Dimension Reduction: 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

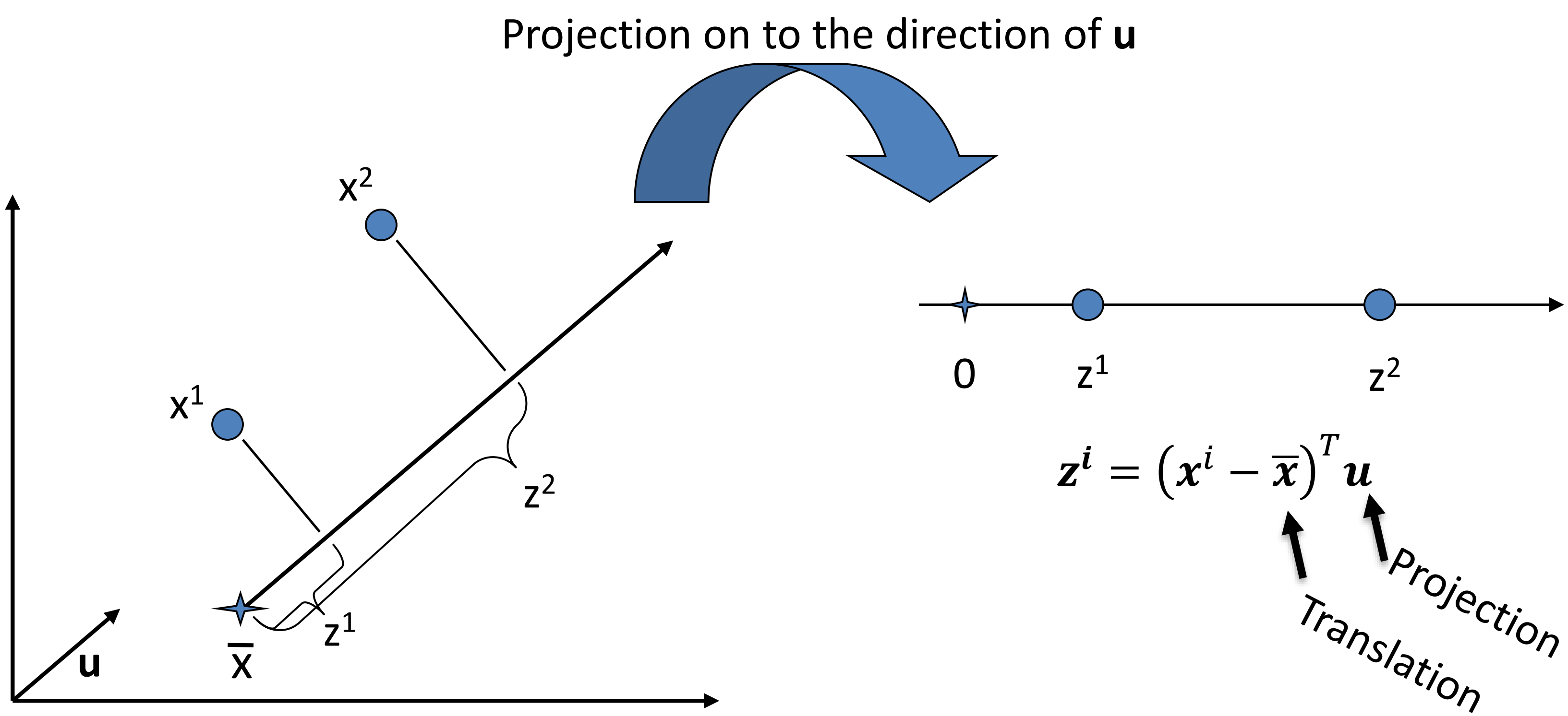
- A classic dimensionality reduction technique
- It linearly projects n-dimensional data onto a k-dimensional space while preserving information (assuming k is given):
  - e.g., project space of 10k words onto a 3-dimensional space
- How to preserve information?
  - Suppose we have two features f_1 and 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

