Introduction to Graph-based Dimensionality Reduction

Christoph Luther

March 7, 2021

Seminar on Manifold Learning, MSc Statistics @LMU Munich

Table of Contents

Introduction

Motivation

- Abundance of high-dimensional data but difficult to interpret
- Find lower-dimensional embedding, e.g. helpful for exploratory analysis
- Makes assumption that high-dimensional data truly lie on lower-dimensional submanifold
- Classical methods: Linear map between original domain and embedding space (PCA, MDS)
- Graph-based methods introduce non-linearity $\rightarrow$ more flexible
- Isomap and Laplacian Eigenmaps (LEM)

Graphs and Graph Matrices

Graph Terminology

- Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Set of vertices $\mathcal{V} = \{v_1, ..., v_n\}$ represents input data $\mathbf{X} = \{x_1, ..., x_n\} \rightarrow v_i$ stands for a data point x_i and $|\mathcal{V}| = |\mathbf{X}| = n$
- Set of edges $\mathcal{E}$, where: $e_{i,j} \in \mathcal{E} \Leftrightarrow$ nodes v_i and v_j are adjacent [1]
- Set of weights for weighted graphs $\mathcal{W}$, where $w_{i,j} > 0 \Leftrightarrow e_{i,j} \in \mathcal{E}$
- Path: $\{v_k, ..., v_{k+m}\}$, such that for every two nodes v_l and v_{l+1}, $l \in \{k, ..., k+m-1\}$ it holds that $e_{l,l+1} \in \mathcal{E}$

[1] Equivalence only holds for undirected graphs, here we also do not consider directed graphs

Graph Terminology ctd.

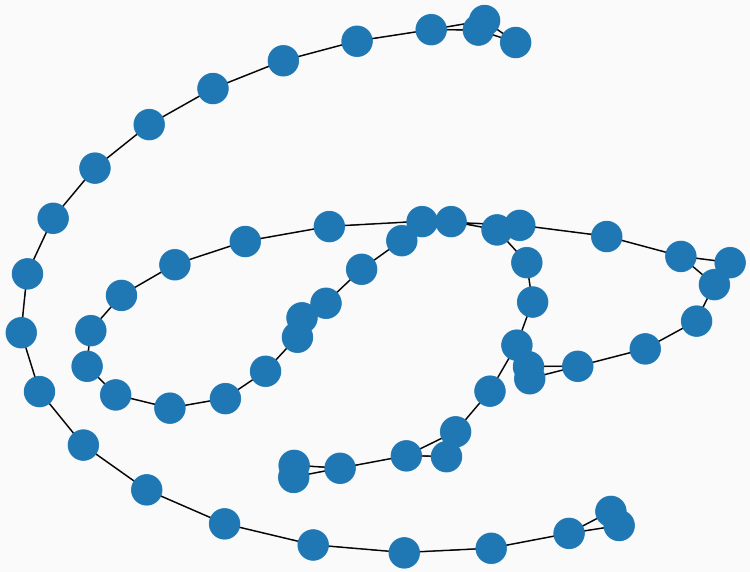
- Connected graph: For every node $v_i, i = 1, ..., n$, there is a path to every other node v_j, $j \in \{1, ..., n\} \setminus j$
- Neighbouring/adjacency graphs: Nodes representing close data points are adjacent or at least close
- Degree of a node in unweighted graph: $d(v_i) = |\{j : e_{i,j} \in \mathcal{E}\}|$
- Degree of a node in weighted graph: $d(v_i) = \sum_{j=1}^{n} w_{i,j}$

Vector-Valued Data and Graphs

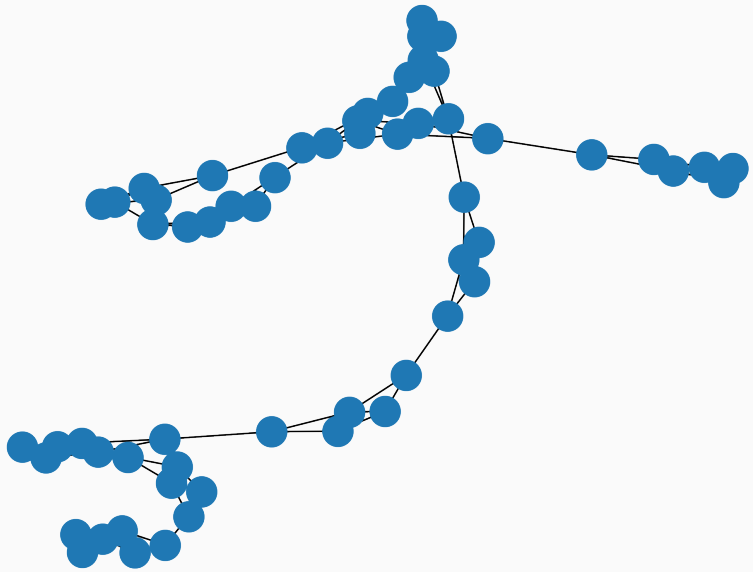
- Two ways to construct a graph from input data $\mathbf{X} = \{x_1, ..., x_n\}$, $x_i \in \mathbb{R}^D$ for all $i = 1, ..., n$
- Used in either method (Isomap, LEM)
- Via ϵ-**neighbourhood**: $e_{i,j} \in \mathcal{E} \Leftrightarrow d(x_i, x_j) < \epsilon$, where $d(\cdot)$ typically is Euclidean distance
- Via **k nearest neighbours** (kNN) $\rightarrow$ two options:
  1. $e_{i,j} \in \mathcal{E} \Leftrightarrow x_j \in kNN_i$[2]
  2. $e_{i,j} \in \mathcal{E} \Leftrightarrow x_j \in kNN_i \wedge x_i \in kNN_j$
- kNN graphs typically weighted, ϵ-neighbourhood graphs not (no additional info, because we know that distance is below threshold ϵ)
- kNN graphs more often able to ensure a connected graph

[2] kNN_i is set of k nearest neighbours of x_i

Example Graphs



2-NN Graph of Subset of Circle Data ($n = 60$) - 2 Connected Components



3-NN Graph of Subset of Circle Data ($n = 60$) - Connected Graph

Graph Matrices

Adjacency Matrix:

$$\mathbf{A}_{i,j} = \begin{cases} 1, & e_{i,j} \in \mathcal{E} \\ 0, & else \end{cases}$$

Weighted Adjacency Matrix [3]:

$$\mathbf{W}_{i,j} = \begin{cases} w_{i,j} > 0, & e_{i,j} \in \mathcal{E} \\ 0, & else \end{cases}$$

