ML&MEA (2024)
Lecture 11 - PCA and SVD

Quanying Liu

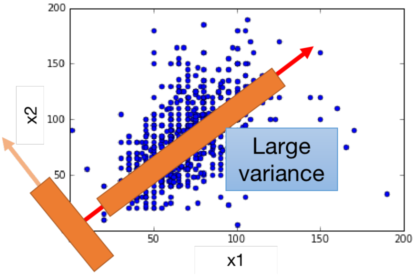
BME, SUSTech

2024.4.9

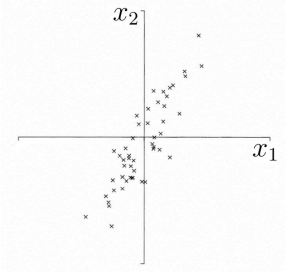
Content

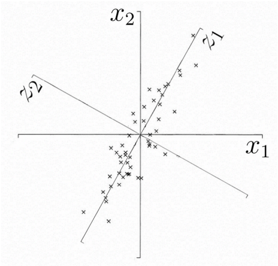
What is Principal Component Analysis?

- (Wikipedia) **PCA** is the process of computing the principal components (PCs) and using them to perform a change of **basis** on the data. (转换坐标系：orthonormal basis)
- **Variance of samples**. The principal components are ordered by the variance of PCs.
- **Goals**: 最大化投影方差 $\Longleftrightarrow$ 最小化重构距离



Geometric view of Principal Components





The first PC z_1:

- 中心化: $\mathbf{x}_i - \bar{\mathbf{x}}$
- Projection: project the data to the vector z_1
- Maximize the variance of data in z_1 basis, with $|\mathbf{z}_1| = 1$

Project data to the vector z_1

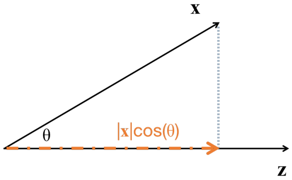
Data: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \in \mathbb{R}^{N \times p}$

- **Demean**: $\mathbf{x}_i - \bar{\mathbf{x}}$
- **Projection**: Project each data point to the vector $\mathbf{z}_1$
  $\implies a_i = (\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{z}_1$
- **Variance**:

$$\mathrm{var}[a_i] = \mathbb{E}[(a_i - \bar{a})^2]$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left((\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{z}_1 \right)^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} \mathbf{z}_1^T (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{z}_1 = \mathbf{z}_1^T S \mathbf{z}_1$$



$<\mathbf{x}, \mathbf{z}> = \mathbf{x}^T \mathbf{z} = |\mathbf{x}||\mathbf{z}|\cos(\theta)$

PCA: maximizing projection variance

- To find the optimal vector $\mathbf{z}_1^*$
- Objective: maximize the variance after projection $\mathbf{z}_1^T S \mathbf{z}_1$
- Constraints: $\mathbf{z}_1^T \mathbf{z}_1 = 1$

  Let us denote λ as a Lagrange multiplier.
  The generalized Lagrangian function is:

$$\mathcal{L}(\mathbf{z}_1, \lambda) = -\mathbf{z}_1^T S \mathbf{z}_1 + \lambda(\mathbf{z}_1^T \mathbf{z}_1 - 1)$$

  The derivative of $\mathcal{L}$ is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_1} = -2S\mathbf{z}_1 + 2\lambda\mathbf{z}_1 = S\mathbf{z}_1 - \lambda\mathbf{z}_1 = 0$$

  $\mathbf{z}_1$ is an eigenvector of S, $\lambda = \lambda_1$ is the largest eigenvalue

Eigendecomposition of covariance matrix

Similarly, $\mathbf{z}_2$ is also an eigenvector of S whose eigenvalue $\lambda = \lambda_2$ is the second largest.
In general, we have the kth PCs:

$$\mathrm{var}\left[\mathbf{z}_k\right] = \mathbf{z}_k^T S \mathbf{z}_k = \lambda_k \tag{1}$$

The k^{th} largest eigenvalue of $S \longrightarrow k^{th}$ PC $\mathbf{z}_k$.

Dimensionality reduction:
$\longrightarrow$ From p-D to q-D with $q < p$
$\longrightarrow$ Find q largest eigenvalues of the covariance matrix S.

PCA for dimensionality reduction

- **1. Feature standardization.**
  To standardize the range of the raw data so that each feature contributes equally in the following analysis.
- **2. Obtain the covariance matrix** S.
  The covariance matrix $S \in \mathbb{R}^{p \times p}$,

$$S = \frac{1}{N}\mathbf{X}^T H \mathbf{X}, \quad H = \mathbf{I}_N - \frac{1}{N}\mathbf{1}_N^T\mathbf{1}_N$$

- **3. Eigendecomposition of the covariance matrix** S.
  Obtain all eigenvectors and eigenvalues

$$S = Z\Lambda Z^T$$

- **4. Sort the eigenvalues in a descending order.**

$$\Lambda = diag([\lambda_1,\ \lambda_2, \dots,\ \lambda_p])$$

Main Steps for Principal Component Analysis

- **5. Select the number of principal components.**
  Select the top q eigenvectors (based on their eigenvalues) as the top q PCs.
- **6. Transformation matrix** G consists of the top q eigenvectors ($G \in \mathbb{R}^{p \times q}$).

