
COMP 435 Spring 2010

Review II

Outline

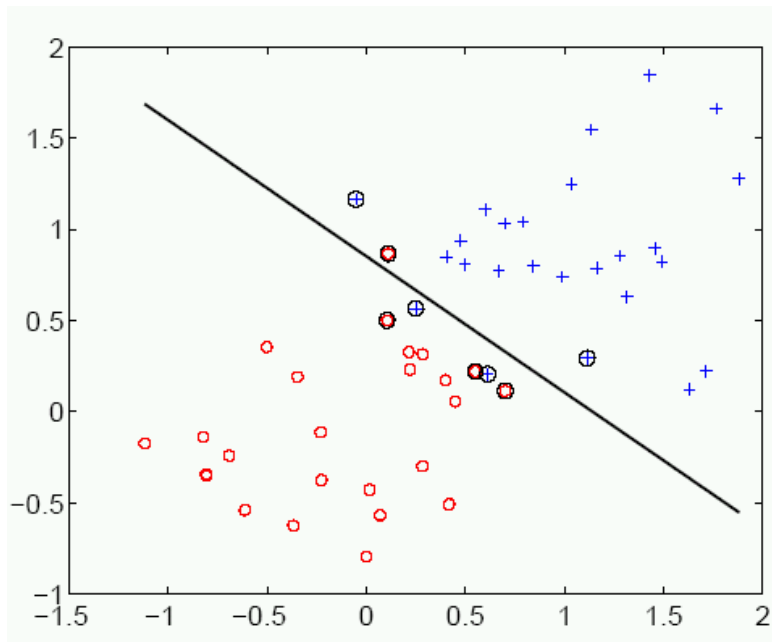
- **Pattern Recognition**
 - kNN classifier
 - » Editing
 - » Condensing
 - Bayesian classifier
 - » Prob modeling for classification
 - » Likelihood, Posterior
 - » Bayesian decision with Gaussian distribution
 - Support Vector Machine & Kernel Method
 - » SVM formulation & solution
 - » Kernel Method for SVM
 - » Kernel method for PCA

Outline

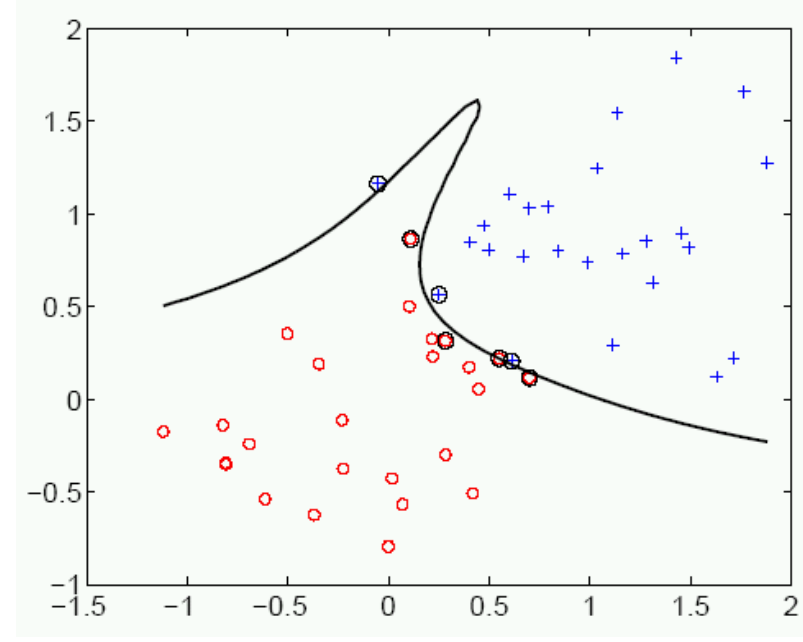
- **Biometric Techniques & Systems**
 - Face
 - » Detection - Boosting, Bootstrapping
 - » Recognition - Eigenface, Fisherface, Randomface
 - Fingerprint
 - » Singular point, Minutia
 - Palm
 - » PalmCode to encode directions at each pixel and matching by hamming distance, texture based
 - Iris & Retina
 - » IrisCode - texture feature based
 - Behavior & Multi Biometrics

Pattern Recognition

- **The problem:**
 - Given two (or more) class of data points, $\{x_k\}$, how to find a function $f(x)$ that separates the two classes ?



Linear decision boundary



Non-linear decision boundary

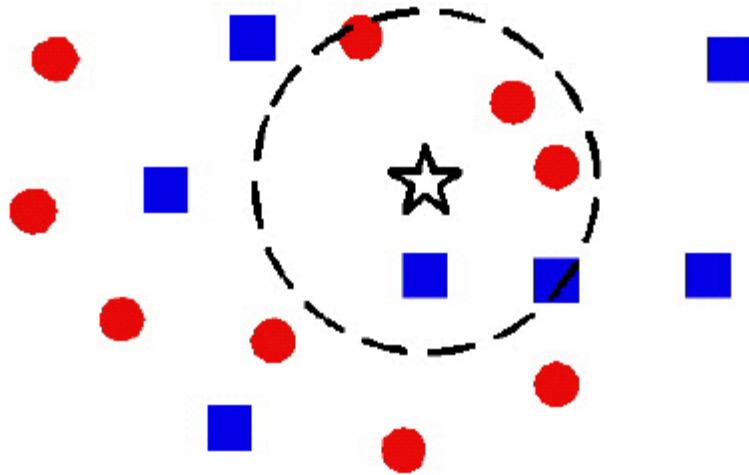
kNN

K-NN Classifier

- **K-Nearest Neighbor classifier rule:**

- ▶ The labelled dataset $\mathcal{D}_n = \{(\mathbf{x}_1, \theta_1), \dots, (\mathbf{x}_n, \theta_n)\}$
- ▶ where $\theta_i \in \{\omega_1, \dots, \omega_c\}$ is the class label for \mathbf{x}_i .
- ▶ For an input \mathbf{x} , find its k nearest neighbors
- ▶ The k -nearest-neighbor decision rule

$\mathbf{x} \rightarrow \{majority\ class\ label\ of\ the\ k\ nearest\ neighbors\}$

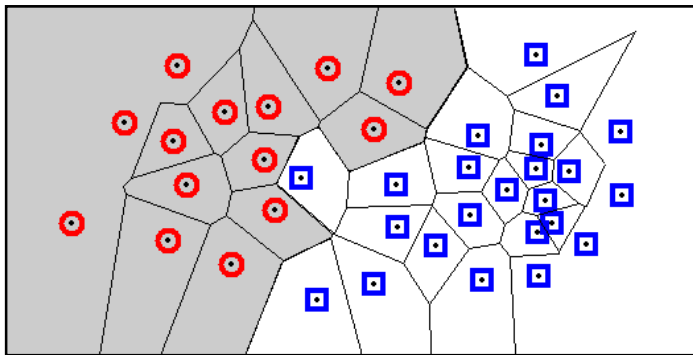


$$\theta(x) = \bullet$$

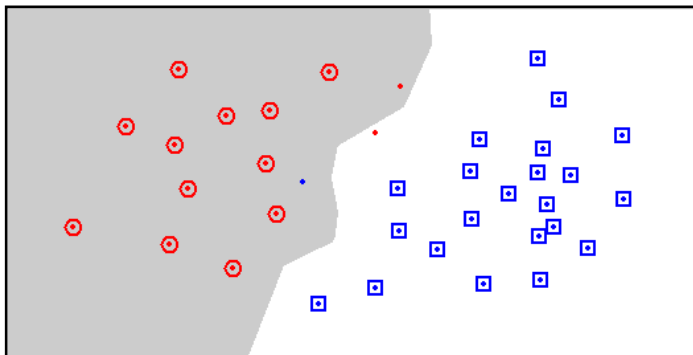
kNN Decision Boundary (non-linear)

