



# Pattern Recognition

Pattern recognition is:

1. The name of the journal of the Pattern Recognition Society.
2. A research area in which patterns in data are found, recognized, discovered, ...whatever.
3. A catchall phrase that includes
  - classification
  - clustering
  - data mining
  - .....



# Two Schools of Thought

## 1. Statistical Pattern Recognition

The data is reduced to vectors of numbers and statistical techniques are used for the tasks to be performed.

## 2. Structural Pattern Recognition

The data is converted to a discrete structure (such as a grammar or a graph) and the techniques are related to computer science subjects (such as parsing and graph matching).



## In this course

1. How should objects to be classified be represented?
2. What algorithms can be used for recognition (or matching)?
3. How should learning (training) be done?



# Classification in Statistical PR

- A **class** is a set of objects having some important properties in common
- A **feature extractor** is a program that inputs the data (image) and extracts features that can be used in classification.
- A **classifier** is a program that inputs the feature vector and assigns it to one of a set of designated classes or to the “reject” class.

With what kinds of classes do you work?



# Feature Vector Representation

- ◆  $X=[x_1, x_2, \dots, x_n]$ , each  $x_j$  a real number
- ◆  $x_j$  may be an object measurement
- ◆  $x_j$  may be count of object parts
- ◆ Example: object rep. [#holes, #strokes, moments, ...]

```
00000000010000000000      00000000011110000000
00000000011000000000      00000001100001100000
00000000010100000000      00000011000000110000
0000000010000100000000    00001100000000011000
0000001000001000000000    00001000000000001000
0000010000000100000000    00001100000000010000
0000100000000010000000    00000111000000100000
0000110011111111000000    00000011100111100000
0000111111000001000000    00000001111000000000
0001100000000001100000    00000011000111000000
0001000000000000110000    00001100000000110000
0011000000000000010000    00011000000000110000
0011000000000000011000    00110000000000010000
0010000000000000001000    00100000000000001100
0010000000000000000100    00010000000000110000
0110000000000000001000    00011000000000100000
0100000000000000000000    00001000000001100000
0000000000000000000000    00000011111100000000
```



# Possible features for char rec.

| (class)<br>character | area   | height | width | number<br>#holes | number<br>#strokes | (cx,cy)<br>center | best<br>axis | least<br>inertia |
|----------------------|--------|--------|-------|------------------|--------------------|-------------------|--------------|------------------|
| 'A'                  | medium | high   | 3/4   | 1                | 3                  | 1/2,2/3           | 90           | medium           |
| 'B'                  | medium | high   | 3/4   | 2                | 1                  | 1/3,1/2           | 90           | large            |
| '8'                  | medium | high   | 2/3   | 2                | 0                  | 1/2,1/2           | 90           | medium           |
| '0'                  | medium | high   | 2/3   | 1                | 0                  | 1/2,1/2           | 90           | large            |
| '1'                  | low    | high   | 1/4   | 0                | 1                  | 1/2,1/2           | 90           | low              |
| '4'                  | high   | high   | 1     | 0                | 4                  | 1/2,2/3           | 90           | large            |
| 'I'                  | high   | high   | 3/4   | 0                | 2                  | 1/2,1/2           | ?            | large            |
| '*'                  | medium | low    | 1/2   | 0                | 0                  | 1/2,1/2           | ?            | large            |
| '_'                  | low    | low    | 2/3   | 0                | 1                  | 1/2,1/2           | 0            | low              |
| '/'                  | low    | high   | 2/3   | 0                | 1                  | 1/2,1/2           | 60           | low              |