Degree Matrix:

$$\mathbf{D}_{i,j} = \begin{cases} d(v_i), & i = j \\ 0, & else \end{cases}$$

[3] Same as set of weights $\mathcal{W}$

Graph Laplacian

- Matrix that contains information about neighbourhood structure/interconnectedness of a graph
- Link between discrete graph and continuous representation (here, the latter is the embedding of the original data)
- Unnormalized Laplacian (for unweighted graph): $\mathbf{L} = \mathbf{D} - \mathbf{A}$
- Unnormalized Laplacian (for weighted graph): $\mathbf{L} = \mathbf{D} - \mathbf{W}$

Laplacian Eigenmaps

Overview

- *Local* method for non-linear dimensionality reduction, i.e. only preserves local geometry (unlike *global* methods like Isomap)
- Locality allows for almost arbitrary curvature of the learned manifold, since only Euclidean 'patches' are tried to be inferred correctly
- Faster than *global* methods as it involves sparse matrix operations
- Natural link to clustering due to locality
- Insensitive to outliers through focus on local geometry

Laplacian Eigenmaps

- Input $\mathbf{X} = \{x_1, ..., x_n\}$, $x_i \in \mathbb{R}^D$; output $\mathbf{Y} = \{y_1, ..., y_n\}$, $y_i \in \mathbb{R}^d$, $d << D$
- Basically LEM minimizes the following sum with respect to $\mathbf{Y}$:

$$E_{LEM} = \sum_{i,j=1}^{n} ||y_i - y_j||_2^2 \cdot w_{i,j}$$

- Recommended choice for $w_{i,j}$ is:

$$w_{i,j} = \begin{cases} exp(-\frac{||x_i - x_j||_2^2}{t}), & e_{i,j} \in \mathcal{E} \\ 0, & else \end{cases} \tag{1}$$

- Weight highlights locality $\rightarrow$ the closer two input vectors, the higher the penalty for distant embedding vectors
- $t = \infty$ makes weight binary (no need for hyperparameter tuning)

Laplacian Eigenmaps ctd.

- Crux of LEM: Rewrite minimization problem to sparse eigenvalue problem of graph Laplacian
- Obtain ϵ-neighbourhood or kNN-graph from data and add weights as in E_{LEM}
- Assume that the graph is connected (else you can treat every connected component separately $\rightarrow$ not covered here)
- Compute graph Laplacian $\mathbf{L}_{LEM} = \mathbf{D} - \mathbf{W}$ for this graph, where $\mathbf{W}$ consists of the weights as in equation (1) (other choices possible)

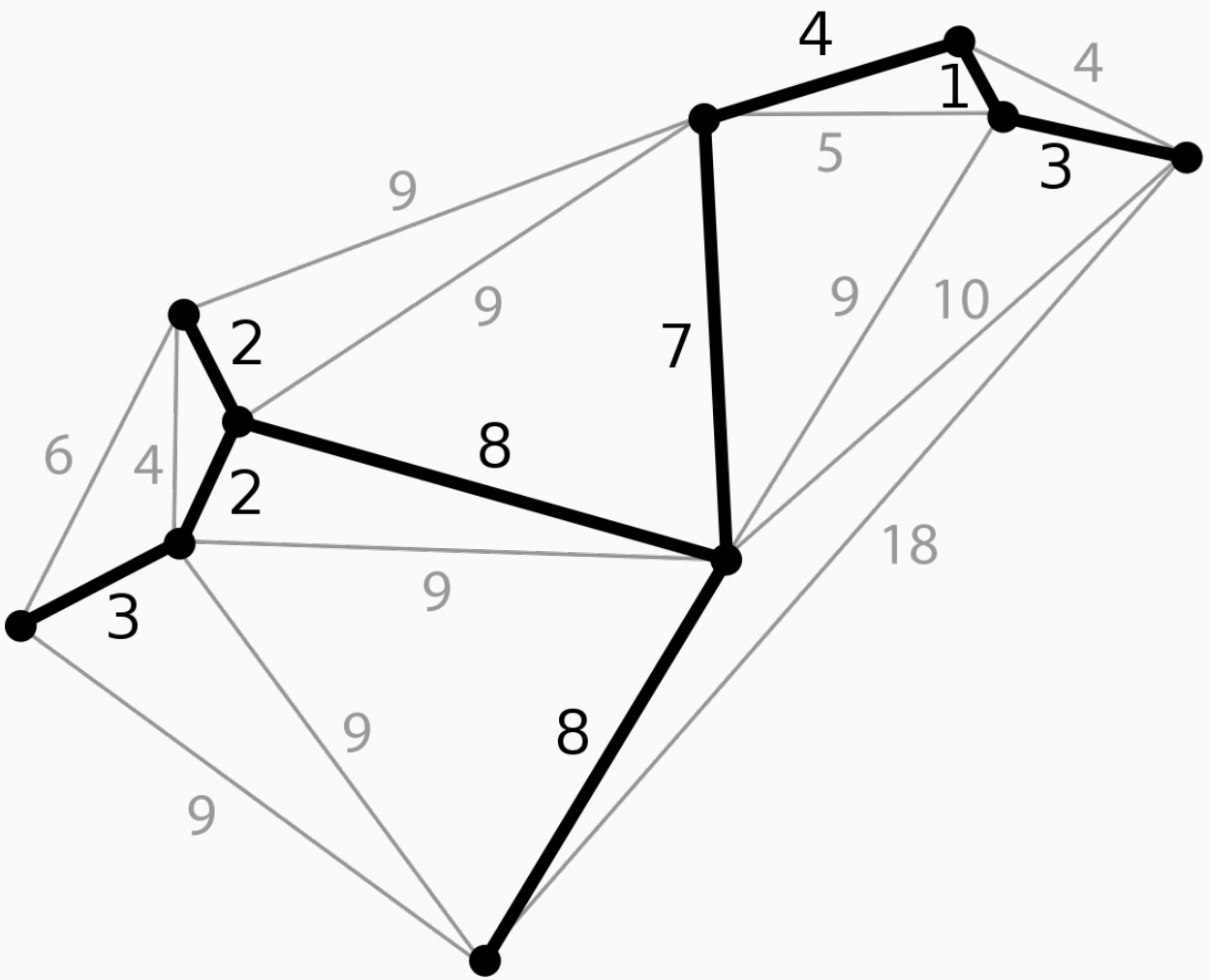
Laplacian Eigenmaps ctd.

