Dimensionality Reduction

Nishanth Nair

April 12, 2020

Overview

- Dimensionality reduction is an unsupervised learning technique.
- It is the problem of taking high dimensional data and embedding it in a lower dimension space.
- Used when we have high dimensional data (large k, but small N)
  - Curse of dimensionality
  - Problems when the model has more degrees of freedom than our data can support.
  - Danger of overfitting
- Collapsing data to the primary axes of variation can be useful.
  - Easier to interpret (statistically) and visualize.
  - Often there is a fundamental structure in high-dimensional data that can be represented in a small number of dimensions.
  - Helps in data compression: more efficient to reduce multiple related dimensions to a single dimension.
  - For example, if we have height and weight, a new dimension "size" maybe created that captures the variation of both height and weight.
  - Increase computational efficiency

Goal

- Reduce each item from k dimensions to a corresponding item with m dimensions (with m < k) by collapsing dimensions (features).

$$x^{(i)} \in \mathbb{R}^{\mathrm{k}} \implies \mathrm{z}^{(\mathrm{i})} \in \mathbb{R}^{\mathrm{m}}$$

- GOAL:
  - Minimize the loss of information during the transformation
  - Find new dimensions that still accurately represent variation in the original data
- Approaches:
  - Linear methods
    - PCA
  - Non Linear methods
  - Feature selection

Principal Component Analysis (PCA)

- Most common form of dimensionality reduction
- Idea: find a lower-dimensional surface onto which to project the original data so as to minimize the projection error

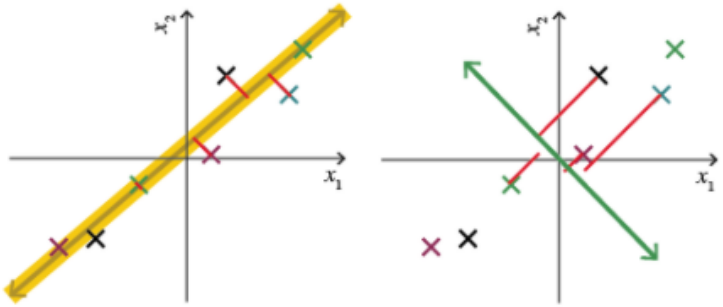

Figure 1: example

- Goal: Find vectors (principal components) and compute or project old data onto new space.

$$x^{(i)} \in \mathbb{R}^{\mathrm{k}} \implies \mathrm{z}^{(\mathrm{i})} \in \mathbb{R}^{\mathrm{m}}$$

- Generally, $z = f(x)$
- In Linear transformation, $z = U^T.x$
- How do we compute U?

Singular Value Decomposition (SVD)

- Given a nxn matrix A, the vector X (non-zero) of dimension n is called the **Eigen vector** of A if it satisfies the linear equation

$$AX = \lambda X$$

  - λ is a scalar and is called the **Eigen value** corresponding to X.
  - Eigenvectors are the vectors that the linear transformation A merely elongates or shrinks,
  - The amount that they elongate or shrink by is the eigenvalue.

- **Singular Value Decomposition**: This states

$$A_{n\times p} = U_{n\times n} S_{n\times p} V^T_{p\times p}$$

  - The Eigenvectors of A^TA make up the columns of V
  - The Eigenvectors of AA^T make up the columns of U
  - The singular values in S are the square roots of the eigenvalues from A^TA and AA^T

Example SVD computation

- Given A

$$A = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- **STEP 1: Compute U**
  - First compute AA^T

$$AA^T = W = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} 2 & 4 & 0 & 0 \\ 1 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 20 & 14 & 0 & 0 \\ 14 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

  - Compute Eigen Values of W

$$WX = \lambda X \text{ (Eigen value equation)}$$

$$\implies (W - \lambda I)X = 0$$

$$\implies \begin{bmatrix} 20-\lambda & 14 & 0 & 0 \\ 14 & 10-\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix} \times X = 0$$

From the above,

$$W - \lambda I = 0$$

$$\implies det|W - \lambda I| = 0$$

This gives us the following:

$\lambda = 0$

$\lambda = 0$

$\lambda = 29.883$

$\lambda = 0.117$

  - These values can be used to determine the eigenvector that can be placed in the

columns of U

$$\text{With } \lambda = 0.117$$

$$\begin{aligned}(20 - 0.117)x_1 + 14.x_2 &= 0\\ 14x_1 + (10 - 0.117).x_2 &= 0\\ x_3 &= 0\\ x_4 &= 0\end{aligned}$$

- Upon simplifying the first two equations we obtain a ratio which relates the value of x_1 to x_2.

