MACHINE LEARNING

kernel CCA, kernel Kmeans
Spectral Clustering
Change in timetable: We have practical session next week!

<table>
<thead>
<tr>
<th>Date</th>
<th>Sections</th>
<th>Topics</th>
<th>Exercises</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 March</td>
<td>Sections 2.1-2.2</td>
<td>Spectral methods: Structure Discovery, Dim. Reduction PCA, Probabilistic PCA, kernel PCA</td>
<td>Exercises Matlab code for kPCA</td>
<td>Solutions</td>
</tr>
<tr>
<td></td>
<td>(must be read before class)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sections 5.1-5.3.</td>
<td></td>
<td>Practicals Continued</td>
<td>Assignment Datasets (Description) Solution</td>
</tr>
<tr>
<td>8 March</td>
<td></td>
<td>Practicals in room GRC002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(must be read before class)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sections 5.4-5.6.</td>
<td></td>
<td>Practicals Continued</td>
<td>Assignment Datasets</td>
</tr>
<tr>
<td>22 March</td>
<td></td>
<td>Practicals in room GRC002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29 March</td>
<td></td>
<td>Good Friday (public holiday)</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>5 April</td>
<td></td>
<td>Easter Break</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>12 April</td>
<td></td>
<td>Non-linear Classification Support Vector Machine, RVM</td>
<td>Exercises Matlab Code SVM</td>
<td></td>
</tr>
</tbody>
</table>
Structure of today’s and next week’s class

1) Briefly go through some extension or variants on the principle of kernel PCA, namely kernel CCA.

2) Look at one particular set of clustering algorithms for structure discovery, kernel K-Means

3) Describe the general concept of Spectral Clustering, highlighting equivalency between kernel PCA, ISOMAP, etc and their use for clustering.

4) Introduce the notion of unsupervised and semi-supervised learning and how this can be used to evaluate clustering methods

5) Compare kernel K-means and Gaussian Mixture Model for unsupervised and semi-supervised clustering
 Canonical Correlation Analysis (CCA)

\[ x \in \mathbb{R}^N \quad \text{and} \quad y \in \mathbb{R}^P \]

Determine features in two (or more) separate descriptions of the dataset that best explain each datapoint.

Extract hidden structure that maximize correlation across two different projections.
Canonical Correlation Analysis (CCA)

Pair of multidimensional zero mean variables

We have $M$ instances of the pairs.

Search two projections $w_x$ and $w_y$:

$X' = w_x^T X$ and $Y' = w_y^T Y$

solutions of:

$max \rho = \max_{w_x, w_y} corr(X', Y')$
Canonical Correlation Analysis (CCA)

\[
\text{max } \rho = \max_{w_x,w_y} \text{corr}(X',Y')
\]

\[
= \max_{w_x,w_y} \frac{w_x^T E\{XY^T\} w_y}{\|w_x^T X\| \|w_y^T Y\|}
\]

\[
= \max_{w_x,w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x w_y^T C_{yy} w_y}}
\]

Crosscovariance matrix

\[C_{xy}\] is \(N \times q\)

Measure crosscorrelation between \(X\) and \(Y\).
Canonical Correlation Analysis (CCA)

\[
\max \rho = \max_{w_x, w_y} \text{corr}(X', Y')
\]

\[
= \max_{w_x, w_y} \frac{w_x^T E \{ XY^T \} w_y}{\| w_x^T X \| \| w_y^T Y \|}
\]

\[
= \max_{w_x, w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x} \sqrt{w_y^T C_{yy} w_y}}
\]

Correlation not affected by rescaling the norm of the vectors,
\[
\Rightarrow \text{we can ask that } \| w_x^T C_{xx} w_x \| = \| w_y^T C_{yy} w_y \| = 1
\]

\[
\max \rho = \max_{w_x, w_y} w_x^T C_{xy} w_y
\]

u. c. \[
\| w_x^T C_{xx} w_x \| = \| w_y^T C_{yy} w_y \| = 1
\]
To determine the optimum (maximum) of $\rho$, solve by Lagrange:

$$\max_{w_x,w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x w_y^T C_{yy} w_y}}$$

u.c. $\|w_x^T C_{xx} w_x\| = \|w_y^T C_{yy} w_y\| = 1$

$$\Rightarrow C_{xy} C_{yy}^{-1} C_{yx} w_x = \lambda^2 C_{xx} w_x$$

Generalized Eigenvalue Problem; can be reduced to a classical eigenvalue problem if $C_{xx}$ is invertible
Kernel Canonical Correlation Analysis

- CCA finds basis vectors, s.t. the correlation between the projections is mutually maximized.

→ generalized version of PCA for two or more multi-dimensional datasets.

CCA depends on the coordinate system in which the variables are described.

Even if strong linear relationship between variables, depending on the coordinate system used, this relationship might not be visible as a correlation → Kernel CCA.
**Principle of Kernel Methods**

Determine a metric which brings out features of the data so as to make subsequent computation easier.

