Assignment 2:
Spectral Theory on Graphs and Manifold Learning

TA in charge of the exercise: Ya-Wei Eileen Lin (lin.ya-wei@campus.technion.ac.il)
Submission date: December 20th (20/12/2020)
Relevant course material: Lectures 4, 5, 6 & Recitations 4, 5, 6

1. **Homework scale:**
   (a) **Single submission** - This assignment holds five questions out of which you are expected to present **four**. If you would like, you may present all five for 20% extra credit. You are free to choose the questions you are most interested to solve – all hold equal weight.
   (b) **Submitted in pairs** - This assignment holds five questions out of which you are expected to present **five**.
2. **Submission Format.** Please submit your solutions to the appropriate homework submission box on Moodle as a ***zip*** file – with the file name ***HW2_ID.zip*** for single submission and ***HW2_ID1_ID2.zip*** for pair submission (where IDs are you and your partner's ID number).
   Your solution zip should contain:
   - A single PDF file with the name ***HW2_ID_Solution.pdf*** for single work and ***HW2_ID1_ID2_Solution.pdf*** for pair submission.
     Your report should be **typed** and presented in **English**. No other presentation format will be accepted. **The report should conclude all the results and the figures you obtained in the wet parts.**
   - Source code implementation for you wet part. Please make sure the code runs directly after uncompressing the zip file. Code must be written **solely** in the Python language unless directly stated otherwise. You may use any Python module you like, but we expect to see mostly your own coding here.
   - Additional miscellaneous (such as GIF files, images etc.) as requested in the assignment.
3. **Grading.** Please clearly show your work. Partial credit will be awarded for ideas *toward* the solution, so please submit your thoughts on an exercise even if you cannot find a full solution. *If you don't know where to get started with some of these exercises, just ask!* A great way to do this is to leave questions on our Piazza forum or to schedule reception hours with the TA in charge of the exercise. Maximal grade in this exercise is **120**, including all possible bonuses.
4. **Collaboration and External Resources.** You are encouraged to discuss all course material with your peers, including the written and coding assignments. You are especially encouraged to seek out new friends from other disciplines (CS, Math, EE, etc.) whose experience might complement your own. However, your final work must be your own, i.e., direct collaboration on assignments is prohibited. You are allowed to refer to any external resources - if you use one, cite such help on your submission. If you are caught cheating, you will get a zero for the entire course.
5. **Late Days.** Due to the large period of time given to this exercise's submission, we will accept no late day requests unless for official Technion reasons - reserve duty, hospitalization, or tragedy. Please be responsible with this
6. **Parenting Concessions.** By Technion regulations, parents are eligible for concessions with the homework. Please contact the Head TA for details if you are eligible.
7. **Warning!** With probability 1, there are typos in this assignment. If anything in this handout does not make sense (or is blatantly wrong), let us know!

1 Spectral Theory, Eigenvalues and Optimization (Dry)

In the following exercises, we consider $\boldsymbol{M}$ as an $n \times n$ real symmetric matrix. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of $\boldsymbol{M}$ with the corresponding eigenvectors $\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_n$.

