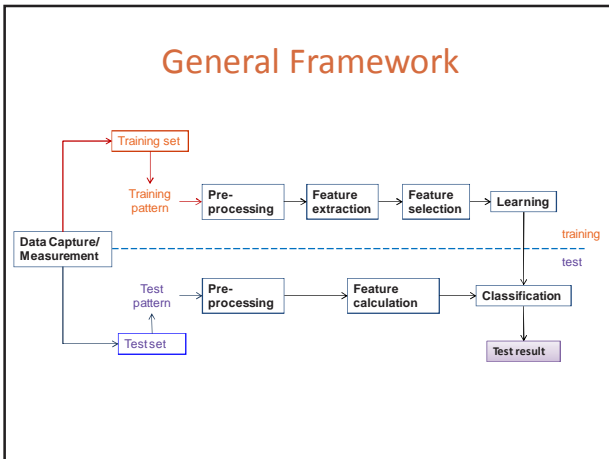


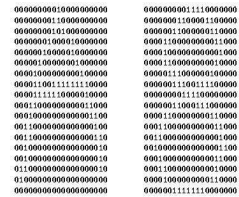
# Computer Vision Week 5

## An Introduction to Pattern Recognition

- ### Pattern Recognition Concepts
- Chapter 4: Shapiro and Stockman
  - Representation
    - How should objects be represented?
    - Real life concepts to numbers
  - Algorithms for recognition/matching
  - How should learning/training be done?



- ### Feature Vector Representation
- Features are patterns
  - $X = [x_1, x_2, \dots, x_n]$ 
    - each  $x_j$  is a real number
  - $x_j$  may be
    - object measurement
    - count of object parts
    - ...
  - Example: object rep.
    - [#holes, area, moments...]



- ### Some Terminology
- Classes (set of  $m$  known classes of objects)
    - might have known description for each
    - might have 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

### Performance issues 1

- In two-class problems, you can talk about two types of error:
  - False positive, Type I error
    - i.e. false alarm
  - False negative, Type II error
    - i.e. miss, false rejection

Actual	Decision	Error? Type?
Apple	Apple	True positive (no error)
Not an apple	Apple	False positive (error)
Apple	Not an apple	False negative (error)
Not an apple	Not an apple	True negative (no error)

		Actual Class	
		True	False
Predicted Class	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True negative (TN)

True Positive Rate =  $TPR = \frac{TP}{TP + FN}$

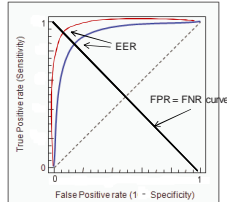
False Positive Rate =  $FPR = \frac{FP}{FP + TN}$

False Negative Rate =  $FNR = \frac{FN}{TP + FN}$

$FNR = 1 - TPR$

## Receiver Operating Characteristic (ROC) Curve

- To compare the performance of systems
- True positive rate (TPR) vs false positive rate (FPR)
  - Sensitivity vs (1-specificity)
  - Correct detection rate vs. false alarm rate
  - As TPR↑, FPR↑ (in general)



- Quality analysis
  - Area under the curve (AUC)
    - Higher AUC is better
  - Equal error rate (EER)
    - FPR = FNR = 1 - TPR
    - Lower EER is better

## Performance issues 2

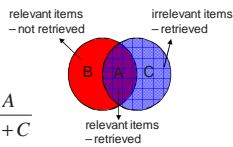
- In multi-class problems, you can talk about:
  - Confusion matrix

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
'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	1	96	0

## Performance issues 3

- In a matching/retrieval framework, we can talk about
  - Precision
  - Recall



$$\text{Precision} = \frac{\text{\# relevant objects retrieved}}{\text{total \# retrieved objects}} = \frac{A}{A + C}$$

$$\text{Recall} = \frac{\text{\# relevant objects retrieved}}{\text{total \# relevant objects}} = \frac{A}{A + B}$$

## Pattern Recognition Models\*

Approach	Representation	Recognition Function	Typical Criterion
Template matching	Samples, pixels, curves	Correlation, distance measure	Classification error
Statistical	Features	Discriminant Function	Classification error
Syntactic or structural	Primitives	Rules, Grammar	Acceptance error

