Machine Learning (CSEN 240)

Lecture 14

Sid Nath

Agenda

- Administrative
  - Term project

Dimension Reduction

What is a “good” # of features beyond which accuracy does not increase (may decrease)?

Curse of dimensionality

- High-dimensional datasets tend to be sparse (e.g., finding a water body in a desert)
- #samples required to maintain statistical significance grows exponentially with #dimensions
    - Hard to tell “similar” / neighboring datapoints

What can be challenging at high dimensions?

- Curse of dimensionality
- Some dimensions may be correlated (or, redundant)
- Computation/storage cost
- Difficult to visualize (explain the model)

How to Reduce Dimensions?

Reducing dimensions can cause “loss of information”

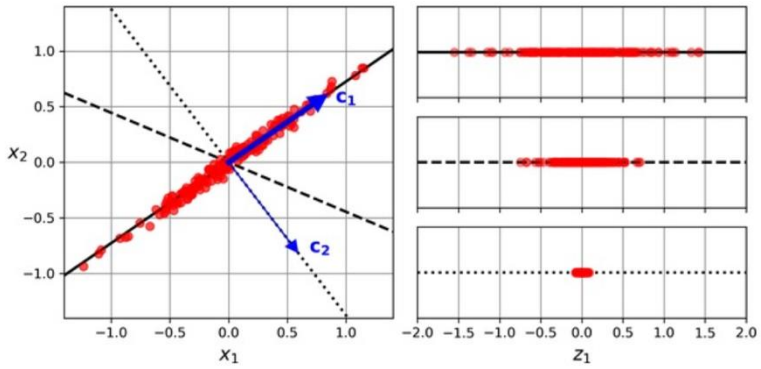
Objectives:

- Eliminate unnecessary features
- Preserve only the important characteristics of the original data

Project high-dimensional data to a lower dimension based on the significance of the dimensions

Principal component analysis (PCA) or Karhunen-Loeve Hotelling Transform (KLT) is one popular way to achieve this.

- Find a projection that maximizes variance



Maximizing Variance

Let u have unit norm, i.e., $\left|\left|u\right|\right|_2 = 1$

Let $X_{M\times N} = [x_1, x_2, \dots, x_N]^T$

Projection of $X_{M\times N}$ on u: $[x_1^T u, x_2^T u, \dots, x_N^T u]$

Variance (also referred to as energy assuming zero mean): $\sum_{i=1}^{N} |x_i^T u|^2$

Max variance is thus: u $= \ argmax_u \sum_{i=1}^{N} |x_i^T u|^2$

How can we compute u?

Rank of a Matrix

r$ank(A)$: Largest # linearly independent rows (or, columns) in matrix $A_{M\times N}$

$$rank(A) \leq \min(M, N)$$

A has full rank if $rank(A) = \min(M, N)$

What is the rank of A? (3 because the 4th column is a linear combination of the other 3 columns)

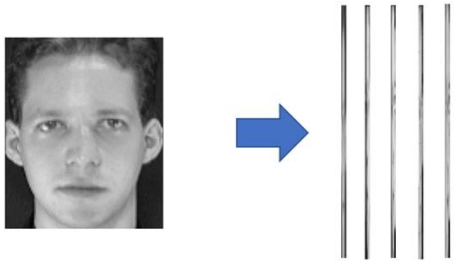
$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Rank Examples

Example #1: Five different face images (vector each image as a long vector) in a matrix $D \times 5, D \gg 5$. What is the rank?