1. [*Eigenvector orthogonality for symmetric matrices*] Show that if $\lambda_i \neq \lambda_j$, then $\boldsymbol{\psi}_i$ must be orthogonal to $\boldsymbol{\psi}_j$.
2. [*Spectrum's invariance to permutations*] Let V be a set of $\{1, 2, \ldots, n\}$. Let $\boldsymbol{\Pi}$ be a permutation matrix corresponding to a permutation $\pi: V \to V$ where
$$\boldsymbol{\Pi}(i,j) = \begin{cases} 1, \text{if } i = \pi(j) \\ 0, \text{ otherwise} \end{cases}$$
and $i, j \in V$. Show that the permutation of the coordinates of the matrix permutes the eigenvector coordinates with no change to the eigenvalues, i.e.:
$$(\boldsymbol{\Pi}\boldsymbol{M}\boldsymbol{\Pi}^\top)\boldsymbol{\Pi}\boldsymbol{\psi}_i = \lambda_i \boldsymbol{\Pi}\boldsymbol{\psi}_i.$$
3. [*Courant-Fischer Theorem*] Show that
$$\lambda_k = \min_{\boldsymbol{x} \perp \boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_{k-1}} \frac{\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}} \text{ and } \boldsymbol{\psi}_k = \operatorname*{argmin}_{\boldsymbol{x} \perp \boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_{k-1}} \frac{\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}}.$$
4. [*Eigenvalue Interlacing Theorem*] Consider a matrix $\boldsymbol{H} \in \mathbb{R}^{m \times m}$ with $m < n$ as a principal submatrix of $\boldsymbol{M}$, that is, the matrix $\boldsymbol{H}$ is obtained by deleting both the i-th row and the i-th column from $\boldsymbol{M}$. Let $h_1 \leq h_2 \leq \cdots \leq h_m$ be the eigenvalues of $\boldsymbol{H}$. Show that
$$\lambda_k \leq h_k \leq \lambda_{k-m+n} \text{ for } k = 1, \ldots, m,$$
and if $m = n - 1$, then
$$\lambda_1 \leq h_1 \leq \lambda_2 \leq h_2 \leq \cdots \leq h_{n-1} \leq \lambda_n.$$
5. [*Courant-Weyl inequality*] Consider a real symmetric matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ whose eigenvalues are $c_1 \leq c_2 \leq \cdots \leq c_n$ and another matrix $\boldsymbol{Z} = \boldsymbol{M} + \boldsymbol{C}$ whose eigenvalues are $z_1 \leq z_2 \leq \cdots \leq z_n$. Show that
   (a) $z_i \geq \lambda_j + c_{i-j+1}$ for $1 \leq j \leq i \leq n$
   (b) $z_i \leq \lambda_j + c_{i-j+n}$ for $1 \leq i \leq j \leq n$

2 Graph Adjacency and Graph Laplacian (Dry)

1. [*Perron-Frobenius Theorem in the symmetric case*] The adjacency matrix $\boldsymbol{A}$ of an undirected and unweighted graph $G = (V, E)$ is an $n \times n$ matrix given by
$$\mathbf{A}(i,j) = \begin{cases} 1, & (i,j) \in E \\ 0, & \text{otherwise,} \end{cases}$$
where the number of the vertices is denoted by $n \coloneqq |V|$.
Assume that G is connected. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathbf{A}$ with the corresponding eigenvectors $\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_n$. Show that
   (a) every coordinate of the eigenvector $\boldsymbol{\psi}_1$ is strictly positive
   (b) $\lambda_1 \geq -\lambda_n$
   (c) $\lambda_1 > \lambda_2$
2. [*Laplacian is SPD*] We define the Laplacian matrix as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{M}$, where $\boldsymbol{M}$ is the adjacency matrix and $\boldsymbol{D} = \text{diag}(\boldsymbol{M}\mathbf{1})$ is the degree matrix. Show that the **smallest** eigenvalue of any Laplacian matrix $\boldsymbol{L}$ is 0 and find its corresponding eigenvector.
3. [*Interlacing Theorem for the graph Laplacian*] Consider two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ that have the same vertex set and $E_2 = E_1 - \{e\}$, where $e \in E_1$. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$ be the eigenvalues of the Laplacian matrix of G_1 and G_2, respectively.
   (a) Show that
$$0 = \mu_1 = \lambda_1 \leq \mu_2 \leq \lambda_2 \leq \cdots \leq \mu_n \leq \lambda_n.$$
   (Hint: You may like to consider the *Eigenvalue Interlacing Theorem* and the *Courant-Weyl inequality* in exercise 1.)
   (b) Demonstrate the above relation for $G_1 = R_n$ and $G_2 = P_n$ for any $n \geq 3$ by using the closed-form expression of their eigenvalues. (See recitation 6 for the definition of ring graph R_n and path graph P_n.)