- Dealing with kNN's complex non-linear boundary
 - Editing
 - Condensing

Earlier example

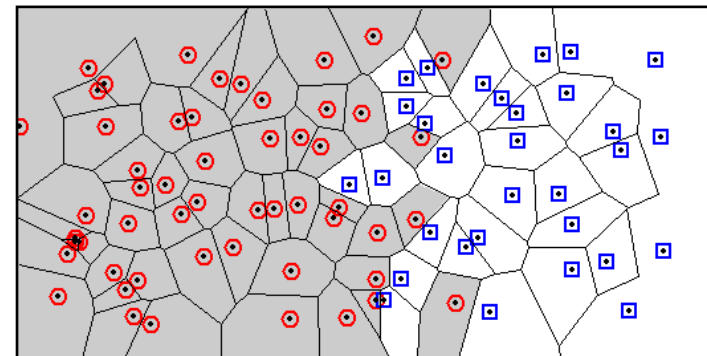


Original data

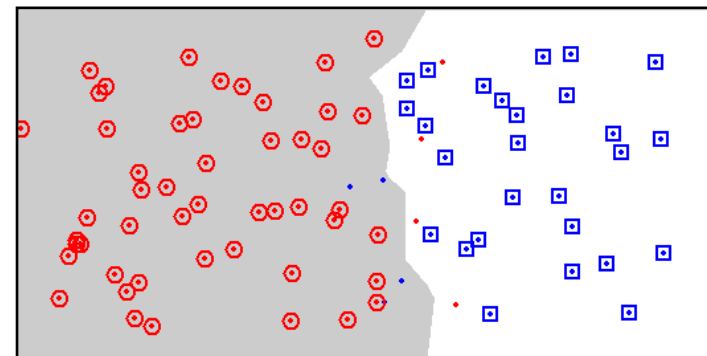


Wilson editing with $k=7$

Overlapping classes



Original data



Wilson editing with $k=7$

kNN classifier challenges

- **Large Data Set Size**
 - Complex decision boundaries
 - Access/Storage
- **Solutions:**
 - Editing:
 - » merge cells, simplify decision boundaries
 - Condensing:
 - » keep only data points that can influence decision boundaries
 - Indexing:
 - » Improve access, but not reduce storage size

Bayesian Decision

Bayesian Classifier with Gaussian Distribution Data

- Bayesian Decision:

- ▶ $p(\mathbf{x}|\omega_i)$ Likelihood
- ▶ $p(\omega_i)$ Prior
- ▶ $p(\omega_i|\mathbf{x})$ Posterior
- ▶ Bayes Rule

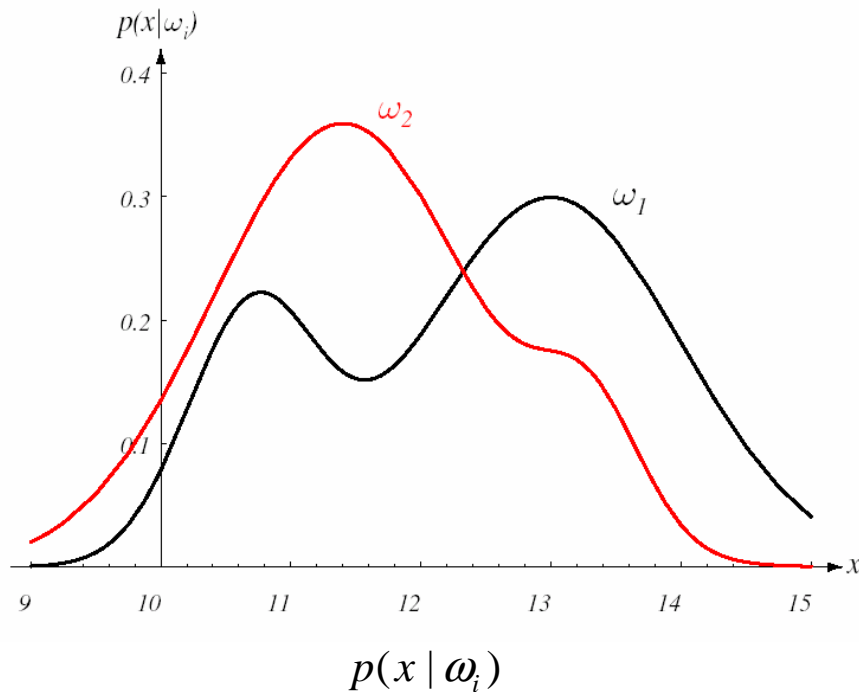
$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)}$$

- ▶ In other words

posterior \propto likelihood \times prior

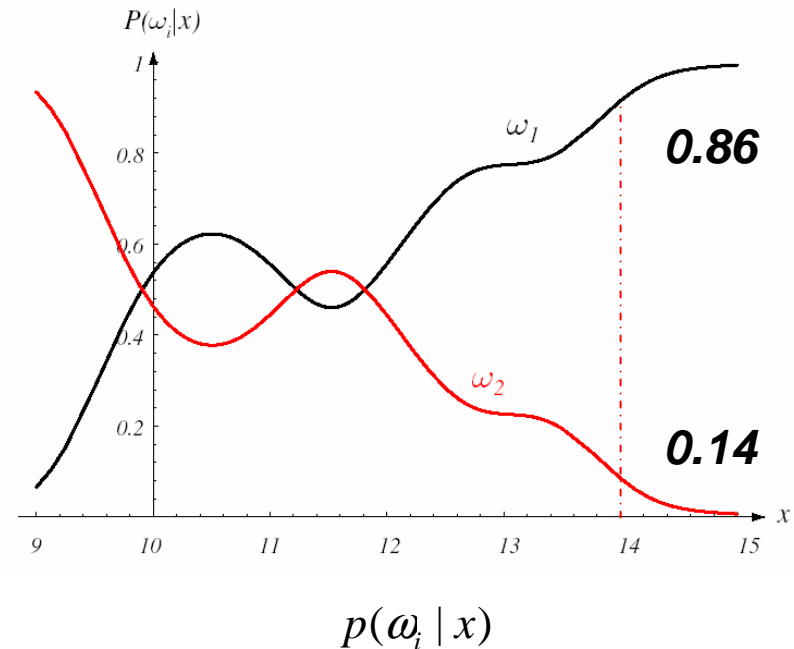
Example

Decision Boundary ?