\*Jain, Duin, Mao, IEEE PAMI, 2000

## Template Matching Approach

- Aim
  - Find small parts of an image which match a template image
- Demo
  - Template matching by a model
    - [http://bigwww.epfl.ch/demo/templatematching/tm\\_correlation/demo.html](http://bigwww.epfl.ch/demo/templatematching/tm_correlation/demo.html)
  - Template matching by a mask / kernel
    - [http://bigwww.epfl.ch/demo/templatematching/tm\\_kernel33/demo.html](http://bigwww.epfl.ch/demo/templatematching/tm_kernel33/demo.html)

## Matching by correlation

- Find matches of a subimage  $w(x,y)$  within an image  $f(x,y)$
- The correlation between  $f$  and  $w$ :

$$c(s,t) = \sum_{x \in W} \sum_{y \in W} f(x,y)w(x-s,w-t)$$

- Correlation coefficient:

$$\gamma(s,t) = \frac{\sum_{x \in W} \sum_{y \in W} [f(x,y) - \bar{f}(x,y)][w(x-s,w-t) - \bar{w}]}{\left\{ \sum_{x \in W} \sum_{y \in W} [f(x,y) - \bar{f}(x,y)]^2 \sum_{x \in W} \sum_{y \in W} [w(x-s,w-t) - \bar{w}]^2 \right\}^{1/2}}$$

### Statistical Approach

- Class conditional densities,  $P(x|w_i)$ 
  - Probability of observation  $x$  given the class/model  $w_i$
- Parametric
  - Model structure is determined a priori, based on a distribution
- Nonparametric
  - Model structure is determined from data, distribution free methods

### Bayesian decision-making

Thomas Bayes (1702-1761)

- Bayes' Theorem:
 

What you know about a parameter  $\theta$  after the data  $D$  arrive is what you knew before about  $\theta$  and what the data  $D$  told you.

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

Posterior =  $\frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$

### Bayesian decision-making

- Problem
  - Observations:  $x = [x_1, x_2, \dots, x_N]$
  - Classes / models:  $W = [w_1, w_2, \dots, w_M]$
  - $P(w_i|x)$ ?
- Known (for all  $i$ )
  - Class conditional density, likelihood:  $P(x|w_i)$
  - Prior probability:  $P(w_i)$
  - Unconditional distribution:  $P(x)$  ???
- Use Bayes rule to calculate the posterior,  $P(w_i|x)$ 
  - Assumption: All classes are independent
$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{P(x)} = \frac{P(x|w_i)P(w_i)}{\sum_{j=1}^M P(x|w_j)P(w_j)}$$

$$P(w_i|x) \approx P(x|w_i)P(w_i)$$

### Discriminant functions & Decision rules

- M discriminant functions
  - $d_1(x), d_2(x), \dots, d_M(x)$
- Decision rule
  - $x$  belongs to class  $i$  if  $d_i(x) > d_j(x)$  for all  $j \neq i$
- Decision boundary
  - $d_i(x) - d_j(x) = 0$
- i.e. Bayes classifier
  - $d_i(x) = P(w_i|x)$

### Unsupervised Learning

- The samples presented to the system are not pre-classified in some cases
  - No class labels
- One must find "natural" pattern classes first
  - Clustering
- Divide pattern space into K disjoint regions  $R_i$  such that
  - $R_i$  are mutually exclusive
  - $\cup R_i = \text{pattern space}$

### K-means clustering algorithm

- Given: data points  $\{x_1, \dots, x_N\}$  & number of clusters, K
  - Choose K initial cluster centers, for step 1
    - $c_1(1), \dots, c_k(1)$ , according to an initialization rule
  - At each step  $t$ ,
    - Assign the sample  $x_i$  to cluster  $m$  if  $\|x_i - c_m(t)\| < \|x_i - c_n(t)\| \quad \forall i \in [1, N], \forall n \neq m$
    - Determine new cluster centers by:
 
$$c_m(t+1) = \frac{1}{N_m} \sum_{x \in C_m(t)} x$$
  - Repeat 2 until convergence

[K-means demo](#)

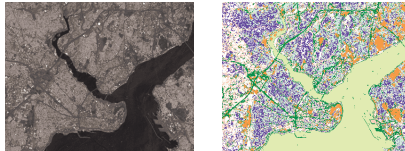
## Sample applications K-means

- Image segmentation

- Object detection



- Urban texture classification



## Parametric vs Nonparametric Methods

### Parametric methods

- Assumption
  - The data comes from a known model/distribution
- Advantage:
  - Only try to estimate the parameters of the model
  - i.e. Gaussian assumption, estimate only mean and the variance of each class
- Disadvantage
  - Assumption may not hold

