Spectral Clustering

Introduction

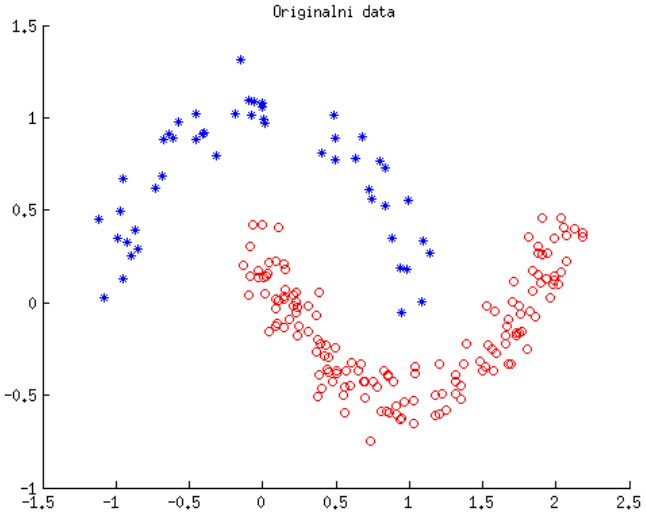
The aim of this tutorial is to get familiar with spectral clustering. You will use available building blocks and implement algorithm of spectral clustering. You will apply this algorithm on input data and compare the result with known annotation and with the result of classic k-means algorithm. You will perform the comparison for different parametrisations of input data and algorithm of spectral clustering.

1 Input data

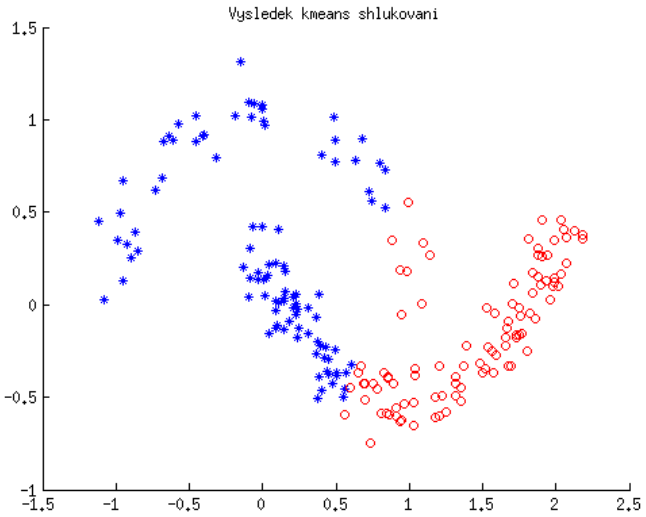
Input data are generated using function *GenerateData.m*. The type of the data is prefixed (crescent), number of crescents ($k = 2$) and number of real attributes ($n = 2$). Annotation G (membership to clusters, i.e. identification of the random process, which was used to generate respective crescent) is available for purpose of evaluation: $\mathcal{X} \subset \mathbb{R}^2 \rightarrow G = \{G_1, G_2\}$. You can modify variance within clusters and proportion of samples in clusters. We can consider the task with more noise and unequal number of samples in clusters as more difficult. Visualisation of a typical output is shown in figure 1.

2 k-means Algorithm

Here, we are dealing with a clustering task with non-compact clusters. Algorithm k-means will probably generate clusters different from the golden standard (application of Matlab function *kmeans* directly on input data - see figure 2). Since k-means algorithm is also the last step of spectral clustering, one of our aims is to find out how transformation performed in spectral clustering before application of k-means algorithm influences its output.



Obrázek 1: Two crescents ($n = 200$, $Pr(G) = [0.2, 0.8]$, $\sigma^2 = 0.01$).



Obrázek 2: Two crescents - clusters using k-means algorithm.

