ECE598: Representation Learning Fall 2017

Lecture 4: Isomap, SNE, t-SNE

Lecturer: Pramod Viswanath Scribe: Bryan Clifford and Pranav Rao, Sept. 21, 2017

4.1 Problem Setup

The goal of this topic is the same as the previous lectures. We want to reduce the dimensionality of our data while preserving the inherent structure of the data.

- We're given a data set consisting of N samples
$$\{\boldsymbol{x}_n\}_{n=1}^N \subset \mathbb{R}^{d_x}.$$
- We want to reduce the dimensionality of the data but preserve the structure of the data. In this case we mean that we want to preserve relative distances ("similarities") between different $\boldsymbol{x}$s and their lower dimensional representations ($\boldsymbol{y}$s). Specifically, for a given pair of high dimensional (HD) data elements $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and their low-dimensional (LD) counterparts $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ (both in $\mathbb{R}^{d_y}$) then we want to have
$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathcal{X}} \approx \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_{\mathcal{Y}}$$
where $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$ are the measures of distance (or similarity) in the HD data space $\mathcal{X}$ and the LD ("induced") space $\mathcal{Y}$ respectively.

One way to do this, called the Sammon mapping [Sam69], is to solve the following optimization problem
$$\min_{\{\hat{\boldsymbol{y}}_k\}_{k=1}^N} \frac{1}{\sum_{i\leq j}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} \sum_{i\leq j} \frac{(\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|)^2}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}, \tag{4.1}$$
which can be done using a gradient descent method. Here (and in the rest of these notes we leave out the subscripts on the $\|\cdot\|$ operators as they can be determined from context. The denominator in Eq. (4.1) is particularly important since it prevents the summation from being dominated by outliers (data points for which $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ is large for all j).

Unfortunately, many of the classical methods for dimensionality reduction like PCA or Multidimensional Scaling (MDS) [Gow15] which use a Euclidean distance between the data points fail to preserve the non-linear structure of the data. For example, the Euclidean distance between two points on the surface of a sphere may be small but the distance along the sphere between the points may be much larger. For to points on the poles of the Earth the Euclidean distance is the diameter of the Earth d_E, but the surface distance is half the circumference $\frac{1}{2}\pi d_E$.

We will look at three methods that do preserve non-linear or neighborhood structures of the data called **Isomap** [TdSL00], **Stochastic Neighbor Embedding (SNE)** [BF85], and **t-SNE** [vdMH08] which are described in the following sections. But first, we will take a closer look at the classical technique, MDS, in order to better set the stage for our descriptions of the more modern techniques.

4.2 MDS

MDS is a classical technique of dimensionality reduction that can account for both Euclidean and non-Euclidean distances. Most generally, the input to MDS is 'dissimilarity' between data points $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$, and not the data itself, so we assume we only have access to the matrix $\mathbf{D} = [d_{ij}]$. The goal of reducing dimensionality while preserving data structure thus leads to an objective or *stress* function that looks like [FHT01]:

$$S_{\text{stress}} = \sum_{i \neq j} (d_{ij} - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|)^2 \tag{4.2}$$

The above would have quite large contributions from outliers with $d_{ii'}$ large, and thus would weight those outliers considerably more than other data points, so the so-called *Sammon mapping* is to normalize, e.g.

$$S_{\text{sam}} = \sum_{i \neq j} \frac{(d_{ij} - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|)^2}{d_{ij}} \tag{4.3}$$

and minimize the above function with respect to the lower dimensional representation, using gradient descent or something similar. If, however, one was to use *classical scaling*, we would minimize the following *strain* function:

$$S_{\text{strain}} = \sum_{i \neq j} \left(s_{ij} - \langle \boldsymbol{y}_i - \bar{\boldsymbol{y}}, \boldsymbol{y}_j - \bar{\boldsymbol{y}} \rangle\right)^2 \tag{4.4}$$

