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'

The SVM Equation

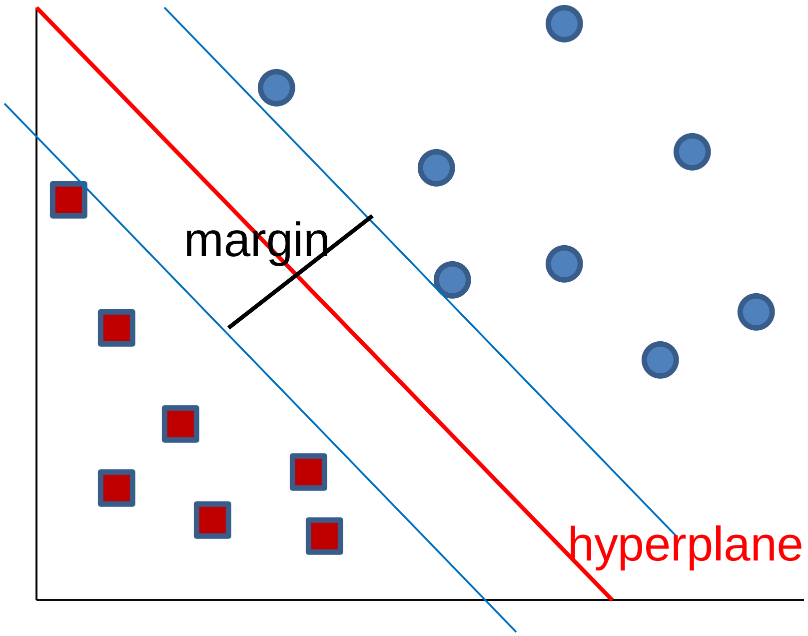
- $y_{SVM}(x_q) = \underset{c}{\text{argmax}} \sum_{i=1,m} \alpha_{i,c} K(x_i, x_q)$

- x_q is a query or unknown object
- c indexes the classes
- there are m support vectors x_i with weights $\alpha_{i,c}$, i=1 to m for class c
- K is the kernel function that compares x_i to x_q

Maximal Margin (2 class problem)

In 2D space,
a hyperplane is
a line.

In 3D space,
it is a plane.