Answer: 5

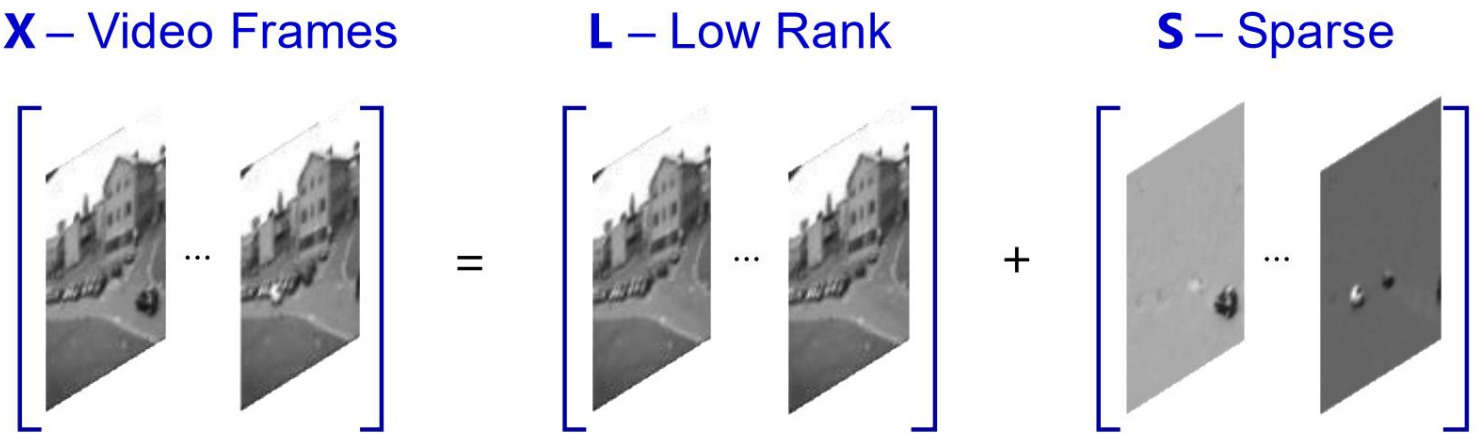
Example #2: Pick one of the above face images and repeat it five times to form a matrix. What is the rank?



Answer: 1

Rank Exercise

Suppose you have a sequence of surveillance video frames. The background is pure static and only the foreground has moving objects.



What is the rank of the background matrix L?

Eigenvalues and Eigenvectors

Suppose A is a square matrix $N \times N$. Which vectors get mapped to a scalar multiple of themselves?

That is, which vectors u satisfy the following: $Au = \lambda u$?

Vectors u are the eigenvectors of A, scalars λ are the associated eigenvalues.
$u's$ represent the direction of variation of the dataset and the $\lambda's$ equal to the number of variables or dimension of the dataset in that direction.

$A_{N\times N}$ has full rank $\rightarrow N$ eigenvectors $u_1, u_2, \ldots, u_N$ and N eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$

These eigenvectors are orthogonal to each other: $u_i^T u_j = 0, \forall i, j$

Each eigenvector also has unit norm: $u_i^T u_i = 1, \forall i$

Usually, we sort the eigenvalues in descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$

Eigenvalues and Eigenvectors

Eigenvectors: $U = [u_1, u_2, \dots, u_N]$

$$U^T U = UU^T = I_N \Rightarrow U^{-1} = U^T$$

$$\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_N)$$

$$\begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix}$$

Eigenvalue decomposition (EVD): $A = U\Lambda \mathrm{U}^{\mathrm{T}}$

Can you prove this?

$$A\ is\ N \times N; Au = \lambda u$$
$$AU = U\Lambda$$
$$AUU^T = AI = A = U\Lambda U^T$$

$$Au = \lambda u \Rightarrow (A\ - \lambda I)u = 0$$

Eigenvalue Decomposition Example

- $\mathbf{Au} = \lambda\mathbf{u} \Rightarrow \mathbf{Au} - \lambda\mathbf{u} = \mathbf{0} \Rightarrow (\mathbf{A} - \lambda\mathbf{I})\mathbf{u} = \mathbf{0}$

- Example: $\mathbf{A} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$, $\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} 5-\lambda & 4 \\ 4 & 5-\lambda \end{bmatrix}$

Determinant of $(A - \lambda I) = 0$

$$(5-\lambda)^2 - 16 = \lambda^2 - 10\lambda + 9 = (\lambda - 9)(\lambda - 1) = 0$$

So, $\lambda = 9 \; or \; \lambda = 1$

For $\lambda = 9, A - \lambda I = \begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix}$

$$(A - \lambda I)u = 0 \Rightarrow x_1 - x_2 = 0 \; if \; u = [x_1 \; x_2]^T$$

u is also a unit vector, so $u = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]^T$

