Kernel Machines

- A relatively new learning methodology (1992) derived from statistical learning theory.
- Became famous when it gave accuracy comparable to neural nets in a handwriting recognition class.
- Was introduced to computer vision researchers by Tomaso Poggio at MIT who started using it for face detection and got better results than neural nets.
- Has become very popular and widely used with packages available.

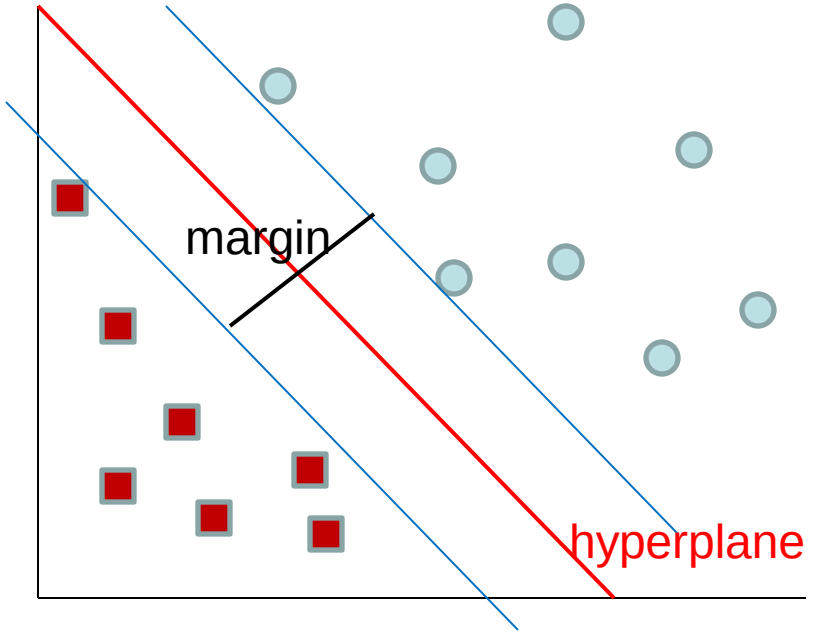
Support Vector Machines (SVM)

- Support vector machines are learning algorithms that try to find a hyperplane that separates the different classes of data the most.
- They are a specific kind of kernel machines based on two key ideas:
  - maximum margin hyperplanes
  - a kernel ‘trick’

Maximal Margin (2 class problem)

In 2D space,
a hyperplane is
a line.

In 3D space,
it is a plane.



