CAP 5610: Machine Learning

Lecture 10:
Dimensionality Reduction

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

Reference Materials From: GuoJun Qi

Reading

Marsland: ML Chapter 6

Bishop: PRML Chapter 12

Murphy: PA Chapter 12

Announcements

Midterm exam in class: Oct 22th

- Closed book, 1 letter sized sheet of notes
- Review in class on Oct 17th

Final project proposals to be due shortly after the midterm exam

Final homework to be due mid November

Final Project

Technical report similar in scope to CS conference paper OR literature review

Final project presentations (5 mins) will start after Thanksgiving.

Proposal should include:

- Project partners
- Key reference papers (1-2 citations)
- Dataset
- Software
- Benchmark to beat
- Evaluation methodology

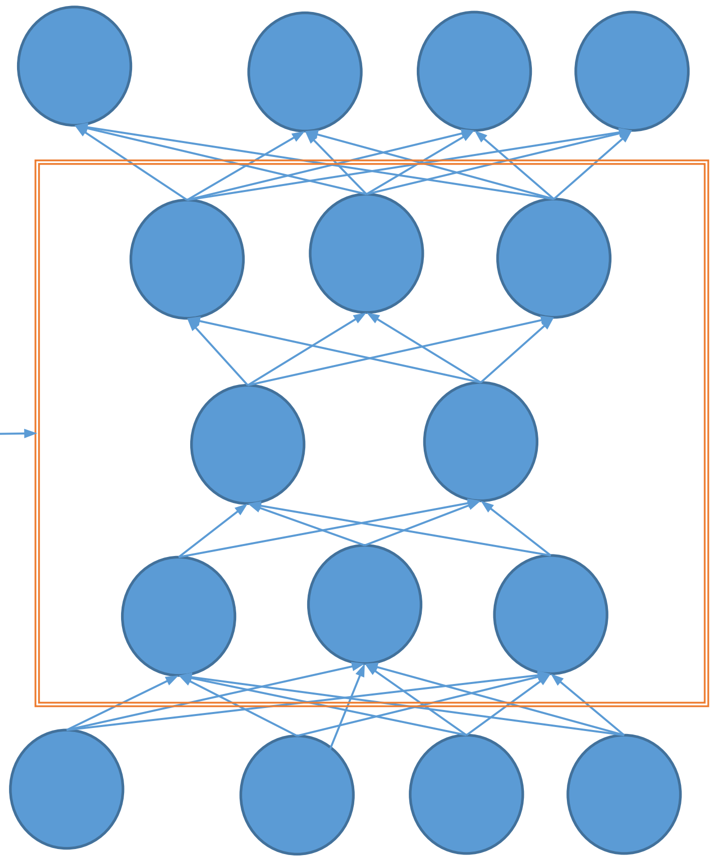
Learning more compact feature representations

- A compact low-dimensional feature representation is preferred in many applications
  - Avoid the curse of high dimensionality
    - Fewer training examples are required to achieve a certain level of accuracy
  - Less computationally demanding
- Autoencoder is a low dimensional feature representation

Autoencoder

- Using fewer neurons to reconstruct the input at the output layer

Low dimensional representations at hidden layers



Two types of dimensionality reduction methods

- Unsupervised learning of lower dimensional representation
  - No label information is available on each example
  - Linear model
    - Principal Component Analysis (PCA)
    - Independent Component Analysis (ICA)
    - Canonical Correlation Analysis (CCA)
  - Nonlinear model
    - Autoencoder
    - Kernel PCA
- Supervised learning of lower dimensional representation
  - Fisher linear discriminant

Principal Component Analysis (PCA)

- Projecting the original feature vectors into a lower dimensional subspace, where the squared distances between the projected points and the original vectors are minimized
  - Preserve as much information as possible
  - E.g., find a plan to approximate the feature vectors in 3D space