3 Fundamental Graphs (Wet)

1. Consider a path graph P_n and a ring graph R_n with $n = 201$ vertices.
   (a) Compute in code their corresponding adjacency matrix, degree matrix, and the Laplacian matrix.
   (b) Compute in code their corresponding Laplacian matrix eigen-decomposition. Order the obtained eigenvalues from the smallest to the largest.
   (c) Plot
      i. the graph topology of P_n and R_n – use the `networkx` package[1]
      ii. the sorted eigenvalues of the Laplacian matrix of P_n and R_n as a function of their order statistic
      iii. the graph topology of P_n and R_n colored by its first, second, fifth, and tenth eigenvectors corresponding to the ordered eigenvalues. Provide intuition of what you see.
   (d) Compare the obtained eigenvalues and eigenvectors of P_n and R_n to their analytic expressions (as shown in recitation 6).
2. Consider a product graph $G_{n_x,n_y} = R_{n_x} \times R_{n_y}$, where R_{n_x} and R_{n_y} are two ring graphs. We consider $n_x = 71$ and $n_y = 31$ in this question.
   (a) Generate the set of the nodes $\{\boldsymbol{x}_i\}_{i=1}^{N}$, where $N = n_x \times n_y$ and $\boldsymbol{x}_i \in \mathbb{R}^3$.
   (Hint: Think about what is the topology of such product graph. Elaborate on this in your explanations.)
   (b) Compute in code the adjacency matrix, degree matrix, and the Laplacian matrix of G_{n_x,n_y}.
   (c) Compute in code the eigen-decomposition of the Laplacian matrix and order the obtained eigenvalues from the smallest to the largest.
   (d) Plot
      i. the graph topology of G_{n_x,n_y}– use the `networkx` package
      ii. the sorted eigenvalues of the Laplacian matrix of G_{n_x,n_y} as a function of their order statistic
      iii. the graph topology of G_{n_x,n_y} colored by its first, second, fifth, seventh, and tenth eigenvectors corresponding to the ordered eigenvalues. Provide intuition of what you see.
   (e) Compare the obtained eigenvalues and eigenvectors of G_{n_x,n_y} to the analytic expressions (as shown in recitation 6) in terms of the product of two ring graphs.
3. In this question we will examine the eigenvector stability under small noise perturbations. We consider the product graph G_{n_x,n_y} from question 2.
   (a) Generate the set of the nodes $\{\boldsymbol{y}_i\}_{i=1}^{N}$, which is a permuted and noisy version of $\{\boldsymbol{x}_i\}_{i=1}^{N}$ given by
   $$\boldsymbol{y}_i = \boldsymbol{x}_{i_{\pi(i)}} + \boldsymbol{\eta}_i,$$
   where π is a random permutation and the noise vector $\boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, 0.01\boldsymbol{I}_3)$. Denote this graph by $\tilde{G}_{n_x,n_y}$.
   (b) Compute in code the adjacency matrix of $\tilde{G}_{n_x,n_y}$, which is determined by Euclidean distances with a proper threshold such that for each vertex, the number of connected vertices is less than or equal to 8 (i.e., epsilon-neighborhood graph).

[1] https://networkx.org/documentation/stable/index.html

(c) Compute in code the degree matrix and the Laplacian matrix of $\tilde{G}_{n_x,n_y}$.

(d) Compute in code the eigen-decomposition of the Laplacian matrix and order the obtained eigenvalues from the smallest to the largest.

(e) Plot

i. the graph topology of $\tilde{G}_{n_x,n_y}$– use the `networkx` package

ii. the sorted eigenvalues of the Laplacian matrix of $\tilde{G}_{n_x,n_y}$ as a function of their order statistic

iii. the graph topology of $\tilde{G}_{n_x,n_y}$ colored by its first, second, fifth, seventh, and tenth eigenvectors corresponding to the ordered eigenvalues. Compare to the results obtained in question 2 (d) iii

4 Normalized Graph Laplacian and Lazy Random Walk (Dry + Wet)

1. Let $\boldsymbol{W}_G$ be the lazy random walk matrix of a **connected** and undirected graph G, whose number of the vertices is n. Let $\boldsymbol{p} \in \mathbb{R}^n$ be an arbitrary probability distribution.
   (a) Show that $\boldsymbol{p}^\top \boldsymbol{\pi} > 0$, where $\boldsymbol{\pi}$ is the stable distribution of $\boldsymbol{W}_G$.
   (b) [*Uniqueness of the stable distribution*] Show that if $\boldsymbol{W}_G \boldsymbol{p} = \boldsymbol{p}$, then $\boldsymbol{p} = \boldsymbol{\pi}$.
   (Hint: Understand why the graph must be connected.)
   (c) [*Spectral dominance of the stable distribution vector*] Prove that $\boldsymbol{\pi}$ is the dominant eigenvector of $\boldsymbol{W}_G$, that is, it corresponds to the largest eigenvalue.