### Nonparametric methods

- Assumption
  - Similar inputs have similar outputs
- Advantage
  - Non need for a model assumption
- Disadvantage
  - All training instances should be stored and computed while searching (O(N) memory and computation requirement)

## Supervised Learning

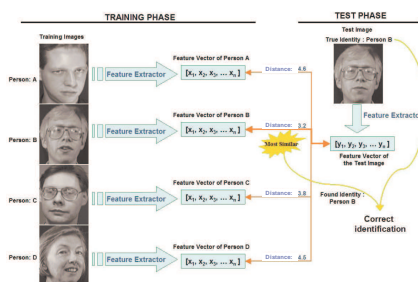
- Class labels of the samples are present
- Non-Parametric methods
  - e.g. Nearest neighbor classifier
- Parametric methods
  - e.g. Nearest mean classifier

## Nearest Neighbor Classifier

- Classify a sample by looking at its neighbors
  - Find the k nearest neighbors of the test sample
  - Assign the class most common amongst its k nearest neighbors,  $k > 0$ 
    - $k = 1$ : the object is simply assigned to the class of its nearest neighbor.
    - In binary classification problems, use an odd number for k to avoid ties.
- [Nearest neighbor demo](#)

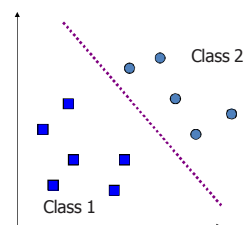
## Sample Application Nearest Neighbor Classifier

- Face recognition (1 nearest neighbor)



## Support Vector Machines

- Consider a two-class, linearly separable classification problem
- What is a good decision boundary?
  - Many decision boundaries!
  - Are all decision boundaries equally good?



### Example Decision Boundaries

→The decision boundary should be as far away from the data of both classes as possible!!!

### Large-margin Decision Boundary

- Maximize the margin,  $m = 2 / \|w\|$
- Distance between the origin and the line  $w^T x = k$  is  $k / \|w\|$

→The decision boundary can be found by solving the following constrained optimization problem:

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

$$\text{subject to } y_i (w^T x_i + b) \geq 1 \quad \forall i$$

### Non-linear case: Transforming the Data

Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

### Kernel Functions

- Informal definition:
  - A kernel function can be seen as a “similarity measure” between the objects
- Example Kernel functions:
  - Polynomial kernel with degree  $d$ :
 
$$K(x, y) = (x^T y + 1)^d$$
  - Radial basis function kernel with width  $\sigma$ :
 
$$K(x, y) = \exp(-\|x - y\|^2 / (2\sigma^2))$$
  - Sigmoid with parameter  $\kappa$  and  $\theta$ :
 
$$K(x, y) = \tanh(\kappa x^T y + \theta)$$

### SVMs

- Lesson learnt:
 

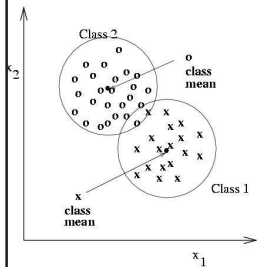
“a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space”
- Standard linear algorithms can be generalized to its non-linear version by going to the feature space

### Nearest mean classifier

1. For each class,  $c$ 
  - compute the mean (or prototype)  $m_c$
2. Compute  $d_c(x) = \|x - m_c\|$
3. Assign  $x$  to class  $c$  if  $d_c(x)$  is smallest

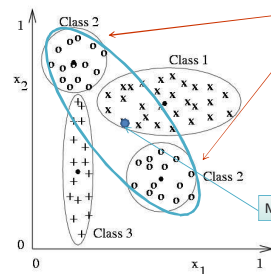
NOTE: minimizing  $\|x - m_c\|$  is equivalent to maximizing  $2x^T m_c - m_c^T m_c$

### Classification using nearest class mean



- Compute the Euclidean distance
  - between feature vector  $x$  and the mean of each class,  $m_c$ .
- Choose closest class, if close enough
  - reject otherwise

