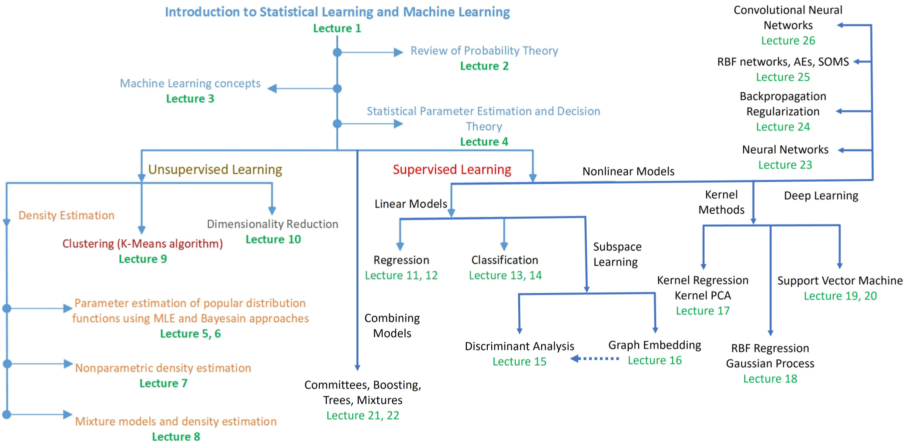
# Statistical Learning and Machine Learning

## Lecture 16 - Graph-based Learning

October 13, 2021

# Course overview and where do we stand



# Definitions

Let us consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$:

- $\mathcal{V} = \{v_1, \ldots, v_N\}$ is called the *vertex set* of the graph
- $\mathcal{E} = \{e_{ij}\}$ is called the *edge set* of the graph
  - $e_{ij}$ is a edge connecting $v_i$ and $v_j$
  - $e_{ij}$ can indicate a link between the $i$-th and the $j$-th vertices (*unweighted graph*)
  - $e_{ij}$ can be associated with a weight value $w_{ij}$ encoding how strong is the link between the $i$-th and the $j$-th vertices (*weighted graph*)

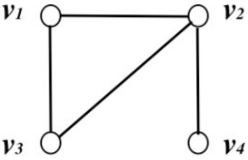
The graph can be:

- fully connected: there is an edge between every pair of vertices (and the associated weight)
- partially connected: each vertex is connected only with some other vertices (those which are more similar to it according to a similarity/distance metric)

# Definitions

The *adjacency* matrix $\mathrm{A} \in \mathbb{R}^{N \times N}$ of the graph is defied as:

$$\mathrm{A} := \begin{cases} A_{ij} = 1, & \text{if there is an edge } e_{ij} \\ A_{ij} = 0, & \text{if there is no edge} \\ A_{ii} = 0 & \end{cases} \tag{1}$$




$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The weight matrix $\mathrm{W} \in \mathbb{R}^{N \times N}$ of the graph has the same form, but each element encodes the strength of connection (similarity) between the corresponding vertices.

# Laplacian Embedding

Consider a value $x_i$ associated with vertex $v_i$ of an undirected weighted graph:

- We want the relation between the values associated with the graph vertices to be the same as those expressed by the corresponding weights $w_{ij}$
- Example of graph weights (Gaussian kernel):

$$w_{ij} = exp\left(-\frac{(\mathrm{x}_i - \mathrm{x}_j)^2}{\sigma^2}\right) \tag{2}$$

$$0 \leq w_{min} \leq w_{ij} \leq w_{max} \leq 1. \tag{3}$$

  This means that if $w_{ij}$ is high, the representations of vertices $v_i$ and $v_j$ should be close using the Euclidean distance
- This means that the vertex representations should be those minimizing:

$$\mathcal{J} = \frac{1}{2}\sum_{e_{ij}} w_{ij}\left(x_i - x_j\right)^2 \tag{4}$$

# Laplacian Embedding

The minimization criterion is:

$$\mathcal{J} = \frac{1}{2} \sum_{e_{ij}} w_{ij} \left( x_i - x_j \right)^2 = \mathrm{x}^T \mathrm{L} \mathrm{x} \tag{5}$$

where $\mathrm{L} = \mathrm{D} - \mathrm{W} \in \mathbb{R}^{N \times N}$ is the *Laplacian* matrix.