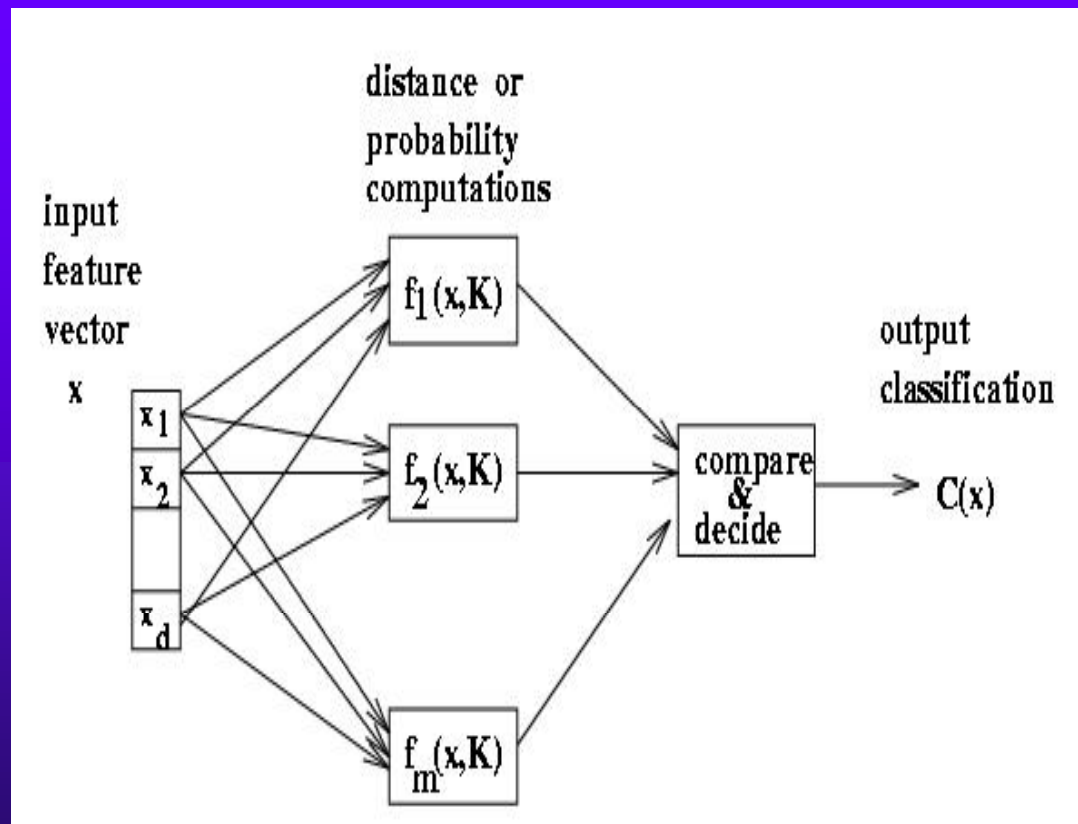
# Some Terminology

- ◆ Classes: set of  $m$  known categories of objects
  - (a) might have a known description for each
  - (b) might have a set of samples for each
- ◆ Reject Class:
  - a generic class for objects not in any of the designated known classes
- ◆ Classifier:
  - Assigns object to a class based on features



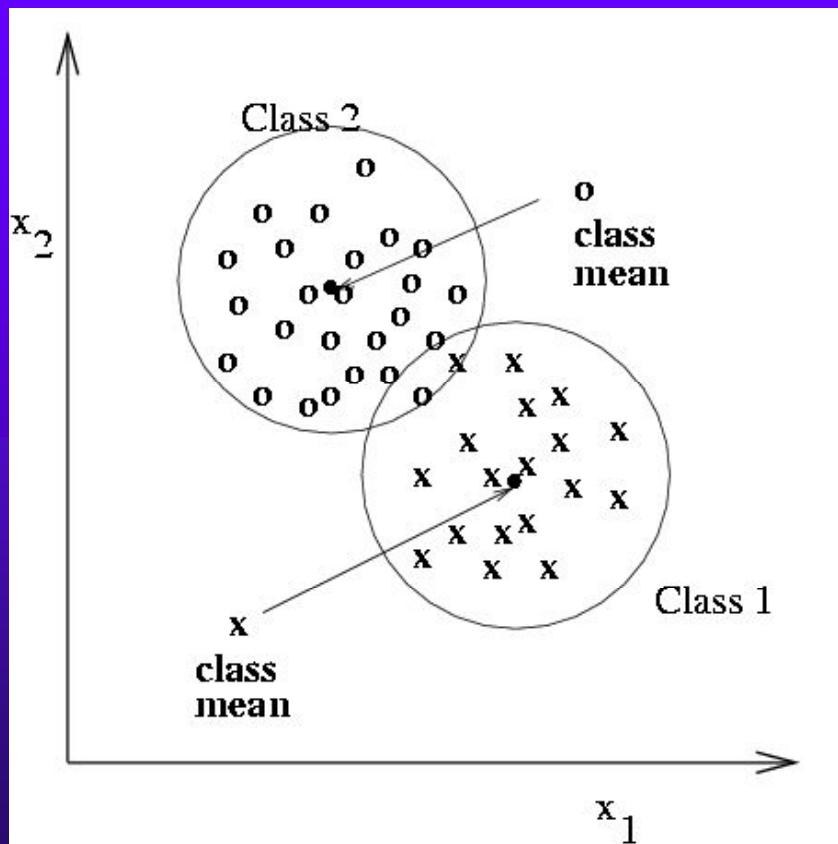
# Discriminant functions

- ◆ Functions  $f(x, K)$  perform some computation on feature vector  $x$
- ◆ Knowledge  $K$  from training or programming is used
- ◆ Final stage determines class