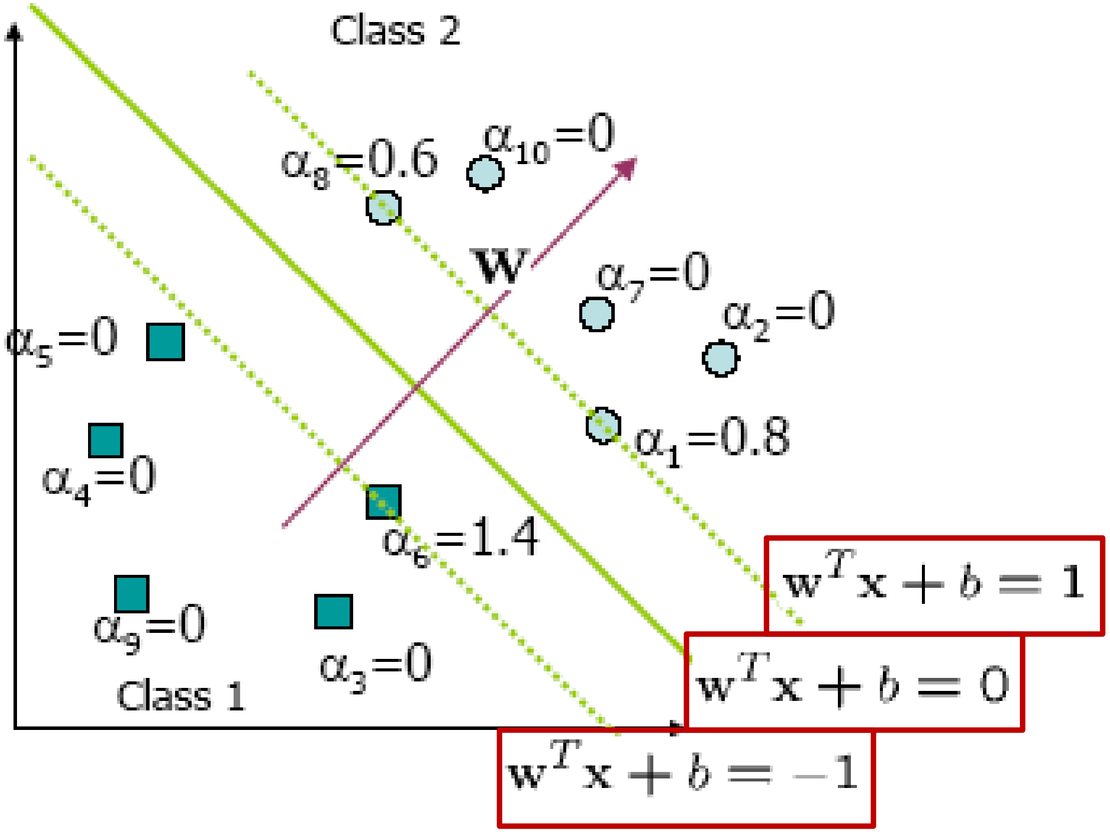


Find the hyperplane with maximal margin for all the points. This originates an optimization problem which has a unique solution.

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



Kernels

- A kernel is just a similarity function. It takes 2 inputs and decides how similar they are.

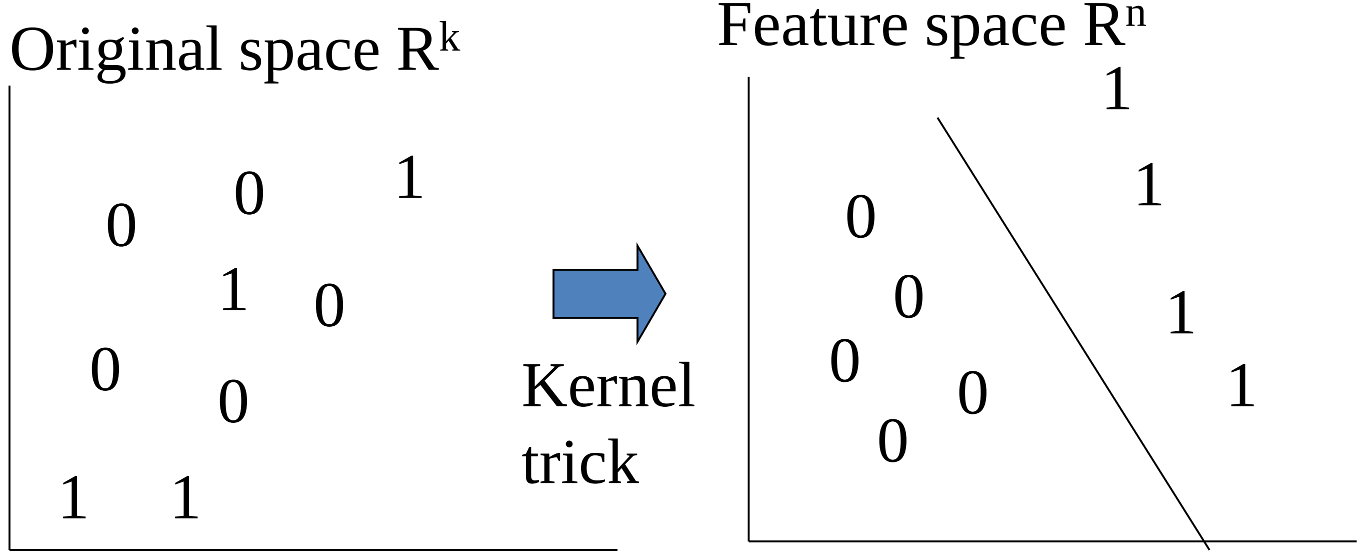
- Kernels offer an alternative to standard feature vectors. Instead of using a bunch of features, you define a single kernel to decide the similarity between two objects.

Kernels and SVMs

- Under some conditions, every kernel function can be expressed as a dot product in a (possibly infinite dimensional) feature space (Mercer's theorem)
- SVM machine learning can be expressed in terms of dot products.
- So SVM machines can use kernels instead of feature vectors.