2. ***Power method***[2] is an efficient iterative method for finding the eigenpairs of a diagonalizable matrix.
In this exercise, we will show that the proof of the convergence of any density vector $\boldsymbol{p}$ to the steady-state distribution $\boldsymbol{\pi}$ is, in fact, a special case of the ***power method*** algorithm.
Let $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{H}\boldsymbol{V}^{-1}$ be some diagonalizable matrix, where $\boldsymbol{H} = \mathrm{diag}(h_1, h_2, \dots, h_n)$ such that $|h_1| \geq |h_2| \geq \cdots \geq |h_n|$. Let $\boldsymbol{u}_0$ be some arbitrary vector such that $\langle \boldsymbol{z}_1^\top, \boldsymbol{u}_0 \rangle \neq 0$, where $\boldsymbol{z}_1$ is the first row of $\boldsymbol{V}^{-1}$. Consider the following update:

$$\boldsymbol{u}_{t+1} = \frac{\boldsymbol{B}\boldsymbol{u}_t}{||\boldsymbol{B}\boldsymbol{u}_t||}.$$

The power method claims that

$$\boldsymbol{u}_t \xrightarrow{t\to\infty} \pm \boldsymbol{v}_1 \text{ and } \boldsymbol{u}_t^\top \boldsymbol{B} \boldsymbol{u}_t \xrightarrow{t\to\infty} h_1.$$

(a) You will prove it by the following steps:
   i. Expand $\boldsymbol{u}_0$ by the columns of $\boldsymbol{V}$, that is,

$$\boldsymbol{u}_0 = \sum_i c_i \boldsymbol{v}_i.$$

   What can you conclude about c_1?
   ii. Show that

$$\frac{\boldsymbol{u}_t}{||\boldsymbol{u}_t||} = \frac{\boldsymbol{B}^t \boldsymbol{u}_0}{||\boldsymbol{B}^t \boldsymbol{u}_0||}.$$

   iii. Show that

$$\frac{\boldsymbol{B}^t \boldsymbol{u}_0}{||\boldsymbol{B}^t \boldsymbol{u}_0||} = \frac{c_1 h_1^t}{|c_1 h_1^t|} \frac{\boldsymbol{v}_1 + \sum_{i\geq 2} \frac{c_i}{c_1} \left(\frac{h_i}{h_1}\right)^t \boldsymbol{v}_i}{||\boldsymbol{v}_1 + \sum_{i\geq 2} \frac{c_i}{c_1} \left(\frac{h_i}{h_1}\right)^t \boldsymbol{v}_i||}.$$

   iv. Take $t \to \infty$, what can you conclude?
(b) Show that in the proof of the convergence of the lazy walk, the conditions for applying the power method above are satisfied, and thus, it is a special case. In addition, justify why there is no need to normalized the vector $\boldsymbol{p}$ after each iteration in the lazy random walk.
   (Hint: The lazy random walk matrix is similar to a symmetric matrix.)
(c) Implement a function `PowerMethod(B)` that returns the largest (in absolute value) eigenvalue and the corresponding eigenvector. Provide a reasonable stopping criterion to the iterative procedure.
   (Hint: Understand why $||\boldsymbol{u}_t - \boldsymbol{u}_{t-1}|| < \epsilon$ is not appropriate.)
   For this exercise, you may not use any eigen-decomposition tool.
(d) Use `PowerMethod(B)` to implement a function `PowerMethod2(B)` that returns the second largest (in absolute value) eigenvalue and its corresponding eigenvector.

[2] https://en.wikipedia.org/wiki/Power_iteration

For this exercise, you may not use any eigen-decomposition tool.

(Hint: Theorem (***Wielandt Deflation***)

Let $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ be a matrix with eigenvalues $\{\lambda_i\}_{i=1}^n$ and corresponding eigenvectors $\{\boldsymbol{\psi}_i\}_{i=1}^n$. Assume that λ_1 has multiplicity 1. Let $\boldsymbol{x} \in \mathbb{R}^n$. If $\boldsymbol{x}^\top \boldsymbol{\psi}_1 = 1$, then the matrix $\boldsymbol{C} = \boldsymbol{B} - \lambda_1 \boldsymbol{v}_1^\top \boldsymbol{x}$ has eigenvalues $0, \lambda_2, \lambda_3, \dots, \lambda_n$ with corresponding eigenvectors $\boldsymbol{\psi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3, \dots, \boldsymbol{\phi}_n$ such that