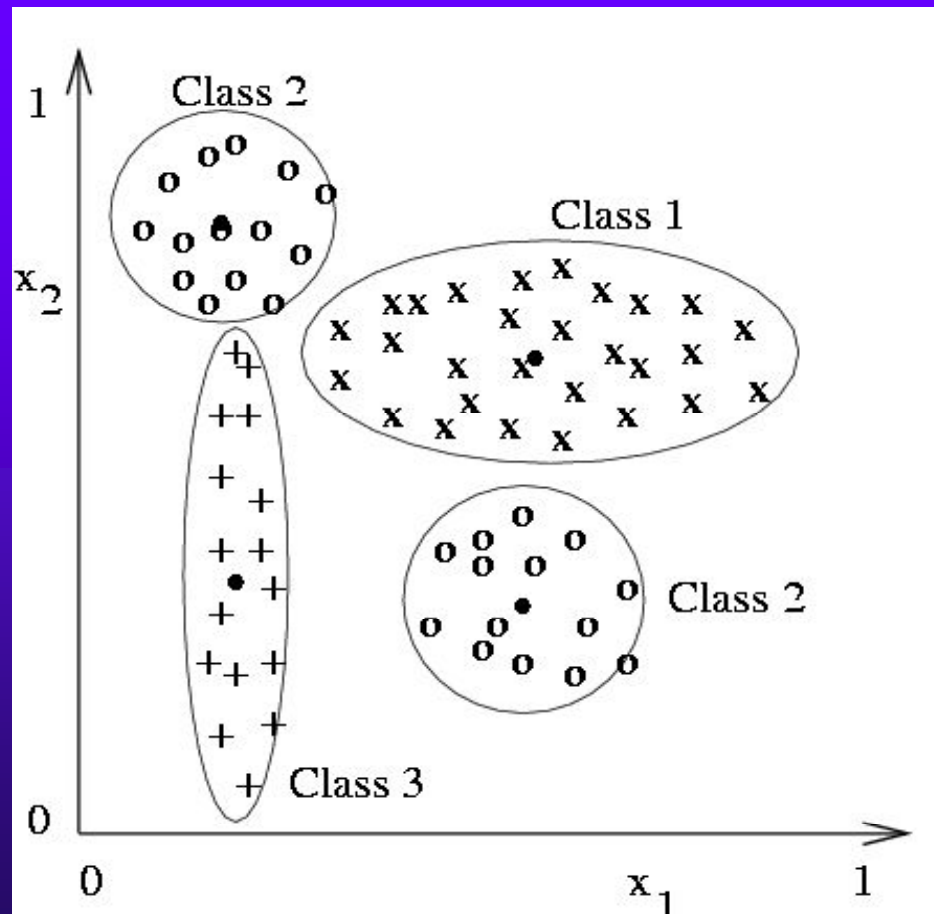


# Classification using nearest class mean



- ◆ Compute the Euclidean distance between feature vector  $X$  and the mean of each class.
- ◆ Choose closest class, if close enough (reject otherwise)

# Nearest mean might yield poor results with complex structure



- ◆ Class 2 has two modes; **where is its mean?**
- ◆ But if modes are detected, two subclass mean vectors can be used



# Scaling coordinates by std dev

We can compute a modified distance from feature vector  $\mathbf{x}$  to class mean vector  $\mathbf{x}_c$  by scaling by the spread, or *standard deviation*,  $\sigma_i$  of class  $c$  along each dimension  $i$ .

**scaled Euclidean distance** from  $\mathbf{x}$  to class mean  $\mathbf{x}_c$  :

$$\| \mathbf{x} - \mathbf{x}_c \| = \sqrt{\sum_{i=1,d} ((\mathbf{x}[i] - \mathbf{x}_c[i]) / \sigma_i)^2}$$

In the previous 3 class problem, an observed  $X$  near the top of the Class 3 distribution will scale to be closer to the mean of Class 3 than to the mean of Class 2. Without scaling,  $X$  would be closer to the mean of Class 2.



