THE UNIVERSITY OF TEXAS AT AUSTIN
Department of Statistics and Data Sciences
College of Natural Sciences

SDS 385: Stat Models for Big Data

Lecture 12: PCA and LDA

Purnamrita Sarkar
Department of Statistics and Data Science
The University of Texas at Austin

https://psarkar.github.io/teaching

Principal Component Analysis

- Goal: Find the direction of the most variance.
- Say X is the data matrix
- The average is $\bar{\boldsymbol{x}} = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i}{n}$
- Let $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}$

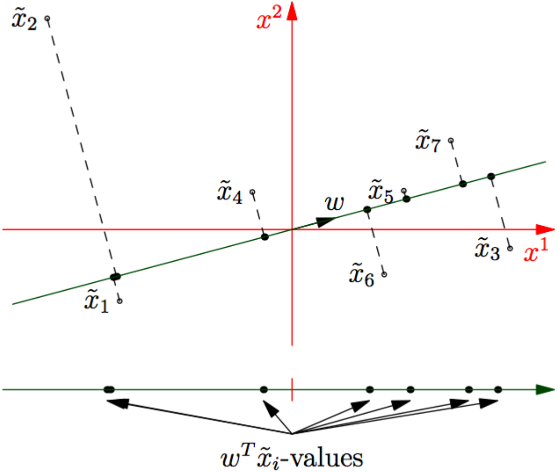
Principal Component Analysis

- Goal: Find the direction of the most variance.
- Say X is the data matrix
- The average is $\bar{\boldsymbol{x}} = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i}{n}$
- Let $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}$
- The sample variance of $(\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n)$ *along a direction* w is give by:

$$\frac{1}{n} \sum_{i=1}^{n} (\tilde{\boldsymbol{x}}_i^T w)^2$$

- What is the sample variance of $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ *along a direction* $\boldsymbol{w}$?

Principal Component Analysis



First component

- So the first PC direction is:

$$\boldsymbol{w}_1 = \arg\max_{\|\boldsymbol{w}\|=1} \frac{1}{n}\sum_{i=1}^{n}(\tilde{\boldsymbol{x}}_i^T \boldsymbol{w})^2$$

- And the first PC component of $\tilde{\boldsymbol{x}}_i$ is $\tilde{\boldsymbol{x}}_i^T \boldsymbol{w}_1$

First component

- So the first PC direction is:

$$\boldsymbol{w}_k = \arg \max_{\substack{\|\boldsymbol{w}\|=1 \\ \boldsymbol{w} \perp \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k-1}}} \frac{1}{n} \sum_{i=1}^{n} (\tilde{\boldsymbol{x}}_i^T \boldsymbol{w})^2$$

- And the k^{th} PC component of $\tilde{\boldsymbol{x}}_i$ is $\tilde{\boldsymbol{x}}_i^T \boldsymbol{w}_k$
- Note that $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ form an orthogonal basis.

Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$

Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg\max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

- This is the first eigenvector of $S = \tilde{X}^T \tilde{X}$

Eigenvector and eigenvalues

- Any square symmetrix matrix S has real eigenvalues
- The i^{th} eigenvalue,vector pair satisfy $S\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i$
- The eigenvectors are orthogonal to each other, and normalized to have length 1.

Eigenvector and eigenvalues

- Any square symmetrix matrix S has real eigenvalues
- The i^{th} eigenvalue,vector pair satisfy $S\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i$
- The eigenvectors are orthogonal to each other, and normalized to have length 1.
- In matrix terms, we can write:

$$S = U\Sigma U^T, \text{ where}$$

  - columns of U are the orgonal eigenvectors, and
  - Σ is a diagonal matrix with eigenvalues on the diagonal
- The larger the magnitude of the eigenvalue, more important the eigenvector

Back to PCA: Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$

Back to PCA: Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

Back to PCA: Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg\max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

- This is the first eigenvector of $S = \tilde{X}^T \tilde{X}$

Back to PCA: Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg\max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

- This is the first eigenvector of $S = \tilde{X}^T \tilde{X}$
- What is S?