**Given Likelihood functions
And Prior:**

$$p(\omega_1) = 1/3, \quad p(\omega_2) = 2/3$$

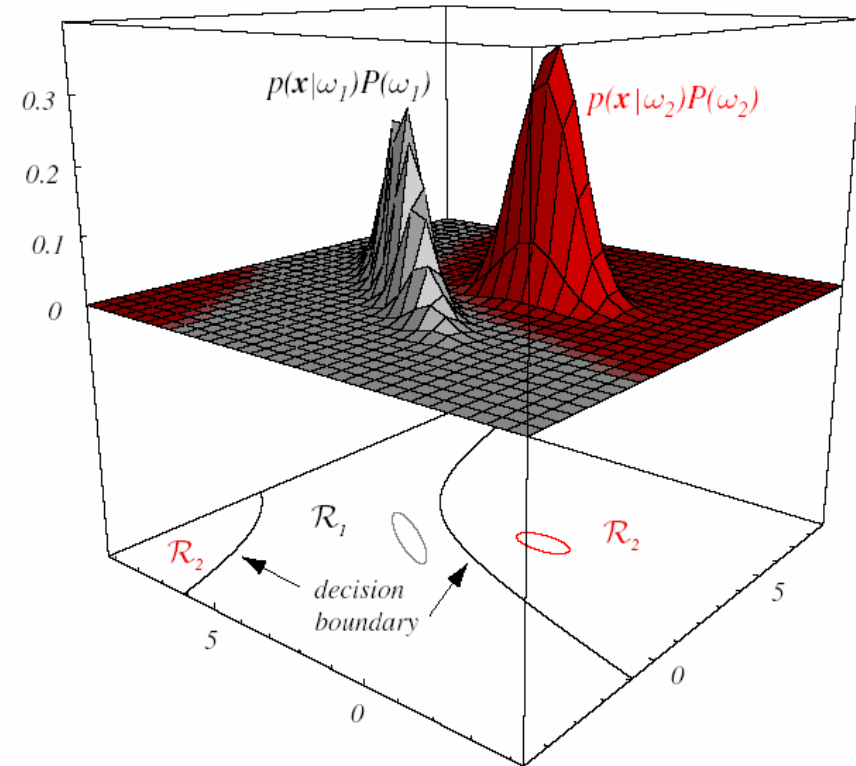


**Posterior functions
Should sum to 1**

**Given the evidence, the prob
Of belonging to certain class**

Bayesian Decision with Gaussian Data

- Apply to Gaussian Data:
 - General case, each class has arbitrary Gaussian distribution : decision boundary is quadratic/non-linear
 - When have identical Gaussian for each class: decision boundary is linear
- Matlab Implementation:
 - `[rec, err]=classify(x1, x0,y0, method)`
 - Method = 'linear', 'quadratic'
 - Training data: x0, y0
 - Test data: x1



SVM & Kernel Machine

- **Support Vector Machine**

- Given two class of (separable) data points,
- Find a linear decision boundary that creates the largest gap between two classes: $y=w*x+b$
- The solution is optimal in the sense that it minimizes structural risk in classification
- Numerically, it involves Lagrangian relaxation and solving the dual problem (not required to completely understand it at this time)

Support Vector Classifier

- The data: $\{x_i, y_i\}$, where $y_i = +1$, or -1 . $i = 1..N$
- A clean-separable hyperplane:
 $y_i[\langle w, x_i \rangle + b] \geq 1$, for $i = 1..N$
- Min distances (gap) of $\{x_i\}$ to hyperplane:

$$\begin{aligned}d(w, b) &= \min_{x_i: s.t. y_i = -1} \frac{|\langle w, x_i \rangle + b|}{\|w\|} + \min_{x_i: s.t. y_i = +1} \frac{|\langle w, x_i \rangle + b|}{\|w\|} \\&= \frac{1}{\|w\|} \left\{ \underbrace{\min_{x_i: s.t. y_i = -1} |\langle w, x_i \rangle + b|}_{=1} + \underbrace{\min_{x_i: s.t. y_i = +1} |\langle w, x_i \rangle + b|}_{=1} \right\} \\&= \frac{2}{\|w\|}\end{aligned}$$

Optimal Solution

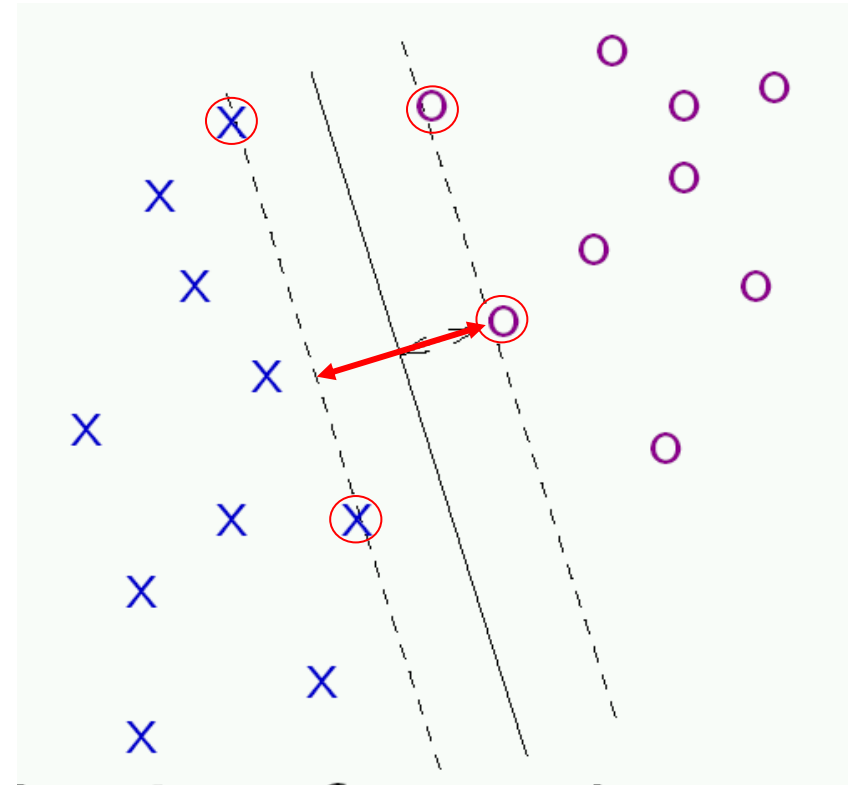
- The optimal hyperplane is given

by:

$$\left\{ \begin{array}{l} w^* = \sum_{j=1}^N \lambda_j^* y_j x_j \\ b^* = -(1/2) \langle w^*, x^+ + x^- \rangle \end{array} \right\}$$

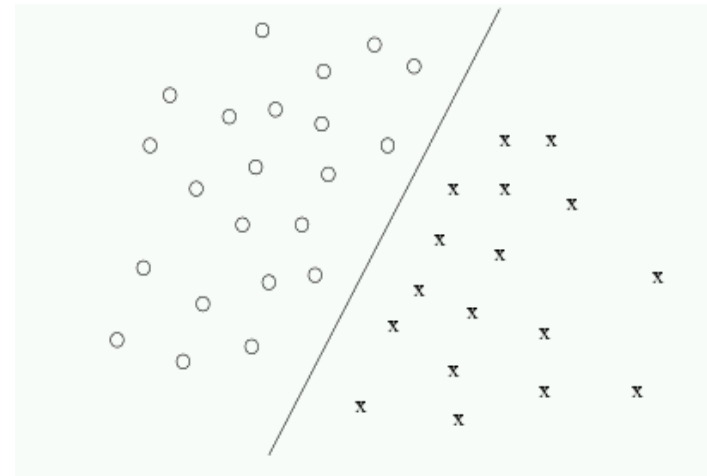
- Support vectors:

$$\{x_k^S \mid \lambda_k > 0\}$$

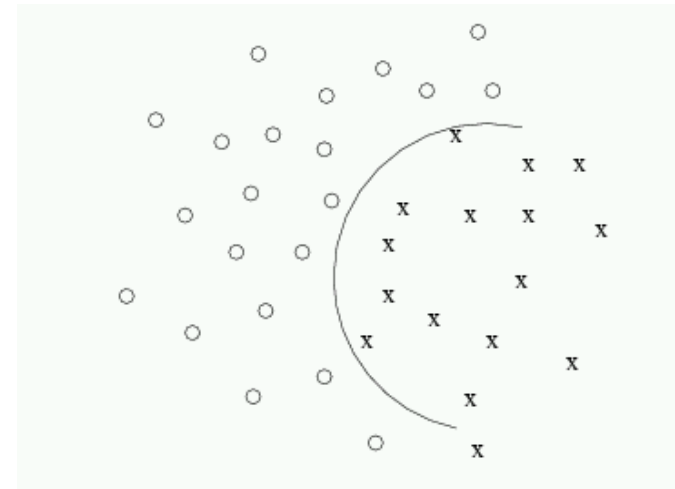


How to fit non-linear decision boundary ?

- Gives the feature space richer structure and geometry for better classification
- Dimension of Y is typically much higher than X , it's well known that the prob of linear separability grows with feature space dimension.
- How to fit into SVC framework ?



SVC in Y



SVC in X

Kernelized SVM (non-linear)

- The optimization problem depends only on inner product evaluation:

$$\lambda^* = \arg \max_{\lambda} \left\{ \sum_{i=1}^N \lambda_i - (1/2) \sum_{j=1}^N \sum_{k=1}^N \lambda_j \lambda_k y_j y_k \langle x_j, x_k \rangle \right\}$$

- Substitute the original space inner product with the kernel:

$$\lambda^* = \arg \max_{\lambda} \left\{ \sum_{i=1}^N \lambda_i - (1/2) \sum_{j=1}^N \sum_{k=1}^N \lambda_j \lambda_k y_j y_k K(x_j, x_k) \right\}$$

Kernel PCA

- The same kernel trick can be used to model data
- 2 ways to compute PCA:
 - P1: eigen decomposition of the scatter matrix: $S=XX^T$, eigen vectors are the basis functions of PCA
 - P2: PCA basis vector as linear combination of all data points, the weights are computed from eigen vectors of Gram matrix $G=X^TX$,
- Kernel PCA
 - Use the P2 approach
 - When computing the Gram matrix, replace $\langle x_j, x_k \rangle$ with $K(x_j, x_k)$

Computation Summary for KPCA

- **Computational steps:**

Learning KPCA from $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$:

1. calculating $\mathbf{m} = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k$
2. centering $\mathbf{D} = [\mathbf{x}_1 - \mathbf{m}, \dots, \mathbf{x}_N - \mathbf{m}] = [\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_n]$
3. calculating \mathbf{K} , where $\mathbf{K}_{ij} = K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$
4. eigenvalue decomposition

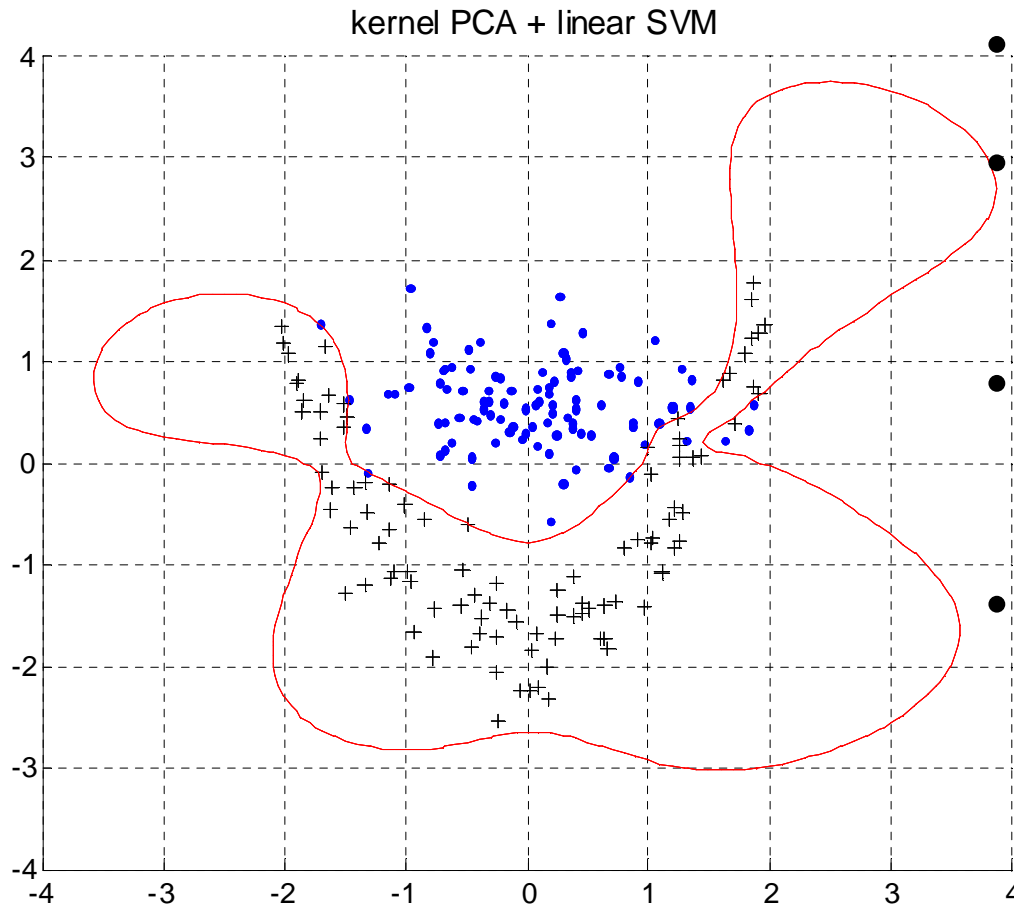
$$\mathbf{K} = \mathbf{U}^T \Sigma \mathbf{U}$$

5. sorting λ_i and \mathbf{a}_i

Note: The i -th kernel principal components for \mathbf{x} is

$$\mathbf{z} = \mathbf{a}_i^T K(\mathbf{Y}, \mathbf{x}) = \sum_{j=1}^n \alpha_{ij} K(\mathbf{x}_j, \mathbf{x})$$

Example: Kernel PCA + SVM



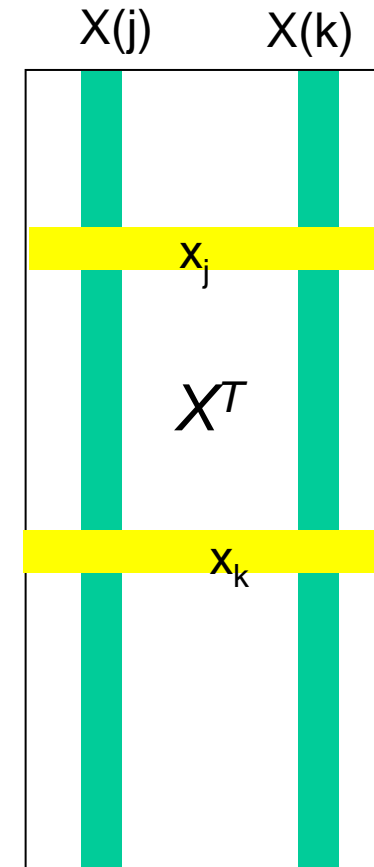
- 200 data points in \mathbb{R}^2
- Highly non linear boundaries between class
- Used a Gaussian kernel to do the kernel PCA
- SVM decision boundary mapped back to the original X space

Check out: [kernel_method.m](#) from the COMP435 web page

PCA vs KPCA Computing

- Given data set: $X = [x_1, x_2, \dots, x_n]$ in \mathbb{R}^d ,

- Remove the mean from X ,
- Gram matrix $G = X^T X$, i.e, $G_{j,k} = \langle x_j, x_k \rangle = x_j^T x_k$.
 - » Replace with kernel $k(x_j, x_k)$ if necessary
- Scatter matrix $S = X X^T$, i.e, $S_{j,k} = \text{Cov}(X(j), X(k))$, j and k are the component across all data points.



