Module 6: Dimensionality Reduction

Lecture 30: Principal Component Analysis (PCA)

Prev topic | Next topic | Prev page | Next page

The Lecture Contains:

Principal Component Analysis (PCA)

- Way of identifying patterns in data
    - How input basis vectors are correlated for the given data
- A transformation from a set of (possibly correlated) axes to another set of uncorrelated axes
- Orthogonal linear transformation (i.e., rotation)
- New axes are principal components
- First principal component produces projections that are best in the squared error sense
- Optimal least squares solution

Algorithm

- Mean center the data (optional)
- Compute the covariance matrix of the dimensions
- Find eigenvectors of covariance matrix
- Sort eigenvectors in decreasing order of eigenvalues
- Project onto eigenvectors in order

Algorithm

- Mean center the data (optional)
- Compute the covariance matrix of the dimensions
- Find eigenvectors of covariance matrix
- Sort eigenvectors in decreasing order of eigenvalues
- Project onto eigenvectors in order
- Assume data matrix is B of size $m \times n$
- For each dimension, compute mean μ_i
- Mean center B by subtracting μ_i from each column i to get A
- Compute covariance matrix C of size $n \times n$
    - If mean centered, $C = A^T A$
- Find eigenvectors and corresponding eigenvalues (V, E) of C
- Sort eigenvalues such that $e_1 \geq e_2 \geq \cdots \geq e_n$
- Project step-by-step onto the principal components $\vec{v_1}, \vec{v_2}, \ldots$, etc.

Example

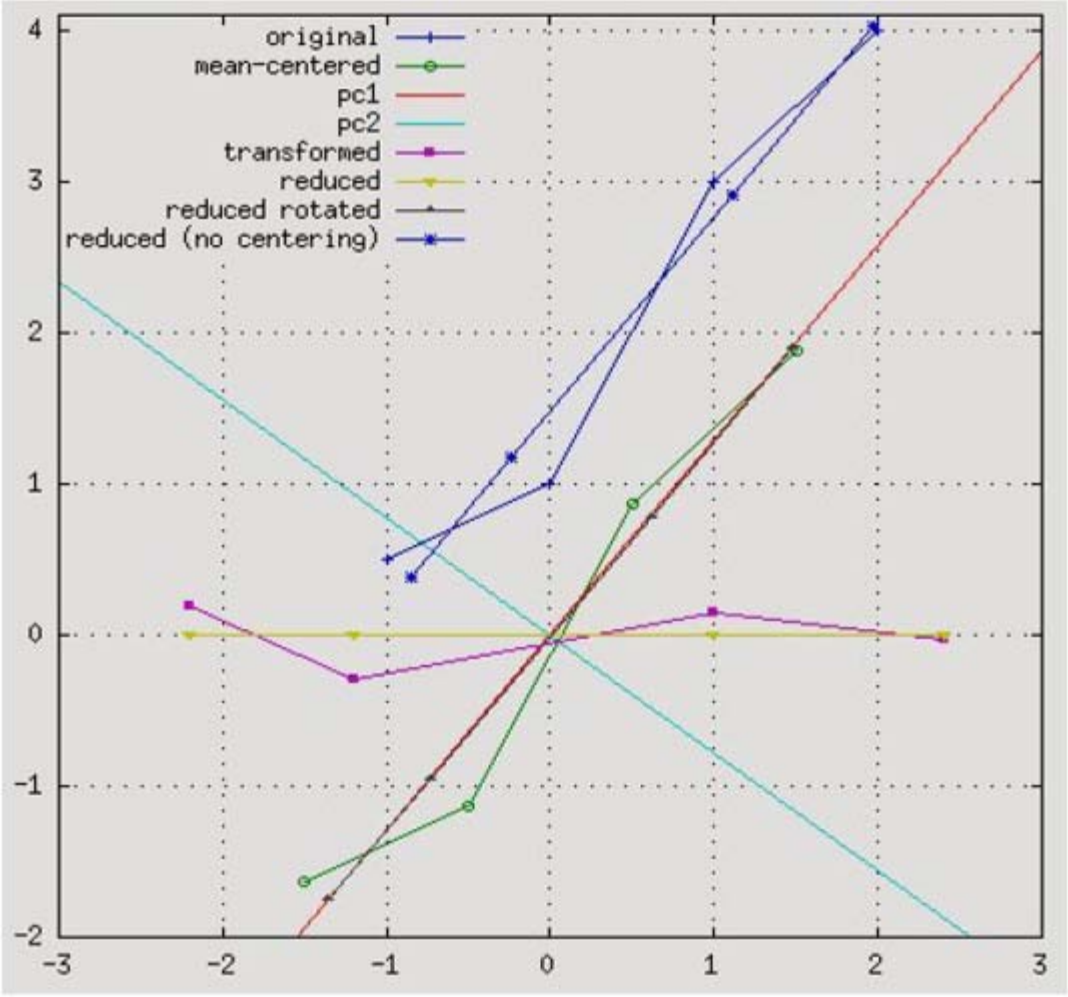
$$B = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 1 \\ -1 & 0.5 \end{bmatrix}; \mu(B) = \begin{bmatrix} 0.500 & 2.125 \end{bmatrix}$$