Data becomes *linearly* separable when using a rbf kernel and projecting onto first 2 PC of kernelPCA.
Principle of Kernel Methods (Recall)

Idea: Send the data $X$ into a *feature space* $H$ through the nonlinear map $\phi$.

In feature space, perform classical *linear* computation

$$X = \{ x^i \in \mathbb{R}^N \}_{i=1,...,M} \quad \mapsto \quad \phi(X) = (\phi(x^1), \ldots, \phi(x^M))$$

Performs linear transformation in feature space
Kernel CCA

\[ X = \left\{ x^i \in \mathbb{R}^N \right\}_{i=1}^M, Y = \left\{ y^i \in \mathbb{R}^q \right\}_{i=1}^M \]

- Project into a feature space

\[ \left\{ \phi_x \left( x^i \right) \right\}_{i=1}^M \quad \text{and} \quad \left\{ \phi_y \left( y^i \right) \right\}_{i=1}^M \]

\[ \text{with } \sum_{i=1}^M \phi_x \left( x^i \right) = 0 \quad \text{and} \quad \sum_{i=1}^M \phi_y \left( y^i \right) = 0 \]

- Construct associated kernel matrices:

\[ K_x = F_x F_x^T, K_y = F_y F_y^T, \quad \text{columns of } F_x, F_y \text{ are } \phi_x \left( x^i \right), \phi_y \left( y^i \right) \]

- The projection vectors can be expressed as a linear combination in feature space:

\[ w_x = F_x \alpha_x \quad \text{and} \quad w_y = F_y \alpha_y \]
Kernel CCA

Kernel CCA can then become an optimization problem of the

\[
\max_{\alpha_x, \alpha_y} \rho = \max_{\alpha_x, \alpha_y} \frac{\alpha_x^T K_x \alpha_y}{\left(\alpha_x^T K_x^2 \alpha_x\right)^{1/2} \left(\alpha_y^T K_y^2 \alpha_y\right)^{1/2}}
\]

u.c. \( \alpha_x^T K_x^2 \alpha_x = \alpha_y^T K_y^2 \alpha_y = 1 \)

In Linear CCA, we were solving for:

\[
\max_{w_x, w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x} \sqrt{w_y^T C_{yy} w_y}}
\]

u.c. \( \|w_x^T C_{xx} w_x\| = \|w_y^T C_{yy} w_y\| = 1 \)
Kernel CCA

Kernel CCA can then become an optimization problem of the

$$\max_{\alpha_x, \alpha_y} \rho = \max_{\alpha_x, \alpha_y} \frac{\alpha_x^T K_x K_y \alpha_y}{\left(\alpha_x^T K_x^2 \alpha_x\right)^{1/2} \left(\alpha_y^T K_y^2 \alpha_y\right)^{1/2}}$$

u.c. \(\alpha_x^T K_x^2 \alpha_x = \alpha_y^T K_y^2 \alpha_y = 1\)

In practice, the intersection between the spaces spanned by \(K_x \alpha_x, K_y \alpha_y\) is non-zero, then the problem has a trivial solution, as \(\rho \sim \cos\left(K_x \alpha_x, K_y \alpha_y\right) = 1\).
Kernel CCA

Generalized eigenvalue problem:
\[
\begin{pmatrix}
0 & K_x K_y \\
K_y K_x & 0
\end{pmatrix}
\begin{pmatrix}
\alpha_x \\
\alpha_y
\end{pmatrix}
= \lambda
\begin{pmatrix}
K_x^2 & 0 \\
0 & K_y^2
\end{pmatrix}
\begin{pmatrix}
\alpha_x \\
\alpha_y
\end{pmatrix}
\]

Add a regularization term to increase the rank of the matrix and make it invertible:
\[
K_x^2 \rightarrow \left( K_x + \frac{M \kappa}{2} I \right)^2
\]

Several methods have been proposed to choose carefully the regularizing term so as to get projections that are as close as possible to the “true” projections.
Kernel CCA

\[
\begin{pmatrix} 0 & K_x K_y \\ K_y K_x & 0 \end{pmatrix} \begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix} = \lambda \begin{pmatrix} \left( K_x + \frac{M \kappa}{2} I \right)^2 & 0 \\ 0 & \left( K_y + \frac{M \kappa}{2} I \right)^2 \end{pmatrix} \begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix}
\]

Set: \( B = C^T C \) and \( \beta = C \alpha \)

Becomes a classical eigenvalue problem \( (C^T)^{-1} AC^{-1} \beta = \lambda \beta \)
Kernel CCA

Two datasets case

\[ X = \left\{ x^i \in \mathbb{R}^N \right\}_{i=1}^M, Y = \left\{ y^i \in \mathbb{R}^q \right\}_{i=1}^M \]

Can be extended to multiple datasets:

$L$ datasets: \( X_1, \ldots, X_L \) with $M$ observations each

Dimensions \( N_1, \ldots, N_L \), i.e. \( X_i : N_i \times M \)

Applying a non-linear transformation \( \phi \) to \( X_1, \ldots, X_L \)

\( \rightarrow \) construct $L$ Gram matrices: \( K_1, \ldots, K_L \)
Kernel CCA

Two datasets case

Was formulated as generalized eigenvalue problem:

\( X = \{ x^i \in \mathbb{R}^N \}_{i=1}^{M}, Y = \{ y^i \in \mathbb{R}^q \}_{i=1}^{M} \)

\[
\begin{pmatrix}
0 & K_xK_y \\
K_yK_x & 0
\end{pmatrix}
\begin{pmatrix}
\alpha_x \\
\alpha_y
\end{pmatrix}
= \lambda
\begin{pmatrix}
(K_x + \frac{M\kappa}{2} I)^2 & 0 \\
0 & (K_y + \frac{M\kappa}{2} I)^2
\end{pmatrix}
\begin{pmatrix}
\alpha_x \\
\alpha_y
\end{pmatrix}
\]

Can be extended to multiple datasets:

\[
\begin{pmatrix}
0 & K_1K_2 & \ldots & K_1K_L \\
K_2K_1 & 0 & \ldots & K_2K_L \\
\vdots & \vdots & \ddots & \vdots \\
K_LK_1 & K_LK_2 & \ldots & K_LK_L
\end{pmatrix}
\begin{pmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_L
\end{pmatrix}
= \lambda
\begin{pmatrix}
(K_1^2 + \frac{M\kappa}{2} I)^2 & 0 \\
\vdots & \ddots \\
0 & (K_L^2 + \frac{M\kappa}{2} I)^2
\end{pmatrix}
\begin{pmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_L
\end{pmatrix}
\]
Kernel CCA

Figure 3: Kernel canonical correlation example. The data consists of two sets of 100 points each. For X the points are lying on a circle (solid points) while Y (circles) describe a sine curve (points correspond by arclength). For X we used a RBF kernel ($\sigma = 1$) and for Y a homogeneous polynomial kernel of degree ($d = 2$). The lines plotted describe regions of equal score on the first canonical vectors, which can be thought of as orthogonal (see Schölkopf et al. (1998)). This is shown for $v_1 \in \mathcal{L}\{\Phi_X\}$ (upper) and for $w_1 \in \mathcal{L}\{\Phi_Y\}$ (middle). The bottom plot shows the first pair of kernel canonical variates ($a_1, b_1$) showing that $\langle \phi(x_i), v_1 \rangle_F$ and $\langle \phi(y_i), w_1 \rangle_F$ are highly correlated for $i = 1, \ldots, m$.

Kernel CCA

Matlab Example  File: demo_CCA-KCCA.m
Applications of Kernel CCA

Goal: To measure correlation between heterogeneous datasets and to extract sets of genes which share similarities with respect to multiple biological attributes.

Kernel matrices $K_1$, $K_2$ and $K_3$ correspond to gene-gene similarities in pathways, genome position, and microarray expression data respectively. Use RBF kernel with fixed kernel width.

Correlation scores in MKCCA:
- pathway vs. genome vs. expression.

Applications of Kernel CCA

Goal: To measure correlation between heterogeneous datasets and to extract sets of genes which share similarities with respect to multiple biological attributes.

- Correlation scores in MKCCA: pathway vs. genome vs. expression.

A readout of the entries with equal projection onto the first canonical vectors $\alpha$ give the genes which belong to each cluster.

Two clusters correspond to genes close to each other with respect to their positions in the pathways, in the genome, and to their expression.

Applications of Kernel CCA

Goal: To construct appearance models for estimating an object’s pose from raw brightness images

X: Set of images

Y: pose parameters (pan and tilt angle of the object w.r.t. the camera in degrees)

Example of two image datapoints with different poses

Use linear kernel on X and RBF kernel on Y and compare performance to applying PCA on the (X, Y) dataset directly

Applications of Kernel CCA

Goal: To construct appearance models for estimating an object’s pose from raw brightness images

kernel-CCA performs better than PCA, especially for small k=testing/training ratio (i.e., for larger training sets).

The kernel-CCA estimators tend to produce less outliers, i.e., gross errors, and consequently yield a smaller standard deviation of the pose estimation error than their PCA-based counterparts.

For very small training sets, the performance of both approaches becomes similar.

Kernel K-means

Spectral Clustering
Structure Discovery: Clustering

Groups pair of points according to how similar these are.

Density based clustering methods (Soft K-means, Kernel K-means, Gaussian Mixture Models) compare the relative distributions.
K-means is a hard partitioning of the space through K clusters equidistant according to norm 2 measure.

