Clustertools - An overview and comparison of different clustering methods in Python

Johannes von Lindheim, Mattes Mollenhauer and Luzie Helfmann

July 16, 2017

1 Introduction

Cluster analysis of numerical and categorical data is a key task in unsupervised learning. Due to its general applicability, it has been successfully used in a big variety of fields such as image processing (see for example [11]), bioinformatics [17], time series modeling [16], economical models and many more. It has also been a major focus in the Machine Learning research in the last years, henceforth spawning a wide range of methodologies and approaches associated to different clustering algorithms.

Given a sample $x_1, \dots, x_n$ of objects in a set X and (usually) a pairwise defined similarity measure $s(x_i, x_j)$, the goal of clustering is to define a set of clusters (a clustering) $\mathcal{C} = \{C_1, \dots, C_k\}$, such that one is able to map the x_i either directly onto the clusters C_j or the probability vectors in the k-dimensional probability simplex associated to the clusters C_j (fuzzy clustering). The resulting categorization of the data should then yield information about the intrinsic structures of either the given sample or the set X.

Note that this very general problem setting might be interpreted in various ways depending on the set X and the similarity measure s, for example as discovering structures in a (weighted) graph defined by s on the vertices x_i. The classical clustering problem is usually given in a metric space $(X, dist)$ with either $s = dist$ or s being a map given in terms of $dist$. Throughout this paper, n will denote the number of d-dimensional (if given in a finite dimensional vector space) data points, k the number of clusters and $iter$ the total iteration count for iterative methods.

We set ourselves the task to implement and compare various clustering algorithms, ranging from well known and classical ones to newer clustering techniques like consensus clustering with less theoretical results. Our Python package Clustertools provides a collection of clustering algorithms as well as test data sets [7] and some example jupyter notebooks. The code relies only on NumPy and SciPy and can be found on GitHub [9].

This paper is structured as follows: in Section 2 we will first give a brief overview on the clustering approaches and methodologies implemented in Clustertools. In section 3.1 we will compare the behaviour of these clustering algorithms on different data sets, their computational complexity and discuss their advantages and disadvantages. We will also present some results and new ideas about consensus clustering.

2 Clustering Algorithms

2.1 Distance-based clustering

2.1.1 Regular space clustering

Regular space clustering was proposed in [17] as a method for state space discretization to estimate Markov models from molecular trajectory data. The requirements for these purposes differ a lot from the classical requirements that one would intuitively give for clustering algorithms. Regular space clustering was not mainly designed to give a crisp and clear understanding of the structures in the data, but to be fast and to decompose time series data in a metric space into (usually many small) state components for a model estimation pipeline. Regular space clustering is one of the fastest clustering algorithms available.

Regular space clustering iterates through the data points x_i step-by-step and checks, if the current data point is further away than a minimal distance ϵ_{min} from all available clusters determined before. If so, it is chosen as a cluster center itself. The cluster assignment for the non-center points is done in a Voronoi

fashion (see figure 1) with respect to the given metric. It is important to note that the order of the iteration through the data points plays a fundamental role in the outcome of regular space clustering, since it was designed specifically for time series data. Furthermore, due to the Voronoi cluster assignment, it is well suited for blob-shaped and convex structures in clustering problems.

2.1.2 K-Means

The K-Means problem has originally been described in [12] and [13] as the minimization of the objective function

$$\sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 \tag{1}$$

with respect to data points x_j by finding the optimal cluster centers μ_i corresponding to clusters C_i. The assignments $x_j \in C_i$ are to be understood in a Voronoi-style with respect to the centers μ_i. The solution to this problem can be approximated in various ways. The classical K-Means approach is the so called Lloyd-type iteration or Lloyd-Forgy method, where the centers $\mu_1, \ldots, \mu_k$ are initialized by randomly choosing k members of the given dataset X and then updating them successively as the barycenters of the resulting Voronoi clusters. However, note that this iteration is not deterministic and that K-Means is prone to only finding local optima of the given objective expression. There have been several approaches to solve this problem. The most commonly used method is K-Means++, where the initializations of the cluster centers μ_i are chosen with respect to a discrete probability distribution defined on the data set as described in [2].

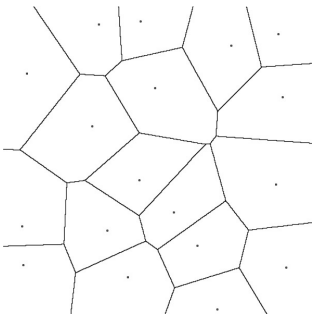

Figure 1: Voronoi diagram: cluster labels are assigned with respect to the nearest cluster center. For the euclidean space, this results the classical Voronoi partition diagram. [20]

K-means can be seen as a hard clustering limiting case of Fuzzy C-Means, see figure 4 and section 2.4. It is one of the most used clustering algorithms with a lot of possible modifications and theoretical results on complexity and iteration counts. K-Means is suited well for data containing a known (or robustly estimated) number of convex or blob-shaped structures.

