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.

What kinds of object classes?

Feature Vector Representation

- X=[x1, x2, … , xn], each xj a real number
- xj may be an object measurement
- xj may be count of object parts
- Example: object rep. [#holes, #strokes, moments, …]

```
00000000010000000000    00000000011110000000
00000000110000000000    00000001100001100000
00000000101000000000    00000011000000110000
00000001000010000000    00001100000000011000
00000010000010000000    00001000000000001000
00000100000001000000    00001100000000010000
00001000000000100000    00000111000000100000
00001100111111110000    00000011100111100000
00001111110000010000    00000000111100000000
00011000000000011000    00000011000111000000
00010000000000001100    00001100000000110000
00110000000000000100    00011000000000011000
00110000000000000110    00110000000000001000
00100000000000000010    00100000000000001100
00100000000000000010    00010000000000011000
01100000000000000010    00011000000000010000
01000000000000000000    00001000000000110000
00000000000000000000    00000011111110000000
```

Possible features for char rec.

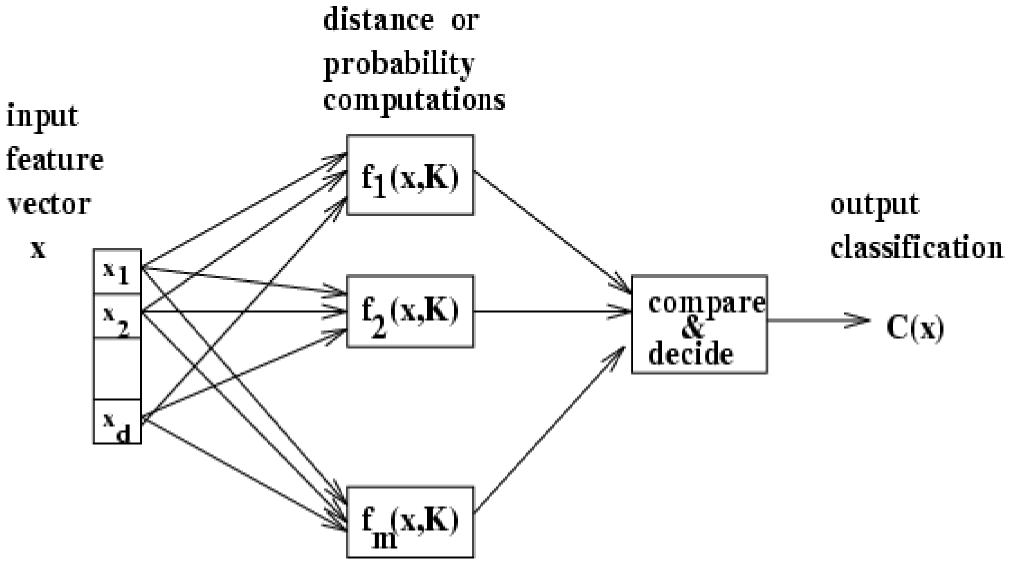
(class) character	area	height	width	number #holes	number #strokes	(cx,cy) center	best axis	least inertia
'A'	medium	high	3/4	1	3	1/2,2/3	90	medium
'B'	medium	high	3/4	2	1	1/3,1/2	90	large
'8'	medium	high	2/3	2	0	1/2,1/2	90	medium
'0'	medium	high	2/3	1	0	1/2,1/2	90	large
'1'	low	high	1/4	0	1	1/2,1/2	90	low
'W'	high	high	1	0	4	1/2,2/3	90	large
'X'	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

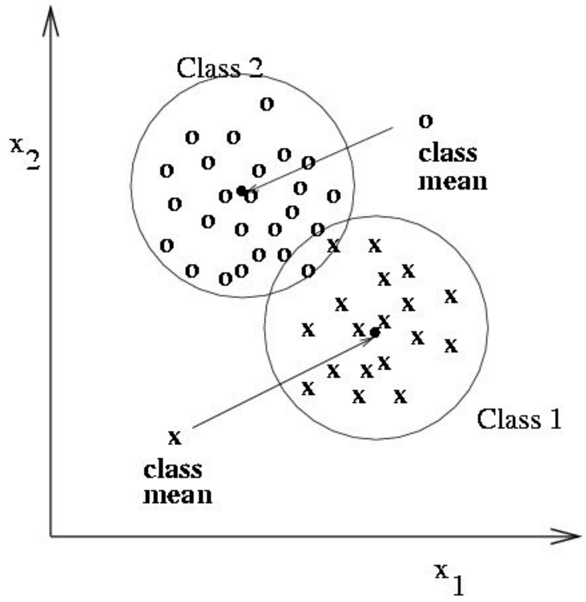
- Classes: set of m known categories of objects
  (a) might have a known description for each
  (b) might have a set of samples for each
- Reject Class:
  a generic class for objects not in any of
  the designated known classes
- Classifier:
  Assigns object to a class based on features

