Large Scale Manifold Learning
M.Tech Thesis Presentation

Debojyoti Dey

Supervisor-Prof. Amitabha Mukerjee

February 7, 2014
Outline

1 Motive
   - Objective
   - Previous works and Citation

2 Traditional Approaches
   - Isomap Technique
   - Challenges for Traditional approaches
   - Proposed Solution

3 New Approach
   - Methods
   - Nyström method
   - Correctness Proof of Triangulation Method
   - Improved Space Time bound
   - Some results

4 Parallel computation to handle large data
   - MPI and MapReduce
Given $n$ input points $X = \{x_i\}_{i=1}^n$ and $x_i \in \mathbb{R}^d$ the goal is to find corresponding outputs $Y = \{y_i\}_{i=1}^n$ and $x_i \in \mathbb{R}^k$, $k \ll d$ such that $Y$ faithfully represents $X$. 


Steps:

- Build ‘neighbourhood(t-nearest for each point) matrix’ for $N$ points.
- Compute $N \times N$ ‘Geodesic Distance Matrix’ using the neighbourhood matrix computed. Algorithm- Dijkstra.
- Construct $N \times N$ similarity matrix $S$ by centering Geodesic dist matrix
- Find the max $k$ eigenvalues and corresponding eigenvectors of $S$.
- $U$ be the orthonormal eigenvector matrix. $\Sigma$ be diagonal eigenvalue matrix. $Y$ are obtained by taking projection of the data points on the eigenvectors, $Y = \Sigma^{1/2} U^T$
Steps:

- Build ‘neighbourhood(t-nearest for each point) matrix’ for $N$ points.
- Compute $N \times N$ ‘Geodesic Distance Matrix’ using the neighbourhood matrix computed. Algorithm- Dijkstra.
- Construct $N \times N$ similarity matrix $S$ by centering Geodesic dist matrix.
- Find the max $k$ eigenvalues and corresponding eigenvectors of $S$.
- $U$ be the orthonormal eigenvector matrix. $\Sigma$ be diagonal eigenvalue matrix. $Y$ are obtained by taking projection of the data points on the eigenvectors, $Y = \Sigma^{1/2}U^T$
Steps:

- Build ‘neighbourhood(t-nearest for each point) matrix’ for $N$ points.
- Compute $N \times N$ ‘Geodesic Distance Matrix’ using the neighbourhood matrix computed. Algorithm- Dijkstra.
- Construct $N \times N$ similarity matrix $S$ by centering Geodesic dist matrix
  - Find the max $k$ eigenvalues and corresponding eigenvectors of $S$.
  - $U$ be the orthonormal eigenvector matrix. $\Sigma$ be diagonal eigenvalue matrix. $Y$ are obtained by taking projection of the data points on the eigenvectors, $Y = \Sigma^{1/2} U^T$
Steps:

- Build ‘neighbourhood(t-nearest for each point) matrix’ for \( N \) points.
- Compute \( N \times N \) ‘Geodesic Distance Matrix’ using the neighbourhood matrix computed. Algorithm- Dijkstra.
- Construct \( N \times N \) similarity matrix \( S \) by centering Geodesic dist matrix
- Find the max \( k \) eigenvalues and corresponding eigenvectors of \( S \).
- \( U \) be the orthonormal eigenvector matrix. \( \Sigma \) be diagonal eigenvalue matrix. \( Y \) are obtained by taking projection of the data points on the eigenvectors, \( Y = \Sigma^{1/2} U^T \)
Steps:

- Build ‘neighbourhood(t-nearest for each point) matrix’ for $N$ points.
- Compute $N \times N$ ‘Geodesic Distance Matrix’ using the neighbourhood matrix computed. Algorithm- Dijkstra.
- Construct $N \times N$ similarity matrix $S$ by centering Geodesic dist matrix
- Find the max $k$ eigenvalues and corresponding eigenvectors of $S$.
- $U$ be the orthonormal eigenvector matrix. $\Sigma$ be diagonal eigenvalue matrix. $Y$ are obtained by taking projection of the data points on the eigenvectors, $Y = \Sigma^{1/2} U^T$
Steps:

- Build ‘neighbourhood(t-nearest for each point) matrix’ for \( N \) points.
- Compute \( N \times N \) ‘Geodesic Distance Matrix’ using the neighbourhood matrix computed. Algorithm- Dijkstra.
- Construct \( N \times N \) similarity matrix \( S \) by centering Geodesic dist matrix
- Find the max \( k \) eigenvalues and corresponding eigenvectors of \( S \).
- \( U \) be the orthonormal eigenvector matrix. \( \Sigma \) be diagonal eigenvalue matrix. \( Y \) are obtained by taking projection of the data points on the eigenvectors, \( Y = \Sigma^{1/2} U^T \)
Challenges:

- Classical MDS requires $O(N^2)$ space for storing the $N \times N$ distance matrix. For large $N$ this matrix cannot be stored in memory.
- Using Dijkstra to compute all pair shortest path distance takes $O(tN^2 \log_2 N)$ time where $t$ is the no of neighbours(degree) of a vertex.
- Finding eigenvalue and eigenvectors of $N \times N$ matrix takes $O(N^3)$ time by typical iterative algorithms.
Main Idea: Storing a subset of $L$ data points ($L \ll N$) and using $L \times L$ similarity matrix to approximate the spectral decomposition of the original $N \times N$ symmetric matrix. We can rearrange the similarity matrix $G$ as following:

\[
\begin{pmatrix}
W & G_{21}^T \\
G_{21} & G_{22}
\end{pmatrix}
\]

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

\[
\begin{pmatrix}
W \\
G_{21}
\end{pmatrix}
\]

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

\[
\begin{pmatrix}
W & G_{21}^T \\
G_{21} & G_{21}W^{-1}G_{21}^T
\end{pmatrix}
\]
Two methods using matrix decomposition by taking subset of data points:

- Nyström method
- Column sampling approximation

We concentrate upon Nyström method in our next slides
Summary of steps:

1. **Selecting the Landmark points:** We designate $L \geq k + 1$ landmark points randomly for $k$ dimensional embedding. Works as good as greedy Maxmin approach and with less cost.

2. **Classical MDS on Landmark:** Produces $k \times L$ matrix $Y$ representing an embedding of $L$ landmark points in $\mathbb{R}^k$.

3. **Distance based Triangulation:** Used to find $k \times N$ matrix representing an embedding of $N$ data points in $\mathbb{R}^k$ by extending $Y$. 
Diving deep into MDS

From the Distance(Geodesic) square matrix $D$ we construct inner-product $B$ matrix by mean centering by the below formula:

$$B_{ij} = -\frac{1}{2}(D_{ij} - \frac{1}{L} \sum_{j=1}^{L} D_{ij} - \frac{1}{L} \sum_{i=1}^{L} D_{ij} + \frac{1}{L^2} \sum_{i,j=1}^{L} D_{ij})$$

We compute $k$ largest positive eigenvalues $\lambda_i$ together with their corresponding 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.
Diving deep into MDS

The required k dimensional embedding vectors $l_1, l_2, \ldots, l_L$ are given by the columns of the following matrix $Y$

$$Y = \begin{pmatrix} \sqrt{\lambda_1} \cdot \vec{v}_1^T \\ \sqrt{\lambda_2} \cdot \vec{v}_2^T \\ \vdots \\ \sqrt{\lambda_k} \cdot \vec{v}_k^T \end{pmatrix}$$

This embedding is automatically mean centered. We can validate it from the fact that $\vec{v}_i^T \mathbf{1} = 0$ for all $i$. 
Embedding coordinates of remaining data points are calculated based on their distances from the landmark points. We describe the method below.

- $\tilde{\delta}_\mu = (\tilde{\delta}_1 + \tilde{\delta}_1 + \ldots + \tilde{\delta}_L)/L$, where $\tilde{\delta}_i$ denotes the $i^{th}$ column in the distance matrix $D$.

