FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Lecture Pattern Analysis

Part 14: Multidimensional Scaling

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2022

Introduction

- Multidimensional Scaling (MDS) is equivalent to PCA, but operates on distances between samples
- This variant may be useful if only relative information about samples is available, e.g., from a learned similarity measure
- MDS operates on a distance matrix

$$D^2 = [d_{ij}^2] \text{ where } 1 \leq i, j \leq N, \tag{1}$$

and

$$d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}(\mathbf{x}_i - \mathbf{x}_j) \ . \tag{2}$$

- However, the absolute locations $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ are the unknown quantity
- Ultimately, we seek to reconstruct $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N)^\top \in \mathbb{R}^{N \times d'}$, where $d' \leq d$ is an orthogonal projection onto a d'-dimensional subspace
- Since we can not distinguish spatial translations of $\mathbf{X}$, we require the unique solution $\mathbf{X}$ to be mean-free, i.e., its mean value is 0

Linking Distances to Coordinates

- The components of the distance matrix can be written in matrix notation

$$d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}(\mathbf{x}_i - \mathbf{x}_j) \tag{3}$$

$$= \mathbf{x}_i^\mathsf{T}\mathbf{x}_i + \mathbf{x}_j^\mathsf{T}\mathbf{x}_j - 2\mathbf{x}_i^\mathsf{T}\mathbf{x}_j \tag{4}$$

$$\Rightarrow D^2 = \text{diag}(\mathbf{X}^\mathsf{T}\mathbf{X}) \cdot \mathbb{1}\mathbb{1}^\mathsf{T} + \mathbb{1}\mathbb{1}^\mathsf{T} \cdot \text{diag}(\mathbf{X}^\mathsf{T}\mathbf{X})^\mathsf{T} - 2\mathbf{X}^\mathsf{T}\mathbf{X} \tag{5}$$

  where $\mathbb{1} = (1, \ldots, 1)^\mathsf{T} \in \mathbb{R}^N$ and $\text{diag}(\mathbf{X})$ is a matrix with the diagonal elements of $\mathbf{X}$ and 0 everywhere else

- This equation links the distances to the actual coordinates
- $\mathbf{X}$ can be recovered after multiplying $\mathbf{D}^2$ by a centering matrix $\mathbf{C}$ with

$$\mathbf{C} = (\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}) \tag{6}$$

  where $\mathbf{I}$ is the $N \times N$ identity matrix

Multiplication by the Centering Matrix

- The multiplication itself is somewhat messy, but not difficult. The equation is

$$-\frac{1}{2}\mathbf{C}\mathbf{D}^2\mathbf{C} = -\frac{1}{2}\left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right) \cdot \tag{7}$$

$$\left(\underbrace{\operatorname{diag}\left(\mathbf{X}^\mathsf{T}\mathbf{X}\right) \cdot \mathbb{1}\mathbb{1}^\mathsf{T}}_{(1)} + \underbrace{\mathbb{1}\mathbb{1}^\mathsf{T} \cdot \operatorname{diag}\left(\mathbf{X}^\mathsf{T}\mathbf{X}\right)^\mathsf{T}}_{(2)} - \underbrace{2\mathbf{X}^\mathsf{T}\mathbf{X}}_{(3)}\right)\left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right) \tag{8}$$

- The sum in the middle has three elements, we will look at them one by one
- Element (1):

$$\left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right)\left(\operatorname{diag}\left(\mathbf{X}^\mathsf{T}\mathbf{X}\right) \cdot \mathbb{1}\mathbb{1}^\mathsf{T}\right)\left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right) \tag{9}$$

$$= \left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right)\operatorname{diag}\left(\mathbf{X}^\mathsf{T}\mathbf{X}\right) \cdot \left(\mathbb{1}\mathbb{1}^\mathsf{T} \cdot \mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T} \cdot \mathbb{1} \cdot \mathbb{1}^\mathsf{T}\right) \tag{10}$$

Solving Elements (1), (2) and (3)