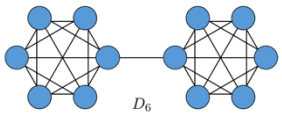
$$\boldsymbol{\psi}_i = (\lambda_i - \lambda_1)\boldsymbol{\phi}_i + \lambda_1(\boldsymbol{x}^\top \boldsymbol{\phi}_i)\boldsymbol{\psi}_1,$$

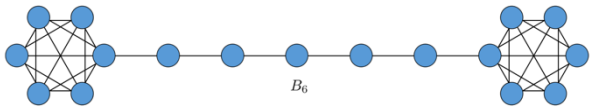
$\forall\, i \geq 2$.)

(e) Implement a program that verifies your implementations using a few. Use eigen-decomposition tools for getting the "ground truth".

As input, you may use random symmetric matrices[3] or the Laplacian matrices presented in class.

3. Consider a dumbbell graph D_n consisting of two complete graphs with n vertices, joined by one edge. That is, there are $2n$ vertices in total. Consider a Bolas graph B_n consisting of two complete graphs with n vertices connected by a path of length n. For example:





(a) Compute in code the adjacency matrix, degree matrix, the normalized Laplacian matrix and the lazy random walk matrix of D_{30} and B_{30}.

(b) Compute in code their eigen-decomposition of the normalized Laplacian matrix and order the obtained eigenvalues from the smallest to the largest. Plot the sorted eigenvalues of the normalized Laplacian matrix of D_{30} and B_{30} as a function of their order statistic. Is the largest eigenvalue tightly bounded by 2?

(c) Compute in code their eigen-decomposition of the lazy random walk matrix. Present the relation of the eigenpairs of the random walk matrix and the normalized Laplacian matrix (as shown in Recitation 5).

(d) For both D_{30} and B_{30}:

i. Set an initial mass distribution $\boldsymbol{p}_0$ as a delta function (concentrated at some vertex)

ii. Run the lazy walk until convergence

iii. Plot 6 intermediate states of $\boldsymbol{p}_t$ (including the initial and final states) with exponentially spaced time steps

iv. Demonstrate the convergence in log scale and compare in a plot the obtained **convergence rate** to the upper bound shown in class, that is, for all a, b and t, if $\boldsymbol{p}_0 = \boldsymbol{\delta}_a$, then

$$|\boldsymbol{p}_t(b) - \boldsymbol{\pi}(b)| \leq \sqrt{\frac{\boldsymbol{d}(b)}{\boldsymbol{d}(a)}} w_2^t,$$

where $\boldsymbol{\pi}$ is the stable distribution, $\boldsymbol{p}_t$ is the distribution at t step, $\boldsymbol{d}$ is the degree vector, and w_2 is the second largest eigenvalue of the lazy random walk matrix.

[3] https://stackoverflow.com/questions/10806790/generating-symmetric-matrices-in-numpy

5 Manifold Learning and Dimension Reduction (Wet)

In the following exercises, you will apply manifold learnings and dimensional reduction techniques to high-dimensional data, which is denoted by $\boldsymbol{Z} \in \mathbb{R}^{N\times m}$, where N is the number of samples and m is the number of features/dimensions.

1. Implement a function `AffinityMat(Z, kernel_method, n_neighbor, epsilon, …)` that returns an affinity matrix of size $N \times N$ and receives as input the kernel method (e.g., Gaussian, linear, etc) as a string, the number of the neighbors and the kernel hyperparameter. You may add in additional input arguments (e.g., normalization for the data).
2. Implement **two** of the following manifold learning and dimension reduction functions you saw in lecture 5 & 6 and recitation 6:
   - `Diffusion_Maps(n_dim, …)`
     that returns a low-dimensional representation from of size $N \times d$, where d is the new representation dimension
   - `Locally_Linear_Embedding(n_nei,n_dim,…)`
     that returns a low dimensional representation of size $N \times d$, where d is the new representation dimension
   - `Isometric_Mapping(n_nei,n_dim,…)`
     that returns a low-dimensional representation of size $N \times d$, where d is the new representation dimension
3. Apply the implemented manifold learning and dimensional reduction techniques to the following two examples. Compare and discuss the obtained results.
   (a) [Torus] Consider two independent and identically distributed random variables X and Y sampled uniformly in [0,1]. Generate from these variables 2000 pairs (x_i, y_i) and construct a three-dimensional vector via a non-linear function $f(x, y)$ such that

