Notes on Statistical and Machine Learning

Nonlinear Manifold Learning

Chapter: *29* **Prepared by:** *Chenxi Zhou*

This note is prepared based on

- *Chapter 16, Nonlinear Dimensionality Reduction and Manifold Learning* in Izenman (2009),
- *Chapter 14, Unsupervised Learning* in Hastie, Tibshirani, and Friedman (2009), and
- Visualizing Data Using t-SNE by van der Maaten and Hinton (2008).

I. Introduction

1. **Goal:** The goal of this chapter is to study new algorithms that recover full low-dimensional representation of an unknown nonlinear manifold $\mathcal{M}$ embedded in some high-dimensional space.

   Remark. We hope that the learned low-dimensional representation can retain the neighborhood structure of $\mathcal{M}$.

2. **Space Embedding:** A space $\mathcal{A}$ is said to be *embedded* in a bigger space $\mathcal{B}$ if the properties of $\mathcal{B}$ when restricted to $\mathcal{A}$ are identical to the properties of $\mathcal{A}$.

3. **General Approach:** Algorithms covered in this chapter (except SNE and t-SNE presented in the last section) consist of a three-step approach with the first and the third steps are common to all:

   (a) *Step 1:* Incorporate neighborhood information from each data point to construct a weighted graph with the data points being the vertices;

   (b) *Step 2:* Transform the weighted neighborhood graph into suitable input for the embedding step (Step 3);

   (c) *Step 3:* Solve an $n \times n$ eigen problem.

4. **Manifold:** A *manifold*, also known as a *topological manifold*, is a topological space that *locally* look flat and featureless and behaves like Euclidean space.

5. **Sub-manifold:** A *sub-manifold* is a manifold lying inside a manifold of higher dimension.

6. **Smooth (Differentiable) Manifold:** If a manifold $\mathcal{M}$ is continuously differentiable to any order, we call it *smooth manifold*, also known as *differentiable manifold*.

7. **Riemannian Manifold:** If we endow a smooth manifold $\mathcal{M}$ a metric $d_{\mathcal{M}}$, which calculates the distance between points in $\mathcal{M}$, we obtain a *Riemannian manifold*, denoted by $(\mathcal{M}, d_{\mathcal{M}})$.

   Remark. If $\mathcal{M}$ is connected, it is a metric space and $d_{\mathcal{M}}$ determines its structure.

8. **Distance in Riemannian Manifold:** Let $\mathcal{C}(\mathbf{y}, \mathbf{y}')$ denote the set of all differentiable curves in $\mathcal{M}$ connecting points $\mathbf{y}, \mathbf{y}' \in \mathcal{M}$. Then, the *distance* between $\mathbf{y}$ and $\mathbf{y}'$ is defined as

$$d_{\mathcal{M}}(\mathbf{y}, \mathbf{y}') := \inf_{c \in \mathcal{C}(\mathbf{y}, \mathbf{y}')} L(c), \tag{1}$$

   where $L(c)$ denotes the arc-length of the curve c. In other words, $d_{\mathcal{M}}$ finds the shortest curve (or *geodesic*) between any two points on $\mathcal{M}$, and $d_{\mathcal{M}}(\mathbf{y}, \mathbf{y}')$ is the geodesic distance between the points.

9. **Data on Manifold:**

   (a) *Data on Manifold $\mathcal{M}$:* Suppose we have finitely many data points $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$ that are randomly sampled from a smooth s-dimensional Riemannian manifold $(\mathcal{M}, d_{\mathcal{M}})$;

   (b) *Data on a Higher-dimensional Manifold:* Suppose $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$ are nonlinearly embedded by a smooth map ψ to a high-dimensional Riemannian space $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, where $\mathcal{X} = \mathbb{R}^p$, with

$$s \ll p.$$

   We let

$$\mathbf{x}_i = \psi(\mathbf{y}_i), \qquad \text{for all } i = 1, 2, \cdots, n.$$

   Remark 1. In the development above,

$$\psi : \mathcal{M} \to \mathcal{X}$$

   is the *embedding map*, and a point on the manifold, $\mathbf{y} \in \mathcal{M}$, can be expressed as

$$\mathbf{y}_i = \varphi(\mathbf{x}_i), \qquad \text{for all } i = 1, 2, \cdots, n,$$

   where $\varphi = \psi^{-1}$.