- Compute eigenvectors of generalized eigenvector problem: $\mathbf{L}_{LEM}\mathbf{u} = \lambda\mathbf{D}\mathbf{u}$
- Let $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{n-1}$ be the eigenvalues and $\mathbf{u}_0, ..., \mathbf{u}_{n-1}$ the corresponding eigenvectors, such that $\mathbf{L}_{LEM}\mathbf{u}_i = \lambda_i\mathbf{D}\mathbf{u}_i$ for $i = 0, ..., n-1$
- Now the d eigenvectors corresponding to the d smallest non-zero eigenvalues, i.e. $\mathbf{u}_1, ..., \mathbf{u}_d$, span the embedding space
- The embedding itself is $x_i \mapsto (\mathbf{u}_1(i), ..., \mathbf{u}_d(i))$

Robust Laplacian Eigenmaps

- For LEM, connected graph was assumed for simplicity and even though separate connected components do not render dimension reduction infeasible, the results may be more difficult to interpret
- This is mitigated by the inclusion of global information
- A **Minimum Spanning Tree** (MST) can capture the *global* structure of the data
- MST: Subset of edges of connected graphs that connects all vertices avoiding cycles and with minimum combined edge weight

Minimum Spanning Tree



Example for Minimum Spanning Tree from Wikipedia

Robust Laplacian Eigenmaps ctd.

- Build two graphs: One as in LEM and a Minimum Spanning Tree
- Instead of minimizing E_{LEM}, minimize:

$$E_{GLEM} = \sum_{i,j=1}^{n} ||y_i - y_j||_2^2 \cdot w_{i,j}^{LEM} w_{i,j}^{MST}$$

  where $w_{i,j}^{LEM}$ is the weight as in LEM and $w_{i,j}^{MST}$ the weight from the MST (the weight function is the same, but the set of edges is different)

- Algorithm is very similar to LEM but uses the *combined* Laplacian of the normal LEM graph and the MST
- If neighbourhood information is restricted (many small connected components in graph), MSTs help to recover the continuity of the original manifold

Isomap

Overview

- *Global* method for dimensionality reduction, i.e. tries to preserve the global structure, e.g. also get distance of faraway points right
- Makes use of Multidimensional Scaling
- Slower than *local* methods
- Can be sped up by focusing on so called landmark points in the graph

Multidimensional Scaling

Algorithm 1: Multidimensional Scaling

Input: Data $\mathbf{X} = \{x_1, ..., x_n\}$, $x_i \in \mathbb{R}^D$

Output: Embeddings $\mathbf{Y} = \{y_1, ..., y_n\}$, $y_i \in \mathbb{R}^d$, $d << D$

Steps:

1. Create $n \times n$ matrix of dissimilarities of data points: $\mathbf{D}_{MDS} \in \mathbb{R}$
2. From $\mathbf{D}_{MDS}$ calculate $\mathbf{A}$, where $a_{i,j} = -\frac{1}{2} d_{i,j}^2$ and $d_{i,j} = \left[\mathbf{D}_{MDS}\right]_{i,j}$
3. From $\mathbf{A}$ calculate $\mathbf{B}$, where $b_{i,j} = a_{i,j} - a_{i\cdot} - a_{\cdot j} + \bar{a}$ and $a_{i\cdot}$ is the average of the i-the row, $a_{\cdot j}$ of the j-th column and $\bar{a}$ of all entries of $\mathbf{A}$
4. Let $\mathbf{U} \in \mathbb{R}^{n \times d}$ be the matrix of normalized eigenvectors $\mathbf{u}_0, ..., \mathbf{u}_{d-1}$ corresponding to the d largest eigenvalues of $\mathbf{B}$, i.e. $\mathbf{u}_i' \mathbf{u}_i = \lambda_i$
5. y_i is i-th row of $\mathbf{U}$

Isomap

Algorithm 2: Isomap

Input: Data $\mathbf{X} = \{x_1, ..., x_n\}$, $x_i \in \mathbb{R}^D$

Output: Embeddings $\mathbf{Y} = \{y_1, ..., y_n\}$, $y_i \in \mathbb{R}^d$, $d << D$

Steps:

1. Obtain ϵ-neigbourhood or kNN-graph from data
2. Assign weights $d(x_i, x_j)$ to each edge $e_{i,j} \in \mathcal{E}$, where $d(\cdot)$ is Euclidean distance
3. Approximate geodesics of each data pair (x_i, x_j) with shortest path distances $d_{\mathcal{G}}(x_i, x_j)$
4. Apply classical MDS to matrix of shortest path distances $D_{\mathcal{G}}$

Shortest Path Distance

Algorithm 3: Shortest Path Distances

Input: Weighted Graph

Output: Matrix of shortest path distances of every pair of nodes $D_{\mathcal{G}}$

Steps:

1. Initiate $d_{\mathcal{G}}(v_i, v_j)$ as $d(x_i, x_j)$ for $(i,j) \in \{(i,j) : e_{i,j} \in \mathcal{E}\}$, else set $d_{\mathcal{G}}(v_i, v_j) = +\infty$
2. Iteratively replace all path distances according to $d_{\mathcal{G}}(v_i, v_j) = min\{d_{\mathcal{G}}(v_i, v_j), d_{\mathcal{G}}(v_i, v_k) + d_{\mathcal{G}}(v_k, v_j)\}$ for $k = 1, ..., n$
3. This generates the matrix $D_{\mathcal{G}}$ with $[D_{\mathcal{G}}]_{i,j} = d_{\mathcal{G}}(v_i, v_j)$

- If $n \to \infty$: $d_{\mathcal{G}}(v_i, v_k)$ converges to true geodesic of x_i and x_j

Isomap with Landmark Points

- Only use so called landmark points of the graph instead of all nodes, which leads to sparser matrices and thus faster computation
- Designate m points as landmark points, $m << n$
- Obtain matrix of shortest path distances of the landmark points to all other points: $D_{\mathcal{G},Landmark} \in \mathbb{R}^{m \times n}$
- Apply so called Landmark MDS to $D_{\mathcal{G},Landmark}$ which is substantially smaller than $D_{\mathcal{G}}$
- Landmark points can be chosen randomly
- Silva and Tenenbaum (2003) show that this does not degrade performance of Isomap too drastically

Application

Overview

- Application of LEM, MDS and Isomap to three datasets including the standard Swiss Roll data
- kNN-graphs only (especially for circle data, the graphs were only connected for kNN with a large k)
- As distance measure Euclidean distance was used throughout
- Focus on LEM vs Isomap