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

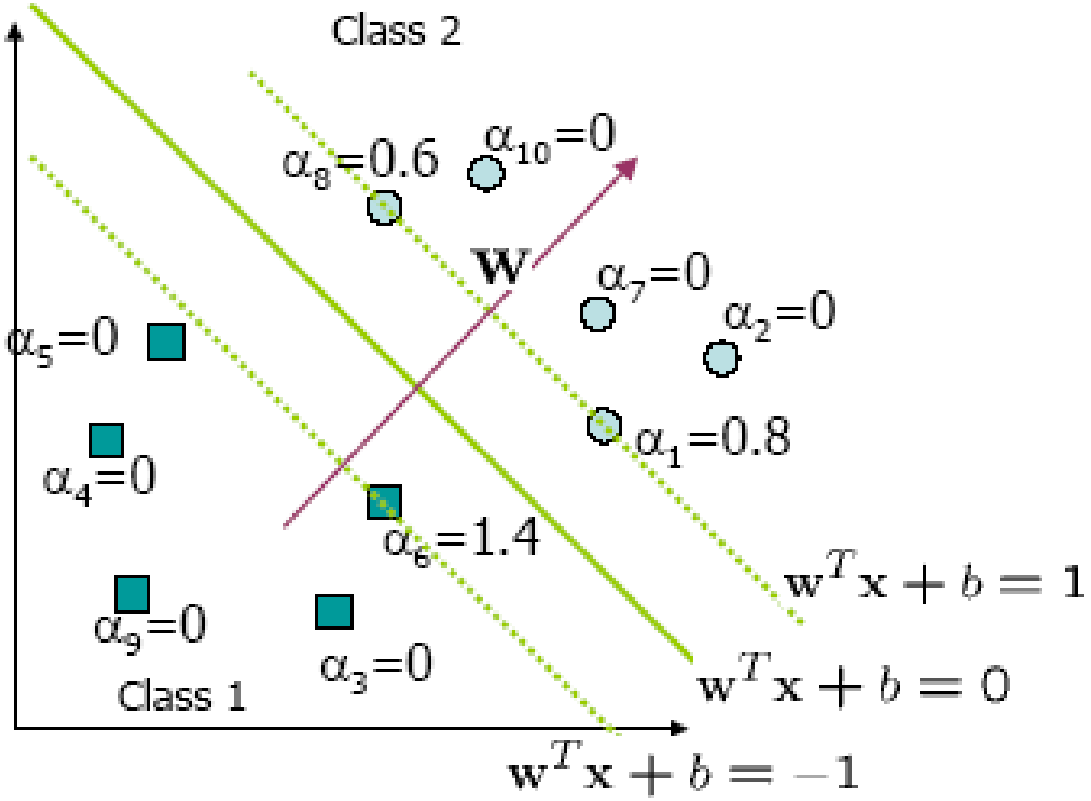
Support Vectors

- The weights α_i associated with data points are zero, except for those points closest to the separator.

- The points with nonzero weights are called the support vectors (because they hold up the separating plane).

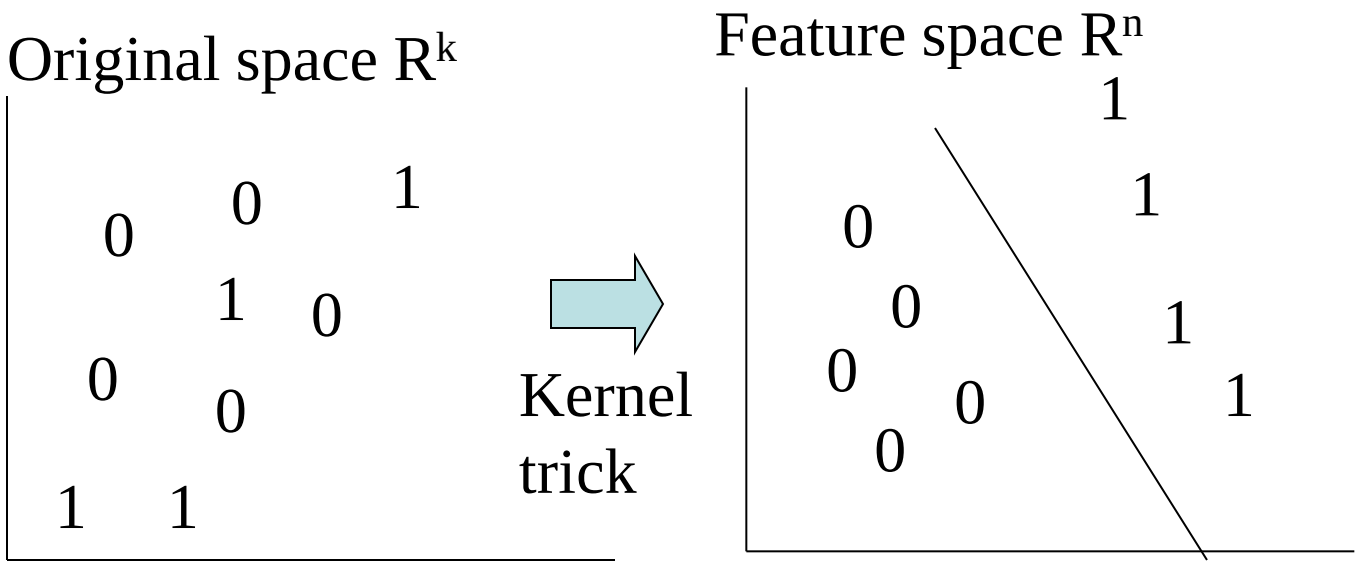
- Because there are many fewer support vectors than total data points, the number of parameters defining the optimal separator is small.