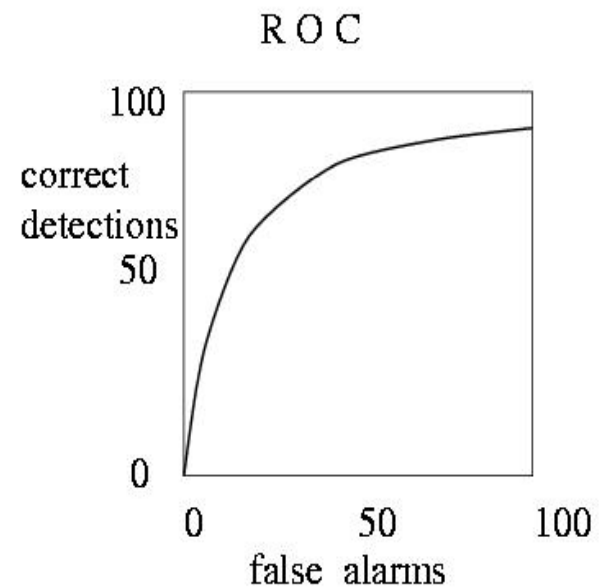
# Nearest Neighbor Classification

- Keep all the training samples in some efficient look-up structure.
- Find the nearest neighbor of the feature vector to be classified and assign the class of the neighbor.
- Can be extended to  $K$  nearest neighbors.

# Receiver Operating Curve ROC



- ◆ Plots correct detection rate versus false alarm rate
- ◆ Generally, false alarms go up with attempts to detect higher percentages of known objects



| actual input object | decision    | error type?                  |
|---------------------|-------------|------------------------------|
| frack               | frack       | correct alarm (no error)     |
| not a frack         | frack       | false alarm (error)          |
| frack               | not a frack | false dismissal (error)      |
| not a frack         | not a frack | correct dismissal (no error) |



# Confusion matrix shows empirical performance



class j output by the pattern recognition system

|                         |     | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | 'R' |
|-------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| true<br>object<br>class | '0' | 97  | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 1   |
|                         | '1' | 0   | 98  | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 0   | 0   |
|                         | '2' | 0   | 0   | 96  | 1   | 0   | 1   | 0   | 1   | 0   | 0   | 1   |
|                         | '3' | 0   | 0   | 2   | 95  | 0   | 1   | 0   | 0   | 1   | 0   | 1   |
|                         | '4' | 0   | 0   | 0   | 0   | 98  | 0   | 0   | 0   | 0   | 0   | 2   |
|                         | '5' | 0   | 0   | 0   | 1   | 0   | 97  | 0   | 0   | 0   | 0   | 2   |
|                         | '6' | 1   | 0   | 0   | 0   | 0   | 1   | 98  | 0   | 0   | 0   | 0   |
|                         | '7' | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 98  | 0   | 0   | 1   |
|                         | '8' | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 96  | 1   | 1   |
|                         | '9' | 1   | 0   | 0   | 0   | 3   | 0   | 0   | 0   | 0   | 1   | 95  |

confusion may be unavoidable between some classes  
for example, between 9's and 4's, or between u's and j's  
for handprinted characters



# Bayesian decision-making

- classify into class  $\omega_i$  that is most likely based on observations  $\mathbf{X}$
- In order to compute the likelihoods given the measurement  $\mathbf{X}$ , the following distributions are needed.

**class conditional distribution :**  $p(x|\omega_i)$  for each class  $\omega_i$  (1)

**a priori probability :**  $P(\omega_i)$  for each class  $\omega_i$  (2)

**unconditional distribution :**  $p(x)$  (3)

- use Bayes rule if all of the classes  $\omega_i$  are disjoint

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} = \frac{p(x|\omega_i)P(\omega_i)}{\sum_{i=1,m} p(x|\omega_i)P(\omega_i)} \quad (4)$$