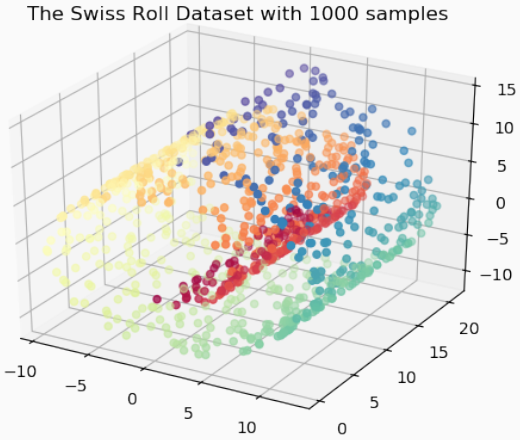
LEM vs Isomap - Summarized

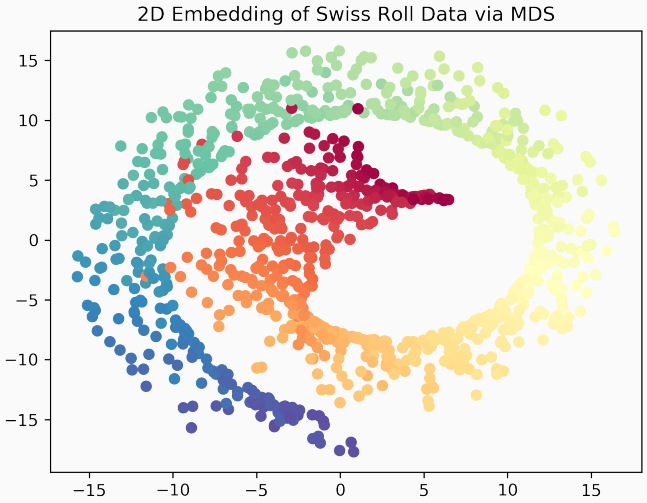
- LEM is faster than Isomap (but for the data at hand computation time was negligible)
- LEM can capture almost arbitrary curvature because of focus on getting local, approximately Euclidean patches right
- Isomap delivers more faithful picture of global structure, i.e. also the distances of faraway points are interpretable
- LEM has the additional link to clustering

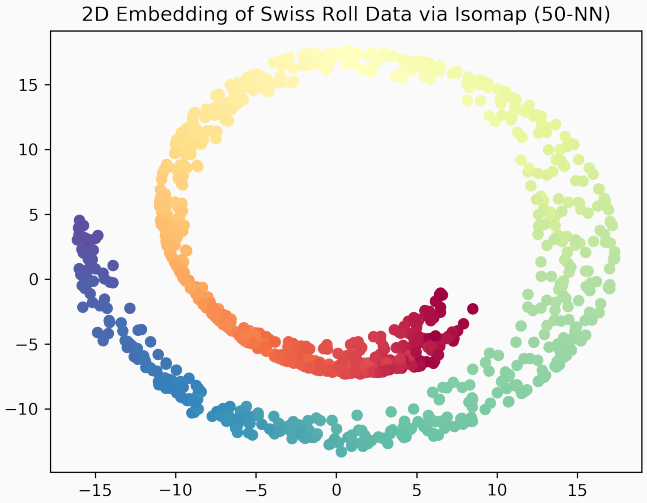
Swiss Roll Data

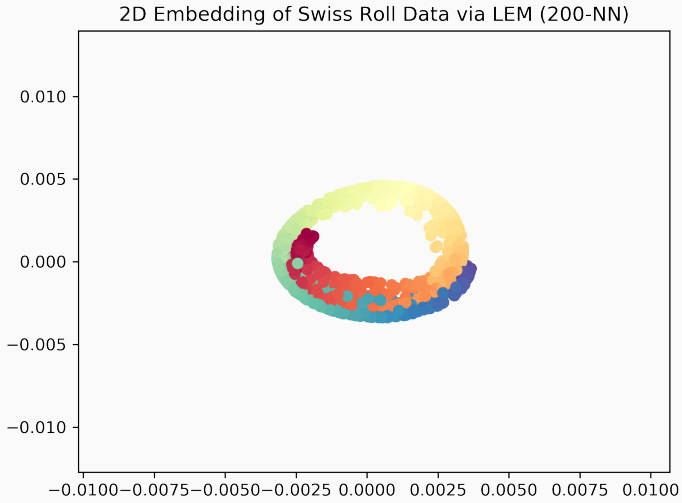
- Standard data set for dimensionality reduction: 3D to 2D
- $n = 1000$
- Either method is expected to perform well
- LEM worked better with particularly high number of neighbours (but results for fewer neighbours still reasonable)

