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)

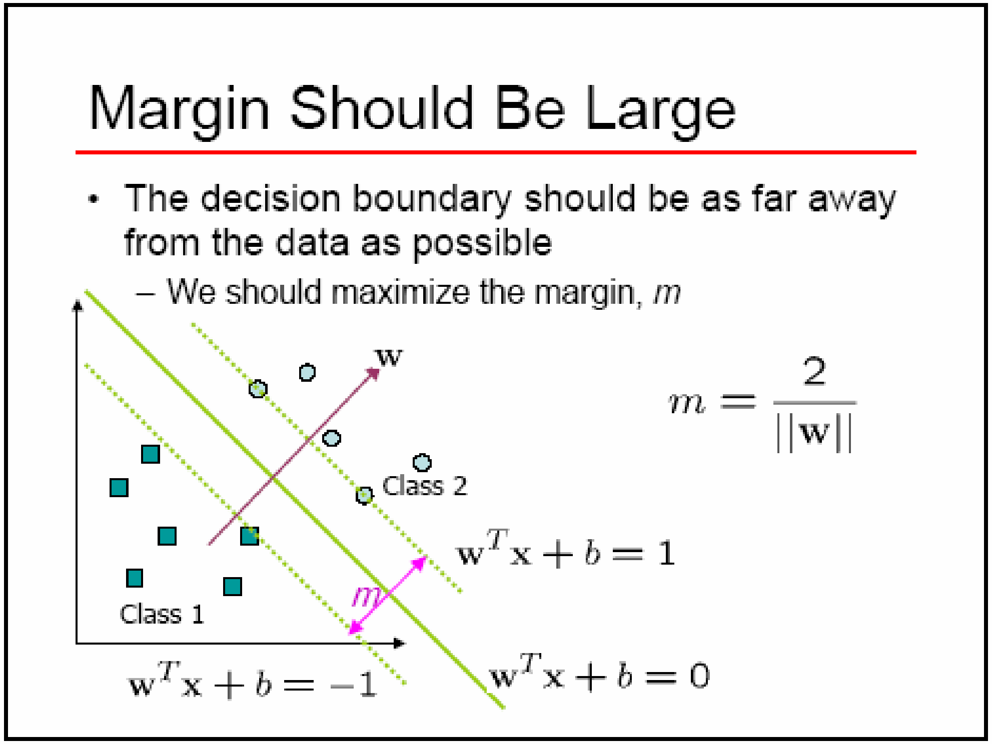
- Support vector machines are learning algorithms that try to find a hyperplane that separates the different classes of data the most.

- They are a specific kind of kernel machines based on two key ideas:

  - maximum margin hyperplanes

  - a kernel ‘trick’

Maximal Margin (2 class problem)



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

The Optimization Problem

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$
- A constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