$$\therefore A = \begin{bmatrix} 1.5 & 1.875 \\ 0.5 & 0.875 \\ -0.5 & -1.125 \\ -1.5 & -1.625 \end{bmatrix} \text{ and, } C = A^T A = \begin{bmatrix} 5.000 & 6.250 \\ 6.250 & 8.187 \end{bmatrix}$$

$$\text{Eigenvectors } V = \begin{bmatrix} 0.613 & -0.789 \\ 0.789 & 0.613 \end{bmatrix}; \text{ eigenvalues } E = \begin{bmatrix} 13.043 \\ 0.143 \end{bmatrix}$$

$$\text{Transformed data } T = AV = \begin{bmatrix} 2.400 & -0.034 \\ 0.997 & 0.142 \\ -1.195 & -0.295 \\ -2.203 & 0.187 \end{bmatrix}$$

Visual example



Properties

- Also known as Karhunen-Loeve transform (KLT)

Properties

- Also known as Karhunen-Loeve transform (KLT)
- Works for L_2 distances only as others are not invariant to rotation

Properties

- Also known as Karhunen-Loeve transform (KLT)
- Works for L_2 distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance: $A^T A$ is covariance matrix
  - Allows use of SVD to compute PCA

Properties

- Also known as Karhunen-Loeve transform (KLT)
- Works for L_2 distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance: $A^T A$ is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix V of C is really the SVD transform matrix V for A
  - Different from SVD of B though

Properties

- Also known as Karhunen-Loeve transform (KLT)
- Works for L_2 distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance: $A^T A$ is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix V of C is really the SVD transform matrix V for A
  - Different from SVD of B though
- How many dimensions to retain?
  - Based on energy (similar to SVD)
  - Total energy is sum of eigenvalues e_i
  - Retain k dimensions such that $80\% - 95\%$ of the energy is retained

Properties

- Also known as Karhunen-Loeve transform (KLT)
- Works for L_2 distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance: $A^T A$ is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix V of C is really the SVD transform matrix V for A
  - Different from SVD of B though
- How many dimensions to retain?
  - Based on energy (similar to SVD)
  - Total energy is sum of eigenvalues e_i
  - Retain k dimensions such that $80\% - 95\%$ of the energy is retained
  - In the above example, k = 1 retains 98.91% of the energy

Properties

- Also known as Karhunen-Loeve transform (KLT)
- Works for L_2 distances only as others are not invariant to rotation
- Mean-centering
  - Easier way to compute covariance: $A^T A$ is covariance matrix
  - Allows use of SVD to compute PCA
- Can be done using SVD
  - Eigenvector matrix V of C is really the SVD transform matrix V for A
  - Different from SVD of B though
- How many dimensions to retain?
  - Based on energy (similar to SVD)
  - Total energy is sum of eigenvalues e_i
  - Retain k dimensions such that $80\% - 95\%$ of the energy is retained
  - In the above example, k = 1 retains 98.91% of the energy
- Running time: $O(mn^2 + n^3)$ for A of size $m \times n$

Multi-dimensional scaling (MDS)

- Tries to minimize stress

$$\text{Stress} = \sqrt{\frac{\sum_{\forall x,y} \left(d'(f(x), f(y)) - d(x,y)\right)^2}{\sum_{\forall x,y} \left(d(x,y)\right)^2}}$$

- Stress measures the overall or average performance of the embedding
    - In comparison, distortion measures the worst possible performance

Multi-dimensional scaling (MDS)

- Tries to minimize stress

$$\text{Stress} = \sqrt{\frac{\sum_{\forall x,y} \left(d'(f(x), f(y)) - d(x,y)\right)^2}{\sum_{\forall x,y} \left(d(x,y)\right)^2}}$$

- Stress measures the overall or average performance of the embedding
  - In comparison, distortion measures the worst possible performance
- Algorithm
  - Initializes random points
  - Moves points as long as stress is reduced or for a fixed number of iterations or when change in stress is less than a pre-defined threshold

Multi-dimensional scaling (MDS)

- Tries to minimize stress

$$\text{Stress} = \sqrt{\frac{\sum_{\forall x,y} \left(d'(f(x), f(y)) - d(x,y)\right)^2}{\sum_{\forall x,y} \left(d(x,y)\right)^2}}$$

- Stress measures the overall or average performance of the embedding
  - In comparison, distortion measures the worst possible performance
- Algorithm
  - Initializes random points
  - Moves points as long as stress is reduced or for a fixed number of iterations or when change in stress is less than a pre-defined threshold
- Sammon embedding tries to minimize another form of stress

$$\sum_{\forall x,y} \frac{\left(d'(f(x), f(y)) - d(x,y)\right)^2}{\left(d(x,y)\right)^2}$$