MAI-URL Course Work 1: Option B

# Implementation of the Sheared Nearest Neighbor algorithm and its comparison with different clustering algorithms.

Jana Reventós Presmanes - jana.reventos@est.fib.upc.edu

June 1st of 2020

# Contents

# List of Figures

## List of Tables

## List of Algorithms

# 1 Introduction

Clustering algorithms is a machine learning technique that divides data into groups (clusters) for the purpose of classify each data point in a specific group. In theory, data points that belong to the same cluster have similar properties/features according to a similarity metric. Clustering is a unsupervised technique that is commonly used in data analysis to gain some valuable insights from our data by seeing what groups the data points fall into.

Large number of clustering techniques have been developed in the last decades but significant challenges still remain when dealing with high dimensional data. In this course work it has been analyzed and implemented the Sheared Nearest Neighbor (SNN) clustering algorithm developed by L. Ertöz et al. in 2003 [1] which deals with some of the clustering challenges such as how to find clusters with different sizes, shapes and densities, how to handle with noise and outliers, and how to determine the number of clusters. A comparison between SNN and other clustering algorithms (K-means,DBSCAN,etc...) is done in order to conclude whether or not the SNN method outperforms on the clustering process.

## 1.1 Clustering algorithms road map

K-means is one of the most widely known clustering algorithm, proposed by S. Lloyd in 1957 [6]. It is commonly used in market segmentation and computer vision among many other applications. The K-means algorithm aims to partition n examples into k clusters in which each example belongs to the cluster with the nearest mean (cluster centroid). In practice, K-means have several problems such being sensitive to initialization, clusters of different sizes, densities and outliers. The spatial complexity of K-means make its no suitable for high dimensional datasets.

Spectral clustering deals with connectivity and not compactness as K-means does. With this idea it is able to find non-globular clusters. In spectral clustering, data points are treated as nodes of a graph. Thus, clustering is treated as a graph partitioning problem. Data points are mapped to a low-dimensional space. To achieve this, first, a similarity matrix of the data is computed and from it a Laplace matrix is acquired. Second, the K eigenvalues and convectors of the Laplace matrix are computed to then use the eigenvectors as new data points. Finally, K-means is used as clustering algorithm. The limitation of the Spectral clustering algorithm is that is computationally expensive for large data sets, it has a time complexity of $O(n^3)$.

The single link agglomerative hierarchical algorithm is another technique that performs well in capturing clusters with non-globular shapes but the main drawbacks is that is very sensitive to noise and clusters with varying density. On the other hand, complete or average agglormerative algorithms can handle noise but then are biased when finding globular clusters. [1] Furthermore, in the best case the computational cost is $O(n^2)$ but usually is $O(n^3)$ which is too high. One example of agglomerative hierarchical shame is CURE, created by S.Guha et al. (1998) [3]. CURE uses the concept representative points to find non-globular clusters and wide variances in size. The algorithm defines clusters with multiple representative points that capture the its shape. It selects well scattered points from the cluster and then it shrinks them towards the center of the cluster by a specified fraction which helps to dampen the effects of outliers (points located at the cluster boundaries. However, L. Ertöz et al. showed that although the methodology that uses allows CURE to find clusters of varying sizes and shapes, in some data sets, it is still biased towards finding globular clusters as CURE ignores the information of inter-connectivity of objects in two clusters because it still has the notion of cluster center.

To challenge some of the described limitations other approaches have been developed such as DBSCAN and OPTICS which are density based algorithms. DBSCAN and OPTICS were described by Ester et al. (1996) [2] and Ankerst et all. (2000) respectively. These algorithms are based on finding areas of high density to find arbitrary shapes. In the case of DBSCAN, density is associated with the number of points in a region of specific radius, Eps, around a specific point. Then, it defines core points as points that have a density above a specific threshold, MinPts, while noisy points are those that do not belong to any core point within the specific radius. Clusters are build around core points and noise points are discarded, if two core points are within a radius Eps it's clusters are joined together. The boarder-points, which are those that can have more than one core point within their radius, are associated to any core point. These border points flesh out the skeleton of the clusters created by the core points. [1]. The difference between DBSCAN and OPTICS is that OPTICS uses an heuristic to find good values for the Eps and MinPts parameters. Even though, DBSCAN and OPTICS can find clusters of different shapes, the main problem is that they have difficult to handle clusters of different densities, as they cannot find core points of varying densities clusters.

