Pattern Recognition 2

KNN, PCA, LDA

Dr. Terence Sim

School of Computing
National University of Singapore

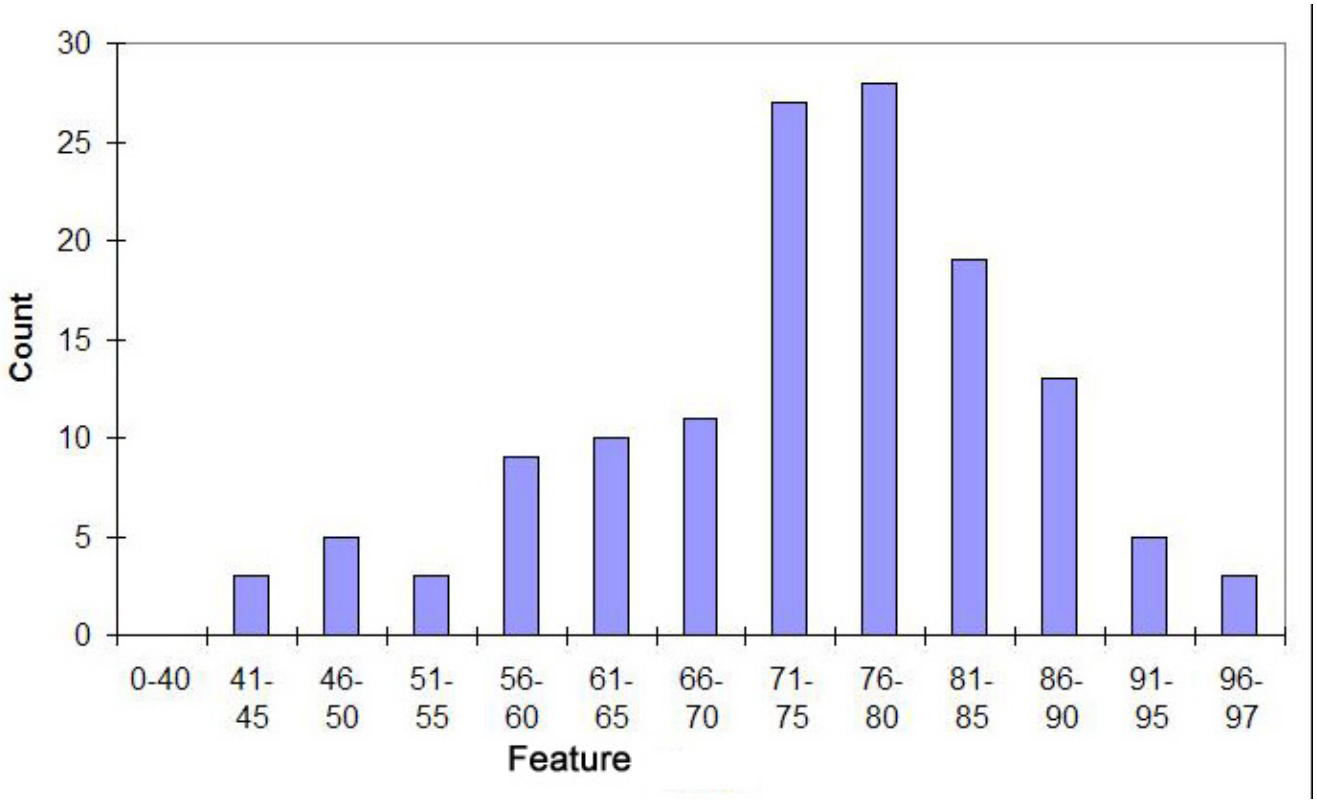
Outline

Outline

More on Bayes' Classifier

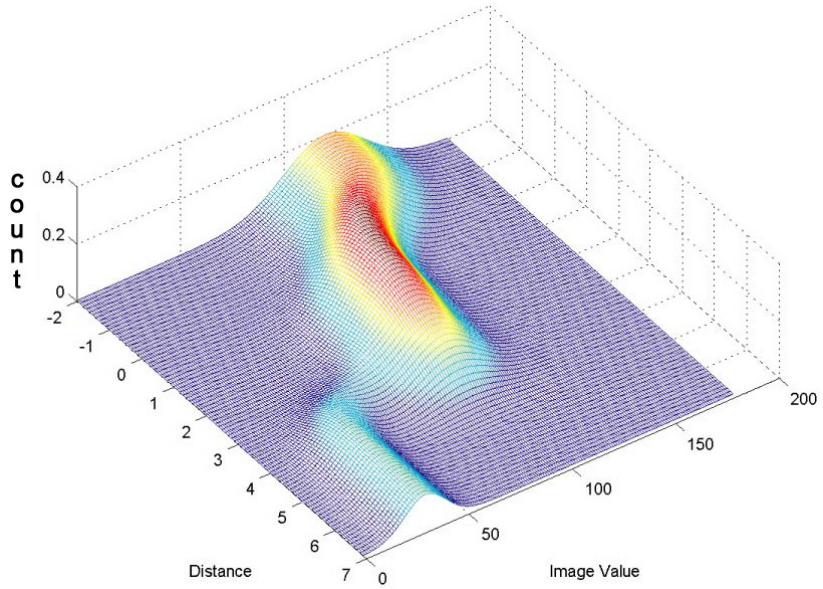
- The Bayes' Classifier is theoretically optimum.
  - That is, prob. of error (misclassification), $P(error)$, is smallest, given the set of features.
- But this requires knowing the *true* class-conditional pdfs.
  - In reality, we don't know, so we *estimate* from training data.
  - This may break the optimality, because our training data is not representative of all possible data.
- Nevertheless, the Bayes' Classifier is commonly used.
- See also: [1]

