Lecture 41 - Manifold Learning: MDS

Review of PCA

PCA is an **unsupervised** model that can be used to:

1. Perform Dimensionality Reduction, by projecting data into directions with maximum explained variance
2. Uncorrelate Data, by rotating the data space such that data becomes uncorrelated

PCA uses **linear transformations** (projections or rotations) of the input data X:

$$Y = AX$$

where A is a $D \times D$ linear transformation matrix, X is an $D \times N$ data matrix, and Y is a $D \times N$ transformed data matrix.

Therefore PCA will work well when the relationship between features are linear. Moreover, PCA is unsupervised because it **does not** use the class labels to find vector projections (or rotations). So the projections may not necessarily be in the direction that maximize class separability.

PCA can be formulated from two points-of-view:

1. Maximum explained variance
2. Minimum reconstruction error

Steps of PCA

Consider the data X:

1. Subtract the mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

2. Compute the covariance matrix R_X (by definition, the covariance already subtracts the data's mean)

3. Compute eigenvectors and eigenvalues of the matrix R_X, and store the sorted eigenvectors (e_i) in decreasing eigenvalue (λ_i) order.

4. The linear transformation A will be: $A = \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix}$, where $\lambda_1 > \lambda_2$

5. Projection: $\mathbf{Y} = A\mathbf{X}$

Note that the formal definition of covariance already accounts for demeaning the data.

Example

Consider the scaled data matrix X of size $D \times N$ (where N is the number of samples and D the number of features), with covariance matrix K of size $D \times D$. The eigenvalues of K are $\lambda_1 = 0.99$, $\lambda_2 = 0.5$ and $\lambda_3 = 2$, and the respective eigenvectors are $v_1 = \left[-0.99, 0.09, 0\right]^T$, $v_2 = \left[0, 0, 1\right]^T$ and $v_3 = \left[0.09, 0.99, 0\right]^T$. Answer the following questions:

1. Suppose Y is the data transformation obtained by principal component transform of X into a 2-dimensional space. Using the information provided, write down the linear transformation matrix A, and write down the formula for computing Y from X and A. Justify your answer.
2. What is the amount of explained variance of the 2-D projection? Justify your answer.
3. What is the covariance matrix of Y? Justify your answer.

```
In [1]: (2+0.99)/(2+0.99+0.5)
```

```
Out[1]: 0.8567335243553009
```

Manifold Learning

As we have already noted, many natural sources of data correspond to low-dimensional, possibly noisy, non-linear manifolds embedded within the higher dimensional observed data space. Capturing this property explicitly can lead to improved density modelling compared with more general methods.

PCA and LDA are often used to project a data set onto a lower-dimensional space. However both of then assume that the data samples *live* in an underlying linear manifold.

There are other dimensionality reduction techniques that do not assume the manifold is linear. They include:

1. **Multi-Dimensional Scaling (MDS)**

2. **Isometric Mapping (ISOMAP)**

3. **Locally Linear Embedding (LLE)**

MultiDimensional Scaling (MDS)

Another linear technique with a similar aim is **multidimensional scaling**, or **MDS**. It finds a low-dimensional projection of the data such as to preserve, as closely as possible, the pairwise distances between data points, and involves finding the eigenvectors of the distance matrix. In the case where the distances are Euclidean, it gives equivalent results to PCA. Therefore, MDS is a generalization of PCA.

Consider a set of mean-centered observations $X = \{x_1, x_2, \ldots, x_N\}$ where $x_i \in \mathbb{R}^D$. By mean-centered samples X, I mean that $\mu_j = \sum_{i=1}^N x_{ij} = 0, \forall j = 1, 2, \ldots, D$.

Consider the **proximity matrix** D that stores pairwise distances of data points $d_{ij} = \text{distance}(x_i, x_j)$:

$$D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1N} \\ d_{21} & d_{22} & \cdots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \cdots & d_{NN} \end{bmatrix}$$

Note that D is an $N \times N$ symmetric matrix.

A proximity matrix is:

- A *metric* if
    1. $d_{ii} = 0$
    2. $d_{ij} \geq 0, i \neq j$
    3. $d_{ij} = d_{ji}, \forall i, j$
    4. $d_{ij} \leq d_{ik} + d_{jk}, \forall i, j, k$ (triangle inequality).
- *Euclidean* if there exists a configuration of points in Euclidean space with the same d_{ij} values.

Given an assumed Euclidean proximity matrix, D, the **goal** of MDS is to find a set of points, Y, that have the same proximity matrix in an M-dimensional space, where $M < D$.

Let:

$$B = YY^T = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \begin{bmatrix} y_1^T & y_2^T & \cdots & y_N^T \end{bmatrix} = \begin{bmatrix} y_1 y_1^T & y_1 y_2^T & \cdots & y_1 y_N^T \\ y_2 y_1^T & y_2 y_2^T & \cdots & y_2 y_N^T \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1^T & y_N y_2^T & \cdots & y_N y_N^T \end{bmatrix}$$

Then

$$b_{ij} = \sum_{k=1}^M y_{ik} y_{jk}$$

- So, if we want to find B, then we can determine Y by taking:

$$Y \approx B^{1/2}$$

since $B = YY^T$.