EVD to Singular Value Decomposition (SVD)

We need to compute u $= \ argmax_u \sum_{i=1}^{N} |x_i^T u|^2$

$$XX^T = U\Lambda U^T \Rightarrow \Lambda = U^T XX^T U$$

$u = u_1$ is the top eigenvector with the largest λ_1 for matrix XX^T

If X is full rank, SVD: $X_{M\times N} = U_{M\times M}\Sigma_{M\times N}V_{N\times N}^T$

$U_{M\times M}$: left singulars; and $U^T U = UU^T = I_M$

$V_{N\times N}$: right singulars; and $V^T V = VV^T = I_N$

If $M > N$, $\mathbf{\Sigma}_{M\times N} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & \\ & & \ddots & \vdots \\ \vdots & & & \sigma_N \\ & & & \vdots \\ 0 & & \cdots & 0 \end{bmatrix}$

Singular values are also sorted in decreasing order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0$

EVD to Singular Value Decomposition (SVD)

We need to compute u $= \ argmax_u \sum_{i=1}^{N} |x_i^T u|^2$

$$XX^T = U\Lambda U^T \Rightarrow \Lambda = U^T XX^T U$$

$$X_{M\times N} = U_{M\times M}\Sigma_{M\times N}V_{N\times N}^T$$

$$\begin{aligned} XX^T &= U_{M\times M}\Sigma_{M\times N}\, V_{N\times N}^T V_{N\times N}\Sigma_{M\times N}^T U_{M\times M}^T \\ &= \ U_{M\times M}\Sigma_{M\times N}\,\Sigma_{M\times N}^T U_{M\times M}^T \\ &= U\ \Lambda\ U^T \end{aligned}$$

The left singulars of $X_{M\times N}$ are the same as the eigenvectors of XX^T

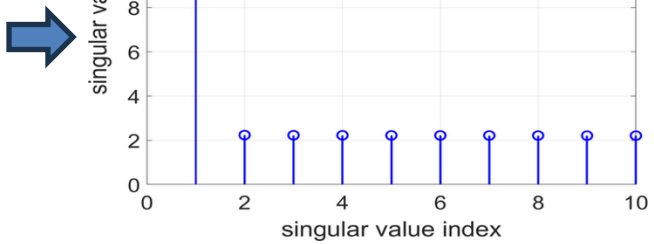
This suggests that the principal components of U can also be found using SVD

SVD Application

I have 10 copies of the same image (each with D >> 10 pixels). Each image is corrupted by noise. Is there a way for me to recover the image with min computations?

Original Image

Noise Corrupted Image



Pick u_1 and first image: x_1, project image onto u_1: $\alpha_1 = u_1^T x_1$
Recovered first image, $\widehat{x_1} = u_1 u_1^T x_1$

Recovered 1st Image with $\mathbf{u}_1$

Projection to D-Dimensions

Let's say the original dimensions of $X\ are\ M \times N.\ M \gg D$

Use D top left singular vectors of X for projection that correspond to the D largest singular values

If D = 1, pick $[u_1^T x]$

If D = 3, pick $[u_1^T x\ \ u_2^T x\ \ u_3^T x]^T$

How to choose D?
Compute projection residual: XX^T with top-D principal components (PC) vs. (N-D) PCs
Essentially, $\Lambda = diag(\lambda_1, \dots, \lambda_D, 0, \dots, 0)$ vs. $diag(0, 0, \dots, 0, \lambda_{D+1}, \dots, \lambda_N)$