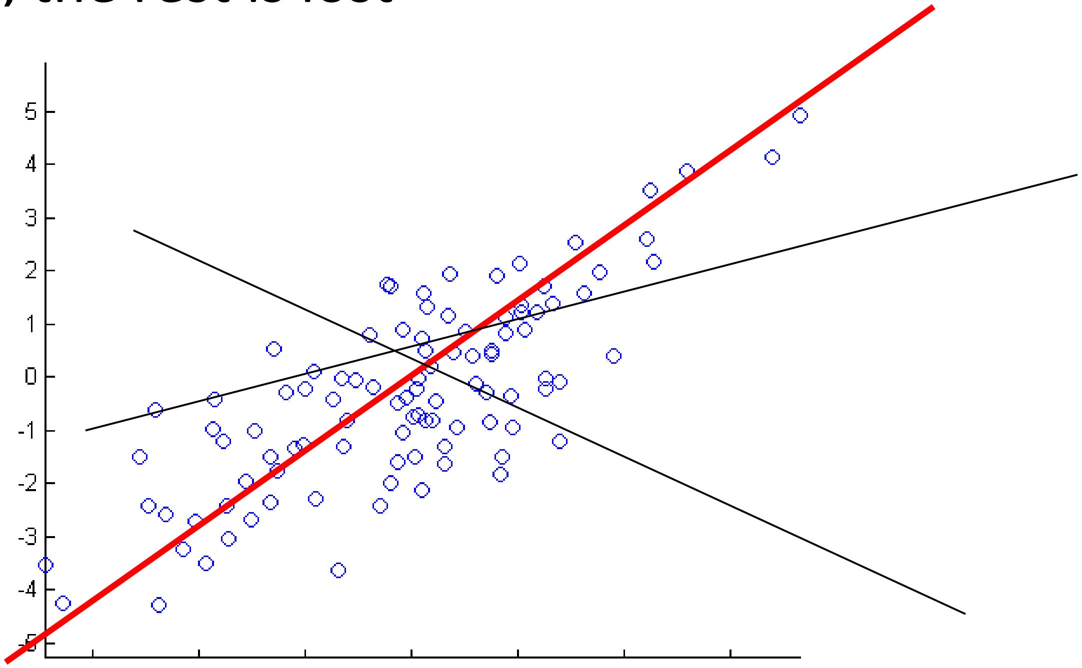
1. A linear projection with a projection vector $\mathbf{u}$ can be viewed as rotating the co-ordinate system to align the axis with $\mathbf{u}$
2. It can be used with or without ***translation*** – moving the origin of the coordinate system.



u is a unit length vector

A Conceptual Algorithm

- Find a line such that when data is projected to that line, we preserve the maximum variance
- the variance of the projected data is considered as retained by the projection, the rest is lost