The distribution of data within each cluster is encapsulated in a sphere.
K-means Algorithm

Iterative Method (variant on Expectation-Maximization)

1. Initialization: for $C^k$, $k = 1 \ldots K$, clusters, pick $K$ arbitrary centroids $\mu^k$, $k = 1 \ldots K$, and set their geometric means to random values.
K-means Algorithm

Iterative Method (variant on Expectation-Maximization)

2. Calculate the distance from each data point to each centroid.

\[ d(x^j, \mu^k) = \| x^j - \mu^k \|^p \]

3. **Assignment Step**: Assign the responsibility of each data point to its “closest” centroid (E-step). If a tie happens (i.e. two centroids are equidistant to a data point, one assigns the data point to the smallest winning centroid).

\[ \arg \min_k \{ d(x^j, \mu^k) \} \]
2. Calculate the distance from each data point to each centroid.

\[ d(x^j, \mu^k) = \| x^j - \mu^k \|^p \]

3. **Assignment Step:** Assign the responsibility of each data point to its “closest” centroid (**E-step**). If a tie happens (i.e. two centroids are equidistant to a data point, one assigns the data point to the smallest winning centroid).

\[ \arg \min_k \{ d(x^j, \mu^k) \} \]

4. **Update Step:** Adjust the centroids to be the means of all data points assigned to them (**M-step**)

5. Go back to step 2 and repeat the process until the clusters are stable.
K-means Clustering: **Weaknesses**

**Two hyperparameters**: number of clusters $K$ and power $p$ of the metric

Very sensitive to the choice of the number of clusters $K$ and the initialization.
K-means Clustering: Hyperparameters

Two hyperparameters: number of clusters $K$ and power $p$ of the metric

Choice of power determines the form of the decision boundaries

- $P=1$
- $P=2$
- $P=3$
- $P=4$
Kernel K-means

K-Means algorithm consists of minimization of:

$$J(\mu^1, \ldots, \mu^K) = \sum_{k=1}^{K} \sum_{x^j \in C^k} \|x^j - \mu^k\|^p$$

with

$$\mu^k = \frac{\sum_{x^j \in C^k} x^j}{m_k}$$

$m_k$ : number of datapoints in cluster $C^k$

Project into a feature space

$$J(\phi(x^1), \ldots, \phi(x^M)) = \sum_{k=1}^{K} \sum_{x^j \in C^k} \|\phi(x^j) - \phi(\mu^k)\|^p$$

We cannot observe the mean in feature space.

→ Construct the mean in feature space using image of points in same cluster
Kernel K-means

\[ J(\mu^1, \ldots, \mu^K) = \sum_{k=1}^{K} \sum_{x^j \in C^k} \left\| \phi(x^j) - \phi(\mu^k) \right\|^2 \]

\[ = \sum_{k=1}^{K} \sum_{x^j \in C^k} \left( \phi(x^j) \phi(x^j) - \frac{2}{m_k} \sum_{x^j' \in C^k} \phi(x^j') \phi(x^j) + \frac{\sum_{x^j, x^j' \in C^k} \phi(x^j) \phi(x^j')}{(m_k)^2} \right) \]

\[ = \sum_{k=1}^{K} \sum_{x^j \in C^k} \left( k(x^i, x^i) - \frac{2}{m_k} \sum_{x^j' \in C^k} k(x^i, x^j') + \frac{\sum_{x^j, x^j' \in C^k} k(x^j, x^j')}{(m_k)^2} \right) \]
Kernel K-means

Kernel K-means algorithm is also an iterative procedure:

1. **Initialization:** pick K clusters

2. **Assignment Step:** Assign each data point to its “closest” centroid (E-step). If a tie happens (i.e. two centroids are equidistant to a data point, one assigns the data point to the smallest winning centroid) by computing the distance in feature space.

\[
\min_{k} d\left(x^i, C^k\right) = \min_{k} \left( k\left(x^i, x^i\right) - \frac{2 \sum_{x^j \in C^k} k\left(x^i, x^j\right)}{m_k} + \frac{\sum_{x^j, x^l \in C^k} k\left(x^j, x^l\right)}{(m_k)^2} \right)
\]

3. **Update Step:** Update the list of points belonging to each centroid (M-step)

4. Go back to step 2 and repeat the process until the clusters are stable.
Kernel K-means

**With a RBF kernel**

\[
\min_{k} d \left( x^i, C^k \right) = \min_{k} \left( \kappa \left( x^i, x^i \right) - \frac{2 \sum_{x^j \in C^k} \kappa \left( x^i, x^j \right)}{m_k} + \frac{\sum_{x^j, x^l \in C^k} \kappa \left( x^j, x^l \right)}{\left( m_k \right)^2} \right)
\]

- If \( x^i \) is close to all points in cluster \( k \), this is close to 1.
- Cst of value 1
- If the points are well grouped in cluster \( k \), this sum is close to 1.

**With homogeneous polynomial kernel?**
Kernel K-means

With a polynomial kernel

Positive value

Some of the terms change sign depending on their position with respect to the origin.

If the points are aligned in the same Quadrant, the sum is maximal

\[
\min d(x^i, C^k) = \min_k \left\{ k(x^i, x^i) - \frac{2 \sum_{x^j \in C^k} k(x^i, x^j)}{m_k} + \sum_{x^j, x^l \in C^k} k(x^j, x^l) \left( \frac{m_k}{(m_k)^2} \right) \right\}
\]
Kernel K-means: examples

Rbf Kernel, 2 Clusters
Kernel K-means: examples

Rbf Kernel, 2 Clusters
Kernel K-means: examples

Rbf Kernel, 2 Clusters

Kernel width: 0.5

Kernel width: 0.05
Kernel K-means: examples

Polynomial Kernel, 2 Clusters
Kernel K-means: examples

Polynomial Kernel (p=8), 2 Clusters
Kernel K-means: examples
Polynomial Kernel, 2 Clusters

Order 2

Order 4

Order 6
Kernel K-means: examples
Polynomial Kernel, 2 Clusters

The separating line will always be perpendicular to the line passing by the origin (which is located at the mean of the datapoints) and parallel to the axis of the ordinates (because of the change in sign of the cosine function in the inner product).

