Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

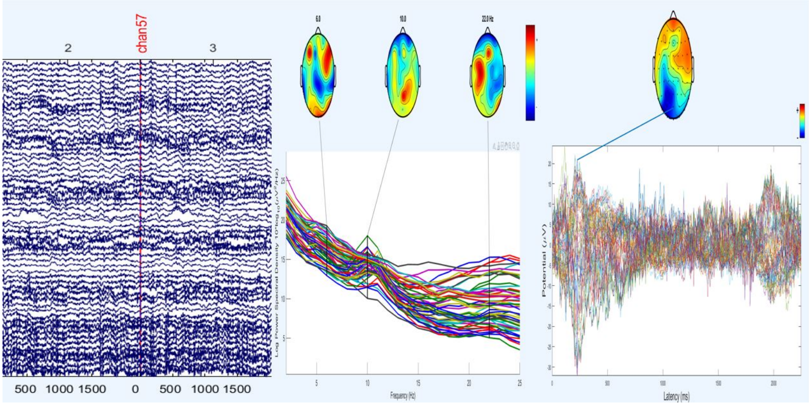
September 20, 2024

1. Introduction
2. Principal Component Analysis (PCA)
3. Choose PCs
4. Applications
5. Shortcomings
6. Conclusion
7. References

1. Introduction
2. Principal Component Analysis (PCA)
3. Choose PCs
4. Applications
5. Shortcomings
6. Conclusion
7. References

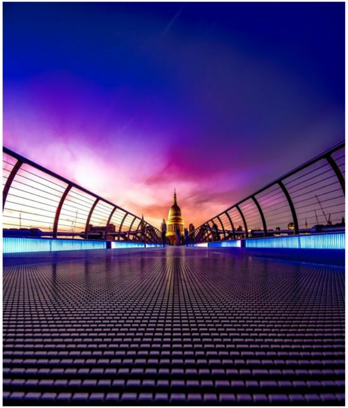
High Dimensional Data

- High-Dimensions = Lots of Features
- EEG Signals of Brain 64 Channels $*$ 3000 Time Points For Each Trial



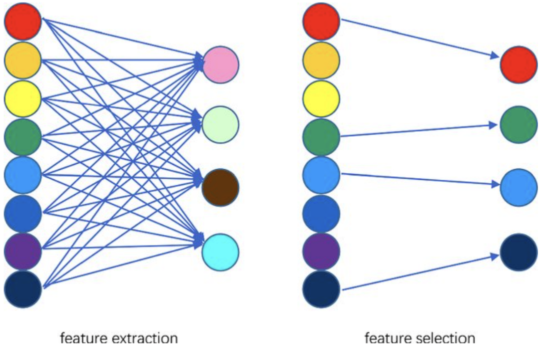
High Dimensional Data

- High-Dimensions = Lots of Features
- High Resolution Images (Millions of Pixels)



Dimensionality Reduction

- **Feature Selection**
  - Select a subset of a given feature set
- **Feature Extraction**
  - A linear or non-linear transform on the original feature space



Dimensionality Reduction Benefits

- **Visualization**
  - Project high dimensional data into 2D or 3D
- **More efficient use of resources**
  - Time, Memory, CPU
- **Statistical**
  - Fewer dimensions leads to better generalization
- **Removing Noise**
- **Pre-Process**
  - Improve accuracy by reducing features
  - As a Preprocessing step to reduce dimensions for supervised learning tasks
  - Helps avoiding overfitting