PCA vs. Autoencoder

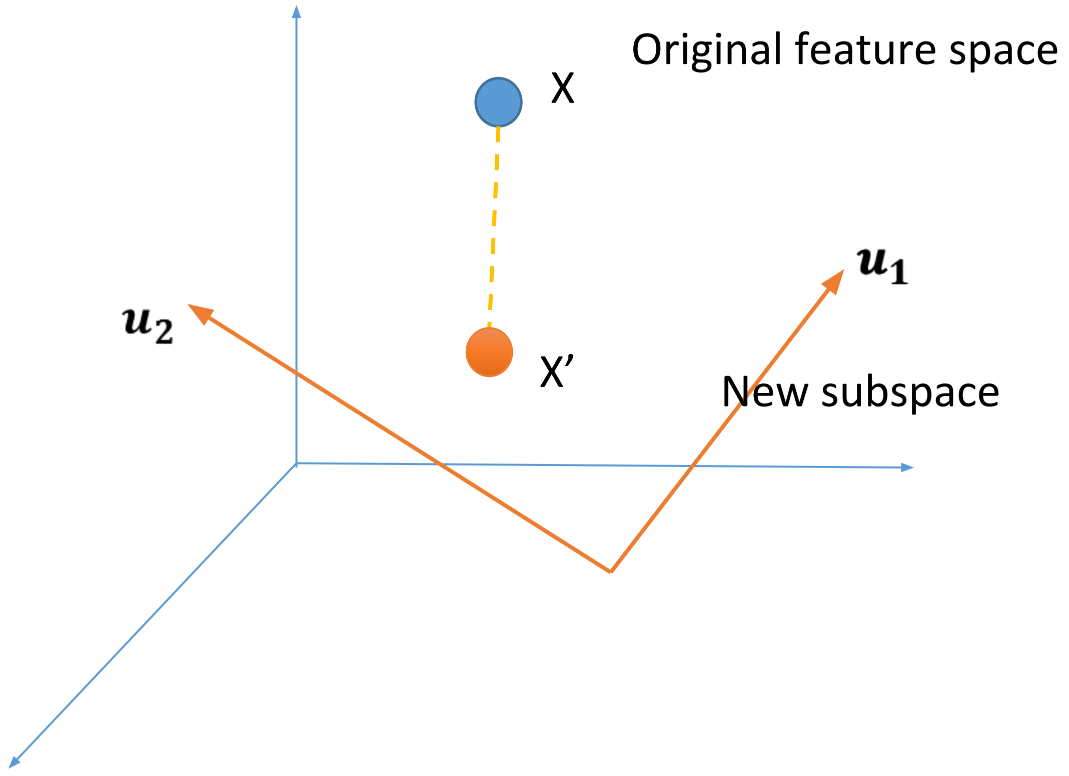
- **PCA** uses a linear transformation to map the feature vectors into a lower dimensional representation
- Its objective function has global closed-form solution
- **Autoencoder** representation is nonlinear
- It has many local optimum solutions.

PCA

- Given a data X of d dimensional feature vector, learns k-dimensional representation, in which
  - They are orthogonal

$$X' = \sum_{i=1}^{k} z_i \boldsymbol{u_i}$$

where $\boldsymbol{u_i}$ are the orthogonal bases that span the new subspace, and z_i is the reconstruction coefficients.



PCA formulation

- Suppose we are given a set of data vectors, with nth vector as
$$\mathbf{x}^n = (x_1^n, x_2^n, \dots, x_d^n)$$
- PCA will represent these vectors in a new k-dimensional subspace spanned by $\{\boldsymbol{u_i} | i = 1, \dots, k\}$, i.e.,

$$\widehat{\mathbf{x}^n} = \mathbf{m} + \sum_{i=1}^{k} z_i^n \boldsymbol{u}_i$$

where m is the bias vector, usually set to the mean of data vectors, and all $\boldsymbol{u_i}$ are orthonormal vectors (unit and orthogonal)

PCA Objective Function

- Given k<d, find $\{\boldsymbol{u_i} | i = 1, \dots, k\}$ that minimizes the reconstruction error

$$E_k = \sum_{n=1}^{N} ||\widehat{\mathbf{x}^n} - \mathbf{x}^n||^2$$

where

$$\widehat{\mathbf{x}^n} = \mathbf{m} + \sum_{i=1}^{k} z_i^n \boldsymbol{u}_i$$

PCA

- Given $\{\boldsymbol{u_i} | i = 1, \dots, k\}$ of k orthonormal bases (principal components), we can find another d-k orthonormal vectors$\{\boldsymbol{u_i} | i = k + 1, \dots, d\}$ (complement components), so that they constitute a complete set of bases for d-dimensional feature space
- Any original vector $\mathbf{x}^n$ can be represented by this complete set of bases as

$$\mathbf{x}^n = \mathbf{m} + \sum_{i=1}^{d} z_i^n \boldsymbol{u}_i$$

PCA

- Reconstruction error:

$$\left.\begin{aligned}\mathbf{x}^n &= \mathbf{m} + \sum_{i=1}^{d} z_i^n \boldsymbol{u}_i \\ \widehat{\mathbf{x}^n} &= \mathbf{m} + \sum_{i=1}^{k} z_i^n \boldsymbol{u}_i\end{aligned}\right\} \mathbf{x}^n - \widehat{\mathbf{x}^n} = \sum_{i=k+1}^{d} z_i^n \boldsymbol{u}_i$$

PCA

- Reconstruction error

$$\mathbf{x}^n - \widehat{\mathbf{x}^n} = \sum_{i=k+1}^{d} z_i^n \boldsymbol{u}_i$$

- Squared reconstruction error

$$E_k = \left\|\mathbf{x}^n - \widehat{\mathbf{x}^n}\right\|^2 = \sum_{i=k+1}^{d} (z_i^n)^2$$

Finding the reconstruction coefficients

$$\mathbf{x}^n = \mathbf{m} + \sum_{i=1}^{k} z_i^n \boldsymbol{u}_i$$

By the orthonormality, subtracting **m**, and multiplying $\boldsymbol{u}_i^T$ with both sides, we have

$$z_i^n = \boldsymbol{u}_i^T(\mathbf{x}^n - \mathbf{m})$$

$$E_k = \left\|\mathbf{x}^n - \widehat{\mathbf{x}^n}\right\|^2 = \sum_{i=k+1}^{d} (z_i^n)^2 = \sum_{i=k+1}^{d} \left(\boldsymbol{u}_i^T(\mathbf{x}^n - \mathbf{m})\right)^2$$

PCA

- Reconstruction error

$$E_k = \left\|\mathbf{x}^n - \widehat{\mathbf{x}^n}\right\|^2 = \sum_{i=k+1}^{d} \left(\boldsymbol{u}_i^T(\mathbf{x}^n - \mathbf{m})\right)^2 = \sum_{i=k+1}^{d} \boldsymbol{u}_i^T \Sigma \boldsymbol{u}_i$$

where matrix $\Sigma = \sum_{n=1}^{N}(\mathbf{x}^n - \mathbf{m})(\mathbf{x}^n - \mathbf{m})^T$ is covariance matrix of the input data vectors.

