are used
  - The transformation matrix is determined a priori
    - E.g., discrete Fourier transform (DFT), discrete wavelet transform (DWT)
  - The distance between two signals in the time domain is the same as their Euclidean distance in the frequency domain DFT does a good job of concentrating energy in the first few coefficients
  - If we keep only first a few coefficients in DFT, we can compute the lower bounds of the actual distance

Multidimensional Indexing

- Multidimensional index
  - Constructed for efficient accessing using the first few Fourier
  - coefficients
- Use the index to retrieve the sequences that are at most a certain small distance away from the query sequence Perform post-processing by computing the actual distance between sequences in the time domain and discard any false matches

Subsequence Matching

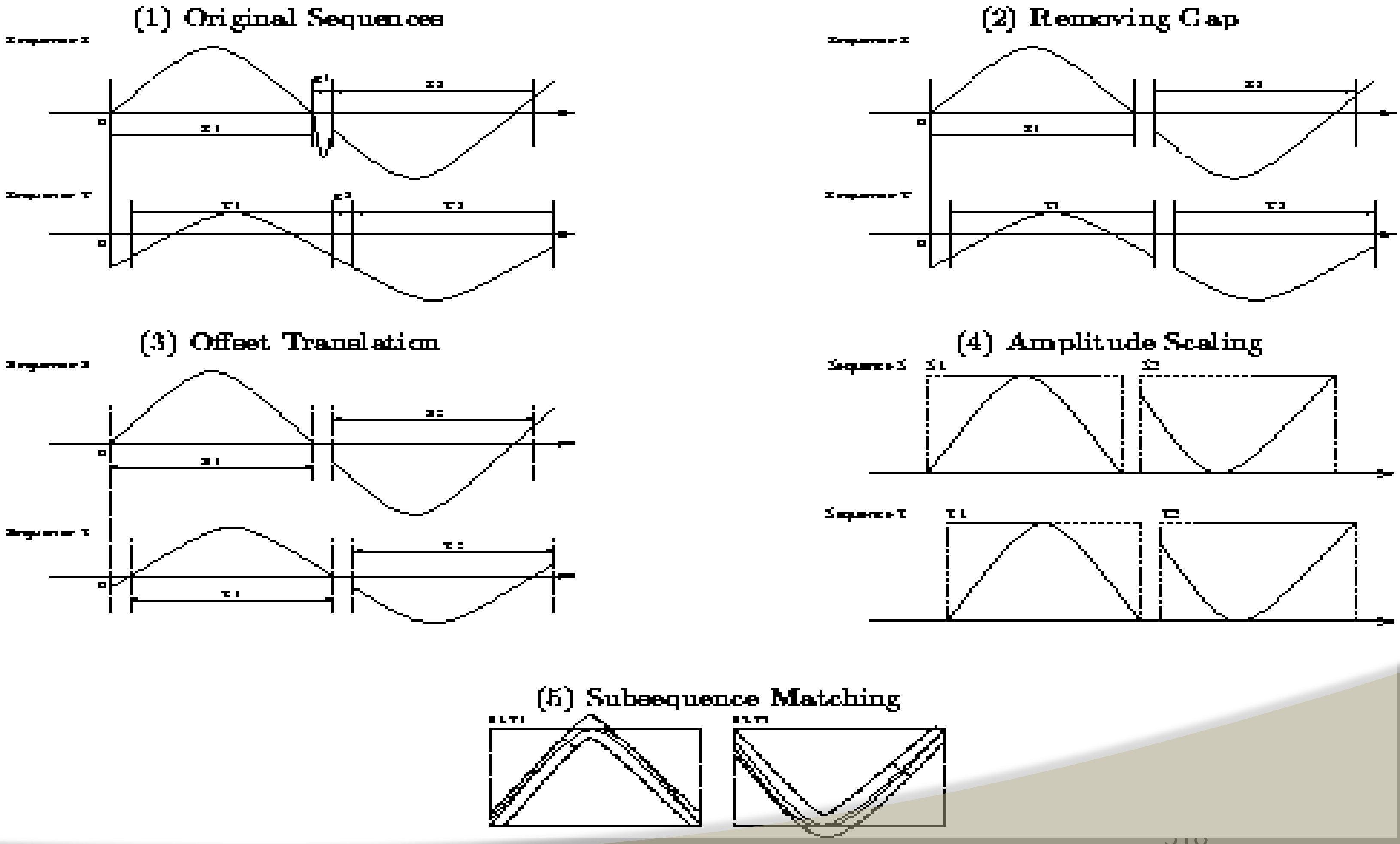
- •Break each sequence into a set of pieces of window with length *w* Extract the features of the subsequence inside the window
- •Map each sequence to a “trail” in the feature space
- •Divide the trail of each sequence into “subtrails” and represent each of them with minimum bounding rectangle
- •Use a multipiece assembly algorithm to search for longer sequence matches



Enhanced similarity search methods

- Allow for gaps within a sequence or differences in offsets or amplitudes
- Normalize sequences with amplitude scaling and offset translation
- Two subsequences are considered similar if one lies within an envelope of width around the other, ignoring outliers
- Two sequences are said to be similar if they have enough non-overlapping time-ordered pairs of similar subsequences Parameters specified by a user or expert: sliding window size, width of an envelope for similarity, maximum gap, and matching fraction

Similar time series analysis

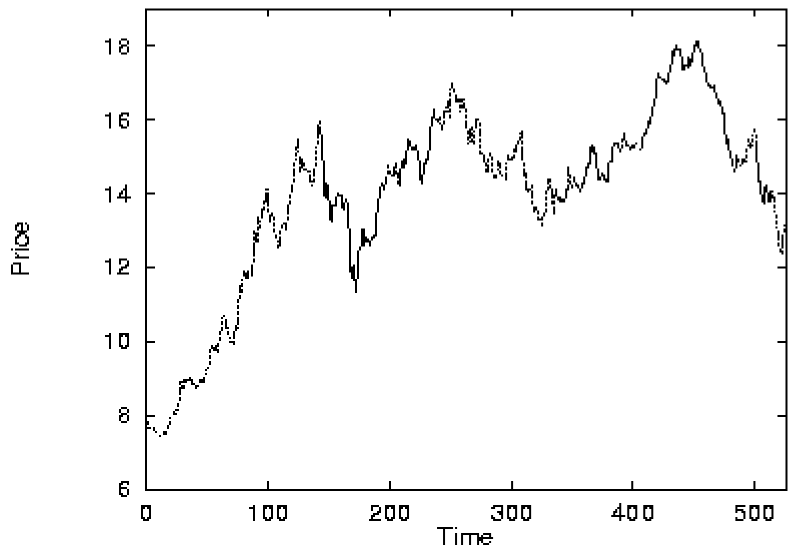


Steps for Performing a Similarity Search

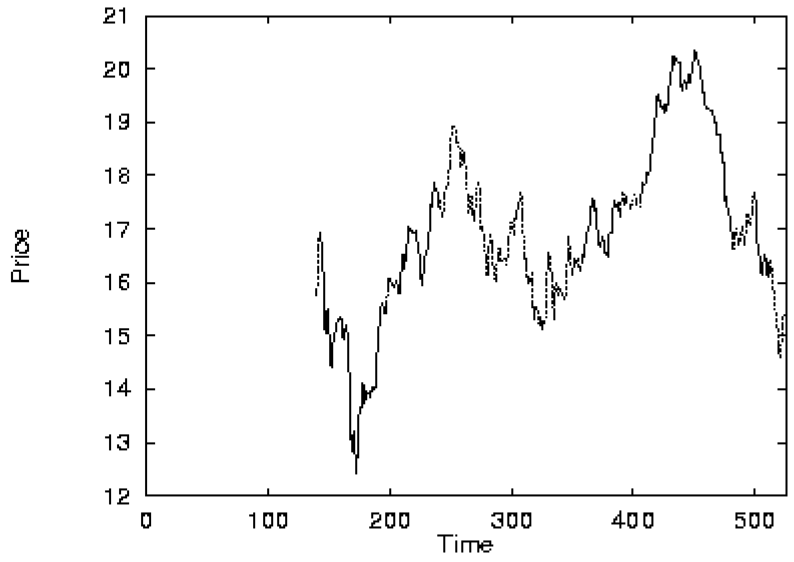
- Atomic matching
  - Find all pairs of gap-free windows of a small length that are
  - similar
- Window stitching
  - Stitch similar windows to form pairs of large similar subsequences allowing gaps between atomic matches
- Subsequence Ordering
  - Linearly order the subsequence matches to determine whether enough similar pieces exist

Similar time series analysis

VanEck International Fund



Fidelity Selective Precious Metal and Mineral Fund



Two similar mutual funds in the different fund group

Query Languages for Time Sequences

- Time-sequence query language
    - Should be able to specify sophisticated queries like
- Find all of the sequences that are similar to some sequence in class *A*, but not similar to any sequence in class *B*
    - Should be able to support various kinds of queries: range queries, all-pair
    - queries, and nearest neighbor queries
- Shape definition language
    - Allows users to define and query the overall shape of time sequences Uses human readable series of sequence transitions or macros Ignores the specific details
        - E.g., the pattern up, Up, UP can be used to describe increasing degrees of rising slopes
        - Macros: spike, valley, etc.