2.2 Density-based clustering

2.2.1 DBSCAN

Density Based Spatial Clustering of Applications with Noise (DBSCAN, first proposed in [6]) is the most commonly used density based clustering approach in metric spaces. Given a density specification consisting of a radius parameter ϵ and a minimal point count parameter m_{points}, the algorithm iterates over the dataset and searches clusters of density-connected structures, i.e. areas that satisfy a number of at least m_{points} in a ϵ-neighborhood of given cluster members and expanding the detected clusters successively. Points which can not be assigned to any detected cluster with the given density settings will be labeled as noise. Note that DBSCAN is fully capable of detecting highly non-convex density-connected structures. However, DBSCAN is extremely sensitive to its parameters and might yield completely different clusters with only a slight parameter change. DBSCAN is suited well for density-homogeneous problems but will inevitably fail on data containing structures of different point densities. Note that the concept of "noise" in the data set is fundamentally determined by the choice of parameters and can quickly become unintuitive or misleading for suboptimal parameter choices.

2.2.2 Mean shift

The Mean Shift clustering approach is an intuitive approach towards finding groups in sets of data. It builds on the idea that the data points $x_i \in \mathbb{R}^d,\ i = 1, \ldots, n$ represent samples from some underlying probability density function. In that way, dense regions correspond to clusters, local maxima correspond to cluster centers and points in sparse areas might be outliers or noise. The mean shift algorithm is an iterative algorithm in which the data points are shifted in the direction of maximum increase in the datasets' density until convergence [4]. Data points shifted to the same local maximum belong to the same cluster. See figure 2 for a visualization of those steps. Given some distance metric, kernel and bandwidth parameter, the density can be estimated using kernel density estimation (KDE). Choosing a

sensible bandwidth parameter is essential for a good clustering outcome. If the bandwidth parameter is too small this will result in eventually a cluster center being placed at every data point. On the other hand, if the bandwidth parameter becomes very big, the density estimation is smoothed out with just one peak and thus one cluster for all data points. We implemented some basic bandwidth estimators [19], they rely on the data being pre-processed (i.e. scaled to having mean 0 and standard deviation 1).

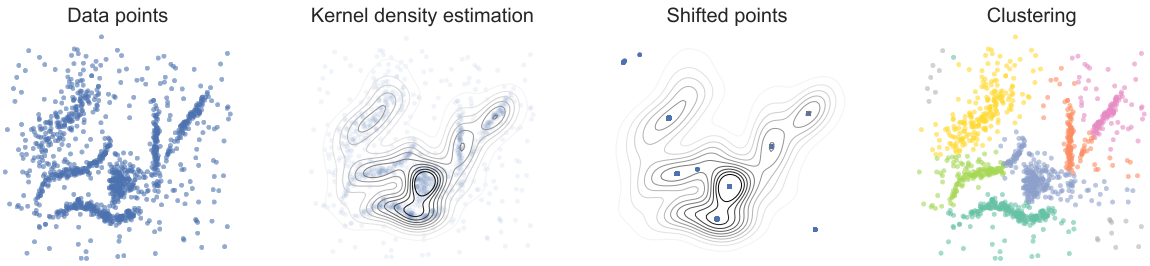

Figure 2: Mean Shift Clustering, data points are shifted to peaks in the point density and then grouped to clusters.

2.3 Similarity/graph-based clustering

2.3.1 Spectral clustering

Spectral clustering is an abstract clustering technique that is defined on a given graph G. Each vertex of G corresponds to a point in the data set, the edges of G can be interpreted as similarities or distances between the data points. Therefore, the big advantage of spectral clustering lies in the huge variability of applications. It is able to process data with abstract similarity definitions just as well as data in metric spaces in combination with (unweighted) k-Nearest-Neighbor graphs, weighted distance graphs, RBF similarity graphs and lot a more.

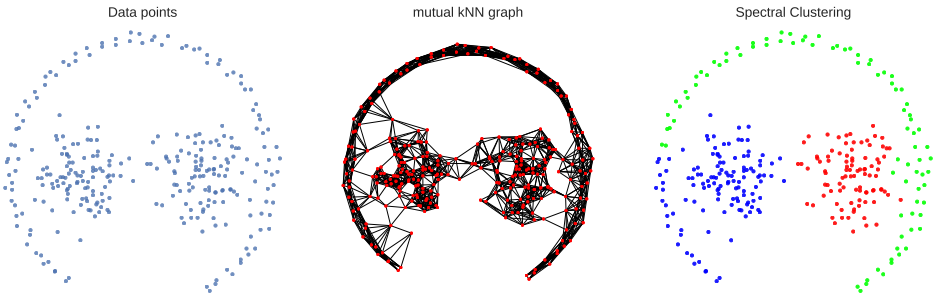

Figure 3: Building a kNN-graph from data and applying Spectral clustering.

