Dimensionality Reduction
Feature Selection vs. Dimensionality Reduction

• Feature Selection (last time)
  – Select a subset of features.
  – When classifying novel patterns, only a small number of features need to be computed (i.e., faster classification).
  – The measurement units (length, weight, etc.) of the features are preserved.

• Dimensionality Reduction (this time)
  – Transform features into a smaller set.
  – When classifying novel patterns, all features need to be computed.
  – The measurement units (length, weight, etc.) of the features are lost.
How Can We Visualize High Dimensional Data?

- E.g., 53 blood and urine tests for 65 patients

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>8.0000</td>
<td>4.8200</td>
<td>14.1000</td>
<td>41.0000</td>
<td>85.0000</td>
<td>29.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A2</td>
<td>7.3000</td>
<td>5.0200</td>
<td>14.7000</td>
<td>43.0000</td>
<td>86.0000</td>
<td>29.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A3</td>
<td>4.3000</td>
<td>4.4800</td>
<td>14.1000</td>
<td>41.0000</td>
<td>91.0000</td>
<td>32.0000</td>
<td>35.0000</td>
</tr>
<tr>
<td>A4</td>
<td>7.5000</td>
<td>4.4700</td>
<td>14.9000</td>
<td>45.0000</td>
<td>101.0000</td>
<td>33.0000</td>
<td>33.0000</td>
</tr>
<tr>
<td>A5</td>
<td>7.3000</td>
<td>5.5200</td>
<td>15.4000</td>
<td>46.0000</td>
<td>84.0000</td>
<td>28.0000</td>
<td>33.0000</td>
</tr>
<tr>
<td>A6</td>
<td>6.9000</td>
<td>4.8600</td>
<td>16.0000</td>
<td>47.0000</td>
<td>97.0000</td>
<td>33.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A7</td>
<td>7.8000</td>
<td>4.6800</td>
<td>14.7000</td>
<td>43.0000</td>
<td>92.0000</td>
<td>31.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A8</td>
<td>8.6000</td>
<td>4.8200</td>
<td>15.8000</td>
<td>42.0000</td>
<td>88.0000</td>
<td>33.0000</td>
<td>37.0000</td>
</tr>
<tr>
<td>A9</td>
<td>5.1000</td>
<td>4.7100</td>
<td>14.0000</td>
<td>43.0000</td>
<td>92.0000</td>
<td>30.0000</td>
<td>32.0000</td>
</tr>
</tbody>
</table>

Features

Difficult to see the correlations between the features...
Data Visualization

- Is there a representation better than the raw features?
  - Is it really necessary to show all the 53 dimensions?
  - ... what if there are strong correlations between the features?

Could we find the *smallest* subspace of the 53-D space that keeps the *most information* about the original data?

One solution: **Principal Component Analysis**
Principle Component Analysis

Orthogonal projection of data onto lower-dimension linear space that...

- maximizes variance of projected data (purple line)
- minimizes mean squared distance between data point and projections (sum of blue lines)
The Principal Components

• **Vectors** originating from the center of mass

• Principal component #1 points in the direction of the largest variance

• Each subsequent principal component...
  • is **orthogonal** to the previous ones, and
  • points in the directions of the largest variance of the residual subspace
2D Gaussian Dataset
1st PCA axis
2\textsuperscript{nd} PCA axis
PCA Algorithm

• Given data \{x_1, ..., x_n\}, compute covariance matrix \(\Sigma\)
  • \(X\) is the \(n \times d\) data matrix
  • Compute data mean (average over all rows of \(X\))
  • Subtract mean from each row of \(X\) (centering the data)
  • Compute covariance matrix \(\Sigma = X^TX\)  \(\text{(}\Sigma\text{ is } d \times d\text{)}\)

• **PCA** basis vectors are given by the eigenvectors of \(\Sigma\)
  • \(Q, \Lambda = \text{numpy.linalg.eig}(\Sigma)\)
  • \(\{q_i, \lambda_i\}_{i=1..n}\) are the eigenvectors/eigenvalues of \(\Sigma\)
    
    \(\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n\)

• Larger eigenvalue \(\Rightarrow\) more important eigenvectors
Dimensionality Reduction

Can *ignore* the components of lesser significance

You do *lose some information*, but if the eigenvalues are small, you don’t lose much

– choose only the first $k$ eigenvectors, based on their eigenvalues

– final data set has only $k$ dimensions
PCA

\[ X = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \ldots \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\
\vdots \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \ldots 
\end{bmatrix} \]

\[ Q \] are the eigenvectors of \( \Sigma \); columns are ordered by importance!

\[ Q = \begin{bmatrix}
0.34 & 0.23 & -0.30 & -0.23 & \ldots \\
0.04 & 0.13 & -0.40 & 0.21 & \ldots \\
-0.64 & 0.93 & 0.61 & 0.28 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
-0.20 & -0.83 & 0.78 & -0.93 & \ldots 
\end{bmatrix} \]

\( X \) has \( d \) columns

\( Q \) is \( d \times d \)

Slide by Eric Eaton
PCA

\[ X = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \ldots \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\
\vdots \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \ldots \\
\end{bmatrix} \]

Each row of \( Q \) corresponds to a feature; keep only first \( k \) columns of \( Q \)

\[ Q = \begin{bmatrix}
0.34 & 0.23 & -0.30 & -0.23 & \ldots \\
0.04 & 0.13 & -0.40 & 0.21 & \ldots \\
-0.64 & 0.93 & 0.61 & 0.28 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
-0.20 & -0.83 & 0.78 & -0.93 & \ldots \\
\end{bmatrix} \]
PCA