   Remark 2. We typically taken $\|\cdot\|_{\mathcal{X}}$ to be the Euclidean distance but may use a different distance function.

10. **Main Goal:** The main goal is to recover $\mathcal{M}$ and find an implicit representation of the embedding map ψ and, hence, recover the $\mathbf{y}_i$'s, given only the input data points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \in \mathcal{X}$.

II. Isomap

1. **Assumptions:** The *isometric feature mapping*, or simply *Isomap*, algorithm assumes that

   (a) the smooth manifold $\mathcal{M}$ is a *convex* region of $\mathbb{R}^s$ and

   (b) the embedding map $\psi : \mathcal{M} \to \mathcal{X}$ is an *isometry*.

2. **More on Isometry Assumption:** The isometry assumption implies that the geodesic distance is *invariant* under the map ψ; mathematically, this means

$$d_{\mathcal{M}}(\mathbf{y}, \mathbf{y}') = \|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}, \tag{2}$$

   where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, $\mathbf{y}, \mathbf{y}' \in \mathcal{M}$, and $\mathbf{x} = \psi(\mathbf{y})$ and $\mathbf{x}' = \psi(\mathbf{y}')$.

3. **Comparison to Multidimensional Scaling:** Isomap uses isometry and convexity assumptions to form a nonlinear generalization of multidimensional scaling (MDS).

   - MDS searches for a low-dimensional subspace to embed input data and to preserve the Euclidean distances between pairs of data points;
   - Isomap extends the MDS paradigm by attempting to preserve the global geometric properties of the underlying nonlinear manifold, and it does so by approximating *all* geodesic distances on the manifold.

4. **Procedure — Step 1 (Construct Neighborhood Graph):** Calculate the distances between input data points

$$d_{\mathcal{X},i,j} := d_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathcal{X}}, \qquad \text{for all } i, j = 1, 2, \cdots, n.$$

   With a choice of either an integer K or an $\varepsilon > 0$, determine which data points are "neighbors" on the manifold $\mathcal{M}$ by connecting each point

   - to its K nearest neighbors, or
   - to all points lying within a ball of radius ε of that point.

   After neighbors are identified, we can obtain a *weighted* neighborhood graph

$$\mathcal{G} = (V, E, W),$$

   where

   - the set of vertices $V = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ are the input data points,
   - the set of edges $E = \{e_{i,j}\}_{i,j}$ indicate neighborhood relationships between the points, and
   - the set of weights $W = \{w_{i,j}\}_{i,j}$ indicates the distance between pairs of points, and $w_{i,j} = d_{\mathcal{X},i,j}$ for all $i, j = 1, 2, \cdots, n$.

Remark 1. The choice of K or ε controls the neighborhood size and also the success of Isomap. More specifically,

(a) if K or ε is too large with respect to the manifold structure, the resulting reconstruction is very noisy and slight modification of data can lead to a drastically different (or even incorrect) low-dimensional embedding;

(b) if K or ε is too small, the neighborhood graph may become too sparse to approximate geodesic paths accurately.

Remark 2. If there is no edge present between a pair of points, the corresponding weight is zero.

5. **Procedure — Step 2 (Compute Graph Distances):** In this step, we estimate the unknown true *geodesic distances* between pairs of points. We call the resulting estimates *graph distances* and denote by $d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j)$ for all $i, j = 1, 2, \cdots, n$.

   To this end, we perform the following:

   (a) Initialize $d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j) = d_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are linked by an edge in G (i.e., if $\mathbf{x}_i$ is a neighbor of $\mathbf{x}_j$, or $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$, or $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors of each other), and let $d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j) = \infty$ otherwise;

   (b) Fix a pair of observations $(\mathbf{x}_i, \mathbf{x}_j)$. For each value of $k = 1, 2, \cdots, n$, set

   $$d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j) = \min\Big\{ d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j), d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_k) + d_{\mathcal{G}}(\mathbf{x}_k, \mathbf{x}_j) \Big\}.$$

   Then, the matrix of the final values $\mathbf{D}_{\mathcal{G}} = \{d_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,2,\cdots,n}$ will contain the shortest path distances between all pairs of vertices in $\mathcal{G}$.