PCA

- Objective function

$$\min \sum_{i=k+1}^{d} \boldsymbol{u}_i^T \Sigma \boldsymbol{u}_i$$

Subject to orthonormal constraints on $\boldsymbol{u}_i$

Recap: Singular Vector Decomposition

- Given a positive semi-definite matrix Σ, it has the following form of decomposition:

$$\Sigma = \sum_{i=1}^{d} \rho_i \mathbf{v}_i \mathbf{v}_i^T = \mathbf{V}\Psi\mathbf{V}^T$$

Where . ρ_i is called eigenvalue, and $\mathbf{v}_i$ is the eigenvector. V is the matrix with all eigenvectors, and Ψ is the diagonal matrix with eigenvalues.

All $\mathbf{v}_i$ form a complete set of orthonormal bases for d dimensional space. Because $\boldsymbol{u}_i$ belongs to d dimensional space, we have

$$\boldsymbol{u}_i = \mathbf{V}\boldsymbol{c}_i$$

PCA

- $$\min \sum_{i=k+1}^{d} \boldsymbol{u}_i^T \Sigma \boldsymbol{u}_i = \sum_{i=k+1}^{d} \boldsymbol{c}_i^T \boldsymbol{V}^T \Sigma \boldsymbol{V} \boldsymbol{c}_i = \sum_{i=k+1}^{d} \boldsymbol{c}_i^T \Psi \boldsymbol{c}_i$$

- Orthonormal constraints $\boldsymbol{u}_i^T \boldsymbol{u}_j = \boldsymbol{c}_i^T \boldsymbol{c}_i$, becomes constraints on $\boldsymbol{c}_i$

- The minimum possible value is $\sum_{i=k+1}^{d} \rho_i$, i.e., the sum of the d-k smallest eigenvalues.
- $\boldsymbol{u}_i$ are set to the corresponding eigenvectors of covariance matrix Σ.

PCA Algorithm

1. create a N by d matrix X, with each column vector $\mathbf{x}^n$ per data point;
2. X←subtract mean m from each column vector in X
3. Compute covariance matrix $\Sigma = XX^T$
4. Find eigenvectors and eigenvalues of Σ
5. Set *k* principal components to the eigenvectors with the largest k eigenvalues

Explanation

- PCA representation

$$\widehat{\mathbf{x}^n} = \mathbf{m} + \sum_{i=1}^{k} z_i^n \boldsymbol{u}_i$$

- Each $\boldsymbol{u}_i$ represents an atomic building block that can be combined to reconstruct the original vector.
- Each z_i^n is a coefficient for combining these components

Problems

- What if dimension d is super large?
  - For an image of 256X256 pixels, d = 65536
  - Covariance matrix is of 65536 X 65536, expensive to find SVD
- If the size of dataset is relatively small, then we have an alternative approach to find SVD of covariance matrix

Auxiliary matrix $L = X^T X$ of size N^2 Eigenvector $\boldsymbol{v}$ of L

$$\Downarrow$$

Covariance matrix $\Sigma = X X^T$ of size d^2 Eigenvector $X\boldsymbol{v}$ of Σ

Proof

- Suppose $L\mathbf{v} = \lambda\mathbf{v}$ is the eigenvector decomposition of *L, i.e.,*

$X^T X\mathbf{v} = \lambda\mathbf{v}$

Multiplying both sides with X, we have

$XX^T X\mathbf{v} = \lambda X\mathbf{v}$

Then, $\Sigma(X\mathbf{v}) = \lambda(X\mathbf{v})$, so

$X\mathbf{v}$ is the eigenvector of Σ

(subject to Xv is not zero, i.e., v dose not belong to null space of X)

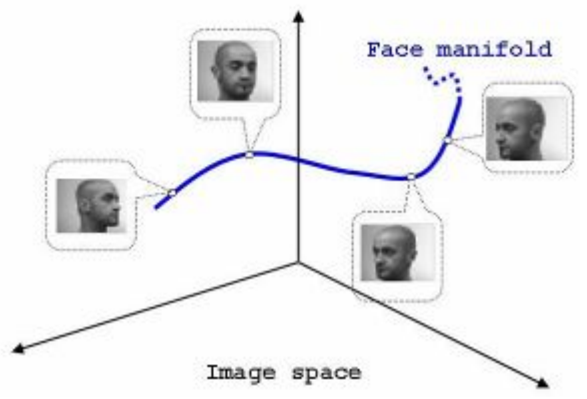
Application: PCA for Face Recognition

- Each person has been taken in different conditions
    - Expression – angry, happy, disgust
    - With/without glasses

Why PCA?