Above, $s_{ij} = \langle \boldsymbol{x}_i - \bar{\boldsymbol{x}}, \boldsymbol{x}_j - \bar{\boldsymbol{x}} \rangle$ is referred to as a similarity. The similarity matrix is then $S = \boldsymbol{X}_c \boldsymbol{X}_c^T$ if the matrix $\boldsymbol{X}$ stores the data $(\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ and $\boldsymbol{X}_c$ represents the centered version of $\boldsymbol{X}$. The cost function above is readily minimized by PCA (if the distances are Euclidean, which we will assume henceforth), but in MDS, we are given dissimilarity data, not the data itself. We can relate the similarities in (4.4) to dissimilarities d_{ij} by noting that:

$$d_{ij}^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2 + \|\boldsymbol{x}_j - \bar{\boldsymbol{x}}\|^2 - 2\langle \boldsymbol{x}_i - \bar{\boldsymbol{x}}, \boldsymbol{x}_j - \bar{\boldsymbol{x}} \rangle \tag{4.5}$$

By completing the square. If we wanted to solve for the last term, which is s_{ij}, we'd first perform a double centering on the matrix $\mathbf{D}^2$ (element-wise squaring, not a matrix square) with the matrix $\mathbf{H} = \left(\mathbf{I} - \frac{1}{N}\mathbf{1}_N\right)$ where $\mathbf{1}_N$ is an $N \times N$ matrix of ones. Also, we would need to multiply by $-\frac{1}{2}$, yielding:

$$\mathbf{S} = -\mathbf{H}\frac{\mathbf{D}^2}{2}\mathbf{H} \tag{4.6}$$

As the $\mathbf{S}$ matrix is in the form $\mathbf{X}_c\mathbf{X}_c^T$ which is the Gram matrix one diagonalizes in PCA, we can now take $\mathbf{S} \to \mathbf{U}\mathbf{S}'$ (if we define the SVD as $\mathbf{X}_c = \mathbf{U}\mathbf{S}'\mathbf{V}^T$). Classical scaling is, thus, equivalent to PCA (if the norms used are all Euclidean) – we just receive a different input – dissimilarity instead of similarity – which can be related via double-centering.

4.3 Isomap

Classical methods such as PCA or MDS don't preserve the non-linear structure of the data. Hence Isomap was developed as a means of computing an embedding that preserves the local structure of the data. It is very similar to MDS; however, it assumes that the data lie on some sort of LD non-linear manifold that is embedded in the HD space $\mathcal{X}$, and computes the distances (or dissimilarities) between points with geodesic[1] distances of the manifold. The key assumption is that the geodesic distances on the manifold can be computed using neighborhood graphs. With these distances computed Isomap then solves the typical MDS problem using the PCA-based method described in the previous section. Specifically Isomap solves

$$\min_{\{\hat{\boldsymbol{y}}_k\}_{k=1}^N} \sum_{i \leq j} \left(d_{ij}^G - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|\right)^2. \tag{4.7}$$

where d_{ij}^G is the graph-based estimate of the geodesic distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The major difficulty of the method lies in computing the d_{ij}^G from the data. Once these values are known the $\boldsymbol{y}$s can be estimated very efficiently by rewriting Eq. (4.7) as

$$\min_{\{\hat{\boldsymbol{y}}_k\}_{k=1}^N} \|\tau(\mathbf{D}_G) - \tau(\mathbf{D}_y)\|_F \tag{4.8}$$

where $\{\mathbf{D}_G\}_{i,j} = d_{ij}^G$, $\{\mathbf{D}_y\}_{i,j} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|$ and $\|\cdot\|_{\mathcal{Y}}$ is the Euclidean distance. The operator $\tau(\cdot)$ is called the "centering" operator and is defined as

$$\tau(\mathbf{D}) = -\frac{1}{2}\mathbf{H}\mathbf{D}^2\mathbf{H} \tag{4.9}$$

with $\{\mathbf{D}^2\}_{i,j} = \{\mathbf{D}\}_{i,j}^2$.

