COMPUTATIONAL INTELLIGENCE

Clustering Algorithms for AI, CI, KE and Data Mining

Adrian Horzyk

horzyk@agh.edu.pl

AGH

AGH University of Science and Technology Krakow, Poland

Clustering

Clustering

- is a data mining technique and a group of machine learning methods that involves the grouping of data points taking into account the similarity of these points in a hyperspace of input data;
- does not require input data to be labeled as in the classification tasks;
- can group data into various groups, and next, these groups can be labeled;
- use unsupervised learning to adapt a model.

Main rule of clustering: Data points of the same cluster (group) should be similar, i.e. have similar properties and/or features, while data points in different clusters (groups) should be dissimilar, i.e. have highly dissimilar properties and/or features.

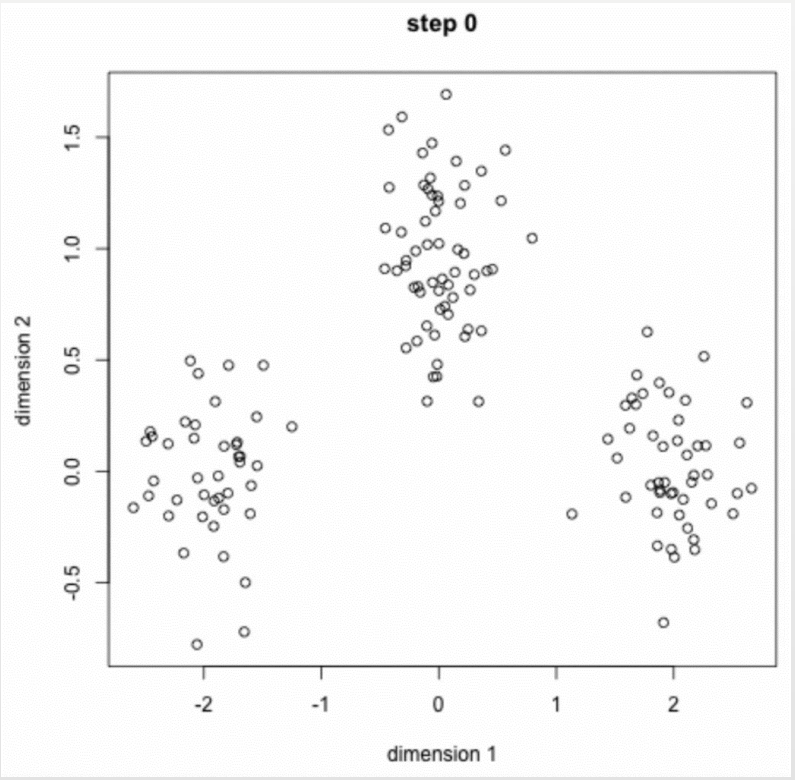
Clustering methods can be:

- Strong – data points from different clusters must be separable, i.e. each belongs to only a single cluster (clusters cannot overlap);
- Weak – clusters can share some data points (clusters can overlap).

K-Means and K-Medians Clustering

K-Means Clustering

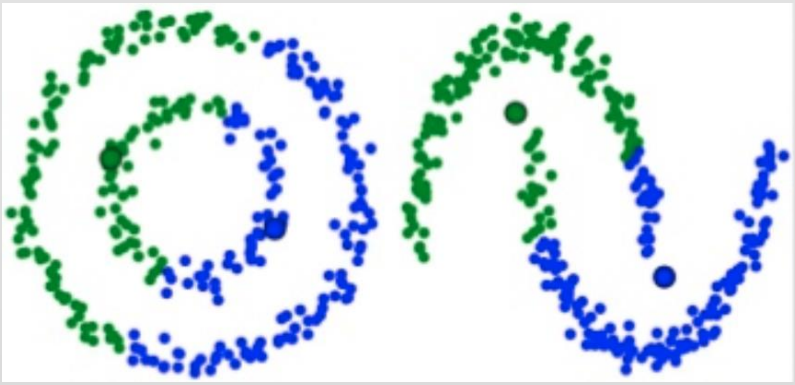
- **is probably the most well-known clustering algorithm;**
- **is easy to understand and implement;**
- **requires to set up the number of clusters first;**
- **produces different results according to the starting cluster centers;**
- **fails in cases where clusters are not circular.**



K-Means Algorithm

1. Select a number of clusters (k) and randomly initialize their respective center points that are vectors of the same length as data point vectors.
2. Each data point is encompassed to the cluster point (group center) that distance to this data point is the smallest, i.e. each data point is assigned to the closest cluster (its center).
3. Based on these assigned points, we compute the group center by taking the mean of all the points in the group.
4. Repeat these steps until the group centers do not change much between subsequent iterations.