Estimating pdfs



- One common method is to use *histograms*.
- Normalized histogram = pdf.

Estimating pdfs



- Multidimensional histograms are possible.
- But require a lot of training data to estimate.
  - Heuristic: a D-dimensional histogram needs 10^D training samples.
- This is the *Curse of Dimensionality*

Naive Bayes'

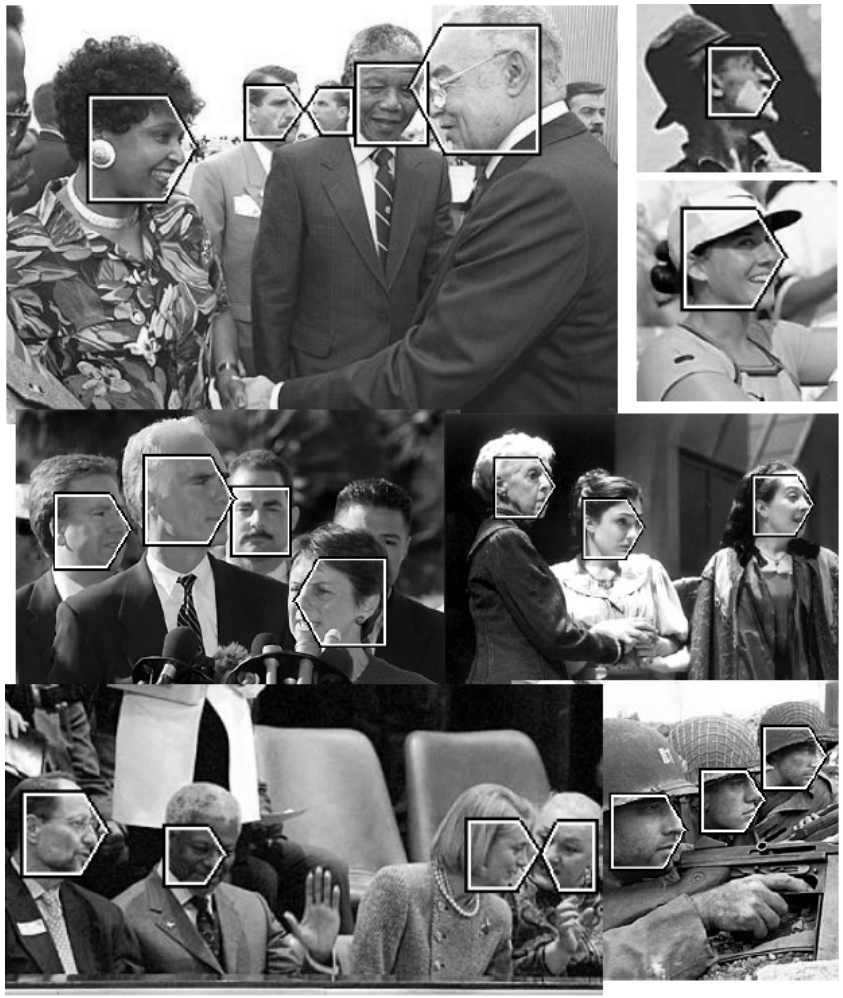
- To overcome the Curse of Dimensionality, one way is to assume that the D features are *statistically independent*.
  - Recap: two random variables X, Y are statistically independent iff $P(X, Y) = P(X)\,P(Y)$
- Suppose feature vector $\mathbf{x} = [x_1, \ldots, x_D]^\top$, then:

$$\omega^* = \arg\max_{\omega_j} P(\mathbf{x} \mid \omega_j)\, P(\omega_j)$$

$$= \arg\max_{\omega_j} P(\omega_j) \prod_{i=1}^{D} P(x_i \mid \omega_j)$$

  In practice, use log probabilities to avoid underflow.
- This is called the *Naive Bayes* assumption.
  - Surprisingly, it works well in practice!