The Optimization Problem

- We can transform the problem to its dual

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

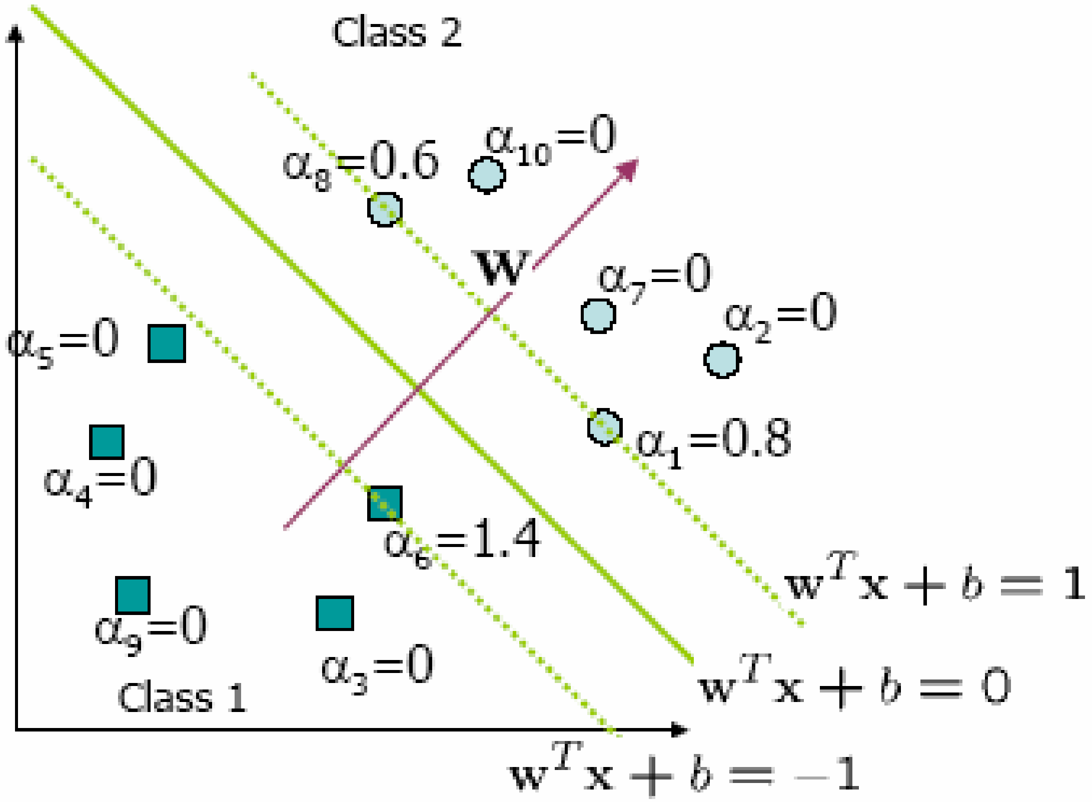
$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is a quadratic programming (QP) problem
  - Global maximum of α_i can always be found
- w can be recovered by $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

Support Vectors

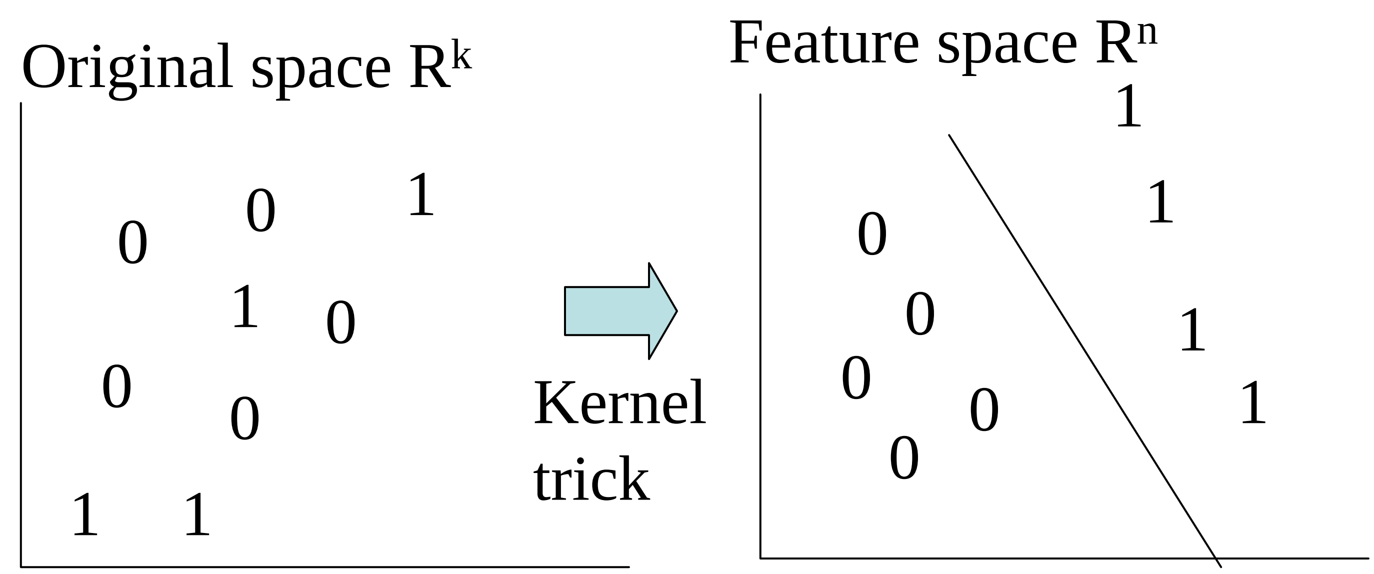
- The weights α_i associated with data points are zero, except for those points closest to the separator.
- The points with nonzero weights are called the support vectors (because they hold up the separating plane).
- Because there are many fewer support vectors than total data points, the number of 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.