1 Introduction

2 Principal Component Analysis (PCA)
  - Sequential Algorithm
  - Sample Covariance Matrix Algorithm

3 Choose PCs

4 Applications

5 Shortcomings

6 Conclusion

7 References

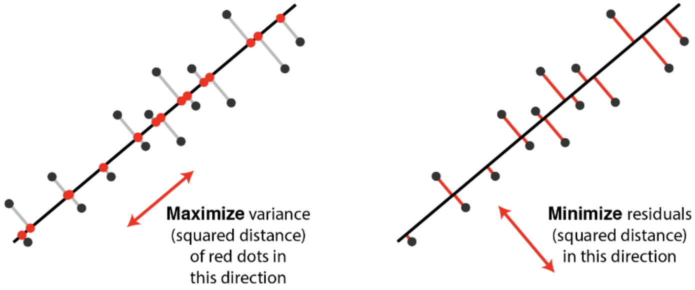
Definition

- **Goal** is reducing the dimensionality of the data while preserving important aspects of the data
- **Principal Components (PCs)** are orthogonal vectors that are ordered by the fraction of the total information (variation) in the corresponding directions
  - Find the directions at which data approximately lie

Definition

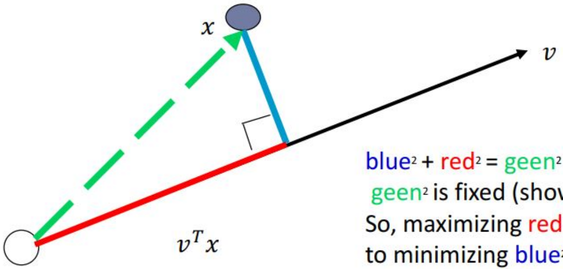
Orthogonal projection of the data onto a **lower-dimensional** linear space that:

- Maximizes variance of projected data
- Minimizes sum of squared distances to the line



Definition

- Minimizing sum of square distances to the line is **equivalent** to maximizing the sum of squares of the projections on that line.



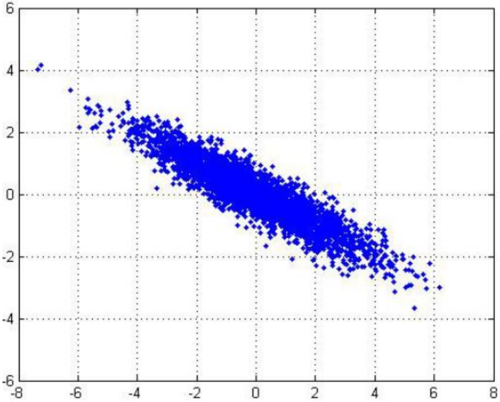
blue² + red² = geen²
geen² is fixed (shows data)
So, maximizing red² is equivalent to minimizing blue²

Idea

- Given data points in a d-dimensional space, project them into a lower dimensional space while preserving as much information as possible,
  - Find best planar approximation of 3D data
  - Find best 12-D approximation of 104-D data
- In particular, choose projection that minimizes squared error in reconstructing the original data