As shown in the previous section, the solution to Eq. (4.8) is to compute the eigenpairs of $\tau(\mathbf{D}_G)$. Denoting the p^{th} eigenvalue and vector as λ_p and $\boldsymbol{v}_p$ respectively, then the p^{th} coordinate of $\boldsymbol{y}_i$ is given by

$$y_{i,p} = \sqrt{\lambda_p}\boldsymbol{v}_{p,i}. \tag{4.10}$$

Computing d_{ij}^G consists of two steps: (a) form the graph from the data, and (b) find the shortest path between each pair of points on the graph. To compute the graph we first compute the distances between all the data points $d_{ij}^{\mathcal{X}} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. Then for each point we compute its neighbors. Either we choose the K nearest neighbors or we choose $\{\boldsymbol{x}_j | d_{ij}^{\mathcal{X}} \leq \epsilon\}$. The graph then has a node for each point with links between the point's neighbors that have edge lengths given by the distance between the points. Since small local regions of Euclidean manifolds can be approximated as planes (linear approximation), if the neighborhood size ϵ is made small enough, then the geodesic distance between neighbors can be approximated with d_{ij}^G. Then we can find d_{ij}^G between two unconnected points on the graph by finding the shortest path between them.

Finding the shortest path between two points can be done using Floyd's algorithm or other efficient path finding algorithms such as Dijkstras Algorithm (which works for non-symmetric graphs too[2]). Floyd's algorithm is as follows.

[1] A geodesic is the shortest possible path between two points on a curved surface

[2] This is particularly useful when the graph is determined by selecting a fixed number of nearest neighbors, since this can lead to asymmetric graphs if not handled carefully

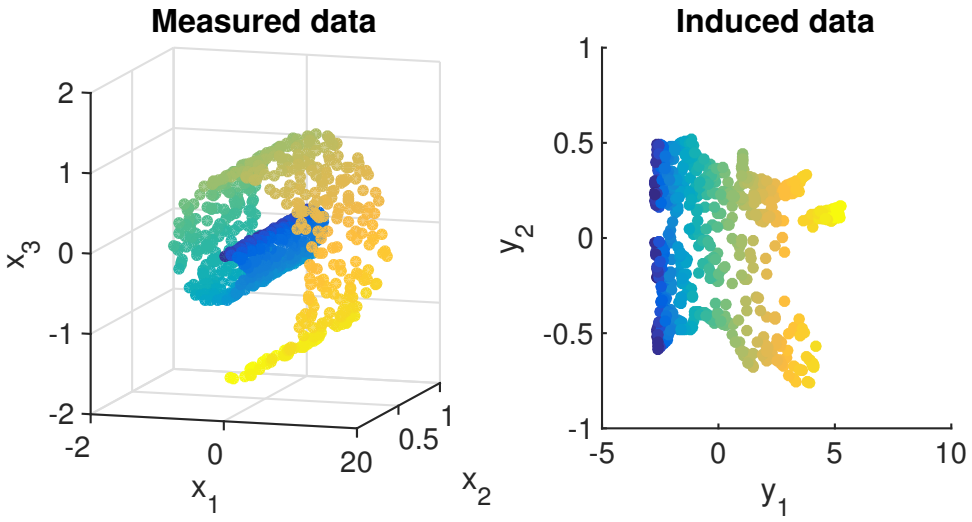

Figure 4.1: Simulation results for order an Isomap example where the data are distributed in a 3D swiss roll. The left plot shows the original measured 3D data and the right plot shows the 2D induced (transformed) data. It is obvious that the mapping has preserved the local structure of the data and "unfolded" the swiss roll.

1. Initialize the $\boldsymbol{d}_{ij}^G$ for neighbors with their associated edge weights and set the values of $\boldsymbol{d}_{ij}^G$ for non-neighbors to infinity.
2. For $k = 1, 2, \ldots, N$ set $d_{ij}^G = \min\{d_{ij}^G, d_{ik}^G + d_{kj}^G\}$.

As mentioned in the beginning of this section, Isomap hinges on the ability to approximate geodesic or manifold distance d_{ij}^M by the graph distance d_{ij}^G, which are based on small Euclidean point-to-point distances. The authors of Isomap provide a proof illustrating the bounds of this approximation. A sketch of their proof follows: it can be shown that geodesic distances from point i to point j can be approximated by geodesics of points connecting i and j, and these geodesics can be approximated to d_{ij}^G using geometric considerations, and finally they present an inequality showing $d_{ij}^M \approx d_{ij}^G$ [TdSL00],

$$1 - \lambda_1 \leq \frac{d_{ij}^G}{d_{ij}^M} \leq 1 + \lambda_2 \tag{4.11}$$

Above, λ_1, λ_2 are arbitrarily small parameters, and given another small parameter μ, they can guarantee with probability $1 - \mu$ that all points i, j can satisfy the above inequality (the parameters are coupled to quantities in the algorithm like k the number of nearest neighbors, and each other, though μ looks arbitrary above).