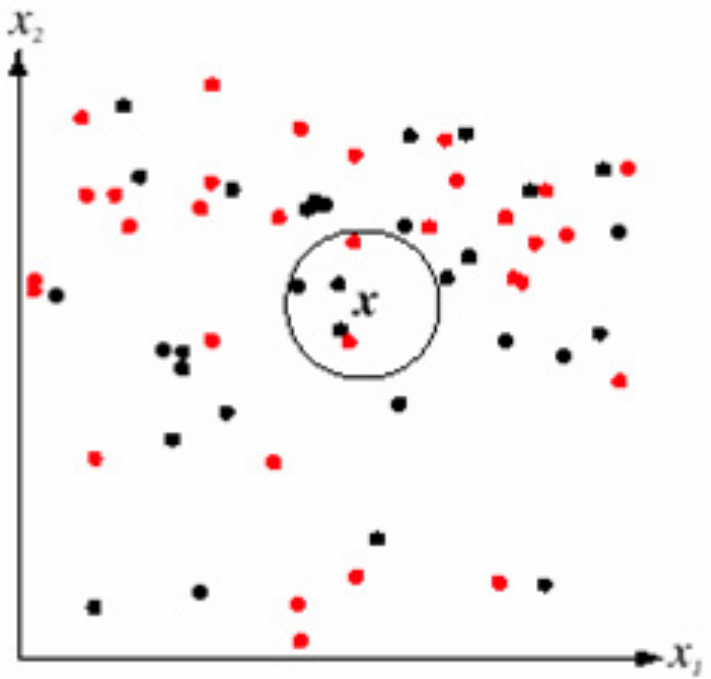
Face Detection using Naive Bayes' [2]



Outline

K-Nearest Neighbor

- Instead of using pdfs, why not simply estimate the decision boundary?
- The *K*-Nearest Neighbor (*KNN*) method estimates this implicitly.

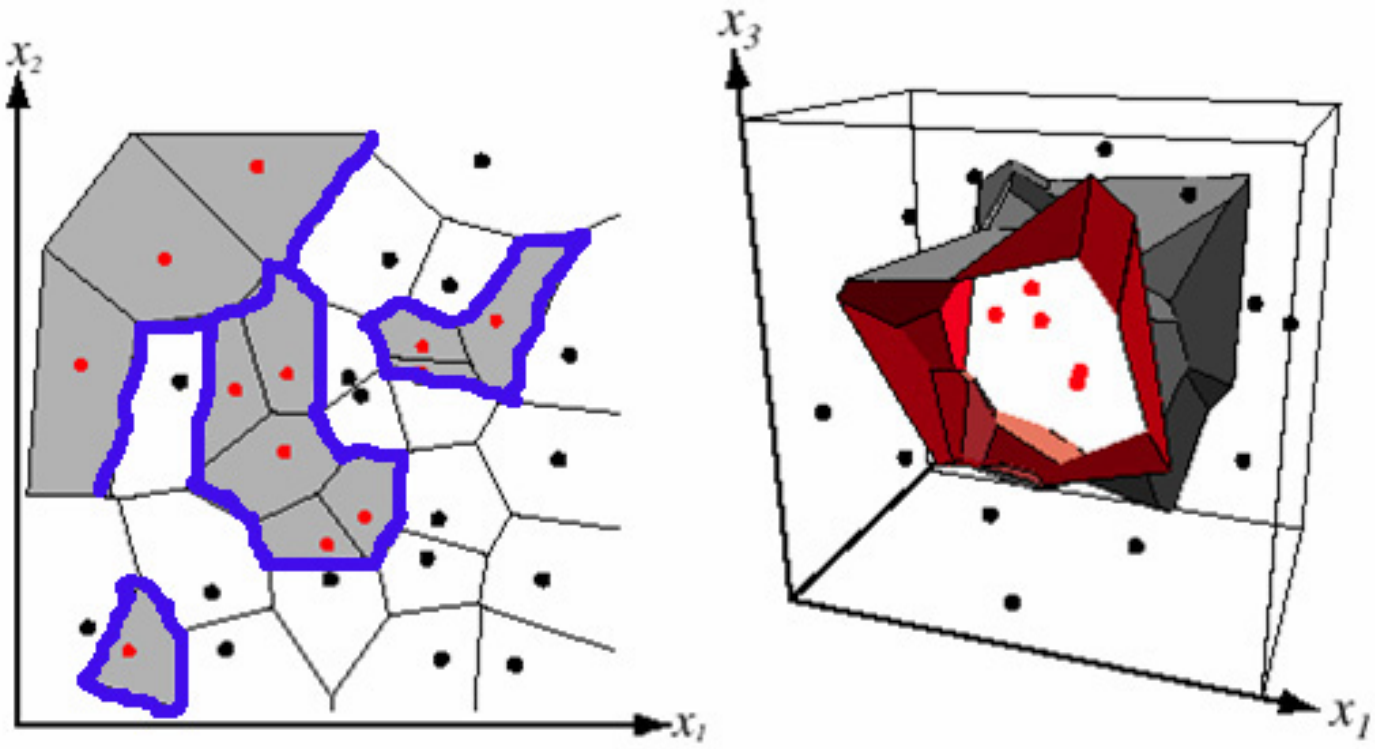


K-Nearest Neighbor

- All training data are simply stored, with their class labels.
- Given a point $\mathbf{x}$ to be classified,
  - Select the K nearest neighbors of $\mathbf{x}$.
  - Assign to $\mathbf{x}$ the majority label of these K neighbors.
- Usually, K is odd.
  - Ties can still occur: e.g. 3 classes, and $K = 3$.
- This is not optimal classifier, but given "enough" training data, $P(error_{KNN}) \leq 2\, P(error_{Bayes})$
- Notion of "nearness": Distance metric important.
  - Choice of distance metric affects accuracy.
- Pros: easy to implement. Cons: cannot scale.