You can also randomly initialize the group centers a few times, and then rerun the algorithm to find the best clusters.



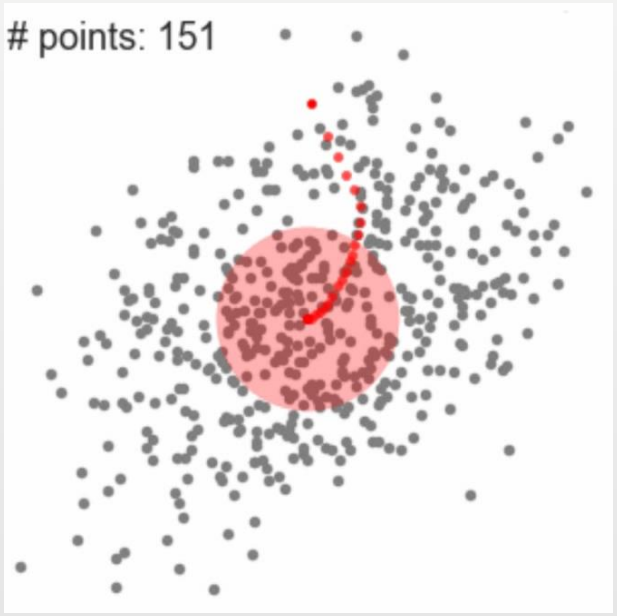
K-Medians Clustering

- **is very similar to K-Means Clustering;**
- **computes the cluster centers using the median instead of the mean;**
- **is less sensitive to outliers;**
- **but it is slower because it requires sorting of cluster points to compute medians.**

Mean-Shift Clustering

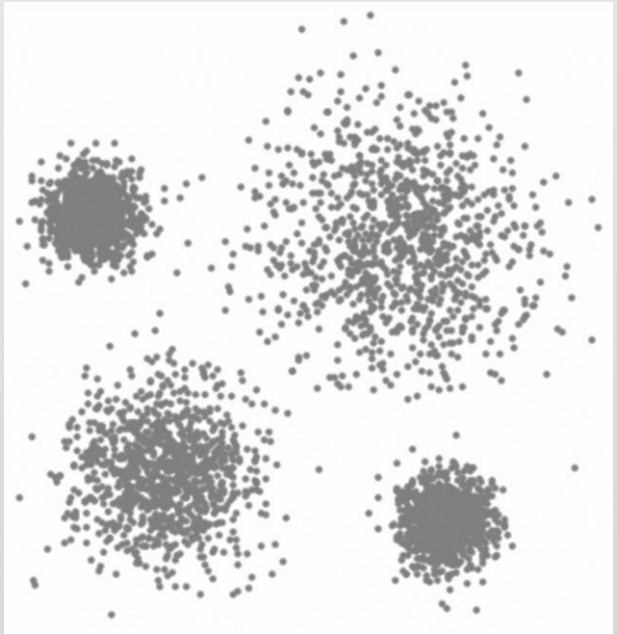
Mean-Shift Clustering

- **Is a hill climbing and sliding-window-based algorithm that attempts to find dense areas of data points in each step until convergence.**
- **Is trying to locate the center points of clusters by updating candidates for center points to be the mean of the points within the sliding window. Means automatically attract the sliding windows.**
- **These candidates are then filtered to eliminate near duplicates, forming the final set of center points and their corresponding clusters.**
- **Each black dot in the animation represents the centroid of a sliding window and each gray dot is a data point.**
- **The selection of the window radius of clusters is not a trivial task.**