$\mathrm{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix (called *Degree* matrix) having elements:

$$D_{ii} = \sum_j W_{ij} \tag{6}$$

We introduce two constraints:

$$\mathrm{x}^T \mathrm{x} = 1 \qquad \text{and} \qquad \mathrm{x}^T 1 = 0 \tag{7}$$

The solution is obtained by solving the eigenanalysis problem:

$$\mathrm{L}\mathrm{x} = \lambda \mathrm{x} \tag{8}$$

# Graph Embedding

Graph Embedding uses a graph $\mathcal{G} = \{\mathcal{V}, \mathrm{W}, \mathrm{X}\}$:

- $\mathcal{V} = \{v_1, \dots, v_N\}$ is the vertex set of the graph
- $\mathrm{W} \in \mathbb{R}^{N \times N}$ is the weight matrix the elements of which encode the similarity between vertex pairs
- $\mathrm{X} = [\mathrm{x}_1, \dots, \mathrm{x}_N] \in \mathbb{R}^{D \times N}$ is a set of $D$- dimensional vectors with $\mathrm{x}_i$ representing $v_i$

Graph Embedding defines two graphs:

- an (*intrinsic*) graph $\mathcal{G}^I$ expressing properties of the data that we want the embedding to enhance
- a *penalty* graph $\mathcal{G}^P$ expressing properties of the data that we want the embedding to penalize

# Graph Embedding

Suppose that:

- $\mathrm{L} \in \mathbb{R}^{N \times N}$ is the Laplacian matrix of $\mathcal{G}^I$
- $\mathrm{B} \in \mathbb{R}^{N \times N}$ is the Laplacian matrix of $\mathcal{G}^P$

Graph Embedding optimizes the following criterion:

$$y^* = \underset{\mathrm{y}^T \mathrm{B} \mathrm{y} = q}{\arg\min} \sum_{i \neq i} (y_i - y_j)^2 W_{ij} = \underset{\mathrm{y}^T \mathrm{B} \mathrm{y} = q}{\arg\min}\, \mathrm{y}^T \mathrm{L} \mathrm{y} \tag{9}$$

where:

- $q$ is a constant value
- $y_i \in \mathbb{R}$ is the value associated with vertex $v_i$
- $\mathrm{y} = [y_1, \ldots, y_N]^T$

# Graph Embedding

If we define a linear mapping:

$$y_i = \mathrm{w}^T \mathrm{x}_i \tag{10}$$

then the criterion of obtaining the projection matrix $\mathrm{w} \in \mathbb{R}^D$ is:

$$\begin{aligned} \mathrm{w}^* &= \underset{\substack{\mathrm{w}^T\mathrm{XBX}^T\mathrm{w}=q \\ \text{or } \mathrm{w}^T\mathrm{w}=q}}{\arg\min} \sum_{i \neq j} (\mathrm{w}^T \mathrm{x}_i - \mathrm{w}^T \mathrm{x}_j)^2 W_{ij} \\ &= \underset{\substack{\mathrm{w}^T\mathrm{XBX}^T\mathrm{w}=q \\ \text{or } \mathrm{w}^T\mathrm{w}=q}}{\arg\min} \mathrm{w}^T \mathrm{XLX}^T \mathrm{w} \end{aligned} \tag{11}$$

$\mathrm{w}^*$ is calculated by solving the *generalized eigenanalysis* problem:

$$\tilde{\mathrm{L}}\mathrm{w} = \lambda \tilde{\mathrm{B}}\mathrm{w} \tag{12}$$

where:

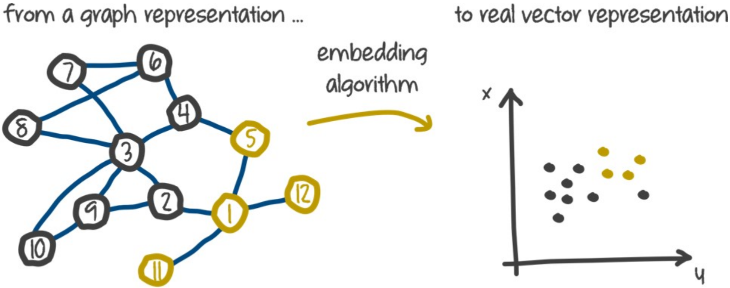
$$\tilde{\mathrm{L}} = \mathrm{XLX}^T \quad \text{and} \quad \tilde{\mathrm{B}} = \mathrm{XBX}^T \tag{13}$$

# Graph Embedding

To define more than one dimensions $y \in \mathbb{R}^d$ we solve again the same generalized eigenanalysis problem:

$$\tilde{L}w = \lambda \tilde{B}w \tag{14}$$

and we keep the eigenvectors corresponding to the $d$ smallest eigenvalues.



