Lecture Pattern Analysis

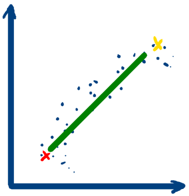
Part 15: Laplacian Eigenmaps (LE)

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 3, 2021

Introduction

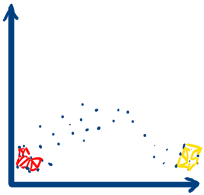
- Laplacian Eigenmaps (LE) advance the use of local distances one step further
- Instead of preserving the global variance (like PCA, MDS), LE aims to preserve local neighborhoods:

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

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

where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$

- Additional constraint to prevent the trivial solution $x_1 = \ldots = x_N$:

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

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

Derivation of the LE Algorithm: Graph Laplacian

- Rewrite the objective function:

$$
\begin{aligned}
&\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i' - x_j')^2 w_{ij} && (4)\\
=&\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i'^2 + x_j'^2 - 2x_i'x_j')w_{ij} && (5)\\
=&2\cdot\sum_{i=1}^{N} x_i'^2 w_{ii} - 2\cdot\sum_{i,j=1}^{N} x_i'x_j'w_{ij} && (6)\\
=&2\cdot\left(\tilde{\mathbf{x}}^\mathsf{T}\mathbf{D}\tilde{\mathbf{x}} - \tilde{\mathbf{x}}^\mathsf{T}\mathbf{W}\tilde{\mathbf{x}}\right) && (7)\\
=&2\cdot\left(\tilde{\mathbf{x}}^\mathsf{T}\left(\mathbf{D}-\mathbf{W}\right)\tilde{\mathbf{x}}\right) && (8)
\end{aligned}
$$

- 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 $\tilde{\mathbf{x}}\mathbf{D}\tilde{\mathbf{x}}$:

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

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

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

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

- 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 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:

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

  - Binary affinity:

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

[1] This topic is covered in Sec. 6 of the Criminisi/Shotton/Konukoglu random forests paper, which we used Lecture 04 — you can find it there 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 **L**,[2]

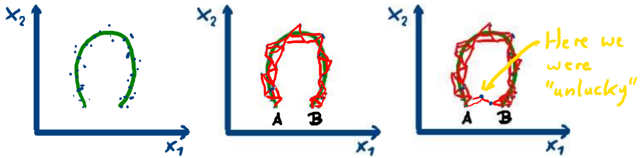
$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{17}$$

  where **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 d' smallest eigenvectors

[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, the sample affinities are preserved by a joint optimization criterion, which is less sensitive to isolated shortcuts
- 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