   Remark 1. The resulting matrix of graph distances, $\mathbf{D}_{\mathcal{G}}$, is symmetric.

   Remark 2. From the procedure above, note that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are *not* neighbors of one another but are connected by a sequence of neighbor-to-neighbor links, the sum of the link weights along the sequence is taken to be the graph distance between them.

   Remark 3. The procedure of finding the shortest path between all pairs of data points is known as Floyd's algorithm and requires $\mathcal{O}(n^3)$ operations.

6. **Procedure — Step 3 (Embed via MDS):** Apply classical MDS to $\mathbf{D}_{\mathcal{G}}$ to give the reconstructed data points in an s'-dimensional feature space $\mathcal{Y}$.

   Note that $\mathcal{Y}$ is an estimate of the underlying true s-dimensional manifold $\mathcal{M}$, which may or may not coincide with $\mathcal{M}$. Furthermore, s' is an estimate of s and it is possible that $s' \neq s$.

   The procedure is the following:

   (a) Form the doubly centered symmetric $n \times n$ matrix

   $$\mathbf{A}_{\mathcal{G}} = -\frac{1}{2}\mathbf{H}\mathbf{S}_{\mathcal{G}}\mathbf{H},$$

   where the (i, j)-th entry of $\mathbf{S}_{\mathcal{G}}$ is $d^2_{\mathcal{G}}(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{H} := \mathbf{I}_n - \frac{1}{n}\mathbf{J}_n$ is the centering matrix, and $\mathbf{J}_n$ is the $n \times n$ matrix with all entries being 1.

(b) The embedding vectors $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$ are chosen to minimize

$$\|\mathbf{A}_{\mathcal{G}} - \mathbf{A}_{\mathcal{Y}}\|_F^2,$$

where

$$\mathbf{A}_{\mathcal{Y}} = -\frac{1}{2}\mathbf{H}\mathbf{S}_{\mathcal{Y}}\mathbf{H},$$

the (i, j)-entry of $\mathbf{S}_{\mathcal{Y}}$ is the squared Euclidean distance between $\mathbf{y}_i$ and $\mathbf{y}_j$.

The optimal solution is given by the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{s'}$ corresponding to the s' largest positive eigenvalues of $\mathbf{A}_{\mathcal{G}}$.

(c) The graph $\mathcal{G}$ is embedded into $\mathcal{Y}$ by the matrix of shape $s' \times n$ given by

$$\mathbf{Y} := (\widehat{\mathbf{y}}_1, \widehat{\mathbf{y}}_2, \cdots, \widehat{\mathbf{y}}_n) = (\sqrt{\lambda_1}\mathbf{v}_1, \sqrt{\lambda_2}\mathbf{v}_2, \cdots, \sqrt{\lambda_{s'}}\mathbf{v}_{s'})^\top.$$

The i-th column of $\widehat{\mathbf{Y}}$ yields the embedding coordinates in $\mathbf{Y}$ of the i-th data point.

7. **Measurement of the Goodness of Isomap Solution:** With the embedded coordinates given from $\mathbf{Y}$, we can compute the $n \times n$ distance matrix containing the distances between pairs of points in the space $\mathcal{Y}$, which is denoted by $\mathbf{D}_{\mathcal{Y},s'}$.

   To measure how good the Isomap solution is and how closely the distance matrix $\mathbf{D}_{\mathcal{Y},s'}$ approximates the graph distance matrix $\mathbf{D}_{\mathcal{G}}$, we calculate $R^2(s')$, the squared correlation coefficient of all corresponding pairs of entries in $\mathbf{D}_{\mathcal{Y},s'}$ and $\mathbf{D}_{\mathcal{G}}$.

8. **How to Choose the Best s':** To choose the best value of s', we plot $1 - R^2(s')$ against s' for $s' = 1, 2, \cdots, s^*$, where s^* is some pre-specified integer. The intrinsic dimensionality is taken to be the integer at which an "elbow" appears in the plot.