Mean-Shift Algorithm

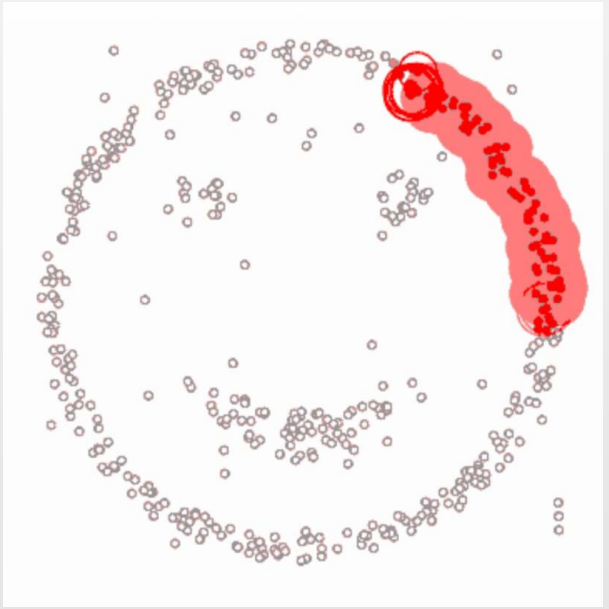
1. Begin with a circular sliding window centered at a randomly selected point C having a radius r as the kernel.
2. In every iteration, the sliding windows are shifted towards regions of higher density by shifting the center point to the mean of the points within the window. The density within the sliding window is proportional to the number of points inside it.
3. Continue shifting the sliding of the windows according to the means (red points) until there are no directions at which they can be shifted and accommodate more points inside the kernels. In the figure, keep moving the circle until we can no longer increase the density, i.e. the number of points in this window.
4. Repeat this process until all points lie within windows. When multiple sliding windows overlap, the window containing the most points is preserved. The data points are then clustered according to the sliding window in which they reside.



Density-Based Spatial Clustering

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

- **Is a density-based clustering algorithm:**



1. Begin with an arbitrary starting data point that has not yet been visited. Consider a neighborhood of this point defined by the circle with radius ε.
2. If there are a sufficient number of points (more than minPointsNumber) within this neighborhood then the clustering process starts and the current data point becomes the first point in a new cluster. Otherwise, the point will be labeled as noise (later this noisy point might become the part of any cluster).
   In both cases that point is marked as “visited”.
3. For this first point in the new cluster, the points within its ε distance neighborhood also become part of the same cluster. This procedure of making all points in the ε neighborhood belong to the same cluster is then repeated for all of the new points that have been just added to the currently developing cluster.
4. The process is repeated until all points in the cluster are determined,
   i.e. all points within the ε neighborhood of the cluster have been visited and labeled as „visited”.
5. Once we are done with the current cluster, a new unvisited point is retrieved and processed in the same way, creating another cluster or a noise point. This process repeats until all points are marked as visited and belong to a cluster or being a noise.

DBSCAN poses many great advantages over other clustering algorithms:

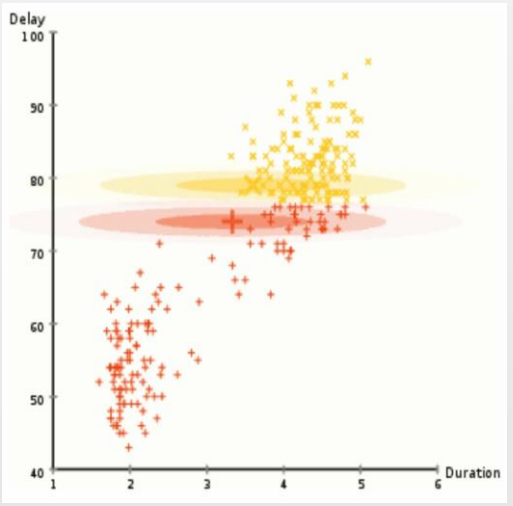
- **It does not require a preset number of clusters.**
- **It identifies outliers as noise.**
- **It finds arbitrarily sized and arbitrarily shaped clusters quite well.**

The main drawback of DBSCAN is that it does not perform as well as other algorithms when the clusters are of varying density. This is because the setting of the distance threshold ε and minPointsNumber for identifying the neighborhood points will vary from cluster to cluster when the density varies. This drawback occurs with high-dimensional data since again the distance threshold ε becomes challenging to estimate.