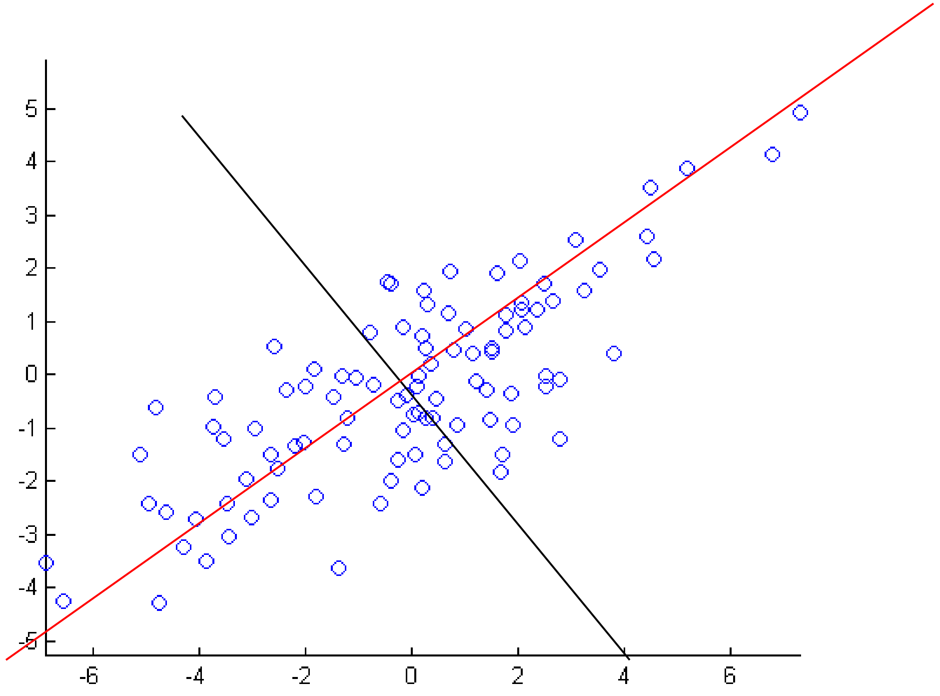
Conceptual Algorithm

- 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:

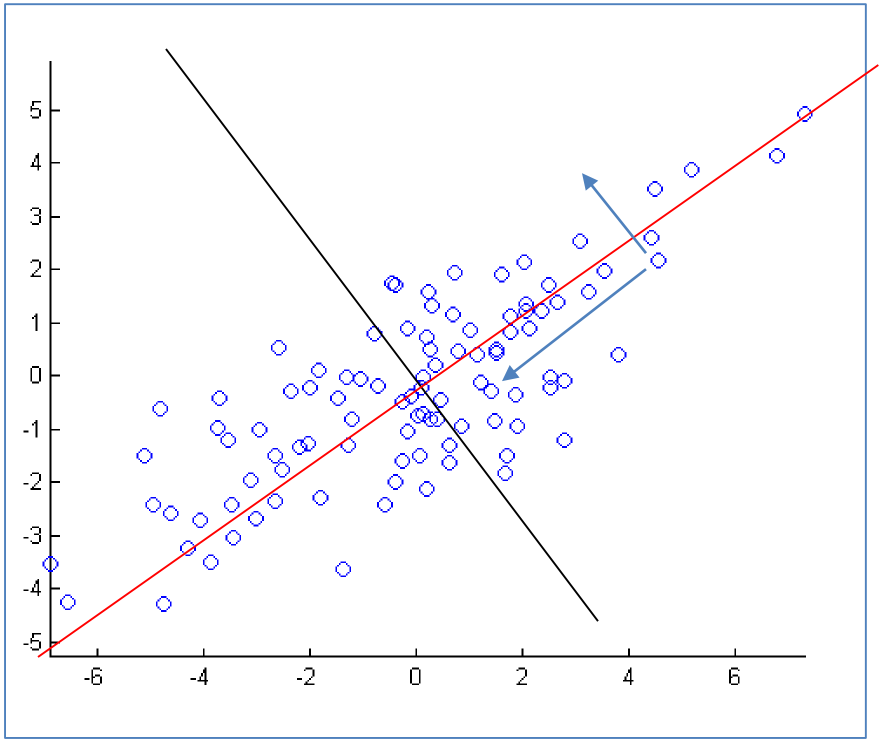
In this case, we have to stop after two iterations, because the original data is 2-d.