9. **Drawbacks of Isomap:** The Isomap algorithm performs bad with manifolds that

   (a) contain holes,

   (b) have too much curvature, or

   (c) are not convex.

III. Local Linear Embedding

1. **Overview:** The local linear embedding (LLE) algorithm for nonlinear dimensionality reduction is similar in spirit to the Isomap algorithm, but attempts to preserve *local* neighborhood information on the manifold (without estimating the true geodesic distances).

2. **Procedure — Step 1 (Search Nearest Neighbor):** Fix $K \ll p$ and let $\mathcal{N}_{i,K}$ denote the neighborhood of $\mathbf{x}_i$ that contains only its K nearest points measured by Euclidean distance.

   Remark. Here, K could be different for each point $\mathbf{x}_i$.

3. **Procedure — Step 2 (Compute Constrained Least-Squares Fits):** The goal of this step is to reconstruct each $\mathbf{x}_i$ by a linear function of its K nearest neighbors

$$\widehat{\mathbf{x}}_i = \sum_{j=1}^{n} w_{i,j}\mathbf{x}_i,$$

where $w_{i,j} > 0$ if $\mathbf{x}_j \in \mathcal{N}_{i,K}$, and $w_{i,j} = 0$ if $\mathbf{x}_j \notin \mathcal{N}_{i,K}$, and $\sum_{j=1}^{n} w_{i,j} = 1$.

To determine the optimal weights, we let $\mathbf{W} \in \mathbb{R}^{n\times n}$ be a matrix whose (i,j)-th entry is $w_{i,j}$ and solve the following optimization problem

$$\underset{\mathbf{W}\in\mathbb{R}^{n\times n}}{\text{minimize}} \sum_{i=1}^{n}\left\|\mathbf{x}_i - \sum_{j=1}^{n} w_{i,j}\mathbf{x}_j\right\|_2^2, \tag{3}$$

subject to the non-negative constraint $w_{i,j} \geq 0$ for all $i,j = 1,2,\cdots,n$, the row unity constraint

$$\mathbf{W}\mathbf{1}_n = \mathbf{1}_n,$$

the sparseness constraint $w_{i,j} = 0$ if $\mathbf{x}_j \notin \mathcal{N}_{i,K}$.

For convenience, we let the first K components of $\mathbf{w}_i$, the i-th row of $\mathbf{W}$, correspond to K nearest neighbors of $\mathbf{x}_i$ and the remaining $n-K$ components correspond to the other data points. Then, automatically, each of the last $n-K$ components of $\mathbf{w}_i$ is 0.

The optimal values of the first K components of $\mathbf{w}_i$ is given by

$$\widehat{\mathbf{w}}_i = \frac{\mathbf{G}_i^{-1}\mathbf{1}_n}{\mathbf{1}_n^\top\mathbf{G}_i^{-1}\mathbf{1}_n},$$

where the (j,k)-th entry of $\mathbf{G}_i$ is given by

$$(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_k), \qquad \text{for } \mathbf{x}_j, \mathbf{x}_k \in \mathcal{N}_{i,K},$$

for all $j,k = 1,2,\cdots,K$.

4. **Procedure — Step 3 (Solve Eigen Problem):** With the optimal weight matrix $\widehat{\mathbf{W}}$, we find the matrix $\mathbf{Y} \in \mathbb{R}^{s'\times n}$, where $s' \ll p$, of the embedding coordinates that solves

$$\underset{\mathbf{Y}}{\text{minimize}} \sum_{i=1}^{n}\left\|\mathbf{y}_i - \sum_{j=1}^{n} \widehat{w}_{i,j}\mathbf{y}_j\right\|_2^2, \tag{4}$$

subject to the constraints

$$\mathbf{Y}\mathbf{1}_n = \mathbf{0}_{s'}, \qquad \text{and} \qquad \frac{1}{n}\mathbf{Y}\mathbf{Y}^\top = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i\mathbf{y}_i^\top = \mathbf{I}_{s'}.$$

These constraints are adopted to fix the translation, rotation and the scale of the embedding coordinates so that the objective function is invariant.

