Machine Learning for Data Science (CS 4786)

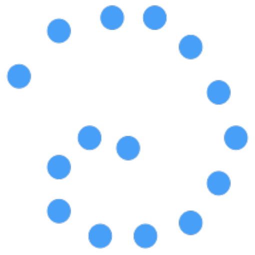
Lecture 9: ISOMAP and TSNE

The text in black outlines high level ideas. The text in blue provides simple mathematical details to "derive" or get to the algorithm or method. The text in red are mathematical details for those who are interested.

1 Motivation: Points on a Low Dimensional Manifold

Linear dimensionality reduction methods can only go so far! Often times, data might look like it lives in a large dimensional linear space, however, it might still be truly low dimensional, only that the low dimensional points might be folded in to higher dimensions. As a visual example, think of the case where we have points on a sheet of paper (so they are truly 2 dimensional by construction). Now say we fold/bend this sheet so the points now cannot be seen as living in a 2 dimensional plane in the 3d world. We still know the points are in 2D, only that no linear method can identify the 2 dimensional representation. The question we are interested in is of course how do we recover the 2 dimensional representation.

So I can give you a quick caricature, consider the following set of points shown in 2D.



Can we unfold this?

Looking at these set of points, there is a pattern, they all lie on a curved line. So you could think of them as being 1 dimensional points rolled in to look like they are in 2 dimensional. We would like to unfold these points and recover their one dimensional representation as:

How do we do this, how do we formalize this problem?

A nice mathematical way of thinking about these problems is via the idea of a low dimensional "manifold". A manifold is a space that looks locally Euclidean. Meaning, in a small neighborhood, the space looks flat (and low dimensional). The best picture to have in mind and perhaps the most famous example that lead to lots of confusion in human history ;): if we think of the surface of earth, locally, it looks like the earth is flat while now we know that we live on a round planet. This is because the surface of the earth is a 2 dimensional manifold that lives in 3 dimensions.

2 ISOMAP

Now back to the question of how to unfold the set of points in Figure 1 to get the points in Figure 2. ISOMAP aims to achieve this by first finding (approximately) the distances between all pairs of points on the low dimensional manifold, and then finding a set of $\mathbf{y}_t$'s such that distances between pairs of points in the low dimensional space is roughly the same as the distances between the pairs of points on the manifold. Let me first mention the steps pf the algorithm fully and then we can try to understand how or why it works.

1. For every point, find its k-Nearest Neighbor (w.r.t. the original distance in the high dimensional space) amongst the set of data points provided.
2. Form the k-nearest neighbor graph $G = (V, E)$. Every point is a vertex in the graph (n-vertices in total). Every point is connected by an edge to its k-nearest neighbors.
3. Computer pairwise distances between all pairs of points in the graph using the shortest distance on the graph as the metric. Represent this by $n \times x$ matrix A
4. Find points in the low dimensional space so that pairwise distances between points is roughly the same as distances between the points on the graph.

The output of the algorithm is the low dimensional points computed in step 4.

The idea is the following, if we have enough points on the manifold, that is they are packed closely, then since a manifold is locally Euclidean, the k-nearest neighbors in the original high-dimensional space are also the k-nearest neighbors on the manifold. Next we form the nearest neighbor graph and the idea is that (shortest) distances on the graph correspond to distances on the manifold. Think about people in the Americas as being points in 3 dimension. If we form nearest neighbor graph by taking k-closest people to every person on in the Americas, that forms our nearest neighbor graph. Now consider the shortest distance on the graph between a person in the south eastern tip of Argentina and the north western tip of Alaska. The distance on the graph will roughly correspond to the distance we need to walk which is exactly the distance on the manifold. However the euclidean distance between the two points will be the distance we would travel if we drilled a hole between the two locations. How do we compute all pairs shortest path between points on a graph? Well we use FloydWarshall algorithm (or if you prefer Dikstra's algorithm between all pairs).

Now we have the manifold distances between all pairs of points (which we shall represent). Now we simply need to find points on low dimensional space such that Euclidean distances in the low dimensional space is roughly the manifold distances between these points. Given all pari-wise distances between n points, the question of how do we find low dimensional points whose distances are roughly the same as the given n^2 distances is exactly the problem solved by multidimensional scaling.

2.1 Multidimensional Scaling

The problem dealt with in multidimensional scaling is the following: Given, an $n \times n$ matrix M of all pairwise distances between n points (i.e. $M_{i,j}$ is the distance between i'th and the j'th point). Find points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ such that $\|\mathbf{y}_i - \mathbf{y}_j\|_2 \approx M_{i,j}$.