• Each column of $Q$ gives weights for a linear combination of the original features.

$$Q = \begin{bmatrix}
0.34 & 0.23 & -0.30 & -0.23 & \ldots \\
0.04 & 0.13 & -0.40 & 0.21 & \ldots \\
-0.64 & 0.93 & 0.61 & 0.28 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
-0.20 & -0.83 & 0.78 & -0.93 & \ldots \\
\end{bmatrix}$$

= \ 0.34 \text{ feature1} + 0.04 \text{ feature2} - 0.64 \text{ feature3} + \ldots
We can apply these formulas to get the new representation for each instance $x$

$$X = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \ldots \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\
\vdots \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \ldots
\end{bmatrix} x_3 = \hat{Q} = \begin{bmatrix}
0.34 & 0.23 \\
0.04 & 0.13 \\
-0.64 & 0.93 \\
-0.20 & -0.83
\end{bmatrix}$$

The new 2D representation for $x_3$ is given by:

$$\hat{x}_{31} = 0.34(0) + 0.04(0) - 0.64(1) + \ldots$$

$$\hat{x}_{32} = 0.23(0) + 0.13(0) + 0.93(1) + \ldots$$

The re-projected data matrix is given by $\hat{X} = X\hat{Q}$
PCA Example

Data

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>2.4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>2.2</td>
<td>2.9</td>
</tr>
<tr>
<td>1.9</td>
<td>2.2</td>
</tr>
<tr>
<td>3.1</td>
<td>3.0</td>
</tr>
<tr>
<td>2.3</td>
<td>2.7</td>
</tr>
<tr>
<td>2</td>
<td>1.6</td>
</tr>
<tr>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td>1.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Center Data

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>.69</td>
<td>.49</td>
</tr>
<tr>
<td>-1.31</td>
<td>-1.21</td>
</tr>
<tr>
<td>.39</td>
<td>.99</td>
</tr>
<tr>
<td>.09</td>
<td>.29</td>
</tr>
<tr>
<td>1.29</td>
<td>1.09</td>
</tr>
<tr>
<td>.49</td>
<td>.79</td>
</tr>
<tr>
<td>.19</td>
<td>-.31</td>
</tr>
<tr>
<td>-.81</td>
<td>-.81</td>
</tr>
<tr>
<td>-.31</td>
<td>-.31</td>
</tr>
<tr>
<td>-.71</td>
<td>-1.01</td>
</tr>
</tbody>
</table>

Covariance Matrix

\[
\begin{pmatrix}
0.61655 & 0.61544 \\
0.61544 & 0.71655
\end{pmatrix}
\]

Eigenvectors

\[
\begin{pmatrix}
-0.73518 & -0.67787 \\
0.67787 & -0.73518
\end{pmatrix}
\]

Eigenvalues

| 0.04908 | 0 |
| 1.28403 |
PCA Visualization of MNIST Digits

PCA (16% Variance Expained)
Challenge: Facial Recognition

- Want to identify specific person, based on facial image
- Robust to glasses, lighting,...
  ⇒ Can’t just use the given 256 x 256 pixels
PCA applications - Eigenfaces

• Eigenfaces are the eigenvectors of the covariance matrix of the probability distribution of the vector space of human faces

• Eigenfaces are the ‘standardized face ingredients’ derived from the statistical analysis of many pictures of human faces

• A human face may be considered to be a combination of these standard face ingredients
PCA applications - Eigenfaces

To generate a set of eigenfaces:

1. Large set of digitized images of human faces is taken under the same lighting conditions.
2. The images are normalized to line up the eyes and mouths.
3. The eigenvectors of the covariance matrix of the statistical distribution of face image vectors are then extracted.
4. These eigenvectors are called eigenfaces.
PCA applications - Eigenfaces

- the principal eigenface looks like a bland androgynous average human face

Eigenfaces
Eigenfaces – Face Recognition

• When properly weighted, eigenfaces can be summed together to create an approximate gray-scale rendering of a human face.
• Remarkably few eigenvector terms are needed to give a fair likeness of most people's faces.
• Hence eigenfaces provide a means of applying data compression to faces for identification purposes.
• Similarly, Expert Object Recognition in Video
Eigenfaces

• **Experiment and Results**
  Data used here are from the ORL database of faces. Facial images of 16 people each with 10 views are used.
  - Training set contains $16 \times 7$ images.
  - Test set contains $16 \times 3$ images.

**First three** eigenfaces:
Classification Using Nearest Neighbor

- Save average coefficients for each person. Classify new face as the person with the closest average.
- Recognition accuracy increases with number of eigenfaces until ~15.

Best recognition rates
Training set 99%
Test set 89%
Image Compression
• Divide the original 372x492 image into patches:
  • Each patch is an instance that contains 12x12 pixels on a grid
• View each as a 144-D vector
$L_2$ error and PCA dim
PCA compression: 144D $\rightarrow$ 60D
PCA compression: 144D $\rightarrow$ 16D
16 most important eigenvectors
PCA compression: 144D → 6D
6 most important eigenvectors
PCA compression: 144D $\rightarrow$ 3D
3 most important eigenvectors

![Graph 1](image1)

![Graph 2](image2)

![Graph 3](image3)
PCA compression: 144D $\rightarrow$ 1D
60 most important eigenvectors

Looks like the discrete cosine bases of JPG!...
2D Discrete Cosine Basis