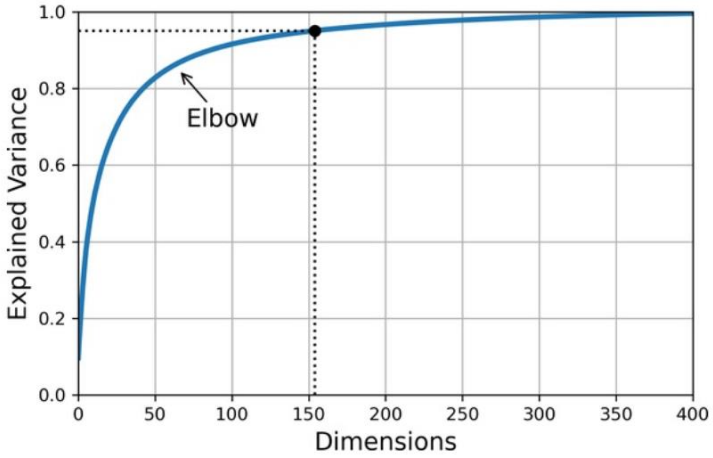
The differences decreases as we increase D till some point
Small D captures significant amount of the original information

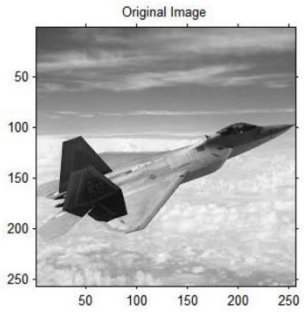
Explained Variance

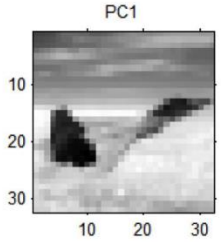
Computed as $\sum_{i=1}^{D} \lambda_i$

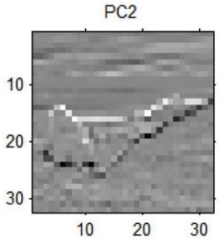
Choose D that such that explained variance is large, e.g., $\geq 95\%$

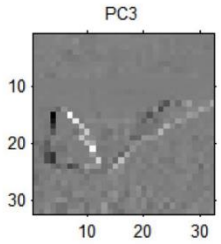
D is sensitive to the dataset
E.g., $M = 1000$, 95% variance can be explained from $D = 1, \ldots, 950$

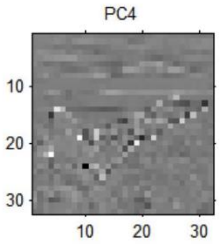


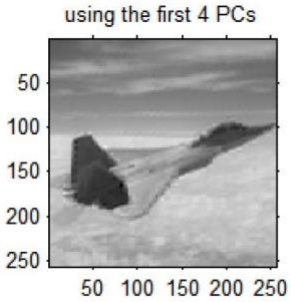


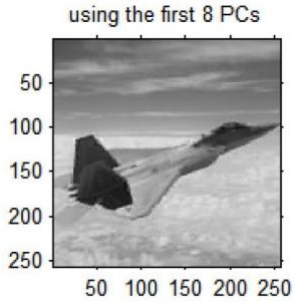


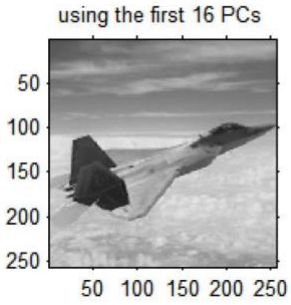


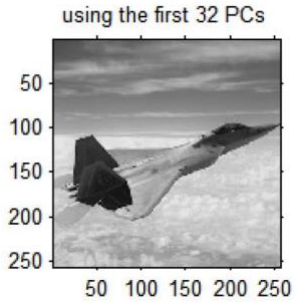












Application: Face Recognition

- Two major steps
  - Feature extraction

  - Classification

Training vs. inferencing

Feature extraction
$\tilde{\mathbf{x}} = f(\mathbf{x})$

$\mathbf{x} \in \mathbb{R}^D, \tilde{\mathbf{x}} \in \mathbb{R}^d, d \ll D$

Classification
$c_i = g(\tilde{\mathbf{x}}_i)$

$\{\mathbf{x}_i, c_i\}$ is the training dataset

Feature extraction
$\tilde{\mathbf{x}} = f(\mathbf{x})$

Classification
$\hat{c} = g(\tilde{\mathbf{x}})$

$\hat{c} = g(f(\mathbf{x}))$

Application: Face Recognition

C classes (subjects) each with N training images: x_{ci} (i^{th} image for subject c)

$$X = [x_{11}, \dots, x_{1N}, x_{21}, \dots, x_{2N}, \dots, x_{C1}, \dots, x_{CN}]^T$$