To solve this problem, we use linear algebra. First let us define an $n \times n$ matrix D such that for any i, j, $D_{i,j} = M_{i,j}^2$. That is entry-wise square of matrix M. Now say there existed $\mathbf{y}'_1, \ldots, \mathbf{y}'_n$ such that their pairwise distances were equal to the distances indicated in matrix M. Then, we would have that for any i, j,

$$D_{i,j} = \|\mathbf{y}'_i - \mathbf{y}'_j\|_2^2 = \|\mathbf{y}'_i\|_2^2 + \|\mathbf{y}'_j\|_2^2 - 2{\mathbf{y}'_i}^\top \mathbf{y}'_j$$

Or in other words,

$${\mathbf{y}'_i}^\top \mathbf{y}'_j = \frac{1}{2}\left(\|\mathbf{y}'_i\|_2^2 + \|\mathbf{y}'_j\|_2^2 - D_{i,j}\right) \tag{1}$$

Now note that if we center n points, that does not change their inter point distances. Hence without loss of generality let us assume that $\sum_i \mathbf{y}'_i = 0$. Using this, note that:

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^n D_{i,j} &= \frac{1}{n}\sum_{i=1}^n \|\mathbf{y}'_i\|_2^2 + \|\mathbf{y}'_j\|_2^2 - \frac{2}{n}\sum_{i=1}^n {\mathbf{y}'_i}^\top \mathbf{y}'_j \\
&= \frac{1}{n}\sum_{i=1}^n \|\mathbf{y}'_i\|_2^2 + \|\mathbf{y}'_j\|_2^2 - \frac{2}{n}\left(\sum_{i=1}^n \mathbf{y}'_i\right)^\top \mathbf{y}'_j \\
&= \frac{1}{n}\sum_{i=1}^n \|\mathbf{y}'_i\|_2^2 + \|\mathbf{y}'_j\|_2^2
\end{aligned}$$

Similarly, we have that

$$\frac{1}{n}\sum_{j=1}^n D_{i,j} = \frac{1}{n}\sum_{j=1}^n \|\mathbf{y}'_j\|_2^2 + \|\mathbf{y}'_i\|_2^2$$

Also note that

$$\frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n D_{i,j} = \frac{2}{n}\sum_{i=1}^n \|\mathbf{y}'_i\|_2^2$$

Hence using this in Eq. 1 we conclude that:

$$\begin{aligned}
{\mathbf{y}'_i}^\top \mathbf{y}'_j &= \frac{1}{2}\left(\|\mathbf{y}'_i\|_2^2 + \|\mathbf{y}'_j\|_2^2 - D_{i,j}\right) \\
&= \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^n D_{i,j} + \frac{1}{n}\sum_{j=1}^n D_{i,j} - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n D_{i,j} - D_{i,j}\right)
\end{aligned}$$

The above written in matrix equations succinctly can be written as:

$$\begin{aligned}
Y'{Y'}^\top &= \frac{1}{2}\left(\frac{1}{n}\mathbf{1}\mathbf{1}^\top D + \frac{1}{n}D\mathbf{1}\mathbf{1}^\top - \frac{1}{n^2}\mathbf{1}\mathbf{1}^\top D\mathbf{1}\mathbf{1}^\top - D\right) \\
&= -\frac{1}{2}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)D(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)
\end{aligned}$$

where Y' is the matrix with n rows and whose rows are ${\mathbf{y}'_t}^\top$. Thus we have shown that the matrix $(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)D(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$ is a matrix of inner products of centered points (That is the $\tilde{K}$

matrix we talked about in kernel PCA). So now we know exactly how to extract the low dimensional representation of the n points. We simply find the top K eigenvectors of the matrix $(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)D(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$ and scale each one by the squareroot of the corresponding eignevalue. This gives us our final $n \times K$ matrix Y.

An interesting fact: the matrix $(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)D(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$ is positive semidefinite if and only if the distances in matrix M can be represented as Euclidean distances between some n points (in whatever dimensions you want). The top K eignevectors just give us a K dimensional approximation.

Hence the final algorithm is the following:

1. Given matrix M compute matrix D whose entries are the squares of the entries of M
2. Compute $\tilde{K} = -\frac{1}{2}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)D(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$
3. Compute $P_1, \ldots, P_K$, the top K eigen vectors of $\tilde{K}$ with corresponding eigenvalues $\gamma_1, \ldots, \gamma_K$
4. Compute the $n \times K$ matrix Y by setting its i'th column to $\sqrt{\gamma_i}P_i$.

2.2 Pitfalls of ISOMAP

ISOMAP relies on first finding distances of the manifold, this step can easily be faulty if we don't have dense enough points on the manifold. For instance, say instead of every person in the americas being a data point, we only have data from every person carrying an ISOMAP smartphone which say only one in 100,000 people posses. In this case, the distance on the graph could be far from distance on the manifold. Other ways this problem manifests is that if k for k nearest neighbor is too small, graph may not even be connected. If k is large, graph might be too dense giving us wrong distances on manifold. In general finding right k if it even exists is hard. Next we will try to develop a method called stochastic neighborhood embedding that addresses some of these shortcomings.

