Supplementary Material for: On the View of Clustering as Manifold Learning

Daniyal Kazempour, Florentin Schwarzer, Peer Kröger and Thomas Seidl

In this document the reader may find additional experiments which are beyond the scope of the paper itself but where the authors deemed them to be interesting to investigate, since they address different targets of the described method such as parameter change over iterations, observed behaviors in using different distance functions and clustering results on a small real-world dataset.

1 On iteration-wise varying neighborhood size

One property of the original Mean Shift algorithm is that one obtains a static bandwidth for all objects, over all iterations. Since all data objects are not necessarily in an equally dense region, objects may shift "faster" over the iterations. For this we asked at this point of what would be the potential effects if we introduce variable search masks based on the local structure. We decided to determine an individual search mask for each data point *in each iteration* based on their neighborhood. There are several ways to define the neighborhood of a data point. In this work we used k-nearest neighbors (k-NN). Based on k data points we defined the search mask by computing PCA. It is vital to note that the choice of the parameter k has a large impact on the resulting search mask. There is one critical decision we need to make: How do we change the k over the iterations of the algorithm? We decided to investigate three different changes of k over the iterations.

1. Constant neighborhood: The size of the inspected neighborhood does not change over iterations.

2. Linear growth with threshold: The growth rate is constant per iteration. The maximum$k$size is defined by a threshold.

3. Logistic growth: The growth of the neighborhood is characterized by a logistic function and changes over iterations.

A logistic function is defined by

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{1}$$

, where L is the maximum size of the neighborhood, x_o the midpoint of the sigmoidal function and k describes the growth rate.

In the upcoming subsections the hyperparameter k growth schemes over the iterations and their impacts are analysed in more detail.

1.1 Constant Parameter

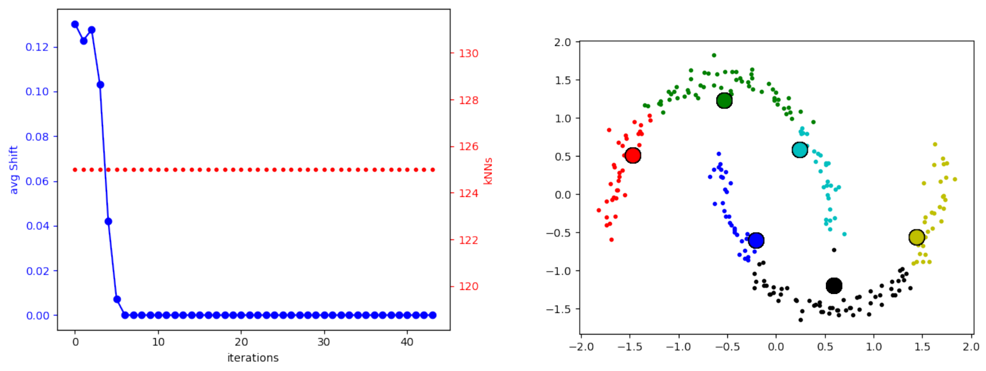

Figure 1: left: the constant size of the neighbourhood per iteration in red and the average shift per data point per iteration in blue. right: the result of the first part of the algorithm using the constant neighbourhood approach.

In this variant of modelling the parameter k, the size of the neighbourhood does not change over the iteration and stays the same (constant). On the left image in figure 1 two different plots are presented. The horizontal axis displays the iteration. Each data point is visited and shifted once per

iteration. The red line (right vertical axis) shows the size of the k per iteration. In this case the size is constant and stays the same. In this model the user can only select the constant once. Since we used the two moons data set with 250 data points and a noise level of 0.05, we selected $k = 125$.
The blue line (left vertical axis) shows the average shift per iteration. For the sake of clarity the points are connected in the figure. Actually, the shift is only defined per one iteration.
It can be seen that there is a high average shift during the first iterations but it flattens quickly. This makes sense since the size of the neighbourhood stays the same. The resulting neighbourhood per data point in each iteration does not change after a few iterations. Due to that. the data points only shift a little until they converge to their respective modes. Logically, the choice of the size k is critical and very significant for this model.
On the right image of figure 1 the resulting modes with the corresponding data points are shown (*mode sets*). A Gaussian kernel was used. As a great result, each half moon is divided into three *mode sets* of almost the same size. Only one data point of the upper half moon converges to a mode of the lower half moon (black data point next to the turquoise data points). It is striking that the modes are not in the middle of the corresponding data points but slightly moved to the center point of the ellipse. This can be justified by the fact that the size of $k = 125$ (half of all data points) may be too large for this data set.
Altogether, good results can be obtained with this model but the user has only one constant and very critical parameter. In most of the cases the algorithm only needs a few iterations since the average shift is flattened really fast.