From G, a graph Laplacian matrix is constructed. In [14], the normalized Laplacian $L = I - D^{-1/2}AD^{-1/2}$ is proposed, where A is the adjacency matrix corresponding to the graph G and D is the nodal degree matrix containing vertex degrees on the diagonal. This technique is commonly known as the Normalized-Cuts algorithm. Depending on the choice of the Laplacian and number k of clusters to be determined, a generalized eigenproblem is solved and the state space reduced to the eigenspace spanned by the k smallest eigenvalues, not including 0. The actual clustering itself happens in the reduced space (usually via K-Means). Due to the big amount of possible modifications in the underlying graph generation, the computation algorithm itself, its close relationship to dimensionality reduction and fairly complex mathematical background, spectral clustering is one of the most important research topics in cluster analysis. There have been several approaches to link spectral clustering to other Machine Learning related methods, for example Kernel K-Means (see [5]). Spectral clustering can be used to produce non-convex cluster outcomes. Note that for the classical spectral clustering approaches, the number of clusters k must be known. Also, the computationally expensive solution of the generalized eigenproblem can yield numerical pitfalls and may need matrix preconditioning or similar features.

2.3.2 Affinity Propagation

Although depending on graph and similarity based structures much like spectral clustering, Affinity Propagation relies on a completely different strategy. Given a similarity matrix $S = (S_{ij})$ that describes pairwise similarities between data points, an availability matrix A and a responsibility matrix R are iteratively updated by matrix computations proposed in [8]. For every data point x_i, the availability matrix A captures the relevance for all other points x_j to be a representative "exemplar" for x_i under the given similarity measure. Conversely, the responsibility matrix R captures how relevant the representativity of a point x_i is for all other points x_j. Similar to K-Means, one usually prescribes a maximal iteration count. After the iteration, a number of exemplars is determined from information aggregated in A and R. These exemplars serve as cluster "centers" and the remaining points are assigned via the similarity matrix S as corresponding to the exemplar with the closest similarity.

When working with data in a metric space, one usually prescribes a similarity measure with a range in $(-\infty, 0]$ such as $S_{ij} = s(x_i, x_j) = -d(x_i, x_j)^2$ to construct the similarity matrix S for $i \neq j$. An important feature are the specifically determined diagonal entries S_{ii} of S. A high value of S_{ii} corresponds to a high relevance of x_i as an exemplar, analogously for low values in S_{ii}. Constant values $S_{ii} = c$ for all i lead to evenly distributed relevance. One is also able to control the sensitivity of the algorithm with the choice of c: for small c, Affinity Propagation tends to yield a smaller number of cluster centers, conversely bigger c lead to a higher sensitivity. A common choice for c is the median of all off-diagonal entries in S.

2.3.3 Hierarchical clustering

Hierarchical clustering tries to find a hierarchy of clusters. We implemented the agglomerative, i.e. "bottom-up-approach", which first puts every point into its own cluster. Then in every iteration it merges the two closest clusters together. To do so, one introduces a linkage distance $l : \{C_1, \dots, C_k\} \times \{C_1, \dots, C_k\} \to \mathbb{R}_0^+$ on the clusters. This distance relies solely on a distance or dissimilarity function $d : \{x_1, \dots, x_n\} \times \{x_1, \dots, x_n\} \to \mathbb{R}_0^+$ between the data points.

The algorithm terminates, when either the number of clusters has reduced to a number given by the user, or when the smallest distance $\min_{i,j} l(C_i, C_j)$ between any two clusters exceeds a threshold given by the user, or the first one of them, or both.

There are numerous choices for the linkage distance l. In our package, we implemented three of the most popular ones. Single resp. complete linkage, considers the minimum resp. maximum distance of all pairs of points from any two fixed clusters C_i, C_j. Moreover, average linkage obviously takes the average distance of all point pairs.

For more details, see e.g. [10].

2.4 Fuzzy Methods

So far we only looked at examples of hard clustering, where each data point is assigned to exactly one cluster. Thus the membership value which is the degree of assignment $u_j(x_i) = u_{ji}$ of a point x_i to a cluster C_j is $u_{ji} \in \{0, 1\}$ where $i = 1, \dots, n,\ j = 1, \dots, k$. In fuzzy clustering (sometimes also called soft clustering) on the other hand, a point has some degree of membership to each cluster, so that membership values are $u_{ji} \in [0, 1]$ with the constraint that $\sum_{j=1}^{k} u_{ji} = 1\ \forall\ x_i$.
Fuzzy clustering gives particularly good results when the data set is "fuzzy", i.e. when there are many points that might belong to several or no cluster at all. Outliers (points that don't belong to a cluster) or noise can be understood as points where the highest membership value to some cluster is still quite low.
The most popular fuzzy clustering method is Fuzzy C-Means (see figure 4). It can be seen as an extension of K-Means. The cluster centers (necessary to compute the membership matrix) are found by minimizing the squared error which is induced by representing data points x_i by cluster center v_j weighted with the membership value u_{ij} [3]. The objective function to minimize is thus