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.

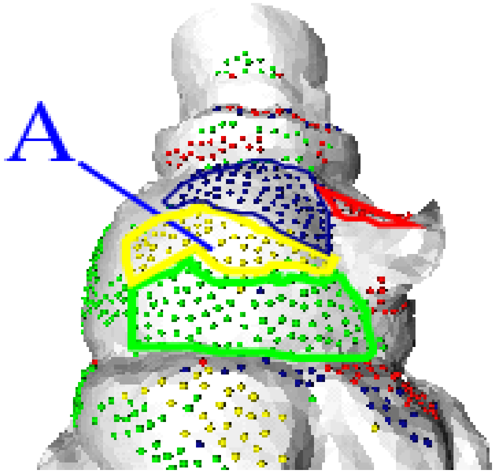


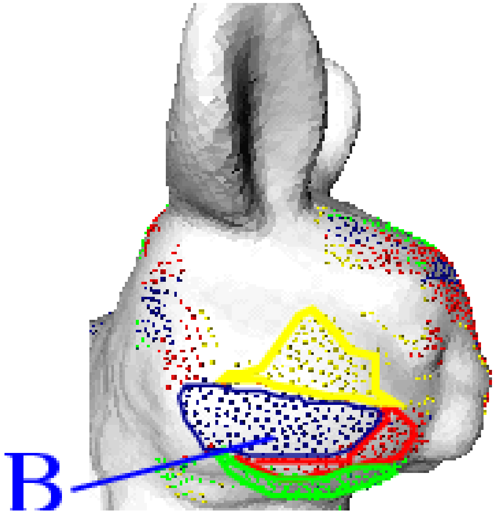
Kernel Functions

- 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.





What does SVM learning solve?

- The SVM is looking for the best separating plane in its alternate space.

- It solves a quadratic programming optimization problem

$$\underset{\alpha}{\mathit{argmax}} \sum_{j} \alpha_j - 1/2 \sum_{j,k} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j \bullet \mathbf{x}_k)$$

subject to $\alpha_j > 0$ and $\Sigma\alpha_j y_j = 0$.

- The equation for the separator for these optimal α_j is

$$h(\mathbf{x}) = \mathrm{sign}\Big(\sum_{j} \alpha_j \, y_j \, (x \bullet x_j) - b\Big)$$

Time taken to build model: 0.15 seconds

Correctly Classified Instances	319	83.5079 %
Incorrectly Classified Instances	63	16.4921 %
Kappa statistic	0.6685	
Mean absolute error	0.1649	
Root mean squared error	0.4061	
Relative absolute error	33.0372 %	
Root relative squared error	81.1136 %	
Total Number of Instances	382	

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.722	0.056	0.925	0.722	0.811	0.833	cal
	0.944	0.278	0.78	0.944	0.854	0.833	dor
W Avg.	0.835	0.17	0.851	0.835	0.833	0.833	

=== Confusion Matrix ===

```
  a   b   <-- classified as
 135  52 |   a = cal
  11 184 |   b = dor
```