Back to PCA: Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg\max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

- This is the first eigenvector of $S = \tilde{X}^T \tilde{X}$
- What is S?
- Its the scalar multiple of the sample covariance matrix

$$\hat{\Sigma} = \frac{1}{n} \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^T = \frac{S}{n}$$

Back to PCA: Simple algorithm

- Let W is a matrix with $\boldsymbol{w}_k$ along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of $\tilde{X}$
- We can frame the optimization problem also as

$$\boldsymbol{w}_1 = \arg \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^T \tilde{X}^T \tilde{X} \boldsymbol{w}$$

- This is the first eigenvector of $S = \tilde{X}^T \tilde{X}$
- What is S?
- Its the scalar multiple of the sample covariance matrix

$$\hat{\Sigma} = \frac{1}{n} \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^T = \frac{S}{n}$$

- So, all you have to do is to calculate eigenvectors of the covariance matrix.

Back to PCA: Simple algorithm

- So, all you have to do is to calculate eigenvectors of the covariance matrix.
- But, do I even need to do that?

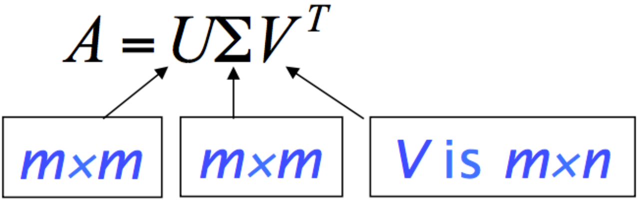
Back to PCA: Simple algorithm

- So, all you have to do is to calculate eigenvectors of the covariance matrix.
- But, do I even need to do that?
- The right singular vectors of $\tilde{X}$ is just fine.

Back to PCA: Simple algorithm

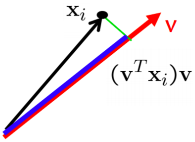
- So, all you have to do is to calculate eigenvectors of the covariance matrix.
- But, do I even need to do that?
- The right singular vectors of $\tilde{X}$ is just fine.
- How many PC's? (more of a dissertaiton question)

Singular value decomposition



- The columns of U are orthogonal eigenvectos of AA^T
- The columns of V are orthogonal eigenvectos of A^TA
- A^TA and AA^T have the same eigenvalues

Second interpretation



- Minimum reconstruction error:

$$(\boldsymbol{x}_i - (\boldsymbol{x}_i^T\boldsymbol{w})\boldsymbol{w})^T(\boldsymbol{x}_i - (\boldsymbol{x}_i^T\boldsymbol{w})\boldsymbol{w}) = \boldsymbol{x}_i^T\boldsymbol{x}_i - (\boldsymbol{x}_i^T\boldsymbol{w})^2$$

- So, the first PC direction gives the direction projecting on which has the **minimum reconstruction error**.

Low rank approximation

- Take the centered data matrix $\tilde{X}$ with SVD

$$\tilde{X} = USV^T$$

- Project on the top k PC's $W \in \mathbb{R}^{p \times k}$

Low rank approximation

- Take the centered data matrix $\tilde{X}$ with SVD

$$\tilde{X} = USV^T$$

- Project on the top k PC's $W \in \mathbb{R}^{p \times k}$
- You get $\tilde{X}W = US_kV^T$, where S_k has zeroed out all singular values $< \sigma_k$

Low rank approximation

- Take the centered data matrix $\tilde{X}$ with SVD

$$\tilde{X} = USV^T$$

- Project on the top k PC's $W \in \mathbb{R}^{p \times k}$
- You get $\tilde{X}W = US_kV^T$, where S_k has zeroed out all singular values $< \sigma_k$
- So $W = \arg \min_{\text{rank}(B)=k, B \in \mathbb{R}^{n \times p}} \|\tilde{X} - B\|_F^2$ and the reconstruction error is $\sum_{i=k+1}^{p} \sigma_i^2$