3 Spectral Clustering

Spectral clustering can be divided into a few building blocks. The following list describes these functional blocks and defines what is available at the beginning and what functionality has to be implemented:

- computation of similarity matrix $\mathcal{S}$: available in function *CalcSimMatrix.m*, implementation calculates Euclidean distances between pairs of points and afterwards applies Gaussian kernel, modification is not required, but it is good to verify the importance of parameter σ (a higher value means greater similarity between more distant points, i.e. similarity is less local),

- construction of similarity graph: a trivial variant is to keep $\mathcal{S}$ without changes (complete similarity graph), function *BuildEpsilonGraph.m* generates similarity graph based on ϵ-neighbourhood; complete function *BuildKNNGraph.m*, which will generate both of the two variants of similarity graphs based on the k-nearest neighbours (symmetric variant: connect i and j if i is one of the k-nearest neighbours of j or vice versa, undirected variant: connect i and j if i is one of the k-nearest neighbours and vice versa), function *BuildDirectedKNNGraph.m* is available, which generates orientated similarity graph; you can use function *PlotGraph.m* to check correctness,

- derivation of Laplace matrices $\mathcal{L}$ with subsequent projection to the space of their k smallest eigenvectors: function *CalcLaplacian.m* performs this step for non-normalized Laplacian, you are expected to complete at least one variant of computation of normalized Laplacian (see [1]),

- application of k-means algorithm on the output of the previous function: a straightforward application of Matlab function *kmeans*.

4 Step by Step

You should go through the following steps:

1. select suitable parameters and generate input data (to check the output use function *PlotData.m* which displays scatter plot (see figures in this text)),

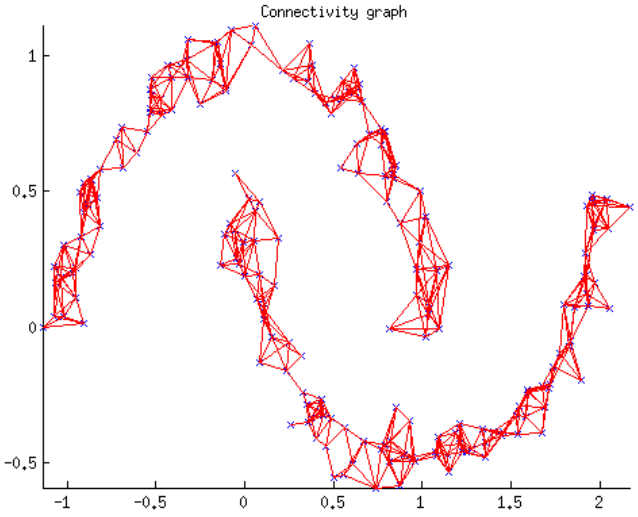
2. perform clustering using k-means algorithm, evaluate its success rate visually using function *PlotData.m* and numerically using function *Purity.m*,
3. create the basic variant of the algorithm of spectral clustering from the existing functions,
4. perform clustering using spectral clustering, check its success visually using function *PlotData.m* and numerically using function *Purity.m*,
5. implement function *BuildKNNGraph.m*, to check it use function *PlotGraph.m*,
6. implement extension of function *CalcLaplacian.m*, alternatively implement a new function *CalcNormLaplacian.m*,
7. repeat step 4 with different variants of the spectral clustering algorithm(different variants of similarity graph and Laplacian), change parameters of input data generated in step 1,
8. summarize your experience from experiments (what option is important and what option does not have influence, how optimal option is related to the difficulty of input data).

5 Expected results

Unlike k-means algorithm, spectral clustering should achieve perfect match with the golden standard. It should be possible to achieve this even without implementing function extension (selecting σ or ϵ properly should be enough). To check correctness you should create a similarity graph, which has exactly two connected components corresponding to crescents (see figure 3, the output of function *plotGraph.m*). The existing of two components corresponding to real annotations is not a necessary condition for a successful solution, the graph can be also connected.

Similarity graph based on ϵ-neighbourhood is not suitable for input data with unequal distribution of samples among clusters. The density of points in different parts of the space is different, an universal ϵ cannot be found. The approaches based on k-nearest neighbours are more suitable. The symmetric variant can connect points from regions with different density of points. Undirected variant divides regions with constant density of points.

An important criterion for selecting Laplacian is the degree-distribution of vertices in similarity graphs. In case of regular graphs (all vertices have similar degree) the results should be similar for non-normalized and normalized



Obrázek 3: Two crescents - ideal similarity graph with two connected components.

variants. Otherwise, it is suitable to normalize. In general, the normalized variant should not be worse than the simplest non-normalized variant. Apart from simplicity there is no reason to use it.

Parameter selection for spectral clustering is not trivial. Some heuristic rules for automatic selection are available in [1].

Reference

[1] Luxburg, Ulrike: *A tutorial on spectral clustering*, Statistics and Computing, 17/4, pp. 395–416, 2007.