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



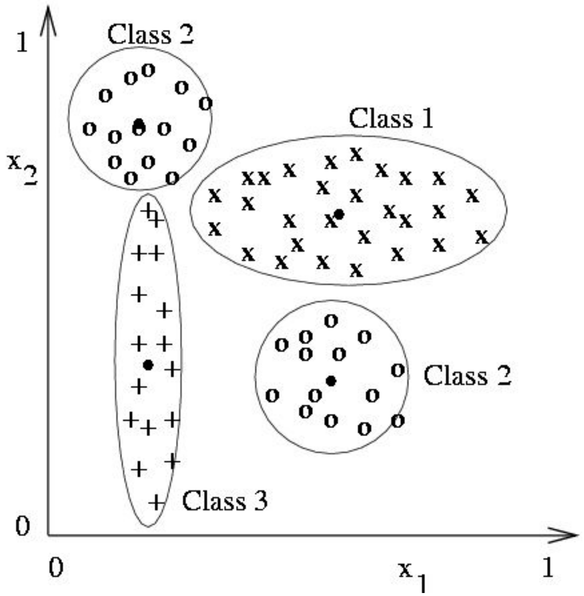
Classification using nearest class mean



- 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 x to class mean vector x_c by scaling by the spread or *standard deviation* σ_i of class c along each dimension *i*.

scaled Euclidean distance from x to class mean x_c

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

Nearest Mean

- What’s good about the nearest mean approach?

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.

Nearest Neighbor

- Pros

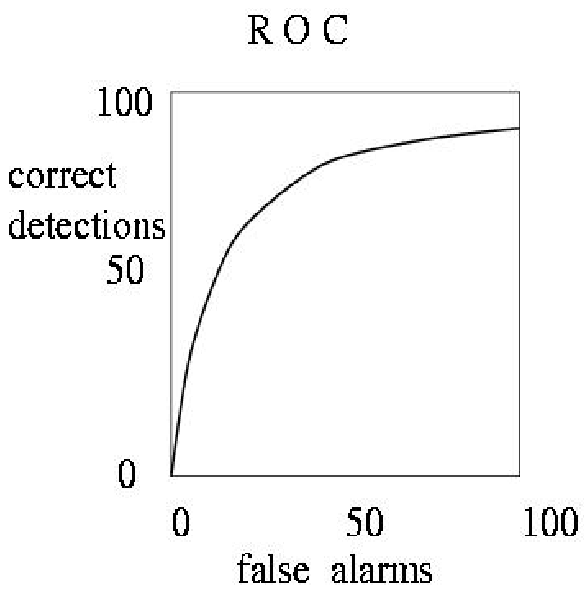
- Cons

Evaluating Results

- We need a way to measure the performance of any classification task.
- Binary classifier: Face or not Face
  - We can talk about true positives, false positives, true negatives, false negatives
- Multiway classifier: ‘a’ or ‘b’ or ‘c’ .....
  - For each class, what percentage correct and what percentage for each of the wrong classes

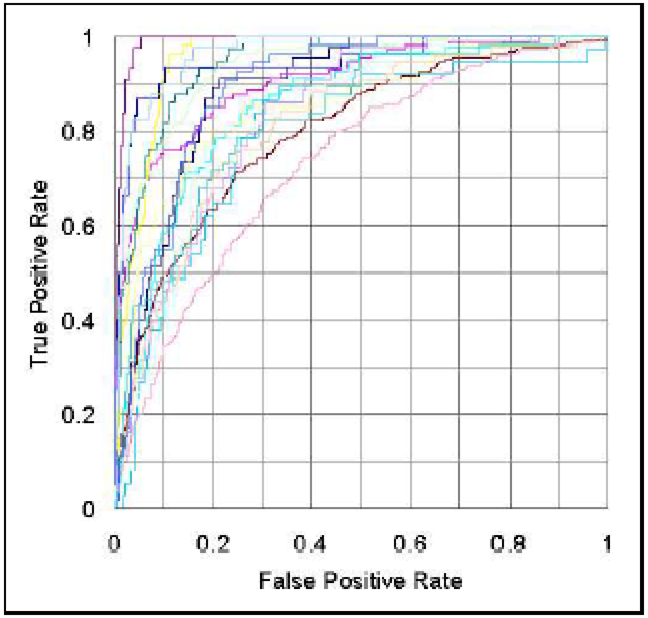
Receiver Operating Curve ROC

- 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)