A Few Details

- Consider the expression being maximized.

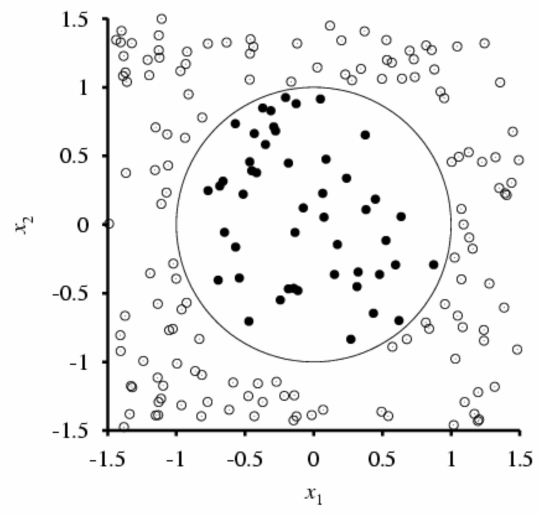
$$\sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (x_i \bullet x_j)$$

- In order to find linear separators in a high-dimensional space F(x), we can replace

  $x_i \bullet x_j$ with $F(x_i) \bullet F(x_j)$.

- Most important, $F(x_i) \bullet F(x_j)$ can often be computed without first computing F for each point.

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.

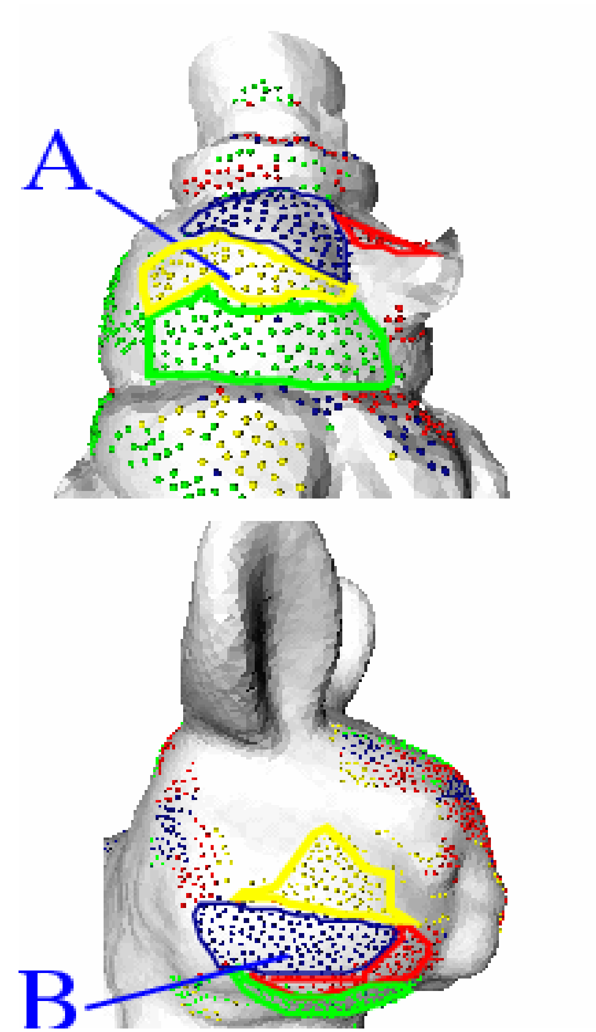
- For this problem $F(x_i) \cdot F(x_j)$ is just $(xi \cdot xj)^{2,}$ which is called a kernel function.