Sequential Pattern Mining

•Mining of frequently occurring patterns related to time or other sequences

•Sequential pattern mining usually concentrate on symbolic

•patterns

Examples

- •Renting “Terminator I”, then “Terminator II”, then “Terminator III” in that order
- •Collection of ordered events within an interval

•Applications

- •Targeted marketing
- •Customer retention
- •Weather prediction

Mining Sequences (cont.)

Customer-sequence

CustId	Video sequence
1	{(C), (H)}
2	{(AB), (C), (DFG)}
3	{(CEG)}
4	{(C), (DG), (H)}
5	{(H)}

Map Large Itemsets

~~Large Itemsets~~	~~MappedID~~
(C)	1
(D)	2
(G)	3
(DG)	4
(H)	5

Sequential patterns with support > 0.25

{(C), (H)}

{(C), (DG)}

Sequential pattern mining: Cases and Parameters

•Duration of a time sequence *T*

•Sequential pattern mining can then be confined to the data within a specified duration

•Ex. Subsequence corresponding to the year of 1999

•Ex. Partitioned sequences, such as every year, or every week after stock crashes, or every two weeks before and after a volcanoeruption

•Event folding window *w*

•If $w = T$, time-insensitive frequent patterns are found

•If $w = 0$ (no event sequence folding), sequential patterns are found where each event occurs at a distinct time instant

•If $0 < w < T$, sequences occurring within the same period *w*are

•folded in the analysis

Sequential pattern mining: Cases and Parameters

- Time interval, *int*, between events in the discovered pattern *int* = 0: no interval gap is allowed, i.e., only strictly consecutive sequences are found

    •Ex. “Find frequent patterns occurring in consecutiveweeks”

•*min_int* ▯*int* ▯ *max_int*: find patterns that are separated by at least *min_int* but at most *max_int*

    •Ex. “If a person rents movie A, it is likely she will rent movie B within 30 days” (*int* 30) *int* = ▯c 0: find patte▯rns carrying an exactinterval

    •Ex. “Every time when Dow Jones drops more than 5%, whatwill

    •happen exactly two days later?” (*int* = 2)

Episodes and Sequential Pattern Mining Methods

- •Other methods for specifying the kinds ofpatterns
  - •Serial episodes: A ⍰ B

    Parallel episodes: A & B
  - •Regular expressions: (A | B)C*(D ⍰E)
- Methods for sequential pattern mining Variations of Apriori-like algorithms, e.g., GSP Database projection-based pattern growth
  - •Similar to the frequent pattern growth without candidate generation

Periodicity Analysis

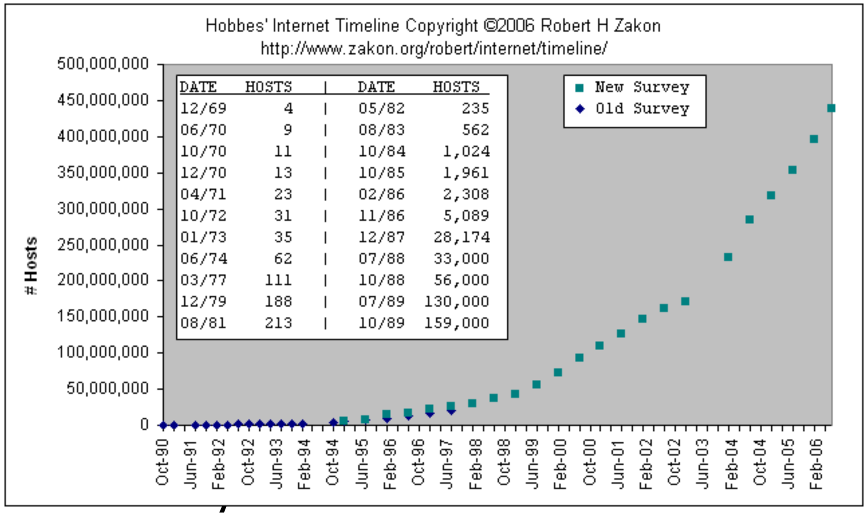
- Periodicity is everywhere: tides, seasons, daily power consumption, etc.
- Full periodicity
  - Every point in time contributes (precisely or approximately) to the periodicity
- Partial periodicity: A more general notion
  - Only some segments contribute to the periodicity
    - Jim reads NY Times 7:00-7:30 am every week day
- Cyclic association rules
  - Associations which form cycles
- Methods
  - Full periodicity: FFT, other statistical analysis methods Partial and cyclic periodicity: Variations of Apriori-like mining methods

Mining the World-Wide Web

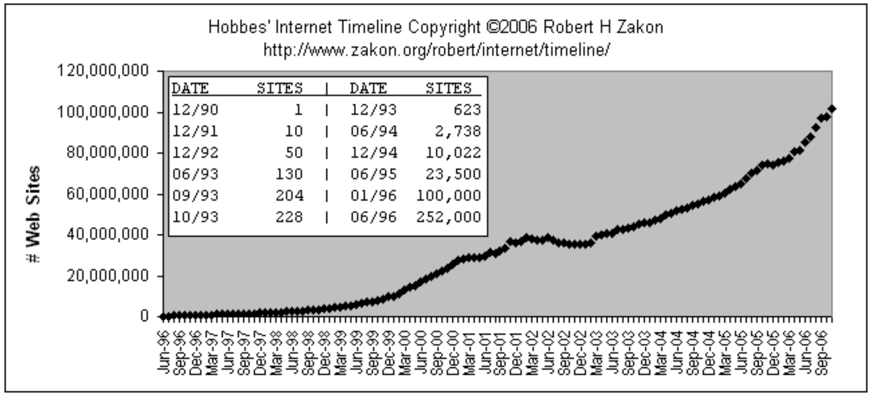
- The WWW is huge, widely distributed, global information service center for
  - Information services: news, advertisements, consumer
  - information, financial management, education, government, e-commerce, etc.
  - Hyper-link information Access and usage information
- WWW provides rich sources for data mining Challenges
  - Too huge for effective data warehousing and data mining
  - Too complex and heterogeneous: no standards and structure

Mining the World-Wide Web

Growing and changing very rapidly



DATE	HOSTS	DATE	HOSTS
12/69	4	05/82	235
06/70	9	08/83	562
10/70	11	10/84	1,024
12/70	13	10/85	1,961
04/71	23	02/86	2,308
10/72	31	11/86	5,089
01/73	35	12/87	28,174
06/74	62	07/88	33,000
03/77	111	10/88	56,000
12/79	188	07/89	130,000
08/81	213	10/89	159,000



DATE	SITES	DATE	SITES
12/90	1	12/93	623
12/91	10	06/94	2,738
12/92	50	12/94	10,022
06/93	130	06/95	23,500
09/93	204	01/96	100,000
10/93	228	06/96	252,000

Broa Only a small portion of the information on the Web is truly relevant or useful

99% of the Web information is useless to 99% of Web users How can we find high-quality Web pages on a specified topic?