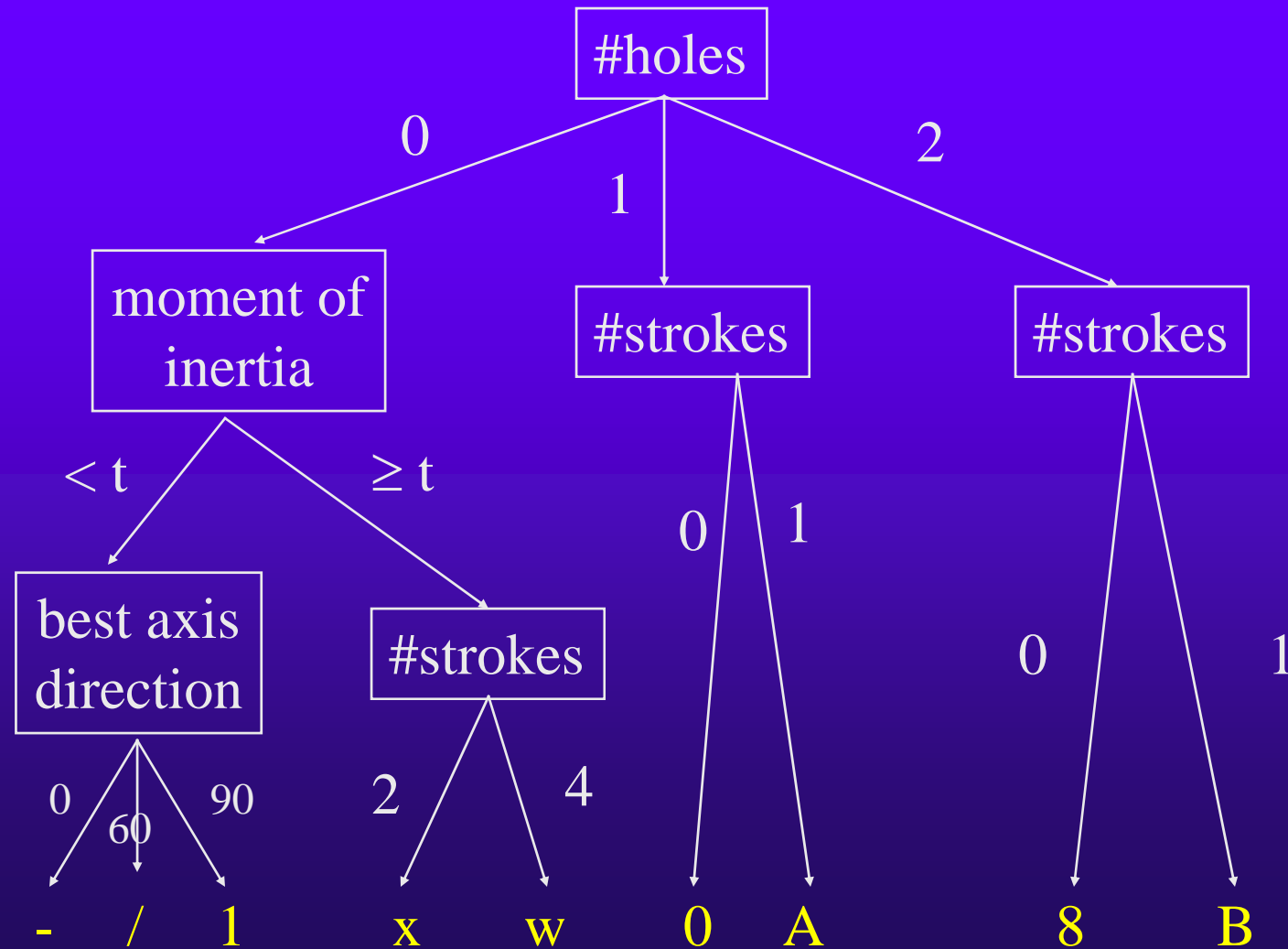


# Classifiers often used in CV

- Decision Tree Classifiers
- Artificial Neural Net Classifiers
- Bayesian Classifiers and Bayesian Networks (Graphical Models)
- Support Vector Machines



# Decision Trees





# Decision Tree Characteristics

## 1. Training

How do you construct one from training data?

Entropy-based Methods

## 2. Strengths

Easy to Understand

## 3. Weaknesses

Overtraining

# Entropy-Based Automatic Decision Tree Construction



Training Set S

$x_1 = (f_{11}, f_{12}, \dots, f_{1m})$

$x_2 = (f_{21}, f_{22}, \dots, f_{2m})$

⋮

⋮

$x_n = (f_{n1}, f_{n2}, \dots, f_{nm})$

Node 1

What feature  
should be used?



What values?

Quinlan suggested **information gain** in his ID3 system and later the **gain ratio**, both based on **entropy**.



# Entropy

Given a set of training vectors  $S$ , if there are  $c$  classes,

$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

Where  $p_i$  is the proportion of category  $i$  examples in  $S$ .