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

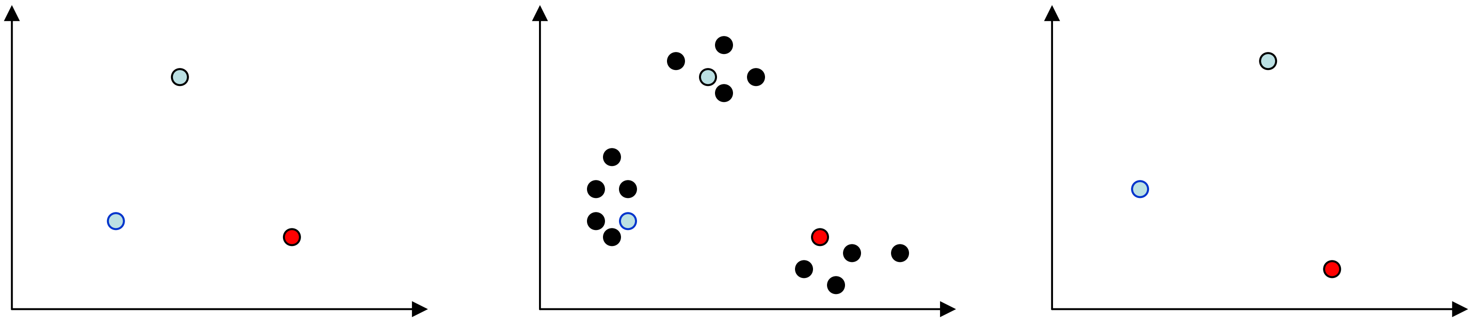
1. Set ic (iteration count) to 1

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)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

$x_i=\{r_i, g_i, b_i\}$

…

→ Classifier (K-Means) →

Classification Results

$x_1 \rightarrow C(x_1)$

$x_2 \rightarrow C(x_2)$

…

$x_i \rightarrow C(x_i)$

…

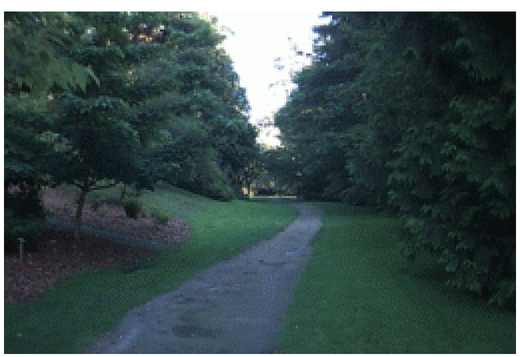
Cluster Parameters

m_1 for C_1

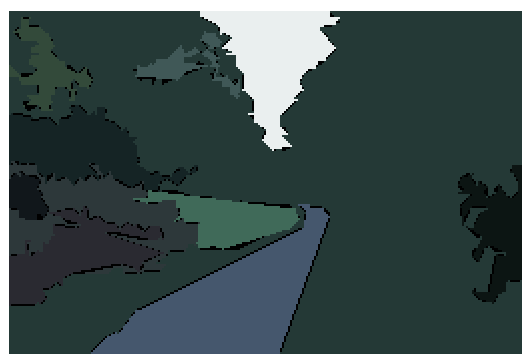
m_2 for C_2

…

m_k for C_k



original data
one RGB per pixel



color clusters

K-Means Classifier (Cont.)

