Week-2: Enhancing PCA with Kernels

Sherry Thomas
21f3001449

Contents

Introduction 1

Reducing the Time Complexity to Find Eigenvalues and Eigenvectors 2

Finding PCA for Non-Linear Relationships 3

Transforming Features 3

Kernel Functions 3

Kernel PCA 4

Acknowledgments 4

Abstract

The week's discourse concentrates on the two primary challenges inherent in Principal Component Analysis (PCA) and endeavors to provide solutions. The solutions are achieved through the utilization of kernel functions and culminates in a more comprehensive and generalized algorithm for PCA.

Introduction

In the context of a given dataset $\mathbf{X} \in \mathbb{R}^{d\times n}$, where $\mathbf{C} \in \mathbb{R}^{d\times d}$ represents the covariance matrix, Principal Component Analysis (PCA) encounters two critical challenges:

1. **Time Complexity**: Computing the eigenvalues and eigenvectors of $\mathbf{C}$ requires an algorithmic complexity of $O(d^3)$. Consequently, as the dimensionality d increases, the computational time becomes prohibitively large.
2. **Non-Linear Dataset**: In situations where the dataset resides in a non-linear subspace, PCA's attempt to derive linear combinations of Principal Components may yield suboptimal results.

To address these issues, we propose methods for reducing the time complexity of finding eigenvalues and eigenvectors and for handling non-linear relationships in PCA.

Reducing the Time Complexity to Find Eigenvalues and Eigenvectors

Let us consider a dataset $\mathbf{X}$ with a large number of features (d), denoted as $[d >> n]$, where:

- d: number of features
- n: number of datapoints

The dataset $\mathbf{X}$ is represented as follows:

$$\mathbf{X} = \begin{bmatrix} | & | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_n \\ | & | & | & & | \end{bmatrix} \quad \mathbf{X} \in \mathbb{R}^{d \times n}$$

The covariance matrix $\mathbf{C}$ of $\mathbf{X}$ is given by:

$$\mathbf{C} = \frac{1}{n}(\mathbf{X}\mathbf{X}^T) \quad \text{where } \mathbf{C} \in \mathbb{R}^{d \times d} \quad \dots [1]$$

Let $\mathbf{w}_k$ be the eigenvector corresponding to the k-th largest eigenvalue λ_k of $\mathbf{C}$. We know:

$$\mathbf{C}\mathbf{w}_k = \lambda_k \mathbf{w}_k$$

Substituting $\mathbf{C}$ from equation [1] into the above equation and solving for $\mathbf{w}_k$, we obtain:

$$\mathbf{w}_k = \sum_{i=1}^{n} \left(\frac{\mathbf{x}_i^T \mathbf{w}_k}{n\lambda_k} \right) \cdot \mathbf{x}_i$$

Thus, $\mathbf{w}_k$ is a linear combination of the datapoints. Consequently, we can express:

$$\mathbf{w}_k = \mathbf{X}\alpha_k \quad \text{for some } \alpha_k \quad \dots [2]$$

Let $\mathbf{X}^T\mathbf{X} = \mathbf{K}$, where $\mathbf{K} \in \mathbb{R}^{n \times n}$. We can utilize this to solve for α_k. After some algebra (refer to relevant literature), we obtain:

$$\mathbf{K}\alpha_k = (n\lambda_k)\alpha_k$$

According to the Spectral Theorem, the non-zero eigenvalues of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$ are identical. Therefore, we can assert that if $\lambda \neq 0$ is an eigenvalue of $\mathbf{X}\mathbf{X}^T$ with eigenvector $\mathbf{w}$, then $\mathbf{X}\mathbf{X}^T\mathbf{w} = \lambda\mathbf{w}$. By multiplying both sides by $\mathbf{X}^T$, we find $\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{w}) = \lambda\mathbf{X}^T\mathbf{w}$, which confirms that λ is also an eigenvalue for $\mathbf{X}^T\mathbf{X}$, with the corresponding eigenvector being $\mathbf{X}^T\mathbf{w}$.