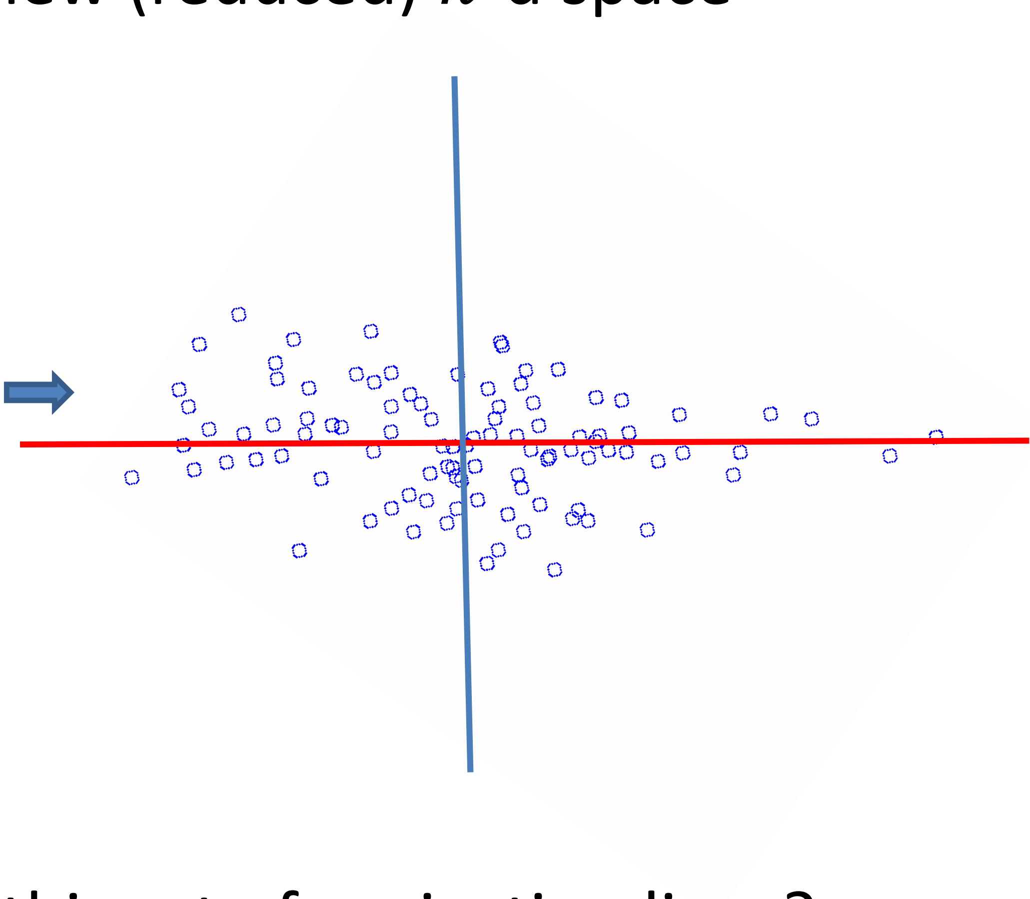
But you can imagine this procedure being continued for higher dimensional data.



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: $X = \begin{bmatrix} x_1^T \\ \dots \\ x_n^T \end{bmatrix}$
  - Each row is a data point, each column is dimension
- Compute the center of the data: $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Compute the Covariance matrix

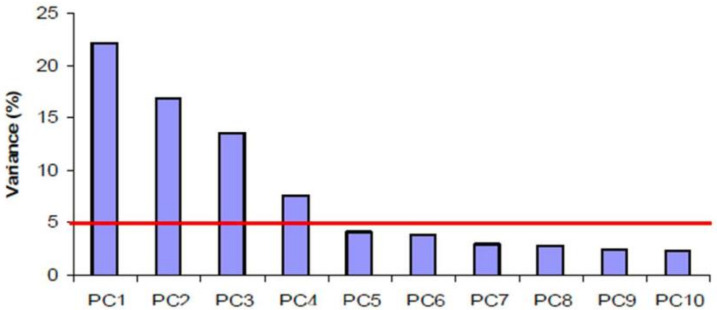
$$\Sigma = \frac{1}{n}\sum_{i}(x_i - \mu)(x_i - \mu)^T$$

- Compute the eigen-vectors and eigen-values of Σ
  - An eigen vector of Σ satisfies: $\Sigma\,\boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j$ where λ_j is its eigen-value
- Sort the Eigenvalues in decreasing order $\lambda_1 \geq \lambda_2 \geq \lambda_3 \dots$
- $v_1, v_2, \dots, v_k$ are the top k PCA projection directions