$$J_m(U, v) = \sum_{i=1}^{n} \sum_{j=1}^{k} (u_{ji})^m\ \|x_i - v_j\|^2, \tag{2}$$

where the norm has to be chosen appropriately and $m \geq 1$ is the fuzzier (determines level of fuzziness).The limiting cases for m are $m = 1$, in which case the algorithm converges to hard clustering (K-Means) and $m \to \infty$, in which case each data point will have the same membership value $1/c$ to every cluster.

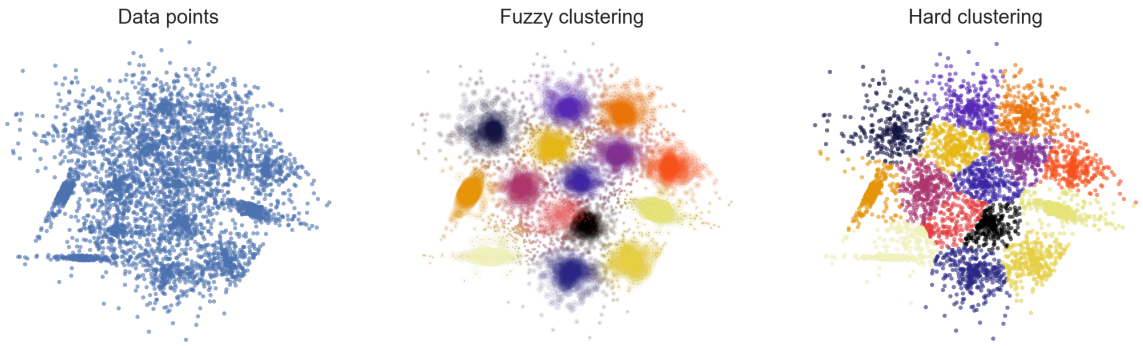

Figure 4: Fuzzy C-Means on Gaussian blobs data set, the size and the colour of the circles represent the membership degree of a point to a certain coloured cluster. Outliers can be recognized by their low membership value to all clusters.

2.5 Consensus Clustering

The idea of consensus clustering is to find a clustering $\mathcal{C}^*$ when a set of clusterings $\Lambda = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ is given. As the name suggests, $\mathcal{C}^*$ should represent a "consensus" or common denominator among all given clusterings. There are several reasons, which motivate this:

- One wants to obtain a clustering method by mixing several algorithms, which is more robust on different datasets opposed to just one algorithm. Examples are considered in 3.2.1.
- Knowledge reuse: Consider the case that multiple clusterings are available but not the features that were considered in their calculation (e.g. because they were created by human experts or proprietary companies). This knowledge can be reused and unified in a consensus clustering. See e.g. [18].
- Distributed computing: One splits the number of features or objects and produces a consensus clustering from that ensemble. For more details, see [18].
- One of our own ideas for making use of the abilities of consensus clustering and the NMI criterion defined in (3) is the search for good parameters in clustering algorithms. This is a serious problem in practice, since because of the lack of labels, there is no objective criterion measuring the quality of a clustering. What we propose is, to choose the set of parameters, for which its clustering yields the the highest NMI value with the consensus. For an experimental example, see 3.3.

2.5.1 Consensus Evaluation Criterion

We aim to define an objective criterion to measure how good the consensus clustering $\mathcal{C}^*$, found from comparing all clusterings, is. This will give us an idea of what to maximize and we can evaluate the quality of a consensus clustering afterwards.

This is done with the notion of mutual information $I(X, Y)$ of two random variables from information theory as well as their entropies $H(X)$, $H(Y)$. In an analogy to the correlation coefficient the normalized mutual information (NMI) is defined by

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}} \tag{3}$$

for two random variables X and Y, which is always a number between zero and one. Instead of just counting, how many common cluster labels two clusterings have, this definition accounts for the probability of the clusters to appear. Also, a permutation of the indices of cluster labels in one clustering, for instance, does not change the NMI.

There is a straightforward estimation $\phi^{\mathrm{NMI}}(\mathcal{C}_i, \mathcal{C}_j)$ for any two clusterings $\mathcal{C}_i$, $\mathcal{C}_j$. To then measure the quality of a consensus clustering $\mathcal{C}^*$, we just compute the mean value of the NMI's of the consensus clustering with all given clusterings, i.e. estimate the average normalized mutual information (ANMI) by

$$\phi^{\mathrm{ANMI}}(\mathcal{C}^*, \Lambda) = \frac{1}{m} \sum_{i=1}^{m} \phi^{\mathrm{NMI}}(\mathcal{C}^*, \mathcal{C}_i). \tag{4}$$

For more details on the theory, see e.g. [18].