An ROC from our work:



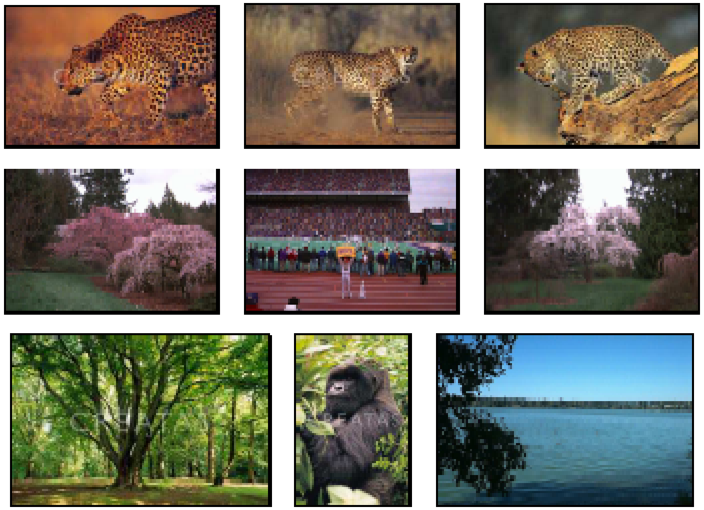


Figure 4: The top 3 test results for cheetah, cherry tree, and tree.

Confusion matrix shows empirical performance for multiclass problems

		class j output by the pattern recognition system										
		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'R'
	'0'	97	0	0	0	0	0	1	0	0	1	1
	'1'	0	98	0	0	1	0	0	1	0	0	0
true	'2'	0	0	96	1	0	1	0	1	0	0	1
object	'3'	0	0	2	95	0	1	0	0	1	0	1
class	'4'	0	0	0	0	98	0	0	0	0	2	0
	'5'	0	0	0	1	0	97	0	0	0	0	2
i	'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	1	95	0

Confusion may be unavoidable between some classes, for example, between 9's and 4's.

Bayesian decision-making

- Classify into class w that is most likely based on observations X. The following distributions are needed.

$$\begin{aligned} \textbf{class conditional distribution}: &\quad p(x|\omega_i)\ for\ each\ class\ \omega_i \\ \textbf{a priori probability}: &\quad P(\omega_i)\ for\ each\ class\ \omega_i \\ \textbf{unconditional distribution}: &\quad p(x) \end{aligned}$$