- PCA is solved by EVD of S
- kPCA is solved by EVD of G

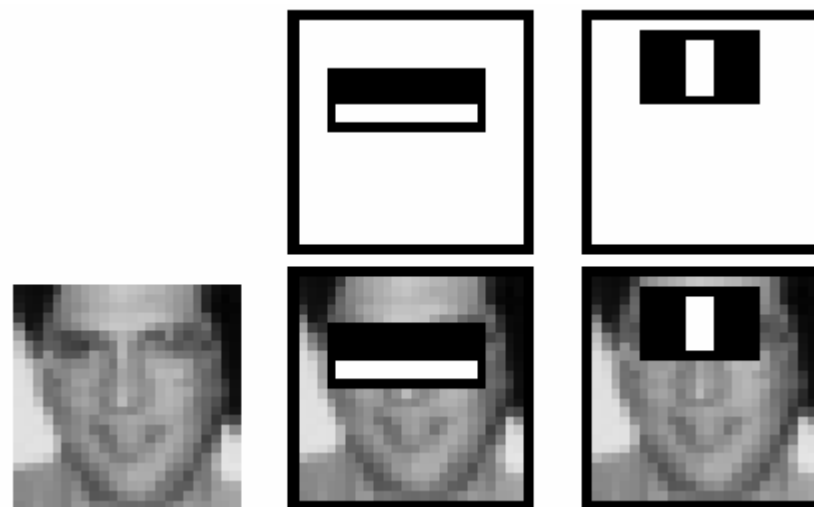
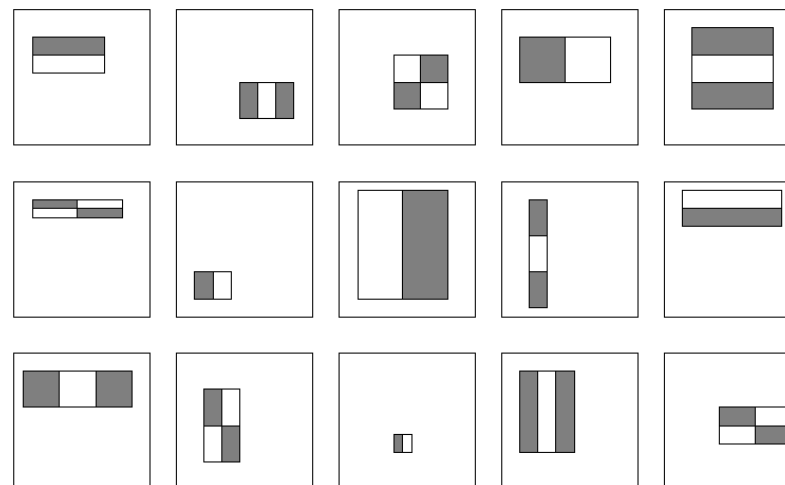
Summary for PCA, LDA, Kernel Method

- Given data set X , and Y in \mathbb{R}^d , Need to know how to compute:
 - Unsupervised: Gram Matrix, Scatter Matrix
 - Supervised: Between Class Scatter, Within Class Scatter
- Understand
 - How PCA basis is computed directly (P1)
 - How PCA basis is computed as linear combination of all data points (P2)
 - LDA formulation and computing

Biometric Systems

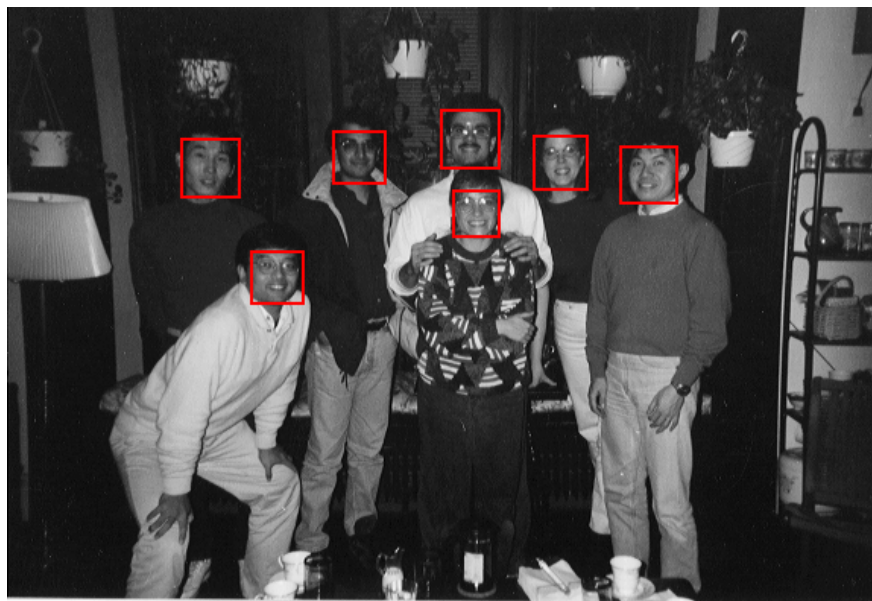
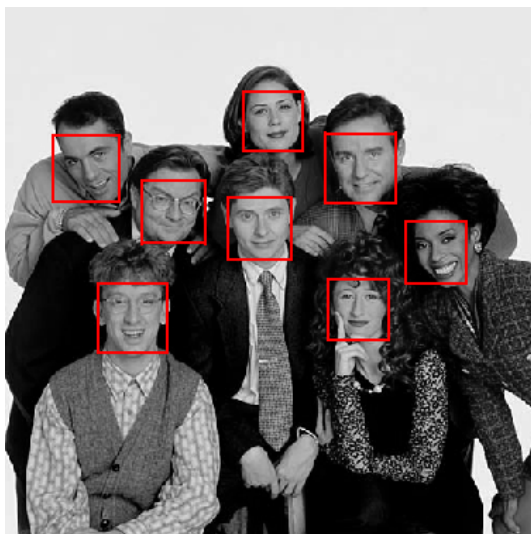
Local feature based Face Detection (Boosting)

- Boundaries between face and non-face patterns are quite complex to characterize.
- **Boosting**: Instead of growing a super horse, use a group of small horses to do the job:
 - Weak classifier: integral images
 - Iteratively training to find what classifiers are good.
 - Cascading the classifier to reduce run time complexity
 - Works well and being implemented in OpenCV