A Geometric Interpretation

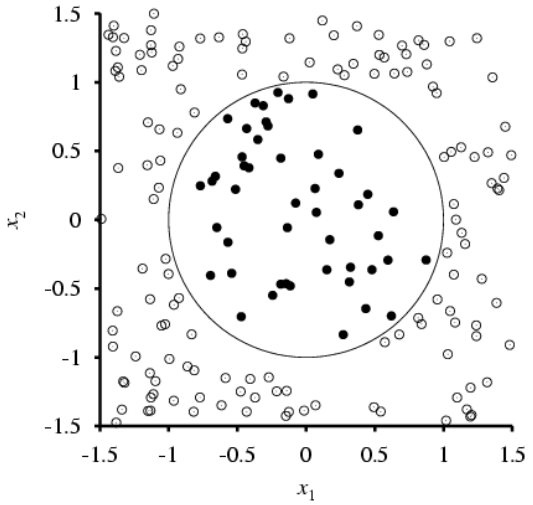


The Kernel Trick

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.



Example from Text



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

- Mapping the data to the 3D space defined by
  $f_1 = x_1^2$, $f_2 = x_2^2$, $f_3 = 2^{1/2} x_1 x_2$
  makes it linearly separable by a plane in 3D.

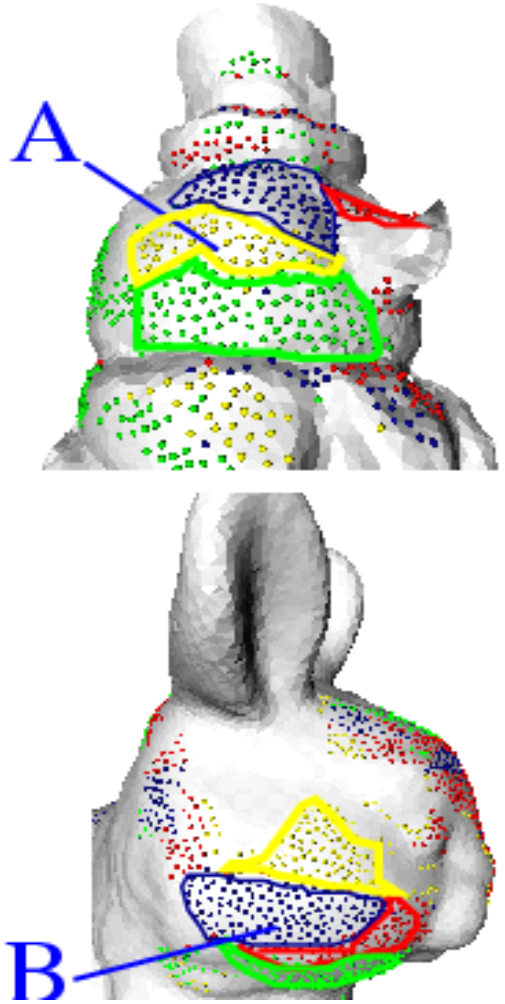
- For this problem $F(x_i) \bullet F(x_j)$ is just $(xi \bullet xj)^2$, which is called a kernel function.

Kernel Functions

- The kernel function is designed by the developer of the SVM.
- It is applied to pairs of input data to evaluate dot products in some corresponding feature space.
- Kernels can be all sorts of functions including polynomials and exponentials.

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.



Unsupervised Learning

- Find patterns in the data.
- Group the data into clusters.
- Many clustering algorithms.
  - K means clustering
  - EM clustering
  - Graph-Theoretic Clustering
  - Clustering by Graph Cuts
  - etc