3 Stochastic Neighborhood Embedding (SNE)

In stochastic neighborhood embedding we don't have a deterministic notion of neighbors or nearest neighbors. We form a so called stochastic neighborhood. For every pair of points $t, s \in [n]$, we assign a probability that they are neighbors so that close by points have higher probability of being neighbors. AS an example, if the original points are $\mathbf{x}_1, \ldots, \mathbf{x}_n$, then we might want to set

$$P_{s,t} = P_{t,s} = \frac{1}{2n}(p_{s\mapsto t} + p_{t\mapsto s})$$

where

$$p_{s\mapsto t} = \frac{\exp\left(-\frac{\|\mathbf{x}_s - \mathbf{x}_t\|_2^2}{2\sigma^2}\right)}{\sum_{u\neq s}\exp\left(-\frac{\|\mathbf{x}_s - \mathbf{x}_u\|_2^2}{2\sigma^2}\right)}$$

In the above, $P_{s,t}$ is the probability that points s and t are neighbors according to stochastic neighborhood distribution P. Our goal in stochastic neighborhood embedding is to find points (in K dimensions) $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and a corresponding stochastic neighborhood distribution Q such that

P and Q are "close". For the purposes of SNE algorithms, we shall use the notion of closeness by the so called Kullback Leibler (KL) divergence between P and Q defines as:

$$\mathrm{KL}(P|Q) = \sum_{s,t} P_{s,t}, \log\left(\frac{P_{s,t}}{Q_{s,t}}\right)$$

With all this defined, we can now state the goal of Stochastic Neighborhood Embedding as follows:

Find points $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^K$ that minimize $\mathrm{KL}(P|Q)$

Now to define stochastic neighborhood Q for the low dimensional representations $\mathbf{y}_1, \ldots, \mathbf{y}_n$ just like for the input points, we can define stochastic neighborhood for the $\mathbf{y}_1, \ldots, \mathbf{y}_n$ as:

$$Q_{s,t} = Q_{t,s} = \frac{1}{2n}(q_{s\mapsto t} + q_{t\mapsto s})$$

where

$$q_{s\mapsto t} = \frac{\exp\left(-\frac{\|\mathbf{y}_s - \mathbf{y}_t\|_2^2}{2\sigma^2}\right)}{\sum_{u\neq s}\exp\left(-\frac{\|\mathbf{x}_s - \mathbf{x}_u\|_2^2}{2\sigma^2}\right)} \tag{2}$$

However as we will discuss next, this definition of stochastic neighborhood will suffer from the so called crowding problem.

3.1 The Crowding Problem

In higher dimensions we have more space! So if we simply embed points to lower dimensions, the points tend to overcrowd. For instance, in 2 dimensions, we can have three points that are equidistant from each other, (and more generally, in d dimensions $d+1$ points). However, if we need to sq uish them in lower dimensions, these points need to be moved closer to each other and we will have to distort distances. This is problematic because in high dimension, most points are outliers. As an example, if we consider points in d dimensions distributed as standard normal, then most points are at a distance of $\sqrt{d}$ from the mean. Hence as dimension is large, we will have many points that in high dimensions that are far apart in high dimension but that all need to be accommodated in lower dimensional space with small distortion. Hence if we use the stochastic neighborhood model given by Equation 3, then we will have to squish all these high dimensional points into a light tailed distribution in low dimensional space which will squish all points into a small area near the center.

3.2 t-SNE

To alleviate the crowding problem, the t-SNE algorithm uses the following idea. For the high dimensional space (i.e. for the stochastic neighborhood P), we use a distribution like the gaussian distribution that has an exponentially decaying tail. However for the low dimensional points though, we use a heavy tailed distribution, specifically the student t-distribution. Specifically, we will use:

$$Q_{s,t} = Q_{t,s} = \frac{1}{2n}(q_{s\mapsto t} + q_{t\mapsto s})$$

where

$$q_{s\mapsto t} = \frac{(1 + \|\mathbf{y}_s - \mathbf{y}_t\|_2^2)^{-1}}{\sum_{u\neq s}\left(1 + \|\mathbf{y}_s - \mathbf{y}_u\|_2^2\right)^{-1}} \tag{3}$$

The idea being that heavy tailed distribution allow more space for points (allows for more outliers in the low dimensional space).

Finally for the above definition of Q, we would like to perform the optimization:

Find points $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^K$ **that minimize** $\mathrm{KL}(P|Q)$

This in t-SNE is performed using gradient descent on $\mathrm{KL}(P|Q)$ w.r.t. $\mathbf{y}_1, \ldots, \mathbf{y}_n$.