If all examples belong to the same category, the entropy is 0.

If the examples are equally mixed ( $1/c$  examples of each class), the entropy is a maximum at 1.0.

e.g. for  $c=2$ ,  $-.5 \log_2 .5 - .5 \log_2 .5 = -.5(-1) - .5(-1) = 1$



# Information Gain

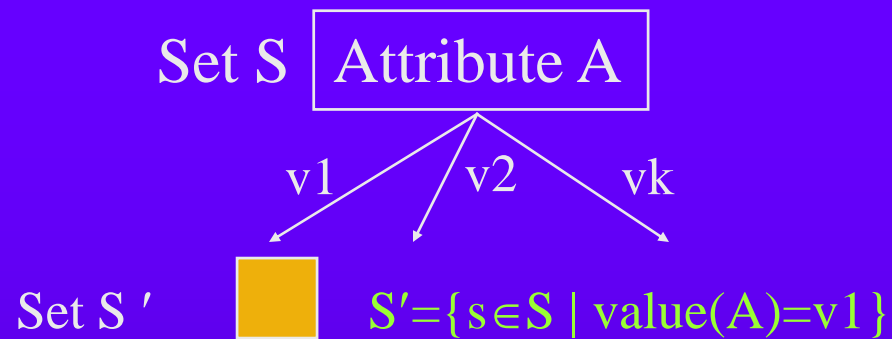
The **information gain** of an attribute  $A$  is the expected reduction in entropy caused by partitioning on this attribute.

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

where  $S_v$  is the subset of  $S$  for which attribute  $A$  has value  $v$ .

Choose the attribute  $A$  that gives the maximum information gain.

# Information Gain (cont)



repeat  
recursively

Information gain has the disadvantage that it prefers attributes with large number of values that split the data into small, pure subsets.





# Gain Ratio

Gain ratio is an alternative metric from Quinlan's 1986 paper and used in the popular C4.5 package (free!).

$$\text{GainRatio}(S,A) = \frac{\text{Gain}(S,a)}{\text{SplitInfo}(S,A)}$$

$$\text{SplitInfo}(S,A) = \sum_{i=1}^{n_i} \frac{|S_i|}{|S|} \log_2 \left[ \frac{|S_i|}{|S|} \right]$$

where  $S_i$  is the subset of  $S$  in which attribute  $A$  has its  $i$ th value.

SplitInfo measures the amount of information provided by an attribute that is not specific to the category.



# Information Content

## Note:

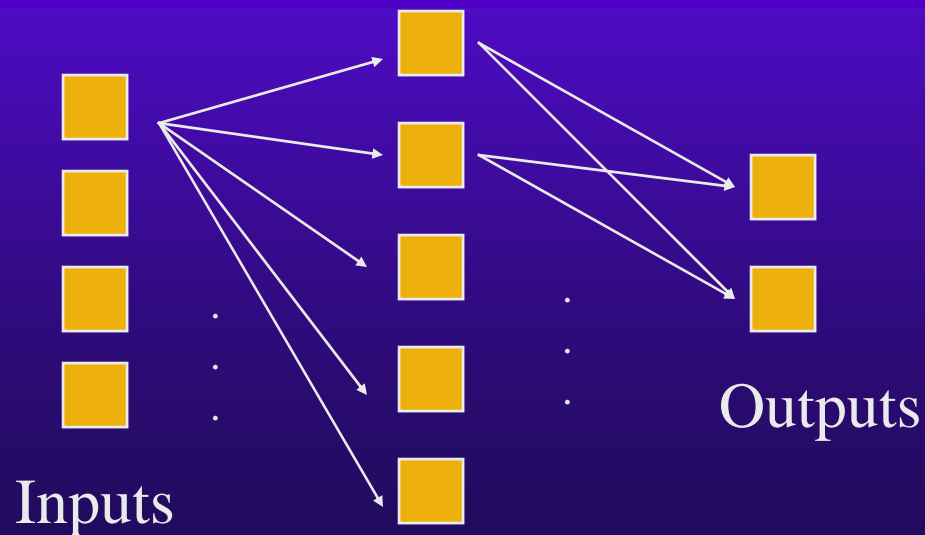
A related method of decision tree construction using a measure called **Information Content** is given in the text, with full numeric example of its use.