- Element (1) evaluates to 0, since the rightmost brackets are 0:

$$\mathbb{1}^\mathsf{T}\mathbb{1} = N \;,\; \text{hence } \frac{1}{N}\mathbb{1}^\mathsf{T}\mathbb{1} = 1 \;,\; \text{and finally } \mathbb{1}\mathbb{1}^\mathsf{T} - \mathbb{1}\mathbb{1}^\mathsf{T} = \mathbf{0} \;. \tag{11}$$

- With the analogous calculation, element (2) also becomes 0
- Hence, if there will be anything interesting, it must be in element (3):

$$-\frac{1}{2}\left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right) \cdot \left(-2\mathbf{X}^\mathsf{T}\mathbf{X}\right) \left(\mathbf{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right) \tag{12}$$

$$= \left(\mathbf{I}\cdot\mathbf{X}^\mathsf{T} - \underbrace{\frac{1}{N}\mathbb{1}\cdot\mathbb{1}^\mathsf{T}\cdot\mathbf{X}^\mathsf{T}}_{=0 \text{ (zero mean!)}}\right) \cdot \left(\mathbf{X}\cdot\mathbf{I} - \underbrace{\frac{1}{N}\mathbf{X}\cdot\mathbb{1}\cdot\mathbb{1}^\mathsf{T}}_{=0 \text{ (zero mean!)}}\right) \tag{13}$$

$$= \mathbf{X}^\mathsf{T}\mathbf{X} \tag{14}$$

Obtaining the Coordinates

- To summarize, we showed that

$$-\frac{1}{2}\mathbf{C}\mathbf{D}^2\mathbf{C} = \mathbf{X}^\mathsf{T}\mathbf{X} \tag{15}$$

- Hence, $\mathbf{X}$ can be obtained via singular value decomposition (SVD),

$$\mathrm{SVD}\left(-\frac{1}{2}\mathbf{C}\mathbf{D}^2\mathbf{C}\right) = \mathrm{SVD}\left(\mathbf{X}^\mathsf{T}\mathbf{X}\right) = \mathbf{U}^\mathsf{T}\Sigma\mathbf{V} \tag{16}$$

$$\Rightarrow \mathbf{X} = \Sigma^{\frac{1}{2}}\mathbf{V} \tag{17}$$

- Σ is a diagonal matrix, so its square root is the square root of its elements
- Analogously to PCA, we could now also select the number of dimensions d' from the magnitudes of the singular values
- The orthogonal projection to $d' < d$ components can be done by setting all singular values in Σ beyond the first d' dimensions to 0

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

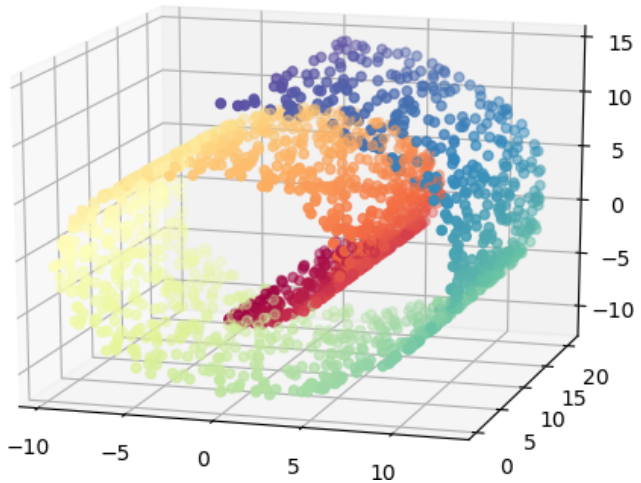
Lecture Pattern Analysis

Part 15: Isometric Feature Mapping (ISOMAP)

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2022

Introduction

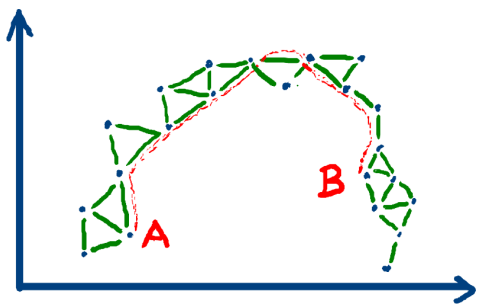
- PCA and MDS perform linear projections to transform the data
- Non-linear manifolds require a non-linear mapping
- One popular (academic) example for a non-linear manifold is the swiss roll:



- An orthogonal projection as done by PCA or MDS would “squash” the roll
- There are many non-linear manifold learning methods, e.g., “Sammon Mapping” and “Locally Linear Embedding” (LLE), and the modern t-SNE
- Key to these approaches is the MDS concept of distances between samples
- In Pattern Analysis, we look into ISOMAP and Laplacian Eigenmaps

ISOMAP

- ISOMAP is a straightforward “non-linearity hack” for MDS
- It preserves a non-linear manifold by exchanging the Euclidean distances of MDS by “geodesic” distances on the manifold
- More specifically, distances are calculated as shortest paths from a graph where each sample is connected to its nearest neighbors



- Algorithm:
  1. Define a graph where the edge weights are Euclidean distances to nearest samples (using k-NN or fixed distance threshold)
  2. The distance matrix are all-pairs shortest paths between the samples (e.g., via Floyd-Warshall algorithm or repeated use of Dijkstra’s algorithm)
  3. Perform MDS on the distance matrix

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Lecture Pattern Analysis

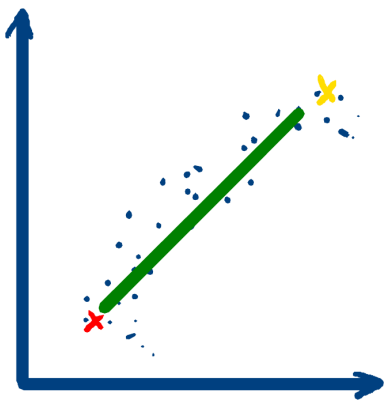
Part 16: Laplacian Eigenmaps (LE)

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2022

Introduction

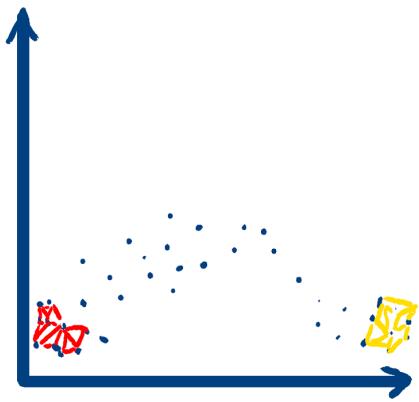
- Laplacian Eigenmaps (LE) advance the use of local distances one step further
- Instead of preserving the global variance (like PCA, MDS), LE aims to preserve local neighborhoods:

PCA & MDS:



"Large distances remain large"

LE:



"Local neighborhoods remain together"

- LE organizes local neighborhoods in a **Graph Laplacian**, which is a general-purpose matrix representation of graphs
- For example, this makes LE more robust to noise than ISOMAP

LE Algorithm

- The algorithm is very straightforward:
  1. Build adjacency graph from samples (e.g., via k-NN or fixed distance threshold)
  2. Weight the edges of the graph by the sample similarity, e.g., using a kernel. In this context, such similarities are called **affinities**
  3. Perform an eigendecomposition of the graph Laplacian
  4. Project samples into lower-dimensional space
- Note that the first two steps can be replaced by learned edge weights
- For example, we can use a density forest
  1. Train a density forest
  2. Calculate the sample affinity, for example the relative frequency that both samples end up in the same leaf node of the trees
- Analogous to our discussion on density estimation, learned weights may bring the advantage that the similarity measure better adapts to the data

Derivation of the LE Algorithm: Objective Function

- We consider the task of mapping a weighted graph G onto a line ($d' = 1$)
- The generalization to arbitrary d' is not too difficult, but omitted here
- Objective function:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}(x'_i - x'_j)^2 w_{ij} \rightarrow \min \tag{1}$$

where $x'_i, x'_j \in \mathbb{R}^{d'}$ and w_{ij} is an edge weight, e.g. from the heat kernel

$$w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{t}\right) \tag{2}$$

calculated in the high-dimensional space(!), i.e., with $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$