Input (Known)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

$x_i=\{r_i, g_i, b_i\}$

…

Output (Unknown)

Cluster Parameters
m_1 for C_1
m_2 for C_2
…
m_k for C_k

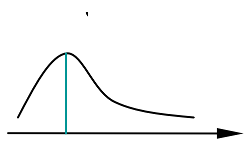
Classification Results
$x_1 \rightarrow C(x_1)$
$x_2 \rightarrow C(x_2)$
…
$x_i \rightarrow C(x_i)$
…

K-Means → EM

The clusters are usually Gaussian distributions.

- <u>Boot Step</u>:
  - Initialize K clusters: $C_1, \ldots, C_K$

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



- <u>Iteration Step</u>:
  - Estimate the cluster of each datum

    $$p(C_j \mid x_i)$$

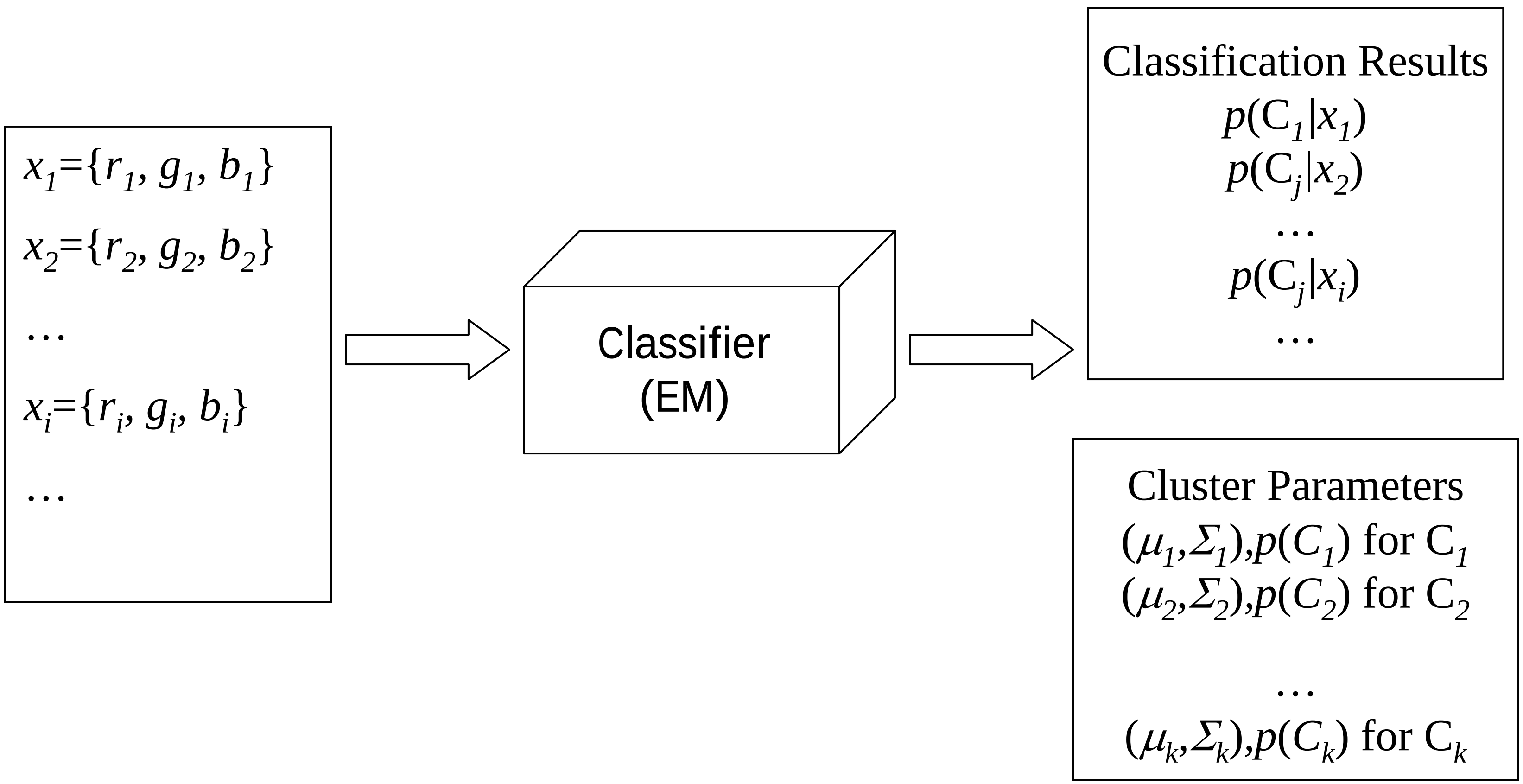
    ⇨ Expectation
  - Re-estimate the cluster parameters