SVD for $X = U\,\Lambda\,V^T$

$f(x) = y_{d\times 1} = U^T_{:,[1:d]} x_{M\times 1}$ (using top-d left singular vectors. $U^T_{:,[1:d]} = [u_1, \dots, u_d]$)

Extracted features for the i^{th} image for subject c:

$$y_{\mathrm{ci}} = U^T_{:,[1:d]} x_{ci} = \left[u_1^T x_{ci}\; u_2^T x_{ci} \dots u_d^T x_{ci}\right]^T$$

For a test image, x_t, extract features $y_t = U^T_{:,[1:d]} x_t = \left[u_1^T x_t\; u_2^T x_t \dots u_d^T x_t\right]^T$

Determine the class label by: $\hat{c} = \underset{\substack{\{c=1,\dots,C\}\\ \{i=1,\dots,N\}}}{argmin} \left|\left|\mathrm{y} - \mathrm{y_{ci}}\right|\right|_2$

PCA Summary

Unsupervised learning algorithm

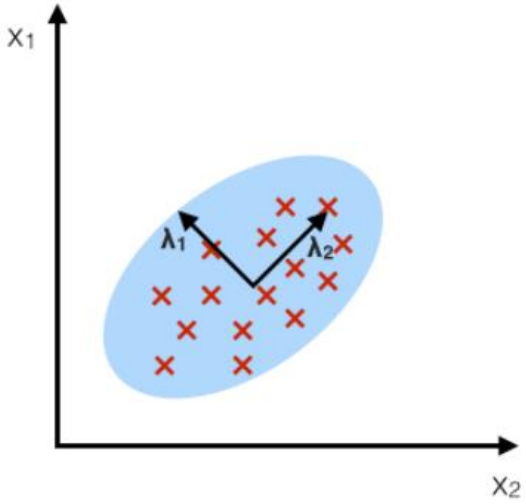
Can be done via EVD on XX^T (X is the training dataset with zero mean)

Can be done via SVD, factorizes matrix into three matrices, and the left singular vectors of X are the same as the eigenvectors of XX^T

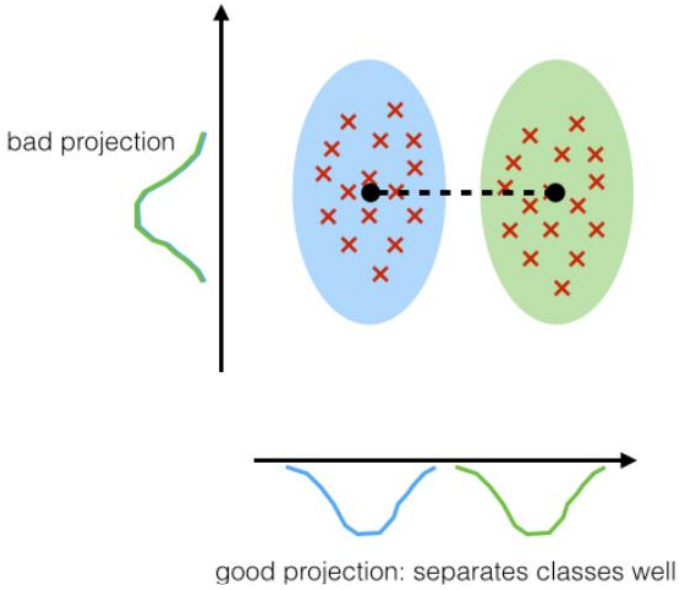
PCA is used in data compression, feature extraction

Linear Discriminant Analysis (LDA)

Supervised algorithm to select the component axis that separates classes (vs. PCA that selects the component axis that maximizes data variance)



PCA



LDA

Projection Direction

Assume two classes:
C_1 with N_1 data samples and C_2 with N_2data samples

Class means:
$m_1 = \frac{1}{N_1}\sum_{i\in C_1} x_i \qquad m_2 = \frac{1}{N_2}\sum_{j\in C_2} x_j$