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

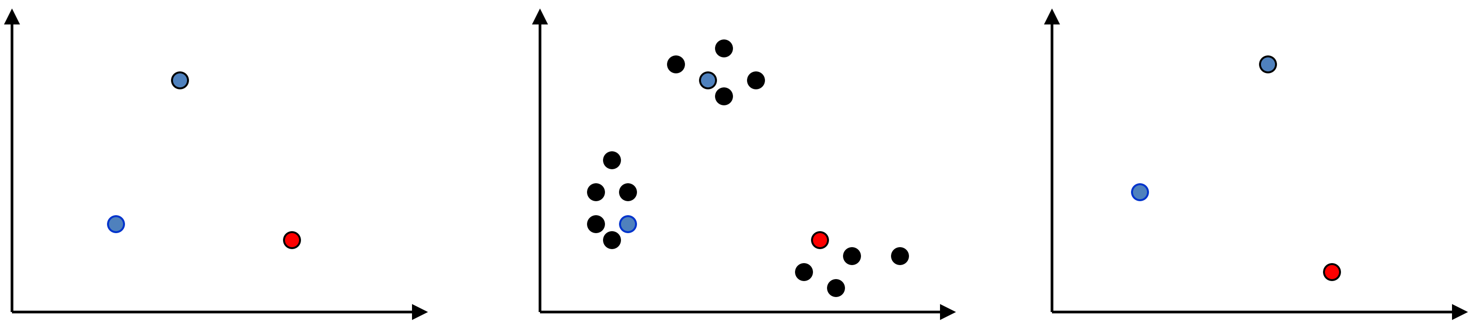
1. Set ic (iteration count) to 1

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.

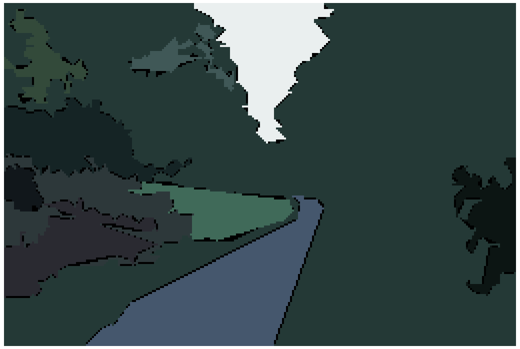
4. Increment ic by 1, update the means to get $m_1(ic),\ldots,m_K(ic)$.

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



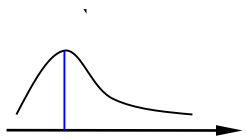
color clusters

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

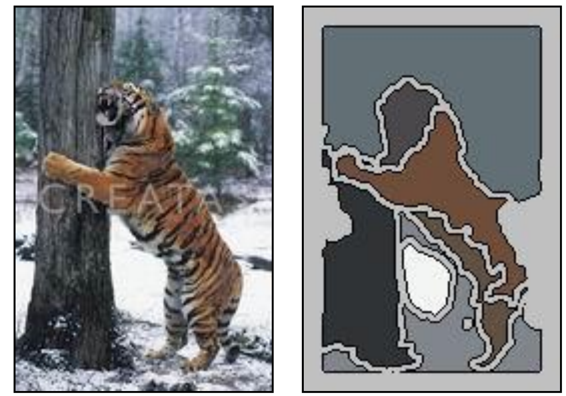
$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

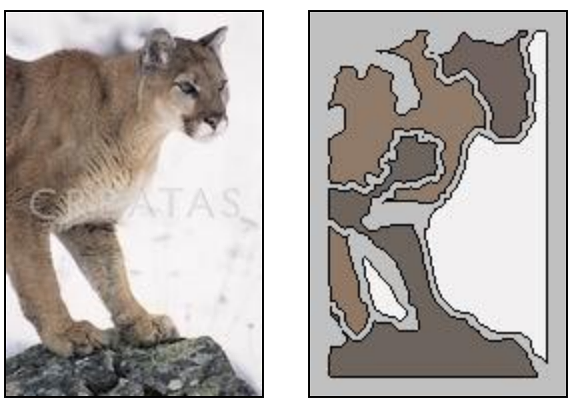
  - Maximization Step

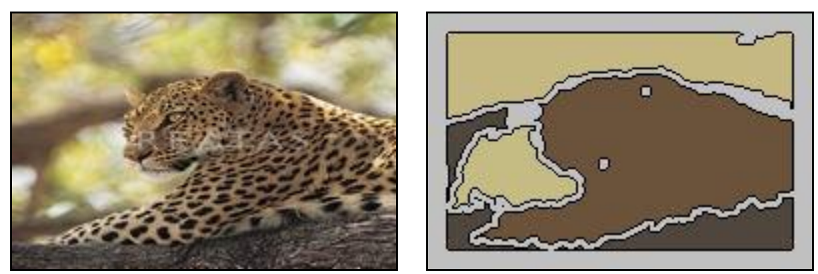
$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