All these algorithms (K-means, Spectral clustering, CURE, DBSCAN, OPTICS...) have been designed to deal with low dimensional data but high dimensional data bring new challenge to face and one of the most important is the similarity function. Similarity defines how examples are related to each other but in high dimensions typical metrics such as cosine or Jaccard coefficient cannot be trusted when the similarity between two points is low. Typically, high dimensional data is sparse, thus, the similarity between two points on the average is low. The similarity based in shared nearest neighbor approach defined by Javis and Patric (1973) [4] can solve this problem. They described a new similarity based on the number of neighbors that two points share which is very valuable for finding clusters of different densities.

With all these clustering and similarity discoveries, L.Ertöz et al. developed a new technique called SNN clustering algorithm that uses the main ideas of the SNN similarity and DBSCAN algorithm. In this practical work the SNN clustering algorithm is compared with the K-means, Spectral clustering, CURE, OPTICS and DBSCAN to demonstrate its advantages and limitations when it comes to find clusters of different shapes, sizes and densities in high dimensional data.

## 2 SNN clustering algorithm

The SNN clustering algorithm [1] is an extension of DBSCAN [2] that is based on finding the nearest neighbors (NN) of each data point and it redefines the similarity between two points in terms of how many nearest neighbors the two points share [4]. Using this approach, the SNN algorithm it searches the core points and builds clusters around them. Thanks to the use of this new similarity it deals those clusters with different density, which is a problem in high dimensional data sets. A single data point may be significant because could be representative of a large number of data points that are located far away one from the other, besides, deals with the shape and size problem. One of the main good aspects is that the number of clusters is not defined by the user, it is automatically acquired. Furthermore, the algorithm does not cluster all the data points, as it has the ability to detect noise. The run time complexity of the SNN clustering algorithm is $O(n^2)$ where n is the number of points, when the similarity matrix has to be computed.[1]

Before explaining the theoretically implementation of the SNN clustering algorithm first we are going to describe the alternative definitions of similarity and density that L. Ertöz et al. [1] used to develop the algorithm:

- **Similarity:** the similarity between two points p and q is described as the size of the intersection between the NN of point p and the NN of point q. Which is defined as follows:

$similarity(p, q) = size(NN(p) \cap (NN(q))$

- **Density:** is defined as the number of links within a given radius, specified in terms of SNN similarity, that a point has. Here, density uses SNN similarity with the K-nearest neighbor approach to estimate density. For instance, if the $K^{th}$ nearest neighbor of a point is close (it has high similarity), the algorithm says that there's high density at this point.

To understand better this definitions we will explain the SNN clustering algorithm concepts using a similarity graph. From a similarity matrix, a SNN graph can be constructed. A link is created between p and q if and only if p and q have each other in the NN list. Figure 1 left shows the K-nearest neighbor of each data point, while Figure 1 right, displays the SNN graph, where those points that wasn't in the nearest neighbor lists of its own nearest neighbors ended up loosing all its links.

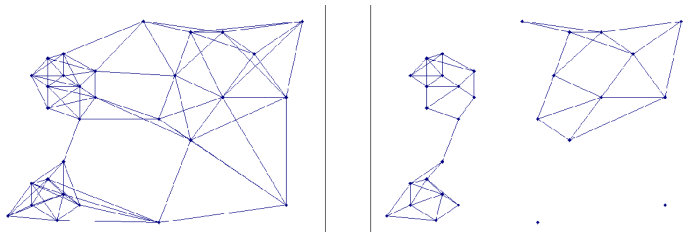


Figure 1: Left: nearest neighbors graph. Right: sheared nearest neighbors graph.

Then, clusters can be obtained by removing links that have a size less than a user specified threshold, or in other words, that have less weight/similarity. This threshold in terms of DBSCAN is defined as Eps. From the clusters, the points that have high density, i.e. high connectivity in the SNN graph, are candidates to be defined as core points, which tend to be located inside clusters. On the other hand, those point that have low connectivity (low density) are candidates to the noise points and outliers. Using again, DBSCAN terminology, MinPts, will we minimum number of links (connections) that a point must have in order to be a core point candidate.