Web search engines

•Index-based: search the Web, index Web pages, and build and store huge keyword-based indices.Help locate sets of Web pages containing certain keywords Deficiencies

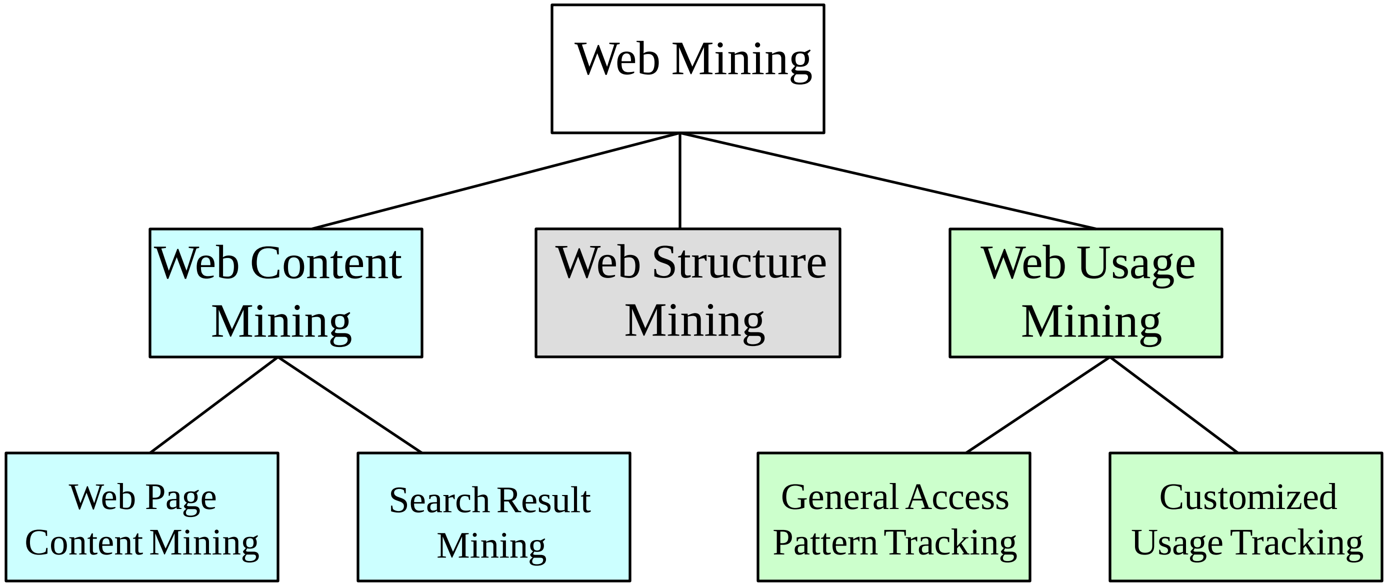
- A topic of any breadth may easily contain hundreds of thousands of documents

•Many documents that are highly relevant to a topic may not contain keywords defining them

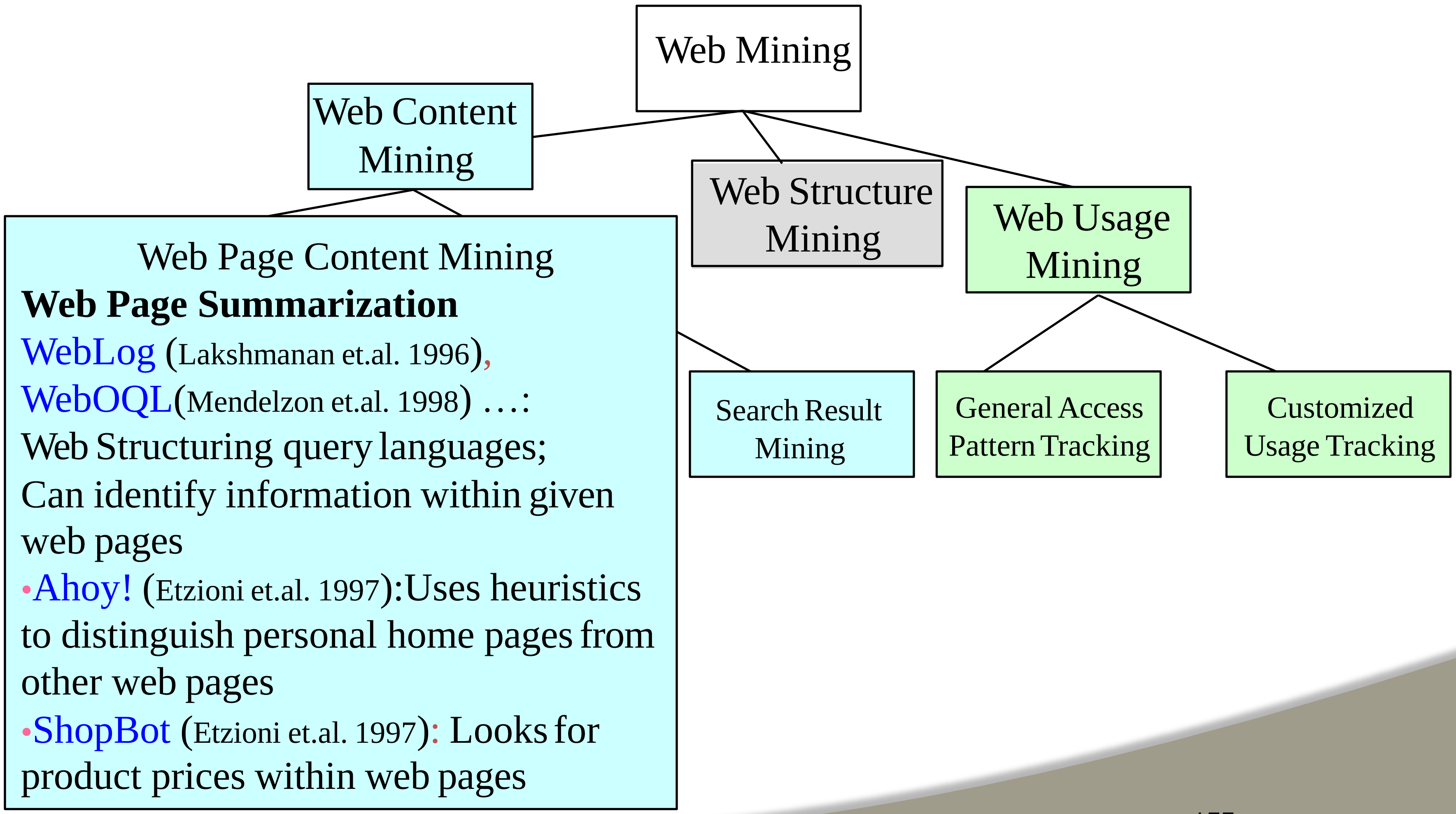
Web Mining: A more challenging task

- Searches for
  - Web access patterns
  - Web structures
  - Regularity and dynamics of Web contents
- Problems
  - The "abundance" problem
  - Limited coverage of the Web: hidden Web sources, majority of data in DBMS
  - Limited query interface based on keyword-oriented search Limited customization to individual users