- The change of face images in a high dimensional space is not arbitrary
    - Often the feature vectors corresponding to valid face images can only change within a particular subspace

    - In a low dimensional subspace, each dimension may correspond to a distinct condition under which these faces are taken.
        - Left figure shows a one dimensional subspace within which the faces are taken in changing angles.

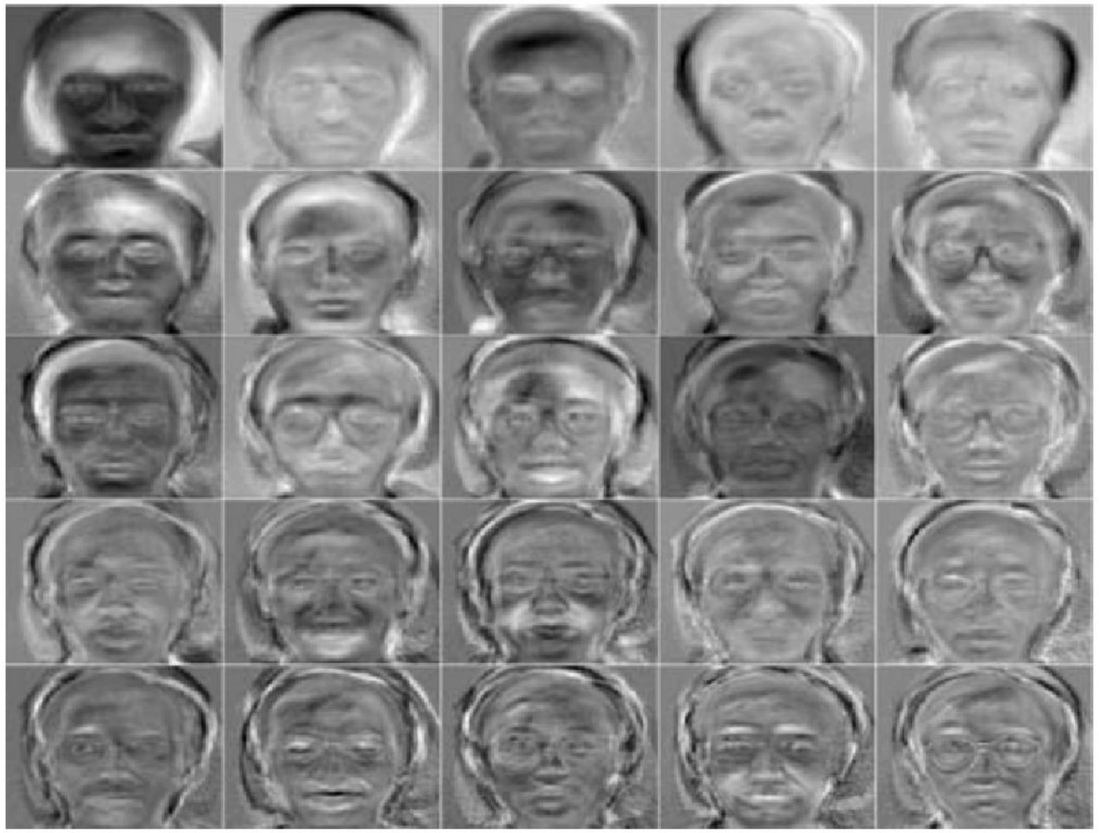


How to use PCA to recognize faces?

- Method A:
  - For each person, construct a subspace by PCA.
  - Given a test face, map it into these subspaces, and assign the face to the subspace with the minimal distance (reconstruction error as in objective function)
- Method B:
  - For all faces independent of people, construct a subspace by PCA
  - Using the reconstruction coefficients as a feature vector for each face
  - Applying existing classification algorithms (KNN, naive Bayes)
  - PCA is often used as a preprocessing procedure for naive Bayes

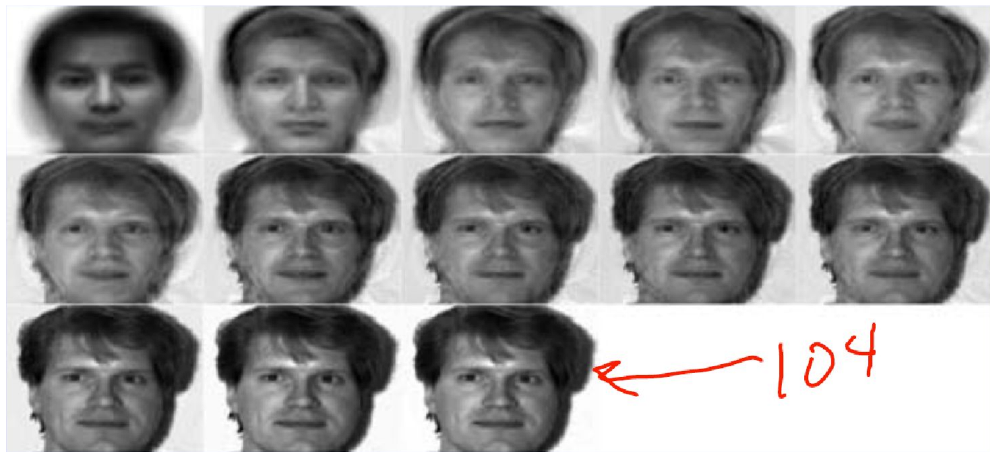
Discovered principal components

- Eigenface



Reconstructed Faces

- The first is the mean face
- Everytime, 8 new components are added to reconstruct it.