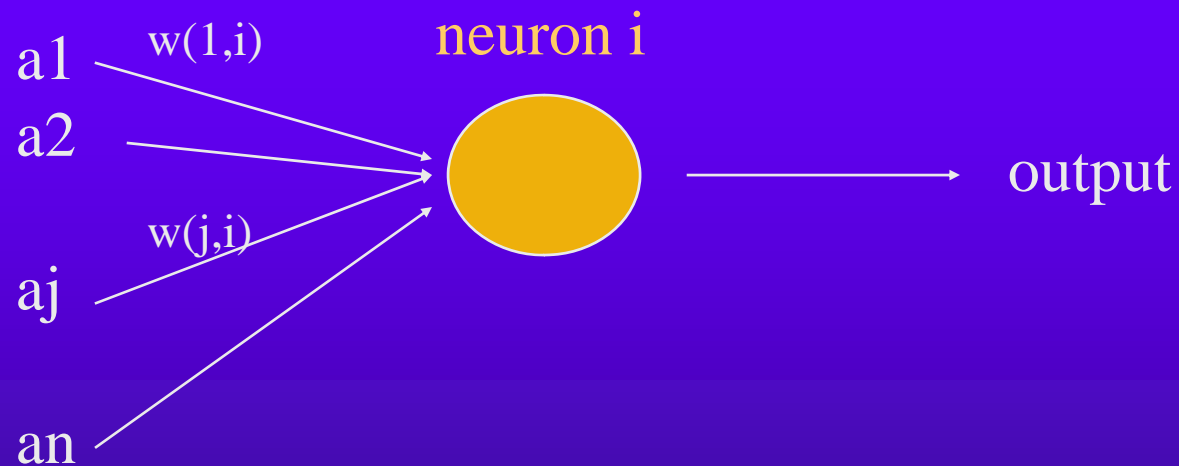
# Artificial Neural Nets

Artificial Neural Nets (ANNs) are networks of artificial neuron nodes, each of which computes a simple function.

An ANN has an input layer, an output layer, and “hidden” layers of nodes.



# Node Functions



$$\text{output} = g \left( \sum a_j * w(j,i) \right)$$

Function  $g$  is commonly a step function, sign function, or sigmoid function (see text).



# Neural Net Learning

That's beyond the scope of this text; only simple feed-forward learning is covered.

The most common method is called **back propagation**.

We've been using a free package called NevProp.

What do you use?



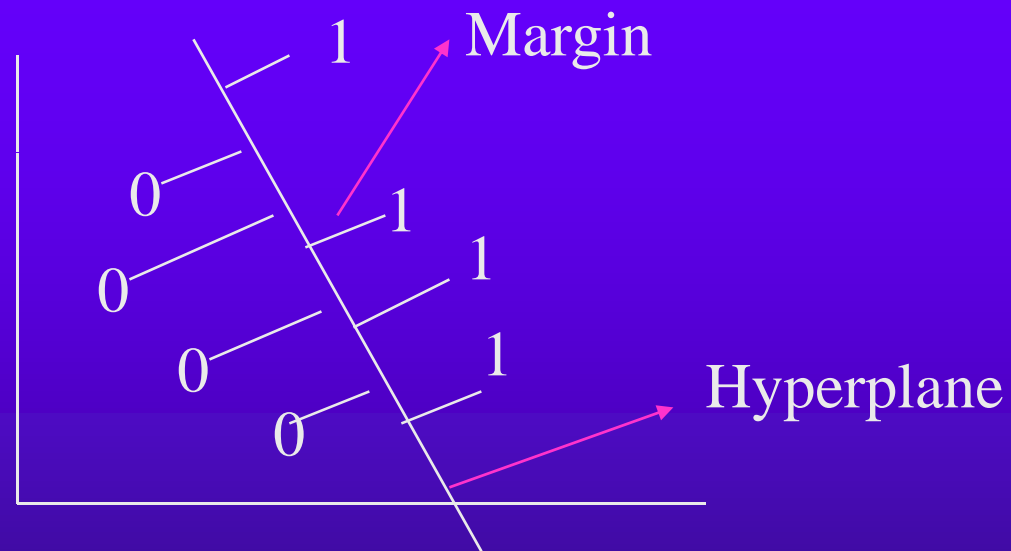
# Support Vector Machines (SVM)

Support vector machines are learning algorithms that try to find a hyperplane that separates the differently classified data the most. They are based on two key ideas:

- Maximum margin hyperplanes
- A kernel ‘trick’.



