Computer Vision
CSC9010-005

Instructor: Dr. Edward Kim
Classification in general

- “Sorting incoming Fish on a conveyor according to species using optical sensing”

Species
  - Sea bass
  - Salmon
What differentiates the two?
Problem Analysis

• Set up a camera and take some sample images to extract features
  • Length
  • Lightness
  • Width
  • Number and shape of fins
  • Position of the mouth, etc...

• This is the set of all suggested features to explore for use in our classifier!
• Preprocessing

  – Use a segmentation operation to isolate fishes from one another and from the background

• Information from a single fish is sent to a feature extractor whose purpose is to reduce the data by measuring certain features

• The features are passed to a classifier
• Classification

  – Select the length of the fish as a possible feature for discrimination
The **length** is a poor feature alone!

Select the **lightness** as a possible feature.
• Threshold decision boundary and cost relationship

  – Move our decision boundary toward smaller values of lightness in order to minimize the cost (reduce the number of sea bass that are classified salmon!)

Task of decision theory
• Adopt the lightness and add the width of the fish

\[ x^T = [x_1, x_2] \]
• We might add other features that are not correlated with the ones we already have. A precaution should be taken not to reduce the performance by adding such “noisy features”

• Ideally, the best decision boundary should be the one which provides an optimal performance such as in the following figure:
• However, our satisfaction is premature because the central aim of designing a
classifier is to correctly classify novel input

Issue of generalization!
The Design Cycle

- Data collection
- Feature Choice
- Model Choice
- Training
- Evaluation
- Computational Complexity
start

collect data

choose features

choose model

train classifier

evaluate classifier

end

prior knowledge (e.g., invariances)
• Data Collection

  – How do we know when we have collected an adequately large and representative set of examples for training and testing the system?
• Feature Choice

  – Depends on the characteristics of the problem domain.

  – (obsolete with DL) Find features that are: Simple to extract, invariant to irrelevant transformations, insensitive to noise, and useful in discriminating patterns in different categories.

  – (obsolete with DL) Prior knowledge plays a major role- how to combine prior knowledge and empirical data to find relevant and effective features?
• Model Choice
  
  – Unsatisfied with the performance of our fish classifier and want to jump to another class of model

  • Based on some function of the number and position of the fins, the color of the eyes, the weight, shape of the mouth ...
• Training

  – Use data to determine the classifier. Many different procedures for training classifiers and choosing models
• Evaluation

  – Measure the error rate (or performance and switch from one set of features to another one
• Computational Complexity

  – What is the trade-off between computational ease and performance?

  – (How an algorithm scales as a function of the number of features, patterns or categories?)
Dimensionality

• Adding more and more features...
• Get a population, predict some properties
  – Instances represented as \{urefu, height\} pairs
  – What is the dimensionality of this data?
• Get a population, predict some properties
  – Instances represented as \{urefu, height\} pairs
  – What is the dimensionality of this data?
• Get a population, predict some properties
  – Instances represented as \{urefu, height\} pairs
  – What is the dimensionality of this data?

“height” = ‘urefu” in Swahili
• Get a population, predict some properties
  – Instances represented as \{\text{urefu, height}\} pairs
  – What is the dimensionality of this data?

• Data points over time from different geographic areas over time:
  • $X_1$: # of skidding accidents
  • $X_2$: # of burst water pipes
  • $X_3$: snow-plow expenditures
  • $X_4$: # of school closures
  • $X_5$: # patients with heat stroke
Curse of dimensionality

• Datasets typically high dimensional
• But the true dimensionality is often much lower

• Total dimensionality
  – 20x20 bitmap
  – \(2^{400}\) events
Curse of dimensionality

- Datasets typically high dimensional
- But the true dimensionality is often much lower

- Total dimensionality
  - 20x20 bitmap
  - $2^{400}$ events
  - True dimensionality is lower
Dealing with high dimensionality