Unfortunately, direct maximization of the ANMI as an objective function by an exhaustive search is not tractable since the number of possible consensus labellings grows exponentially in the number of data points. Also, greedy algorithms are computationally not feasible and lead to local maxima [18]. Therefore, we need to find a more efficient algorithm than the greedy approach.

2.5.2 Algorithms

Two approaches were implemented in our software package:

Re-clustering points: The straightforward approach. One constructs a distance $d_p : \{x_1, \ldots, x_n\} \times \{x_1, \ldots, x_n\} \to \mathbb{R}_0^+$ on the data points, by counting for each pair (x_i, x_j) of points, in how many labels they differ among all clusterings, i.e. d_p is the Hamming distance. Then one re-clusters the points with a similarity based clustering algorithm. In our case, one can use spectral and hierarchical clustering.

Re-clustering clusters: Let the set of all clusters of all clusterings be denoted by $\mathcal{A} := \{C_1^{(1)}, \ldots, C_{k_{(1)}}^{(1)}, \ldots, C_1^{(m)}, \ldots, C_{k_{(m)}}^{(m)}\}$. Then one introduces a distance function $d_c : \mathcal{A} \times \mathcal{A} \to \mathbb{R}_0^+$ on the clusters, by comparing the size of their intersection with the size of their union. That is, one takes their Jaccard distance, which is a number between zero and one. One now again re-clusters the clusters by a similarity based algorithm and receives meta-clusters $\mathcal{C}_1^M, \ldots, \mathcal{C}_{k_M}^M$, so that each cluster from any clustering belongs to some meta-cluster. Finally, the meta-clusters compete for points, i.e. every point – which is in m (original) clusters – is assigned the label of that meta-cluster, it is in most often. Ties are broken randomly.

3 Results

3.1 Comparison of different algorithms

Using the clustering algorithms we implemented on different data sets, three main issues appeared: (i) the choice of parameters has a big effect on the outcome of the clustering and it is not always possible to set the parameters without "knowing" the data, (ii) some algorithms don't scale well with the number of data points n, dimension d and total number of iterations *iter*, and (iii) depending on the shape of the clusters and the existence of outliers and noise, some algorithms work better than others. See the table 1 for a complete comparison of the different clustering methods and their advantages and disadvantages.
The outcome of the clustering depends heavily on the chosen method. Therefore it's essential to understand the limitations and strengths of the different methods in order to choose the "right" method for a clustering task. Direct validation of the clustering result is not possible as the data set – by definition of the clustering task – is usually unlabeled and there is no correct result. A wide range of clustering quality measures exists, but the choice of a quality measure is also very problem-dependent. Therefore, we introduce a general objective function, that helps with parameter search in 3.3.

We also looked at how the running time of our implemented algorithms grows for increasing values of n, see figure 5. The time complexity of the algorithms is inferred by assuming a power-law dependence and fitting a straight line via least squares minimization on a log-log plot. It can be seen that the most expensive algorithm is spectral clustering, whereas the cheapest algorithm is Regular space clustering. A general trend seems to be that distance-based methods are around $\mathcal{O}(n)$, density-based methods are around $\mathcal{O}(n^2)$ and Spectral Clustering is around $\mathcal{O}(n^3)$. Those results agree well with the upper bounds from the literature, compare table 1. We didn't look at the dependence of the methods on dimension d and number of iterations *iter*.

3.2 Consensus

Consensus clustering is definitely not a very cheap way of clustering data sets. But it might very well help with the issues (i) and (iii) mentioned above by aggregating different clusterings and finding a consensus between them.

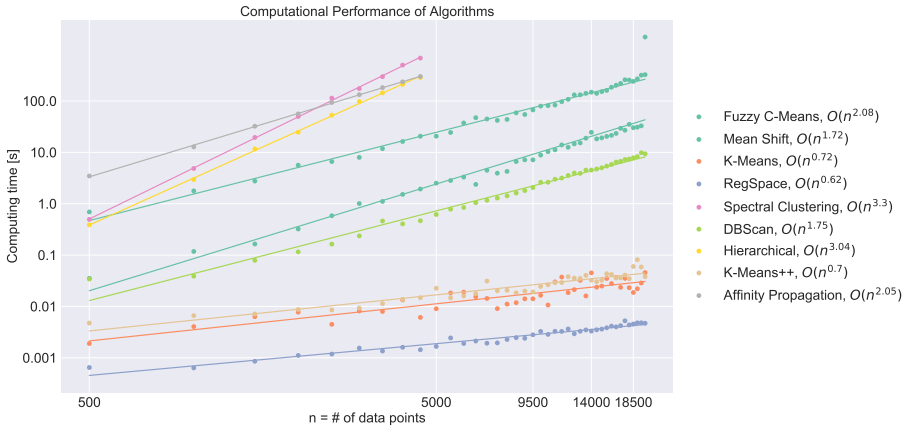

Figure 5: Computing time of algorithms vs number of data points on a log-log plot