$$G = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_q]$$

- **7. Project p-D data to q-D PC space**

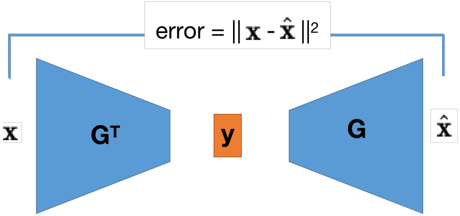
$$\text{For each sample:} \quad \mathbf{y}_i = G^T \mathbf{x}_i$$
$$\text{For entire dataset:} \quad Y = XG$$

PCA from reconstruction perspective

- **Data**: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)^T \in \mathbb{R}^{N \times p}$
- **Projections** with PCA as encoder and decoder,

$$\begin{aligned} \text{Encoder:} \quad \mathbf{y}_i &= G^T \mathbf{x}_i \in \mathbb{R}^q \\ \text{Decoder:} \quad \hat{\mathbf{x}}_i &= G \mathbf{y}_i \quad\; \in \mathbb{R}^p \end{aligned}$$

where $G = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_q] \in \mathbb{R}^{p \times q}$.



Reconstruction error

- Reconstruction error across N samples:

$$\mathcal{L}(W) = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \tag{2}$$

- Substitute $G = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_q]$ to Eq. (2)

$$= \frac{1}{N}\sum_{i=1}^{N} \|ZZ^T\mathbf{x}_i - GG^T\mathbf{x}_i\|^2$$

$$= \frac{1}{N}\sum_{i=1}^{N} \|\sum_{k=q+1}^{p} \mathbf{z}_k\mathbf{z}_k^T\mathbf{x}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N} (\sum_{k=q+1}^{p} \mathbf{z}_k^T\mathbf{x}_i)^2$$

$$= \sum_{k=q+1}^{p} \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i^T\mathbf{z}_k)^2 = \sum_{k=q+1}^{p} \mathbf{z}_k^T {\color{red}S} \mathbf{z}_k$$

minimize error = maximize variance

Minimize reconstruction error（最小化重构误差）：

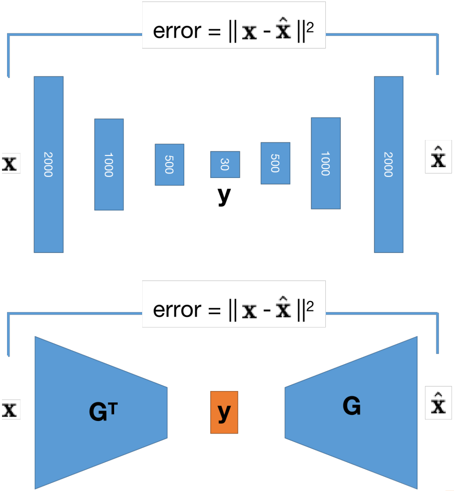
$$\mathbf{z}_k = \arg\min_{\mathbf{z}_k} \sum_{k=q+1}^{p} \mathbf{z}_k^T S \mathbf{z}_k$$
$$\text{s.t.} \quad \mathbf{z}_k^T \mathbf{z}_k = 1$$

Maximizing projection variance（最大化投影方差）：

$$\mathbf{z}_i = \arg\max_{\mathbf{z}_i} \mathbf{z}_i^T S \mathbf{z}_i$$
$$\text{s.t.} \quad \mathbf{z}_i^T \mathbf{z}_i = 1$$

*Autoencoder: nonlinear data reconstruction

Autoencoder: [from *Hinton & Salakhutdinov (2006) Science*]

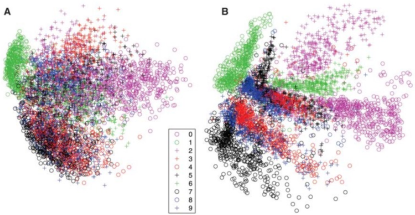


*Hinton & Salakhutdinov (2006) Science

Title: Reducing the Dimensionality of Data with Neural Networks

Abstract: High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in *such "autoencoder" networks*, **but** this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than *principal components analysis* as a tool to reduce the dimensionality of data.