Means projected onto w: $m_1 = w^T m_1; m_2 = w^T m_2$

Measure of class separation: $m_1 - m_2 = w^T(m_1 - m_2)$

To select a component axis that separate classes, we want to maximize $(m_1 - m_2)$

$$w = argmax_w w^T(m_1 - m_2)$$

Subject to, $w^T w = 1$ (to avoid arbitrarily large w)

Solution: $w = \frac{m_1 - m_2}{||m_1 - m_2||_2}$ [please try to derive this using Lagrange multipliers]

Projection Direction

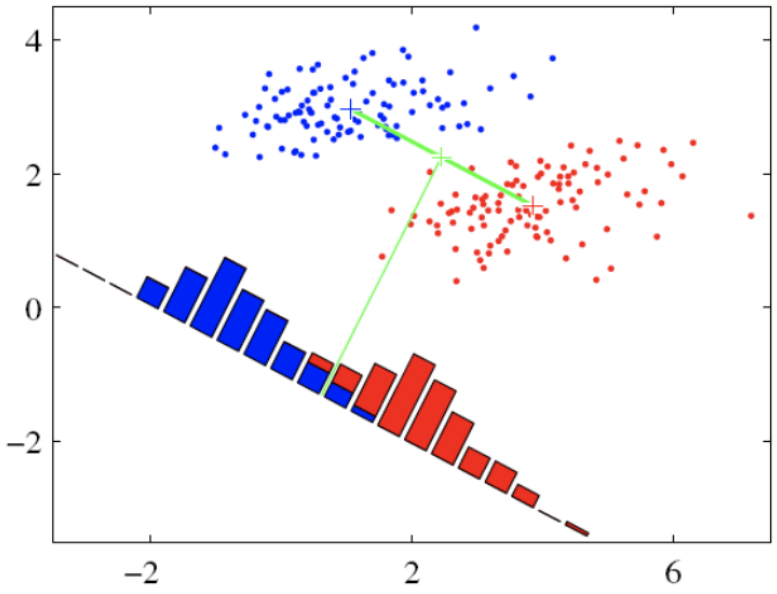
$$w = argmax_w w^T(m_1 - m_2)$$

Subject to, $w^T w = 1$ (to avoid arbitrarily large w)

Solution: $w = \frac{m_1 - m_2}{||m_1 - m_2||_2}$

In the projection space

- Centroids/ means are well-separated
- Some datapoints overlap



LDA

Also, called Fisher's Linear Discriminant

Maximize a function that will give a large separation between the projected class means, while giving a small variance within each class, thereby minimizing the class overlap

Within-class variance in the projection space:

$s_1 = \sum_{i \in C_1} (y_i - m_1)^2$ and $s_2 = \sum_{j \in C_2} (y_j - m_2)^2$

Where, $y_i = w^T x_i$

Total within-class variance: $s_1^2 + s_2^2$

Fisher's criterion: maximize the ratio of between-class variance to within-class variance