Implicit Decision Boundary

- Each Voronoi cell is a region whose nearest neighbor is the training sample enclosed in the cell.



Outline

How Good is the Classifier?

- Test it with another set of labelled data (not training set).

	ω_1	ω_2	ω_3	ω_4
ω_1	**20**		3	1
ω_2	14	**10**		
ω_3		5	**15**	4
ω_4	3		3	**18**

Confusion Matrix

- The row labels are the true labels of the test data; the columns labels are the ones assigned by the classifier.
- Diagonal entries are the number of correct classifications. Off-diagonal entries are misclassifications.
- Ideally, matrix should be diagonal, meaning 0 error.
- Accuracy $= trace/total = 63/96 = 65.6\ \%$

Cross Validation

- To guard against a "fluke test".
- Divide T training samples into N equal bins.
- Leave out 1^{st} bin. Train using the rest. Test using 1^{st} bin.
- Repeat by leaving out each bin in turn, training using the rest, and testing using the omitted bin.
- Average the accuracies from all runs.
- This is called *N-fold Cross Validation*.
- Leave-1-out cross validation: when $N = T$

Outline

What Features to Use?

- Choosing features more of an art than a science.
  - No theory says which feature is best for given problem.
  - Experience, insight, familiarity help.
- Why not try all possible sets of features?
  - NP-hard!
- Greedy approach:
  - From F possible features, select 1 that gives best accuracy.
  - From $F - 1$ possible features, select 1 that, when combined with previous feature, gives best accuracy.
  - etc.
  - Common features: edges, lines, color histogram, wavelets, Fourier transform, shape, derivatives.

Desirable Properties

- Easy to compute (efficient)
- Compact (less storage)
- Good discriminative power
- Efficient distance metric
- Robust to image distortions (e.g. rotation, illumination)

Principal Components Analysis (PCA)

- a.k.a. Discrete Karhunen Loeve Transform, Hotelling Transform
- Let $\mathbf{y} \in \mathbb{R}^k$ be feature vector computed from image (or another feature vector) $\mathbf{x} \in \mathbb{R}^d$, where $k \ll d$.

$$\mathbf{y} = \mathbf{W}^\top \mathbf{x} \tag{1}$$

- $\mathbf{W}$ is $d \times k$ and orthogonal.
  - $\mathbf{W}$ to be determined from statistics of $\mathbf{x}$.
  - Let's suppose mean and covariance matrix are: $E[\mathbf{x}] = \mathbf{0}, \quad E[\mathbf{x}\mathbf{x}^\top] = \mathbf{C}_x$
- We want expected error to be small. How to compute $\mathbf{W}$?

Recap

Expectation is the mean (average) of random variable x :

$$E[x] = \int x\, p(x) dx$$

Variance is the expected squared difference from mean m :

$$Var[x] = E[(x - m)^2] = E[x^2] - (E(x))^2$$

For vectors,

$$\text{Mean: } \mathbf{m} = E[\mathbf{x}] = [E[x_1]\ E[x_2]\ \cdots E[x_d]]^\top$$

$$\text{Covariance matrix: } Var[\mathbf{x}] = E[(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^\top] = E[\mathbf{x}\mathbf{x}^\top] - \mathbf{m}\mathbf{m}^\top$$

Note: covariance matrix is *symmetric* and *positive semi-definite*

PCA

- Recovered vector $\mathbf{x}_r = \mathbf{W}\mathbf{y}$
- Error: $\epsilon = \mathbf{x} - \mathbf{x}_r = \mathbf{x} - \mathbf{W}\mathbf{W}^\top\mathbf{x}$
- We want small expected error:

$$
\begin{aligned}
||\epsilon||^2 &= \epsilon^\top\epsilon \\
&= (\mathbf{x} - \mathbf{W}\mathbf{W}^\top\mathbf{x})^\top(\mathbf{x} - \mathbf{W}\mathbf{W}^\top\mathbf{x}) \\
= \mathbf{x}^\top\mathbf{x} - \mathbf{x}^\top\mathbf{W}\mathbf{W}^\top\mathbf{x} - \mathbf{x}^\top\mathbf{W}\mathbf{W}^\top\mathbf{x} &+ \mathbf{x}^\top\mathbf{W}\mathbf{W}^\top\mathbf{W}\mathbf{W}^\top\mathbf{x} \\
&= \mathbf{x}^\top\mathbf{x} - \mathbf{x}^\top\mathbf{W}\mathbf{W}^\top\mathbf{x}
\end{aligned}
$$

Note that $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$.