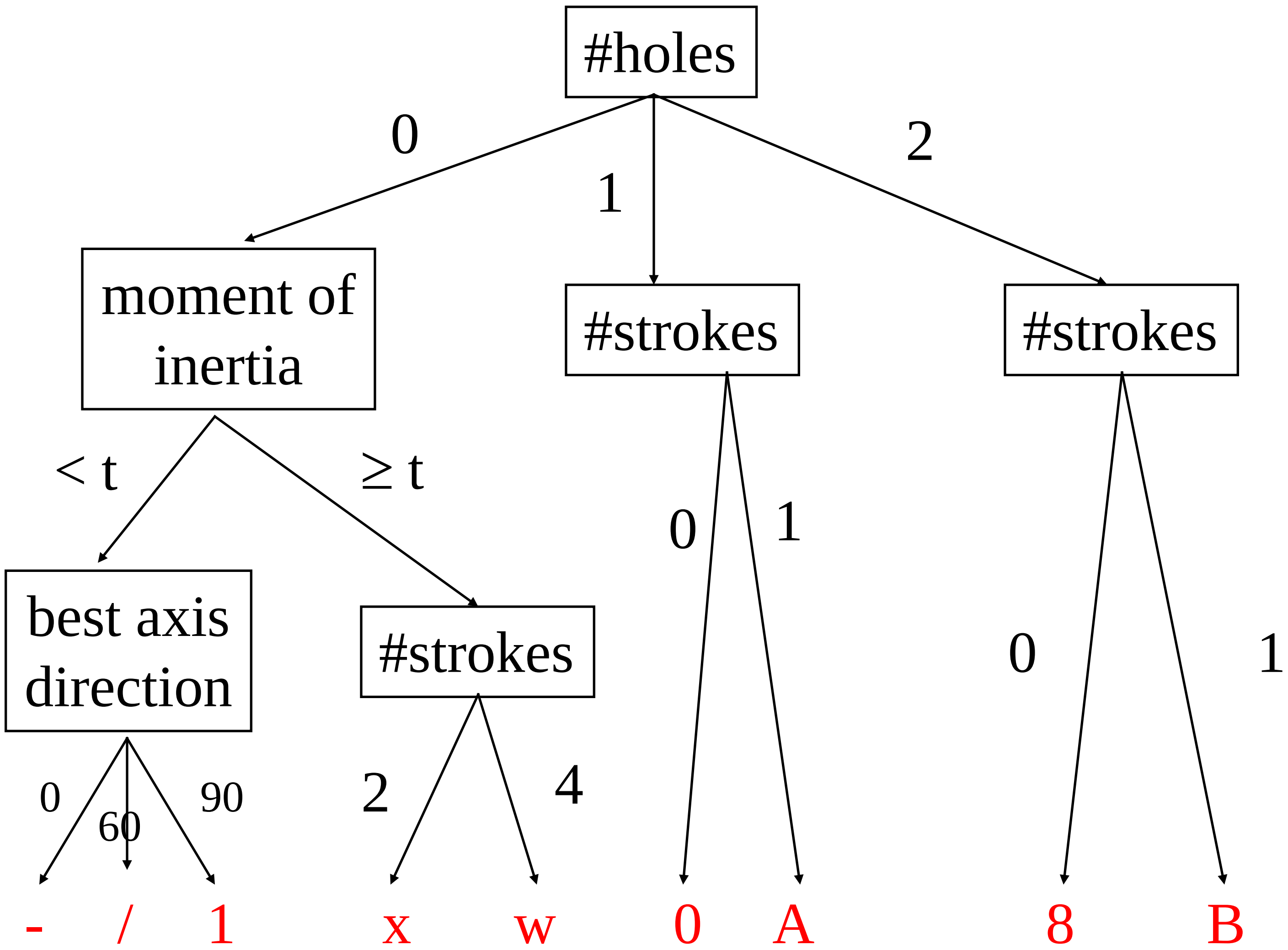
- Then we have:

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

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

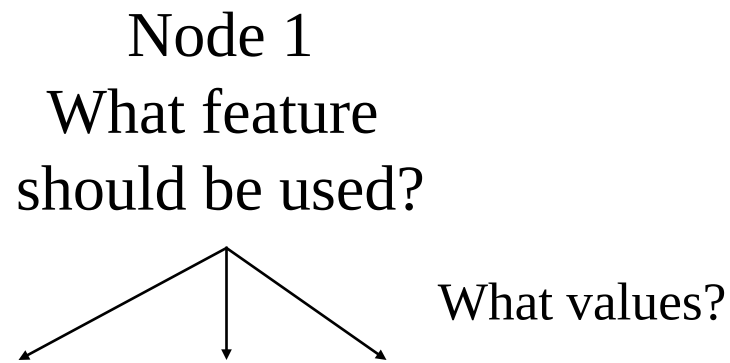
$x_1=(f_{11},f_{12},\ldots f_{1m})$

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

.

.

$x_n=(f_{n1},f_{22}, \quad f_{2m})$



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

We'll look at a variant called **information content**.

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 (no discrimination).

The greater the discrimination power, the larger the entropy will be.

Entropy

- Two class problem: class I and class II
- Suppose ½ the set belongs to class I and ½ belongs to class II?
- Then

$$\text{entropy} = -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2}$$

$$= (-\tfrac{1}{2})(-1) - \tfrac{1}{2}(-1)$$

$$= 1$$

Information Content

The information content I(C;F) of the class variable C with possible values $\{c_1, c_2, \ldots c_m\}$ with respect to the feature variable F with possible values $\{f_1, f_2, \ldots, f_d\}$ is defined by:

$$I(C;F) = \sum_{i=1}^{m} \sum_{j=1}^{d} P(C = c_i, F = f_j) log_2 \frac{P(C = c_i, F = f_j)}{P(C = c_i)P(F = f_j)}$$

- $P(C = c_i)$ is the probability of class C having value c_i.
- $P(F=f_j)$ is the probability of feature F having value f_j.
- $P(C=c_i,F=f_j)$ is the joint probability of class $C = c_i$ and variable $F = f_j$.

These are estimated from frequencies in the training data.

