Principal Component Analysis

CS434
Unsupervised dimensionality reduction

• Consider a collection of data points in a high dimensional feature space (e.g., 5000-d)
  – Try to find a more compact data representation
  – Create new features defined as functions over all of the original features

• Why?
  – Visualization: need to display low-dimensional version of a data set for visual inspection
  – Preprocessing: learning algorithms (supervised and unsupervised) often work better with smaller numbers of features both in terms of runtime and accuracy (why?)
Principal Component Analysis

• A classic dimensionality reduction technique
• It linearly projects n-dimensional data onto a $k$-dimensional space while preserving information (assuming $k$ is given):
  – e.g., project space of 10k words onto a 3d space
• How to preserve information?
  – Suppose we have two features $f_1$ and $f_2$, and we can only keep one
  – For $f_1$, most examples have similar value (small variance)
  – For $f_2$, most examples differ from each other
  – Which one to keep?
    • $f_2$ because it retains information about the data items
• Basic idea for PCA: find a linear projection that retains the most information (variance) in data
First, what is a linear projection

1. A linear projection can be viewed as defining a new axis which is the result of **rotating** existing ones
2. It can be used with **translation** – moving the origin of the coordinate system.

\[ z = (x - \bar{x})^T u \]
A Conceptual Algorithm

• Find a line such that when data is projected to that line, it has the maximum variance
• the variance of the projected data is considered as retained by the projection, the rest is lost
Conceptual Algorithm

• Once you have found the first projection line, we continue to search for the next projection line by:
  finding a new line, orthogonal to the first, that has maximum projected variance:

In this case, we have to stop after two iterations, because the original data is 2-d.

But you can imagine this procedure being continued for higher dimensional data.
Repeat Until k Lines

• The projected position of a point on these lines gives the coordinates in the new (reduced) $k$-d space

How can we compute this set of projection lines?
Basic PCA algorithm

• Start from n by d data matrix: \( X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \)

• Compute the center of the data: \( \mu = \frac{1}{n} \sum_{i=1}^{n} x_i \)

• Compute the Covariance matrix

\[
\Sigma = \frac{1}{n} \sum_{i} (x_i - \mu) (x_i - \mu)^T
\]

• Compute the eigen-vectors and eigen-values of \( \Sigma \)

\( \Sigma v_j = \lambda_j v_j, \ j=1, 2, \ldots k \)

\( \lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots \)

• \( v_1, v_2, \ldots, v_k \) are the projection directions
Example: Face Recognition

• An typical image of size 256 x 128 is described by \( n = 256 \times 128 = 32768 \) dimensions – each dimension described by a grayscale value
• Each face image lies somewhere in this high-dimensional space
• Images of faces are generally similar in overall configuration, thus
  – They should not be randomly distributed in this space
  – We should be able to describe them in a much lower dimensional space
PCA for Face Images: Eigen-faces

• Database of 128 carefully-aligned faces.

• Here are the mean and the first 15 eigenvectors.

• Each eigenvector (32768 –d vector) can be shown as an image – each element is a pixel on the image.

• These images are face-like, thus called eigen-faces
Face Recognition in Eigenface space
(Turk and Pentland 1991)

• Nearest Neighbor classifier in the eigenface space
• Training set always contains 16 face images of 16 people, all taken under the same set of conditions of lighting, head orientation and image size
• Accuracy:
  – variation in lighting: 96%
  – variation in orientation: 85%
  – variation in image size: 64%
Face Image Retrieval

- Left-top image is the query image
- Return 15 nearest neighbor in the eigenface space
- Able to find the same person despite
  - different expressions
  - variations such as glasses