    $$(\mu_j, \Sigma_j), p(C_j)$$ 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 Classifier



EM Classifier (Cont.)

Input (Known)	Output (Unknown)	
$x_1=\{r_1, g_1, b_1\}$ $x_2=\{r_2, g_2, b_2\}$ … $x_i=\{r_i, g_i, b_i\}$ …	Cluster Parameters (μ_1,Σ_1), $p(C_1)$ for C_1 (μ_2,Σ_2), $p(C_2)$ for C_2 … (μ_k,Σ_k), $p(C_k)$ for C_k	Classification Results $p(C_1\|x_1)$ $p(C_j\|x_2)$ … $p(C_j\|x_i)$ …

Expectation Step

Input (Known)		Input (Estimation)		Output
$x_1=\{r_1, g_1, b_1\}$ $x_2=\{r_2, g_2, b_2\}$ … $x_i=\{r_i, g_i, b_i\}$ …	+	Cluster Parameters (μ_1,Σ_1), $p(C_1)$ for C_1 (μ_2,Σ_2), $p(C_2)$ for C_2 … (μ_k,Σ_k), $p(C_k)$ for C_k	⇨	Classification Results $p(C_1\|x_1)$ $p(C_j\|x_2)$ … $p(C_j\|x_i)$ …

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

Maximization Step

Input (Known)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

$x_i=\{r_i, g_i, b_i\}$

…

+

Input (Estimation)