Example (from text)

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

How would you distinguish class I from class II?

Example (cont)

$$
\begin{aligned}
I(C,X) &= P(C=I,X=1)log_2\frac{P(C=I,X=1)}{P(C=I)P(X=1)} \\
&+ P(C=I,X=0)log_2\frac{P(C=I,X=0)}{P(C=I)P(X=0)} \\
&+ P(C=II,X=1)log_2\frac{P(C=II,X=1)}{P(C=II)P(X=1)} \\
&+ P(C=II,X=0)log_2\frac{P(C=II,X=0)}{P(C=II)P(X=0)} \\
&= .5log_2\frac{.5}{.5\times .75} + 0 + .25log_2\frac{.25}{.5\times .25} + .25log_2\frac{.25}{.5\times .75} \\
&= 0.311
\end{aligned}
$$

$$
\begin{aligned}
I(C,Y) &= .5log_2\frac{.5}{.5\times .5} + 0 + .5log_2\frac{.5}{.5\times .5} + 0 \\
&= 1.0
\end{aligned}
$$

$$
\begin{aligned}
I(C,Z) &= .25log_2\frac{.25}{.5\times .5} + .25log_2\frac{.25}{.5\times .5} + .25log_2\frac{.25}{.5\times .5} + .25log_2\frac{.25}{.5\times .5} \\
&= 0.0
\end{aligned}
$$

Using Information Content

- Start with the root of the decision tree and the whole training set.
- Compute I(C,F) for each feature F.
- Choose the feature F with highest information content for the root node.
- Create branches for each value f of F.
- On each branch, create a new node with reduced training set and repeat recursively.

Using Information Content

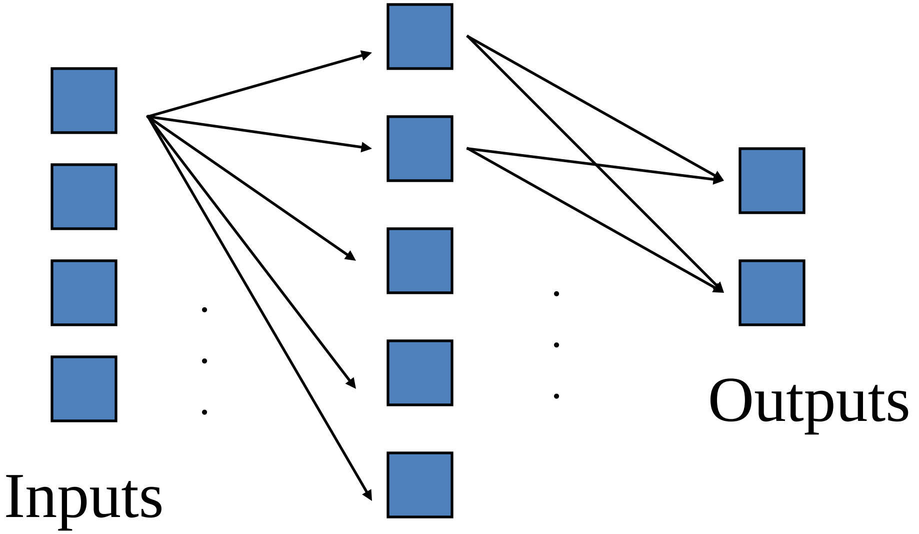
X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

- What would the optimal tree look like for features X, Y, and Z and class I and II in the small example?