Clustering by K-means Algorithm

Form K-means clusters from a set of n-dimensional feature vectors

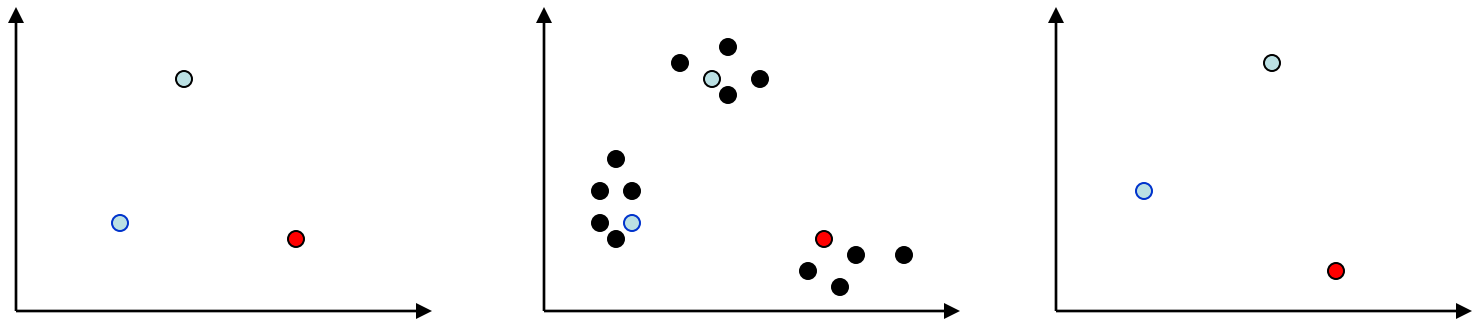
1. Set ic (iteration count) to 1

2. Choose randomly a set of K means $m_1(1), \ldots, m_K(1)$.

3. For each vector x_i, compute $D(x_i, m_k(ic))$, $k=1,\ldots K$
   and assign x_i to the cluster C_j with nearest mean.

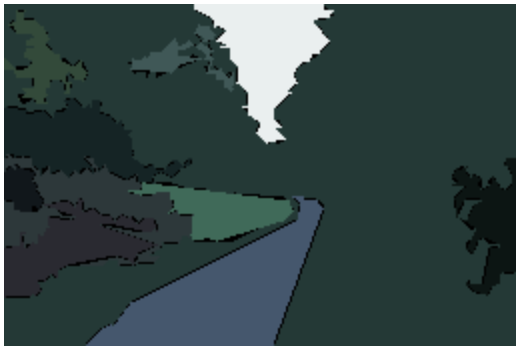
4. Increment ic by 1, update the means to get $m_1(ic),\ldots,m_K(ic)$.

5. Repeat steps 3 and 4 until $C_k(ic) = C_k(ic+1)$ for all k.



K-Means Classifier (shown on RGB color data)

original data
one RGB per pixel



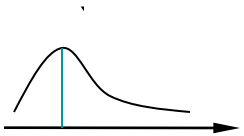
color clusters

K-Means → EM

The clusters are usually Gaussian distributions.

- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$

    (μ_j, Σ_j) and $P(C_j)$ for each cluster j.



- Iteration Step:
  - Estimate the cluster of each datum

    $p(C_j \mid x_i)$ ➡ Expectation
  - Re-estimate the cluster parameters

    $(\mu_j, \Sigma_j), p(C_j)$ For each cluster j ➡ Maximization

The resultant set of clusters is called a **mixture model**; if the distributions are Gaussian, it's a Gaussian mixture.

EM Algorithm Summary

- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$

    (μ_j, Σ_j) and $p(C_j)$ for each cluster j.
- Iteration Step:
  - Expectation Step