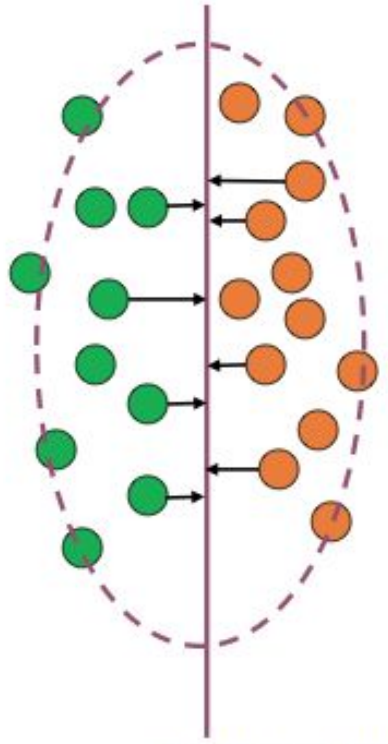
https://www.youtube.com/watch?v=jQOZrXZTXcw

Supervised vs. Unsupervised Dimensionality Reduction

- Unsupervised dimensionality reduction
  - There is no labeled data in the input dataset
  - Principal component analysis – find the most significant factors that account for the change of data points in a feature space
  - Criterion: minimizing the reconstruction error between the projected points in the subspace and the original points
- Supervised:
  - Data is labeled in the input dataset
  - Fisher Discriminant Analysis
  - Closely related to ANOVA

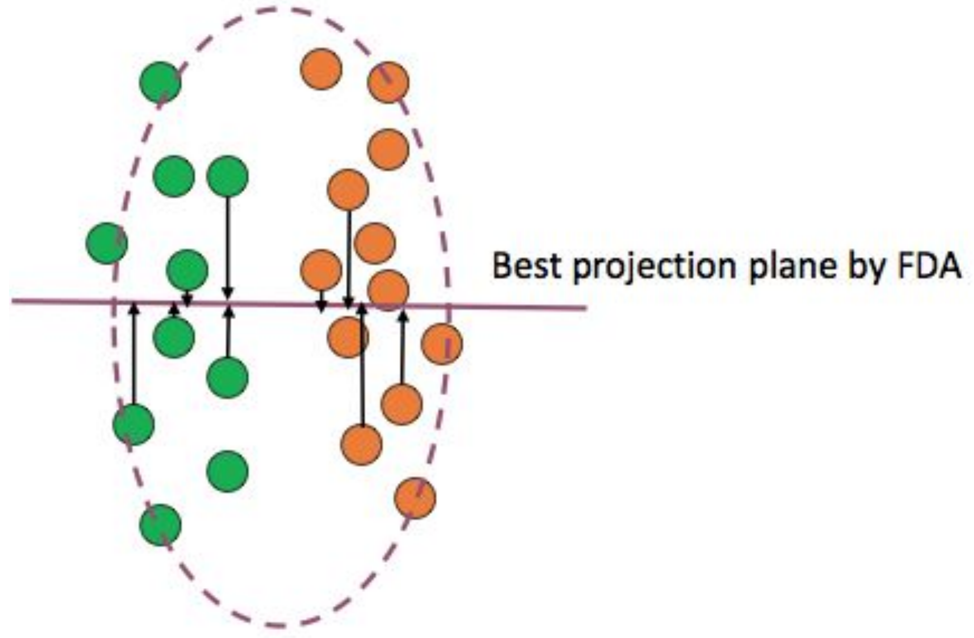
PCA vs. FDA

- PCA finds the most significant direction along which the data changes much
  - The direction along which the data points have a large variance.
- PCA may fail to distinguish between different classes
  - All data points are mixed.



PCA vs. FDA

- Fisher discriminant analysis instead finds the projection plane within which the data points from different classes are separated as much as possible.
- Multi-class FDA finds k-1 projection planes.



FDA Objectives

- Suppose we are given a set of data vectors, with nth vector as
$$\mathbf{x}^n = (x_1^n, x_2^n, \dots, x_d^n)$$
- Each $\mathbf{x}^n$ belongs to one of *k* classes $\{C_1, C_2, \dots, C_k\}$
- FDA projects $\mathbf{x}^n$ into a subspace by a linear transformation
$$\boldsymbol{y}^n = \boldsymbol{w}^T\mathbf{x}^n$$
- FDA finds the best linear transformation that maximizes the separability of the projected points $\{\boldsymbol{y}^n\}$

Measure of Between-Class Separability

- Maximizing the distance between the mean vectors of different classes
    - Mean vector of class i in the original feature space