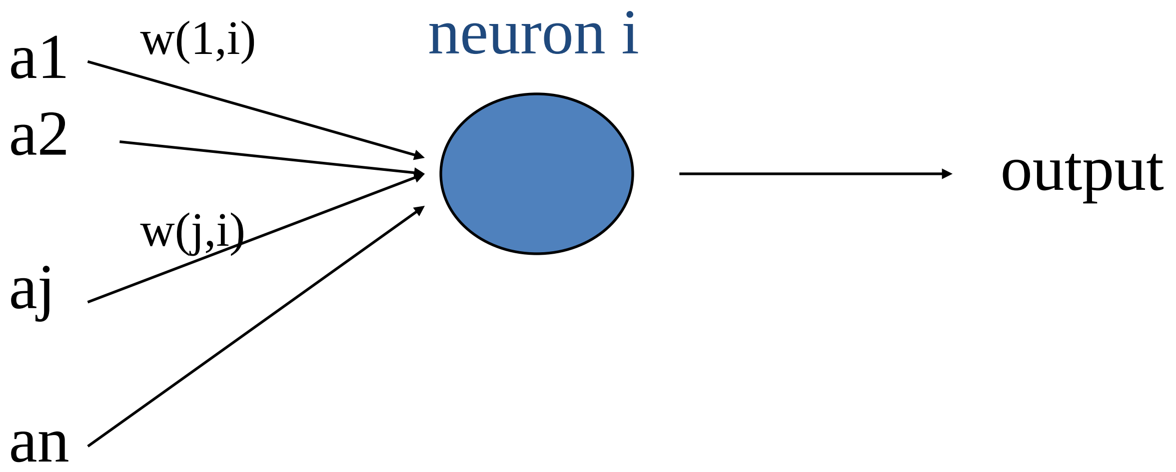
Artificial Neural Nets

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



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

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.

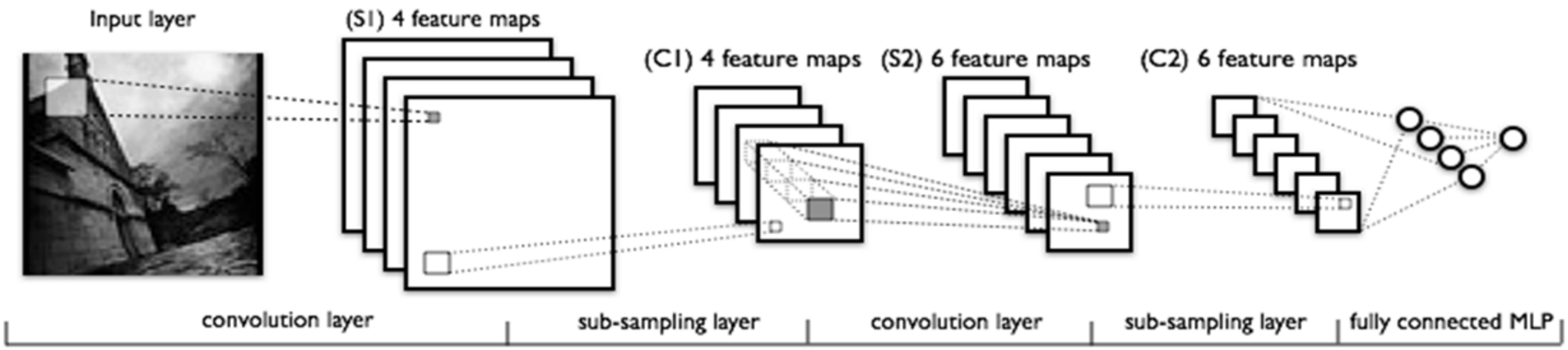
The most common method is called back propagation.

We've used a free package called NevProp or just
The WEKA machine learning package.

Convolutional Neural Nets

- CNNs were invented in the 90s, but they have returned and become very popular in computer vision in the last few years, because they have been used to achieve higher than competitor accuracy on several benchmark data sets in object recognition.
- They are related to “deep learning”.
- They have multiple layers, some of them do convolutions instead of having full connections.

Simple CNN

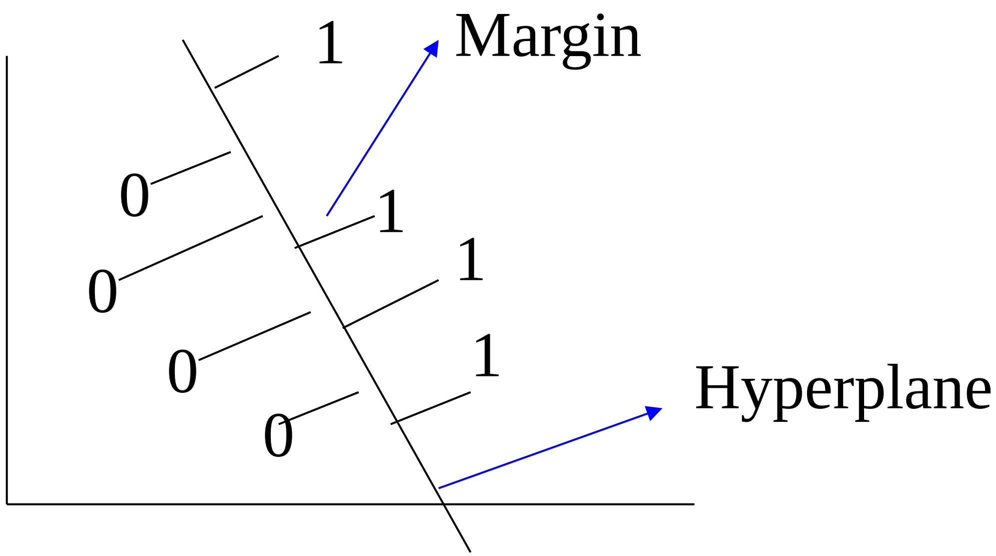

Input layer
(S1) 4 feature maps
(C1) 4 feature maps
(S2) 6 feature maps
(C2) 6 feature maps
convolution layer
sub-sampling layer
convolution layer
sub-sampling layer
fully connected MLP