Let β_k be the eigenvector corresponding to the k-th largest eigenvalue $n\lambda_k$ of $\mathbf{K}$. Solving the eigen equation for $\mathbf{K}$ yields:

$$\alpha_k = \frac{\beta_k}{\sqrt{n\lambda_k}} \quad \dots [3]$$

By employing equations [2] and [3], we can compute the eigenvalues and eigenvectors of $\mathbf{C}$ using $\mathbf{K}$, effectively reducing the time complexity from $O(d^3)$ to $O(n^3)$.

Finding PCA for Non-Linear Relationships

Transforming Features

To address non-linear relationships, we propose mapping the dataset to higher dimensions as follows:

$$\mathbf{x} \to \phi(\mathbf{x}) \quad \mathbb{R}^d \to \mathbb{R}^D \quad \text{where } [D >> d]$$

To compute D, let $\mathbf{x} = \begin{bmatrix} f_1 & f_2 \end{bmatrix}$ represent features of a dataset containing datapoints lying on a second-degree curve in a two-dimensional space.

To convert it from quadratic to linear, we map the features to:

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 & f_1^2 & f_2^2 & f_1 f_2 & f_1 & f_2 \end{bmatrix}$$

Mapping d features to the polynomial power p results in ${}^{d+p}C_d$ new features.

However, it is essential to note that finding $\phi(\mathbf{x})$ may be computationally demanding.

To overcome this issue, we present the solution in the subsequent section.

Kernel Functions

A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is considered a "valid" Kernel Function if it maps data points to the real numbers.

Proof of a "Valid" Kernel: There are two methods to establish the validity of a kernel:

1. **Method 1**: Explicitly exhibit the mapping to ϕ, which may be challenging in certain cases.
2. **Method 2**: Utilize Mercer's Theorem, which states that $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a valid kernel if and only if:
   - k is symmetric, i.e., $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
   - For any dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, where $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is Positive Semi-Definite.

Two popular kernel functions are:

1. **Polynomial Kernel**: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x} + 1)^p$
2. **Radial Basis Function Kernel or Gaussian Kernel**: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$

Kernel PCA

Consider a dataset $\mathbf{X}$ with a large number of features (d), represented as $[d >> n]$, where:

- d: number of features
- n: number of datapoints

The dataset $\mathbf{X}$ is given by:

$$\mathbf{X} = \begin{bmatrix} | & | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_4 \\ | & | & | & & | \end{bmatrix}$$

To perform Kernel PCA, follow these steps:

1. **Step 1**: Calculate the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ using a kernel function, where $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.
2. **Step 2**: Center the kernel using the formula:

$$\mathbf{K}^C = \mathbf{K} - \mathbf{IK} - \mathbf{KI} + \mathbf{IKI}$$

   where $\mathbf{K}^C$ is the centered kernel, and $\mathbf{I} \in \mathbb{R}^{n \times n}$ is a matrix with all elements equal to $\frac{1}{n}$.

3. **Step 3**: Compute the eigenvectors $\{\beta_1, \beta_2, \dots, \beta_l\}$ and eigenvalues $\{n\lambda_1, n\lambda_2, \dots, n\lambda_l\}$ of $\mathbf{K}^C$ and normalize them to obtain:

$$\forall u \quad \alpha_u = \frac{\beta_u}{\sqrt{n\lambda_u}}$$

4. **Step 4**: Compute the transformed data points:

$$\phi(\mathbf{x}_i)^T \mathbf{w} \in \mathbb{R}^d \rightarrow \begin{bmatrix} \sum_{j=1}^{n} \alpha_{1j} \mathbf{K}^C_{ij} & \sum_{j=1}^{n} \alpha_{2j} \mathbf{K}^C_{ij} & \dots & \sum_{j=1}^{n} \alpha_{nj} \mathbf{K}^C_{ij} \end{bmatrix}$$

Acknowledgments

Professor Arun Rajkumar: The content, including the concepts and notations presented in this document, has been sourced from his slides and lectures. His expertise and educational materials have greatly contributed to the development of this document.

ChatGPT: The AI language model used in this document has made corrections and improvements to the notations and language, ensuring clarity and accuracy in the presentation of the material. Its capabilities have enhanced the overall quality and readability of this document.