- Trivial solution $x'_1 = \ldots = x'_N$ is prevented with an additional constraint

$$\tilde{\mathbf{x}}^\mathsf{T}\mathbf{D}\tilde{\mathbf{x}} = 1 \tag{3}$$

where $\tilde{\mathbf{x}} = (x'_1, \ldots, x'_N)^\mathsf{T}$ and $D = \text{diag}\left(\sum_{i=1}^{N} w_{1i}, \ldots, \sum_{i=1}^{N} w_{Ni}\right)$

Derivation of the LE Algorithm: Graph Laplacian

- Rewrite the objective function:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i' - x_j')^2 w_{ij} \tag{4}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N}(x_i'^2 + x_j'^2 - 2x_i' x_j')w_{ij} \tag{5}$$

$$= 2 \cdot \sum_{i,j=1}^{N} x_i'^2 w_{ij} - 2 \cdot \sum_{i,j=1}^{N} x_i' x_j' w_{ij} \tag{6}$$

$$= 2 \cdot \left(\tilde{\mathbf{x}}^\mathsf{T}\mathbf{D}\tilde{\mathbf{x}} - \tilde{\mathbf{x}}^\mathsf{T}\mathbf{W}\tilde{\mathbf{x}}\right) \tag{7}$$

$$= 2 \cdot \left(\tilde{\mathbf{x}}^\mathsf{T}\left(\mathbf{D} - \mathbf{W}\right)\tilde{\mathbf{x}}\right) \tag{8}$$

- The matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is one variant of the Graph Laplacian: the row-sum of affinities on the diagonal, negative affinities on the off-diagonal entries

Solution for the Lower-Dimensional Basis

- Minimize $\tilde{\mathbf{x}}\mathbf{L}\tilde{\mathbf{x}}$ subject to the constraint $\tilde{\mathbf{x}}\mathbf{D}\tilde{\mathbf{x}}$:

$$\frac{\partial}{\partial \tilde{\mathbf{x}}} \left(\tilde{\mathbf{x}}\mathbf{L}\tilde{\mathbf{x}} + \lambda \tilde{\mathbf{x}}\mathbf{D}\tilde{\mathbf{x}} \right) \stackrel{!}{=} 0 \tag{9}$$

$$2\left(\mathbf{L}\tilde{\mathbf{x}} - \lambda \mathbf{D}\tilde{\mathbf{x}} \right) = 0 \tag{10}$$

$$\mathbf{L}\tilde{\mathbf{x}} = \lambda \mathbf{D}\tilde{\mathbf{x}} \tag{11}$$

$$\mathbf{D}^{-1}\mathbf{L}\tilde{\mathbf{x}} = \lambda \tilde{\mathbf{x}} \tag{12}$$

- Remember that the goal is to minimize the objective function
- Hence, the smallest eigenvalues indicate the solution
- However, discard the eigenvalues $\lambda = 0$, since they only indicate the number of independent components
- In summary, the embedding for sample $\mathbf{x}_i \in \mathbb{R}^d$ onto a 1-D space is just the i-th entry of the eigenvector associated with the smallest non-zero eigenvalue of $\mathbf{D}^{-1}\mathbf{L}$

Manifold Forests: Density Forests + Laplacian Eigenmaps

- **L** can be composed of any set of pairwise affinities
- We hence also look at "Manifold Forests" by Criminisi/Shotton/Konukoglu, where affinities are defined on a density forest[1]
- For each tree t, define an affinity matrix $\mathbf{W}_t$ from a distance $d_t(\mathbf{x}_i, \mathbf{x}_j)$:

$$\mathbf{W}_t = [w_{ij}]_t = \exp(-d_t(\mathbf{x}_i, \mathbf{x}_j)) \tag{13}$$

- Reasonable choices for $d_t(\mathbf{x}_i, \mathbf{x}_j)$ provide for example
  - Gaussian affinity:

$$d_t(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \frac{(\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}(\mathbf{x}_i - \mathbf{x}_j)}{\sigma^2} & \text{if leaf}(\mathbf{x}_i) = \text{leaf}(\mathbf{x}_j) \\ \infty & \text{otherwise} \end{cases} \tag{14}$$