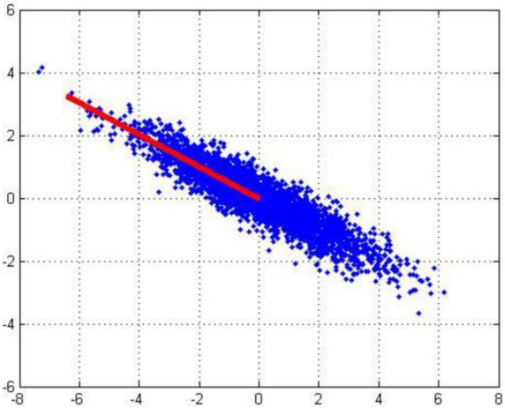
Idea

- 2D Gaussian dataset



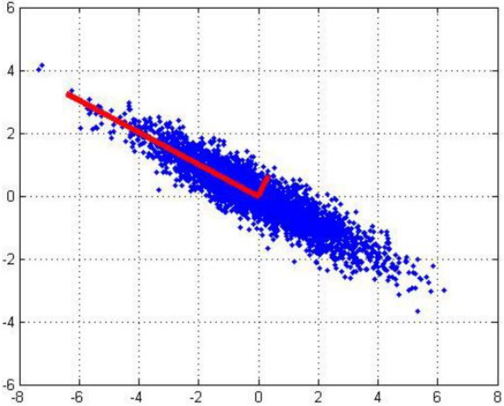
Idea

- 2D Gaussian dataset
- First PCA axis



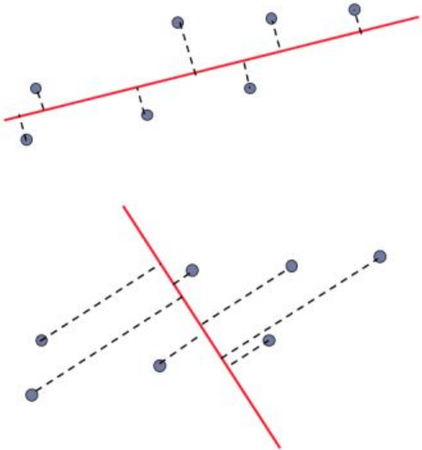
Idea

- 2D Gaussian dataset
- First and second PCA axes



Random vs Principal Projection

- Random direction vs. principal component

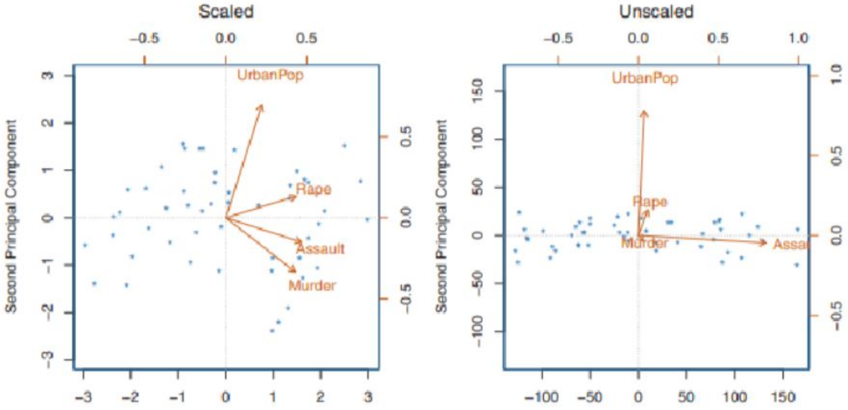


Pre-processing

- **Center the data**
  - **Zero**ing out the **mean** of each feature
- **Scaling to normalize each feature to have variance 1**
  - An arbitrary step (May affect the final result!)
  - It helps when unit of measurements of features are different and some features may be ignored without normalization

Pre-processing

- Scaling to normalize each feature may affect the final result!!



Algorithms

- Algorithm 1: sequential
- Algorithm 2: sample covariance matrix

1 Introduction

2 Principal Component Analysis (PCA)
    Sequential Algorithm
    Sample Covariance Matrix Algorithm

3 Choose PCs

4 Applications

5 Shortcomings

6 Conclusion

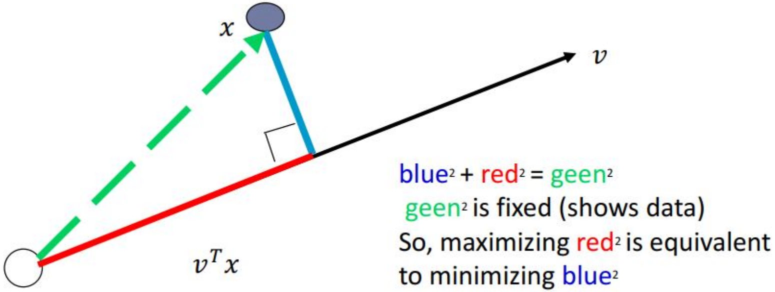
7 References

Sequential Algorithm

- First view
  - Find directions with the maximum variations

$$\max_{v_1} \frac{1}{N}\sum_{n=1}^{N}(v_1^T x_n)^2 = \frac{1}{N}\sum_{n=1}^{N} v_1^T (x_n x_n^T) v_1 = v_1^T \left(\frac{1}{N}\sum_{n=1}^{N}(x_n x_n^T)\right) v_1 = v_1^T S v_1$$