- The squared Euclidean distance between points of M-dimensional data Y can be written in terms of B:

$$\begin{aligned} d_{ij}^2 &= (y_i - y_j)(y_i - y_j)^T \\ &= (y_i - y_j)(y_i^T - y_j^T) \\ &= y_i y_i^T - y_i y_j^T - y_j y_i^T + y_j y_j^T \\ &= y_i y_i^T - 2 y_i y_j^T + y_j y_j^T \\ &= b_{ii} - 2b_{ij} + b_{jj} \end{aligned}$$

- Note that if we *translate* or *rotate* the data, we get the same proximity matrix!
- Let's add some constraints to our transformed data: constraint data Y to have mean zero for all dimensions, which is to say, $\sum_{i=1}^N y_{ik} = 0, \forall k$.
- This implies that each row and column of B sum to 0. Proof:

$$\sum_{j=1}^N b_{ij} = \sum_{j=1}^N \sum_{k=1}^M y_{ik} y_{jk} = \sum_{k=1}^M y_{ik} \left(\sum_{j=1}^N y_{jk} \right) = 0 \text{ (sum of columns)}$$

$$\sum_{i=1}^N b_{ij} = \sum_{i=1}^N \sum_{k=1}^M y_{jk} y_{ik} = \sum_{k=1}^M y_{ik} \left(\sum_{j=1}^N y_{jk} \right) = 0 \text{ (sum of rows)}$$

Given this, we have:

Summing over the rows:

$$\sum_{i=1}^N d_{ij}^2 = \sum_{i=1}^N (b_{ii} + b_{jj} - 2b_{ij}) = \sum_{i=1}^N b_{ii} + \sum_{i=1}^N b_{jj} - \sum_{i=1}^N 2b_{ij} = T + N b_{jj}$$
$$\iff b_{jj} = \frac{1}{N} \left(\sum_{i=1}^N d_{ij}^2 - T \right) = \frac{1}{N} \left(\sum_{i=1}^N d_{ij}^2 - \frac{1}{2N} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 \right)$$

Summing over the columns:

$$\sum_{j=1}^N d_{ij}^2 = \sum_{j=1}^N (b_{ii} + b_{jj} - 2b_{ij}) = \sum_{j=1}^N b_{ii} + \sum_{j=1}^N b_{jj} - \sum_{j=1}^N 2b_{ij} = T + N b_{ii}$$
$$\iff b_{ii} = \frac{1}{N} \left(\sum_{j=1}^N d_{ij}^2 - T \right) = \frac{1}{N} \left(\sum_{j=1}^N d_{ij}^2 - \frac{1}{2N} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 \right)$$

Summing over the rows and columns:

$$\sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 = \sum_{i=1}^N \sum_{j=1}^N (b_{ii} + b_{jj} - 2b_{ij}) = 2NT$$
$$\iff T = \frac{1}{2N} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2$$

where $T = \text{trace}(B) = \sum_{i=1}^N b_{ii}$.

- Coming back to proximity matrix, D:

$$d_{ij}^2 = b_{ii} - 2b_{ij} + b_{jj}$$
$$d_{ij}^2 = \frac{1}{N}\left(\sum_{j=1}^{N} d_{ij}^2 - \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2\right) - 2b_{ij} + \frac{1}{N}\left(\sum_{i=1}^{N} d_{ij}^2 - \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2\right)$$
$$2b_{ij} = -d_{ij}^2 + \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 + \frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 - \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2$$
$$b_{ij} = -\frac{1}{2}\left(d_{ij}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2\right)$$

- So, now, we can estimate B using the proximity matrix D. In a matrix form, we can write:

$$B = -\frac{1}{2}JD^2J$$

where $J = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, I is an $N \times N$ identity matrix, $\mathbf{1}$ is an $N \times 1$ vector of 1's and $D^2 = \left[d_{ij}^2\right]$ is the proximity matrix of size $N \times N$.

- Recall that for any real symmetruc matrix B, the eigenvalues are real and the eigenvectors can be chosen such that they are orthogonal to each other. Thus a real symmetric matrix B can be described in the eigenspace, as follows:

$$B = V\Lambda V^T$$

where V is an orthogonal matrix containing the eigenvectors of B and Λ is a diagonal matrix containing the eigenvalues of B.

Then we can estimate Y as follows:

$$Y = B^{1/2} = V\Lambda^{1/2}$$

keeping only the M dimensions of interest ($M < D$) corresponding to the M largest eigenvalues.

- Note: If we use the Euclidean distance to compute D then MDS is equivalent to PCA! However, D can be computed with any metric and therefore, MDS is a generalization of PCA.

Steps to Implement MDS

The MDS algorithm:

1. Compute the distance/proximity matrix, D.
2. Compute D^2.
3. Compute $J = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$
4. Compute $B = -\frac{1}{2}JD^2J$
5. Compute eigenvectors, V, and eigenvalues,Λ, of B. (store them in a matrix in decreasing eigenvalue order.)
6. Compute $Y = V\Lambda^{1/2}$