Week 7

Lecture 1

Principal Component Analysis Part 1

Motivation and Overview

Principal Component Analysis is a dimensionality reduction technique used for feature selection.
Given high dimensional data, the goal is to project the data onto a lower dimensional subspace such that:

1. Reconstruction error is minimized.
2. Variance of projected data is maximized.
   In this lecture, we adopt the first viewpoint.

Problem Formulation

Dataset

Let the dataset be

$$D = \{x_1, x_2, \dots, x_n\}, \quad x_i \in \mathbb{R}^d$$

Goal

Project the dataset onto an m-dimensional subspace, where $m < d$ is fixed in advance.

Step 1: Fix an m-Dimensional Subspace

Let

$$B = \{u_1, \dots, u_m\}$$

be an orthonormal basis for an m-dimensional subspace of $\mathbb{R}^d$.
Extend B to a full orthonormal basis

$$B' = \{u_1, \dots, u_m, u_{m+1}, \dots, u_d\}$$

for $\mathbb{R}^d$.

Representation of Data Points

Since B' is an orthonormal basis, any $x \in \mathbb{R}^d$ can be written as

$$x = \sum_{j=1}^{d} (x^T u_j) u_j$$

In particular,

$$x_i = \sum_{j=1}^{d} (x_i^T u_j) u_j$$

Projection onto the m-Dimensional Subspace

Approximate x_i by $\tilde{x}_i$:

$$\tilde{x}_i = \sum_{j=1}^{m} z_{ij} u_j + \sum_{j=m+1}^{d} \beta_j u_j$$

where z_{ij} and β_j are coefficients to be determined.

Reconstruction Error

Define the reconstruction error:

$$J = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \tilde{x}_i\|^2$$

Substituting expansions,

$$x_i - \tilde{x}_i = \sum_{j=1}^{m} (x_i^T u_j - z_{ij}) u_j + \sum_{j=m+1}^{d} (x_i^T u_j - \beta_j) u_j$$

Since $\{u_j\}$ is orthonormal,

$$\left\| \sum c_j u_j \right\|^2 = \sum c_j^2$$

Thus,

$$J = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j=1}^{m}(x_i^T u_j - z_{ij})^2 + \sum_{j=m+1}^{d}(x_i^T u_j - \beta_j)^2\right]$$

Minimizing with Respect to z_{ij}

Compute

$$\frac{\partial J}{\partial z_{ij}} = 2(x_i^T u_j - z_{ij}) = 0$$

Hence,

$$z_{ij} = x_i^T u_j$$

Minimizing with Respect to β_j

Compute

$$\frac{\partial J}{\partial \beta_j} = \frac{2}{n}\sum_{i=1}^{n}(x_i^T u_j - \beta_j) = 0$$

Thus,

$$\beta_j = \frac{1}{n}\sum_{i=1}^{n} x_i^T u_j$$

Define the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Then,

$$\beta_j = \bar{x}^T u_j$$

Optimal Projection

With optimal coefficients,

$$\tilde{x}_i = \sum_{j=1}^{m}(x_i^T u_j)u_j + \sum_{j=m+1}^{d}(\bar{x}^T u_j)u_j$$

If data is centered,

$$\bar{x} = 0$$

and the second term vanishes:

$$\tilde{x}_i = \sum_{j=1}^{m} (x_i^T u_j) u_j$$

Expression for Reconstruction Error

Compute

$$x_i - \tilde{x}_i = \sum_{j=m+1}^{d} (x_i^T u_j - \bar{x}^T u_j) u_j$$

Thus,

$$\|x_i - \tilde{x}_i\|^2 = \sum_{j=m+1}^{d} ((x_i - \bar{x})^T u_j)^2$$

Hence optimal error:

$$J^* = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=m+1}^{d} ((x_i - \bar{x})^T u_j)^2$$