- Harris, SIFT, HOG reduce the space
  - Manual

- Or you can reduce the dimensions
  - Automatically
Dimensionality Reduction

• Goal: represent the “numbers” or “faces” with fewer variables
  – Preserve the structure of the data
  – Be discriminative, only structure that affects class separability

• Feature selection
  – Pick a subset of dimensions, X1, X2, X3, X4
    • Use only X2, X4 (height, width)
Feature extraction

• Construct a new set of dimensions

\[ E_i = f(X_1 \ldots X_d) \]

• Linear combination of original
Principal Component Analysis

• Defines a set of principal components
  – Pick a line of greatest variability in the data
  – Next perpendicular to first, greatest variability of what is left... until d (original dimensionality)
Principal Component Analysis

• Defines a set of principal components
• M components become m new dimensions
  – Change coordinates of every data point in these dimensions
Why greatest variability

- Example: reduce 2-dimensional data to 1-d
- Pick e to maximize variability
Why greatest variability

• Example: reduce 2-dimensional data to 1-d
• Pick e to maximize variability

• Preserve distances
  – In projected space
  – And original space
Principal components

- “Center” the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
Principal components

- “Center” the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
**Principal components**

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
Principal components

• “Center” the data at zero: \( x_{i,a} = x_{i,a} - \mu \)
  – subtract mean from each attribute
• Compute covariance matrix \( \Sigma \)
  – covariance of dimensions \( x_1 \) and \( x_2 \):
    • do \( x_1 \) and \( x_2 \) tend to increase together?
    • or does \( x_2 \) decrease as \( x_1 \) increases?

\[
\begin{pmatrix}
2.0 & 0.8 \\
0.8 & 0.6
\end{pmatrix}
\]

\[
\text{cov}(b,a) = \frac{1}{n} \sum_{i=1}^{n} x_{ib} x_{ia}
\]
Principal components

• “Center” the data at zero: $x_{i,a} = x_{i,a} - \mu$
  – subtract mean from each attribute

• Compute covariance matrix $\Sigma$
  – covariance of dimensions $x_1$ and $x_2$:
    • do $x_1$ and $x_2$ tend to increase together?
    • or does $x_2$ decrease as $x_1$ increases?
Principal components

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by $\Sigma$: \[
\begin{pmatrix}
2.0 & 0.8 \\
0.8 & 0.6
\end{pmatrix}
\begin{pmatrix}
-1 \\
+1
\end{pmatrix} \rightarrow \begin{pmatrix}
-1.2 \\
-0.2
\end{pmatrix}
\] again
  - turns towards direction of variance
Principal components

- “Center” the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by $\Sigma$: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again
  - turns towards direction of variance

\[
\text{cov}(a,b) = \frac{1}{n} \sum_{i=1}^{n} x_{ia} x_{ib}
\]

\[
\text{var}(a) = \frac{1}{n} \sum_{i=1}^{n} x_{ia}^2
\]
Principal components

• “Center” the data at zero: \( x_{i,a} = x_{i,a} - \mu \)
  – subtract mean from each attribute

• Compute covariance matrix \( \Sigma \)
  – covariance of dimensions \( x_1 \) and \( x_2 \):
    • do \( x_1 \) and \( x_2 \) tend to increase together?
    • or does \( x_2 \) decrease as \( x_1 \) increases?

• Multiply a vector by \( \Sigma \):
  \[
  \begin{pmatrix}
  2.0 & 0.8 \\
  0.8 & 0.6 \\
  \end{pmatrix}
  \begin{pmatrix}
  -1 \\
  +1 \\
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  -1.2 \\
  -0.2 \\
  \end{pmatrix}
  \]
  again \( \rightarrow \begin{pmatrix}
  -2.5 \\
  -1.0 \\
  \end{pmatrix} \rightarrow \begin{pmatrix}
  -6.0 \\
  -2.7 \\
  \end{pmatrix} \)
  – turns towards direction of variance

\[
\text{var}(a) = \frac{1}{n} \sum_{i=1}^{n} x_{ia}^2
\]
\[
\text{cov}(b,a) = \frac{1}{n} \sum_{i=1}^{n} x_{ib} \cdot x_{ia}
\]
Principal components