  - Binary affinity:

$$d_t(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 0 & \text{if leaf}(\mathbf{x}_i) = \text{leaf}(\mathbf{x}_j) \\ \infty & \text{otherwise} \end{cases} \tag{15}$$

[1] This topic is covered in Sec. 6 of the Criminisi/Shotton/Konukoglu random forests paper, which is available in studOn

Internals of the Learned Affinities

- With binary affinities, the nodes can be permuted s.t. W_t is a block matrix:

$$W_t = \begin{pmatrix} 1 & 1 & 1 & & & \cdots & & & & \\ 1 & 1 & 1 & & & & & & & \\ 1 & 1 & 1 & & & & & & & \\ 1 & 1 & 1 & & & & & & & \\ & & & 1 & 1 & & & & & \\ & & & 1 & 1 & & & & & \\ & & & & & & & & & \\ \vdots & & & & & \ddots & & & & \\ & & & & & & 1 & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 1 & 1 \end{pmatrix} \tag{16}$$

- Averaging $\mathbf{W} = \sum_{t=1}^{T} \mathbf{W}_t$ over the forest removes the block structure
- Affinities are larger for sample pairs that are often in the same leaf
- This creates a sense of locality, which is learned from the data
- The weight matrix $\mathbf{W}$ can directly be used to construct $\mathbf{L}$

Lower-dimensional Embedding with Manifold Forests

- Criminisi/Shotton/Konukoglu use a normalized version of $\mathbf{L}$,[2]

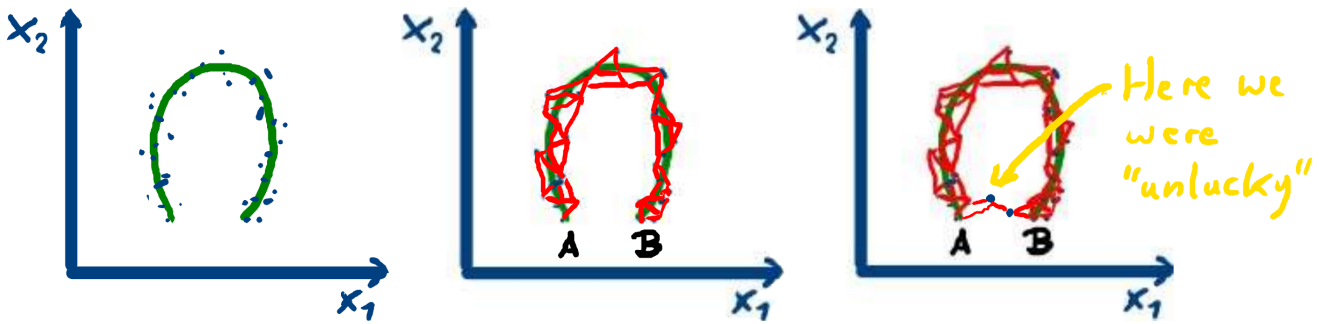
$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{17}$$

  where $\mathbf{I}$ is the $N \times N$ identity matrix
- The embedding to d' dimensions (d' can be larger than 1) is obtained by
  1. Sorting the eigenvalues and their eigenvectors in increasing order
  2. Discarding all eigenvectors with associated eigenvalues $\lambda = 0$
  3. The d'-dimensional embedding is a standard projection:
     the lower-dimensional coordinates of $\mathbf{x}_i$ are the i-th entries of the eigenvectors associated with the d' smallest eigenvalues

[2] Don't let this confuse you, it is equivalent to our previous variant

Remarks

- Laplacian Eigenmaps are more resilient to outliers than ISOMAP:
  - In ISOMAP, few outliers can create a shortcut on the manifold that significantly reduces the shortest paths



  - In LE, all affinities jointly define closely the strongly connected components, hence isolated shortcuts have less impact
- The Graph Laplacian is the central tool in spectral graph theory connecting graphs and linear algebra. A current example is deep learning on graphs
- Another example is spectral clustering: Here, the k eigenvectors of the largest eigenvalues of $\mathbf{L}$ pre-partition the sample neighborhood graph. Then, k-means assigns the clusters