Interchanging summations,

$$J^* = \frac{1}{n} \sum_{j=m+1}^{d} \sum_{i=1}^{n} ((x_i - \bar{x})^T u_j)^2$$

Write as quadratic form:

$$((x_i - \bar{x})^T u_j)^2 = u_j^T (x_i - \bar{x})(x_i - \bar{x})^T u_j$$

Define covariance matrix:

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

Then,

$$J^* = \sum_{j=m+1}^{d} u_j^T C u_j$$

Key Result

For a fixed m-dimensional subspace with orthonormal basis $\{u_1, \dots, u_m\}$, the reconstruction error is

$$J^* = \sum_{j=m+1}^{d} u_j^T C u_j$$

where

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

This reduces PCA to choosing the basis vectors $\{u_j\}$ so as to minimize

$$\sum_{j=m+1}^{d} u_j^T C u_j$$

which leads to the eigenvalue formulation of PCA.

```
********************************************************************************
```

Lecture 2

Principal Component Analysis Efficient Implementation

High Dimensional Setting

Given dataset

$$\{x_1, x_2, \dots, x_n\}, \quad x_i \in \mathbb{R}^d$$

Assume

$$d \gg n$$

Standard PCA requires eigen-decomposition of covariance matrix

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Here,

$$C \in \mathbb{R}^{d \times d}$$

Computing eigenvalues of a $d \times d$ matrix is computationally expensive when d is large.

Rank Observation

Each matrix

$$(x_i - \bar{x})(x_i - \bar{x})^T$$

is rank one.
Thus,

$$\mathrm{rank}(C) \leq n$$

Therefore,

$$d - n$$

eigenvalues of C are zero.
It is unnecessary to compute all d eigenvectors.

Matrix Reformulation

Define matrix

$$A = \begin{bmatrix} (x_1 - \bar{x})^T \\ (x_2 - \bar{x})^T \\ \vdots \\ (x_n - \bar{x})^T \end{bmatrix}$$

Then,

$$C = \frac{1}{n} A^T A$$

where

$$A \in \mathbb{R}^{n \times d}$$

Key Observation

Instead of computing eigenvectors of

$$\frac{1}{n}A^TA$$

which is $d \times d$, compute eigenvectors of

$$\frac{1}{n}AA^T$$

which is $n \times n$.

Eigenvalue Correspondence

Let

$$Cu_i = \lambda_i u_i$$

where

$$C = \frac{1}{n}A^TA$$

Then,

$$\frac{1}{n}A^TAu_i = \lambda_i u_i$$

Multiply both sides by A:

$$\frac{1}{n}AA^T(Au_i) = \lambda_i(Au_i)$$

Thus,

$$\lambda_i$$

is an eigenvalue of

$$\frac{1}{n}AA^T$$

Reverse Direction

Let

$$\frac{1}{n}AA^Tv_i = \lambda_i v_i$$

Multiply by A^T:

$$\frac{1}{n}A^TA(A^Tv_i) = \lambda_i(A^Tv_i)$$

Thus,

$$A^Tv_i$$

is an eigenvector of

$$C = \frac{1}{n}A^TA$$

with eigenvalue λ_i.

Efficient PCA Algorithm

When $d \gg n$:

1. Form matrix

$$A \in \mathbb{R}^{n \times d}$$

2. Compute eigenvalues and eigenvectors of

$$\frac{1}{n}AA^T \in \mathbb{R}^{n \times n}$$

3. Recover eigenvectors of covariance matrix using

$$u_i = A^Tv_i$$

4. Normalize u_i

Connection to SVD

Since

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

then

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

and

$$AA^T = U\Sigma\Sigma^T U^T$$

Thus PCA is equivalent to performing SVD on A.