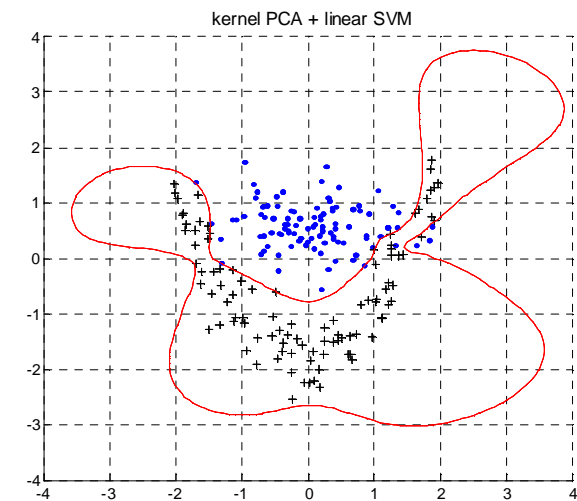
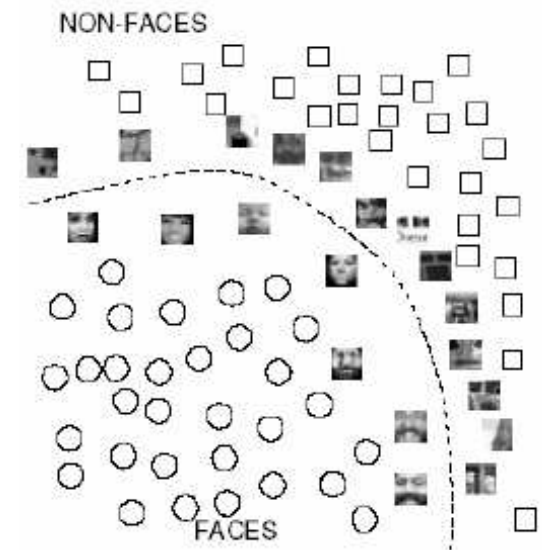
Appearance based Face Detection & Recognition

- **Appearance based solution**
 - Give an image pattern of say $w \times h$ pixels
 - Tell if
 - » Is this a face ?
 - » If a face, who s/he is ?



Solution Summaries

- Face Detection
 - Obtain good training samples close to the decision boundary (support vectors), eg, **Bootstrapping**
 - Subspace modeling and classification, e.g. kNN SVM
- Face Recognition
 - Fisherface:
 - classic approach, use PCA as first step to address the S_w singular issue.
 - Kernel Method:
 - Implicit non-linear mapping via kernel function
 - L1 Magic:
 - Solve for sparse representation $y = Ax$, where x is sparse



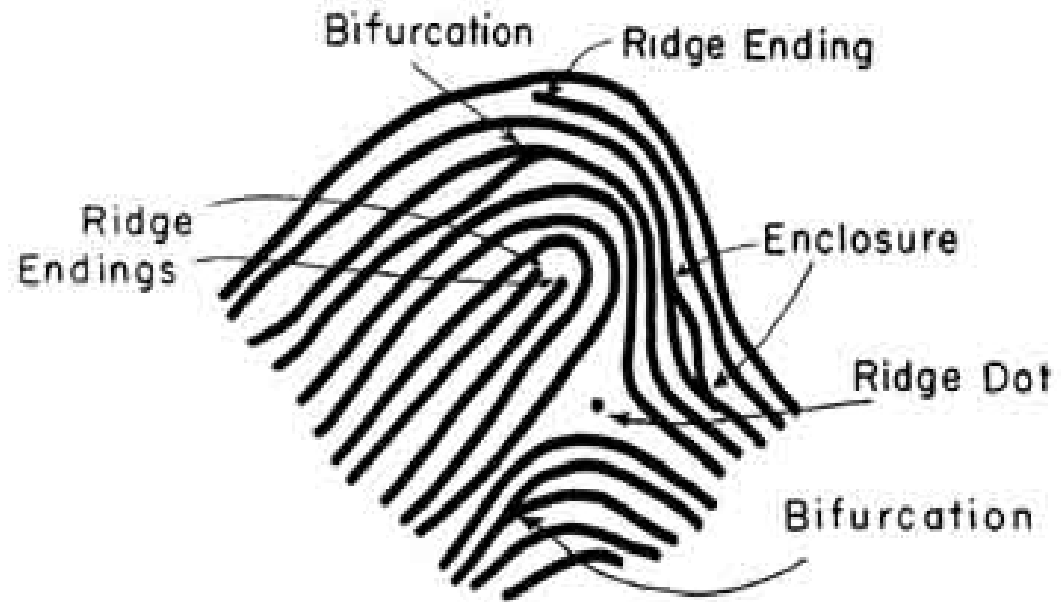
Fingerprint

- **Sensors:**
 - Optical, Solid state, Ultra-Sonic
- **Image processing:**
 - Detect and enhance edges
 - Thinning into single pixel lines
- **Features & Matching**
 - Minutia based:
 - » Location (x,y), type, and orientation
 - Matching minutia points

Minutia

- Known as minutia points which ridges end, fork and change
- Tiny, unique characteristics of fingerprint ridges that are used for positive identification
- Possible for one or more individuals to have identical global features but still have different and unique fingerprints because of the minutia points

- Types
 - Ridge ending
 - Ridge bifurcation
 - Ridge divergence
 - Dot or Island
 - Enclosure
 - Short Ridge



Minutia based Matching

- Given 2 sets of minutia:

$$\begin{aligned} \mathbf{T} &= \{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_m\}, & \mathbf{m}_i &= \{x_i, y_i, \theta_i\}, & i &= 1..m \\ \mathbf{I} &= \{\mathbf{m}'_1, \mathbf{m}'_2, \dots, \mathbf{m}'_n\}, & \mathbf{m}'_j &= \{x'_j, y'_j, \theta'_j\}, & j &= 1..n, \end{aligned}$$

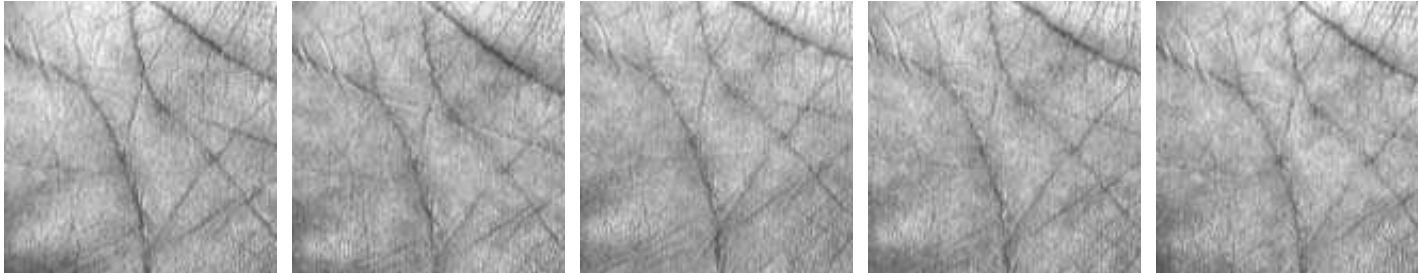
- Matching of minutia is thru location, type and orientation:

$$\begin{aligned} sd(\mathbf{m}'_j, \mathbf{m}_i) &= \sqrt{(x'_j - x_i)^2 + (y'_j - y_i)^2} \leq r_0, & \text{and} \\ dd(\mathbf{m}'_j, \mathbf{m}_i) &= \min(|\theta'_j - \theta_i|, 360^\circ - |\theta'_j - \theta_i|) \leq \theta_0. \end{aligned}$$