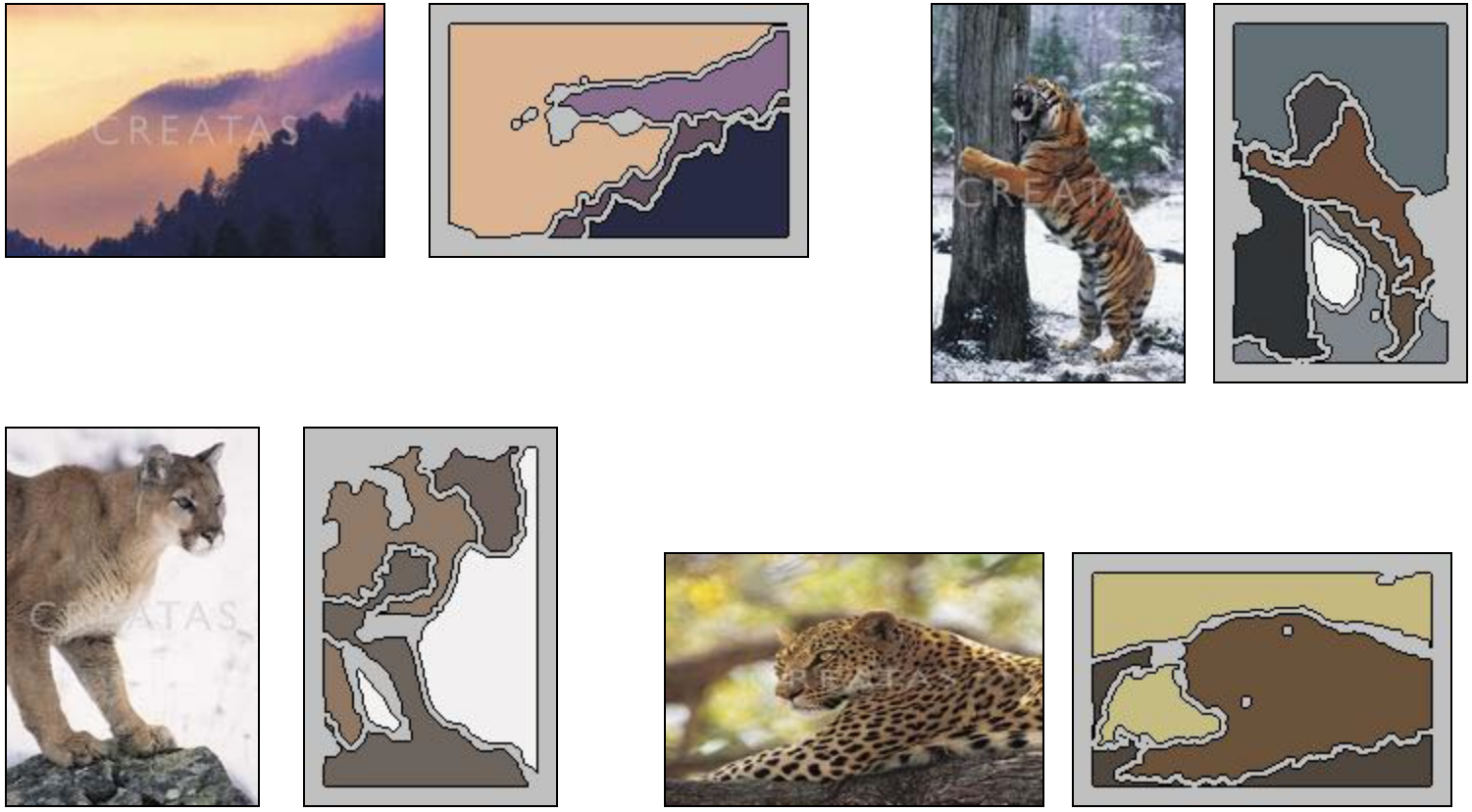
$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

  - Maximization Step

$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

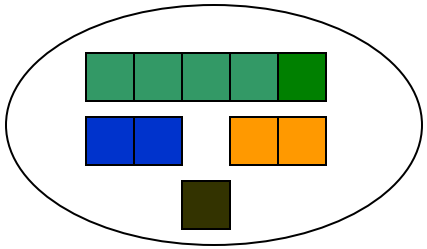
EM Clustering using color and texture information at each pixel

(from Blobworld)

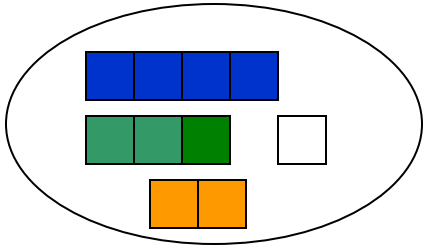


EM for Classification of Images in Terms of their Color Regions

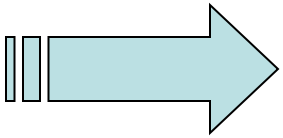
Initial Model for “trees”



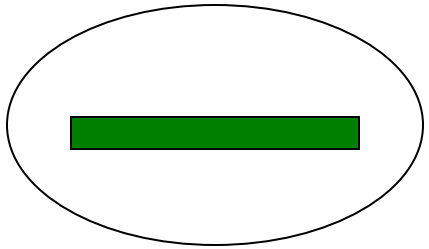
Initial Model for “sky”



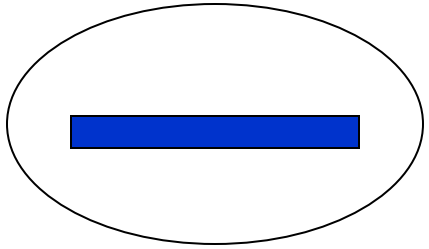
EM



Final Model for “trees”



Final Model for “sky”



Sample Results

cheetah

Sample Results (Cont.)

grass

Sample Results (Cont.)

lion