$$\text{s.t. } v_1^T v_1 = 1$$



Sequential Algorithm

- First view
- why $v_1^T v_1 = 1$?
- Eigenvector with maximum eigenvalue maximizes the objective
  - Using Lagrangian multiplier technique

$$L(v_1, \lambda) = v_1^T S v_1 + \lambda(1 - v_1^T v_1)$$

$$\frac{\partial L}{\partial v_1} = 0 \Rightarrow 2Sv_1 - 2\lambda v_1 = 0$$

$$\Rightarrow Sv_1 = \lambda v_1$$

Sequential Algorithm

- First view
- To find v_2, we maximize the variance of the projection in the residual subspace

$$v_2 = max_{v_2}\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - v_1^T x_i)^2\right)$$

$$\text{s.t. } v_2^T v_2 = 1$$

- To find v_k, we maximize the variance of the projection in the residual subspace

$$v_k = max_{v_k}\left(\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \sum_{j=1}^{k-1} W_j^T x_i\right)^2\right)$$

$$\text{s.t. } v_k^T v_k = 1$$

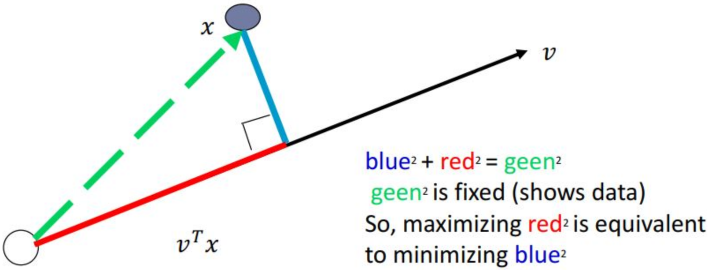
Sequential Algorithm

- Second view
  - Find directions with the minimum reconstruction error

$$\min_{v1} \sum_{n=1}^{N} \|x_n - (v_1^T x_n) v_1\|_2^2$$

$$\text{s.t. } v_1^T v_1 = 1$$

- Show this has an equal solution with the first view



Sequential Algorithm

- As we have $Sv_j = \lambda_j v_j$,

$$\Rightarrow \mathrm{var}(v_j^T x) = v_j^T x x^T v_j = v_j^T S v_j = \lambda_j v_j^T v_j = \lambda_j$$

.

- The variance along an eigenvector v_j equals the eigenvalue λ_j.

Sequential Algorithm

- Eigenvalues: $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots$
  - The first PC v_1 is the the eigenvector of the sample covariance matrix S associated with the largest eigenvalue
  - The 2nd PC v_2 is the the eigenvector of the sample covariance matrix S associated with the second largest eigenvalue
  - And so on ...
- To reduce the dimension of the data to k, we select eigenvectors with the top k eigenvalues

1. Introduction
2. **Principal Component Analysis (PCA)**
   - Sequential Algorithm
   - **Sample Covariance Matrix Algorithm**
3. Choose PCs
4. Applications
5. Shortcomings
6. Conclusion
7. References

Sample Covariance Matrix

- Given data $x_1, \ldots, x_n$, compute covariance matrix Σ

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \text{ where } \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- PCA basis vectors = the eigenvectors of Σ
- Larger eigenvalue → more important eigenvectors

Sample Covariance Matrix

Algorithm 1 Sample Covariance Matrix

1: **Input:** $X \in \mathbb{R}^{N \times d}$ (data matrix with N data points and d dimensions)
2: Compute the mean of each feature: $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
3: Subtract the mean from each data point (center the data): $\tilde{X} \leftarrow X - 1_N \bar{x}^T$
4: Compute the covariance matrix: $S = \frac{1}{N} \tilde{X}^T \tilde{X}$
5: Compute the eigenvalues and eigenvectors of S: $[\lambda_1, \lambda_2, \ldots, \lambda_d], [v_1, v_2, \ldots, v_d] = \text{eig}(S)$
6: Select the top K eigenvectors corresponding to the largest eigenvalues: $A \leftarrow [v_1, v_2, \ldots, v_K]$
7: Transform the data into the new subspace: $X' \leftarrow X \cdot A$
8: **Output:** $X' \in \mathbb{R}^{N \times K}$ (transformed data with reduced dimensions)