1.2 Linear Growth

The next model is the linear growth approach. The results are shown on figure 2. On the left side the linear growing neighbourhood in red colour is indicated. As shown, the size of the neighbourhood equals 0 in the beginning and increases by 5 (a 50th of the entire data set) per iteration.
The average shift per iteration is shown in blue colour on the same figure. It can be seen that this average shift has several peaks in which the shift is specifically high. This goes hand in hand with phases in which a low shift can be observed. This may be explained by the fact that the neighbourhood is not growing fast enough. This phenomenon is similar to the observations

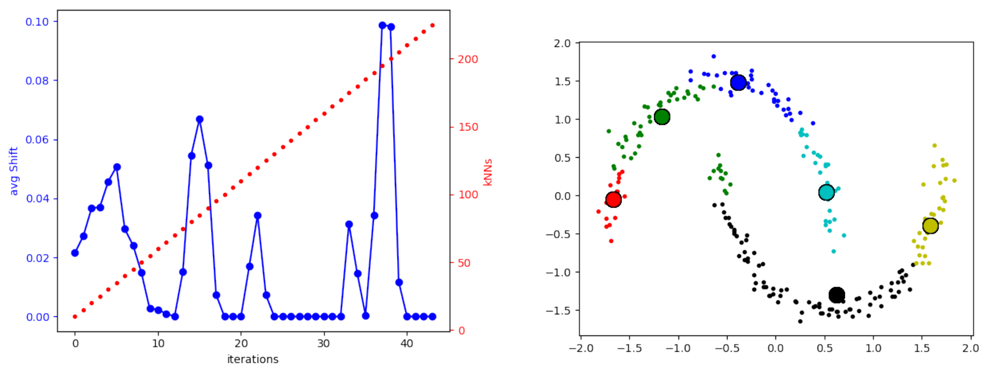

Figure 2: left: linear growing size of the neighbourhood per iteration in red and the average shift per data point per iteration in blue. right: the result of the first part of the algorithm using the linear growing neighbourhood approach.

in the constant case but appears several times. After reaching a peak, the average shift decreases in the next iterations, since the neighbourhood does not change fast enough. However, at some points the neighbourhood has grown enough to have a new high peak.
The heights of the peaks also increase with later iterations. This occurs because most points in a later iteration are already converged to some small modes and these modes then merge together resulting in a high average shift.

The final modes with their corresponding data points are shown on the right image on figure 2. The result is not the one that was expected since the upper half moon is divided into 4 *mode sets* while the lower one has only two *mode sets* with an unequal number of data points. The data points of the left tip of the lower half moon even merged to a mode of the upper half moon. In order to counteract these effects, the gradient of the neighbourhood size (red line) can be changed. The steeper the size growth, the faster the local neighbourhoods disappear to global properties. Another possibility to change the linear growing neighbourhood besides the gradient is to insert an upper bound. In our implementation we usually used the number of all data points divided by two as an upper bound because it has empirically proven to be a proper limit.

1.3 Logistic Growth

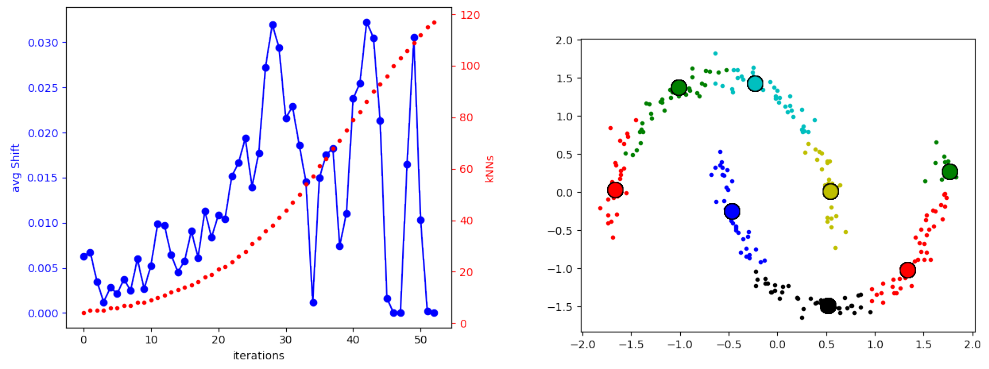

Figure 3: left: the logistic growing size of the neighbourhood per iterations in red and the average shift per data point per iteration in blue. right: the result of the first part of the algorithm using the logistic growing neighbourhood approach.