Final Conclusion

Instead of computing eigen-decomposition of

$$C \in \mathbb{R}^{d\times d}$$

one computes eigen-decomposition of

$$\frac{1}{n}AA^T \in \mathbb{R}^{n\times n}$$

This reduces computational cost when

$$d \gg n$$

and yields the same principal components.

```
********************************************************************************
```

Lecture 3

Principal Component Analysis as Maximizing Variance

Projection onto a Line

Let dataset

$$D = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d$$

Mean of data:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Let u be a unit vector:

$$u^T u = 1$$

Projection of x_i onto direction u:

$$\tilde{x}_i = (x_i^T u)u$$

Mean projected value:

$$(\bar{x}^T u)u$$

Projected Variance

Variance along direction u:

$$\frac{1}{n}\sum_{i=1}^{n}(x_i^T u - \bar{x}^T u)^2$$

Rewrite:

$$\frac{1}{n}\sum_{i=1}^{n}\left((x_i - \bar{x})^T u\right)^2$$

This equals:

$$\frac{1}{n}\sum_{i=1}^{n} u^T (x_i - \bar{x})(x_i - \bar{x})^T u$$

Define covariance matrix:

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$$

Thus projected variance becomes:

$$u^T C u$$

Optimization Problem

Goal:

$$\max_u u^T C u$$

subject to

$$u^T u = 1$$

Equivalently maximize Rayleigh quotient:

$$\max_u \frac{u^T C u}{u^T u}$$

Calculus Derivation

Let

$$u = (u^{(1)}, u^{(2)}, \ldots, u^{(d)})$$

Denominator derivative:

$$\frac{\partial}{\partial u^{(i)}}(u^T u) = 2u^{(i)}$$

Numerator:

$$u^T C u = \sum_{i=1}^{d} \sum_{j=1}^{d} C_{ij} u^{(i)} u^{(j)}$$

Derivative:

$$\frac{\partial}{\partial u^{(i)}}(u^T C u) = 2 \sum_{j=1}^{d} C_{ij} u^{(j)}$$

Using quotient rule and setting derivative to zero:

$$(u^T u) C u = (u^T C u) u$$

Thus:

$$Cu = \lambda u$$

where

$$\lambda = \frac{u^T C u}{u^T u}$$

Key Result

Maximizer of

$$\frac{u^T C u}{u^T u}$$

is an eigenvector of C.
Maximum value equals largest eigenvalue:

$$\lambda_1$$

Thus:
First principal direction = eigenvector of C corresponding to largest eigenvalue.
Projected variance along that direction equals:

$$\lambda_1$$

Extension to m Dimensions

For $m \geq 1$:
Choose orthonormal vectors

$$u_1, u_2, \ldots, u_m$$

Projected variance:

$$\sum_{k=1}^{m} u_k^T C u_k$$

Maximum achieved by selecting eigenvectors corresponding to top m eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$$

These vectors are called principal directions.
Projected coordinates are principal components.

Example

Data:

$$x_1 = (-1, -1), \quad x_2 = (0, 0), \quad x_3 = (1, 1)$$

Mean:

$$\bar{x} = 0$$

Covariance matrix eigenvalues:

$$\lambda_1 = \frac{4}{3}, \quad \lambda_2 = 0$$

Corresponding principal direction:

$$u_1 = \frac{1}{\sqrt{2}}(1, 1)$$

Projections:

$$x_1^T u_1 = -\sqrt{2}$$

$$x_2^T u_1 = 0$$

$$x_3^T u_1 = \sqrt{2}$$

Projected variance:

$$\frac{1}{3}\left((-\sqrt{2})^2 + 0 + (\sqrt{2})^2\right) = \frac{4}{3}$$

Thus:

$$\text{Projected variance} = \lambda_1 = \frac{4}{3}$$

Final Summary

PCA has two equivalent interpretations:

1. Minimize reconstruction error
2. Maximize projected variance
   Both lead to selecting eigenvectors of covariance matrix corresponding to largest eigenvalues.
   PCA algorithm:
3. Compute covariance matrix C
4. Compute eigenvalues and eigenvectors
5. Select top m eigenvectors
6. Project data onto span of these eigenvectors

```
********************************************************************************
```

Lecture 4

PCA in Higher Dimensions

Problem Setup

Dataset:

$$D = \{x_1, x_2, \dots, x_n\}, \quad x_i \in \mathbb{R}^d$$

Assume:

$$d \gg n$$

Feature dimension is much larger than number of data points.