# Graph Embedding: LDA

Let us take again a look at the within-class scatter matrix of LDA:

$$\mathrm{S}_W = \sum_{k=1}^{K} \sum_{\mathrm{x}_i \in \mathcal{C}_k} (\mathrm{x}_i - \boldsymbol{\mu}_k)(\mathrm{x}_i - \boldsymbol{\mu}_k)^T \tag{15}$$

Using:

- $1_k \in \mathbb{R}^N$ a vector having elements:

$$1_k(i) := \begin{cases} 1, & \text{if } \mathrm{x}_i \in \mathcal{C}_k \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

- $\mathrm{J}_k = diag(1_k)$

# Graph Embedding: LDA

Let us take again a look at the within-class scatter matrix of LDA:

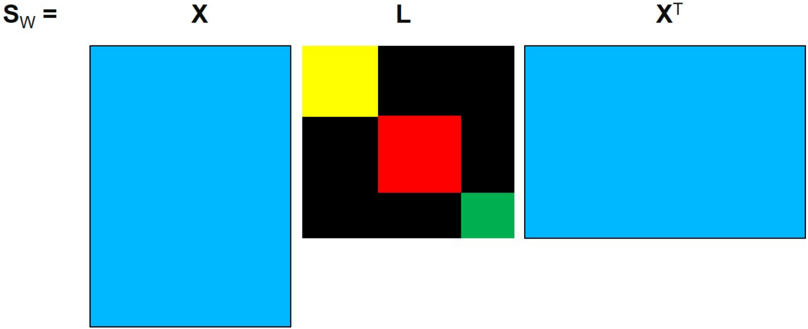
$$\mathrm{S}_W = \sum_{k=1}^{K} \sum_{i, l_i = k} (\mathrm{x}_i - \boldsymbol{\mu}_k)(\mathrm{x}_i - \boldsymbol{\mu}_k)^T \tag{17}$$

$$\begin{aligned}
\mathrm{S}_W &= \sum_{k=1}^{K} \left(\mathrm{X}\mathrm{J}_k - \frac{1}{N_k}\mathrm{X}1_k 1_k^T\right)\left(\mathrm{X}\mathrm{J}_k - \frac{1}{N_k}\mathrm{X}1_k 1_k^T\right)^T \\
&= \sum_{k=1}^{K} \mathrm{X}\left(\mathrm{J}_k - \frac{1}{N_k}1_k 1_k^T\right)\left(\mathrm{J}_k - \frac{1}{N_k}1_k 1_k^T\right)^T \mathrm{X}^T \\
&= \mathrm{X}\left(\sum_{k=1}^{K}\left(\mathrm{J}_k - \frac{1}{N_k}1_k 1_k^T\right)\left(\mathrm{J}_k - \frac{1}{N_k}1_k 1_k^T\right)^T\right)\mathrm{X}^T \\
&= \mathrm{X}\mathrm{L}\mathrm{X}^T
\end{aligned} \tag{18}$$

# Graph Embedding: LDA

Let us take again a look at the within-class scatter matrix of LDA:

$$S_W = \sum_{k=1}^{K} \sum_{i, l_i = k} (x_i - \boldsymbol{\mu}_k)(x_i - \boldsymbol{\mu}_k)^T \tag{19}$$




Black color corresponds to pairs of data points belonging to different classes (these elements take zero values).

# Graph Embedding: LDA

Let us take again a look at the between-class scatter matrix of LDA:

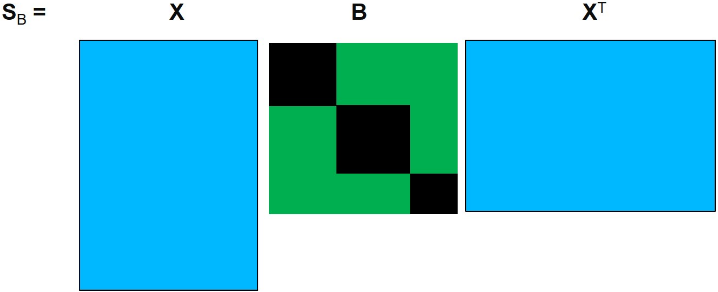
$$\mathrm{S}_B = \sum_{k=1}^{K} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{20}$$

$$\begin{aligned}
\mathrm{S}_B &= \sum_{k=1}^{K} N_k \Big(\frac{1}{N_k}\mathrm{X}1_k - \frac{1}{N}\mathrm{X}1\Big)\Big(\frac{1}{N_k}\mathrm{X}1_k - \frac{1}{N}\mathrm{X}1\Big)^T \\
&= \sum_{k=1}^{K} N_k \mathrm{X}\Big(\frac{1}{N_k}1_k - \frac{1}{N}1\Big)\Big(\frac{1}{N_k}1_k - \frac{1}{N}1\Big)^T \mathrm{X}^T \\
&= \mathrm{X}\left(\sum_{k=1}^{K} N_k \Big(\frac{1}{N_k}1_k - \frac{1}{N}1\Big)\Big(\frac{1}{N_k}1_k - \frac{1}{N}1\Big)^T\right)\mathrm{X}^T \\
&= \mathrm{XBX}^T
\end{aligned} \tag{21}$$