The final model is the logistic growth. As shown in figure 3 on the left, the logistic function is not a basic function like the constant or linear variant since there are several possibilities to have an impact on the model. As explained in equation 1, there are several variables. Empirically we found out that a maximal size of $L = \frac{entire\_dataset}{2}$ is a appropriate choice. That means that the neighbourhood will grow maximum on L. In most cases, we set the midpoint to $x_0 = 40$. In figure 3 on the left, the red curve changes from a left turn to a right turn. When observing the average shift in the same figure (blue chart), a nearly constant growing shift will be recognised till iteration 30. This is exactly what was desired to be achieved. In the beginning only a small shift occurs which grows per iteration. From the 30th iteration on, a few peaks are visible which is due to the fact that modes have already formed and they merge together so that a large shift can be seen for a short number of iterations. All in all this approach seems to suit this data set well.
The resulting modes with the corresponding data points are shown in figure 3 on the right image. Each half moon is divided into four approximately equal sized *mode sets*. It can also be stated that the modes are positioned approximately in the middle of the *mode sets*. No data points of the lower half moon belong to a mode of the upper half moon and vice versa.
Similar to the moon data set, the logistically growing neighbourhood proved

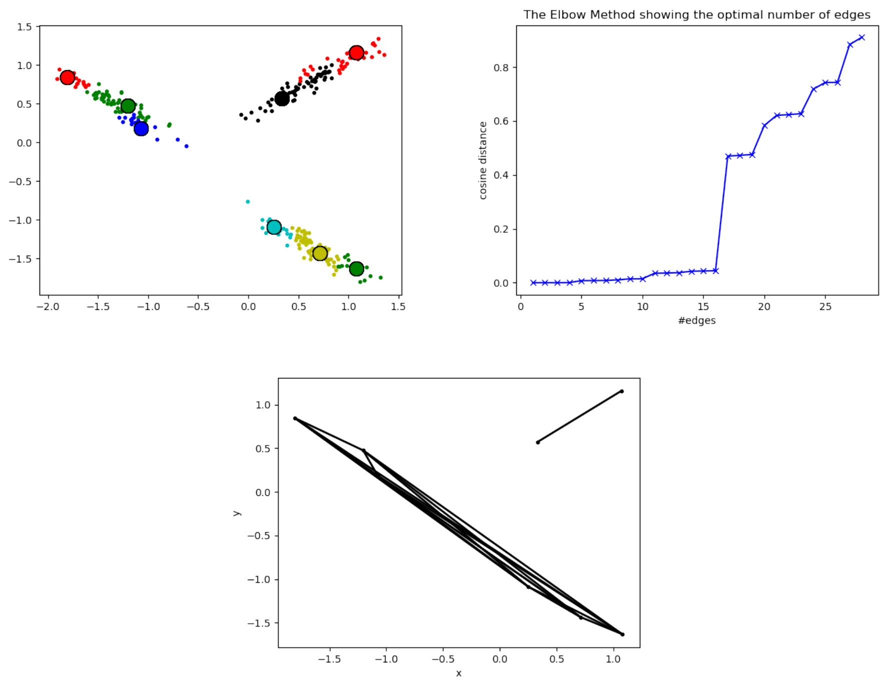

Figure 4: top-left side: three lines data set with their modes and cluster members, top-right side: plot of the number of edges between the modes against the cosine-distance distance, bottom: resulting higher-order topology.

to be the best variant for most data sets. This assumption has been empirically established.

2 On different distance measures

Since we used in the experiments the single-link distance for determining the cut-off value a question that arises is: what happens if one chooses different distance measures? For that purpose we conducted an experiment on a correlation-oriented measure.

The distance measure that we used is based on the orientation of given mode-sets. If two modes have the same orientation they belong to the same cluster. The main orientation is given by the first principal component of the set. We used the cosine similarity to compare the eigenvectors pair-wise.