### Nearest mean with complex data structure

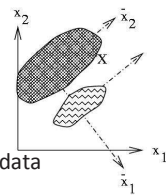


- Class 2 has two modes
  - might yield poor results with the standard, single mode approach
  - If modes are detected, two subclass mean vectors can be used

Mean of class 2 if single mode is assumed

### Unscaled vs scaled nearest mean classification

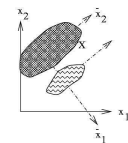
- Object X is equidistant from each class mean
  - Scale the distance with the spread (std. dev.) of each class, along each dimension
  - $d_c(x) = ||x - m_c|| = \sqrt{\sum_{i=1}^n ((x^i - m_c^i) / \sigma_c^i)^2}$
  - With scaling X is closer to left distribution



- Coordinate axes not natural for this data
  - 1D discrimination possible with PCA

### PCA- Principal Component Analysis

- Goal:
  - Find the dimensions of the data that explains the variance the most
- Methodology:
  - Find the covariance matrix of the data:  $C = E[(x-m)(x-m)^T]$
  - Find the eigenvalues ( $\lambda$ ) and eigenvectors ( $e$ ) of C:



$$C e = \lambda e$$

- The eigenvector corresponding to the largest eigenvalue is the principal component
- The matrix of ordered eigenvectors rotates the data so that the axes correspond to the eigenvalues and the first axis is the principal axis.

### PCA for feature extraction and selection

- Use PCA to reduce the number of features
- Project the data into the first K eigenvectors of the covariance matrix



In face recognition, this technique is called "Eigenfaces"

### Eigenfaces-Aim

- Aim:
  - extract the relevant information
  - perform optimal encoding
  - compare one face encoding with a database of faces encoded similarly
- How?
  - treat the faces as points
  - find the principal components of this distribution, i.e. the eigenvectors of the covariance matrix

## EigenFaces

- Eigenvectors are sort of ghostly faces: **eigenfaces**
- Represent faces as sum of these eigenfaces.
- Weights are found by projecting the face onto each eigenface.



## Methodology

- **Training**
  - Select a training set of **M** images
  - Calculate the eigenfaces, keep highest **M\***
  - Calculate the feature vectors by projecting the images onto the face space
- **Recognition:**
  - Find the feature vector of the new image
  - Use the Euclidean distance to find if this is a face image, of known or unknown person.

## Computing the Eigenfaces

- Find the mean image of the training set (**X**)
  - If size of the image is  $N \times N$ , size of **X** is  $N^2 \times 1$
- Subtract mean from each image, **I**
  - $D_i = [I_i - X]_{(N^2 \times 1)}$ ,  $i = 1 \dots M$
- Form **A** matrix
  - $A = [D_1 D_2 \dots D_M]_{(N^2 \times M)}$
- Form covariance matrix
  - $C = [AA^T]_{(N^2 \times N^2)} \Rightarrow$  yields  $N^2$  eigenvectors!!



## Computing Eigenfaces Cont'd

- $N^2$  is too big!!
- Instead, use  $[A^T A]_{M \times M}$ 
  - $A^T A e = \lambda e$  (1)  $\Rightarrow$  yields  $M$  eigenvectors
  - $AA^T A e = \lambda A e$  (2)
  - Substitute  $C = AA^T$ 
    - $C A e = \lambda A e$ ,
    - where  $A e$  is the eigenvectors of  $C$
- So, to find the eigenvectors of  $C$ , compute  $e$  from (1) & multiply them by  $A$

Note the original  
eigenvector calculation:  
 $\bar{C} e = \bar{\lambda} e$

## Recognition

- When a new face comes in,
  - find its difference vector,
  - project this vector onto the face space,
  - compare the weights to those of known individuals
  - threshold this distance to classify the image as recognized or not recognized

## Fisher's Linear Discriminant

- Uses the class discriminatory information
- Seeks to find directions along which the classes are best separated.
- Aim is to maximize the ratio of the between-class scatter and the within-class scatter
  - Linear Discriminant Analysis - LDA
- Known as *FisherFaces* in face recognition literature

### Methodology-FLD

- Between-class scatter matrix is

$$S_B = \sum_{i=1}^C N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

- Within-class scatter matrix is

$$S_W = \sum_{i=1}^C \sum_{x_k \in X_i} N_i (x_k - \mu_i)(x_k - \mu_i)^T$$

- C : number of classes
- $\mu_i$  : mean image of class  $X_i$ ,
- $N_i$  : number of samples in class  $X_i$