3.2.1 Robustness

One of the main strengths of Consensus clustering is that it can be used to easily produce robust clusterings with a variety of different datasets without knowing too much about the parameters. We demonstrate this with K-Means, DBSCAN and two datasets, where exactly one of the algorithm fails. On the one hand, we chose a dataset with three unevenly sized blobs with different variance, where K-Means performs better, and a set with three spirals winded into each other, where K-Means fails but DBSCAN recognizes the three spirals. For both datasets, we create five K-Means clusterings of two to six clusters, and DBSCAN clusterings with $eps = 0.5, 0.75, 1.0, 1, 25, 1.5$ and $minPts = 5, \ldots, 9$. From these clusterings, we create a consensus clustering, re-clustering points with spectral clustering.

The results clearly show the weaknesses and strengths of the two algorithms, but the consensus object is always able to find the correct clustering from the 10 clusterings. In the case of the blobs dataset, which is shown in figure 7, the consensus clusterer is not confused by the DBSCAN clusterings, since they are often just one uniform cluster. For the spiral case shown in figure 6, he is not confused by the K-Means clusterings in the spirals dataset, since it is much harder to find a good consensus for the K-Means objects, opposed to one for the DBSCAN clustering.

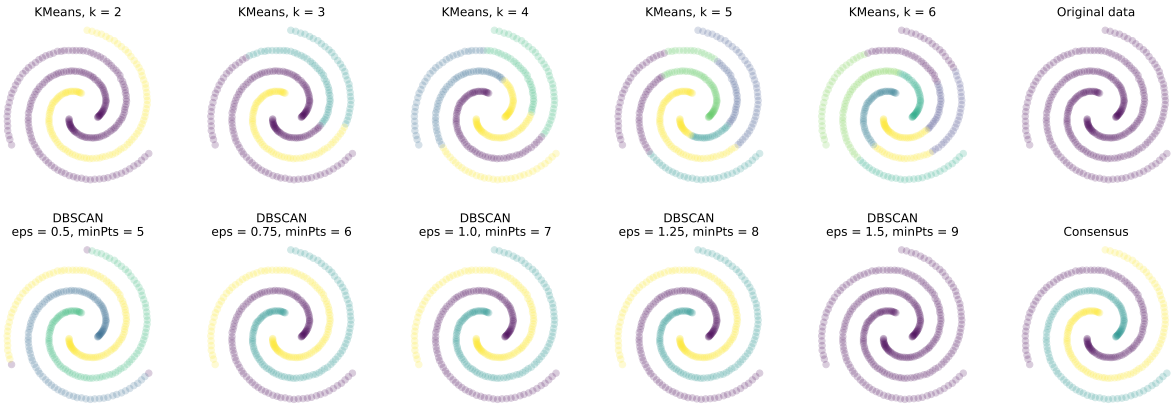

Figure 6: A dataset, where DBSCAN works well and K-Means fails to recognize the three different spirals due to its distance based nature. The consensus clustering is able to find the three correct clusters.

3.3 Parameter Search

In the motivation of consensus clustering (see 2.5), we described a new idea of how to use consensus clustering for the search of good parameters of a clustering algorithm. In our experiment, we considered DBSCAN with a range of 10 different eps-parameters and, for sake of simplicity, a fixed minPts-pamareter of 10. DBSCAN is obviously very sensitive to the eps-parameter. Building a consensus object of all 10

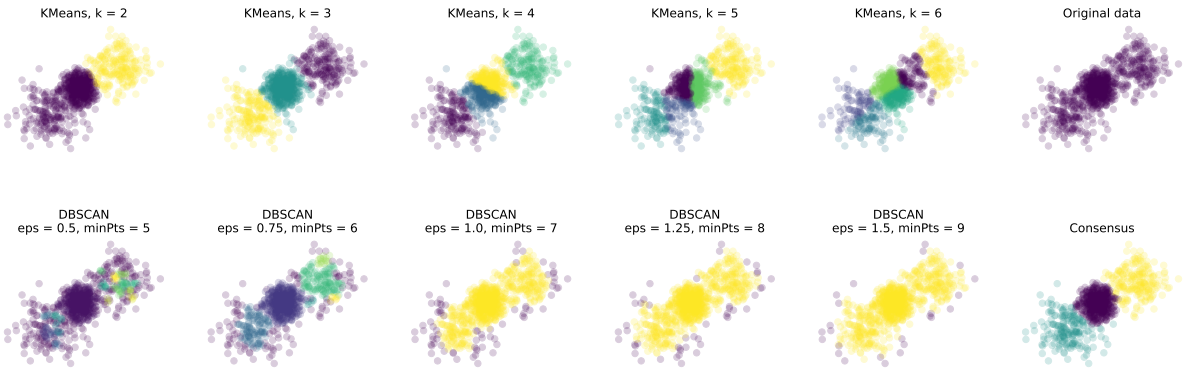

Figure 7: A dataset, where K-Means works well and DBSCAN fails most of the time due to the sensitivity of choice of parameters. The consensus clustering is able to find the three correct clusters.