Web Mining Taxonomy



Mining the World-Wide Web



Web Page Content Mining

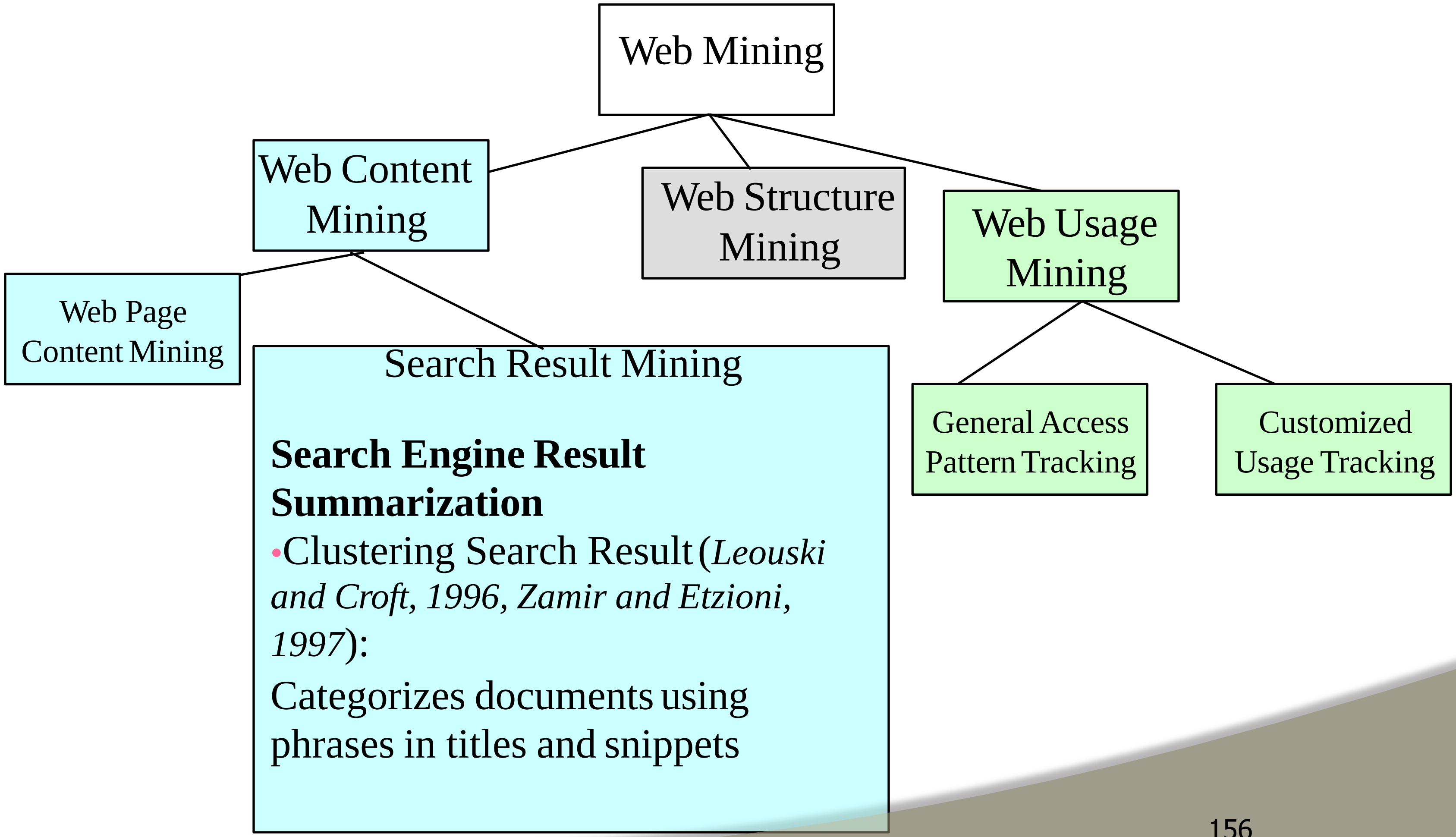
Web Page Summarization

WebLog (Lakshmanan et.al. 1996), WebOQL(Mendelzon et.al. 1998) …: Web Structuring query languages; Can identify information within given web pages

•Ahoy! (Etzioni et.al. 1997):Uses heuristics to distinguish personal home pages from other web pages

•ShopBot (Etzioni et.al. 1997): Looks for product prices within web pages

Mining the World-Wide Web

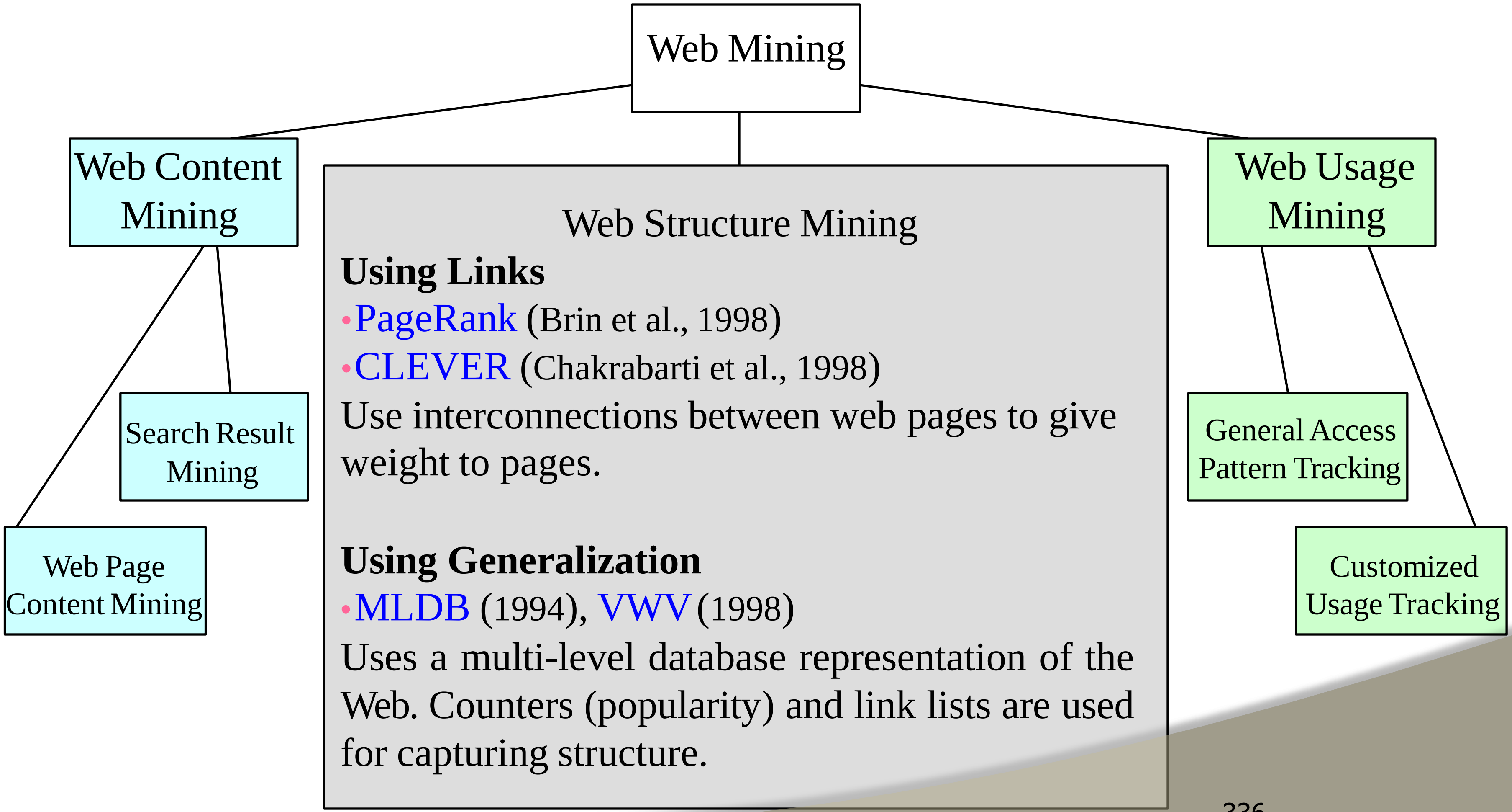


Search Engine Result Summarization

- Clustering Search Result (*Leouski and Croft, 1996, Zamir and Etzioni, 1997*):

Categorizes documents using phrases in titles and snippets

Mining the World-Wide Web



Using Links

- PageRank (Brin et al., 1998)
- CLEVER (Chakrabarti et al., 1998)