### Methodology-FLD

- Find  $W_{opt}$  which maximizes

$$W_{opt} = \arg \max_W \frac{W^T S_B W}{W^T S_W W} = [w_1 w_2 \dots w_M]$$

•  $S_W^{-1} S_B$  has rank C-1, at most  
 -> at most C-1 eigenvectors  
 corresponding to C-1 non-zero  
 eigenvalues ( $M < C-1$ )

- $w_i$  : set of **generalized eigenvectors** of  $S_B$  and  $S_W$ ,  $i=1 \dots M$

$$S_B w_i = \lambda_i S_W w_i$$

$S_W^{-1} S_B w_i = \lambda_i w_i$  -> eigenvectors associated to largest eigenvalues of  $S_W^{-1} S_B$

$S_B^{-1} S_W w_i = \frac{1}{\lambda_i} w_i$  -> eigenvectors associated to smallest eigenvalues of  $S_B^{-1} S_W$

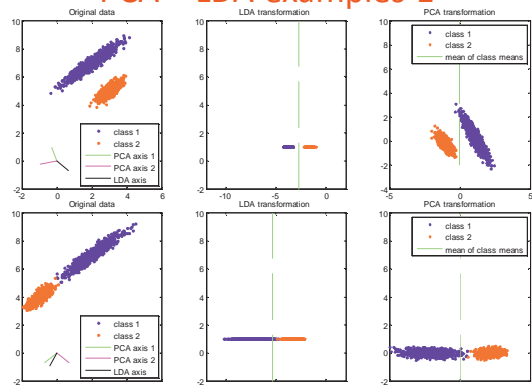
- $S_W$  and  $S_B$  has rank N-C, at most where as their dimensionality is  $V \times V$
- V: the number of dimensions in the feature vector
- Singular when  $N-C < V$
- What happens if  $S_W$  and  $S_B$  are both singular and non-invertible?

### FLD Singularity

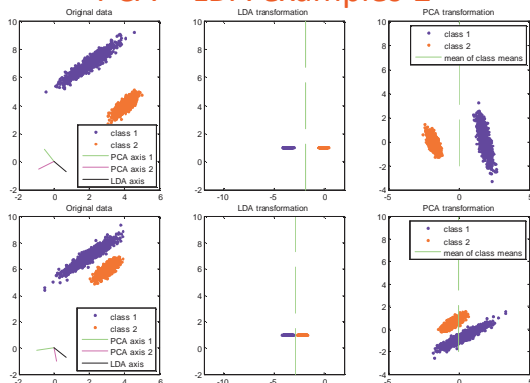
- Apply PCA prior to FLD
  - Reduce the dimensionality of the original data to K s.t.
    - $K \leq N-C$
  - $[x_1, \dots, x_V] \rightarrow \text{PCA} \rightarrow [y_1, \dots, y_K]$
- Apply FLD to the new data
  - $S_W$  and  $S_B$  will now be non-singular
  - Further reduce the dimensionality to M

$$[y_1, \dots, y_K] \rightarrow \text{FLD} \rightarrow [z_1, \dots, z_M] \quad M \leq C-1$$

### PCA – LDA examples 1



### PCA – LDA examples 2



### Temporal modeling

- Modeling low level dynamics
  - Kalman filter, particle filter, etc...
- Modeling high level dynamics
  - Finite state machines -- for simple motion models
  - Rule based approaches -- hard to obtain the rule base
  - **Temporal Templates** -- in image sequences
  - **Markov Models**
    - HMMs, Dynamic Bayesian Networks, IOHMMs, ...
  - Time Delay Neural Networks

### Temporal Templates

- Each gesture is modeled by a sequence of representative images, called temporal templates.
- [Bobick and Davis, 96] proposed
  - Motion History Images (MHI)
  - Motion Energy Images (MEI).
  - MHI and MEI are accumulated images that are calculated over a temporal window.

D: binary image sequence indicating motion regions  
 $\tau$ : duration of the action

$$MEI_{\tau}(x, y, t) = \bigcup_{i=0}^{\tau-1} D(x, y, t - i)$$

$$MHI_{\tau}(x, y, t) = \begin{cases} \tau & \text{if } D(x, y, t) = 1 \\ \max(0, H_{\tau}(x, y, t - 1) - 1) & \text{otherwise} \end{cases}$$

### Motion detection

Assumption:

the gesturer and the background must be stationary while the hands, arms are in motion.

