Unsupervised Learning: K-Means & PCA
Unsupervised Learning

• Supervised learning used labeled data pairs \((x, y)\) to learn a function \(f : X \rightarrow Y\)
  – But, what if we don’t have labels?

• No labels = **unsupervised learning**
• Only some points are labeled = **semi-supervised learning**
  – Labels may be expensive to obtain, so we only get a few

• **Clustering** is the unsupervised grouping of data points. It can be used for **knowledge discovery**.
K-Means Clustering

Some material adapted from slides by Andrew Moore, CMU.

Visit http://www.autonlab.org/tutorials/ for Andrew’s repository of Data Mining tutorials.
Clustering Data
K-Means Clustering

K-Means ($k$, $X$)

- Randomly choose $k$ cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster
K-Means Clustering

K-Means \((k, X)\)

- Randomly choose \(k\) cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster
K-Means Clustering

K-Means \( (k, X) \)

- Randomly choose \( k \) cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster
K-Means Animation

Example generated by Andrew Moore using Dan Pelleg’s super-duper fast K-means system:

K-Means Objective Function

- K-means finds a local optimum of the following objective function:

$$\arg \min_{\mathcal{S}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2_2$$

where $\mathcal{S} = \{S_1, \ldots, S_k\}$ is a partitioning over $X = \{x_1, \ldots, x_n\}$ s.t. $X = \bigcup_{i=1}^{k} S_i$ and $\mu_i = \text{mean}(S_i)$
Problems with K-Means

• **Very** sensitive to the initial points
  – Do many runs of K-Means, each with different initial centroids
  – Seed the centroids using a better method than randomly choosing the centroids
    • e.g., Farthest-first sampling

• Must manually choose $k$
  – Learn the optimal $k$ for the clustering
    • Note that this requires a performance measure
Problems with K-Means

• How do you tell it which clustering you want?

Constrained clustering techniques (semi-supervised)

$\text{Same-cluster constraint (must-link)}$

$\text{Different-cluster constraint (cannot-link)}$
Gaussian Mixture Models

• Recall the Gaussian distribution:

\[
P(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp \left( -\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) \right)
\]
The GMM assumption

- There are $k$ components. The $i$'th component is called $\omega_i$
- Component $\omega_i$ has an associated mean vector $\mu_i$
The GMM assumption

- There are k components. The i’th component is called \( \omega_i \)
- Component \( \omega_i \) has an associated mean vector \( \mu_i \)
- Each component generates data from a Gaussian with mean \( \mu_i \) and covariance matrix \( \sigma^2 I \)

Assume that each datapoint is generated according to the following recipe:
The GMM assumption

- There are $k$ components. The $i$'th component is called $\omega_i$.
- Component $\omega_i$ has an associated mean vector $\mu_i$.
- Each component generates data from a Gaussian with mean $\mu_i$ and covariance matrix $\sigma^2 I$.

Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component $i$ with probability $P(\omega_i)$. 

Copyright © 2001, 2004, Andrew W. Moore
The GMM assumption

- There are \( k \) components. The \( i \)’th component is called \( \omega_i \).
- Component \( \omega_i \) has an associated mean vector \( \mu_i \).
- Each component generates data from a Gaussian with mean \( \mu_i \) and covariance matrix \( \sigma^2 I \).

Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component \( i \) with probability \( P(\omega_i) \).

2. Datapoint \( \sim N(\mu_i, \sigma^2 I) \).
The **General** GMM assumption

- There are $k$ components. The $i$’th component is called $\omega_i$.
- Component $\omega_i$ has an associated mean vector $\mu_i$.
- Each component generates data from a Gaussian with mean $\mu_i$ and covariance matrix $\Sigma_i$.

Assume that each datapoint is generated according to the following recipe:

1. Pick a component at random. Choose component $i$ with probability $P(\omega_i)$.
2. Datapoint $\sim N(\mu_i, \Sigma_i)$.
Fitting a Gaussian Mixture Model

(Optional)
Expectation-Maximization for GMMs

Iterate until convergence:

On the $t'$ th iteration let our estimates be

$$\lambda_t = \{ \mu_1(t), \mu_2(t) ... \mu_c(t) \}$$

E-step: Compute “expected” classes of all datapoints for each class

$$P(w_i | x_k, \lambda_t) = \frac{p(x_k | w_i, \lambda_t)p(w_i | \lambda_t)}{p(x_k | \lambda_t)} = \frac{p(x_k | w_i, \mu_i(t), \sigma^2 I)p_i(t)}{\sum_{j=1}^c p(x_k | w_j, \mu_j(t), \sigma^2 I)p_j(t)}$$