→ No better than linear K-means!
Kernel K-means: examples
Polynomial Kernel, 4 Clusters

Can only group datapoints that do not overlap across quadrans with respect to the origin (careful, data are centered!).
→ No better than linear K-means! (except less sensitive to random initialization)
Kernel K-means: Limitations

Choice of number of Clusters in Kernel K-means is important
Kernel K-means: Limitations

Choice of number of Clusters in Kernel K-means is important
Kernel K-means: Limitations

Choice of number of Clusters in Kernel K-means is important
Limitations of kernel K-means

Raw Data
Limitations of kernel K-means

kernel K-means with K=3, RBF kernel
From Non-Linear Manifolds
Laplacian Eigenmaps, Isomaps

To Spectral Clustering
Non-Linear Manifolds

PCA and Kernel PCA belong to a more general class of methods to create non-linear manifolds based on *spectral decomposition*. (Spectral decomposition of matrices is more frequently referred to as an eigenvalue decomposition.)

Depending on which matrix we decompose, we get a different set of projections.

- PCA decomposes the covariance matrix of the dataset → generate rotation and projection in the original space
- Kernel PCA decomposes the Gram matrix → partition or regroup the datapoints
- The Laplacian matrix is a matrix representation of a graph. Its spectral decomposition can be used for clustering.
Embed Data in a Graph

- Build a *similarity graph*
- Each vertex on the graph is a datapoint
Measure Distances in Graph

Construct the similarity matrix $S$ to denote whether points are close or far away to weight the edges of the graph:
Disconnected Graphs

Disconnected Graph:
Two data-points are connected if:

a) the similarity between them is higher than a threshold.
b) or if they are k-nearest neighbors (according to the similarity metric)

\[
S = \begin{bmatrix}
1 & \ldots & 1 & \ldots & 0 & \ldots & 0 \\
0 & \ldots & 0 & \ldots & 1 & \ldots & 1
\end{bmatrix}
\]
Graph Laplacian

Given the similarity matrix $S = \begin{bmatrix} 1 \ldots \ldots 1 & \ldots & 0 & \ldots & 0 \\ \vdots \\ 0 \ldots \ldots 0 \ldots \ldots 1 \ldots \ldots 1 \end{bmatrix}$

Construct the diagonal matrix $D$ composed of the sum on each line of $K$:

$$D = \begin{bmatrix} \sum_i S_{1i} & \ldots & 0 \\ 0 & \sum_i S_{2i} & \ldots & 0 \\ \vdots \\ 0 & \ldots & \sum_i S_{Mi} \end{bmatrix},$$

and then, build the Laplacian matrix: $L = D - S$

$L$ is positive semi-definite $\rightarrow$ spectral decomposition possible
Graph Laplacian

Eigenvalue decomposition of the Laplacian matrix:

\[ L = U \Lambda U^T \]

All eigenvalues of L are positive and the smallest eigenvalue of L is zero:

⇒ If we order the eigenvalue by increasing order:
\[ \lambda_1 = 0 \leq \lambda_2 \leq \ldots \leq \lambda_M. \]

If the graph has \( k \) connected components, then the eigenvalue \( \lambda = 0 \) has multiplicity \( k \).
Spectral Clustering

⇒ The multiplicity of the eigenvalue 0 determines the number of connected components in a graph. The associated eigenvectors identify these connected components.

For an eigenvalue $\lambda_i = 0$, the corresponding eigenvector $e^i$ has the same value for all vertices in a component, and a different value for each one of their $i + 1$ components.

Identifying the clusters is then trivial when the similarity matrix is composed of zeros and ones (as when using k-nearest neighbor).

What happens when the similarity matrix is full?
Spectral Clustering

Similarity map $S : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$

$S$ can either be binary (k-nearest neighbour) or continuous with a Gaussian kernel:

$$S(x^i, x^j) = e^{-\frac{||x^i - x^j||^2}{2\sigma^2}}$$
Spectral Clustering

1) Build the Laplacian matrix: $L = D - S$

2) Do eigenvalue decomposition of the Laplacian matrix: $L = U \Lambda U^T$

3) Order the eigenvalue by increasing order:

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_M.$$ 

The first eigenvalue is still zero but with multiplicity 1 only (fully connected graph)!