The SNN clustering algorithm paper it gives a list of steps to follow to implement the approach [1]. The algorithm itself is an extension of Javis-Patrick and DBSCAN, where we have to take into account that the last 4 steps correspond to DBSCAN. The NN list has size K, and will determine the granularity of the clusters. If K is too small, the algorithm tens to find small clusters whereas if K is too large the algorithm tends to build large clusters. So, the K parameter is the most important to tune because will fix the number of clusters. Therefore, the MinPts should be a fraction of the K value. The steps of the SNN algorithm are described below:

1. Compute the similarity matrix by obtaining similarity graph, where data points are nodes connected by edges, where the edges weight belongs to the spatial similarity which is the opposite to the spatial distance.

2. The user should define the K parameter in order to sparsify the similarity matrix because for each point we only want to keep the edges of the K most similar neighbors.

3. Construct the shared nearest neighbor (SNN) graph using the sparsified matrix. The method consists in substituting the edges of the graph by the similarity weight between two data points. The similarity weight is defined as the number of nearest neighbors that points share (intersection). So for example, if they do not share any point, the weight will be zero, and therefore, the link is removed from the graph.

4. Compute the SNN density of each point. Here the user needs to specify the Eps value, which is the minimum similarity that two points need to share to be considered as in the neighbor of the other. The algorithm finds the number of points that have an SNN similarity of Eps or grater to each point, which is defined as the SNN density of the point.

5. Find the core points by keeping all those data points that have a SNN density greater than MinPts, which is as a fraction of the K nearest neighbors.

6. Build clusters from the defined core points, two core points are merged in the same clusters when each other have a similarity equal or grater than Eps, in other words, within a radius Eps.

7. Remove all those points that they are not found within the radius Eps of any core points, these points are classified as noise.

8. Finally, assign all the rest of data points (which are neither noise or core-points) to the nearest core point by considering the sheared neighbor similarity.

## 2.1 SNN clustering algorithm pseudocode

The SNN clustering algorithm is summarized in Algorithm 1.

**Algorithm 1** SNN Clustering algorithm

```
1: procedure SNN(X)
2:     K ← number of nearest neighbors
3:     Eps ← minimum similarity
4:     MtsPts ← fraction of K
5:     similarity_matrix ← get_similarity_matrix(X,K)
6:     snn_graph ← get_snn_graph(similarity_matrix)
7:     core_points, labels ← DBSCAN(snn_graph,Eps,MinPts)
8:
   return labels
```

## 2.2 Code implementation

The implementation of the SNN clustering algorithm has done with Python 3 following the API conventions used by the library `scikit-learn`. A class for the SNN clustering approach has been created using the base classes that the DBSCAN algorithms use, the `base.BaseEstimator` and `base.ClusterMixin`, to define the initial parameters and the fit prediction to perform data clustering and obtain the cluster labels. These base classes have been applied because the SNN clustering algorithm is an extension of the DBSCAN approach.

The implementation of the similarity matrix and SNN graph, steps 1-3, fundamentally relies in `scikit-learn` functions and `numpy` arrays in order to obtain a more efficient algorithm. Then, to find the core points of the clusters and the cluster labels for each sample in the dataset it has been used the DBSCAN implementation in `scikit-learn`. However, this DBSCAN function has been designed to use as data a sparse matrix of distances not similarities, so, some little changes in the original code must be done. As the spatial similarity is the opposite to the spatial distance (distance = 1-similarity), what it has been done is to implement distances and not similarities in the SNN graph.

For the of the K-means, spectral clustering, DBSCAN and OPTICS it has been used `scikit-learn` implementations, but, for the CURE algorithm it was used the implementation in `pyclustering` library. `Pyclustering` does not follow the same conventions that `scikit-learn`, however, it was worth it to use it as it was first tested and it worked pretty well, besides, the obtained results with it are consistent.

## 3 Experimental results

The main goal of this course work is to compare experimentally the algorithms described in the 1.1 section against the SNN clustering algorithm to test its properties using artificial and real world datasets. In order to achieve this objective, the K-means, Spectral Clustering, DBSCAN and OPTICS algorithms have been compared to SNN clustering algorithm by using their implementations in `scikit-learn`, which are well-stablished and widely used. In the case of CURE algorithm, the implementation is not based in `scikit-learn` library, instead, it has been found `pyclustering` which is an open source data mining library written in Python and C++ that provides some clustering algorithms such as CURE. The Chameleon algorithm [5] was also interesting to implement in order to do the comparison, as it is used in the SNN clustering paper, but there is lack of Python implementation.