# Graph Embedding: LDA

Let us take again a look at the between-class scatter matrix of LDA:

$$S_B = \sum_{k=1}^{K} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{22}$$

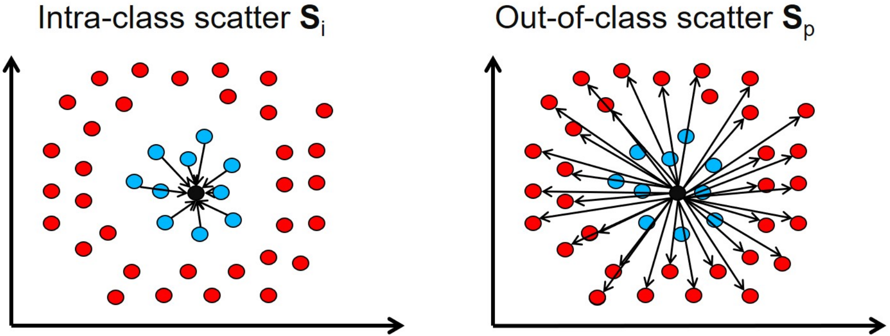



Black color corresponds pairs of data points belonging to the same class (these elements take values equal to $(1/N_k - 1/N)^2$).

# Graph Embedding: Other types of graphs

Using the Graph Embedding procedure, we can define subspace learning methods optimize several types of criteria.

Example: Class-Specific Discriminant Analysis using intra-class variance $S_i$ and out-of-class variance matrix $S_p$.



# Graph Embedding: Generic Graph types

Marginal Discriminant Analysis:

- For each data point find its $K_w$ nearest neighbors considering only the data points belonging to the same class (call the neighborhood of $\mathrm{x}_i$ as $\mathcal{N}_{\mathrm{x}_i}$). Create the intrinsic graph weight matrix $\mathrm{W}_I$ having elements:

$$\mathrm{W}_{I,ij} := \begin{cases} 1, & \text{if } \mathrm{x}_j \in \mathcal{N}_{\mathrm{x}_i} \text{ and } \mathrm{x}_i \in \mathcal{N}_{\mathrm{x}_j} \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

- For each pair of classes $\mathcal{C}_k$ and $\mathcal{C}_l$ find the $K_b$ data points that correspond to the $K_w$ smallest (between-class) distances include them in the set $\mathcal{N}_{kl}$. Create the penalty graph weight matrix $\mathrm{W}_P$ having elements:

$$\mathrm{W}_{P,ij} := \begin{cases} 1, & \text{if}\, \mathrm{x}_j \,\text{and}\, \mathrm{x}_j \in \mathcal{N}_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

- Calculate the Laplacian matrices $\tilde{\mathrm{L}} = \mathrm{D}_I - \mathrm{W}_I$ and $\tilde{\mathrm{B}} = \mathrm{D}_I - \mathrm{W}_P$
- Solve for $\tilde{\mathrm{L}}\mathrm{w} = \lambda\tilde{\mathrm{B}}\mathrm{w}$ and we keep the eigenvectors corresponding to the $d$ smallest eigenvalues

# Spectral Clustering

*Spectral Clustering* exploits a data transformation based on the spectrum (eigenanalysis) of the graph Laplacian matrix for applying nonlinear clustering.

Algorithm:

1. Construct a fully connected graph $\mathcal{G} = \{\mathcal{V}, \mathrm{W.X}\}$ using $\mathrm{x}_i \in \mathbb{R}^D,\ i = 1, \ldots, N$
2. Calculate the graph Laplacian matrix $\mathrm{L} = \mathrm{D} - \mathrm{W}$
3. Apply eigenanalysis to L and keep the eigenvectors corresponding to the $d$ smallest eigenvalues to form new $d$-dimensional data representations $\mathrm{y}_i \in \mathbb{R}^d$
4. Apply a clustering algorithm (e.g. $K$-Means) using $\mathrm{y}_i,\ i = 1, \ldots, N$
5. Assign the cluster labels to the original data $\mathrm{x}_i,\ i = 1, \ldots, N$

# Spectral Clustering