

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 $\bar{\mathbf{x}} = \frac{\sum_{i=1}^n \mathbf{x}_i}{n}$
- Let $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$

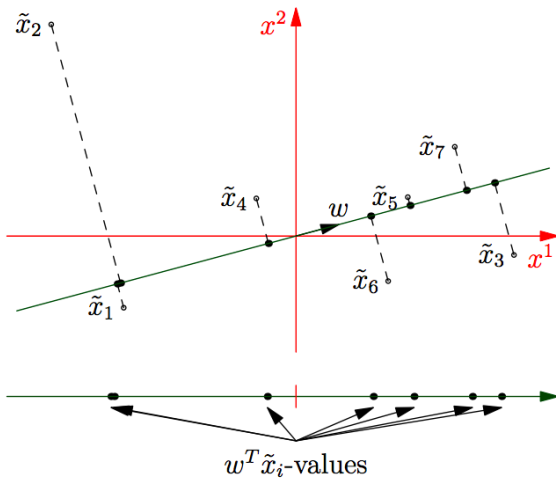
Principal Component Analysis

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

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

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

Principal Component Analysis



First component

- So the first PC direction is:

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

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

First component

- So the first PC direction is:

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

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

Simple algorithm

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

Simple algorithm

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

$$w_1 = \arg \max_{\|w\|=1} w^T \tilde{X}^T \tilde{X} w$$

Simple algorithm

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

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

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

Eigenvector and eigenvalues

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

Eigenvector and eigenvalues

- Any square symmetric matrix S has real eigenvalues
- The i^{th} eigenvalue, vector pair satisfy $S\mathbf{w}_i = \lambda_i\mathbf{w}_i$
- 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 orthonormal 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 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 \mathbf{w}_k along its columns
- $\tilde{X}W$ gives a **low dimensional** representation of \tilde{X}
- We can frame the optimization problem also as

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

Back to PCA: Simple algorithm

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

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

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

Back to PCA: Simple algorithm

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

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

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

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

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

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

Back to PCA: Simple algorithm

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

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

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

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

- 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

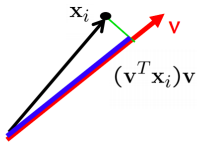
$$A = U \Sigma V^T$$

Diagram illustrating the dimensions of the matrices in the SVD equation $A = U \Sigma V^T$:

- U is $m \times m$
- Σ is $m \times m$
- V is $m \times n$

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

Second interpretation



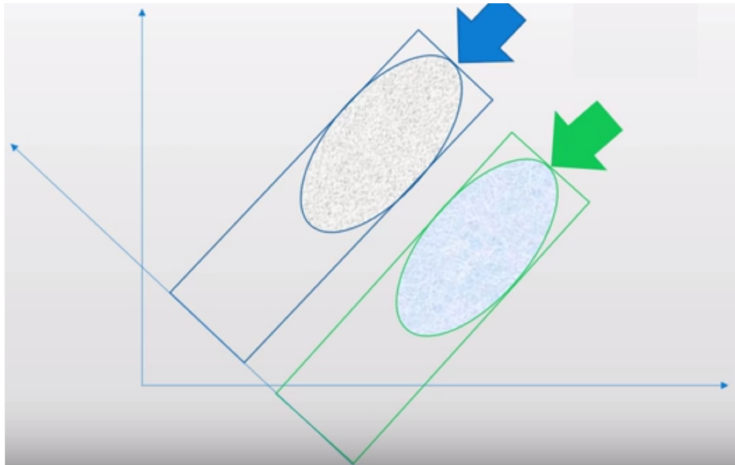
- Minimum reconstruction error:

$$(x_i - (x_i^T w)w)^T (x_i - (x_i^T w)w) = x_i^T x_i - (x_i^T w)^2$$

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

Linear Discriminant Analysis

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

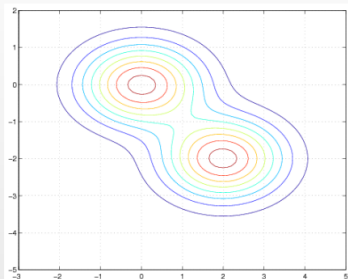
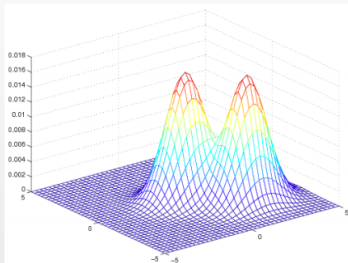


- Assume that the data is coming from a mixture of two Gaussians with parameters $(\mu_k, \Sigma_k), k \in \{1, 2\}$

- 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 gaussian

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

A pretty picture



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

- Assign point \mathbf{x} to the class with maximizes the posterior probability of belonging to that class

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

- Assign point \mathbf{x} to the class with maximizes the posterior probability of belonging to that class

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

- So decision rule for class 1 is

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

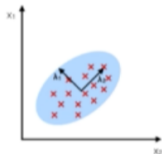
- For $\Sigma_1 \neq \Sigma_2$, this is a quadratic function.

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

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

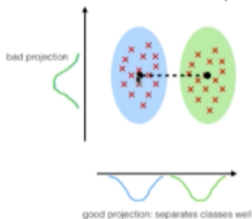
PCA:

component axes that maximize the variance



LDA:

maximizing the component axes for class-separation



- Class proportion

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

- Class proportion

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

- Class mean

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

- 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

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

- 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*

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 maximizes 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 maximizes this function.
- Why not use QDA?

Multiple classes

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

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 $\tilde{x} := D^{-1/2}U^T x$
- Mean becomes: $\tilde{\mu}_j := D^{-1/2}U^T \hat{\mu}_j$

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 $\hat{\Sigma} = UDU^T$,
 - $\hat{\Sigma}^{-1/2} = UD^{-1/2}$
- 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 $\frac{(x - \hat{\mu}_j)^T \hat{\Sigma}^{-1} (x - \hat{\mu}_j)}{2} - \log \hat{\pi}_j$
- 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 $\tilde{x} := D^{-1/2}U^T x$
- 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 $\frac{1}{2}\text{dist}(\tilde{x}, \tilde{\mu}_j)^2 - \log \hat{\pi}_j$

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

- 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 $[\tilde{\mu}_1 \dots \tilde{\mu}_K]$
- Do a PCA of this matrix, and call the top K singular 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 $[\tilde{\mu}_1 \dots \tilde{\mu}_K]$
- Do a PCA of this matrix, and call the top K singular 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 singular 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