### 3.1 Evaluation metrics

To test the properties of the algorithms motioned above it is important to perform a good evaluation, not only in observing the clustering results. The evaluation of unsupervised algorithms is difficult as usually you are dealing with data that do not have a true result, however, by chance in this work we will use artificial and real world data that have been previously classified which will be helpful to analyse correctly the algorithms performance. To study the cluster quality criteria different evaluation methodologies have been proposed in this work:

- **External criteria/indices:** comparison of the predicted results with the ground truth (labeled data).
  - *Adjusted mutual information (AMI):* it is a variation of the mutual information (MI) score. The MI index it is a dimensionless quantity that tells us how much information is shared between two clusters (true and predicted), so, AMI score corrects the effect of agreement solely due to chance between clustering. It ranges from 0 (random partitions) to 1 (when the two partitions are identical). AMI is preferred when the ground truth clustering is unbalanced and there exist small clusters [7].
  - *Adjusted rand index (ARI):* it computes a similarity measure between two clustering considering all pairs of samples, and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. It ranges from 0 (random labeling) to 1 (clusterings are identical). Use ARI when the ground truth clustering has large equal clusters [7].
  - *Fowlkes and Mallows score:* measure of similarity of two clusterings (predicted and true) which is defined as the geometric mean between the precision and recall. It ranges between 0 (no similarity between clusterings) and 1 (similarity).
- **Internal criteria/indices:** they use the predicted results to measure the goodness of a clustering structure without using external information.
  - *Calinski Harabaz index (CHI):* ratio between the inter-cluster dispersion and the intra-cluster dispersion. Compact and well-separated clusters are expected to have high inter-cluster variance and low intra-cluster variance, leading to high values of CHI.

– *Davies Bouldin index (DBI):* defines the average similarity measure of each cluster with its most similar cluster, in other words, the maximum interclass-intraclass distance ratio. Clusters that are farther apart and less dispersed will result in low scores, near to zero.

– *Silhouette coefficient:* is the maximum class spread/variance, which is calculated by using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. It ranges from -1 (samples assigned to the wrong cluster) to 1 (samples assigned to the best cluster), values near zero indicate overlapping clusters.

## 3.2 Artificial datasets results

Artificial datasets are described as synthetic data which has not been obtained by direct measurement. In this experiments data of different sizes and sample-densities coming from artificial datasets have been used. The use of artificial data sets is helpful when we have interest in analysing how clusters are created when algorithms deal with data containing clusters of different shapes, sizes and densities as well as noisy clusters. Thus, they are useful to study the performance of clustering algorithms.

### 3.2.1 Small artificial datasets

Artificial datasets used in the ”Comparing different clustering algorithms on toy datasets” example from `scikit-learn` [1] has been selected as the example uses interesting small 2D datasets (1500 samples) which contain different cluster shapes: noisy concentric circles, noisy moons, blobs of sparse globular shape, blobs of non-globular shape, blobs of compacted globular shape and square. In Figure 2 the results of the clustering algorithms is displayed for the different datasets.