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:

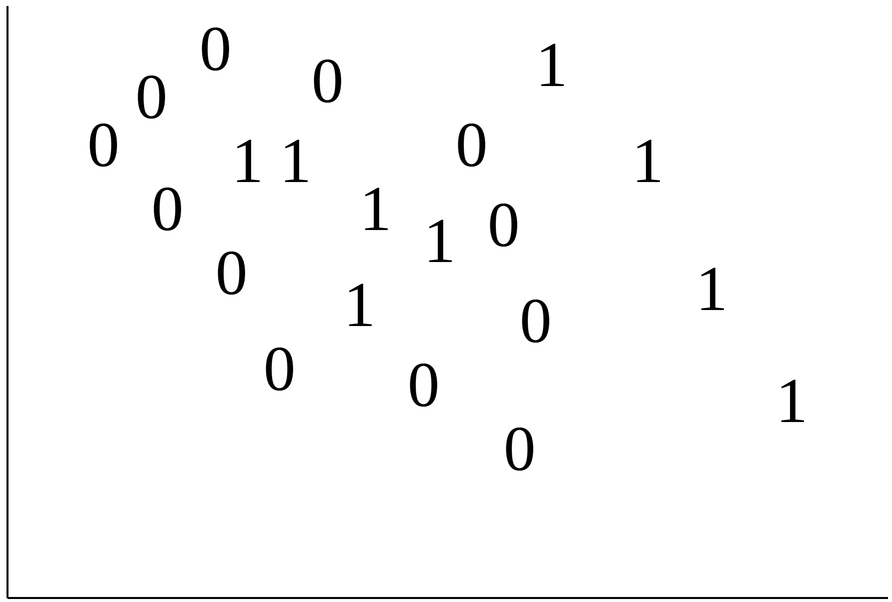
- Maximum margin hyperplanes
- A kernel ‘trick’.