Results - Swiss Roll Data





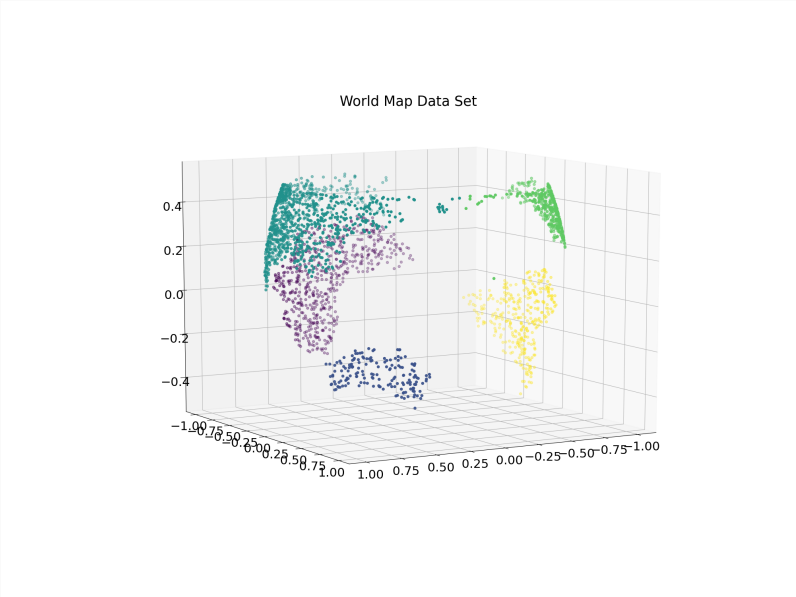


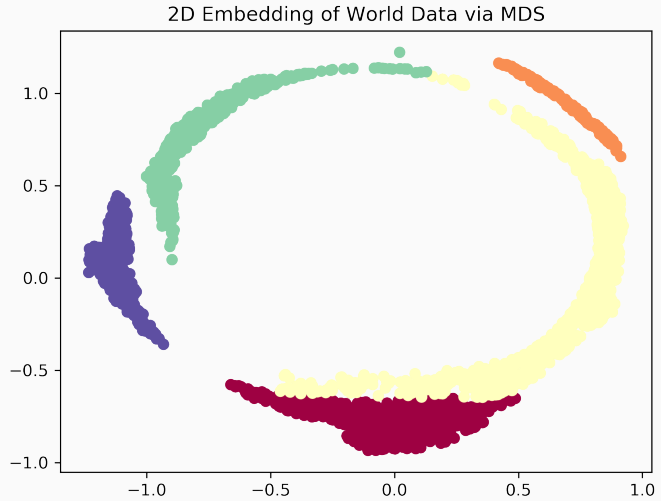


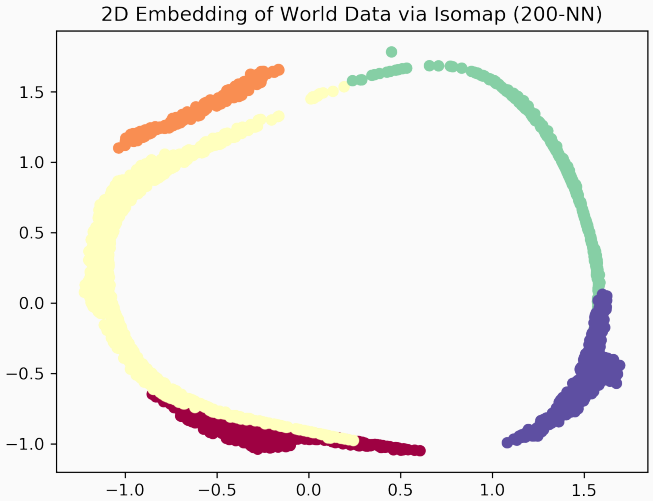
World Data

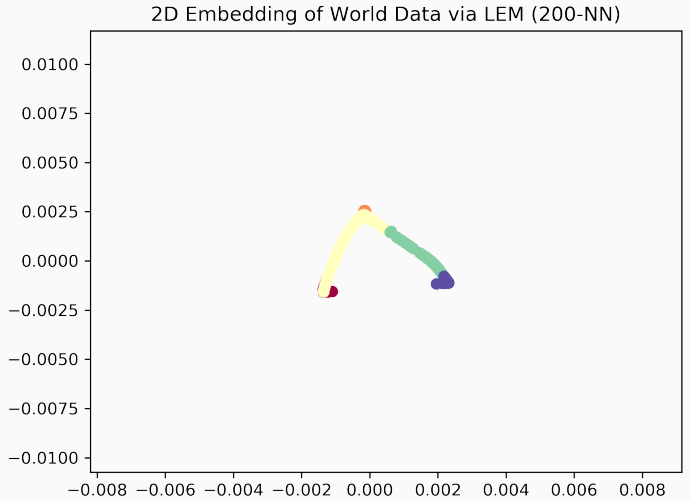
- 3D to 2D
- $n = 2527$
- MDS seems to work reasonably well
- For small number of neighbours Isomap finds a parabola and for increasing number of neighbours (40 and higher) it finds shapes as below, yielding a reasonbale representation for a very high number of neighbours
- LEM does not find continuous manifold for small number of neighbours