Low rank approximation

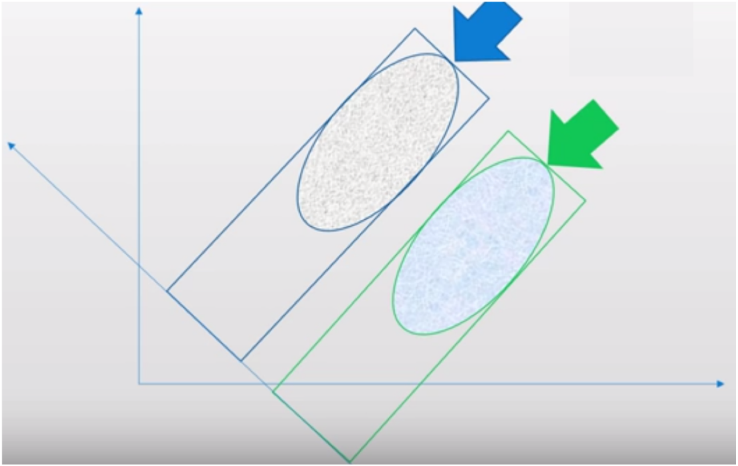
- Take the centered data matrix $\tilde{X}$ with SVD

$$\tilde{X} = USV^T$$

- Project on the top k PC's $W \in \mathbb{R}^{p \times k}$
- You get $\tilde{X}W = US_kV^T$, where S_k has zeroed out all singular values $< \sigma_k$
- So $W = \arg \min_{\text{rank}(B)=k, B \in \mathbb{R}^{n \times p}} \|\tilde{X} - B\|_F^2$ and the reconstruction error is $\sum_{i=k+1}^{p} \sigma_i^2$
- This explains why you want to take large k to reduce approx. error.

Linear Discriminant Analysis

- PCA did not have class information
- LDA does take that into account.
- We will do it for two classes.



LDA

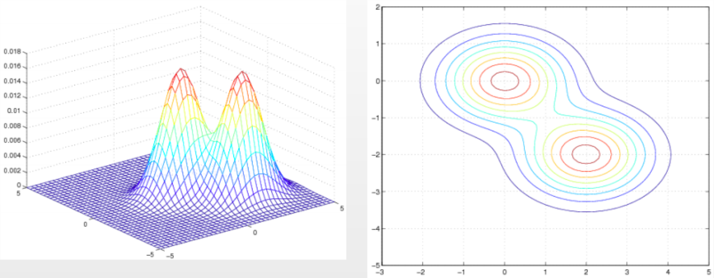
- Assume that the data is coming from a mixture of two Gaussians with parameters $(\mu_k, \Sigma_k), k \in \{1, 2\}$

LDA

- Assume that the data is coming from a mixture of two Gaussians with parameters $(\mu_k, \Sigma_k), k \in \{1, 2\}$
- Recall the density of a multivariate gausssian

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right)$$

A pretty picture



LDA

- Assign point x to the class with maximizes the posterior probability of belonging to that class

LDA

- Assign point x to the class with maximizes the posterior probability of belonging to that class

$$\begin{aligned}\arg\max_k P(y=k|\boldsymbol{x},\Theta) &= \arg\max_k \frac{P(\boldsymbol{x}|y=k,\Theta)P(y=k)}{P(\boldsymbol{x})}\\ &= \arg\max_k P(\boldsymbol{x}|y=k,\Theta)P(y=k)\end{aligned}$$

LDA

- Assign point x to the class with maximizes the posterior probability of belonging to that class

$$\begin{aligned}\arg\max_k P(y=k|\boldsymbol{x},\Theta) &= \arg\max_k \frac{P(\boldsymbol{x}|y=k,\Theta)P(y=k)}{P(\boldsymbol{x})} \\ &= \arg\max_k P(\boldsymbol{x}|y=k,\Theta)P(y=k)\end{aligned}$$

- So decision rule for class 1 is

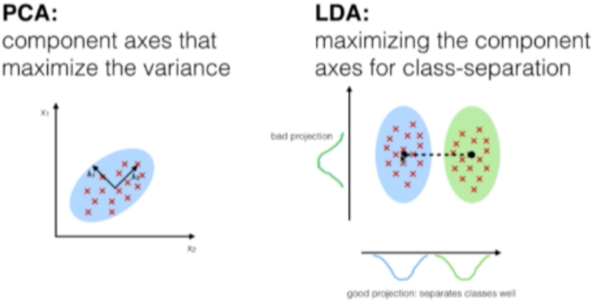
$$\begin{aligned}&-\frac{1}{2}\log|\Sigma_1|^{1/2} - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \log\pi_1 \\ &> -\frac{1}{2}\log|\Sigma_2|^{1/2} - \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) + \log\pi_2\end{aligned}$$

- For $\Sigma_1 \neq \Sigma_2$, this is a quadratic function.