M-step: Estimate $\mu$ given our data’s class membership distributions

$$\mu_i(t + 1) = \frac{\sum_k P(w_i | x_k, \lambda_t)x_k}{\sum_k P(w_i | x_k, \lambda_t)}$$

Just evaluate a Gaussian at $x_k$
Iterate. On the \( t \) th iteration let our estimates be
\[
\lambda_t = \{ \mu_1(t), \mu_2(t) \ldots \mu_c(t), \Sigma_1(t), \Sigma_2(t) \ldots \Sigma_c(t), p_1(t), p_2(t) \ldots p_c(t) \}
\]

E-step: Compute “expected” clusters of all datapoints
\[
P(w_i | x_k, \lambda_t) = \frac{p(x_k | w_i, \lambda_t)P(w_i | \lambda_t)}{p(x_k | \lambda_t)} = \frac{p(x_k | w_i, \mu_i(t), \Sigma_i(t))p_i(t)}{\sum_{j=1}^c p(x_k | w_j, \mu_j(t), \Sigma_j(t))p_j(t)}
\]

M-step: Estimate \( \mu, \Sigma \) given our data’s class membership distributions
\[
\mu_i(t + 1) = \frac{\sum_k P(w_i | x_k, \lambda_t) x_k}{\sum_k P(w_i | x_k, \lambda_t)}
\]
\[
\Sigma_i(t + 1) = \frac{\sum_k P(w_i | x_k, \lambda_t)[x_k - \mu_i(t + 1)][x_k - \mu_i(t + 1)^T]}{\sum_k P(w_i | x_k, \lambda_t)}
\]
\[
p_i(t + 1) = \frac{\sum_k P(w_i | x_k, \lambda_t)}{R}
\]

\( p_i(t) \) is shorthand for estimate of \( P(\omega_i) \) on \( t \) th iteration

Just evaluate a Gaussian at \( x_k \)

\( R = \# \text{records} \)
(End optional section)
Gaussian Mixture Example: Start

Advance apologies: in Black and White this example will be incomprehensible
After first iteration
After 2nd iteration
After 3rd iteration
After 4th iteration
After 5th iteration
After 6th iteration
After 20th iteration
Some Bio Assay data
GMM clustering of the assay data
Resulting Density Estimator
Principal Components Analysis

Based on slides by Barnabás Póczos, UAlberta
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>

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
1\textsuperscript{st} PCA axis
2\textsuperscript{nd} PCA axis
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 Algorithm

- Given data \( \{x_1, \ldots, x_n\} \), 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 \( \Sigma = XX^\top \)

- **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 \( \Sigma \)
    - \( \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \)
  - Larger eigenvalue \( \Rightarrow \) more important eigenvectors
PCA

\[ X = \begin{bmatrix}
  0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \ldots \\
  1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \ldots \\
  0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\
  \vdots \\
  1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \ldots 
\end{bmatrix} \begin{bmatrix} x_3 \end{bmatrix} \]

Columns are ordered by importance!

\[ Q = \begin{bmatrix}
  0.34 & 0.23 & -0.30 & -0.23 & \ldots \\
  0.04 & 0.13 & -0.40 & 0.21 & \ldots \\
  -0.64 & 0.93 & 0.61 & 0.28 & \ldots \\
  \vdots & \vdots & \vdots & \vdots & \vdots \\
  -0.20 & -0.83 & 0.78 & -0.93 & \ldots 
\end{bmatrix} \begin{bmatrix} x_3 \end{bmatrix} \]
PCA

\[ X = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \ldots \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \ldots \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\
\vdots \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \ldots 
\end{bmatrix} \quad x_3 \]

Keep only first \( k \) columns of \( Q \)

\[ Q = \begin{bmatrix}
0.34 & 0.23 & -0.30 & -0.23 & \ldots \\
0.04 & 0.13 & -0.40 & 0.21 & \ldots \\
-0.64 & 0.93 & 0.61 & 0.28 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
-0.20 & -0.83 & 0.78 & -0.93 & \ldots 
\end{bmatrix} \quad x_3 \]
PCA Visualization of MNIST Digits
Challenge: Facial Recognition

- Want to identify specific person, based on facial image
- Robust to glasses, lighting,...
  \[\Rightarrow\] 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 faces.
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 – 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 persons 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 till 15. Later eigenfaces do not help much with recognition.

Best recognition rates
Training set 99%
Test set 89%
Facial Expression Recognition: Happiness subspace
Disgust subspace
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 $\Rightarrow$ 6D
6 most important eigenvectors
PCA compression: 144D $\Rightarrow$ 3D
3 most important eigenvectors
PCA compression: 144D $\Rightarrow$ 1D
60 most important eigenvectors

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

Denoised image using 15 PCA components