Classification Results

$p(C_1|x_1)$

$p(C_j|x_2)$

…

$p(C_j|x_i)$

…

⇒

Output

Cluster Parameters

(μ_1,Σ_1), $p(C_1)$ for C_1

(μ_2,Σ_2), $p(C_2)$ for C_2

…

(μ_k,Σ_k), $p(C_k)$ for C_k

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

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

$$p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

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

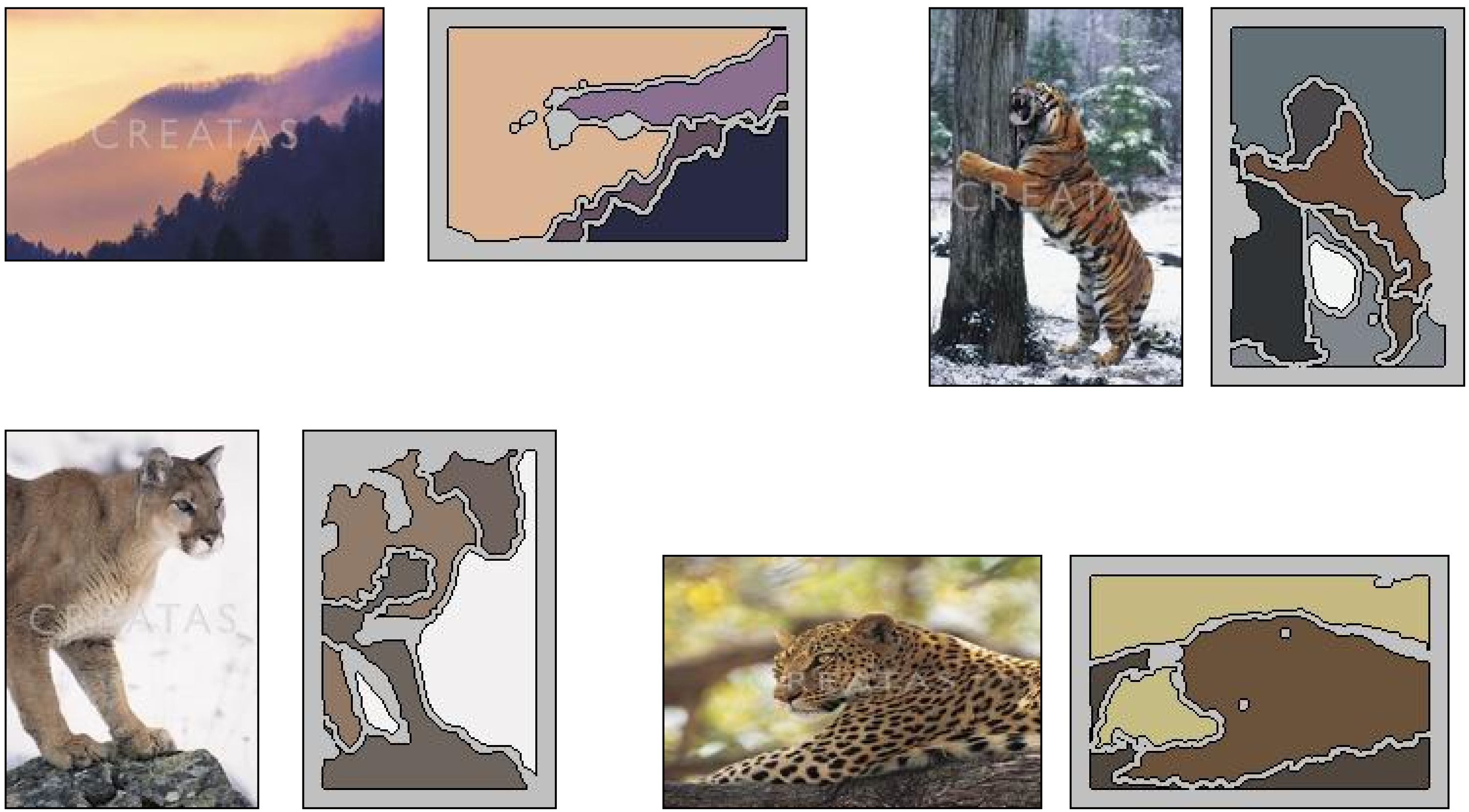
$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