Dimension Reduction Using PCA

- Given data, pack it into $n \times d$ matrix
  - Rows correspond to examples, columns correspond to features
- Compute the $d \times d$ covariance matrix Σ
- Calculate the eigen vectors/values of Σ
- Rank the eigen values in decreasing order
  - i-th eigen value = the variance of data after projecting onto i-th eigenvector
  - Choose the highest -> retain the most variance
- Select the top k eigenvectors
- If we don't have a fixed k, choose k to be the smallest k such that $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} > \alpha$

  where α is a threshold, e.g., 85%



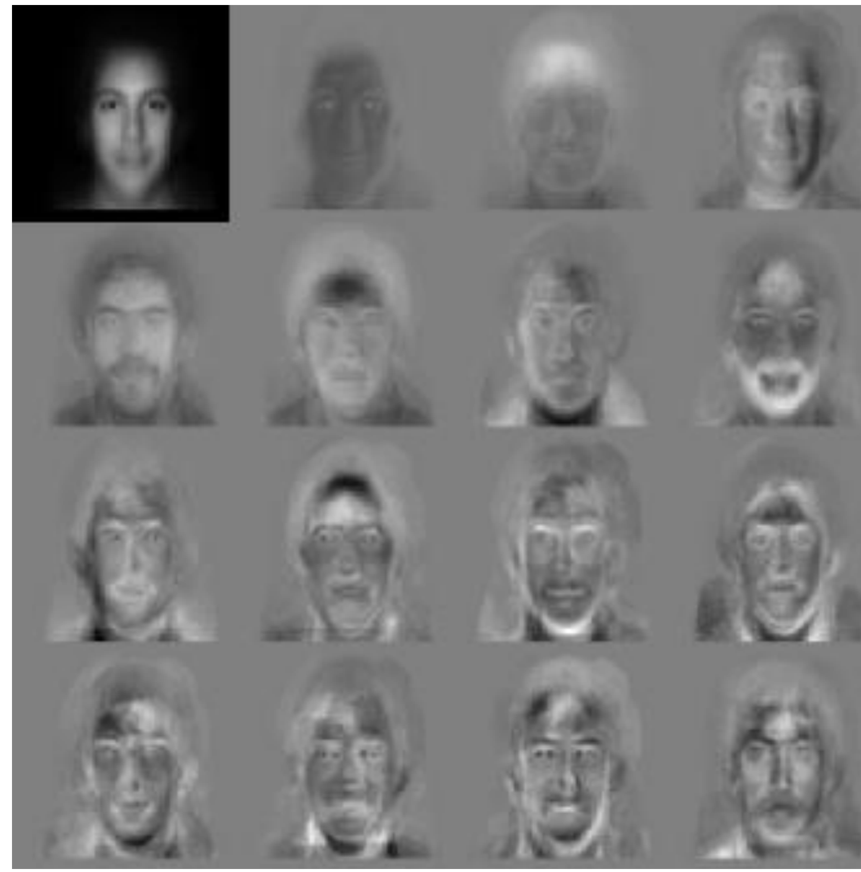
You might loose some info. But if the eigenvalues are small, not much is lost.

Example: Face Recognition

- An typical image of size 256 x 128 is described by n = 256x128 = 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