Original Image



- Calculate the difference image

$$D'(x, y, t) = B(x, y, t) - B(x, y, t + 1)$$



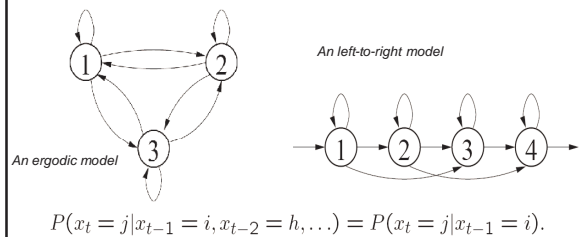
- Apply a threshold to reduce the noise

$$D(x, y, t) = \begin{cases} 0 & \text{for } |D'(x, y, t)| < T \\ D'(x, y, t) & \text{otherwise} \end{cases}$$



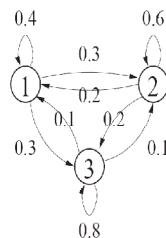
### Markov Models

- For a first-order discrete-time Markov chain, probability of state occupation depends only on the previous step:



### Weather Prediction Example

- State 1: rain
- State 2: cloud
- State 3: sun



State transition probabilities

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

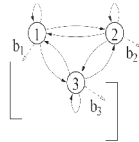
### Weather Prediction Calculation

Given today is sunny (i.e.,  $x_1 = 3$ ), what is the probability of "sun-sun-rain-cloud-cloud-sun" with model  $M$ ?

$$\begin{aligned} P(X|M) &= P(X = \{3, 3, 1, 2, 2, 3\} | M) \\ &= P(x_1 = 3) P(x_2 = 3 | x_1 = 3) \\ &\quad P(x_3 = 1 | x_2 = 3) P(x_4 = 2 | x_3 = 1) \\ &\quad P(x_5 = 2 | x_4 = 2) P(x_6 = 3 | x_5 = 2) \\ &= \pi_3 a_{33} a_{31} a_{12} a_{22} a_{23} \\ &= 1 \cdot (0.8)(0.1)(0.3)(0.6)(0.2) \\ &= 0.00288 \end{aligned}$$

### Hidden Markov Models

- States themselves are not observable
  - But in each state we observe an information related to the state
- Urn and balls example (Ferguson)
  - 3 urns
  - Black, red and white balls in each urn
  - States: urns
  - Observation: the color of the ball

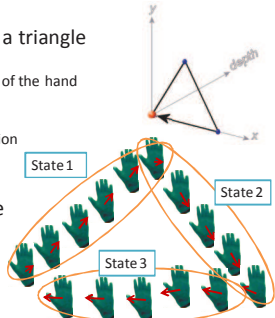


### Three basic problems of HMMs

1. Given a model, evaluate the probability of any given observation sequence
  - Forward-Backward Algorithm
2. Given a model and an observation sequence,  $O$ , find out the state sequence which has the highest probability of generating  $O$ 
  - Viterbi Algorithm
3. Given a training set of observation sequences, learn the model that maximizes the probability of generating the training set
  - Baum-Welch algorithm

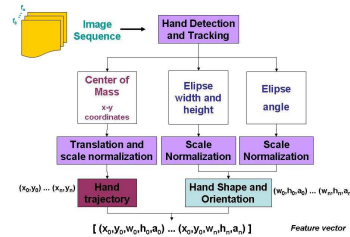
### Simple Example: Gesture Classification

- A gesture can be modeled as a first order markov chain
  - Consider a gesture to draw a triangle
    - Features (observations):
      - The direction of the velocity of the hand
      - Can be noisy due to
        - » Wrong tracking
        - » Wrong hand segmentation
        - » Variations of the user
  - Observations give related information about the state
    - Use a hidden Markov model
      - Left-to-right with 3 states



### Example: Gesture Classification with HMMs

- Apply preprocessing and feature extraction
  - To obtain gesture features



### Example: Gesture Classification with HMMs

- Train one HMM for each gesture (Baum-Welch)
- For a test gesture
  - Run Forward algorithm to obtain the likelihood of each HMM model
  - Select the class of the model with the maximum likelihood