PCA

Let $k = 1$, i.e. $\mathbf{W}$ is vector, $\mathbf{y}$ is scalar. Then

$$\begin{aligned} E[\varepsilon^\top \varepsilon] &= E[\mathbf{x}^\top \mathbf{x}] - E[(\mathbf{x}^\top \mathbf{w})(\mathbf{w}^\top \mathbf{x})] \\ &= E[\mathbf{x}^\top \mathbf{x}] - E[\mathbf{w}^\top \mathbf{x}\mathbf{x}^\top \mathbf{w}] \\ &= E[\mathbf{x}^\top \mathbf{x}] - \mathbf{w}^\top E[\mathbf{x}\mathbf{x}^\top]\mathbf{w} \\ &= E[\mathbf{x}^\top \mathbf{x}] - \mathbf{w}^\top \mathbf{C}_x \mathbf{w} \end{aligned}$$

PCA

We need to find $\mathbf{w}$ that minimizes $E[\epsilon^\top \epsilon]$

This is the same as maximizing the 2^{nd} term on right-hand side:

$$\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{C}_x \mathbf{w}$$

But we should normalize by length of $\mathbf{w}$, so define

$$J = \frac{\mathbf{w}^\top \mathbf{C}_x \mathbf{w}}{\mathbf{w}^\top \mathbf{w}} \tag{2}$$

Goal: find $\mathbf{w}$ to maximize J

PCA

Take derivatives and set to 0:

$$\frac{dJ}{d\mathbf{w}} = \frac{(\mathbf{w}^\top \mathbf{w})2\mathbf{C}_x\mathbf{w} - (\mathbf{w}^\top \mathbf{C}_x \mathbf{w})2\mathbf{w}}{(\mathbf{w}^\top \mathbf{w})^2} = \mathbf{0}$$

$$\mathbf{0} = \frac{2\mathbf{C}_x\mathbf{w}}{\mathbf{w}^\top \mathbf{w}} - \left[\frac{\mathbf{w}^\top \mathbf{C}_x \mathbf{w}}{\mathbf{w}^\top \mathbf{w}}\right] \bullet \frac{2\mathbf{w}}{\mathbf{w}^\top \mathbf{w}}$$

Note: term in brackets is J, so we rearrange to get:

$$\mathbf{C}_x\mathbf{w} = J\,\mathbf{w} \quad \longleftarrow \text{Eigenvalue problem!}$$

Thus, $\mathbf{w}$ is eigenvector of $\mathbf{C}_x$ corresponding to largest eigenvalue (= J).

PCA

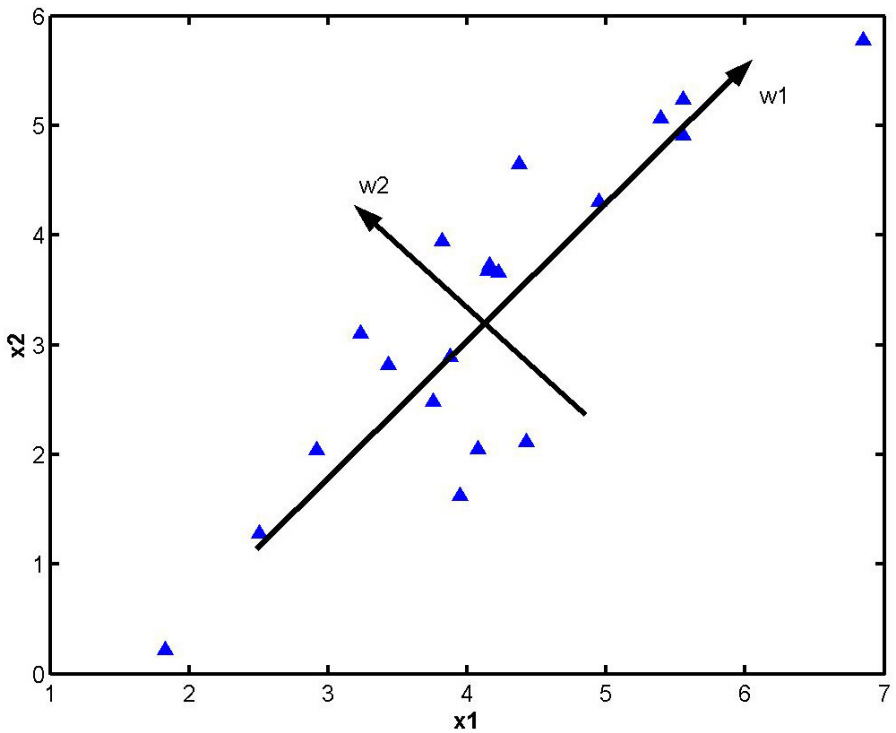
In general, PCA is: $\mathbf{y} = \mathbf{W}^\top(\mathbf{x} - \mathbf{m})$

where $\mathbf{m} = E[\mathbf{x}]$ mean, and $\mathbf{W}$ is $d \times k$ matrix containing the k eigenvectors of $Var[\mathbf{x}]$ (covariance matrix) corresponding to the top k eigenvalues.

$$\mathbf{W} = \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_k \\ | & | & & | \end{bmatrix}$$

$\mathbf{w}_1$: First principal component,
$\mathbf{w}_2$: Second principal component, etc.