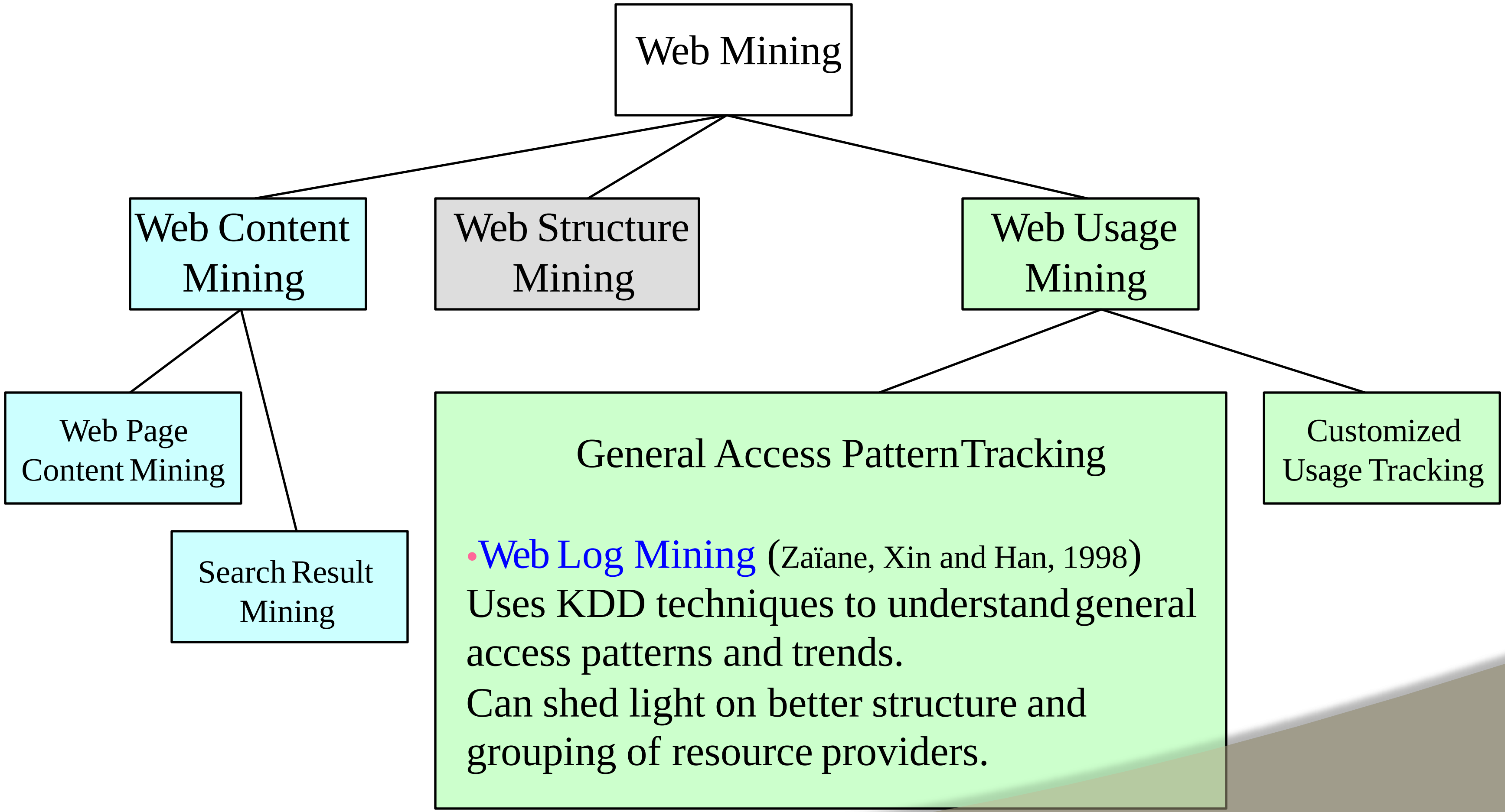
Use interconnections between web pages to give weight to pages.

Using Generalization

- MLDB (1994), VWV (1998)

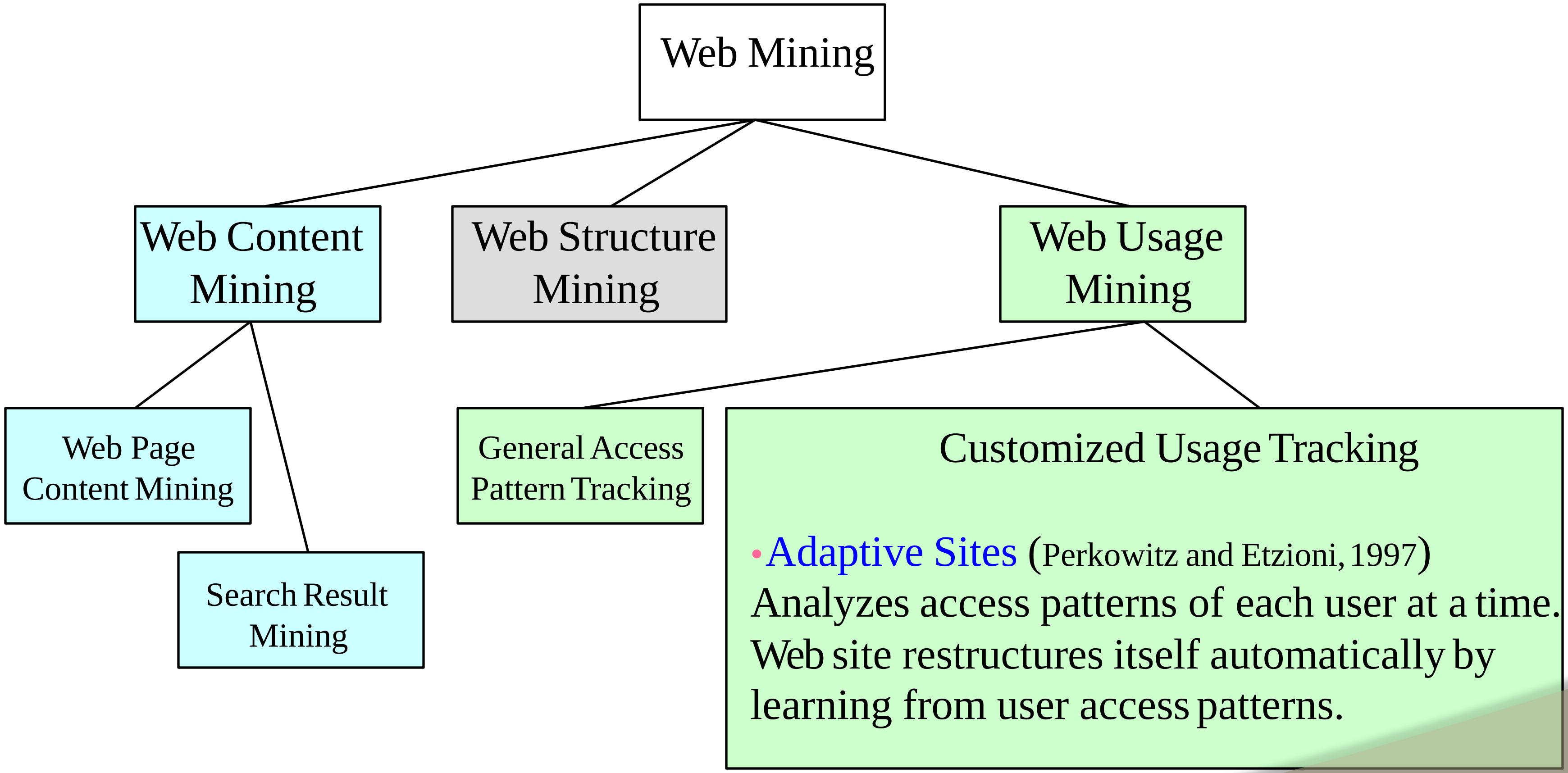
Uses a multi-level database representation of the Web. Counters (popularity) and link lists are used for capturing structure.

Mining the World-Wide Web



• Web Log Mining (Zaïane, Xin and Han, 1998)
Uses KDD techniques to understand general access patterns and trends.
Can shed light on better structure and grouping of resource providers.

Mining the World-Wide Web



- Adaptive Sites (Perkowitz and Etzioni, 1997)

Analyzes access patterns of each user at a time. Web site restructures itself automatically by learning from user access patterns.

