Large Scale Manifold Learning
An initial exploration as a part of M.Tech thesis
DEBOJYOTI DEY
Advisor: Prof. Amitabha Mukerjee
IIT Kanpur

Abstract
In Non-Linear Dimensionality Reduction (NLDR) algorithms, one is given a collection of \( N \) objects, each a vector in \( \mathbb{R}^D \), and the objective is to find an embedding in a low-dimensional Euclidean space \( \mathbb{R}^d \) while preserving the geometry as faithfully as possible. In traditional NLDR methods like classical MDS, space complexity turns out to be \( O(N^2) \) which for large \( N \) becomes unaffordable with reasonable memory. Here we wish to maintain the space complexity of the problem \( O(N) \).

Introduction
The classical techniques for dimensionality reduction PCA and MDS are effectively implemented for low-dimensional embedding of linear data points. Classical MDS finds the embedding that preserves the internal geometry i.e. interpoint distances are preserved. Often such data points lie on a non linear manifold in high dimensional input space. Each point on the manifold represents one data point. The actual distance between two points spaced apart on the manifold can not be measured along a straight line. Thus PCA and traditional MDS fails as they see just the Euclidean structure. So we use a different measure called Geodesic distance. Here we keep distances among the nearby points and discard the others. To compute the distance between two far away points we sum the small distances along the path from one point to another. We need to compute the distance matrix for the data points \( N \) that we have which requires \( O(N^2) \) space. For large dataset this is way too expensive with respect to space.

Objective
Given \( N \) input points \( X = \{x_i\}_{i=1}^N \) and \( x_i \in \mathbb{R}^D \) the goal is to find corresponding embedding \( Y = \{y_i\}_{i=1}^N \) and \( y_i \in \mathbb{R}^d, d \ll D \) such that \( Y \) faithfully preserves the geometry i.e \((1 - \epsilon)||x_i - x_j|| \leq ||y_i - y_j|| \leq (1 + \epsilon)||x_i - x_j|| \) \( \forall i, j \in [N] \) and \( \epsilon > 0 \) is a very small constant.
Proposition

1. Instead of computing the inner product matrix of the whole data-set taking $O(N^2)$ space, use a very small subset of $l \geq d + 1$ landmark points and compute the inner product matrix on that subset which essentially takes $O(l^2)$ space.

2. Perform classical MDS on the $l \times l$ inner product matrix to get $d$ dimensional embedding of $l$ landmarks. Say the matrix is $L$.

3. Use the $l \times N$ distance matrix (taken as input) with $L$ to compute the embedding of all $N$ points.

Inspiration: Matrix decomposition

Taking a subset of $l$ data points ($l \ll N$) and using $l \times N$ matrix to approximate the spectral decomposition of the original $N \times N$ symmetric matrix. We can rearrange the similarity matrix:

$$G = \begin{pmatrix} W & G_{21}^T \\ G_{21} & G_{22} \end{pmatrix}$$

where $W = L \times L$ matrix and is PSD and $G_{21} = (N - L) \times L$ matrix.

$$C = \begin{pmatrix} W \\ G_{21} \end{pmatrix}$$

a $N \times L$ matrix. Using $W$ and $C$ we can approximate $G$ as:

$$G = \begin{pmatrix} W & G_{21}^T \\ G_{21} & G_{21}W^{-1}G_{21}^T \end{pmatrix}$$

We follow Nyström approach which implements the above idea.

Summary of Methods: Landmark MDS

The Landmark MDS algorithm was introduced by V. de Silva & J. B. Tenenbaum in context of finding an efficient approximation to the Isomap algorithm for nonlinear dimensionality reduction [2] suitable for processing large data sets. The algorithm steps are the following:

Preprocessing:
Find K nearest neighbours for all the N datapoints and compute $N \times N$
sparse distance matrix by only retaining the distances between neighbours. 

**Main approach:**

1. We designate \( l \geq k + 1 \) landmark points randomly for \( k \) dimensional embedding. The random selection works as good as greedy Maxmin approach and with less cost. Restrict the distance matrix to landmark points. Let’s call is \( \mathbf{D} \).