One drawback to the given Isomap algorithm is that for noisy data, shortcuts between nodes can be found. If the data are very noisy, a random-walk or diffusion based distance measurement can be found so that an average path length is computed instead. The t-SNE paper uses a method like this when computing the induced vectors from only a subset of the original data [vdMH08].

The code in section 4.6 shows an example of performing Isomap on a set of points with a non-linear structure. The points are originally distributed along a 3D swiss-roll, but after the transformation they are unfolded onto a rectangle as shown in Fig. 4.1.

4.4 SNE

Stochastic neighbor embedding takes a probabilistic approach at modeling neighborhoods instead of a deterministic approach like Isomap. One of the drawbacks of Isomap is that it doesn't work well for situations where the same object belongs to several disparate locations. For example if data are arranged in a circle, the points at the "cut" of the circle could belong to both ends of the reduced dimensional space (a line).

Like Isomap SNE doesn't specify that the distances in the data space need to be Euclidean, but unlike Isomap, SNE models the probabilities that two points are neighbors. Specifically the probability that point $\boldsymbol{x}_j$ is a neighbor of the point $\boldsymbol{x}_i$ is given as

$$p_{j|i} = \frac{\exp(-d_{ij}^{\mathcal{X}})}{\sum_{k \neq i} \exp(-d_{ik}^{\mathcal{X}})} \tag{4.12}$$

and $p_{j|i} := 0$. The distance measure suggested by SNE is

$$d_{ij} = \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma_i^2} \tag{4.13}$$

so that the prior probability will decay as a Gaussian placed over the point, thus preserving neighbor relationships (the local data structure). The σ_i parameter is a way of specifying the neighborhood sizes of points (similar to ϵ in Isomap).

To find the induced coordinates, SNE tries to pick $\boldsymbol{y}_i$ such that the induced neighborhood priors

$$q_{j|i} = \frac{\exp(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|/2)}{\sum_{k \neq i} \exp(-\|\boldsymbol{y}_i - \boldsymbol{y}_k\|/2)} \tag{4.14}$$

are as $p_{j|i}$ even though the dimension as been reduced.[3] To do this SNE solves the following optimization problem

$$\min_{\{\hat{\boldsymbol{y}}_k\}_{k=1}^N} \sum_{i,j} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}. \tag{4.15}$$

The objective function $KL(\{\boldsymbol{y}_n\}_{n=1}^N$ is the Kullback-Leibler divergence. It is asymmetric, and assigns a larger cost to model points that are close in $\mathcal{X}$ (large $p_{j|i}$) as far apart in $\mathcal{Y}$ (small $q_{j|i}$) than it does to points that are far apart in $\mathcal{X}$ (small $p_{j|i}$) but modeled as close in $\mathcal{Y}$ ($q_{j|i}$). This is a beneficial feature as it leads to better clustering.

The objective function is solved using a gradient descent approach with annealing and momentum.[4] Specifically

$$\boldsymbol{y}_i^{(k)} = \boldsymbol{y}_i^{(k-1)} - \alpha \frac{\partial KL}{\partial \boldsymbol{y}_i} + \beta(\boldsymbol{y}_i^{(k-1)} - \boldsymbol{y}_i^{(k-2)}) + \boldsymbol{\eta}(\theta(k))) \tag{4.16}$$

where α is the gradient step size, and β is the momentum parameter and $\boldsymbol{\eta}$ is additive iid noise with parameters that decrease its variance with each iteration. Adding momentum and noise help the descent avoid shallow local minima.

[3] The fact that there is no sigma parameter here is in order to make the induced space more regular.

[4] Annealing in this case means that noise is added to the gradient step, but with the noise variance gradually decreasing with each iteration.

The gradient is given by

$$\frac{\partial KL}{\partial \boldsymbol{y}_i} = -2\sum_j [(p_{j|i} - q_{j|i}) - (p_{i|j} - q_{i|j})](\boldsymbol{y}_j - \boldsymbol{y}_i) \tag{4.17}$$

and has a very useful interpretation as a force of attraction to other points in the embedded space. The force acts as if there are springs attached to $\boldsymbol{y}_i$ from all of the other points. The direction of the force is in the direction of the other points and the force is proportional to the distance and the relative difference between priors.