(a) *Equivalent Expression of* (4)*:* It can be shown that the objective function (4) can be written as

$$\text{trace}(\mathbf{Y}\mathbf{M}\mathbf{Y}^\top),$$

where $\mathbf{M} = (\mathbf{I}_n - \widehat{\mathbf{W}})^\top(\mathbf{I}_n - \widehat{\mathbf{W}}) \in \mathbb{R}^{n\times n}$, which is sparse, symmetric and positive semi-definite.

(b) *Eigenvectors of* $\mathbf{M}$*:* Note that the smallest eigenvalue of $\mathbf{M}$ is 0 with the corresponding eigenvector being $\mathbf{v}_n = n^{-\frac{1}{2}}\mathbf{1}_n$. All other eigenvectors are orthogonal to $\mathbf{v}_n$, implying that the sum of coefficients of each of other eigenvectors is 0. This will constrain the embedding to have mean zero with the constraint $\mathbf{Y}\mathbf{1}_n = \mathbf{0}_{s'}$ being satisfied.

(c) *Optimal Solution of* (4)*:* Let $\widehat{\mathbf{Y}}$ be the minimizer of (4). Then,

$$\widehat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \cdots, \hat{\mathbf{y}}_n) = (\mathbf{v}_{n-1}, \mathbf{v}_{n-2}, \cdots, \mathbf{v}_{n-s'})^\top,$$

where $\mathbf{v}_{n-j}$ is the eigenvector corresponding to the $(j+1)$-st smallest eigenvalue of $\mathbf{M}$.

Remark. The sparseness of $\mathbf{M}$ enables the computation of eigenvectors to be carried out very efficiently.

5. **Comments on LLE:**

(a) *Advantage:* Since LLE preserves *local* (rather than global) properties of the underlying manifold, it is less susceptible to introducing false connections in $\mathcal{G}$ and can successfully embed non-convex manifolds.

(b) *Disadvantage:* Like Isomap, it has difficulty with manifolds that contain holes.

IV. Laplacian Eigenmaps

1. **Overview:** Laplacian eigenmap is very similar to LLE. The main difference is the choice of weight matrix, which also affects the optimization problem solved in the embedding step.

2. **Procedure — Step 1 (Search Nearest Neighbors):** Fix an integer K or an $\varepsilon > 0$. The *neighborhoods* of each data point are symmetrically defined:

(a) for a K-neighborhood $\mathcal{N}_{i,K}$ of the point $\mathbf{x}_i$, let $\mathbf{x}_j \in \mathcal{N}_{i,K}$ if and only if $\mathbf{x}_i \in \mathcal{N}_{j,K}$;

(b) similarly, for an ε-neighborhood $\mathcal{N}_{i,\varepsilon}$, let $\mathbf{x}_j \in \mathcal{N}_{\rangle,\varepsilon}$ if and only if $\|\mathbf{x}_i - \mathbf{x}_j\| < \varepsilon$, where the norm is Euclidean norm.

Remark. In general, let $\mathcal{N}_i$ denote the neighborhood of $\mathbf{x}_i$, regardless of K-neighborhood or ε-neighborhood.

3. **Procedure — Step 2 (Construct Weight Adjacency Matrix):** Let $\mathbf{W} \in \mathbb{R}^{n\times n}$ be a symmetric weighted adjacency matrix defined as

$$w_{i,j} = \begin{cases} \exp\big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\big), & \text{if } \mathbf{x}_j \in \mathcal{N}_i, \\ 0, & \text{otherwise,} \end{cases}$$

where $\sigma > 0$ is the scale parameter. We let the resulting weighted graph be $\mathcal{G}$, where the vertices of $\mathcal{G}$ are the data points, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$.

4. **Procedure — Step 3 (Solve the Eigen-Problem):** Embed the graph $\mathcal{G}$ into the low-dimensional space $\mathbb{R}^{s'}$ by the matrix

$$\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n) \in \mathbb{R}^{s'\times n},$$

where the i-th column of $\mathbf{Y}$ yields the embedding coordinates of the i-th point.