Expectation-Minimization Clustering

Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)

- **assumes that the data points are Gaussian distributed, i.e. we have two parameters to describe the shape of the clusters: the mean and the standard deviation. Therefore, the clusters can take any kind of elliptical shape, and each Gaussian distribution is assigned to a single cluster.**
- **In order to find the parameters of the Gaussian distribution for each cluster (i.e. the appropriate mean and standard deviation), we use an Expectation–Maximization (EM) optimization algorithm.**



Expectation–Maximization Algorithm

1. Select a number of clusters and randomly initialize the Gaussian distribution parameters for each cluster.
2. For the given Gaussian distributions for each cluster, compute the probabilities of belonging of data points to particular clusters. The closer a point is to the Gaussian's center, the more likely it belongs to that cluster. This should make sense since with a Gaussian distribution, we assume that most of the data lie closer to the center of the cluster.
3. Based on these probabilities, we compute a new set of parameters for the Gaussian distributions such that we maximize the probabilities of data points within the clusters. We compute these new parameters using a weighted sum of the data point positions, where the weights are the probabilities of the data point belonging in that particular cluster.
4. The above steps are iteratively repeated until convergence, where the distributions do not change much from iteration to iteration.

There are two key advantages to using GMMs:

- **GMMs (ellipses) are a lot more flexible in terms of cluster covariance than K-Means (circles) due to the standard deviation parameter. K-Means is actually a special case of GMMs in which each cluster's covariance along all dimensions approaches 0.**
- **Since GMMs use probabilities, they can have multiple clusters per data point. So if a data point is in the middle of two overlapping clusters, we can simply define its class by the probability of belonging to class 1 and another probability of belonging to class 2, i.e. GMMs support mixed membership.**

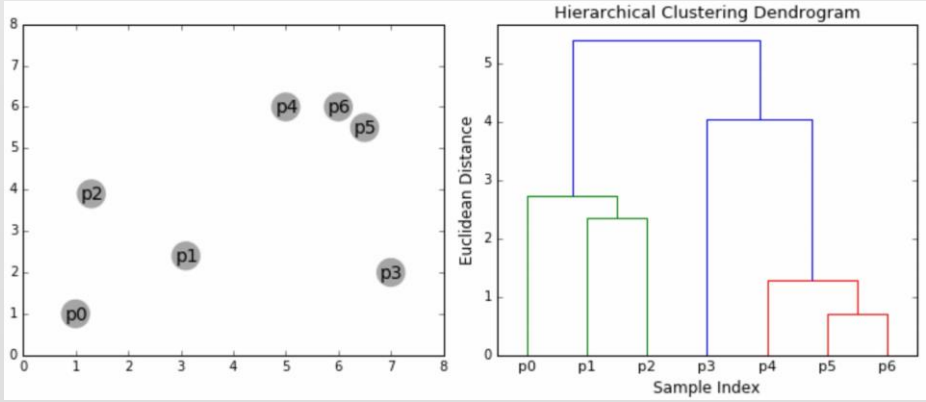
The disadvantage is that EM Clustering requires preset the number of clusters.

Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering

- **Is a bottom-up algorithm which treats each data point as a single cluster at the outset and then it successively merges (agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all data points. This hierarchy of clusters is represented as a tree (dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.**