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by $\Sigma$: \[
\begin{pmatrix}
2.0 & 0.8 \\
0.8 & 0.6 \\
\end{pmatrix}
\begin{pmatrix}
-1 \\
+1 \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
-1.2 \\
+0.8 \\
\end{pmatrix}
\] again \[
\begin{pmatrix}
-2.5 \\
+1.0 \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
-6.0 \\
+2.7 \\
\end{pmatrix}
\]
  - turns towards direction of variance
**Principal components**

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by $\Sigma$: \[
\begin{pmatrix}
2.0 \\ 0.8 \\
0.8 \\ 0.6
\end{pmatrix}
\begin{pmatrix}
-1 \\
+1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
-1.2 \\
+0.2
\end{pmatrix}
\] again
  \[
\begin{pmatrix}
-2.5 \\
-1.0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
-6.0 \\
-2.7
\end{pmatrix}
\rightarrow
\begin{pmatrix}
-14.1 \\
-6.4
\end{pmatrix}
\rightarrow
\begin{pmatrix}
-33.3 \\
-15.1
\end{pmatrix}
\]
  - turns towards direction of variance

\[
\text{var}(a) = \frac{1}{n} \sum_{i=1}^{n} x_{ia}^2
\]
\[
\text{cov}(b,a) = \frac{1}{n} \sum_{i=1}^{n} x_{ib} \cdot x_{ia}
\]
Principal components

- “Center” the data at zero: \( x_{i,a} = x_{i,a} - \mu \)
  - subtract mean from each attribute
- Compute covariance matrix \( \Sigma \)
  - covariance of dimensions \( x_1 \) and \( x_2 \):
    - do \( x_1 \) and \( x_2 \) tend to increase together?
    - or does \( x_2 \) decrease as \( x_1 \) increases?
- Multiply a vector by \( \Sigma \):
  \[
  \begin{pmatrix}
  2.0 & 0.8 \\
  0.8 & 0.6
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  -1.2 \\
  -0.2
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  -2.5 \\
  -1.0
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  -6.0 \\
  -2.7
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  -14.1 \\
  -6.4
  \end{pmatrix}
  \rightarrow
  \begin{pmatrix}
  -33.3 \\
  -15.1
  \end{pmatrix}
  \]
  - turns towards direction of variance
- Want vectors \( e \) which aren’t turned: \( \Sigma e = \lambda e \)
  - \( e \) ... eigenvectors of \( \Sigma \), \( \lambda \) ... corresponding eigenvalues
  - principal components = eigenvectors w. largest eigenvalues
Finding Principal Components

1. find eigenvalues by solving: \( \det(\Sigma - \lambda I) = 0 \)

\[
\begin{vmatrix}
2.0 - \lambda & 0.8 \\
0.8 & 0.6 - \lambda
\end{vmatrix} = (2 - \lambda)(0.6 - \lambda) - (0.8)(0.8) = \lambda^2 - 2.6\lambda + 0.56 = 0
\]

\[
\{\lambda_1, \lambda_2\} = \frac{1}{2} \left( 2.6 \pm \sqrt{2.6^2 - 4 \times 0.56} \right) = \{2.36, 0.23\}
\]

2. find \( i^{th} \) eigenvector by solving: \( \Sigma \mathbf{e}_i = \lambda_i \mathbf{e}_i \)

\[
\begin{pmatrix}
2.0 & 0.8 \\
0.8 & 0.6
\end{pmatrix}
\begin{pmatrix}
e_{1,1} \\
e_{1,2}
\end{pmatrix} = 2.36
\begin{pmatrix}
e_{1,1} \\
e_{1,2}
\end{pmatrix} \quad \Rightarrow \quad 2.0e_{1,1} + 0.8e_{1,2} = 2.36e_{1,1}
\]