EM Clustering using color and texture information at each pixel

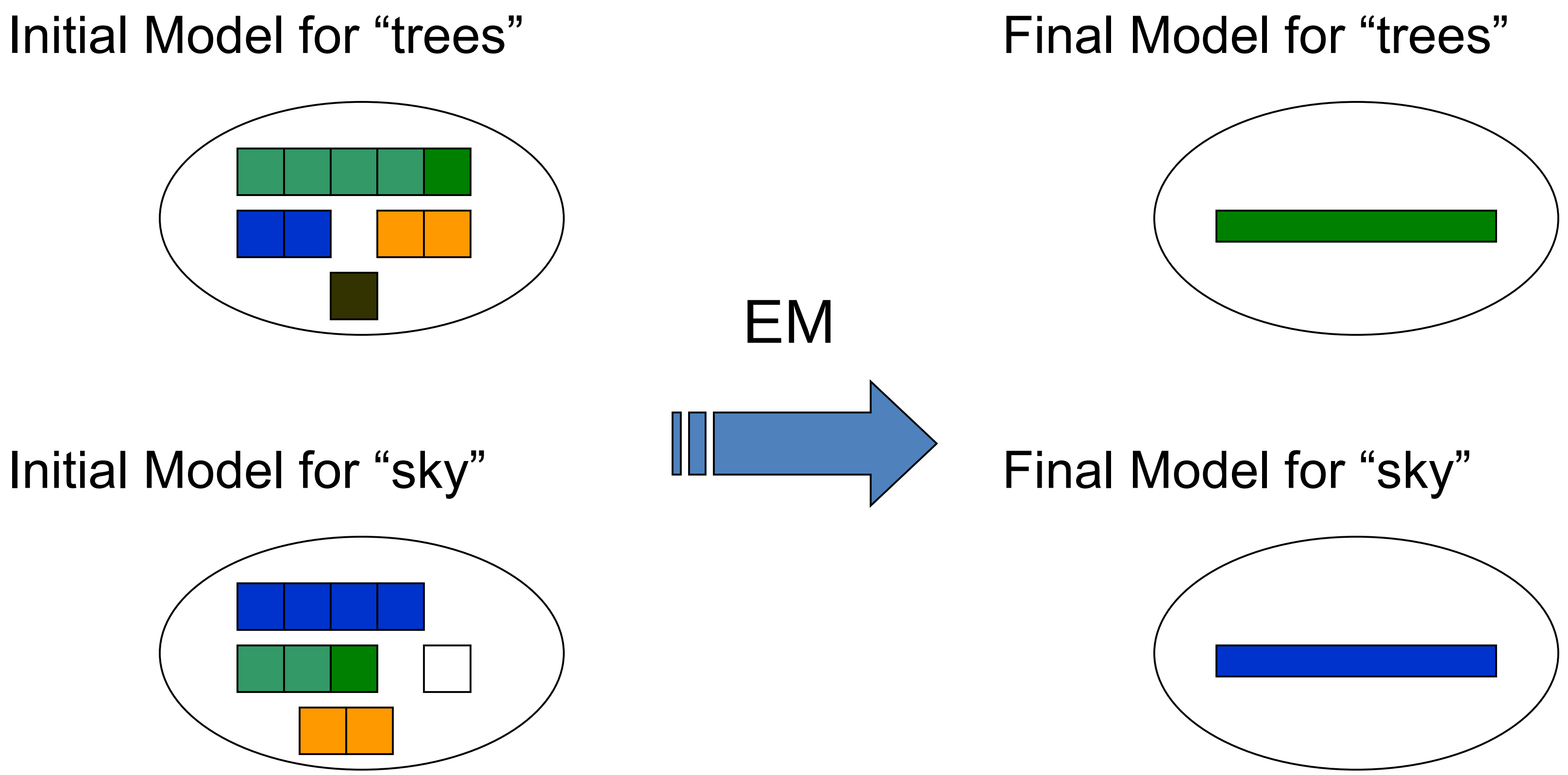
(from Blobworld)