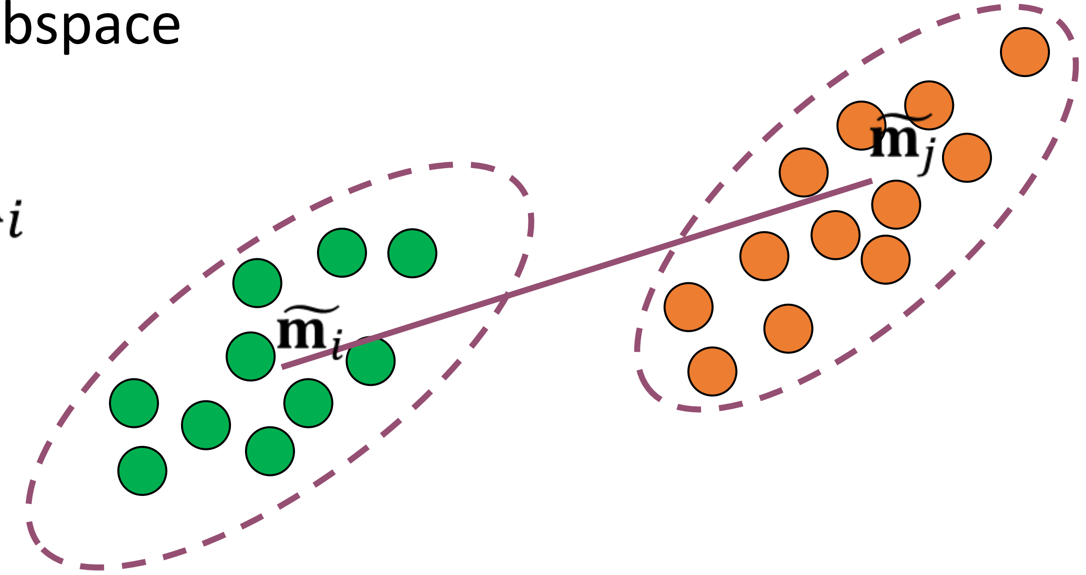
$$\mathbf{m}_i = \frac{1}{N_i}\sum_{n\in C_i}\mathbf{x}^n$$

    - Mean vector of class i in the projection subspace

$$\widetilde{\mathbf{m}_i} = \frac{1}{N_i}\sum_{n\in C_i}\boldsymbol{w}^T\mathbf{x}^n = \boldsymbol{w}^T\mathbf{m}_i$$

    - Maximizing

$$\sum_{i\neq j}\left\|\widetilde{\mathbf{m}_i} - \widetilde{\mathbf{m}_j}\right\|^2$$



Between-Class Scatter

- $\sum_{i\neq j}\left\|\widetilde{\mathbf{m}}_i - \widetilde{\mathbf{m}}_j\right\|^2 = \sum_{i\neq j}\left\|\boldsymbol{w}^T\mathbf{m}_i - \boldsymbol{w}^T\mathbf{m}_j\right\|^2$

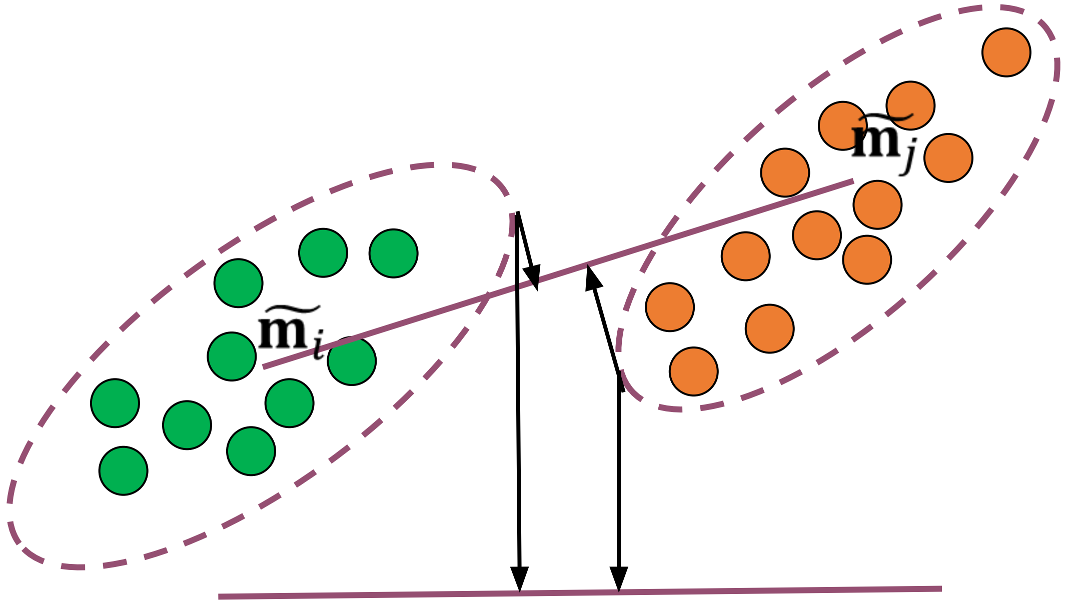
$$= \sum_{i\neq j}\boldsymbol{w}^T(\mathbf{m}_i - \mathbf{m}_j)(\mathbf{m}_i - \mathbf{m}_j)^T\boldsymbol{w}$$

$$= \boldsymbol{w}^T S_B \boldsymbol{w}$$

where $S_B = \sum_{i\neq j}\ (\mathbf{m}_i - \mathbf{m}_j)(\mathbf{m}_i - \mathbf{m}_j)^T$ is the between-class scatter matrix.