Mining the Web's Link Structures

- Finding authoritative Webpages
  - 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

Mining the Web's Link Structures

- •Problems with the Web linkage structure
  - •Not every hyperlink represents an endorsement
    - •Other purposes are for navigation or for paid advertisements
    - •If the majority of hyperlinks are for endorsement, the collective opinion will still dominate
  - •One authority will seldom have its Web page point to its rival authorities in the same field Hub
  - •Set of Web pages that provides collections of links to authorities

Automatic Classification of Web Documents

•Assign a class label to each document from a set of predefined topic categories

•Based on a set of examples of preclassified documents Example

•Use Yahoo!'s taxonomy and its associated documentsas

•training and test sets

•Derive a Web document classification scheme

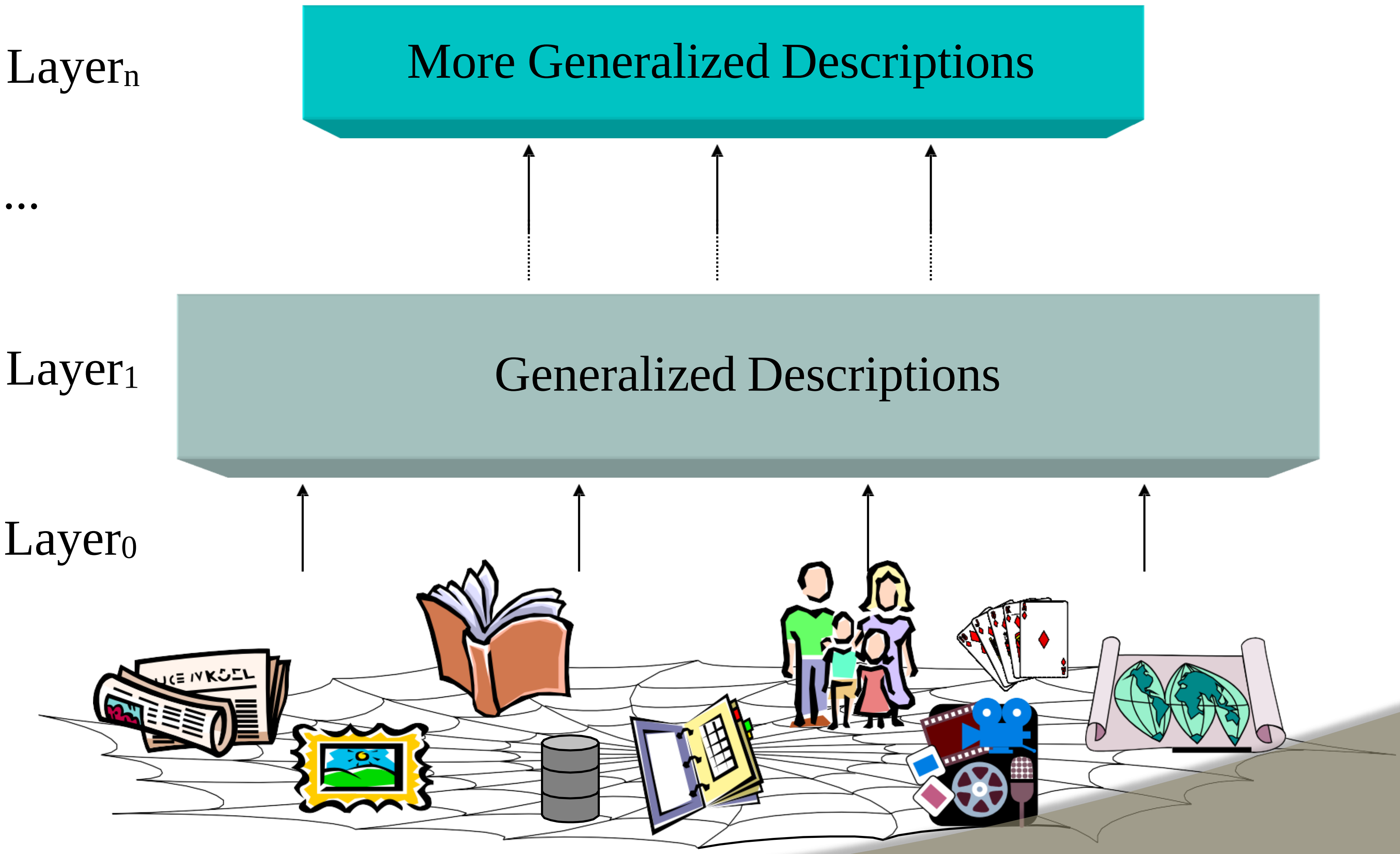
•Use the scheme classify new Web documents by assigning categories from the same taxonomy

•Keyword-based document classification methods Statistical models

Multilayered Web Information Base

- Layer$_0$: the Web itself
- Layer$_1$: the Web page descriptor layer
    - Contains descriptive information for pages on the Web An abstraction of Layer$_0$: substantially smaller but still rich enough to preserve most of the interesting, general information Organized into dozens of semistructured classes
        - *document, person, organization, ads, directory, sales, software, game, stocks,*
        - *library_catalog, geographic_data, scientific_data*, etc.
- Layer$_2$ and up: various Web directory services constructed on top of Layer$_1$
    - provide multidimensional, application-specific services

Multiple Layered Web Architecture



Mining the World-Wide Web

Layer-0: Primitive data

Layer-1: dozen database relations representing types of objects(metadata)

document, organization, person, software, game, map,image,…

•**document**(file_addr, authors, title, publication, publication_date,abstract, language, table_of_contents, category_description, keywords, index, multimedia_attached, num_pages, format, first_paragraphs, size_doc, timestamp, access_frequency, links_out,...)

•**person**(last_name,first_name, home_page_addr, position, picture_attached, phone, e-mail, office_address, education, research_interests, publications, size_of_home_page, timestamp, access_frequency, ...)

•**image**(image_addr, author, title, publication_date, category_description, keywords, size, width, height, duration, format, parent_pages, colour_histogram, Colour_layout, Texture_layout, Movement_vector, localisation_vector, timestamp, access_frequency, ...)

Mining the World-Wide Web

Layer-2: simplification of layer-1

•**doc_brief**(file_addr, authors, title, publication, publication_date, abstract, language, category_description, key_words, major_index, num_pages, format, size_doc,access_frequency, links_out)

•**person_brief** (last_name, first_name, publications,affiliation, e-mail, research_interests, size_home_page, access_frequency)

Layer-3: generalization of layer-2

•**cs_doc**(file_addr, authors, title, publication, publication_date, abstract, language, category_description, keywords, num_pages, form, size_doc, links_out)

•**doc_summary**(affiliation, field, publication_year, count, first_author_list, file_addr_list)

•**doc_author_brief**(file_addr, authors, affiliation, title, publication, pub_date, category_description, keywords, num_pages, format, size_doc, links_out)

•**person_summary**(affiliation, research_interest, year, num_publications, count)

XML and Web Mining

XML can help to extract the correct descriptors

Standardization would greatly facilitate information extraction

<NAME> eXtensible Markup Language**</NAME>**

<RECOM>World-Wide Web Consortium**</RECOM>**

<SINCE>1998**</SINCE>**

<VERSION>1.0**</VERSION>**

<DESC>Meta language that facilitates more meaningful and precise declarations of document content**</DESC>**

<HOW>Define new tags and DTDs**</HOW>**

Potential problem

XML can help solve heterogeneity for vertical applications, but the freedom to define tags can make horizontal applications on the Web more heterogeneous

Benefits of Multi-Layer Meta-Web

- Benefits:
    - Multi-dimensional Web info summary analysis
    - Approximate and intelligent query answering
    - Web high-level query answering (WebSQL, WebML) Web content and structure mining
    - Observing the dynamics/evolution of the Web
- Is it realistic to construct such a meta-Web?
    - Benefits even if it is partially constructed
    - Benefits may justify the cost of tooldevelopment, standardization and partial restructuring

Web Usage Mining

- Mining Web log records to discover user access patterns of Webpages
- Applications
  - Target potential customers for electronic commerce Enhance the quality and delivery of Internetinformation
  - services to the end user
  - Improve Web server system performance Identify potential prime advertisement locations
- Web logs provide rich information about Web dynamics Typical Web log entry includes the URL requested, the IP address from which the request originated, and atimestamp

Techniques for Web usage mining

- Construct multidimensional view on the Weblog database Perform multidimensional OLAP analysis to find the top *N* users, top *N* accessed Web pages, most frequently accessed time periods, etc.
- Perform data mining on Weblog records