Idea: the smallest eigenvalues are close to zero and hence provide also information on the partitioning of the graph (see exercise session)

Similarity map $S : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$

$S$ can either be binary (k-nearest neighbor) or continuous with Gaussian kernel

$$S(x^i, x^j) = e^{-\frac{\|x^i - x^j\|^2}{2\sigma^2}}$$
Spectral Clustering

Eigenvalue decomposition of the Laplacian matrix:

\[ L = U \Lambda U^T \]

\[
U = \begin{bmatrix}
e_1^1 & e_1^2 & \ldots & e_1^K \\
e_2^1 \\
\vdots \\
e_M^1 & e_1^2 & \ldots & e_1^K \\
\end{bmatrix}
\]

Construct an embedding of each of the \( M \) datapoints \( x^i \) through \( y^i \).
Reduce dimensionality by picking \( k < M \) projections \( y^i, i = 1 \ldots K \).

\[
y^i = \begin{bmatrix} e_i^1 \\ \vdots \\ e_i^K \end{bmatrix}
\]

With a clear partitioning of the graph, the entries in \( y \) are split into sets of equal values. Each group of points with same value belong to the same partition (cluster).
Spectral Clustering

Eigenvalue decomposition of the Laplacian matrix:

\[ L = U \Lambda U^T \]

\[
U = \begin{bmatrix}
  e_1^1 & e_1^2 & \ldots & e_1^K \\
  e_2^1 & & & \\
  & \ddots & & \\
  e_M^1 & e_1^2 & \ldots & e_1^K \\
\end{bmatrix}
\]

Construct an embedding of each of the \( M \) datapoints \( x^i \) through \( y^i \).
Reduce dimensionality by picking \( k < M \) projections \( y^i, i = 1 \ldots K \).

\[
y^i = \begin{bmatrix}
  e_1^i \\
  \vdots \\
  \vdots \\
  e^K_i \\
\end{bmatrix}
\]

When we have a fully connected graph, the entries in \( Y \) take any real value.
Spectral Clustering

Example: 3 datapoints in a graph composed of 2 partitions

\[
S = \begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

The similarity matrix is \( S \)

\( L \) has eigenvalue \( \lambda = 0 \) with multiplicity two.

One solution for the two associated eigenvectors is:

\[
e^1 = \sqrt{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad e^2 = \begin{bmatrix} 0.33 \\ -0.88 \\ -0.1 \end{bmatrix}
\]

An other solution for the two associated eigenvectors is:

\[
e^1 = \begin{bmatrix} 0.33 \\ 0.33 \\ -0.1 \end{bmatrix}, \quad e^2 = \begin{bmatrix} 0.99 \\ -0.1 \\ 1 \end{bmatrix}
\]

Entries in the eigenvector for the two first datapoints are equal.

\[
y^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad y^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

for the 1st set of eigenvectors

\[
y^1 = \begin{bmatrix} 0.33 \\ -0.1 \end{bmatrix}, \quad y^2 = \begin{bmatrix} 0.33 \\ -0.1 \end{bmatrix}
\]

for the 2nd set of eigenvectors.
Spectral Clustering

Example: 3 datapoints in a fully connected graph

The similarity matrix is \( S = \begin{bmatrix} 1 & 0.9 & 0.02 \\ 0.9 & 1 & 0.02 \\ 0.01 & 0.02 & 1 \end{bmatrix} \)

\( L \) has eigenvalue \( \lambda = 0 \) with multiplicity 1. The second eigenvalue is small \( \lambda_2 = 0.04 \), whereas the 3rd one is large, \( \lambda_3 = 1.81 \).

with associated eigenvectors:

\[
\begin{align*}
e^1 &= \sqrt{2} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \\
e^2 &= \begin{bmatrix} 0.411 \\ 0.404 \\ -0.81 \end{bmatrix}, \\
e^3 &= \begin{bmatrix} -0.8 \\ 0.7 \\ 0.0 \end{bmatrix}
\end{align*}
\]

\[
y^1 = \begin{bmatrix} 1 & 0.41 \end{bmatrix}, \\
y^2 = \begin{bmatrix} 1 & 0.40 \end{bmatrix}
\]

Entries in the 2nd eigenvector for the two first datapoints are almost equal.

The first two points have almost the same coordinates on the y embedding.

Reduce the dimensionality by considering the smallest eigenvalue.
Spectral Clustering

Example: 3 datapoints in a fully connected graph

The similarity matrix is

$$S = \begin{bmatrix} 1 & 0.9 & 0.8 \\ 0.9 & 1 & 0.7 \\ 0.8 & 0.7 & 1 \end{bmatrix}$$

$L$ has eigenvalue $\lambda = 0$ with multiplicity 1. The second and third eigenvalues are both large $\lambda_2 = 2.23$, $\lambda_3 = 2.57$.

with associated eigenvectors:

$$e^1 = \sqrt{2} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad e^2 = \begin{bmatrix} -0.21 \\ -0.57 \\ 0.79 \end{bmatrix}, \quad e^3 = \begin{bmatrix} -0.78 \\ 0.57 \\ 0.21 \end{bmatrix}$$

The 3rd point is now closer to the two other points.

Entries in the 2nd eigenvector for the two first datapoints are no longer equal.