\[
\begin{pmatrix}
2.0 & 0.8 \\
0.8 & 0.6
\end{pmatrix}
\begin{pmatrix}
e_{2,1} \\
e_{2,2}
\end{pmatrix} = 0.23
\begin{pmatrix}
e_{2,1} \\
e_{2,2}
\end{pmatrix} \quad \Rightarrow \quad 0.8e_{1,1} + 0.6e_{1,2} = 2.36e_{2,2}
\]

\[
e_{1,1} = 2.2e_{1,2}
\]

\[
e_1 \approx \begin{bmatrix} 2.2 \\ 1 \end{bmatrix}
\]

want: \( ||\mathbf{e}_1|| = 1 \)

\[
e_1 = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}
\]

slope: 0.454
Project data points to new dimensions

- $e_1 \ldots e_m$ are new dimension vectors
- Have instance $x = \{x_1 \ldots x_d\}$ (original coordinates)
- Want new coordinates $x' = \{x'_1 \ldots x'_m\}$:
  1. “center” the instance (subtract the mean): $x' - \mu$
  2. “project” to each dimension: $(x' - \mu)^T e_j$ for $j = 1 \ldots m$
Projecting to new dimensions

- $e_1 \ldots e_m$ are new dimension vectors
- Have instance $x = \{x_1 \ldots x_d\}$ (original coordinates)
- Want new coordinates $x' = \{x'_1 \ldots x'_m\}$:
  1. “center” the instance (subtract the mean): $x' - \mu$
  2. “project” to each dimension: $(x' - \mu)^T e_j$ for $j=1\ldots m$

$$\begin{align*}
(x - \mu) &= \begin{bmatrix}
(x_1 - \mu_1) & (x_2 - \mu_2) & \cdots & (x_d - \mu_d)
\end{bmatrix}
\end{align*}$$
Projecting to new dimensions

- $\mathbf{e}_1 \ldots \mathbf{e}_m$ are new dimension vectors
- Have instance $\mathbf{x} = \{x_1 \ldots x_d\}$ (original coordinates)
- Want new coordinates $\mathbf{x}' = \{x'_1 \ldots x'_m\}$:
  1. “center” the instance (subtract the mean): $\mathbf{x}' - \mu$
  2. “project” to each dimension: $(\mathbf{x}' - \mu)^T \mathbf{e}_j$ for $j=1 \ldots m$

\[
\begin{align*}
(\bar{x} - \bar{\mu}) &= \left[ (x_1 - \mu_1) \quad (x_2 - \mu_2) \quad \cdots \quad (x_d - \mu_d) \right] \\
\begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_m \end{bmatrix} &= \begin{bmatrix} (\bar{x} - \bar{\mu})^T \mathbf{e}_1 \\ (\bar{x} - \bar{\mu})^T \mathbf{e}_2 \\ \vdots \\ (\bar{x} - \bar{\mu})^T \mathbf{e}_m \end{bmatrix} = \begin{bmatrix} (x_1 - \mu_1)e_{1,1} + (x_2 - \mu_2)e_{1,2} + \cdots + (x_d - \mu_d)e_{1,d} \\ (x_1 - \mu_1)e_{2,1} + (x_2 - \mu_2)e_{2,2} + \cdots + (x_d - \mu_d)e_{2,d} \\ \vdots \\ (x_1 - \mu_1)e_{m,1} + (x_2 - \mu_2)e_{m,2} + \cdots + (x_d - \mu_d)e_{m,d} \end{bmatrix}
\end{align*}
\]
How many eigenvectors

- Have: eigenvectors $e_1 \ldots e_d$ want: $m \ll d$
- Proved: eigenvalue $\lambda_i =$ variance along $e_i$
- Pick $e_i$ that “explain” the most variance
  - sort eigenvectors s.t. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
  - pick first $m$ eigenvectors which explain 90% or the total variance
    - typical threshold values: 0.9 or 0.95
PCA in a nutshell

1. correlated hi-d data
   ("urefu" means "height" in Swahili)
PCA in a nutshell

1. correlated hi-d data
   ("urefu" means "height" in Swahili)