2. Perform classical MDS on landmarks. We derive the inner product matrix \( B \) from \( \mathbf{D} \) by,

\[
B_{ij} = -\frac{1}{2}(D_{ij} - \frac{1}{l} \sum_{j=1}^{l} D_{ij} - \frac{1}{l} \sum_{i=1}^{l} D_{ij} + \frac{1}{l^2} \sum_{i,j=1}^{l} D_{ij})
\]

We compute \( d \) largest positive eigenvalues \( \lambda_i \) together with their orthonormal set of eigenvectors \( \vec{v}_i \). Just to mention, the nonpositive eigenvalues includes a zero corresponding to eigenvector \( \vec{1} = [1, 1, \ldots, 1]^T \) because \( B \) being inner product matrix of mean centered data points its rowsum is a zero vector. The required \( d \) dimensional embedding vectors \( l_1, l_2, \ldots, l_L \) are given by the columns of the following matrix

\[
L = \begin{pmatrix}
\sqrt{\lambda_1}.\vec{v}_1^T \\
\sqrt{\lambda_2}.\vec{v}_2^T \\
\vdots \\
\sqrt{\lambda_d}.\vec{v}_d^T 
\end{pmatrix}
\]

This embedding is automatically mean centered. We can validate it from the fact that \( \vec{v}_i^T \vec{1} = 0 \) for all \( i \).

3. Distance based Triangulation: Embedding coordinates of remaining data points are calculated based on their distances from the landmark points. We describe the method below.

- \( \vec{\delta}_a = (\vec{\delta}_1 + \vec{\delta}_2 + \ldots + \vec{\delta}_l) / l \), where \( \vec{\delta}_i \) denotes the \( i^{th} \) column in the distance matrix \( \mathbf{D} \).

- Compute pseudoinverse transpose of \( L \).

\[
\tilde{L} = \begin{pmatrix}
\vec{v}_1^T / \sqrt{\lambda_1} \\
\vec{v}_2^T / \sqrt{\lambda_2} \\
\vdots \\
\vec{v}_d^T / \sqrt{\lambda_d}
\end{pmatrix}
\]

Each data point \( a \) is embedded as follows:

\[
y_a = -\frac{1}{2} \tilde{L}(\vec{\delta}_a - \vec{\delta}_\mu) \tag{1}
\]

\( \vec{\delta}_a \) is the vector of squared distances between point \( a \) and \( l \) landmark points.
Improved Space and Time bound in L-Isomap

- L-Isomap requires $O(lN)$ space for storing the $l \times N$ distance matrix, as against $O(N^2)$ in classical MDS of traditional Isomap.

- Using Dijkstra to compute all pair shortest path distance takes $O(tNl \log N)$ time where $t$ is the no of neighbours (degree) of a vertex, as against $O(tN^2 \log_2 N)$ in traditional Isomap.

- Finding eigenvalue and eigenvectors of $l \times l$ matrix takes $O(l^3)$ time by typical iterative algorithms where it takes $O(N^3)$ time for traditional Isomap.

Results

We measure the performance of the algorithm in terms of Residual Variance which is actually the deviation of the Euclidean distance among the $d$-dimensional the vectors $y_i$’s from its counterpart (geodesic distance) among the original $D$-dimensional vectors $x_i$’s. We plot 5000 2D vectors below for 2 DOF arm and corresponding residual variance graph. We expect the embedding to be a toroidal projection.

For robotic arm with 4 degrees (say $\theta_1, \theta_2, \theta_3, \theta_4$) of freedom we obtain the 4-D embedding vectors. To get the visual interpretation, we fix $\theta_2$ and $\theta_4$ to some 40 degree range and vary the $\theta_1$ and $\theta_3$ expecting a toroid and lowest residual variance near 4. But it Fails with 50000 data points!
Figure 2: Low dimensional embedding for 2 DOF robotic arm and 5000 data points (left) and residual variance (experimented with 100 landmark points)

Figure 3: Traditional Isomap on robotic arm data with 2 degrees of freedom

Figure 4: Low dimensional embedding for 4 DOF robotic arm and 50000 data points (left) and residual variance (experimented with 500 landmark points)
Conclusions

- From the result it is clear that even 50000 is very sparse for 4 DOF data and hence the approximate distance between two points on the manifold differs much with the actual. As a result we don’t achieve minimum residual variance at 4.

- When we fix two $\theta_i$’s the number of datapoints obtained is very less(<1000). So we need about $10^6$ data points for best result.

Further Works

Learning the joint angles $\theta_i$’s from the embedded vectors $y_j$ using regression. We shall model a neural network with inputs as the components of $y_j$ and outputs as Sine and Cosine values of $\theta_i$’s with the hope to obtain a function which maps $Y$ to $\theta$ space. Finally we want to model the configuration space where given a source and a destination point in the space having several obstacles the robotic arm needs to reach from source to destination following the shortest path without colliding with any obstacle. We can extend this work to moving obstacles.

References