The first two points have no longer the same coordinates on the y embedding.

$$y^1 = \begin{bmatrix} 1 \\ -0.21 \end{bmatrix}, \quad y^2 = \begin{bmatrix} 1 \\ -0.57 \end{bmatrix}$$
Spectral Clustering

Step 1: Embedding in \( y \)

Idea: Points close to one another have almost the same coordinate on the eigenvectors of \( L \) with small eigenvalues.

**Step 1:** Do an eigenvalue decomposition of the Lagrange matrix \( L \) and project the datapoints onto the first \( K \) eigenvectors with smallest eigenvalue (hence reducing the dimensionality).
Step 2:
Perform K-Means on the set of $y^1, \ldots, y^M$ vectors
Cluster datapoints $x$ according to their clustering in $y$. 
Equivalency to other non-linear Embeddings

Spectral decomposition of the similarity matrix
(which is already positive semi-definite)

\[
\Rightarrow y^i = \begin{bmatrix}
\sqrt{\lambda_1} e_i^1 \\
\vdots \\
\sqrt{\lambda_K} e_i^K \\
\end{bmatrix}
\]

In Isomap, the embedding is normalized by the eigenvalues and uses geodesic distance to build the similarity matrix, see supplementary material.
Laplacian Eigenmaps

Solve the generalized eigenvalue problem: $Ly = \lambda Dy$

Solution to optimization problem: $\min_y y^T Ly$ such that $y^T Dy = 1$.

Ensures minimal distorsion while preventing arbitrary scaling.

The vectors $y^i, \ i = 1...M$, form an embedding of the datapoints.
Equivalency to other non-linear Embeddings

kernel PCA: Eigenvalue decomposition of the matrix of similarity $S$

\[ S = U D U^T \]

The choice of parameters in kernel K-Means can be initialized by doing a readout of the Gram matrix after kernel PCA.
Kernel K-means and Kernel PCA

The optimization problem of kernel K-means is equivalent to:

$$\max_{H} \text{tr} \left[ H^T KH \right], \quad H = YD^{-2}$$

Since $$\text{tr} \left[ H^T KH \right] = \sum_{i=1}^{M} \lambda_i$$,

$$\lambda_i : M \text{ eigenvalues, resulting from the eigenvalue decomposition of the Gram Matrix}$$

See paper by M. Welling, supplementary document on website

Look at the eigenvalues to determine optimal number of clusters.

$$Y : M \times K \quad \text{Each entry of Y is 1 if the datapoint belongs to cluster k, otherwise zero}$$

$$D : K \times K \quad \text{D is diagonal. Element on the diagonal is sum of the datapoint in cluster k}$$
Kernel PCA projections can also help determine the kernel width

From top to bottom
Kernel width of 0.8, 1.5, 2.5
Kernel PCA projections can help determine the kernel width.

The sum of eigenvalue grows as we get a better clustering.
Quick Recap of Gaussian Mixture Model
Clustering with Mixture of Gaussians

Alternative to K-means; soft partitioning with elliptic clusters instead of spheres.

Clustering with Mixtures of Gaussians using spherical Gaussians (left) and non-spherical Gaussians (i.e. with full covariance matrix) (right).

Notice how the clusters become elongated along the direction of the clusters (the grey circles represent the first and second variances of the distributions).
Gaussian Mixture Model (GMM)

Using a set of $M N$-dimensional training datapoints

$$X = \{ x^i_j \}_{j=1, \ldots, N}^{i=1, \ldots, M}$$

The pdf of $X$ will be modeled through a mixture of $K$ Gaussians:

$$p(X) = \sum_{i=1}^{K} \alpha_i \cdot p(X | \mu^i, \Sigma^i), \quad \text{with } p(X | \mu^i, \Sigma^i) = N(\mu^i, \Sigma^i)$$

$\mu^i, \Sigma^i$: mean and covariance matrix of Gaussian $i$

Mixing Coefficients $\sum_{i=1}^{K} \alpha_i = 1$

Probability that the data was explained by Gaussian $i$:

$$\alpha_i = p(i) = \sum_{j=1}^{M} p(i | x^j)$$
Gaussian Mixture Modeling

The parameters of a GMM are the means, covariance matrices and prior pdf:

$$\Theta = \left\{ \mu^1, \ldots, \mu^K, \sum^1, \ldots, \sum^K, \alpha^1, \ldots, \alpha^K \right\}$$

Estimation of all the parameters can be done through *Expectation-Maximization* (E-M). E-M tries to find the optimum of the likelihood of the model given the data, i.e.:

$$\max_{\Theta} \ L(\Theta \mid X) = \max_{\Theta} \ p(X \mid \Theta)$$

See lecture notes for details
Gaussian Mixture Model
Gaussian Mixture Model

GMM using 4 Gaussians with random initialization
Gaussian Mixture Model

Expectation Maximization is very sensitive to initial conditions:

GMM using 4 Gaussians with new random initialization
Gaussian Mixture Model

Very sensitive to choice of number of Gaussians. Number of Gaussians can be optimized iteratively using AIC or BIC, like for K-means:

Here, GMM using 8 Gaussians
Evaluation of Clustering Methods
Evaluation of Clustering Methods