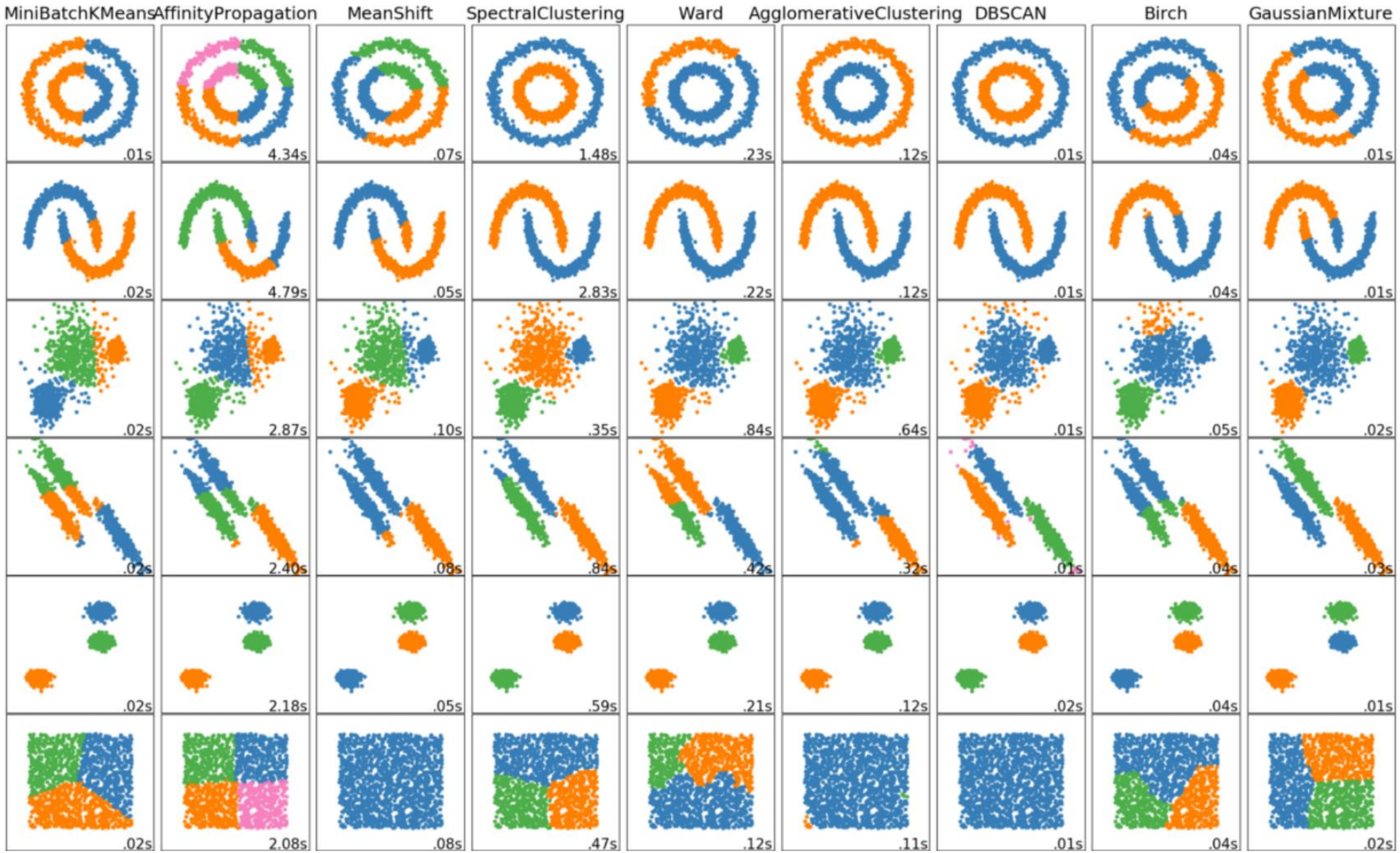
1. At the beginning, treat each data point as a single cluster, i.e. if there are N data points in our dataset then we have N clusters. Then select a distance metric that measures the distance between two clusters. Use an average linkage which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.
2. On each iteration, combine two clusters into one when the clusters have the smallest average linkage, i.e. according to our selected distance metric, these two clusters have the smallest distance between each other and therefore are the most similar and should be combined.
3. Repeat step 2 until the root of the tree is reached, i.e we achieved one cluster that contains all data points.
   In this way, we can select how many clusters we want to have in the end, simply by choosing when to stop combining the clusters, i.e when we stop building the tree.