- Compute pseudoinverse transpose $\tilde{Y}$ of $Y$.

$$\tilde{Y} = \begin{pmatrix} \vec{v}_1^T / \sqrt{\lambda_1} \\ \vec{v}_2^T / \sqrt{\lambda_2} \\ \vdots \\ \vec{v}_k^T / \sqrt{\lambda_k} \end{pmatrix}$$

Each data point $a$ is embedded as follows:
Diving deep into Distance based triangulation II

**INPUT:** the vector $\delta_a$ of squared distances between point $a$ and $L$ landmark points.

**OUTPUT:**

$$\tilde{x}_a = -\frac{1}{2} \tilde{Y} (\delta_a - \delta_\mu)$$  \hfill (1)
Suppose $\vec{y}_a \in \mathbb{R}^d$ is an additional data point and $\vec{\delta}_a$ is its squared-distance vector to the landmark points. Then the vector $\vec{x}_a$ defined by the formula (1) gives the components of $\vec{y}_a$ when projected onto the first $k$ principal axes of the landmark points.
Proof.

We have \( Y = [y_1 y_2 \ldots y_L] \). The principal axes are given by the eigenvectors of \( Y^T Y \). Let \( \vec{p}_i \) be the unit eigenvector for \( i^{th} \) positive eigenvalue \( \lambda_i' \). So we have the relation,

\[
\sqrt{\lambda_i} \vec{v}_i^T = \vec{p}_i^T Y
\]

Then,

\[
\lambda_i = \sqrt{\lambda_i} \vec{v}_i^T \vec{v}_i \sqrt{\lambda_i} = \vec{p}_i^T YY^T \vec{p}_i = \lambda'_i \vec{p}_i^T \vec{p}_i = \lambda_i' \tag{2}
\]

We must show that \( i^{th} \) component of \(-\frac{1}{2} \tilde{Y} (\vec{\delta}_a - \vec{\delta}_\mu)\) is equal to projection of \( \vec{y}_a \) onto \( i^{th} \) principal axis. That is,

\[
-\frac{1}{2} \frac{\vec{v}_i^T}{\sqrt{\lambda_i}} (\vec{\delta}_a - \vec{\delta}_\mu) = \vec{p}_i^T \vec{y}_a \tag{3}
\]
Proof contd.

First we compute the $j^{th}$ entries of $\delta_a$ and $\delta_\mu$:

$$[\delta_a]_j = ||\vec{y}_a||^2 - 2\vec{y}_j^T \vec{y}_a + ||\vec{y}_j||^2$$

$$[\delta_\mu]_j = \frac{1}{L} \sum_{k=1}^L ||\vec{y}_k||^2 - 2\vec{y}_j^T \frac{1}{L} \sum_{k=1}^L ||\vec{y}_k|| + ||\vec{y}_j||^2$$

The left hand terms are independent of $j$ and the middle term of the second equation is 0 since $\vec{y}_k$ are mean centered. Thus we subtract to derive the vector equation:

$$\delta_a - \delta_\mu = c \vec{1} - 2 Y^T \vec{y}_a$$

where $c = ||\vec{y}_a||^2 - \frac{1}{L} \sum_{k=1}^L ||\vec{y}_k||^2$ is a scaler.
Substituting in (3) we can drop the term $c\vec{1}$ as $\vec{v}_i^T \vec{1} = 0$. Using (2) and the last observation, LHS of equation (3) becomes equivalent to

$$\frac{\vec{p}_i^T YY^T \vec{y}_a}{\lambda_i} = \vec{p}_i^T \vec{y}_a$$
Time and Space complexity:

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

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

- Finding eigenvalue and eigenvectors of $L \times L$ matrix takes $O(L^3)$ time by typical iterative algorithms. ($O(N^3)$ in traditional Isomap)
Figure: Traditional Isomap on robotic arm data with 2 degrees of freedom
**Motive**

**Traditional Approaches**

**New Approach**

Parallel computation to handle large data

**Methods**
- Nyström method
- Correctness Proof of Triangulation Method
- Improved Space Time bound

**Some results**

**Figure:** *Landmark Isomap with 50 Landmarks (above) and 10 Landmarks*
Motive
Traditional Approaches
New Approach
Parallel computation to handle large data

Feasibility question of performing parallel computation:
MPI is a message passing library interface specification for parallel programming. In a distributed system framework, processors of different machines work individually on a subset of data while communicating and synchronising with others. Large data can thus be efficiently handled with nominal resources of individual machines.
MapReduce, Google parallel computing framework has been used to compute distance matrix and nearest neighbours for large data. 'Map' distributes the data instances to several computation nodes according to item/value pair and 'Reduce' performs local computation on each node resulting in the final result.