Sample Covariance Matrix

- Eigen-vectors of symmetric matrices are orthogonal
- Covariance matrix is symmetric
  - Principal component are orthonormal
- We have

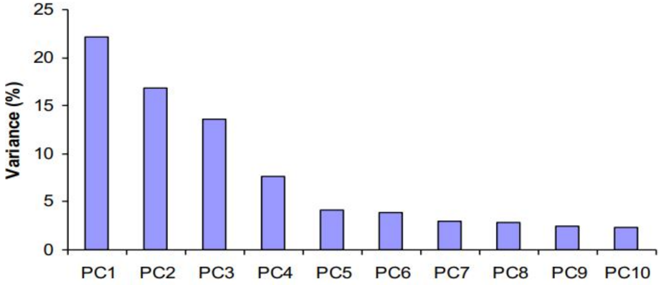
$$v_i^T v_j = 0, \quad \forall i \neq j$$

$$v_i^T v_i = 1, \quad \forall i$$

1. Introduction
2. Principal Component Analysis (PCA)
3. Choose PCs
4. Applications
5. Shortcomings
6. Conclusion
7. References

How many PCs?

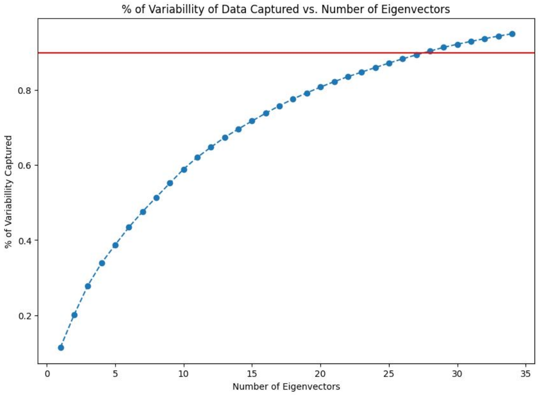
- For n original dimensions, sample covariance matrix is $n * n$, and has up to n eigenvectors. So n PCs
- Can ignore the components of lesser significance



- You do lose some information, but if the eigenvalues are small, you don't lose much

How many PCs?

$$\frac{\sum_{i=d-k+1}^{d} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$



1 Introduction

2 Principal Component Analysis (PCA)

3 Choose PCs

4 Applications

5 Shortcomings

6 Conclusion

7 References

Image Compression

- Divide the original 372x492 image into patches
  - Each patch is an instance that contains 12x12 pixels on a grid
- Consider each as a 144-D vector