PCA notes



- PCA is a shift and rotation of the axes.
- $\mathbf{w}_1$: direction of greatest elongation
- $\mathbf{w}_2$: direction of next greatest elongation, and orthogonal to previous eigenvector; etc.
- $\mathbf{W}$ is orthogonal because $\mathbf{C}_x$ is symmetric.
- $\mathbf{W}\mathbf{W}^\top \neq \mathbf{I}$ unless $k = d$

PCA notes

- Best compression of r.v. $\mathbf{x}$, in the "least-squares error" sense.
  - Guaranteed by the derivation of PCA.

- De-correlation: $\mathbf{y} = \mathbf{W}^\top(\mathbf{x} - \mathbf{m})$
  $Var[\mathbf{y}] = \mathbf{C}_y = \mathbf{W}^\top \mathbf{C}_x \mathbf{W} = \Lambda$ diagonal eigenvalue matrix
  i.e. elements of $\mathbf{y} = [y_1 \cdots y_k]^\top$ are uncorrelated.

PCA notes

Dimensionality Reduction: $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^k, k \ll d$

Typically, choose k so that ratio $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j} > 90\%$ "Energy"

But how to get $\mathbf{C}_x, \mathbf{m}$? Statistics of $\mathbf{x}$

Given data $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x}_N$, estimate $\mathbf{m}, \mathbf{C}_x$

Sample mean $\widehat{\mathbf{m}} = \frac{1}{N} \sum_{i=1}^{N} x_i$

Sample covariance matrix:

$$\widehat{\mathbf{C}_x} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^\top$$

$$\text{or } \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^\top$$

or Scatter matrix $\mathbf{S} = \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^\top$

PCA notes

Usually, $N \ll d$, so $\widehat{\mathbf{C}_x}$ is rank-deficient, also $\widehat{\mathbf{C}_x}$ is too large. $d \times d$ matrix. Computational trick: use inner products.

$$\text{Let } \mathbf{A} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 - \mathbf{m} & \mathbf{x}_2 - \mathbf{m} & \cdots & \mathbf{x}_N - \mathbf{m} \\ | & | & & | \end{bmatrix}, \ \mathbf{A} \text{ is } d \times N$$

Then $\mathbf{A}^\top \mathbf{A}$ is $N \times N$. Find eigenvectors/values of $\mathbf{A}^\top \mathbf{A}$

$$\Rightarrow \mathbf{A}^\top \mathbf{A}\mathbf{v} = \lambda \mathbf{v}$$

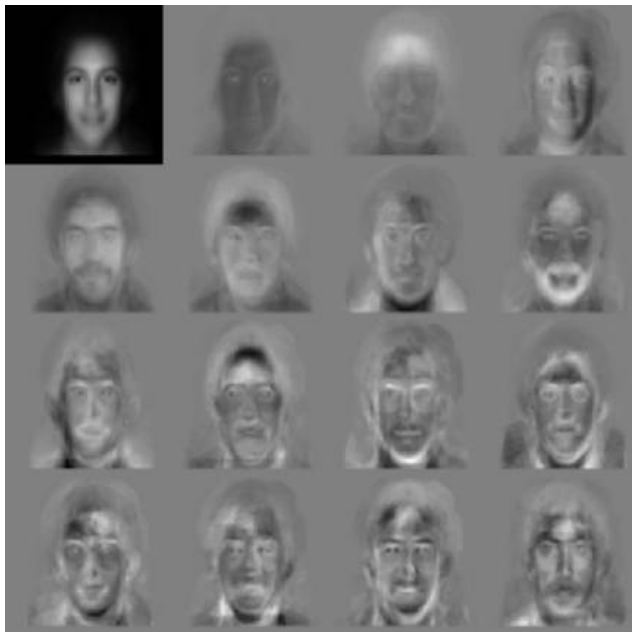
$$(\mathbf{A}\mathbf{A}^\top)\mathbf{A}\mathbf{v} = \lambda(\mathbf{A}\mathbf{v})$$

$$\Rightarrow \mathbf{A}\mathbf{v} \text{ is eigenvector of } \mathbf{A}\mathbf{A}^\top$$

Note that $\mathbf{A}\mathbf{A}^\top = \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^\top = \mathbf{S}$, scatter matrix
Thus we avoid calculating $\mathbf{A}\mathbf{A}^\top$