2. center the points

want dimension of highest variance
PCA in a nutshell

1. correlated hi-d data
   ("urefu" means "height" in Swahili)

2. center the points

3. compute covariance matrix

\[
\begin{bmatrix}
2.0 & 0.8 \\
0.8 & 0.6 \\
\end{bmatrix}
\]

\[
\text{cov}(h,u) = \frac{1}{n} \sum_{i=1}^{n} h_i u_i
\]

want dimension of highest variance
PCA in a nutshell

1. correlated hi-d data
   ("urefu" means "height" in Swahili)

2. center the points

3. compute covariance matrix
   \[
   \text{cov}(h, u) = \frac{1}{n} \sum_{i=1}^{n} h_i u_i
   \]
   \[
   \begin{bmatrix}
   h & u \\
   2.0 & 0.8 \\
   0.8 & 0.6
   \end{bmatrix}
   \]

4. eigenvectors + eigenvalues
   \[
   \begin{bmatrix}
   e_h \\
   e_u
   \end{bmatrix}
   = \lambda_e
   \begin{bmatrix}
   e_h \\
   e_u
   \end{bmatrix}
   \]
   \[
   \begin{bmatrix}
   2.0 & 0.8 \\
   0.8 & 0.6
   \end{bmatrix}
   \begin{bmatrix}
   f_h \\
   f_u
   \end{bmatrix}
   = \lambda_f
   \begin{bmatrix}
   f_h \\
   f_u
   \end{bmatrix}
   \]

want dimension of highest variance
PCA in a nutshell

1. correlated hi-d data
   ("urefu" means "height" in Swahili)

2. center the points

3. compute covariance matrix
   \[
   \text{cov}(h,u) = \frac{1}{n} \sum_{i=1}^{n} h_i u_i
   \]

4. eigenvectors + eigenvalues
   \[
   \begin{pmatrix}
   2.0 & 0.8 \\
   0.8 & 0.6
   \end{pmatrix}
   \begin{pmatrix}
   e_h \\
   e_u
   \end{pmatrix}
   = \lambda_e
   \begin{pmatrix}
   e_h \\
   e_u
   \end{pmatrix}
   \]
   \[
   \begin{pmatrix}
   2.0 & 0.8 \\
   0.8 & 0.6
   \end{pmatrix}
   \begin{pmatrix}
   f_h \\
   f_u
   \end{pmatrix}
   = \lambda_f
   \begin{pmatrix}
   f_h \\
   f_u
   \end{pmatrix}
   \]

5. pick m<d eigenvectors
   w. highest eigenvalues

want dimension of highest variance
**PCA in a nutshell**

1. correlated hi-d data
   ("urefu" means "height" in Swahili)

2. center the points

3. compute covariance matrix
   \[
   \begin{bmatrix} h \\ u \end{bmatrix} \xrightarrow{\text{cov}(h,u)} \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix}
   \]
   \[
   \frac{1}{n} \sum h_i u_i
   \]

4. eigenvectors + eigenvalues
   \[
   \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} e_h \\ e_u \end{bmatrix} = \lambda \begin{bmatrix} e_h \\ e_u \end{bmatrix}
   \]
   \[
   \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} f_h \\ f_u \end{bmatrix} = \lambda \begin{bmatrix} f_h \\ f_u \end{bmatrix}
   \]

5. pick m<d eigenvectors w. highest eigenvalues

6. project data points to those eigenvectors

**Equation:**

\[
\mathbf{x}_e = \mathbf{x}^T \mathbf{e} = \sum_{j=1}^{d} \mathbf{x}_j \mathbf{e}_j
\]
PCA in a nutshell

1. correlated hi-d data
   ("urefu" means "height" in Swahili)

2. center the points

3. compute covariance matrix

\[
\begin{bmatrix}
h & u \\
2.0 & 0.8 \\
0.8 & 0.6
\end{bmatrix}
\]

\[
\text{cov}(h,u) = \frac{1}{n} \sum_{i=1}^{n} h_i u_i
\]