- While allow certain rotation and displacement tolerance:

$$\begin{aligned} \text{map}_{\Delta x, \Delta y, \theta}(\mathbf{m}'_j = \{x'_j, y'_j, \theta'_j\}) &= \mathbf{m}''_j = \{x''_j, y''_j, \theta'_j + \theta\}, \\ \begin{bmatrix} x''_j \\ y''_j \end{bmatrix} &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x'_j \\ y'_j \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \end{aligned}$$

Palm



The same person's palmprints



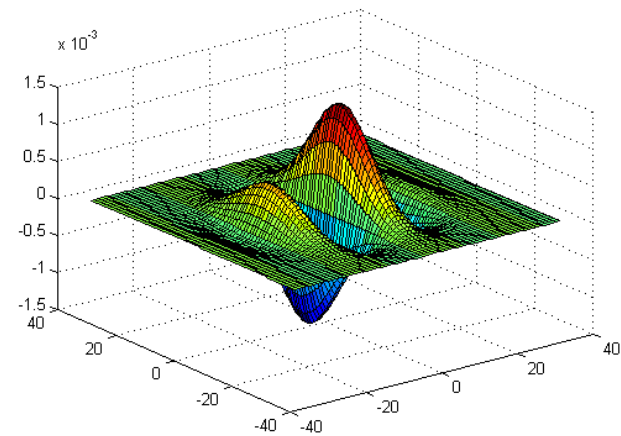
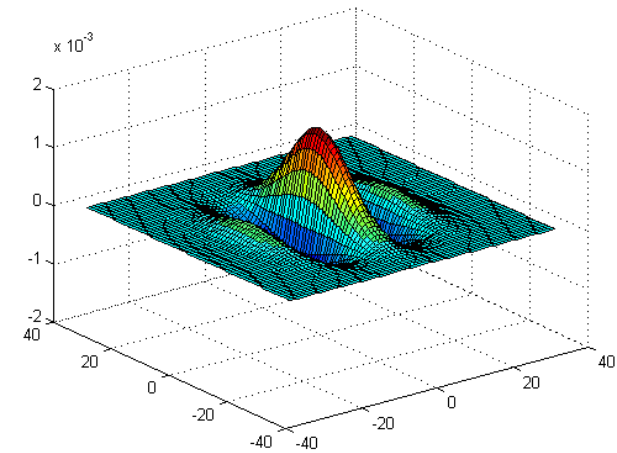
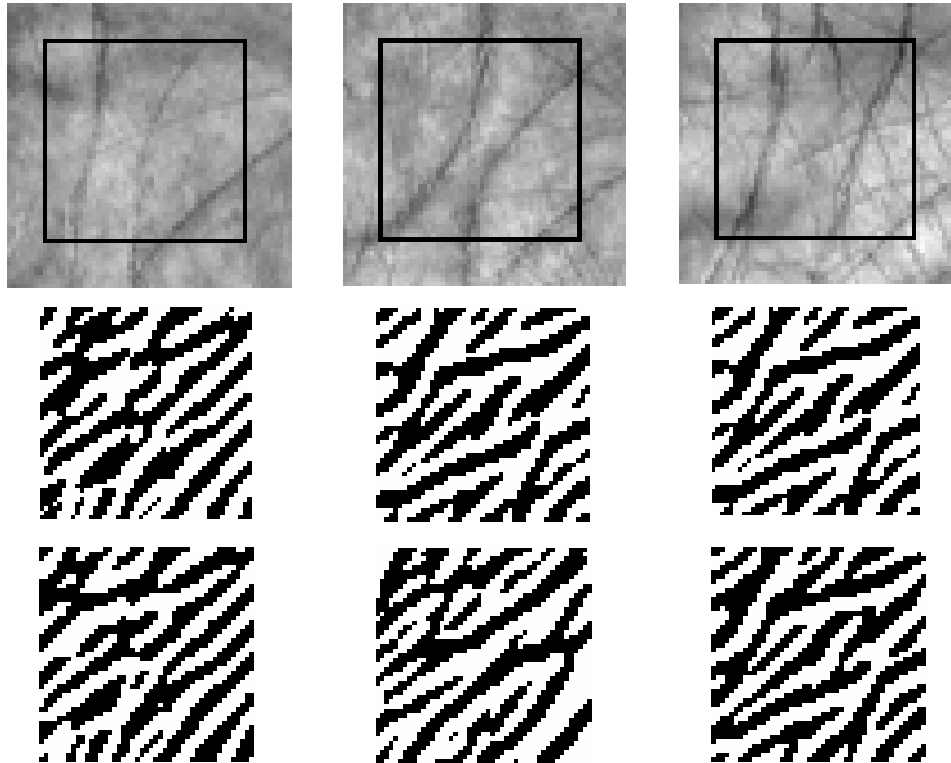
The similar palmprints from different persons



The different palmprints from different persons

Texture Feature Analysis via Wavelet filtering

- **Gabor filtering**



Each feature vector is two 2D matrices: real and imaginary part

Palm Recognition

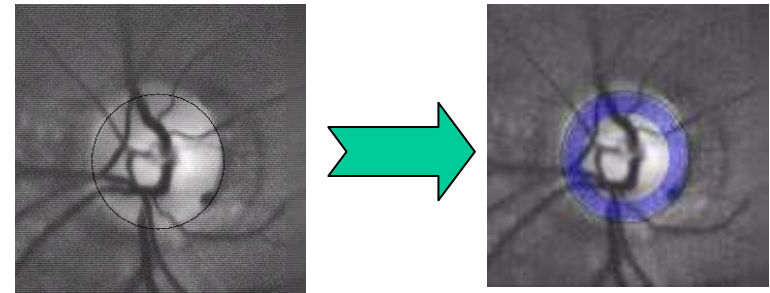
- High-resolution
 - Fingerprint like, minutia based
 - Expensive hardware, computationally complex
- Low-resolution (PolyU invention)
 - Texture feature based, binary coded for direction
 - Matching by hamming distance, $D(P, Q)$
 - Much faster in matching

$$D_o = \frac{\sum_{i=1}^N \sum_{j=1}^N (P_R(i, j) \otimes Q_R(i, j) + P_I(i, j) \otimes Q_I(i, j))}{2N^2}$$

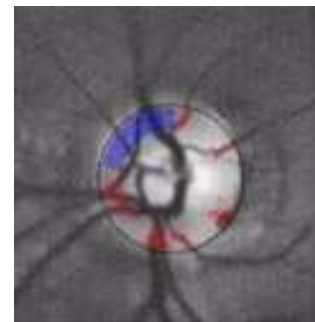
Retina System

The system employs 3 basic steps

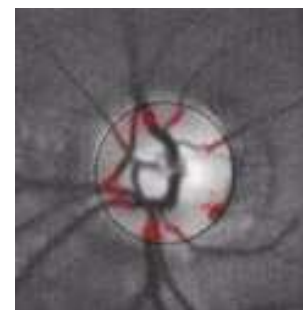
1. Image acquisition: optic disk is located and photographed.