  - Maximization Step

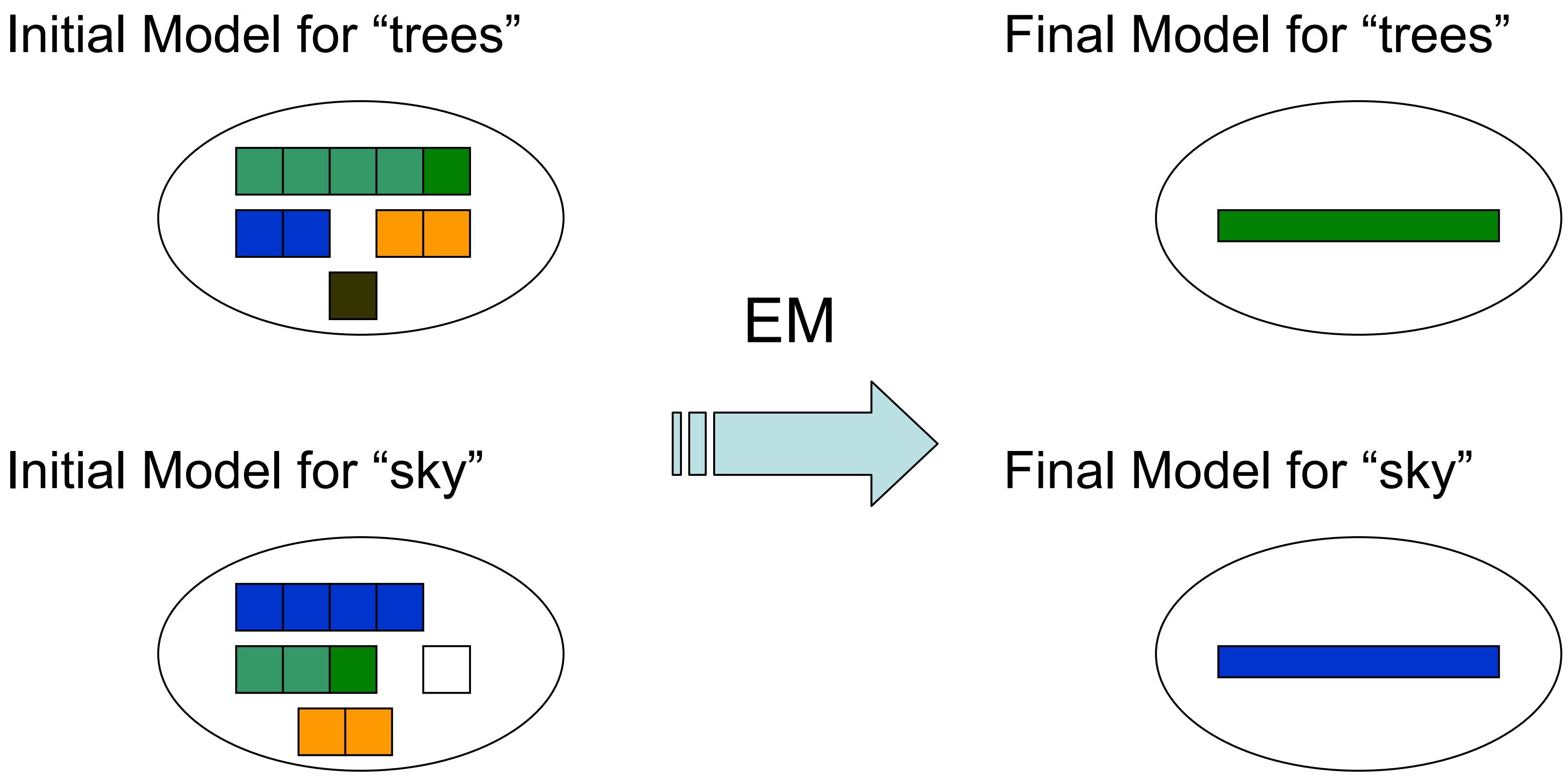
$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

EM Clustering using color and texture information at each pixel

(from Blobworld)

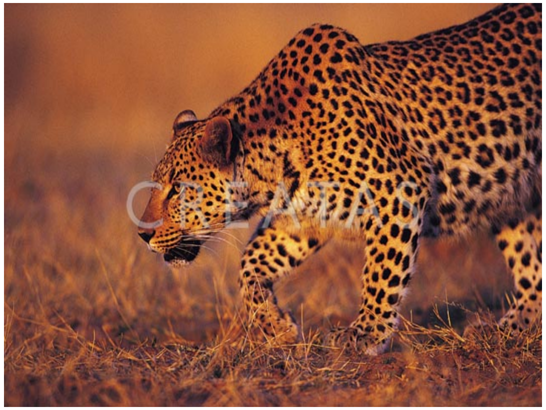


EM for Classification of Images in Terms of their Color Regions



Sample Results

cheetah



Sample Results (Cont.)

grass

Sample Results (Cont.)

lion