(a) *Graph Laplacian:* Let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be a diagonal matrix with diagonal elements being

$$d_{i,i} = \sum_{j\in\mathcal{N}_i} w_{i,j} = [\mathbf{W}\mathbf{1}_n]_i, \qquad \text{for all } i = 1, 2, \cdots, n.$$

The symmetric matrix

$$\mathbf{L} := \mathbf{D} - \mathbf{W} \in \mathbb{R}^{n\times n}$$

is known as the *graph Laplacian* for the graph $\mathcal{G}$.
Let $\mathbf{y} = (y_1, y_2, \cdots, y_n)^\top \in \mathbb{R}^n$ be an arbitrary vector. Then,

$$\mathbf{y}^\top \mathbf{W}\mathbf{y} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}(y_i - y_j)^2,$$

implying that $\mathbf{L}$ is nonnegative definite.

(b) *Optimization Problem:* We determine the optimal matrix $\mathbf{Y}$ by minimizing

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 = \operatorname{trace}(\mathbf{Y}\mathbf{L}\mathbf{Y}^\top), \tag{5}$$

subject to the constraint $\mathbf{Y}\mathbf{D}\mathbf{Y}^\top = \mathbf{I}_{s'}$.
Remark. The constraint $\mathbf{Y}\mathbf{D}\mathbf{Y}^\top = \mathbf{I}_{s'}$ is to prevent a collapse onto a subspace of fewer than $s' - 1$ dimensions.

(c) *Solution:* Minimizing (5) boils down to solving the generalized eigenequation,

$$\mathbf{L}\mathbf{v} = \lambda\mathbf{D}\mathbf{v},$$

or, equivalently, finding the eigenvalues and eigenvectors of the matrix

$$\widetilde{\mathbf{W}} := \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}.$$

The smallest eigenvalue, λ_n, of $\widetilde{\mathbf{W}}$ is zero with the corresponding constant eigenvector $\mathbf{v}_n = \mathbf{1}_n$. We ignore the smallest eigenvalue and its eigenvector. The best embedding in $\mathbb{R}^{s'}$ is given by

$$\widehat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \cdots, \hat{\mathbf{y}}_n) = (\mathbf{v}_{n-1}, \mathbf{v}_{n-2}, \cdots, \mathbf{v}_{n-s'})^\top,$$

corresponding to the next s' smallest eigenvalues, $\lambda_{n-1} \leq \lambda_{n-2} \leq \cdots \leq \lambda_{n-s'}$, of $\widetilde{\mathbf{W}}$.

Remark. Note that the solution to (5) is very similar to that given by the local linear embedding.

V. Stochastic Neighbor Embedding (SNE) and t-SNE

V.1 Stochastic Neighbor Embedding

1. **Overview:** Both SNE and t-SNE visualize high-dimensional data by giving each data point a location in a two- or three-dimensional map. The coordinates in the lower dimensions are obtained by minimizing the Kullback-Leibler divergence.

2. **Setup:**

   (a) *Similarity in the High-dimensional Space:* In the original high-dimensional space, define the similarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$, where $i \neq j$, as the following conditional probability

   $$p_{j|i} := \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/(2\sigma_i^2))}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|_2^2/(2\sigma_i^2))}, \tag{6}$$

   which is the probability that $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at $\mathbf{x}_i$. In addition, we let

   $$p_{i|i} = 0, \qquad \text{for all } i = 1, 2, \cdots, n.$$

   (b) *Similarity in the Low-dimensional Space:* Let $\mathbf{y}_i$ and $\mathbf{y}_j$ be the low-dimensional counterparts of the high-dimensional data points $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, where $i \neq j$. Let the similarity between $\mathbf{y}_i$ and $\mathbf{y}_j$ be

   $$q_{j|i} := \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|_2^2)}{\sum_{k\neq i}\exp(-\|\mathbf{y}_i - \mathbf{y}_k\|_2^2)},$$

   where we still use the density function of a Gaussian distribution and set the variance to be $\frac{1}{2}$. In addition, we let

   $$q_{i|i} = 0, \qquad \text{for all } i = 1, 2, \cdots, n.$$

3. **Notation:** We adopt the following notation:

   (a) $\mathbb{P}_i$ represents the conditional probability distribution over all other datapoints given datapoint $\mathbf{x}_i$, and

   (b) $\mathbb{Q}_i$ represents the conditional probability distribution over all other map points in the low dimension given the point $\mathbf{y}_i$.