Results - World Data

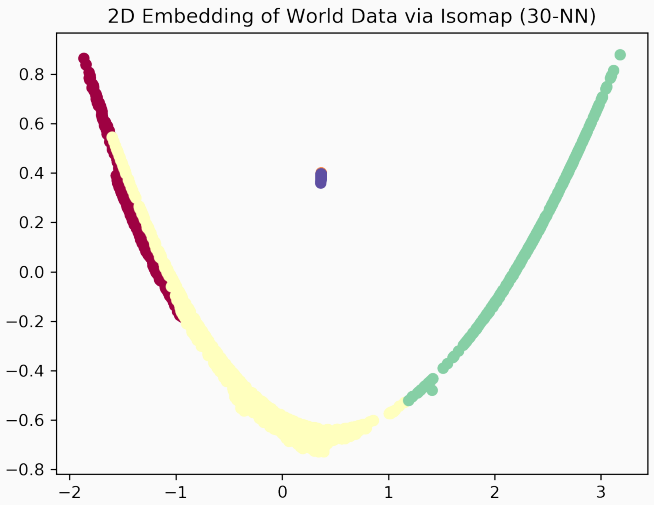


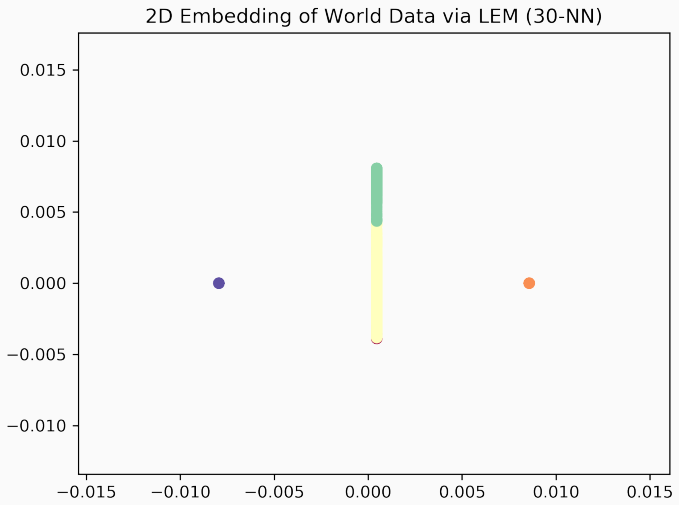


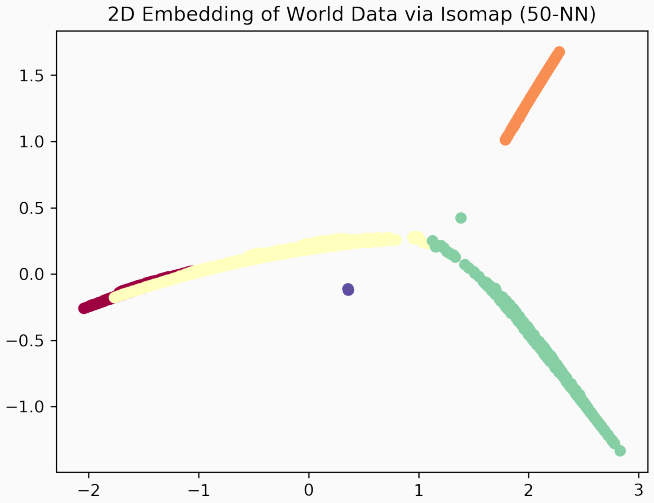


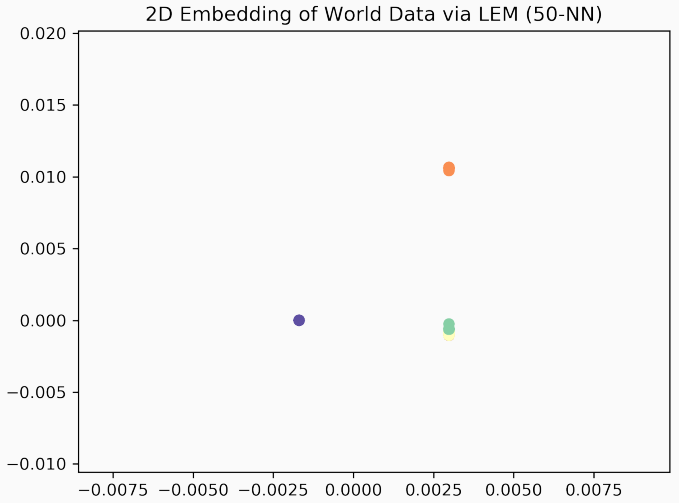


Results - World Data



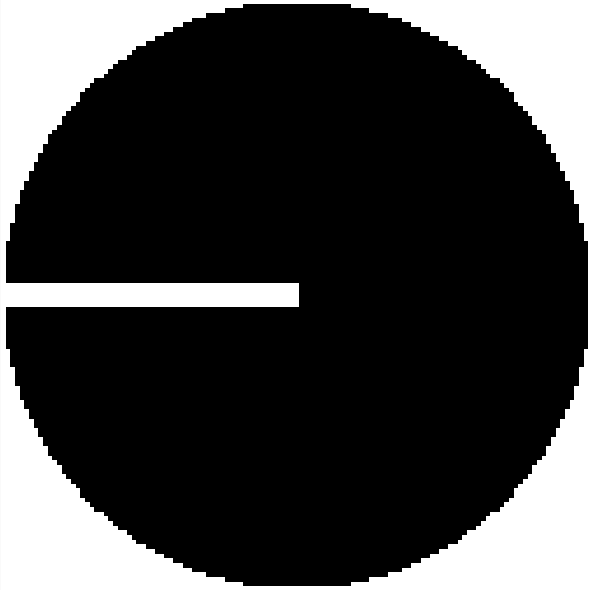




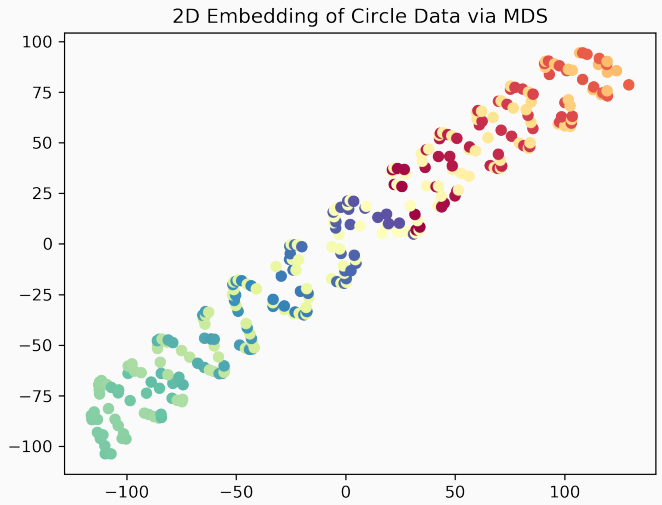


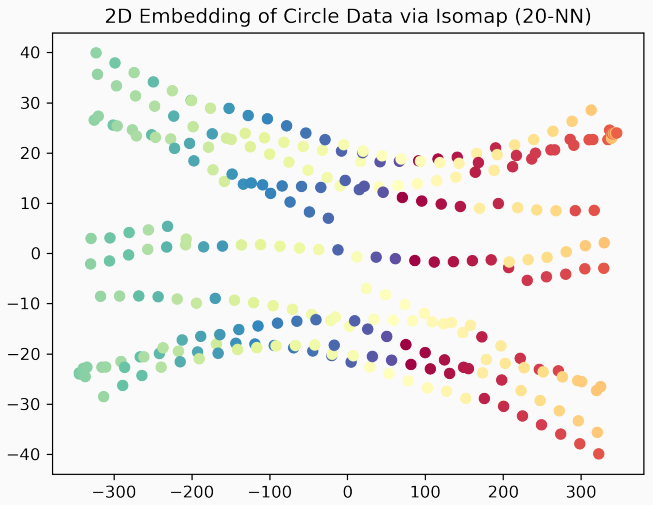
Circle Data

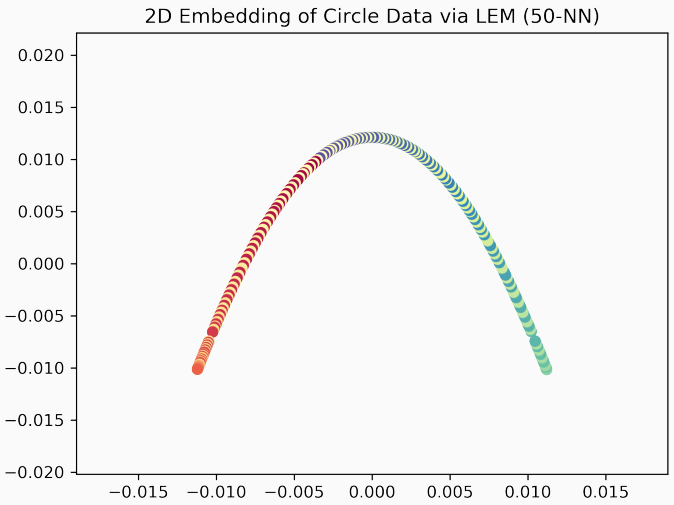
- Represents position of clock hands
- $\sim 16k$ features to 2D and 3D
- $n = 300$
- Same results when applying LEM with varying number of neighbours in small range (but n is only 300!), changes for 200 neighbours (close to n)
- Varying results when applying Isomap