Image Compression

- 144D ⇒ 60D

Image Compression

- 144D ⇒ 16D

Image Compression

- 16 most important eigenvectors

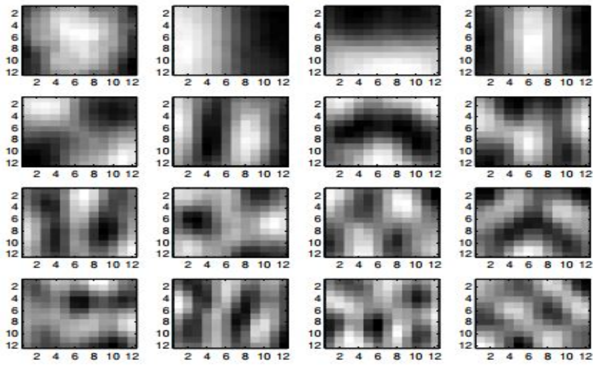


Image Compression

- 144D ⇒ 3D

Image Compression

- 3 most important eigenvectors

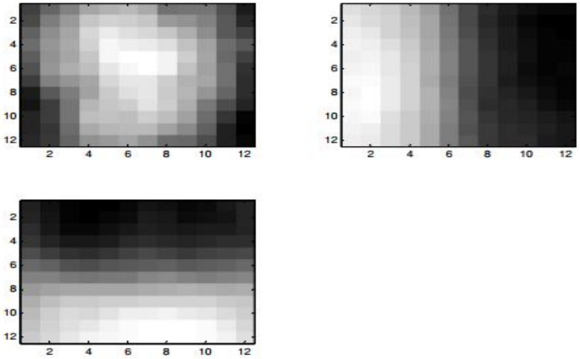
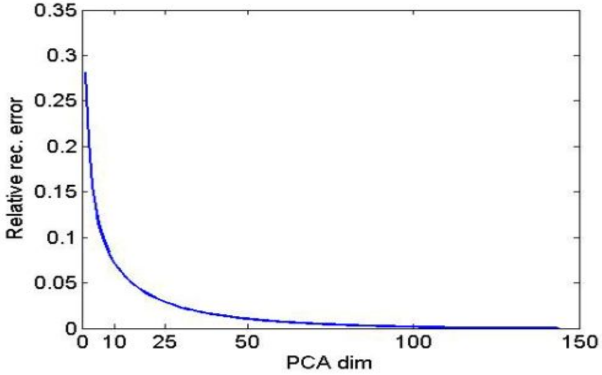


Image Compression

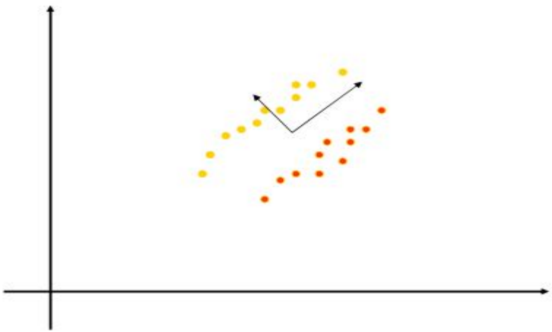
- L2 error and PCA dim



1. Introduction
2. Principal Component Analysis (PCA)
3. Choose PCs
4. Applications
5. Shortcomings
6. Conclusion
7. References

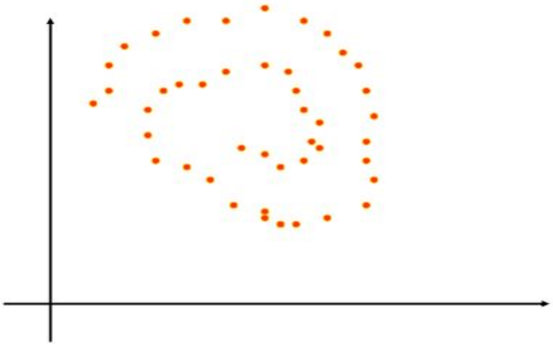
Class Labels

- PCA doesn't know about class labels!



Non-Linear

- PCA cannot capture Non-Linear structure!



1 Introduction

2 Principal Component Analysis (PCA)

3 Choose PCs

4 Applications

5 Shortcomings

6 Conclusion

7 References

Conclusion

- PCA
  - finds orthonormal basis for data
  - Sorts dimensions in order of “importance”
  - Discard low significance dimensions
- Applications
  - Get compact description
  - Remove noise
  - Improve classification (hopefully)
  - More efficient use of resources
  - Statistical
- Not magic
  - Doesn’t know class labels
  - Can only capture linear variations
- One of many tricks to reduce dimensionality!

1. Introduction
2. Principal Component Analysis (PCA)
3. Choose PCs
4. Applications
5. Shortcomings
6. Conclusion
7. References

References

- Advanced Introduction to Machine Learning, CMU-10715, Barnabás Póczos
- 10-701 Introduction to Machine Learning, CMU, Matt Gormley
- 10-301/10-601 Introduction to Machine Learning, CMU, Matt Gormley
- CE-477: Machine Learning - CS-828: Theory of Machine Learning, Sharif University of Technology, Fatemeh Seyyedsalehi