4. eigenvectors + eigenvalues

\[
\begin{bmatrix}
h \\
2.0 & 0.8 \\
0.8 & 0.6
\end{bmatrix}
\begin{bmatrix}
e_h \\
e_u
\end{bmatrix}
= \lambda_e
\begin{bmatrix}
e_h \\
e_u
\end{bmatrix}
\]

\[
\begin{bmatrix}
h \\
2.0 & 0.8 \\
0.8 & 0.6
\end{bmatrix}
\begin{bmatrix}
f_h \\
f_u
\end{bmatrix}
= \lambda_f
\begin{bmatrix}
f_h \\
f_u
\end{bmatrix}
\]

5. pick m<d eigenvectors w. highest eigenvalues

7. uncorrelated low-d data

6. project data points to those eigenvectors

\[
x'_e = x^T e = \sum_{j=1}^{d} x_j e_j
\]
Eigen Faces
input: dataset of N face images
input: dataset of N face images

face: $K \times K$ bitmap of pixels

“unfold” each bitmap to $K^2$-dimensional vector
input: dataset of N face images

face: $K \times K$ bitmap of pixels

“unfold” each bitmap to $K^2$-dimensional vector

arrange in a matrix
each face = column

$K^2 \times N$

PCA

$K^2 \times m$

set of $m$ eigenvectors
each is $K^2$-dimensional
input: dataset of N face images

face: $K \times K$ bitmap of pixels

"unfold" each bitmap to $K^2$-dimensional vector

arrange in a matrix, each face = column

$K^2 \times N$

"fold" into a $K \times K$ bitmap

PCA

set of $m$ eigenvectors, each is $K^2$-dimensional
input: dataset of N face images

face: $K \times K$ bitmap of pixels

“unfold” each bitmap to $K^2$-dimensional vector

arrange in a matrix each face = column

“fold” into a $K \times K$ bitmap

set of $m$ eigenvectors each is $K^2$-dimensional

can visualize eigenvectors: $m$ “aspects” of prototypical facial features
• Project new face to space of eigen-faces
• Represent vector as a linear combination of principal components
• How many do we need?
= mean + 0.9 * - 0.2 * + 0.4 * + ...

- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?
= mean + 0.9 * - 0.2 * + 0.4 * + ...

- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?
applications

- Face similarity
  - in the reduced space
  - insensitive to lighting expression, orientation
- Projecting new “faces”
  - everything is a face
Eigenface Recognition

- Face similarity
  - in the reduced space
  - insensitive to lighting expression, orientation
- Projecting new “faces”
  - everything is a face

new face

projected to eigenfaces
Compute the Face distance

The distance of a projected face DIFS (Distance in face space)

\[ DIFS = \| \tilde{x} - m \| = \sqrt{\sum_{i=0}^{M-1} a_i^2} \]

The distance between two faces

\[ DIFS = \| \tilde{x} - \tilde{y} \| = \sqrt{\sum_{i=0}^{M-1} (a_i - b_i)^2} \]
We are not utilizing the eigen value information, compute Mahalonobis distance

\[ DIFS' = \|\widetilde{x} - m\|_{C^{-1}} = \sqrt{\sum_{i=0}^{M-1} \frac{a_i^2}{\lambda_i}} \]

Pre-scale the eigen vectors by eigenvalues:

\[ \hat{U} = UA^{-1/2} \]

Euclidean

\[ d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}. \]

Mahalonobis

\[ d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}. \]
Linear Discriminant Analysis
Linear Discriminant Analysis

- PCA is unsupervised
  - maximizes overall variance of the data along a small set of directions
  - does not know anything about class labels
  - can pick direction that makes it hard to separate classes

- Discriminative approach
  - look for a dimension that makes it easy to separate classes
Linear Discriminant Analysis

- LDA: pick a new dimension that gives:
  - maximum separation between means of projected classes
  - minimum variance within each projected class
- Solution: eigenvectors based on between-class and within-class covariance matrices

\[
\max \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}
\]