  •Find association patterns, sequential patterns, and trends of Web accessing

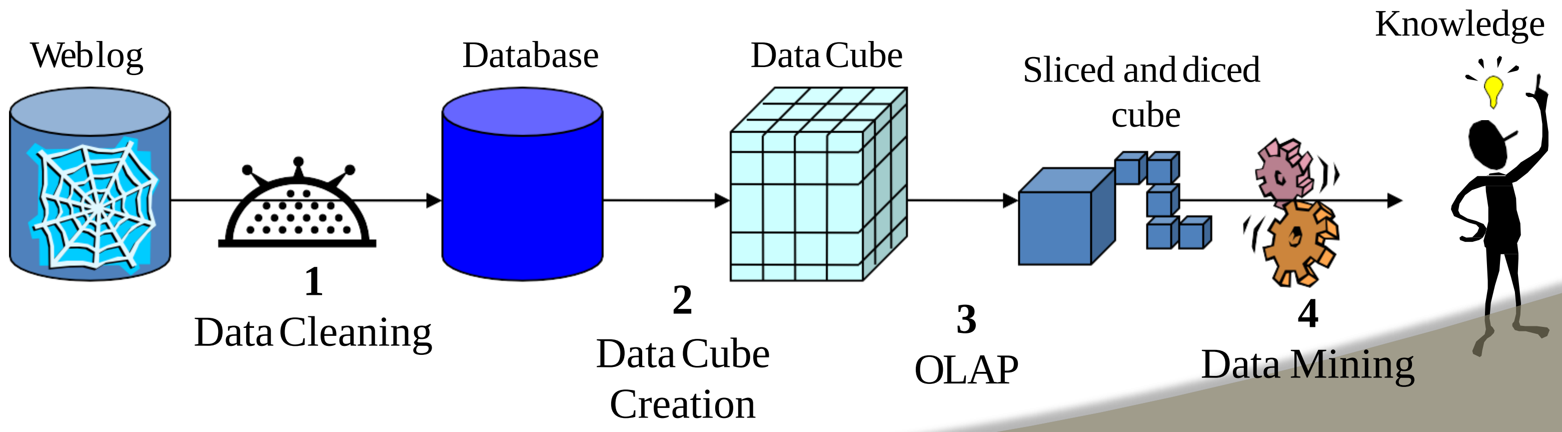
  •May need additional information,e.g., user browsing sequences of the Web pages in the Web server buffer
- Conduct studies to

  •Analyze system performance, improve system design byWeb

  •caching, Web page prefetching, and Web pageswapping

Mining the World-Wide Web

- Design of a Web Log Miner
  - Web log is filtered to generate a relational database
  - A data cube is generated form database
  - OLAP is used to drill-down and roll-up in the cube
  - OLAM is used for mining interesting knowledge



Spatial Association Analysis

Spatial association rule: *A* ⍰ *B* [*s%,c%*]

- A and B are sets of spatial or non-spatial predicates
  - Topological relations: *intersects, overlaps, disjoint,* etc. Spatial orientations: *left_of, west_of, under,* etc.
  - Distance information: *close_to, within_distance,* etc.
  - *s%* is the support and *c%* is the confidence of the rule
- Examples

1) is_a(x, large_town) ^intersect(x,highway) ⍰adjacent_to(x,water) [7%, 85%]

2) What kinds of objects are typically located close to golfcourses?

Progressive Refinement Mining of Spatial Association Rules

- Hierarchy of spatial relationship:
  - *g_close_to*: *near_by*, *touch*, *intersect*, *contain*, etc.
  - First search for rough relationship and then refine it
- Two-step mining of spatial association:
  - Step 1: Rough spatial computation (as afilter)
    - Using MBR or R-tree for rough estimation
  - Step2: Detailed spatial algorithm (as refinement)
    - Apply only to those objects which have passed the rough spatial association test (no less than *min_support*)

Spatial Classification

- Analyze spatial objects to derive classification schemes, such as decision trees, in relevance to certain spatial properties (district, highway, river,etc.)
    - Classifying medium-size families according to income, region, andinfant
    - mortality rates
    - Mining for volcanoes on Venus
- Employ most of the methods in classification
    - Decision-tree classification, Naïve-Bayesian classifier + boosting, neural network, genetic programming, etc.
    - Association-based multi-dimensional classification - Example: classifying house value based on proximity to lakes, highways, mountains, etc.

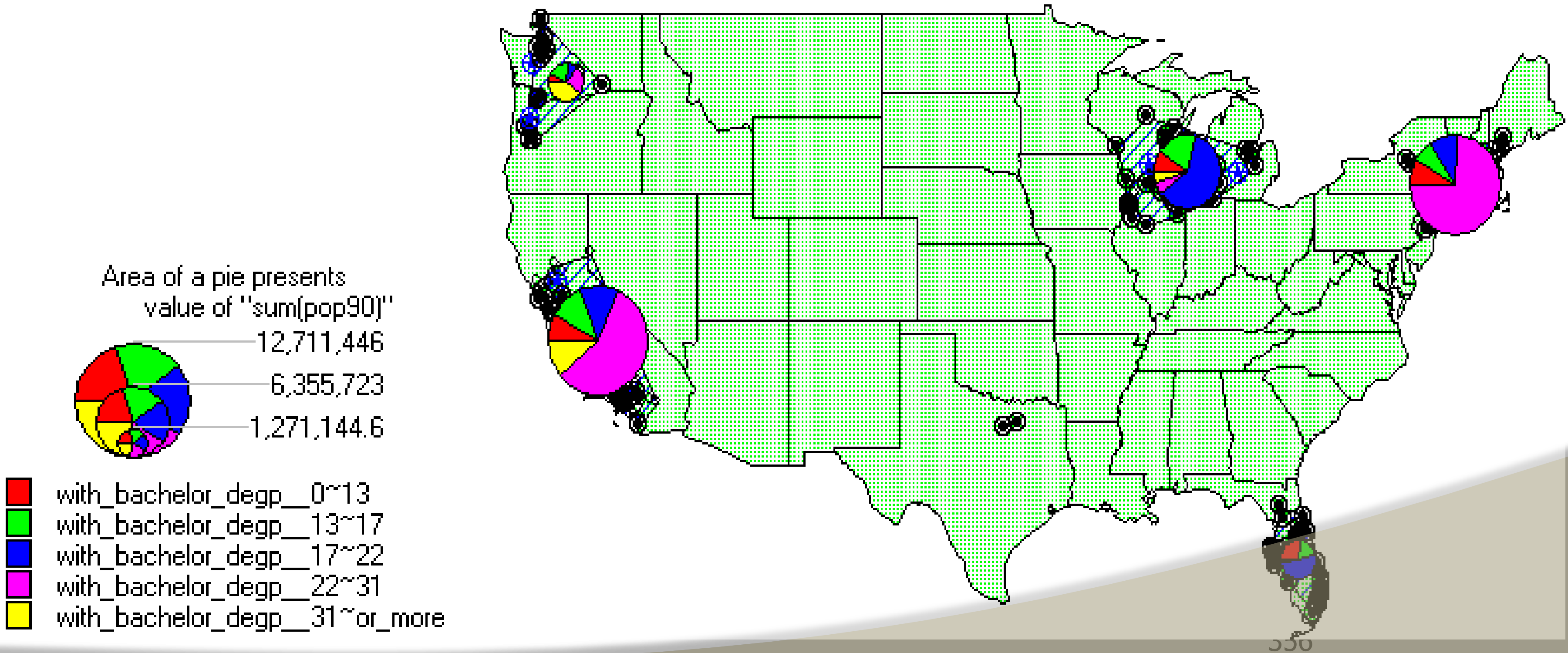
Spatial Trend Analysis

- Function
  - Detect changes and trends along a spatial dimension
  - Study the trend of non-spatial or spatial data changing with space
- Application examples
  - Observe the trend of changes of the climate or vegetation with increasing distance from an ocean
  - Crime rate or unemployment rate change with regard to city geo- distribution