Maximal Margin



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

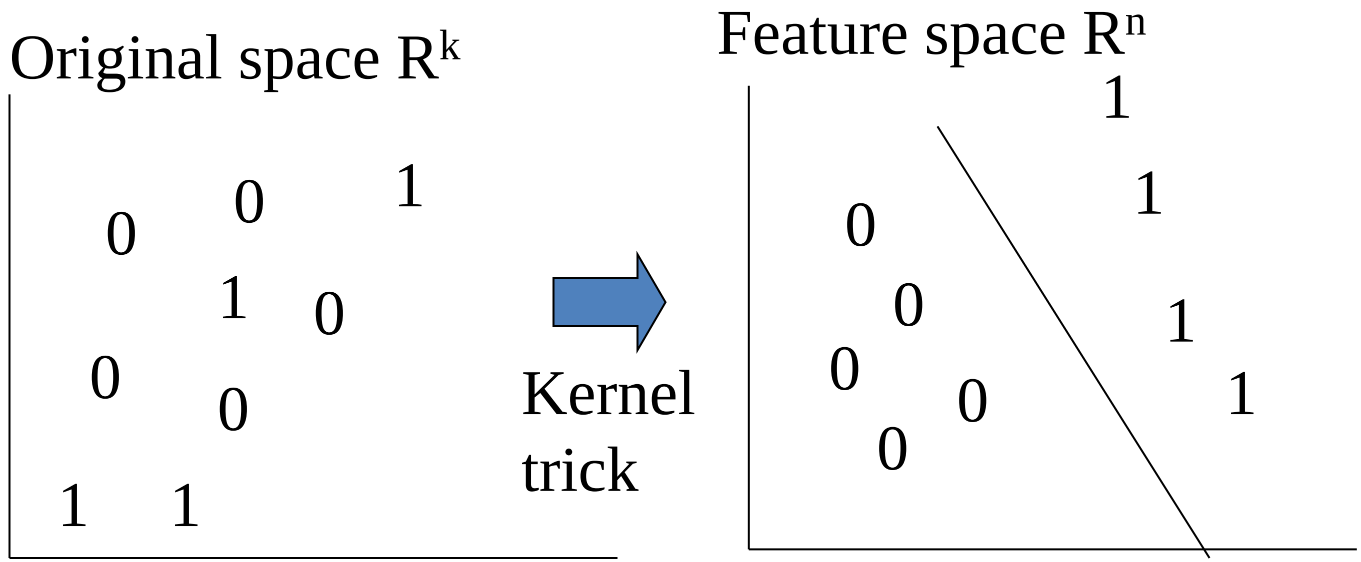
Non-separable data



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

The kernel trick

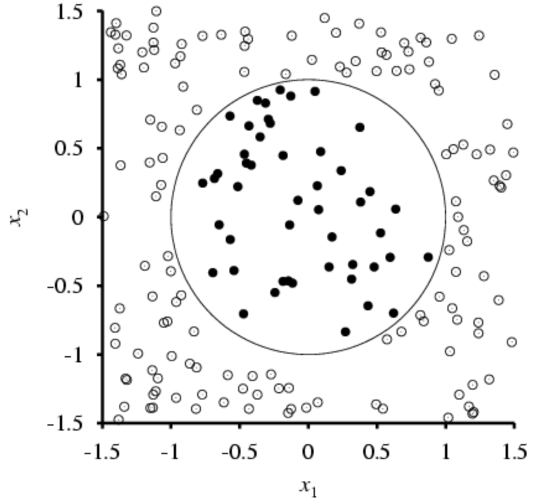
The SVM algorithm implicitly maps the original data to a feature space of possibly infinite dimension in which data (which is not separable in the original space) becomes separable in the feature space.



The kernel trick

- What is this space?
- The user defines it.
- It's usually a dot product space.
- Example: If we have 2 vectors X and Y, we can work with $(X \cdot Y)$ or $\exp(X \cdot Y)$

Example from AI Text



True decision boundary is $x_1^2 + x_2^2 \leq 1$.

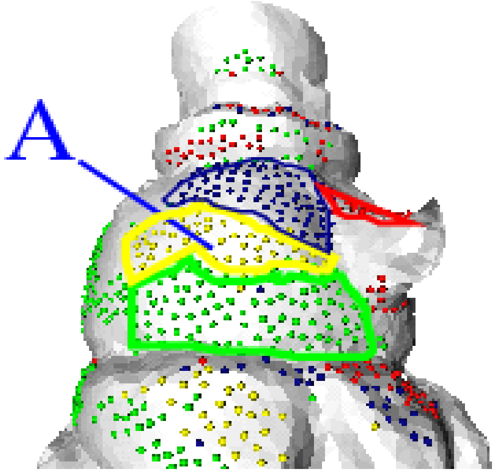
- For this problem $F(\mathbf{x_i}) \cdot F(\mathbf{x_j})$ is just $(\mathbf{x_i} \cdot \mathbf{x_j})^2$, which is called a kernel function.

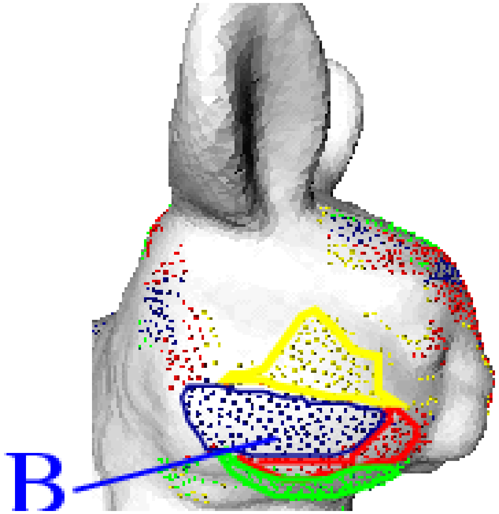
Application

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

Kernel Function used in our 3D Computer Vision Work

- $k(A,B) = \exp(-\theta^2_{AB}/\sigma^2)$
- A and B are shape descriptors (big vectors).
- θ is the angle between these vectors.
- σ^2 is the "width" of the kernel.





We used SVMs for Insect Recognition

EM for Classification

- The EM algorithm was used as a clustering algorithm for image segmentation.

- It can also be used as a classifier, by creating a Gaussian “model” for each class to be learned.

Summary

- There are multiple kinds of classifiers developed in machine learning research.
- We can use and have used pretty much all of them in computer vision classification, detection, and recognition tasks.