PCA example

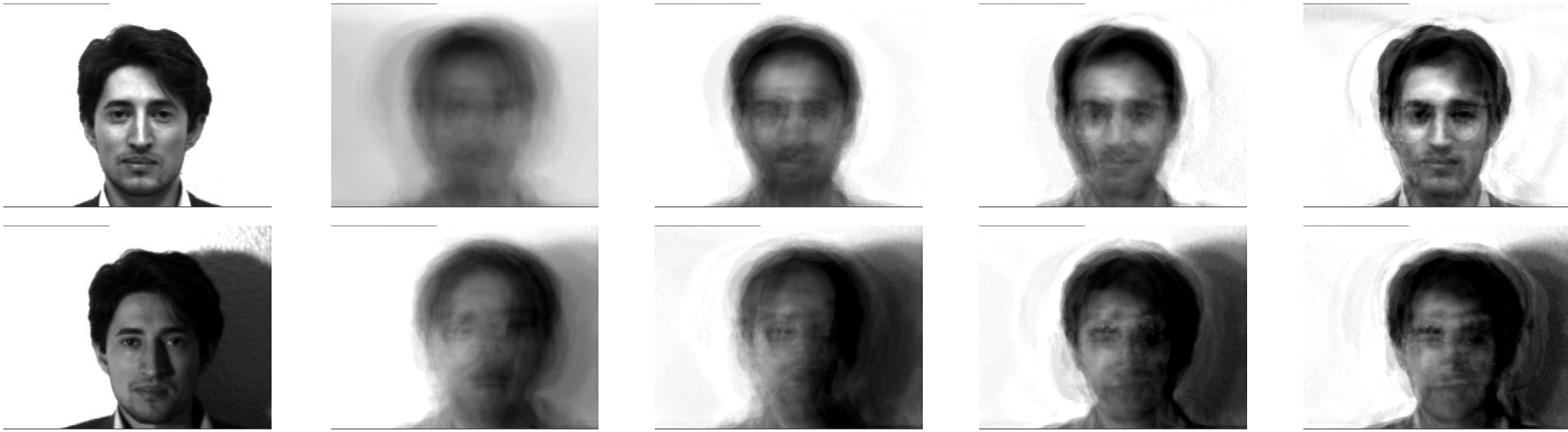
1. Perhaps the most famous use of PCA is in face recognition, as exemplified in the classic paper by Turk and Pentland [3].
2. Start with a set of face images, vectorize them, then compute the PCA. The mean and eigenfaces look like this:



PCA example

http://www.cs.princeton.edu/~mhibbs/class/cs496/eigenfaces/

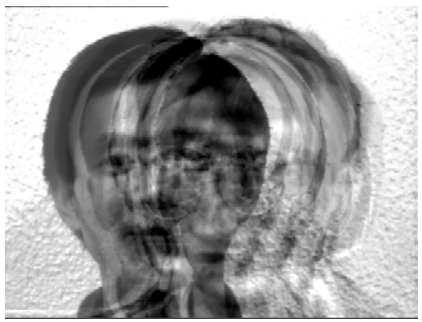
1. A face image is then a weighted sum of these eigenfaces, + mean.
2. Using more eigenfaces leads to better approximation.



Orig. image | 1 EF | 5 EF | 15 EF | 25 EF

PCA example

1. Each face is thus represented as the PCA weights.
2. Recognition can be done by finding closest weight.
3. What about images of non-faces?