$$\boldsymbol{s}_i = f(x_i, y_i) = \begin{pmatrix} (R + r\cos(2\pi y_i))\cos(2\pi x_i) \\ (R + r\cos(2\pi y_i))\sin(2\pi x_i) \\ r\sin(2\pi y_i) \end{pmatrix} \in \mathbb{R}^3,$$

   where $R = 10$ and $r = 4$.
   i. Generate $\{\boldsymbol{s}_i\}_{i=1}^{2000}$.
   ii. Apply the implemented methods to $\{\boldsymbol{s}_i\}_{i=1}^{2000}$.
   Set the number of output dimensions `n_dim` to 2. Set the number of neighbors `n_nei` to 10. Plot the torus topology and color the points by their two coordinates of their low-dimensional representation. Plot the low-dimensional representation and color the points by $\{x_i\}_{i=1}^{2000}$ and $\{y_i\}_{i=1}^{2000}$. Choose two other values of `n_nei` to compare and understand the influence of `n_nei` on the result. Note that you should try to play with other input arguments in each function.
   (b) [Digits] Load the digits dataset from `sklearn.datasets.load_digits` and set `n_class` to 3, 5, and 7. That is, we have three datasets, one consisting of three classes, one of five classes and the last one of seven classes.
   i. Apply the implemented methods to all three datasets.
   Set the number of dimensions `n_dim` to 2. Set the number of neighbors `n_nei` to 10 and to two other values of your choice. Plot the low-dimensional representation obtained by each method and color the points by the digit labels. Compare the differences in the embeddings using different input parameters.

ii. **Bonus (5 points)**

- Classify the labels of the digits based on the obtained low-dimensional representation. You may like to use SVM classifiers[4] (other classifiers are accepted as well, but do not implement these yourselves). Quantify the classification results by k-fold cross-validation with a reasonable choice of k. Report the classification accuracy of each method.
- Load the digits dataset from `sklearn.datasets.load_digits` and set `n_class` to **10**. Apply the implemented manifold learning and dimensional reduction techniques to it. Classify the labels of the digits based on the obtained low-dimensional representation like above. Propose ways to improve the classification accuracy and test them. You may like to tune the parameters, for example, the kernel hyperparameter, the number of neighbors, the number of obtained dimensions, etc.

(c) **Bonus (10 points)**- ***Manifold learning with topology constraints***

i. Generate a swiss roll with topological constraints. It could be a circle or a rectangle or any other shapes. We denote the data as $\{\boldsymbol{x}_i\}_{i=1}^n$. For example, a swiss roll with a circle hole is shown in Fig 1.

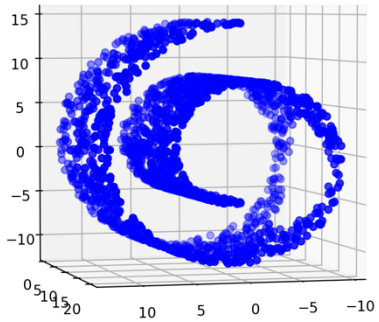


Fig 1

ii. Generate a noisy version of $\{\boldsymbol{x}_i\}_{i=1}^n$ with some Gaussian noise with 0 mean and unit variance. We denote the noisy version as $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^n$. An example can be found in Fig 2.

iii. Apply the implemented methods to $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^n$, respectively. Plot the obtained low-dimensional representation. Is it remaining a plane manifold? Is it with the topological constraints?

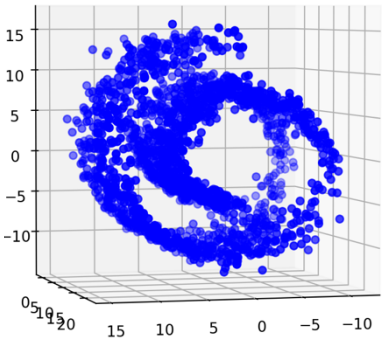


Fig 2

iv. Implement Topologically Constrained Isometric Embedding (TCIE)[5] to $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^n$, respectively.

Compare and discuss the obtained embedding with those in iii.

[4] https://scikit-learn.org/stable/modules/svm.html

[5] http://www.cs.technion.ac.il/~ron/PAPERS/RosBroBroKim_IJCV2010.pdf