4. **Entropy:** Under the conditional probability distribution, the *entropy* of $\mathbb{P}_i$ is defined as

$$H(\mathbb{P}_i) = -\sum_{j=1}^{n} p_{j|i} \log_2 p_{j|i}.$$

5. **Choice of σ_i^2:** We discuss how to choose σ_i^2 for each $\mathbf{x}_i$.

   (a) *Why Choices of σ_i^2 Depend on Data Points:* Since the density of the data is likely to vary, we choose (possibly) different σ_i^2 for different $\mathbf{x}_i$.

   (b) *Effects of σ_i^2 on Entropy:* If we increase σ_i^2, $H(\mathbb{P}_i)$ also increases.

   (c) *How to Choose σ_i^2:* We performs a binary search for the value of σ_i^2 that produces a conditional distribution $\mathbb{P}_i$ with a pre-specified perplexity, where the *perplexity* is defined as

$$\text{Perp}(\mathbb{P}_i) = 2^{H(\mathbb{P}_i)}.$$

   Remark. The perplexity can be interpreted as a smooth measure of the effective number of neighbors.

6. **Stochastic Neighbor Embedding:**

   (a) *Main Idea:* If the points $\mathbf{y}_i$ and $\mathbf{y}_j$ correctly model the similarity between the high-dimensional datapoints $\mathbf{x}_i$ and $\mathbf{x}_j$, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal.

   (b) *Optimization Problem:* SNE aims to find a low-dimensional data representation that minimizes the mismatch between $p_{j\,|\,i}$ and $q_{j\,|\,i}$. The mismatch is measured by the Kullback-Leibler divergence, and the resulting optimization problem is

$$\text{minimize}\, C(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n),$$

   where

$$C(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n) := \sum_{i=1}^{n} \text{KL}(\mathbb{P}_i \| \mathbb{Q}_i) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right). \tag{7}$$

(c) *Gradient Descent Algorithm to Optimize C:* The derivative of C with respect to $\mathbf{y}_i$ is given by

$$\frac{\partial C}{\partial \mathbf{y}_i} = 2\sum_{j=1}^{n}(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(\mathbf{y}_i - \mathbf{y}_j). \tag{8}$$

Then, one can use the gradient descent algorithm to minimize C. The initial points of $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$ can be randomly selected from an isotropic Gaussian with small variance that is centered around the origin.

(d) *Gradient Descent Algorithm with Momentum to Optimize C:* In order to speed up the optimization and to avoid poor local minima, we can add a momentum term to the plain gradient descent algorithm.
If we let $\mathbf{y}_i^{(t)}$ denote the t-th iterate of $\mathbf{y}_i$, then the gradient descent updates with momentum for $\mathbf{y}_i$ are given by

$$\mathbf{y}_i^{(t)} = \mathbf{y}_i^{(t-1)} - \alpha\left(\frac{\partial C}{\partial \mathbf{y}_i}\bigg|_{\mathbf{y}_i=\mathbf{y}_i^{(t-1)}}\right) + \beta_t(\mathbf{y}_i^{(t-1)} - \mathbf{y}_i^{(t-2)}), \qquad \text{for all } t = 1, 2, \cdots,$$

where $\alpha > 0$ is the learning rate, and β_t is the momentum at the t-th iteration.

7. **Symmetric SNE:** A symmetric version of SNE optimizes

$$C_{\text{sym}}(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n) := \text{KL}(\mathbb{P}\|\mathbb{Q}) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}},$$

where $p_{i,i} = q_{i,i} = 0$ for all $i = 1, 2, \cdots, n$,

$$p_{i,j} = p_{j,i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/(2\sigma^2))}{\sum_{k\neq\ell}\exp(-\|\mathbf{x}_k - \mathbf{x}_\ell\|_2^2/(2\sigma^2))},$$

and

$$q_{i,j} = q_{j,i} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|_2^2/(2\sigma^2))}{\sum_{k\neq\ell}\exp(-\|\mathbf{y}_k - \mathbf{y}_\ell\|_2^2/(2\sigma^2))}.$$

We can then use the gradient descent algorithm to minimize C_{sym} by noting

$$\frac{\partial C_{\text{sym}}}{\partial \mathbf{y}_i} = 4\sum_{j=1}^{n}(p_{i,j} - q_{i,j})(\mathbf{y}_i - \mathbf{y}_j).$$

V.2 t-SNE

1. **Crowding Problem:** SNE described earlier suffers the serious *crowding problem*, meaning that we do *not* have enough spaces to accommodate all neighbors in the higher dimensions.