LDA

- LDA assumes that $\Sigma_1 = \Sigma_2$
- So now we get a linear decision boundary

$$x^T \Sigma^{-1}(\mu_1 - \mu_2) > \frac{\mu_1 + \mu_2}{2} \Sigma^{-1}(\mu_1 - \mu_2) - \log \frac{\pi_1}{\pi_2}$$



Estimation

- Class proportion

$$\hat{\pi}_k = \frac{\sum_{y_i=k} y_i}{n}.$$

Estimation

- Class proportion

$$\hat{\pi}_k = \frac{\sum_{y_i=k} y_i}{n}.$$

- Class mean

$$\hat{\mu}_k = \frac{\sum_{y_i=k} x_i}{n}.$$

Estimation

- Class proportion

$$\hat{\pi}_k = \frac{\sum_{y_i=k} y_i}{n}.$$

- Class mean

$$\hat{\mu}_k = \frac{\sum_{y_i=k} x_i}{n}.$$

- Common class covariance matrix

$$\hat{\Sigma} = \frac{\sum_{k=1}^{K} \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{n - K}.$$

Multiple classes

- For datapoint x whose class you want to predict, for each class $k \in \{1, \ldots, K\}$, compute the **linear discriminant function** $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$ *with estimated parameters*

Multiple classes

- For datapoint x whose class you want to predict, for each class $k \in \{1, \ldots, K\}$, compute the **linear discriminant function** $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$ *with estimated parameters*
- Assign x to class that maximixes this function.

Multiple classes

- For datapoint x whose class you want to predict, for each class $k \in \{1, \dots, K\}$, compute the **linear discriminant function** $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$ *with estimated parameters*
- Assign x to class that maximixes this function.
- Why not use QDA?

Multiple classes

- Multiclass LDA minimizes $\dfrac{(x - \hat{\mu}_j)^T \hat{\Sigma}^{-1} (x - \hat{\mu}_j)}{2} - \log \hat{\pi}_j$

Multiple classes

- Multiclass LDA minimizes $\frac{(x-\hat{\mu}_j)^T\hat{\Sigma}^{-1}(x-\hat{\mu}_j)}{2} - \log\hat{\pi}_j$
- Calculate the square root of $\hat{\Sigma}^{-1/2}$
  - Calculate the eigen-decomposition of $\hat{\Sigma} = UDU^T$,
  - $\hat{\Sigma}^{-1/2} = UD^{-1/2}$

Multiple classes

- Multiclass LDA minimizes $\frac{(x-\hat{\mu}_j)^T\hat{\Sigma}^{-1}(x-\hat{\mu}_j)}{2} - \log\hat{\pi}_j$
- Calculate the square root of $\hat{\Sigma}^{-1/2}$
  - Calculate the eigen-decomposition of $\hat{\Sigma} = UDU^T$,
  - $\hat{\Sigma}^{-1/2} = UD^{-1/2}$
- Transform datapoint x to $\tilde{x} := D^{-1/2}U^Tx$
- Mean becomes: $\tilde{\mu}_j := D^{-1/2}U^T\hat{\mu}_j$

Multiple classes

- Multiclass LDA minimizes $\dfrac{(x-\hat{\mu}_j)^T\hat{\Sigma}^{-1}(x-\hat{\mu}_j)}{2} - \log\hat{\pi}_j$
- Calculate the square root of $\hat{\Sigma}^{-1/2}$
  - Calculate the eigen-decomposition of $\hat{\Sigma} = UDU^T$,
  - $\hat{\Sigma}^{-1/2} = UD^{-1/2}$
- Transform datapoint x to $\tilde{x} := D^{-1/2}U^T x$
- Mean becomes: $\tilde{\mu}_j := D^{-1/2}U^T\hat{\mu}_j$
- Remember Mahanobis distance?

$$(x-\hat{\mu}_j)^T\hat{\Sigma}^{-1}(x-\hat{\mu}_j) = (\tilde{x}-\tilde{\mu}_j)^T(\tilde{x}-\tilde{\mu}_j)$$