*Clustering methods rely on hyper parameters*

- Number of clusters
- Kernel parameters

→ Need to determine the goodness of these choices

*Clustering is unsupervised classification*

→ Do not know the real number of clusters and the data labels
→ Difficult to evaluate these choice without *ground truth*
Evaluation of Clustering Methods

Two types of measures: Internal versus external measures

Internal measures rely on measure of similarity (e.g. intra-cluster distance versus inter-cluster distances)

E.g.: Residual Sum of Square is an internal measure (available in mldemos); Gives the squared distance of each vector from its centroid summed over all vectors.

\[ \text{RSS} = \sum_{k=1}^{K} \sum_{x \in C_k} |x - \mu_k|^2 \]

→ Internal measures are problematic as the metric of similarity is often already optimized by clustering algorithm
Evaluation of Clustering Methods

K-Means, soft-K-Means and GMM have several hyperparameters:
(Fixed number of clusters, beta, number of Gaussian functions)

→ Measure to determine how well the choice of hyperparameters fit the dataset (maximum-likelihood measure)

\[ X : \text{dataset}; \quad M : \text{number of datapoints}; \quad B : \text{number of free parameters} \]

- Aikaike Information Criterion: \[ \text{AIC} = -2 \ln L + 2B \]
- Bayesian Information Criterion: \[ \text{BIC} = -2 \ln L + B \ln (M) \]

L: maximum likelihood of the model given B parameters

Choosing AIC versus BIC depends on the application:
→ Is the purpose of the analysis to make predictions, or to decide which model best represents reality?
AIC may have better predictive ability than BIC, but BIC finds a computationally more efficient solution.

Penalty for increase in computational costs
Evaluation of Clustering Methods

Two types of measures: Internal versus external measures

External measures assume that a subset of datapoints have class label and measures how well these datapoints are clustered.

→ Needs to have an idea of the class and have labeled some datapoints

→ Interesting only in cases when labeling is highly time-consuming when the data is very large (e.g. in speech recognition)
Evaluation of Clustering Methods

Raw Data
Semi-Supervised Learning

**Clustering F-Measure:**
(careful: similar but not the same F-measure as the F-measure we will see for classification!)

Tradeoff between clustering correctly all datapoints of the same class in the same cluster and making sure that each cluster contains points of only one class.

\[ M : \text{nm of datapoints}, \quad C = \{c_i\} : \text{the set of classes} \]
\[ K : \text{nm of clusters}, \]
\[ n_{ik} : \text{nm of members of class } c_i \text{ and of cluster } k \]

\[ F(C, K) = \sum_{c_i \in C} \frac{|c_i|}{M} \max_k \{F(c_i, k)\} \]

\[ F(c_i, k) = \frac{2R(c_i, k)P(c_i, k)}{R(c_i, k) + P(c_i, k)} \]

\[ R(c_i, k) = \frac{n_{ik}}{|c_i|} \]

\[ P(c_i, k) = \frac{n_{ik}}{|k|} \]

**Recall:** proportion of datapoints correctly classified/clusterized

**Precision:** proportion of datapoints of the same class in the cluster

Picks for each class the cluster with the maximal number of datapoints
Evaluation of Clustering Methods

RSS with K-means can find the true optimal number of clusters but very sensitive to random initialization (left and right: two different runs).

RSS finds an optimum for K=4 and K=5 for the right run.
Evaluation of Clustering Methods

BIC (left) and AIC (right) perform very poorly here, splitting some clusters into two halves.
Evaluation of Clustering Methods

BIC (left) and AIC (right) perform much better for picking the right number of clusters in GMM.
Evaluation of Clustering Methods

Raw Data
Evaluation of Clustering Methods

Optimization with BIC using K-means
Evaluation of Clustering Methods

Optimization with AIC using K-means

AIC tends to find more clusters
Evaluation of Clustering Methods

Raw Data
Evaluation of Clustering Methods

Optimization with AIC using kernel K-means with RBF
Evaluation of Clustering Methods

Optimization with BIC using kernel K-means with RBF
Semi-Supervised Learning

Raw Data: 3 classes
Semi-Supervised Learning

Clustering with RBF kernel K-Means after optimization with BIC
Semi-Supervised Learning

After semi-supervised learning
Summary

We have seen several methods for extracting structure in data using the notion of kernel and discussed their similarity, differences and complementarity:

• Kernel CCA is a generalization of kernel PCA to determine partial grouping in different dimensions.

• Kernel PCA can be used too bootstrap choice of hyperparameters in kernel K-means.

We have compared the geometrical division of space yielded by Rbf and Polynomial kernels.

We have seen that simpler techniques than kernel K-means such as K-means with norm-p and mixture of Gaussians can yield complex non-linear clustering.
Summary

When to use what? E.g. k-means versus kernel K-means versus GMM

When using any machine learning algorithm, you have to balance a number of factors:

• Computing time at training and at testing
• Number of open parameters
• Curse of dimensionality (order of growth with number of datapoints, dimension, etc)
• Robustness to initial condition, optimality of solution