DBSCAN clusterings and picking the NMI-maximizer yields a very reasonable choice of eps, as it can be seen in figure 8.

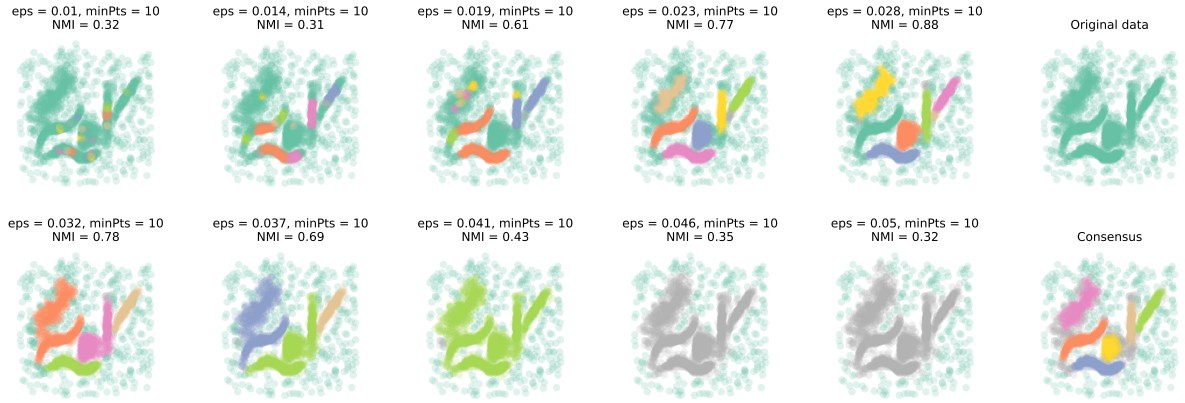

Figure 8: Finding the best DBSCAN eps parameter by taking the one, which yields the highest NMI with the consensus clustering.

4 Conclusion

In conclusion, we have to say that there is no general clustering method working for all data sets. Every method has its up- and downsides. So choosing a method, one has to balance them. Some knowledge about the given data set is definitely needed to decide which clustering approach to use and to choose the parameters wisely. Also pre-processing the data set is a good idea, something simple like scaling the data to have mean zero and standard deviation one is often enough.
Our aim was to implement a wide range of clustering methods and compare them. Table 1 can be used to look up different methods and decide which one is best for a given clustering problem. Consensus clustering addresses many short-comings of existing clustering techniques by finding the consensus across several clustering results such that defects of different clustering methods will be averaged out. It can produce very good and robust clustering results, and it less sensitive to outliers. But it also has its problems, since consensus clustering is expensive and it is still dependent on choosing the number of clusters parameter and the choice of the algorithm. In section 3.3 we introduced a novel idea of how to search for parameters for clustering algorithms using consensus clustering and the NMI criterion.
Our Python package can be used for clustering data with classical as well as state-of-the-art clustering techniques. As machine learning is a very quickly evolving and hot research area, we expect that much will happen in the next years with existing methods being extended and hopefully more focus on consensus clustering.

Method	Parameters	Comput. Complexity	Advantages	Limitations	Possible extensions
Affinity Propagation	Similarity matrix	$\mathcal{O}(n^2\ iter)$	Works on abstract similarity relations, exemplar weighting and sensitivity parameter tuning possible	Needs optimal configuration of iteration damping and sensitivity parameters	Hierarchical Affinity Propagation
DBScan	Density parameters ϵ, m_{points}	$\mathcal{O}(n^2)$	Noise/outlier detection, detection of highly nonlinear shapes	Highly sensitive to density parameters	HDBSCAN, input parameter estimators
Fuzzy C-Means	Number of clusters k and fuzzier m	$\mathcal{O}(nk^2d\ iter)$	Noise/outlier detection	k needs to be known/approximated in advance, strong dependence on chosen distance metric, not deterministic, might only converge to local minimum	estimation of number of clusters, Kernel C-Means, C-Means++
Hierarchical clustering	Number of clusters k and/or maximum linkage distance l	$\mathcal{O}(n^2 \log(n))$ in general, $\mathcal{O}(n^2)$ for single-link and complete-link	Can detect highly nonlinear structures, only works on a distance measure between the points (observations themselves not needed)	The number of clusters or a stopping distance threshold, as well as the linkage function must be specified, depends heavily on the given distance function	Variety of different distance or linkage functions
K-Means K-Means++	Number of clusters k	$\mathcal{O}(nkd\ iter)$ (Lloyd iteration) worst case $iter = 2^{\Omega(\sqrt{2})}$ [1]	Computational speed, many results on convergence, complexity and objective function available	k needs to be known/approximated in advance, not deterministic, might only converge to local minimum	Kernel K-Means, K-Medioids
Mean Shift	Bandwidth parameter for kernel density estimation (kde)	$\mathcal{O}(n^2\ iter)$, kde doesn't scale well with d	no knowledge about shapes or number of clusters assumed, used a lot for image segmentation	too expensive for most applications	bandwidth estimators, code could be parallelized
Regular Space	Minimal distance ϵ_{min}	$\mathcal{O}(nk)$, worst case $\mathcal{O}(n^2)$ k is parameter dependent	Computational speed, useful for large-scale statespace discretizations	Restricted interpretability	Kernel extension
Spectral Clustering	Number of clusters k	$\mathcal{O}(n^3)$	Can detect highly nonlinear structures, works on abstract similarity graph	k has to be know/approximated, few results on different graph/adjacency types, strongly dependent on choice of graph, very slow	Landmark and/or sparse methods, more efficent eigensolvers