Multiple classes

- Multiclass LDA minimizes $\dfrac{(x-\hat{\mu}_j)^T\hat{\Sigma}^{-1}(x-\hat{\mu}_j)}{2} - \log \hat{\pi}_j$
- Calculate the square root of $\hat{\Sigma}^{-1/2}$
  - Calculate the eigen-decomposition of $\hat{\Sigma} = UDU^T$,
  - $\hat{\Sigma}^{-1/2} = UD^{-1/2}$
- Transform datapoint x to $\tilde{x} := D^{-1/2}U^T x$
- Mean becomes: $\tilde{\mu}_j := D^{-1/2}U^T \hat{\mu}_j$
- Remember Mahanobis distance?

$$(x-\hat{\mu}_j)^T\hat{\Sigma}^{-1}(x-\hat{\mu}_j) = (\tilde{x}-\tilde{\mu}_j)^T(\tilde{x}-\tilde{\mu}_j)$$

- After this "whitening", the decision rule becomes very simple:
  - Assign x to class j such that $(\tilde{x}-\tilde{\mu}_j)^T(\tilde{x}-\tilde{\mu}_j) - \log \hat{\pi}_j$

LDA algorithm

- Estimate parameters by $\hat{\pi}_j$, $\hat{\mu}_j$, $\hat{\Sigma}$
- Compute eigendecomposition of $\hat{\Sigma} = UDU^T$
- Transform the means to $\tilde{\mu}_j$
- For a datapoint x, compute the whitened point $\tilde{x}$
- Now assign to class j that minimizes $\frac{1}{2}\text{dist}(\tilde{x}, \tilde{\mu}_j)^2 - \log \hat{\pi}_j$

LDA and dimensionality reduction

- How many dimensions do we need to represent 2 points?

LDA and dimensionality reduction

- How many dimensions do we need to represent 2 points?
    - Just 1

LDA and dimensionality reduction

- How many dimensions do we need to represent 2 points?
  - Just 1
- How many dimensions do we need to represent K means?

LDA and dimensionality reduction

- How many dimensions do we need to represent 2 points?
  - Just 1
- How many dimensions do we need to represent K means?
  - Just $K - 1$

LDA and dimensionality reduction

- Whiten the data
- Create the matrix M of means $[\tilde{\mu}_1 \dots \tilde{\mu}_K]$

LDA and dimensionality reduction

- Whiten the data
- Create the matrix M of means $[\tilde{\mu}_1 \ldots \tilde{\mu}_K]$
- Do a PCA of this matrix, and call the top K singulae vectors $A \in \mathbb{R}^{p \times K}$ and all the rest as $A_\perp$.

LDA and dimensionality reduction

- Whiten the data
- Create the matrix M of means $[\tilde{\mu}_1 \ldots \tilde{\mu}_K]$
- Do a PCA of this matrix, and call the top K singulae vectors $A \in \mathbb{R}^{p \times K}$ and all the rest as $A_\perp$.
- For $\tilde{x}$, compute Ax

$$
\begin{aligned}
\|\tilde{x} - \tilde{\mu}_j\|^2 &= \|AA^T\tilde{x} + A_\perp A_\perp^T \tilde{x} - \tilde{\mu}_j\|^2 \\
&= \|AA^T\tilde{x} - \tilde{\mu}_j\|^2 + \underbrace{\|A_\perp A_\perp^T \tilde{x}\|^2}_{\text{Does not depend on } j}
\end{aligned}
$$

LDA and dimensionality reduction

- Whiten the data
- Create the matrix M of means $[\tilde{\mu}_1 \dots \tilde{\mu}_K]$
- Do a PCA of this matrix, and call the top K singulae vectors $A \in \mathbb{R}^{p \times K}$ and all the rest as $A_\perp$.
- For $\tilde{x}$, compute Ax

$$\begin{aligned} \|\tilde{x} - \tilde{\mu}_j\|^2 &= \|AA^T\tilde{x} + A_\perp A_\perp^T\tilde{x} - \tilde{\mu}_j\|^2 \\ &= \|AA^T\tilde{x} - \tilde{\mu}_j\|^2 + \underbrace{\|A_\perp A_\perp^T\tilde{x}\|^2}_{\text{Does not depend on } j} \end{aligned}$$

- So the LDA decision rule will be unchanged if we project into the subspace spanned by the centers.

Acknowledgment

- Some pictures are borrowed from Brett Bernstein's notes from NYU and Jia Li's notes from PSU
- Some slides are borrowed from Ryan Tibshirani's notes
- Elements of statistical learning, HTF