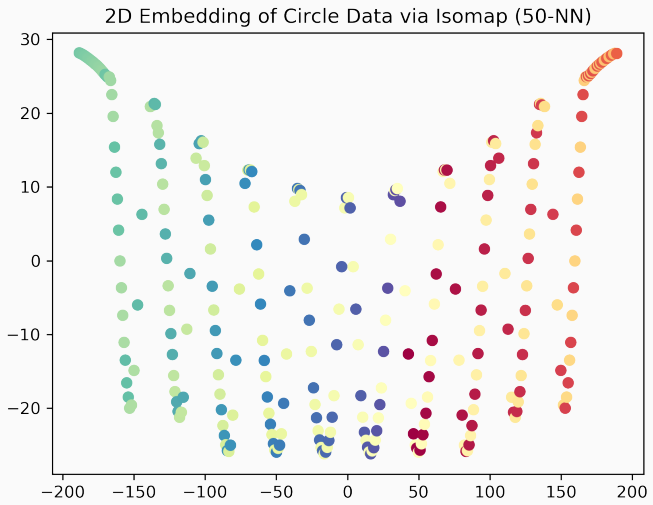


2D Results - Circle Data

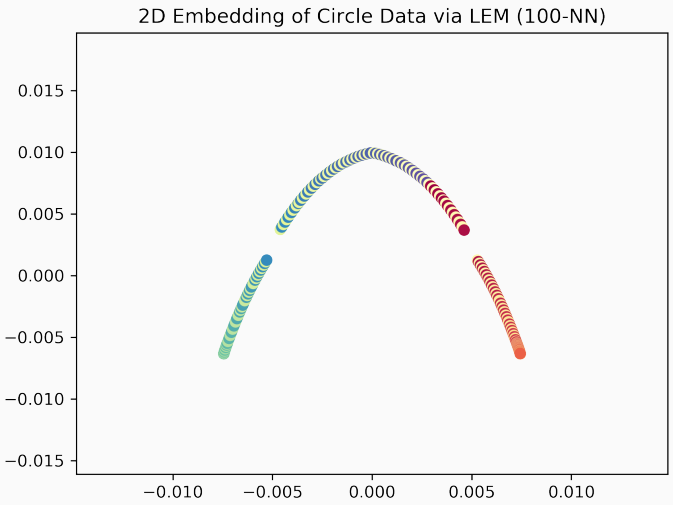


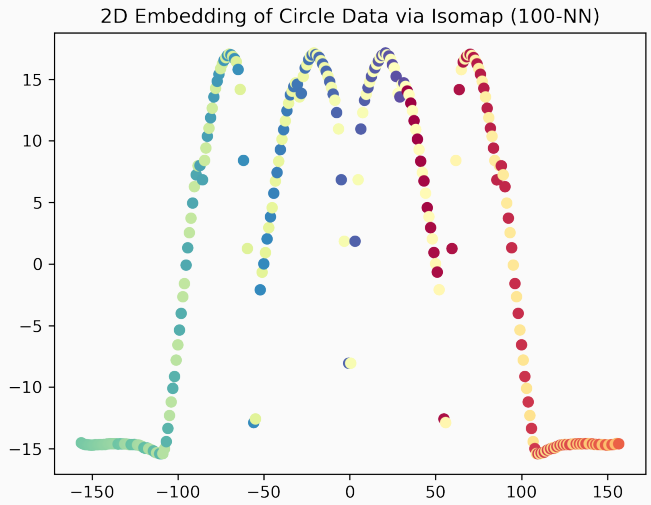


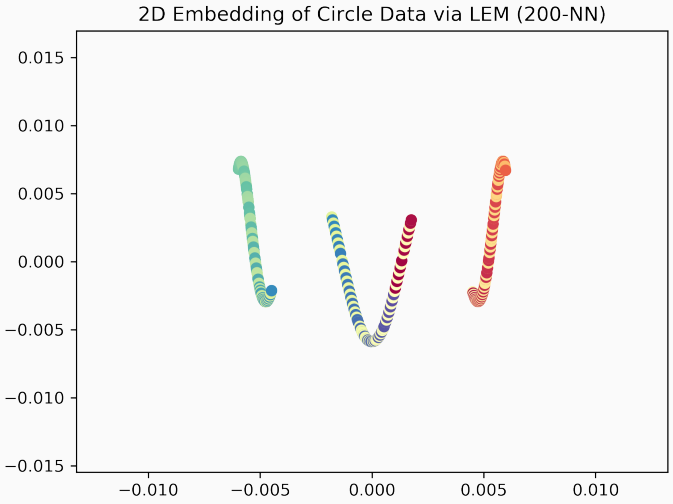


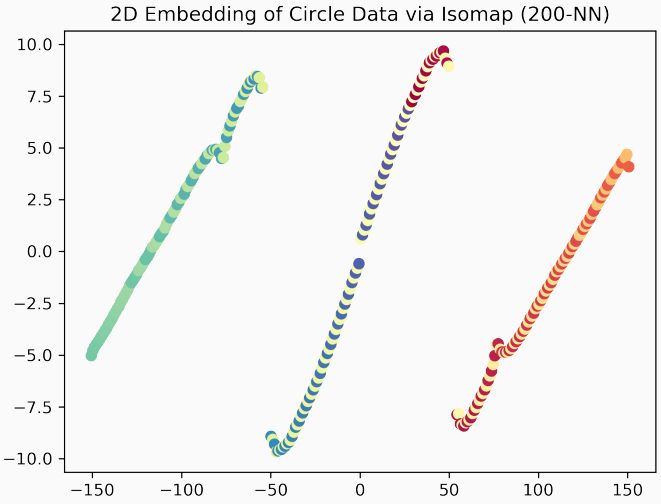


2D Results - Circle Data

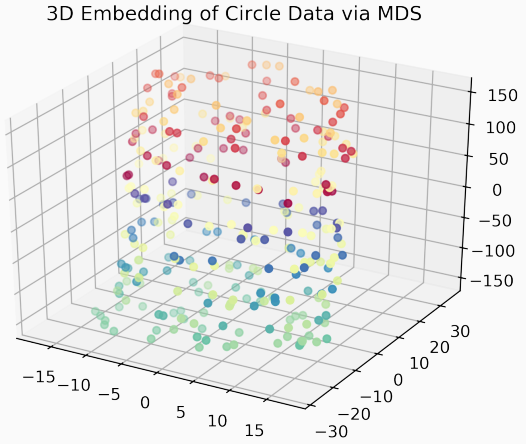


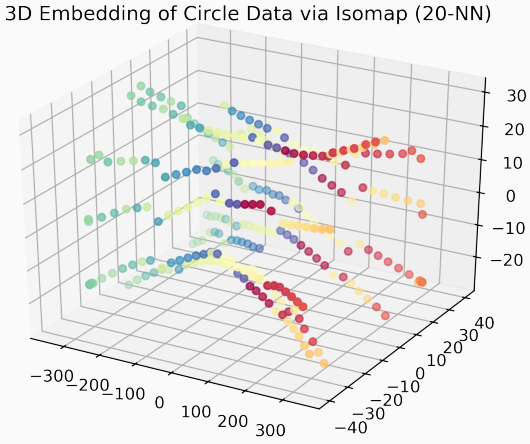


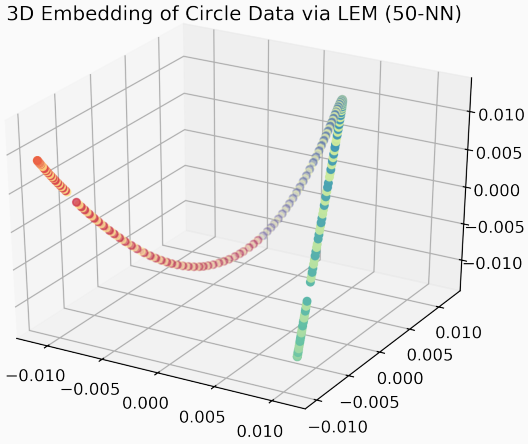


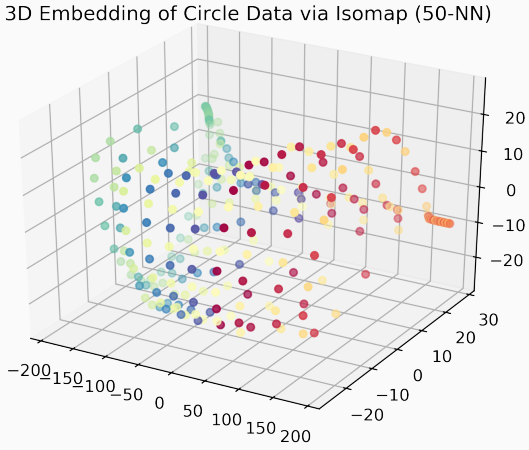


3D Results - Circle Data

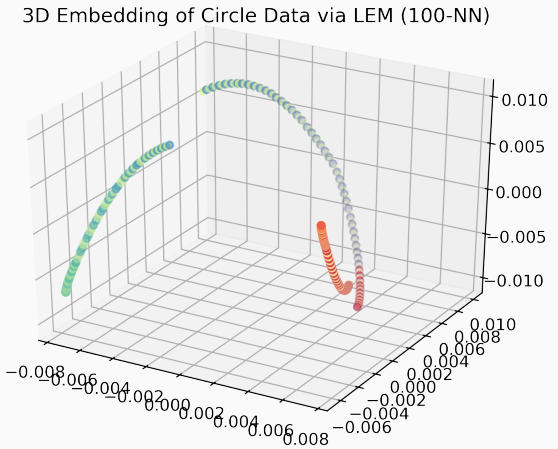


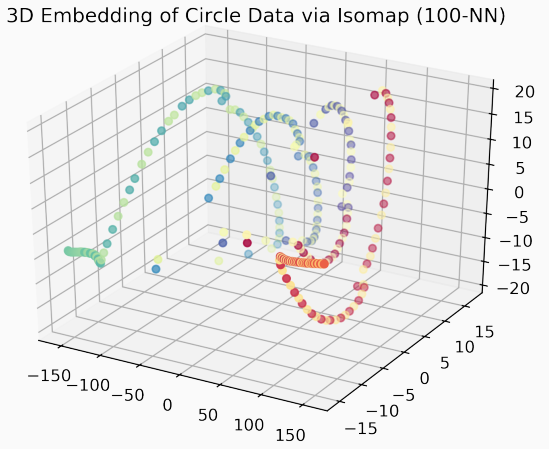


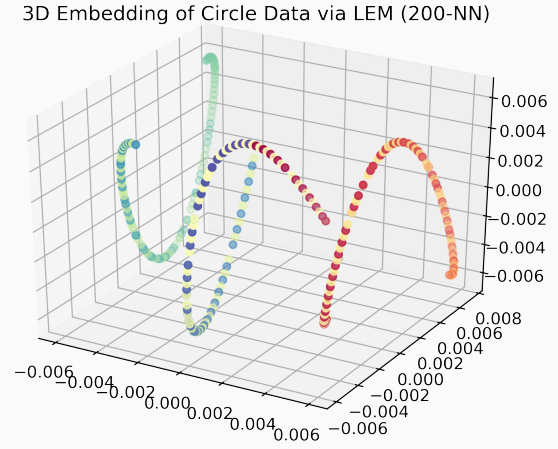


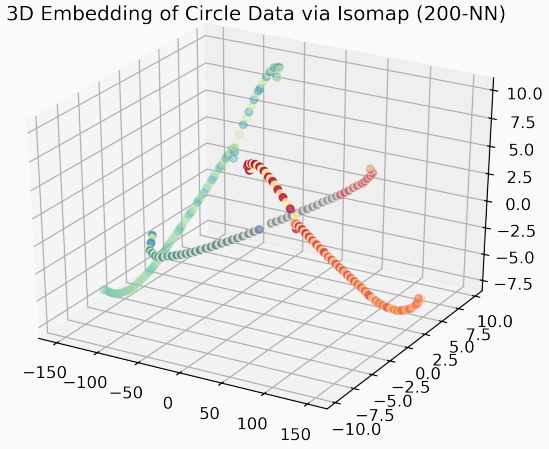


3D Results - Circle Data









Conclusion

Conclusion

- Non-linear methods are more flexible even though computationally more costly
- Hence, if map between high- and low-dimensional space is (assumed to be) approximately linear, linear methods also sufficient (cf. MDS for World data)
- LEM is fast and has a natural connection to clustering
- Global LEM can mitigate problems of insufficient neighbourhood information
- Results of global methods are more faithful to overall geometry of structure but computationally more expensive
- Landmark points in Isomap can mitigate the latter by speeding up the process

Questions?

Thanks for watching :)

References