The problem can also be generalized to allow one-to-many mappings (which was a situation where Isomap would fail), which allows SNE to do things like disambiguate homonyms. To do this, the induced prior can be modeled as a weighted sum of priors (mixture of priors) as

$$q_{j|i} = \sum_b \pi_{ib} \sum_c \pi_{jc} \frac{\exp(-\|\boldsymbol{y}_{ib} - \boldsymbol{y}_{jc}\|/2)}{\sum_{kd} \pi_{kd} \exp(-\|\boldsymbol{y}_{ib} - \boldsymbol{y}_{kd}\|/2)}. \tag{4.18}$$

where the weights π_{ib} are the mixing probabilities for the b^{th} induced location of point $\boldsymbol{x}_i$. The meaning is that each of point $\boldsymbol{y}_i$'s mixtures is the sum of the mixtures of the other points.

Lastly, an interesting fact about SNE is that as σ_i gets very large, the optimization becomes almost equivalent to the Sammon mapping. [BF85].

4.5 t-SNE

There are two major problems with SNE

1. The cost function is hard to optimize
2. **The crowding problem.** In order to maintain average distances in a LD space from a HD space, points that are equidistant in the HD space get crowed together in the LD space. For example in 2D there can be 3 points equidistant from each other (vertices's of an equilateral triangle) but in 1D only 2 points can be equidistant to each other so the solution is to crowd all of the points together. A similar situation occurs for points on a sphere since the surface area of the sphere is much large than the surface area of a circle (Fig. 4.2).

t-SNE was designed to solve both of these problems. To make the cost function easier to optimize, symmetric definitions of the neighborhood probabilities are used, and to solve the crowding problem, t-SNE uses a heavier tailed distribution (Student t-distribution with 1 degree of freedom) in the induced space.

More precisely, in t-SNE replaces the prior distributions $p_{j|i}$ and $q_{j|i}$ with joint distributions p_{ij} and q_{ij}.

$$\begin{aligned} p_{ij} &= \frac{p_{j|i} + p_{i|j}}{2N} \\ q_{ij} &= \frac{(1 - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|/2)^{-1}}{\sum_{k \neq l}(1 - \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}} \end{aligned} \tag{4.19}$$

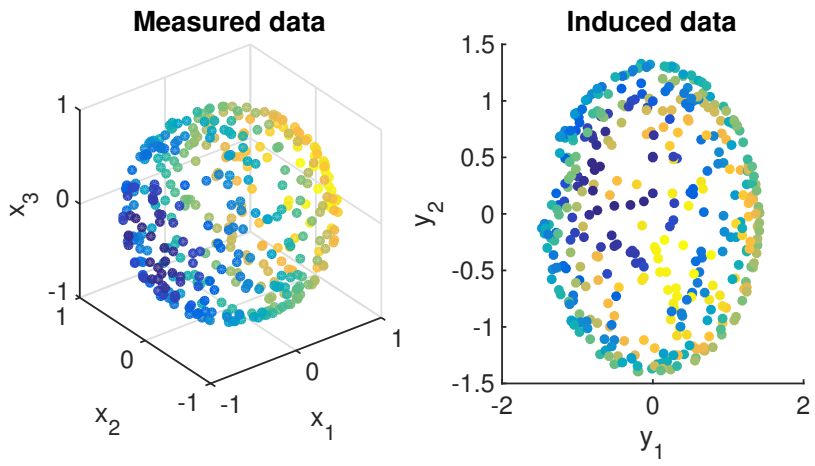

Figure 4.2: Simulation results for order an Isomap example where the data are evenly distributed in a the surface of a sphere. The left plot shows the original measured 3D data and the right plot shows the 2D induced (transformed) data. The color of the data is proportional to the x_1 coordinate. In the reduced dimensionality some of the points get crowed towards the edges of the 2D area.

These are joint distributions in the sense that they both add up to 1 when summed over all i and j, while the prior distributions in SNE summed up to 1 when summed over j. The cost function is the same but with the priors replaced with the joint probabilities.