- The values of x_1 and x_2 are chosen such that the elements of the S are the square roots of the eigenvalues ().

- Thus a solution that satisfies the above equation is $x_1 = -0.58, x_2 = 0.82, x_3 = 0, x_4 = 0$. (Column 2 of U)
- Similarly, using $\lambda = 29.883$, we get $x_1 = 0.82, x_2 = -0.58, x_3 = 0, x_4 = 0$. (Column 1 of U)
- Thus, U is given by:

$$U = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- **STEP 2: Compute V** using $A^T A$.

$$V = \begin{bmatrix} 0.40 & -0.91 \\ 0.91 & 0.40 \end{bmatrix}$$

- **STEP 3: Compute S** using the square root of the eigen values of $A^T A$ and AA^T

$$S = \begin{bmatrix} 5.47 & 0 \\ 0 & 0.37 \end{bmatrix}$$

PCA Algorithm

- Principal components of a dataset can be found from the first m eigenvectors and eigenvalues of the covariance matrix of the data.
- **STEP 0**:
  - Scale and standardize the data (Mean normalization)
  - Determine number of components, m.
- **STEP 1**: Compute covariance matrix of data (mean and covariance of data).

$$Q = \frac{1}{N}\sum_{i=1}^{N}(x-\mu)^T.(x-\mu)(\text{ vector form})$$

$$q_{jk} = \frac{1}{n-1}\sum_{i=1}^{N}(x_{ij}-\mu_j)(x_{ik}-\mu_k)$$

- **STEP 2**: Compute Eigenvectors

$$Q = U.S.V, \text{ (SVD of Q)}$$

  - The columns of U are the eigenvectors of QQ^T.
  - Eigenvectors represent the directions of greatest variation of the data.
  - Eigenvalues tell you how much variation is captured by each corresponding eigenvector.
- **STEP 3**: Compute Projections

$$Z = X^T.U_m, \text{where, } U_m\text{are the first m columns of U}$$

- **NOTE:**
  - One of the most common uses of PCA is to increase efficiency of other learning algorithms (e.g., first PCA, then regression).
  - Make sure to train the PCA algorithm on the training data, not on validation/testing data.
  - Transformation matrix (U) is learned on the data, and will be different depending on the data used to train it.

How many components, m?

- For visualization, m will be 2 or 3.
- How much variation do you want to explain?
  - Common statistic: Retained Variance
  - Retained Variance = Average squared projection error divided by total variation
  - "Inflation" of projected or reduced x(i) = U.z(i), where U represents eigenvectors of Q
  - Rule of thumb: retain 99% (or 95% or 90%) of variance
- Approaches:
  - Run PCA with different values until target is hit (Inefficient)
  - Using SVD:
    - The SVD of an mxn matrix Q is: Q = USV, where S is a diagonal matrix of eigenvalues on the diagonal.
    - Magnitude of eigenvalues of S indicate fraction of variance captured.
  - Run PCA and choose the number of components where the projections explain 99% of the variance of the original data.

Axis	Variance	Cumulative
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100.0%

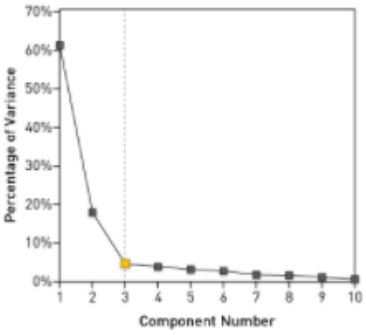

Figure 2: choosing m

- In the above, 3 PCs account for roughly 84% of variance of original data.Adding more PCs add very little value.