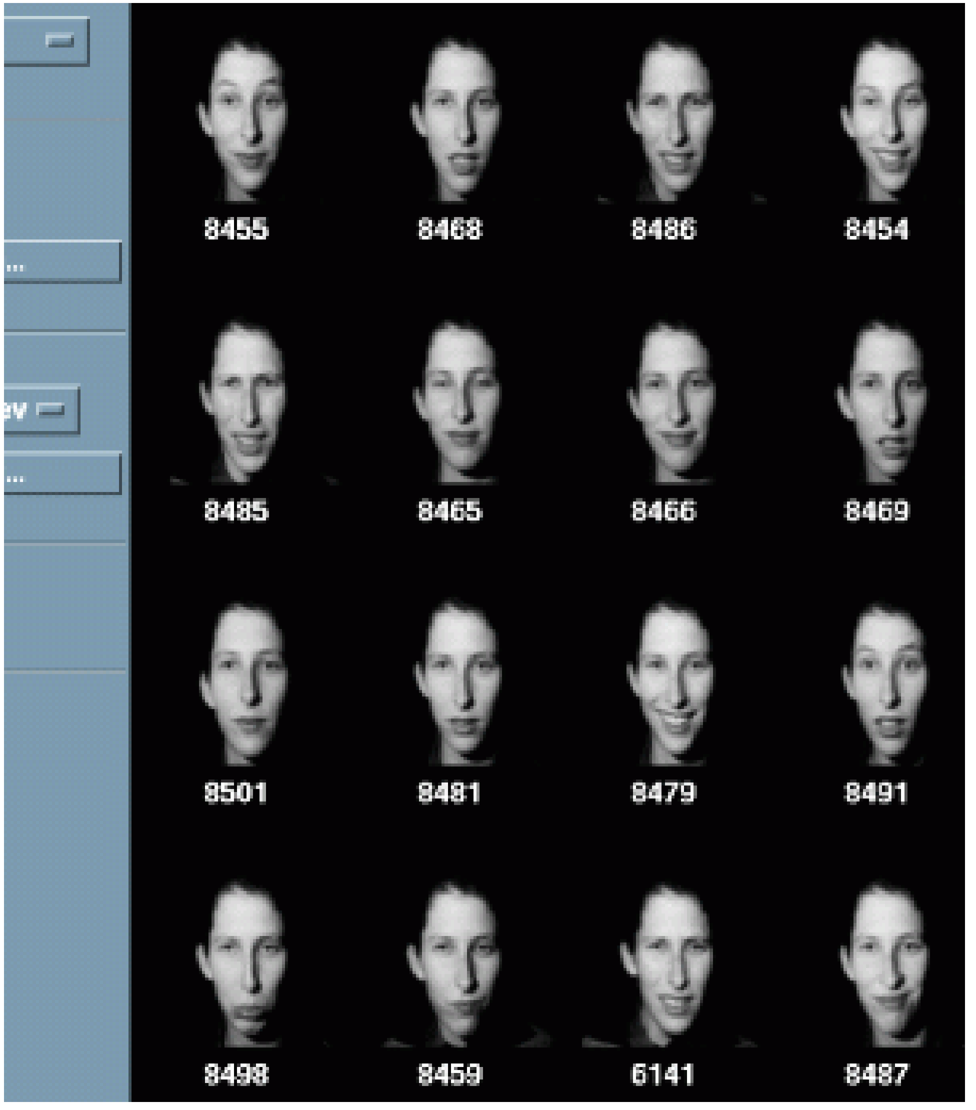


Face Recognition in Eigenface space
(Turk and Pentland 1991)

- 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%

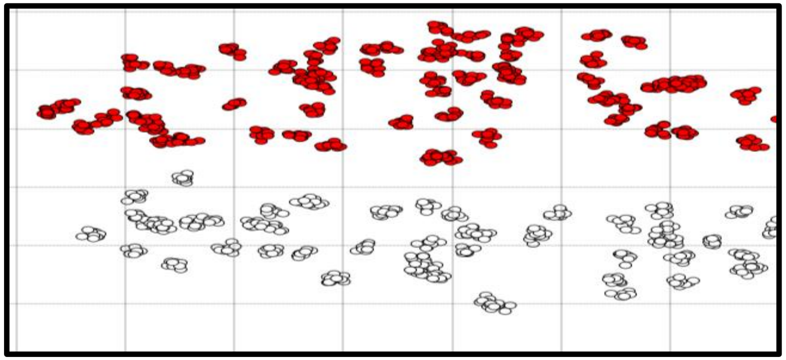
Face Image Retrieval

- 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



Summary on PCA

- An unsupervised dimension reduction
  - Do not use class labels
  - Goal is to maximize variance after reduction
- PCA is useful for various reasons
  - Reducing dimension can reduce overfitting
  - Reducing dimension reduces computational complexity
  - It can be used to reduce noise in data
  - After PCA, the projected features becomes uncorrelated
- Possible downfall: completely unsupervised, if the class separation is not along the large variance direction, PCA can lead to loss of separation between classes.



For this example, applying PCA to reduce to 1 dimension will choose the x axis, and loose class separation