Standard PCA Requirement

Mean:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Covariance matrix:

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

Dimension of C:

$$C \in \mathbb{R}^{d \times d}$$

Standard PCA requires computing eigenvalues and eigenvectors of C.
When d is very large, this is computationally expensive.

Rank Observation

Each matrix:

$$(x_i - \bar{x})(x_i - \bar{x})^T$$

is rank 1.
Sum of n rank 1 matrices has rank at most n.
Thus:

$$\operatorname{rank}(C) \leq n$$

Therefore:

$$d - n$$

eigenvalues of C are zero.
It is unnecessary to compute all d eigenvectors.

Matrix Reformulation

Define matrix A as:

$$A = \begin{bmatrix} (x_1 - \bar{x})^T \\ (x_2 - \bar{x})^T \\ \vdots \\ (x_n - \bar{x})^T \end{bmatrix}$$

Then:

$$C = \frac{1}{n} A^T A$$

Here:

- $A \in \mathbb{R}^{n \times d}$
- $C \in \mathbb{R}^{d \times d}$

Key Eigenvalue Relation

Let u_i be an eigenvector of C with eigenvalue $\lambda_i > 0$:

$$C u_i = \lambda_i u_i$$

Since:

$$C = \frac{1}{n} A^T A$$

We have:

$$\frac{1}{n} A^T A u_i = \lambda_i u_i$$

Multiply both sides by A:

$$\frac{1}{n} A A^T (A u_i) = \lambda_i (A u_i)$$

Thus:

$$\lambda_i$$

is an eigenvalue of:

$$\frac{1}{n}AA^T$$

which is an $n \times n$ matrix.

Reverse Direction

Suppose v_i is an eigenvector of:

$$\frac{1}{n}AA^T$$

so that:

$$\frac{1}{n}AA^T v_i = \lambda_i v_i$$

Multiply both sides by A^T:

$$\frac{1}{n}A^T A(A^T v_i) = \lambda_i (A^T v_i)$$

Thus:

$$A^T v_i$$

is an eigenvector of:

$$C = \frac{1}{n}A^T A$$

Core Result

Instead of computing eigenvalues and eigenvectors of:

$$C = \frac{1}{n}A^T A \quad \text{of size } d \times d$$

it is sufficient to compute eigenvalues and eigenvectors of:

$$\frac{1}{n}AA^T \quad \text{of size } n \times n$$

Since:

$$n \ll d$$

this reduces computational cost significantly.

Practical PCA Algorithm in High Dimensions

1. Construct centered data matrix A.
2. Compute $n \times n$ matrix:

$$\frac{1}{n} A A^T$$

3. Compute its eigenvalues and eigenvectors.
4. Recover eigenvectors of C using:

$$u_i = A^T v_i$$

5. Normalize resulting vectors.

Connection to SVD

Recall Singular Value Decomposition:

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

Then:

$$A^T A = V \Sigma^2 V^T$$

$$A A^T = U \Sigma^2 U^T$$

Thus:

- Eigenvectors of $A^T A$ are right singular vectors.
- Eigenvectors of $A A^T$ are left singular vectors.
- Eigenvalues correspond to squared singular values.

Therefore:
PCA in high dimensions can be implemented efficiently using SVD.

Final Conclusion

When:

$$d \gg n$$

PCA can be implemented efficiently by:

- Working with $n \times n$ matrix instead of $d \times d$ matrix
- Computing eigenvalues and eigenvectors of $\frac{1}{n}AA^T$
- Recovering principal directions from these eigenvectors

This avoids direct computation on large covariance matrices and yields identical PCA results.

```
***************************************************************************************
```