2. Circular bar code: software translates vessel thickness and angulations into a summary pattern.

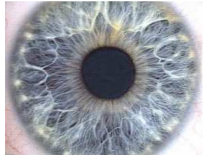


2. Pattern matching: pattern is matched against stored record.



Evaluation

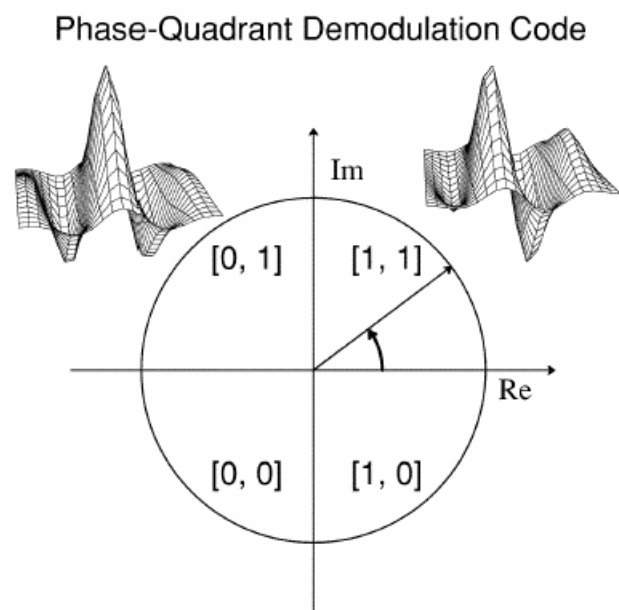
- Unique: Retina scan devices are probably **the most accurate biometric** available today.
- Permanent: retina pattern is stable .
- Collect-ability :
 - requires a laser light into the eyes, resistant from users.
 - Hardware is a bit expensive



Iris

- Iris is a very good biometric feature
 - Unique: it has very large degree of freedom
 - Universal: everyone has this
 - Permanent: not changing over time
 - Collectible: does not require an active light source to scan the eyes
- Solution:
 - Eye localization and image collection
 - Iris image extraction
 - Texture analysis via Gabor filters
 - Matching by hamming distance of texture orientation and scale. (similar to Palm !)
 - Widely adopted by US govt agencies.

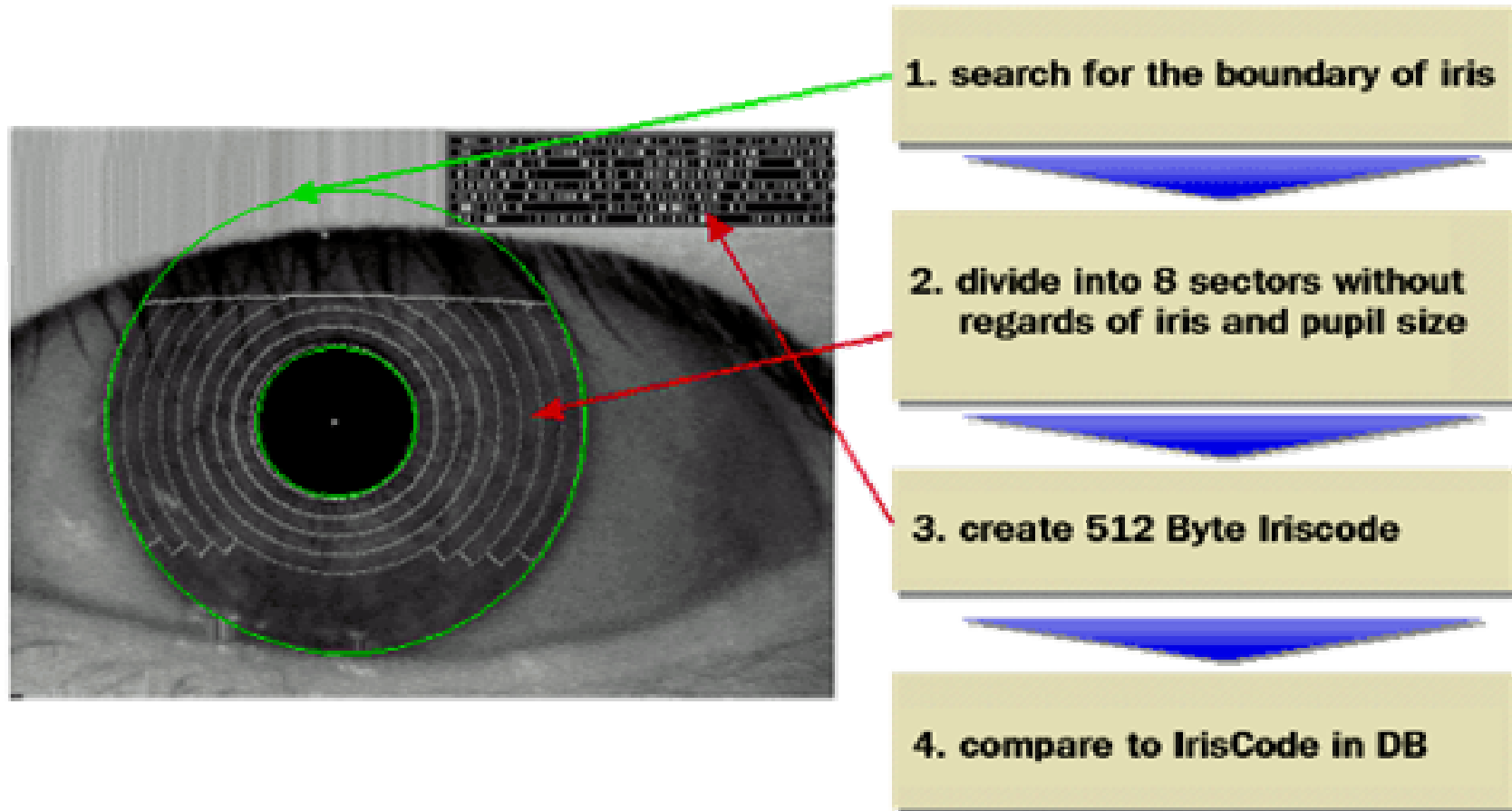
Gabor Filter



- Basically, Gabor filter finds the scale and orientation of the dominant image intensity variations for a given local area
- The output is quantized into IrisCode
- Matching is via Hamming distance on IrisCode (similar to our Palm recognition)

Fig. 2. The phase demodulation process used to encode iris patterns. Local regions of an iris are projected (2) onto quadrature 2-D Gabor wavelets, generating complex-valued coefficients whose real and imaginary parts specify the coordinates of a phasor in the complex plane. The angle of each phasor is quantized to one of the four quadrants, setting two bits of phase information. This process is repeated all across the iris with many wavelet sizes, frequencies, and orientations to extract 2048 bits.

Daugman's Approach



- J. Daugman, "High confidence visual recognition of persons by a test of statistical independence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1148-1161, 1993.

Behavior Biometrics

- **Voice:**
 - Finding the unique features of people in audio signals for matching
 - Application: verification, real-time identification
- **Gait:**
 - How people's behavior is different from each others
 - Not very reliable
- **Signature**
 - Well accepted and understood by users
 - Incorporating not only the bitmap of the signature, but also the stroke and temporal behavior in pressure, speed.
- **Key stroke:**
 - Not very reliable.