# Maximal Margin

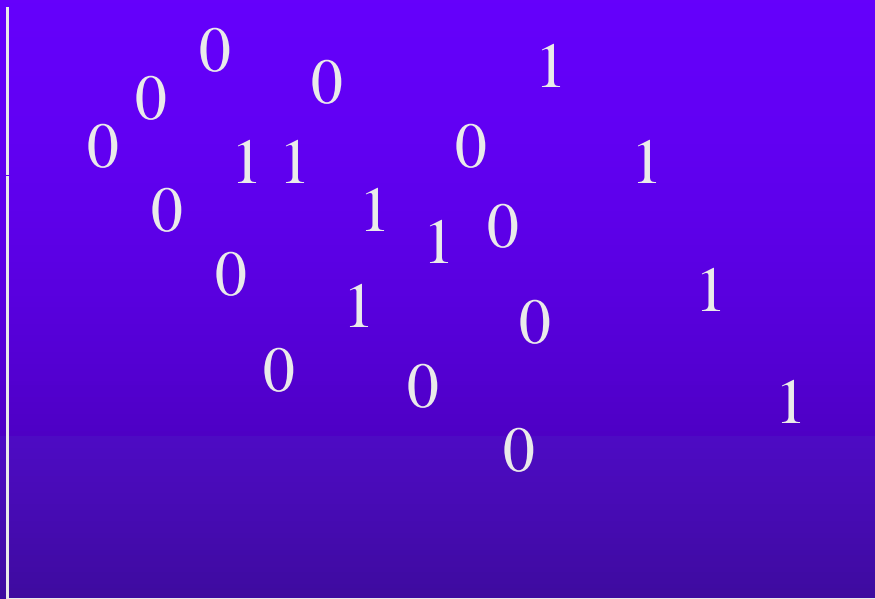


Find the hyperplane with maximal margin for all the points. This originates an optimization problem Which has a unique solution (convex problem).





# Non-separable data



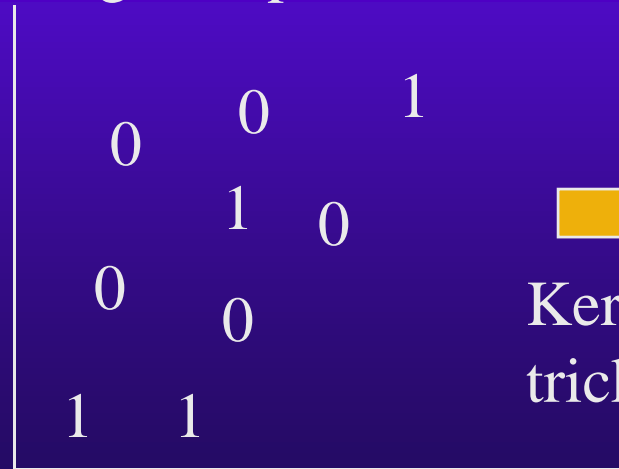
What can be done if data cannot be separated with a hyperplane?



# The kernel trick

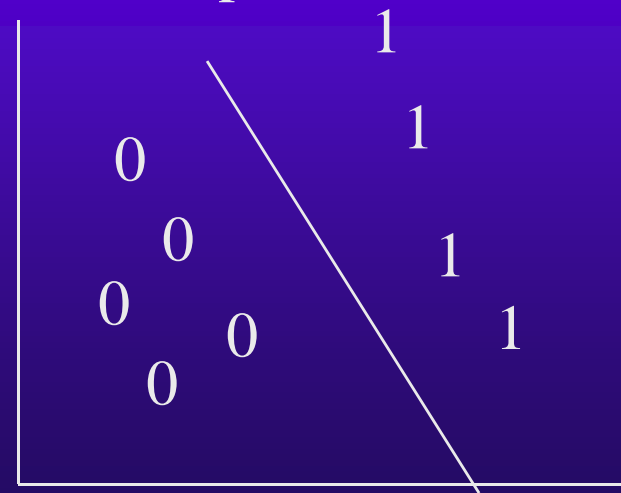
The SVM algorithm implicitly maps the original data to a feature space of possibly infinite dimension in which data (which is not separable in the original space) becomes separable in the feature space.

Original space  $R^k$



Kernel  
trick

Feature space  $R^n$





# Our Current Application

- Sal Ruiz is using support vector machines in his work on 3D object recognition.
- He is training classifiers on data representing deformations of a 3D model of a class of objects.
- The classifiers are starting to learn what kinds of surface patches are related to key parts of the model (ie. A snowman's face)

# Snowman with Patches