Fig. 3. **(A)** The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. **(B)** The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



Mean and variance

- **Data**: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \in \mathbb{R}^{N \times p}$
- **Mean** of $\mathbf{X}$: $(\bar{\mathbf{x}} \in \mathbb{R}^p)$

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i = \frac{1}{N}\mathbf{X}^T \mathbf{1}_N$$

where $\mathbf{1}_N = (1, \ 1, \dots, \ 1)^T$. (N 个 1 的列向量)

- **Covariance** of $\mathbf{X}$: $(S \in \mathbb{R}^{p \times p})$

$$\begin{aligned} S &= \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \\ &= \frac{1}{N}\mathbf{X}^T H H^T \mathbf{X} = \frac{1}{N}\mathbf{X}^T H \mathbf{X} \end{aligned}$$

where $H = \mathbf{I}_N - \frac{1}{N}\mathbf{1}_N^T \mathbf{1}_N, \quad H^T = H, \quad H^2 = H.$

SVD on HX

- **Singular Value Decomposition** (SVD) on the matrix HX

$$HX = U\Sigma V^T \tag{3}$$

$$U^T U = \mathbf{I}_p, \quad V^T V = VV^T = \mathbf{I}_p, \quad \Sigma = diag([\sigma_1, \ldots, \sigma_p])$$

- Substitute Eq. (3) to the covariance matrix S, we derive

$$\begin{aligned} S &= \frac{1}{N}\mathbf{X}^T HH^T \mathbf{X} = \frac{1}{N}(HX)^T(HX) \\ &= \frac{1}{N}(U\Sigma V^T)^T U\Sigma V^T = \frac{1}{N} V\Sigma U^T U\Sigma V^T \\ &= \frac{1}{N} V\Sigma^2 V^T \Longrightarrow \text{Eigendecoposition of } S \end{aligned} \tag{4}$$

Watch Gilbert Strange 30:
`https://www.bilibili.com/video/BV1zx411g7gq?p=30`

Relationship between PCA and SVD

表 1: Relationship between PCA and SVD

PCA	**SVD**
Eigendecomposition on S	SVD on HX
$S = Z\Lambda Z^T$	$HX = U\Sigma V^T$ $S = \frac{1}{N}V\Sigma^2 V^T$
$\Lambda = diag([\lambda_1, \lambda_2, \ldots, \lambda_p])$	$\Sigma^2 = diag([\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2])$

1. Recap
2. PCA: from reconstruction perspective
3. PCA: from SVD perspective
4. PCA: applications
5. Summary

PCA: Using sklearn.decomposition

```
from sklearn.decomposition import PCA
from sklearn import preprocessing

scaled_data = preprocessing.scale(data.T)
pca =PCA()
pca.fit(scaled_data)
pca_data=pca.transform(scaled_data)
```

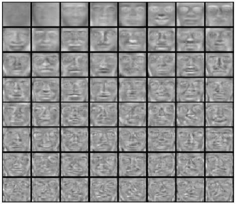
Highly recommend: ‘StatQuest’ in Bilibili
https://www.bilibili.com/video/BV1Tb41187qb/?p=35
PCA in python

A classical example: Eigenface

Dataset

Mean Face

Eigenfaces (PCs Visualization)



Face Reconstruction

Please watch the video by Prof. Steve Brunton

1. `https://www.youtube.com/watch?v=ofWji_wQBEE`
2. `https://www.youtube.com/watch?v=yYdYrAKghF4`
3. `https://www.youtube.com/watch?v=SsNXg6KpLSU`

PCA as feature extraction for classification

We usually use PCA for *dimensionality reduction*.
PCA as *feature extraction* for classification:

- Project both training and testing data into the PCs space

$$Y = XG \quad \text{for the entire dataset}$$

- Run logistic regression or SVM on the q-D PCs space
- **Problem 1**: The classification accuracy may be sensitive to the choice of q
- Solution: $\longrightarrow$ Plot the accuracy curve with the varying q

PCA as feature extraction for classification

PCA as feature extraction for classification:

- **Problem 2**: PCA may not be suitable for classification.
- Q: Why?

  PCA is based on the sample covariance S which characterizes the scatter of the entire data set, **regardless of the class label**.

  The projection axes chosen by PCA might not provide good discrimination between classes.
- Solution: $\longrightarrow$ Supervised feature extraction

PCA: Eigendecomposition on the covariance matrix S

- To find the optimal vector $\mathbf{z}_1^*$
- Objective: maximize the variance after projection $\mathbf{z}_1^T S \mathbf{z}_1$
- Constraints: $\mathbf{z}_1^T \mathbf{z}_1 = 1$

Let us denote λ as a Lagrange multiplier.
The generalized Lagrangian function is:

$$\mathcal{L}(\mathbf{z}_1, \lambda) = -\mathbf{z}_1^T S \mathbf{z}_1 + \lambda(\mathbf{z}_1^T \mathbf{z}_1 - 1)$$

The derivative of $\mathcal{L}$ is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_1} = -2S\mathbf{z}_1 + 2\lambda\mathbf{z}_1 = S\mathbf{z}_1 - \lambda\mathbf{z}_1 = 0$$

$\mathbf{z}_1$ is an eigenvector of S, $\lambda = \lambda_1$ is the largest eigenvalue

Summary

最大化投影方差 = 最小化重构误差

Maximize projection variance = Minimize reconstruction error

Implementation of PCA:
Eigendecomposition on the covariance matrix S
= SVD on the data matrix HX

PCA is a good tool for dimensionality reduction, but it may not be a good tool for feature extraction.

Additional materials:

1. Lecture: Eigendecomposition, SVD by Prof. Gilbert Strang
2. Youtube videos: Eigenface by Prof. Steve Brunton
3. Paper: Hinton & Salakhutdinov (2006) Science