PCA Example: Eigenfaces

- Initialization (training):
  - Use initial set of facial images as training data.
    - 256 pixel x 256 pixel array = 65,536 pixels or features
    - 8 bit intensity
  - Calculate PCs (eigenvectors, a.k.a. "eigenfaces") and retain components with the largest eigenvalues.
  - Project original data onto "face space."

Figure 3: Training

- Recognition:
  - Project unknown image onto "face space."
  - Classify projection as known face, unknown face, or not face.

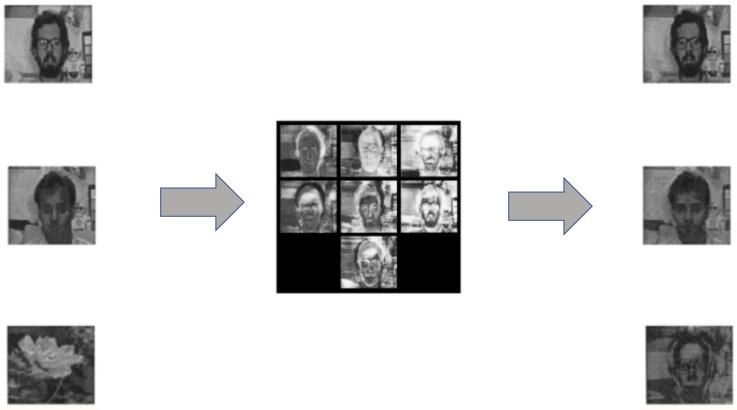

Figure 4: Recognition

- Uses:
  - Real-time recognition (fast and simple)
  - Facial reconstruction

Figure 5: Reconstruction example

Other methods

Canonical Correlation Analysis (CCA)

- Linear method
- Similar to PCA
- Used when we have multiple views of the same data.
- While PCA assumes dimensions are of one type, in this approach we can reduce dimesnions of different types jointly.
- For example, if we have pixels and text in an image
  - With PCA, we might need to seperate the data out and treat independently.
  - With CCA, we can reduce the two views jointly.

Multi-Dimensional Scaling (MDS)

- Linear method
- While PCA looks at preserving variance, MDS tries to preserve interpoint distances.
- Formally, if d_{ij} is the distance in $\mathbb{R}^k$ and δ_{ij} is the distance in $\mathbb{R}^m$, the goal is to minimize

$$\sum_{i,j}(\frac{d_{ij}-\delta_{ij}}{\delta_{ij}})^2$$

- Use gradient descent or other methods to minimize.

Kernel Methods

- Non linear method
- PCA is ineffective when projection space is non-linear
- Kernel PCA applies kernels (inner products) to allow for differently shaped subspaces.

Figure 6: example

- Some non-linear mmethods
  - Fisher discriminant analysis
  - locally linear embedding
  - Laplacian eigenmaps
  - self-organizing maps
  - manifold alignment
  - curvilinear component analysis, etc.

Isometric Feature Mapping (ISO Maps)

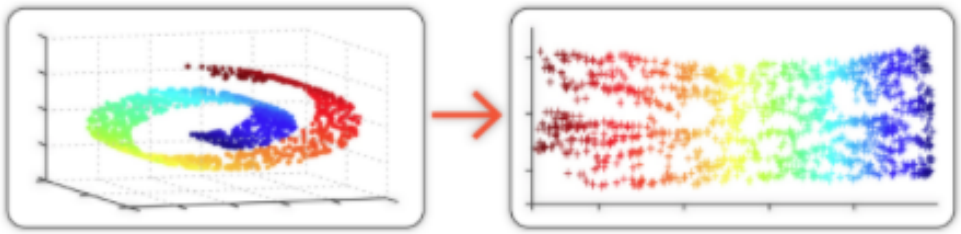

Figure 7: example

- ISO Maps is similar to MDS.
- While MDS uses euclidean distance, ISO maps use "geodesic distance"
- Geodesic distance is the shortest path between two vertices in a graph.
- The approach is as follows:
  - Construct a neighborhood graph,G for the data points.
  - Add edges only if Euclidean distance is less that a vvery small value (ϵ)
  - Compute distance matrix using Geodesic distance in G (Using BFS or Dijkstra's etc.)
  - Use MDS of G