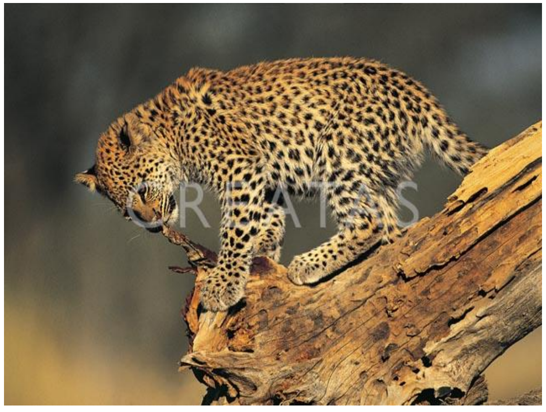


EM for Classification of Images in Terms of their Color Regions



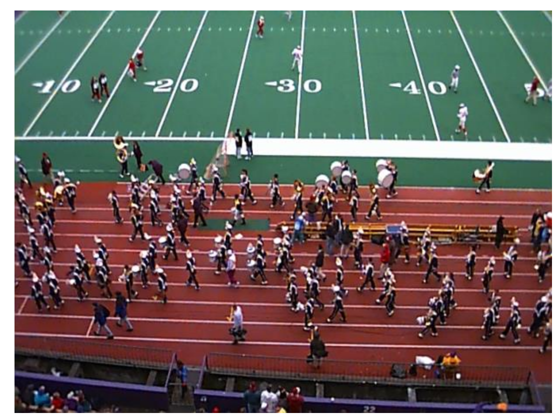
Sample Results

cheetah



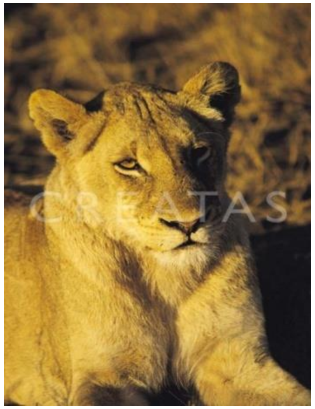
Sample Results (Cont.)

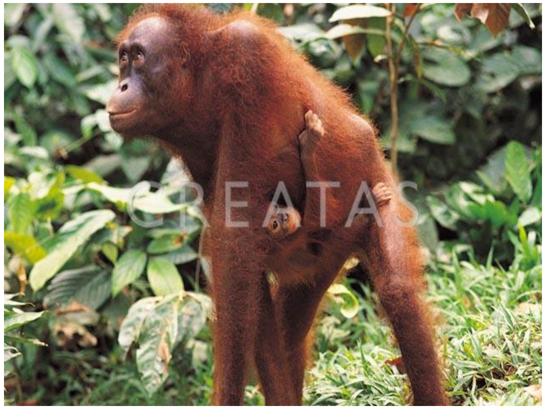
grass

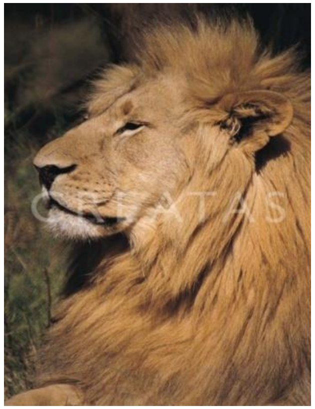


Sample Results (Cont.)