- **Hierarchical clustering does not require to specify the number of clusters, and we can even select which number of clusters looks best after we have built the tree. The algorithm is not sensitive to the choice of distance metric; all of them tend to work well whereas the choice of distance metric is critical for other clustering algorithms.**
- **A particularly good use case of hierarchical clustering methods is when the underlying data has a hierarchical structure and you want to recover this hierarchy. The cost of hierarchical clustering has time complexity of $O(N^3)$, unlike the linear complexity O(N) of K-Means and GMM.**

Comparison of Clustering Algorithms

Here, we can notice the pros and cons of the different clustering algorithms on difficult data points:



Biclustering and Triclustering

Biclustering

- called also block clustering, co-clustering, or two-mode clustering is a data mining technique that allows simultaneous clustering of the rows and columns of a matrix;
- is searching for the similarities of these points in a space of input data taking into account only a subset of attributes (e.g. rows);
- does not require input data to be labeled as in the classification tasks;
- can group data into various groups, and next, these groups can be labeled;
- use unsupervised learning to adapt a model.

Definition: Given a set of *m* samples represented by an n-dimensional feature vector, the entire dataset can be represented as m rows in n columns (i.e., an m × n matrix). The biclustering algorithm generates biclusters – a subset of rows which exhibit similar behavior across a subset of columns, or vice versa.

Triclustering

- additionally takes into account the time and searches for patterns that frequently repeat in time. Clusters of such frequently repeated patterns spanned over time will be the result of triclustering.

Biclustering Types

1.Bicluster with constant values

When a biclustering algorithm tries to find a constant bicluster, the normal way for it is to reorder the rows and columns of the matrix so it can group together similar rows/columns and find biclusters with similar values. This method works well when the data are tidy. But as the data can be noisy most of the times, it cannot satisfy us.

More sophisticated methods should be used. A perfect constant bicluster is a matrix(I,J) where all values a(i,j) are equal to μ. In real data, a(i,j) can be seen as n(i,j) + μ where n(i,j) is the noise. According to the Hartigan's algorithm, by splitting the original data matrix into a set of biclusters, variance is used to compute constant biclusters. So, a perfect bicluster is a matrix with variance zero.

Also, in order to prevent the partitioning of the data matrix into biclusters with only one row and one column, Hartigan assumes that there are K biclusters within the data matrix. When the data matrix is partitioned into K biclusters, the algorithm ends.

Biclustering Types

2.Biclusters with constant values on rows or columns

This kind of biclusters cannot be evaluated just by variance of its values.

To finish the identification, the columns and the rows should be normalized at first.

There are other algorithms, without normalization step, which can find biclusters that have rows and columns with different approaches.

3.Biclusters with coherent values

For biclusters with coherent values on rows and columns, there is used an analysis of variance between groups, using co-variance between both rows and columns.

Cheng and Church's defined a bicluster as a subset of rows and columns with almost the same score. The similarity score is used to measure the coherence of rows and columns.

a) Bicluster with constant values

2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0

b) Bicluster with constant values on rows

1.0	1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0	4.0
5.0	5.0	5.0	5.0	5.0

c) Bicluster with constant values on columns

1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0

d) Bicluster with coherent values (additive)

1.0	4.0	5.0	0.0	1.5
4.0	7.0	8.0	3.0	4.5
3.0	6.0	7.0	2.0	3.5
5.0	8.0	9.0	4.0	5.5
2.0	5.0	6.0	1.0	2.5

e) Bicluster with coherent values (multiplicative)

1.0	0.5	2.0	0.2	0.8
2.0	1.0	4.0	0.4	1.6
3.0	1.5	6.0	0.6	2.4
4.0	2.0	8.0	0.8	3.2
5.0	2.5	10.0	1.0	4.0

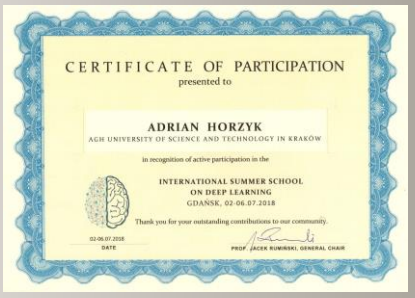
Bibliography and Literature

1. R. Rojas, *Neural Networks*, Springer-Verlag, Berlin, 1996.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2016, ISBN 978-1-59327-741-3 or PWN 2018.
3. Holk Cruse, *Neural Networks as Cybernetic Systems*, 2nd and revised edition
4. *https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68*
5. *https://en.wikipedia.org/wiki/Biclustering*

Adrian Horzyk

horzyk@agh.edu.pl

Google: Horzyk



University of Science and Technology in Krakow, Poland