The formula is given by:

$$sim_{cos}(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$$

, where v_1 and v_2 are the given orientations. The cosine similarity between two vector is one if the angle between the two vectors is zero. The larger the angle, the smaller the similarity. The similarity range goes from -1 (opposite direction, because angle 180 degrees) to 1 (angle between vectors is 0).
Since we compare the orientation of two sets of data, the similarity of two 180 degrees different vectors should be 1. The resulting similarity scale comprises values between 0 and 1. And since we use distances and no similarities, we still have to twist the whole scale. The resulting cosine distance between two vectors is:

$$dist_{cos}(v_1, v_2) = 1 - |sim_{cos}(v_1, v_2)|$$

An experiment on a toy data set is shown in Fig. 4. There are three separated sets of objects where two of them have similar orientation. The group in the upper right corner has a different orientation. The first figure (top-left) shows the result after computing Mean Shift with PCA. The three groups are divided in several mode-sets. The second figure (top-right) shows the pair-wise cosine-distance between the main orientation of these mode-sets in ascending order. The elbow is clearly visible between edge 16 and 17. By permitting these 16 edges between the modes, we receive two clusters as shown in the last figure (bottom). The two groups that had the same orientation were merged together. This result was to be expected, since the two groups have the same orientation. Which of these two measurement methods is the better choice depends on the problem at hand. Both have different approaches. If clusters are defined based on their proximity, the single-link distance would be the better choice. But if clusters are defined through their linear orientation, in the sense of orientation and translation, the correlation-based method is more suitable.

3 Runtime Complexity

In this section we elaborate on the runtime complexity of our method. The core of the algorithm is still a Mean Shift which leads to a runtime complexity of $\mathcal{O}(i \cdot N^2)$ where i denotes the number of iterations and N the number of data objects. Computing the kNNbrings up a complexity of $\mathcal{O}(N \cdot d \cdot k)$

Algorithm	**ARI**
Standard Mean Shift	~ 0
Our approach	0.679
DBSCAN	0.426
SLINK	0.359
Spectral	0.368

Table 1: Comparison of ASSEMBLE against Mean Shift, DBSCAN, SLINK and spectral clustering on the Wine data set.

where d refers to the dimensionality of the data and k stands for the$k$NNof an object. We do akNNquery for each object in the data set. As a consequence we are at a runtime of $\mathcal{O}(i \cdot N^2 + N \cdot (N \cdot d \cdot k))$. As our method performs per object on its$k$NNa PCA at each iteration, we get a runtime complexity of $\mathcal{O}(i \cdot N^2 + N \cdot (N \cdot d \cdot k) + N \cdot k^2)$. The runtime of a PCA on k nearest neighbors is, using the power iteration method, in $\mathcal{O}(k^2)$. Having obtained m number of modes, an agglomerative clustering is performed using SLINK[**?**], which comes with a complexity of $\mathcal{O}(m^2)$. Therefore the total complexity is: $\mathcal{O}((i \cdot N^2 + N \cdot (N \cdot d \cdot k) + N \cdot k^2) + m^2)$. Since it holds that $k << N$, the runtime remains quadratic.

4 Cluster Performance Experiments

By introducing the modifications to Mean Shift in order to learn and capture the manifolds in a given data set, the question is raised if it still performs well in its original task, namely to detect clusters. For that purpose we chose the Wine data set. It consists of 178 data points with each 13 features. Each data point describes the chemical analysis of a wine. All 178 wines are grown in the same region in Italy, but are divided into three different cultivars. The task is to cluster the data points into the three cultivars. The results are shown in figure 1.

The standard Mean Shift algorithm detects only two cluster where one cluster consists of 174 data points and the other one consists of 4 data points. Since most of the data points are in the same cluster, the ARI score is very low (see table 1). In contrast to these results, our new algorithm receives better results for the ARI score. We used a logistic growing neighborhood. So far we selected the standard deviation along each principal component as bandwidth size. But with growing dimension size, the bandwidth search mask gets smaller relative to the entire space. We decided to increase the bandwidths by a multiple of the standard deviation in order to counteract this effect. For the 13 dimensional wine data set, four times the standard deviation turned out to be a good choice. The ARI score we received with these parameters was 0.679 (see table 1). The algorithm detected three large clusters (51, 52 and 51 data points) and 21 very small clusters (each 1-2 data points). In this case we used the single link distance to connect modes pair wise to clusters. We further compared our method against a density based algorithm DBSCAN, a hierarchical namely SLINK and against spectral clustering. For DBSCAN we made a grid scan, testing all ε-ranges from 0.1 to 100 in 0.05 steps and all minpts from 1 to 100 in steps of 1. The hyperparameter setting which yielded the highest ARI is: ε: 42.15 and minpts: 15. For SLINK and Spectral we scanned for all number of partitions $k \in [2, 10]$ and obtained $k = 3$ as the best ARI yielding setting. As we can see in Table 1, ASSEMBLE outperforms all other clustering methods. All in all ASSEMBLE yields much better results than the standard Mean Shift algorithm or other clustering methods, but there is still potential for improvement. Since the data set is high dimensional and the classes blend into each other, it is hard to detect the structure of the clusters.