lion







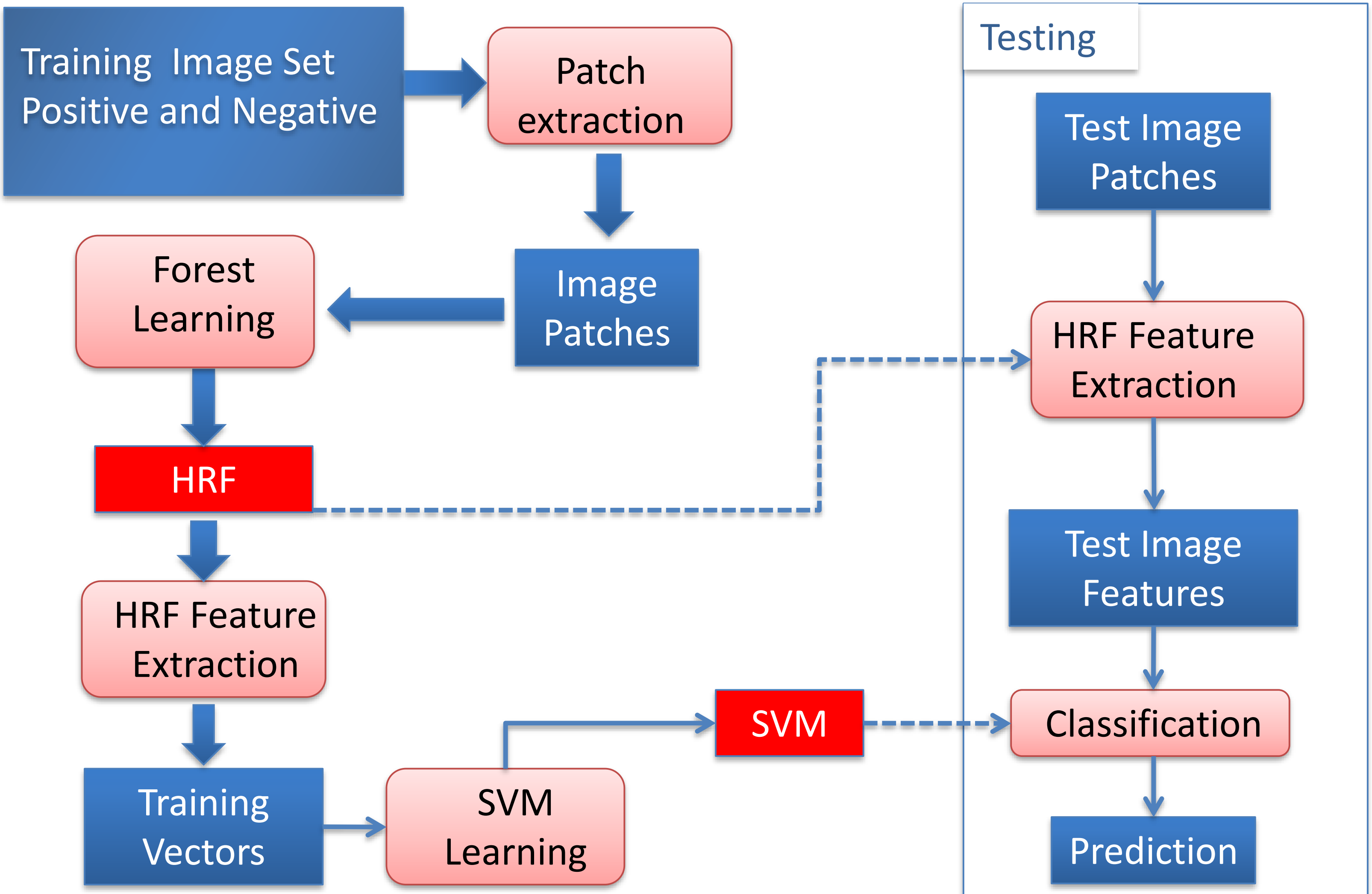
Haar Random Forest Features Combined with a Spatial Matching Kernel for Stonefly Species Identification

Natalia Larios*
Bilge Soran*
Linda Shapiro*
Gonzalo Martinez-Munoz^
Jeffrey Lin+
Tom Dietterich+

*University of Washington
+Oregon State University
^Universidad Autónoma de Madrid

Goal: to identify the species of insect specimens rapidly and accurately

Overview of our Classification Method



RESULTS:
Stonefly Identification: Classification Error [%]

Task	SET	CIELAB color	CIELAB+G	
Cal vs Dor	6.26	10.16	4.60	96.4% accuracy
Hes vs Iso	3.74	9.05	3.55	
Pte vs Swe	2.71	8.75	2.80	
Dor vs Hes	2.25	8.09	2.20	
Mos vs Pte	2.06	7.95	1.92	
Yor vs Zap	1.52	6.89	1.60	
Zap vs Cal	1.52	7.02	1.76	
Swe vs Yor	1.44	6.85	1.50	
Iso vs Mos	1.29	6.90	1.30	
Average	**2.53**	**7.96**	**2.25**	