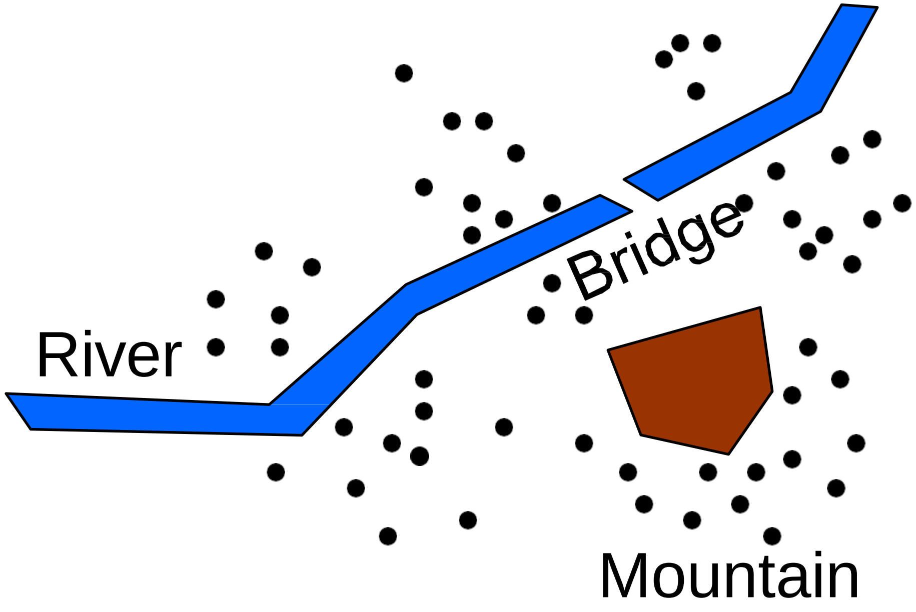
Spatial Cluster Analysis

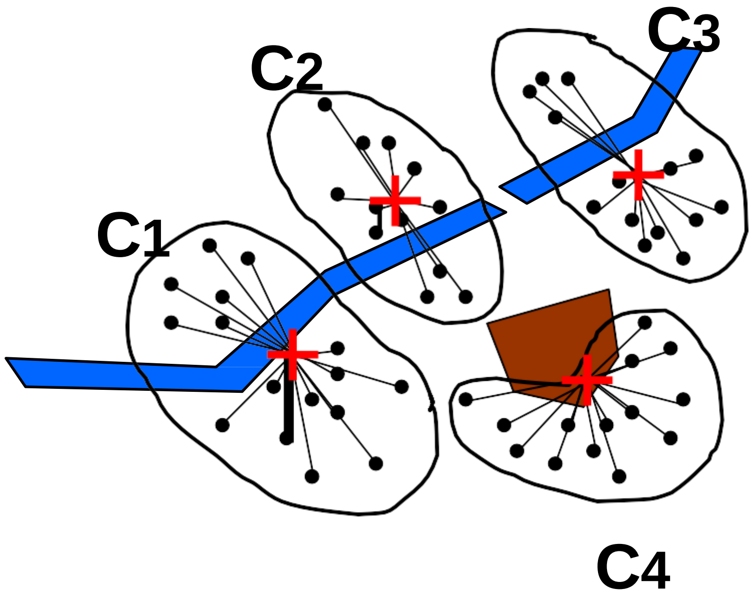
Mining clusters—k-means, k-medoids, hierarchical, density-based, etc.

Analysis of distinct features of the clusters


Area of a pie presents
value of "sum(pop90)"
12,711,446
6,355,723
1,271,144.6
🟥 with_bachelor_degp__0~13
🟩 with_bachelor_degp__13~17
🟦 with_bachelor_degp__17~22
🟪 with_bachelor_degp__22~31
🟨 with_bachelor_degp__31~or_more

Constraint-Based Clustering: Planning ATM Locations





Spatial data with obstacles

Clustering *without* taking obstacles into consideration

Similarity Search in Multimedia Data

- Description-based retrieval systems
    - Build indices and perform object retrieval based on image descriptions, such as keywords, captions, size, and time of creation
    - Labor-intensive if performed manually
    - Results are typically of poor quality if automated
- Content-based retrieval systems
    - Support retrieval based on the image content, such as color histogram, texture, shape, objects, and wavelet transforms

Queries in Content-Based Retrieval Systems

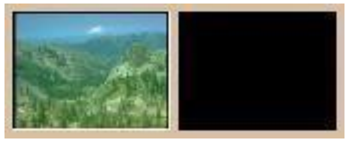
- Image sample-based queries
  - Find all of the images that are similar to the given image sample
  - Compare the feature vector (signature) extracted from the sample with the feature vectors of images that have already been extracted and indexed in the image database
- Image feature specification queries
  - Specify or sketch image features like color, texture, or shape, which are translated into a feature vector
  - Match the feature vector with the feature vectors of the images in the database

Mining Multimedia Databases

Refining or combining searches

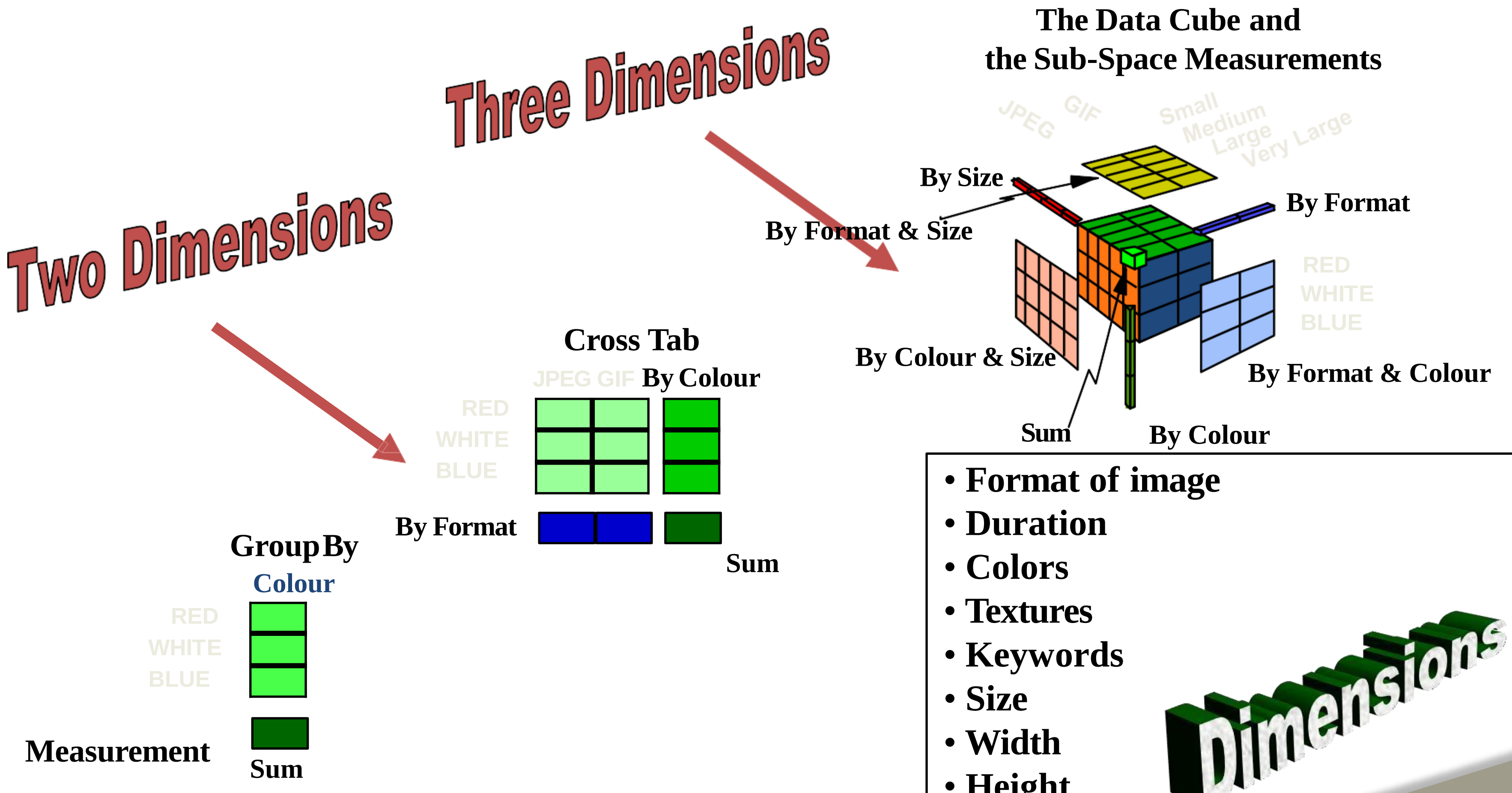
Search for “airplane in bluesky” (top layout grid is blue and keyword = “airplane”)

Search for “blue sky” (top layout grid is blue)



Search for “blue sky and green meadows” (top layout grid is blue and bottom is green)

Mining Multimedia Databases



- Format of image
- Duration
- Colors
- Textures
- Keywords
- Size
- Width
- Height
- Internet domain of image
- Internet domain of parent pages
- Image popularity