Table 1: Comparison of different clustering methods we implemented

References

[1] Arthur, David and Sergei Vassilvitskii. “How Slow is the K-means Method?". Proceedings of the Twenty-second Annual Symposium on Computational Geometry. SCG ’06. New York, NY, USA: ACM: 144–153 (2006).

[2] Arthur, D. and Vassilvitskii, S. “k-means++: the advantages of careful seeding.” Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035 (2007).

[3] Bezdek, James C., Robert Ehrlich, and William Full. “FCM: The fuzzy c-means clustering algorithm.” Computers & Geosciences 10.2-3 (1984): 191-203

[4] Comaniciu, Dorin, and Peter Meer. “Mean shift: A robust approach toward feature space analysis.” IEEE Transactions on pattern analysis and machine intelligence 24.5 (2002): 603-619.

[5] Dhillon, Inderjit S. and Guan, Yuqiang and Kulis, Brian. "Kernel K-means: Spectral Clustering and Normalized Cuts." Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2004): 551–556

[6] Ester, Martin and Hans-Peter Kriegel and Jörg Sander and Xiaowei Xu. “A density-based algorithm for discovering clusters in large spatial databases with noise.” Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (1996): 226–231

[7] Fränti, Pasi. “Clustering datasets.” Clustering datasets. N.p., 2015. Web. 07 July 2017. http://cs.uef.fi/sipu/datasets/.

[8] Frey, Brendan J. and Delbert Dueck. “Clustering by Passing Messages Between Data Points." Science Vol. 315, Issue 5814 (2007): 972-976

[9] Helfmann, Luzie, Johannes von Lindheim and Mattes Mollenhauer. “Clustertools Python package.” https://github.com/clustertoolsgroup/clustertools_project.

[10] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. “Hierarchical clustering.” The elements of statistical learning 2 (2009).

[11] Jordan, Michael I., Andrew Y. Ng and Yair Weiss: “On spectral clustering: Analysis and an algorithm.” In: Advances in Neural Information Processing Systems. 2, (2002), S. 849-856.

[12] Lloyd, S. P. “Least square quantization in PCM". Bell Telephone Laboratories Paper. Published later: Lloyd., S. P. (1982). “Least squares quantization in PCM.” IEEE Transactions on Information Theory. 28 (1957): 129–137.

[13] MacQueen, J. B. “Some Methods for classification and Analysis of Multivariate Observations.” Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press (1967): 281–297.

[14] Malik, Jitendra and Jianbo Shi. “Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence.” 22(8), (2000), S. 888-905.

[15] Ostrovsky, R., Rabani, Y., Schulman, L. J. and Swamy, C. “The Effectiveness of Lloyd-Type Methods for the k-Means Problem". Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06). IEEE. (2006) pp. 165–174.

[16] Perez-Hernandez, Guillermo, Fabian Paul, Toni Giorgino, Gianni de Fabritiis and Frank Noé. "Identification of slow molecular order parameters for Markov model construction” Journal of Chemical Physics, 139(1) (2013): 015102.

[17] Prinz, J.-H., H. Wu, M. Sarich, B. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte and F. Noé: “Markov models of molecular kinetics: Generation and Validation.” J. Chem. Phys. 134, 174105 (2011).

[18] Strehl, Alexander, and Joydeep Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions.” Journal of machine learning research 3.Dec (2002): 583-617.

[19] Turlach, Berwin A. “Bandwidth selection in kernel density estimation: A review.” Louvain-la-Neuve: Université catholique de Louvain, 1993.

[20] Weisstein, Eric W. “Voronoi Diagram.” MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/VoronoiDiagram.html