(a) *Example:* In p-dimensional space, where $p > 1$, there are $p + 1$ data points that are mutually equidistant. Suppose we want to map these $p + 1$ data points to 1-dimensional space in which, if we fix 1 point, there are only exactly 2 data points that have equal distance to this fixed point. Hence, there is no way to model this *faithfully* in a 1-dimensional space.

(b) *Consequence:* If we want to model the small distances accurately in the lower-dimensional map, most of the points that are at a moderate distance from a certain data point will have to be placed too far away in the lower-dimensional map.

2. **Intuition of t-SNE:** In order to solve the crowding problem, the intuition is the following:

   (a) In the high-dimensional space, we convert distances into probabilities using a Gaussian distribution;

   (b) In the low-dimensional map, we use a probability distribution that has much heavier tails than a Gaussian to convert distances into probabilities.

   Why the Intuition Works? The intuition above allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance in the map.

3. **Probabilities in Low-dimensional Space:** In t-SNE, we employ a t-distribution with one degree of freedom (i.e., a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map. Using this distribution, the joint probabilities $q_{i,j}$ are defined as

$$q_{i,j} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^{-1}}{\sum_{k \neq \ell}(1 + \|\mathbf{y}_k - \mathbf{y}_\ell\|_2^2)^{-1}}, \qquad \text{for all } i, j = 1, 2, \cdots, n. \tag{9}$$

4. **Gradient Descent Algorithm for t-SNE:** The gradient of the Kullback-Leibler divergence between $\mathbb{P}$ and the t-distribution based joint probability distribution $\mathbb{Q}$ computed using (9) is given by

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j=1}^{n} (p_{i,j} - q_{i,j})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^{-1}. \tag{10}$$

5. **Algorithm:** The algorithm for t-SNE is given in Algorithm 1.

Algorithm 1 t-Distributed Stochastic Neighbor Embedding

Require: Data, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$;
Require: Cost function parameters, perplexity Perp;
Require: Dimensionality to be mapped to s;
Require: Optimization parameters, number of iterations T, and learning rate η.
1: Compute pairwise similarities $p_{j|i}$ with perplexity Perp;
2: Set $p_{i,j} = \frac{1}{2n}(p_{j|i} + p_{i|j})$;
3: Sample initial points $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n \overset{\text{i.i.d}}{\sim} \text{Normal}(0, 10^{-4}\mathbf{I}_s)$;
4: **for** $t = 1, 2, \cdots, T$ **do**
5: Compute low-dimensional similarity $q_{i,j}$ using (9);
6: Compute the gradient vector by (10);
7: Update $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$ by the gradient descent algorithm.
8: **end for**
9: **return** Low-dimensional data representation, $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$.

6. **Advantages of t-SNE:**

   (a) t-SNE puts emphasis on
   - modeling dissimilar datapoints by means of large pairwise distances, and
   - modeling similar datapoints by means of small pairwise distances.

   (b) The optimization of the t-SNE cost function is much easier than that of SNE.

7. **Disadvantages of t-SNE:**

   (a) Typically, t-SNE is used to reduce the dimensionality to 2 or 3 for visualization purpose. It is not obvious how to extend the t-SNE to perform the more general task of dimensionality reduction (i.e., to reduce the dimensionality to a value greater than 3).

   (b) The t-SNE reduces the dimensionality of data mainly based on *local* properties of the data, which makes it sensitive to the curse of the intrinsic dimensionality of the data.

   (c) The loss function to be minimized in the t-SNE is *not* convex. There is no guarantee that the t-SNE converges to a global optimum. In addition, the constructed solutions depend on these choices of optimization parameters and may be different each time the t-SNE is run from an initial random configuration of map points.

References

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Vol. 1. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Izenman, Alan J (Mar. 2009). *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. en. Springer Science & Business Media.

van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.