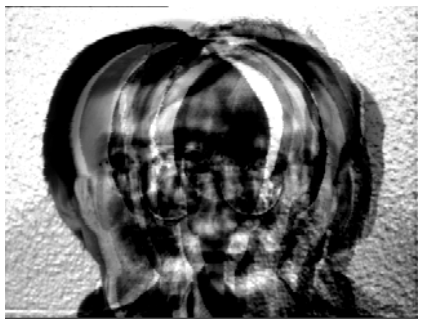


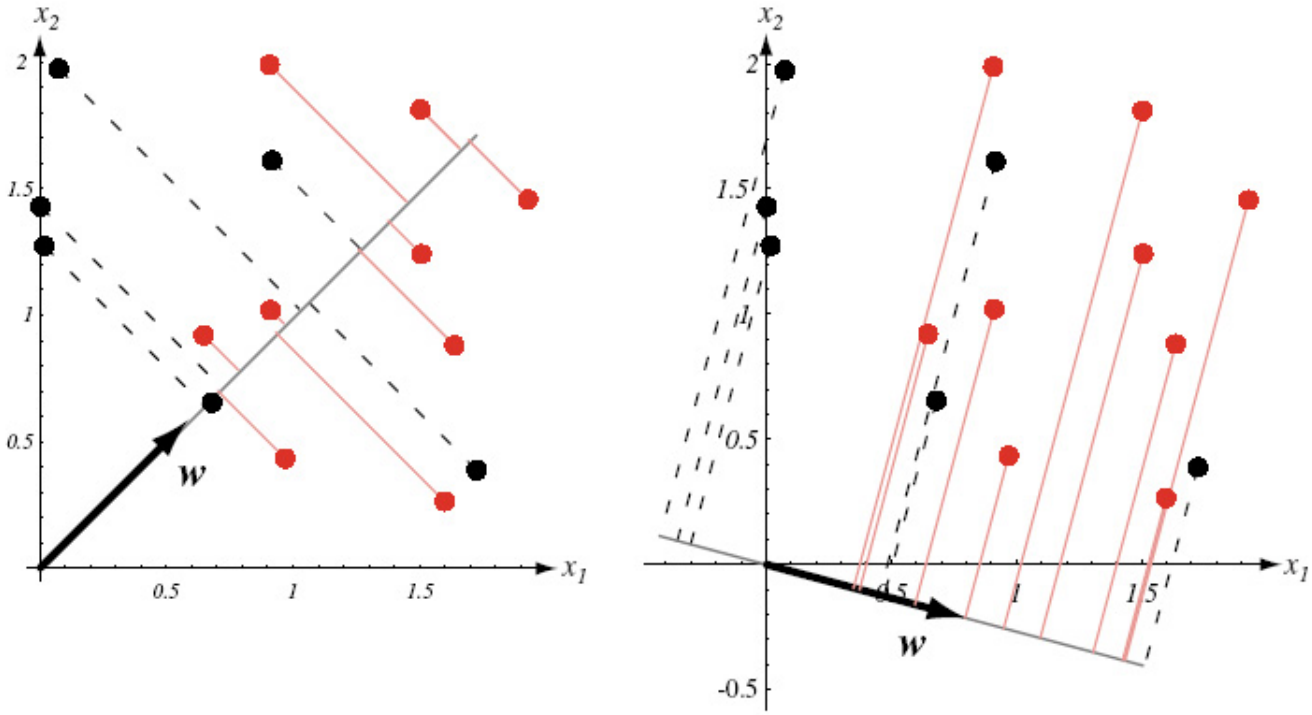
Image Approximation

Linear Discriminant Analysis (LDA)

a.k.a. Fisher Linear Discriminant

- PCA has been used as features in e.g. face recognition. See [3]
- But PCA is good for pattern *representation* rather than for pattern *discrimination*
- Nevertheless, PCA is still useful for Dimensionality Reduction

Linear Discriminant Analysis (LDA)



PCA will result in $\mathbf{w}$ shown in left figure.
LDA will result in $\mathbf{w}$ shown in right figure.

LDA

Suppose we have $C = 2$ classes: ω_1, ω_2
Define the *within-class* scatter matrix as:

$$\mathbf{S}_W = \sum_{i=1}^{C} \sum_{\mathbf{x} \in \omega_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^\top$$

where $\mathbf{m}_i$ is the mean of class ω_i.

Think of this as the sum of the covariance matrix of each class.
It measures the spread of each class.

LDA

Also define the *between-class* scatter matrix as:

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^\top$$

This measures the separation of the two classes.

LDA

Fisher's Criterion is defined as:

$$J_F = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}} \tag{3}$$

- Goal: maximize J_F, i.e. find a vector $\mathbf{w}$ such that different classes are well separated, while spread in each class is reduced.
- Compare this with PCA criteria Equation (2)
  - Total scatter matrix $\mathbf{S} = \mathbf{S}_W + \mathbf{S}_B$, so PCA is maximizing the spread in the total scatter matrix.

LDA

It can be shown that the solution to Equation (3) is:

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w} \tag{4}$$

This is called the Generalized Eigenvalue problem.
If $\mathbf{S}_W$ is invertible, then

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w}$$

which is the regular eigenvalue problem involving $\mathbf{S}_W^{-1}\mathbf{S}_B$

LDA

- Once $\mathbf{w}$ is found, feature is computed as: $\mathbf{y} = \mathbf{w}^\top \mathbf{x}$.
- Using this feature, we can then classify using, say, *KNN*.
- If more than 2 classes, then re-define between-class matrix as:

$$\mathbf{S}_B = \sum_{i=1}^{C} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^\top$$

where n_i is the number of training samples in ω_i, and $\mathbf{m}$ is the mean of all training samples (ignoring classes).

LDA

- There will be $C-1$ eigenvectors $\mathbf{w}_i$, $i = 1 \ldots C$, each solved using Equation (4).
- These eigenvectors are *not* necessarily orthogonal!
  - Because $\mathbf{S}_W^{-1}\mathbf{B}$ may not be symmetric.
- These eigenvectors form a linear subspace such that Fisher's Criterion is maximized.
- Put them into a $\mathbf{W}$ matrix, and compute feature as before: $\mathbf{y} = \mathbf{W}^\top \mathbf{x}$.

Outline

Summary

1. The Bayes' Classifier is the theoretical best, but estimating the pdfs is a problem.
2. The K-nearest neighbor classifier is easy to implement, but doesn't scale up well. Also: choice of distance metric important.
3. Deciding what features are best for a given pattern recognition problem is an art, not a science.
4. Two common techniques to compute useful features are: PCA and LDA.

References


R. Duda, P. Hart, and D. Stork.
Pattern Classification, 2nd edition.
John Wiley and Sons, 2000.


H. Schneiderman and T. Kanade.
A statistical method for 3d object detection applied to faces and cars.
In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.


M. Turk and A. Pentland.
Eigenfaces for Recognition.
Journal of Cognitive Neuroscience, 3(1), 1991.