Measure of Between-Class Separability

- However, simply maximizing the distance between mean vectors is not enough
  - It does not consider the variances of each individual classes
- FDA also will minimize the within class variance after data points are projected into a subspace



This plane gives better between-class separability

Within-class scatter

- Covariance matrix for class i in the original feature space

$$S_i = \frac{1}{N_i} \sum_{n \in C_i} (\mathbf{x}^n - \mathbf{m}_{\boldsymbol{i}})(\mathbf{x}^n - \mathbf{m}_{\boldsymbol{i}})^{\boldsymbol{T}}$$

- Using variance to measure the scatter within class i in the new subspace

$$\widetilde{S}_i = \frac{1}{N_i} \sum_{n \in C_i} \boldsymbol{w}^T (\mathbf{x}^n - \mathbf{m}_{\boldsymbol{i}})(\mathbf{x}^n - \mathbf{m}_{\boldsymbol{i}})^{\boldsymbol{T}} \boldsymbol{w} = \boldsymbol{w}^T S_i \, \boldsymbol{w}$$

- Within-class scatter:

$$\boldsymbol{w}^T S_w \, \boldsymbol{w}, \text{ where } S_w = \sum_{i=1}^{k} S_i$$

FDA Criterion

- Maximizing the between-class scatter, and minimizing the within-class scatter.

Between-class scatter

$$\max J(\mathbf{w}) = \frac{\boldsymbol{w}^T S_B \, \boldsymbol{w}}{\boldsymbol{w}^T S_W \, \boldsymbol{w}}$$

Within-class scatter

- Property, scaling $\boldsymbol{w}$ does not affect the objective function.

Deriving FDA

- Taking derivative of J(w), and equate it to zero

$$\frac{d}{d\boldsymbol{w}} J(\mathbf{w}) = \frac{d}{d\boldsymbol{w}}\left[\frac{\boldsymbol{w}^T S_B\ \boldsymbol{w}}{\boldsymbol{w}^T S_w\ \boldsymbol{w}}\right] = 0$$

$$2[\boldsymbol{w}^T S_w\ \boldsymbol{w}\ ]S_B w - 2[\boldsymbol{w}^T S_B\ \boldsymbol{w}\ ]S_W w\ =\ 0$$

$$S_B \boldsymbol{w} - \frac{\boldsymbol{w}^T S_B\ \boldsymbol{w}}{\boldsymbol{w}^T S_w\ \boldsymbol{w}} S_W \boldsymbol{w}\ =\ 0$$

$$S_B \boldsymbol{w} - J(\boldsymbol{w}) S_W \boldsymbol{w} =\ 0$$

FDA Algorithm

- Generalized Eigenvalue decomposition problem
$$S_B\boldsymbol{w} - J(\boldsymbol{w})S_W\boldsymbol{w} = 0$$
- $J(\boldsymbol{w})$ is the corresponding generalized eigenvalue.

- Finding the K generalized eigenvectors corresponding to the largest K eigenvalues.

How many projections do we need?

- For k classes, only k-1 projections of **w** are needed by solving the following generalized eigenvalue decomposition problem.

$$S_B\boldsymbol{w} - J(\boldsymbol{w})S_W\boldsymbol{w} = 0$$

- Because between-class scatter S_B is of a rank k-1 at most.

$$S_B = \sum_{i \neq j} (\mathbf{m}_i - \mathbf{m}_j)(\mathbf{m}_i - \mathbf{m}_j)^T = \sum_{i=1}^{k} N_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

where **m** is the mean of all data points. This shows S_B is a sum of k matrices of at most rank 1, and **m** is a linear combination of $\{\mathbf{m}_i\}$

- There are at most k-1 nonzero eigenvalues of $J(\boldsymbol{w})$. Thus only k-1 projects are needed, and the others are yielding trivial zero solution to $J(\boldsymbol{w})$.

Two classes: when k=2

- FDA Eigenvalue problem: $S_B\boldsymbol{w} - J(\boldsymbol{w})S_W\boldsymbol{w} = 0$

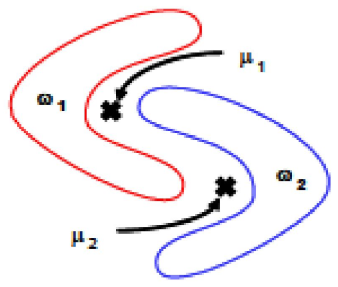
where $S_B = (\mathbf{m_1} - \mathbf{m_2})(\mathbf{m_1} - \mathbf{m_2})^T$

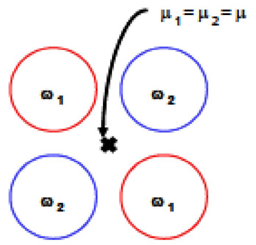
It can be proved that $\mathbf{w} = S_W^{-1}(\mathbf{m_1} - \mathbf{m_2})$

- It is related to the optimal Bayes classifier with Gaussian assumption on class-conditional distribution