Belkin, Mikhail and Partha Niyogi. *Laplacian Eigenmaps for Dimensionality Reduction and Data Representation*. Neural Computation, 15(6), pp. 1373 – 1396. 2002.

von Luxburg, Ulrike. *A Tutorial on Spectral Clustering*. Statictics and Computing, 17(4). 2007

Ng, Andrew Y., Michael I. Jordan and Yair Weiss. *On spectral clustering: analysis and an algorithm*. Advances in Neural Information Processing Systems, 14,pp. 849 – 856. 2002.

Roychowdhury, Shounak and Joydeep Ghosh. *Robust Laplacian Eigenmaps Using Global Information*. Proceedings of AAAI Fall Symposium on Manifold Learning and Its Applications. 2009.

Shi, Jianbo and Jitendra Malik. *Normalized cuts and image segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (8), pp. 888 – 905. 2000.

References

de Silva, Vin and Joshua B. Tenenbaum. *Global versus local methods in nonlinear dimensionality reduction*. Advances in Neural Information Processing Systems, 15, pp. 721–728. 2003.

Tenenbaum, Joshua B., Vin de Silva and John C. Langford. *A Global Geometric Framework for Nonlinear Dimensionality Reduction*. Science, 290, pp. 2319-2323. 2000.

Wickelmaier, Florian. *An Introduction to MDS*. Sound and Quality Research Unit at Aalborg University. 2003.

https://en.wikipedia.org/wiki/Minimum$_s$*panning*$_t$*ree*, Last accessed on Feb 28, 2021.