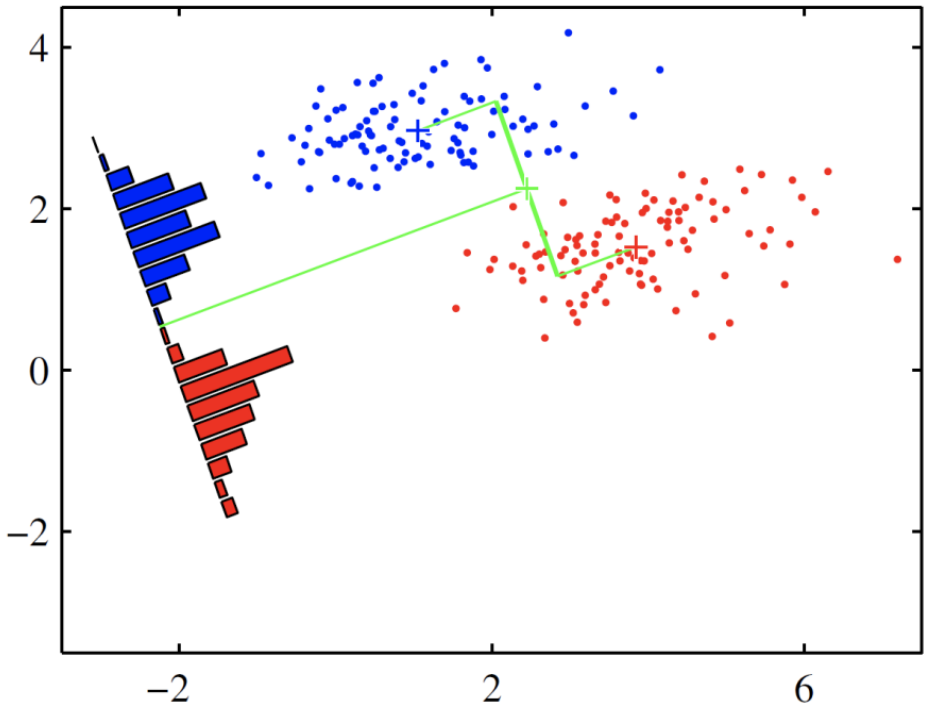
$$\max \mathcal{J}(w) = \max \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

LDA

$s_1 = \sum_{i \in C_1} (y_i - m_1)^2$ and $s_2 = \sum_{j \in C_2} (y_j - m_2)^2$

Fisher's criterion: maximize the ratio of between-class variance to within-class variance

$$\max \mathcal{J}(w) = \max \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$



LDA

$$\max \mathcal{J}(w) = \max \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

How to find w?

$$w^T(m_1 - m_2)[w^T(m_1 - m_2)]^T = w^T(m_1 - m_2)(m_1 - m_2)^T w$$

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

$$s_1^2 + s_2^2 = \sum_{i \in C_1} (w^T x_i - m_1)(w^T x_i - m_1)^T + \sum_{j \in C_2} \left(w^T x_j - m_2\right)\left(w^T x_j - m_2\right)^T$$

$$S_w = \sum_{i \in C_1} (x_i - m_1)(x_i - m_1)^T + \sum_{j \in C_2} \left(x_j - m_2\right)\left(x_j - m_2\right)^T$$

$$J(w) = \frac{w^T S_B w}{w^T S_w w}$$

LDA

$$J(w) = \frac{w^T S_B w}{w^T S_w w}$$

$$\frac{d\mathcal{J}(w)}{dw} = (w^T S_B w) S_w w = (w^T S_W w) S_B w = 0$$

Solution: $w\ is$ the eigenvector of $S_w^{-1} S_B$ (S_w must be full rank)

If we want top-d projection vectors, $w_1, w_2, \ldots, w_d$ then these are the top-d eigenvectors of $S_w^{-1} S_B$

If S_w is not full rank:

- Reduce the dimension of the data using PCA
- Apply LDA to the reduced-dimensional data

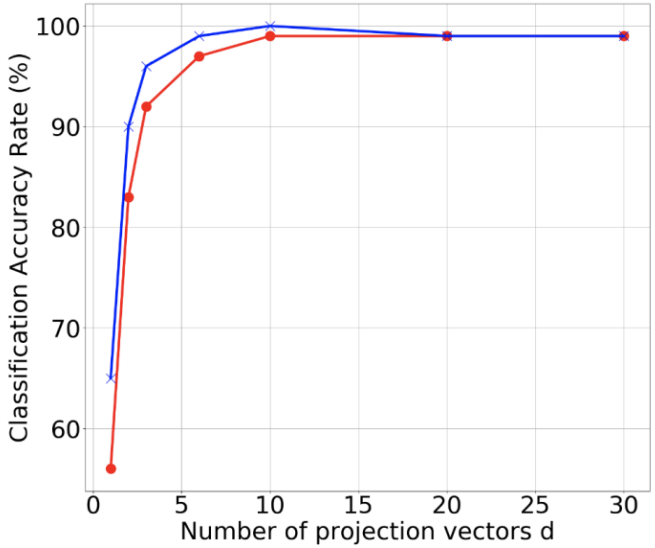
Example

Suppose you want to recognize faces of 10 people (subjects) and you have 9 samples per subject. Thus, the total #training samples is 90.