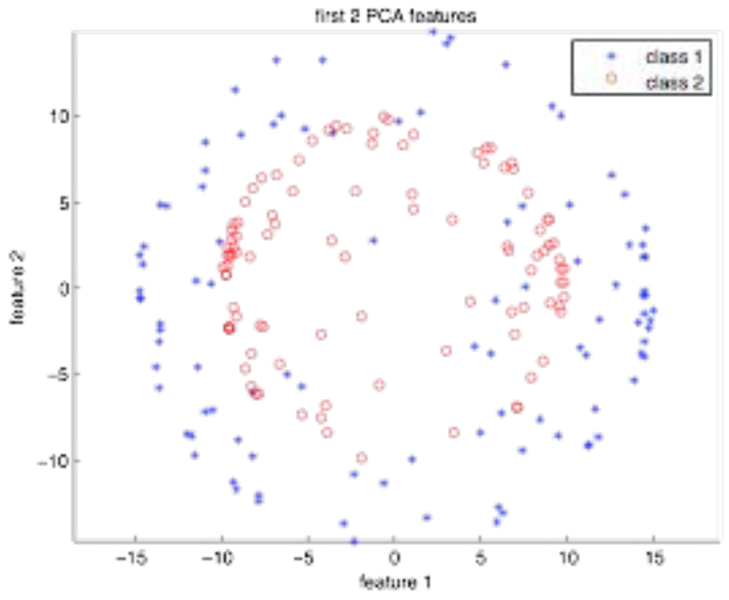
Limitations

- Only k-1 features can be extracted from FDA
  - Each from one of k-1 projections applied to original feature vector
  - This may underestimate the dimensionality necessary for a smaller error rate
    - Will 9 dimensional feature vector be able to give a good performance on MNIST?
- FDA is based on the assumption that all data from each class is generated according to Gaussian distribution.
  - Mean vector and covariance matrix are parameters for these Gaussians
  - What if the distributions are non Gaussian, or even nonlinearly separable?





Two Concentric Spheres: PCA



2

Linear projections perform poorly on this problem.

Non-Linear PCA: Kernel PCA

Use the SVM kernel trick to model non-linearities in dimensionality reduction

Objective to minimize reconstruction error in feature space:

$$\min_{U_k} \sum_{i=1}^{N} \left\| \Phi(\mathbf{x}_i) - U_k U_k^T \Phi(\mathbf{x}_i) \right\|^2$$

Solution found with SVD on covariance matrix:

$$\Phi(X) = U \Sigma V^T$$

Data must be normalized to be zero mean:

$$\tilde{K}(x,y) = (\Phi(x) - E_x[\Phi(x)]).(\Phi(y) - E_y[\Phi(y)])$$

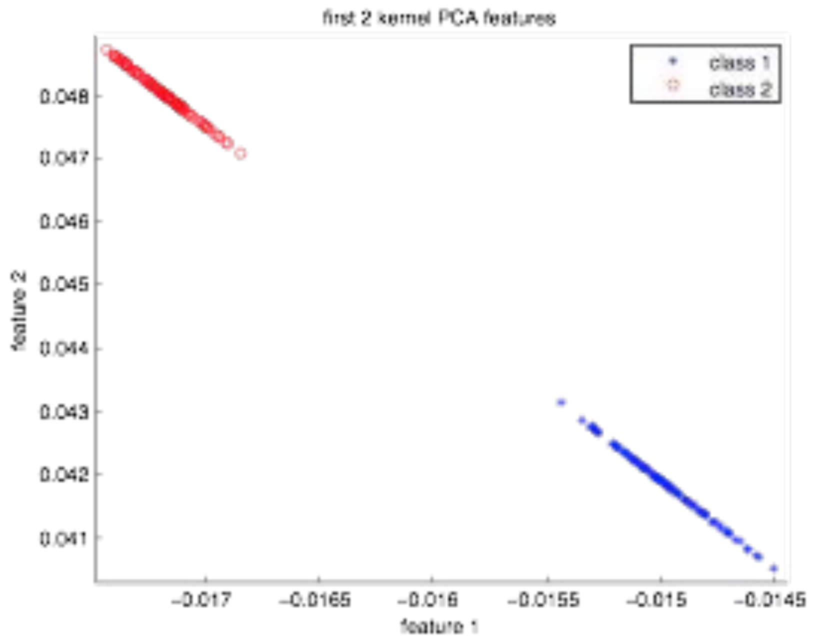
$$= K(x,y) - E_x[K(x,y)] - E_y[K(x,y)] + E_x[E_y[K(x,y)]]$$

Algorithm

- Choose a kernel and apply to all pairs of points to get matrix **K of** distances
- Compute eigenvalues and eigenvectors of **K**
- Retain the eigenvectors corresponding to the largest eigenvalues
- A new datapoint can be represented as the following set of features:

$$y_j = \sum_{i=1}^{m} a_{ji} K(\mathbf{x}, \mathbf{x}_i), j = 1, \ldots m$$

Kernel PCA (Gaussian kernel)



4

Summary

- Dimensionality reduction is a good preprocessing technique for many types of machine learning problems.
- Neural networks often implicitly perform some level of dimensionality reduction along the way but for the other methods we have discussed it has to be explicitly done.