The definition of p_{ij} is left asymmetric from that of q_{ij} so as to make it less sensitive to outliers (if written as q_{ij} the a large outlier $\boldsymbol{x}_i$ could make the p_{ij} small for all j which would mean that the other points have little effect on the cost associated with $\boldsymbol{x}_i$) and therefore easier to optimize. It also makes the gradient a little simpler. The different distribution in $\boldsymbol{q}_{ij}$ has heavier tails than a Gaussian which makes the effect of far away points larger than when the Gaussian distribution was used. This in turn means that distances can be larger in the induced space than they would be for SNE, which helps to avoid the crowding problem.

The gradient of the cost function becomes

$$\frac{\partial KL}{\partial \boldsymbol{y}_i} = -4 \sum_j (p_{ij} - q_{ij}) \frac{\boldsymbol{y}_j - \boldsymbol{y}_i}{1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2}. \tag{4.20}$$

Making the same connected spring system comparison as for Eq. (4.17) we see that the use of the Student t-distribution with 1 degree of freedom has led to a normalization term, which makes the magnitude of the force felt by disparate $\boldsymbol{y}$s proportional only to the difference in probabilities an not to their actual distance. This property leads to much better clustering and avoids the crowding problem (since only probability matters, not Euclidean distance).

4.6 Isomap Example Code

```
%% ------ Perform Isomap on random example ------------------------------------
N = 800; % number of samples
K = 6;
```

```
sigma = 0.01;

% generate data
z  = 1.5*rand(N,1);
x1 = z.*sin(2*pi*z) + sigma*randn(N,1);
x2 = rand(N,1);
x3 = z.*cos(2*pi*z) + sigma*randn(N,1);
x  = [x1,x2,x3];

%% compute distances between the points
Dx = zeros(N,N);
for n = 1:N
    for m = n+1:N
        Dx(n,m) = norm(x(n,:) - x(m,:));
        Dx(m,n) = Dx(n,m);
    end
end

%% compute graph
G = zeros(N,N);
for n = 1:N
    [~,inds] = sort(Dx(n,:),'ascend');
    inds     = inds(2:K+1); % smallest distance is the diag element (the same point
    for m = inds % force symmetry
        G(n,m) = Dx(n,m);
        G(m,n) = G(n,m);
    end
end

%% estimate geodesic distances
Dg = G;
Dg(Dg == 0) = inf;
Dg(1:N+1:end) = 0;
for k = 1:N
    for n = 1:N
        for m = 1:N
            d = Dg(n,k) + Dg(k,m);
            if d < Dg(n,m)
                Dg(n,m) = d;
            end
        end
    end
end

%% center distances
Dg_c = Dg.^2;
Dg_c = Dg_c - repmat(mean(Dg_c,1),N,1);
```

```
Dg_c = Dg_c - repmat(mean(Dg_c,2),1,N);
Dg_c = -Dg_c./2;

%% compute eigen values
[v,lambda] = eig(Dg_c);
v = v(:,1:2);
lambda = lambda(1:2,1:2);

%% compute new coordinates
y = v*sqrt(lambda);


%% display results
figure;
% ---- measured ----
subplot(1,2,1);
scatter3(x(:,1),x(:,2),x(:,3),128,z(:,1),'filled');
title('Measured data');
xlabel('x_1');
ylabel('x_2');
zlabel('x_3');
% ---- induced ----
subplot(1,2,2);
scatter(y(:,1),y(:,2),128,z(:,1),'filled');
title('Induced data');
xlabel('y_1');
ylabel('y_2');
```

Bibliography

[BF85] L. Breiman and J. Friedman. Estimating optimal transformations for multiple regression and correlation. *J. Am. Stat. Assoc.*, 80:580 – 598, 1985.

[FHT01] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[Gow15] J. Gower. *Principal Coordinates Analysis.* John Wiley & Sons, Ltd, 2015.

[Sam69] J. Sammon. A nonlinear mapping for data structure analysis. *IEEE Tran. Comp.*, 18:401 – 409, 1969.

[TdSL00] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Sci.*, 290:2319 – 2323, 2000.

[vdMH08] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *J. Machine Learn. Res.*, 9:2579 – 2605, 2008.