Image size: 112x92 (D = 10304 >> 90)

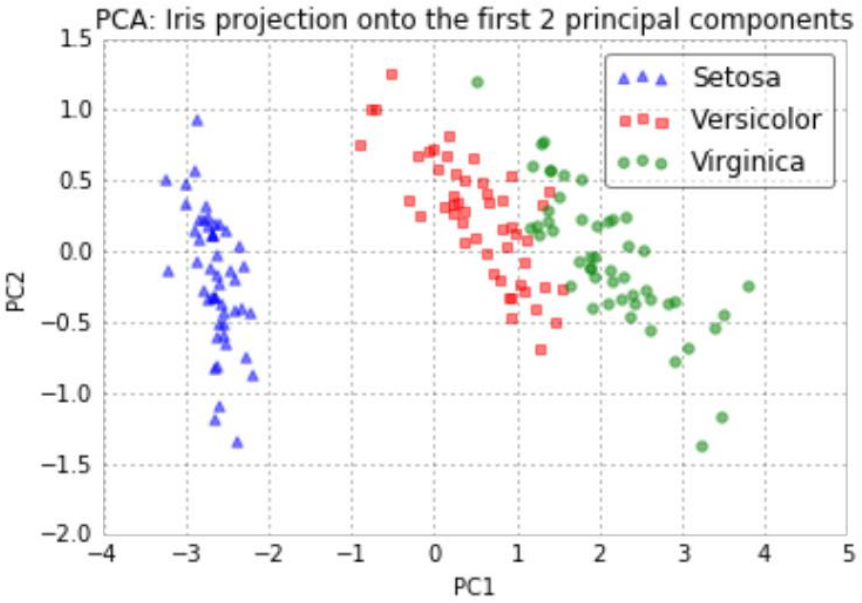
Reduce to $d_0 = 40\ using\ PCA$. LDA performs better for small values of d.

- Blue: LDA
- Red: PCA

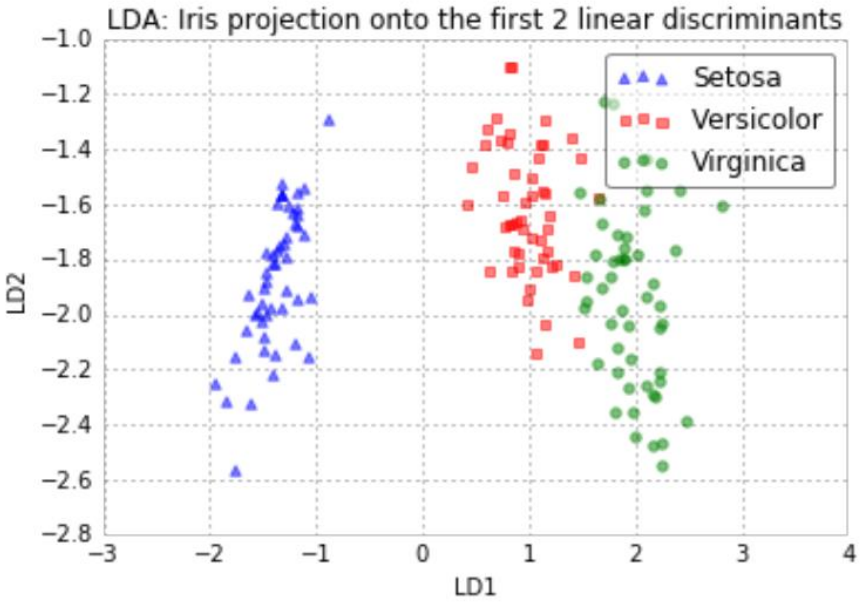


LDA vs. PCA

- PCA



- LDA



LDA vs. PCA

PCA

Find features in the direction of the greatest variance (w/o class labels)

Unsupervised

Reduce dimensions with some loss

Visualize high-dimensional data

LDA

Find features that maximize class separation

Supervised

Reduce dimensions while preserving class information