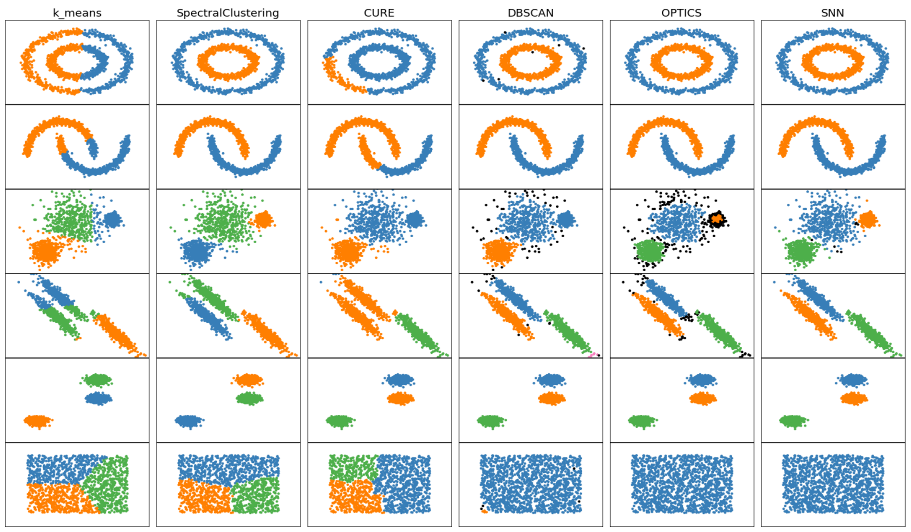


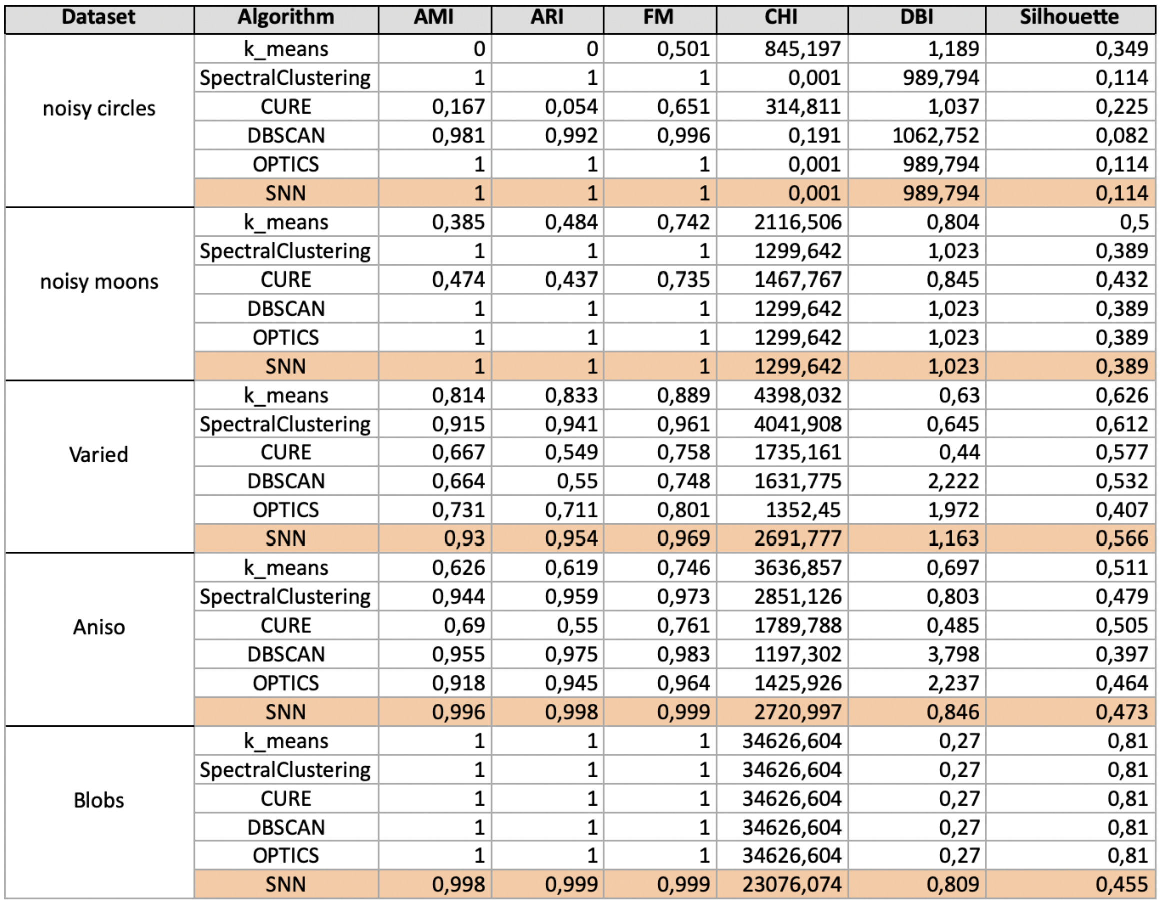
Figure 2: Comparison of different clustering algorithms in small artificial datasets. From top to bottom rows: noisy concentric circles, noisy moons, blobs of sparse globular shape, blobs of non-globular shape, blobs of compacted globular shape and square datasets.

One can observe in Figure 2 the inability of K-means algorithm to find non-globular clusters, of the 6 small dataset, the algorithm is only able to cluster correctly those data sets that contain globular

[1]https://scikit-learn.org/stable/auto$_e xamples/cluster/plot_c luster_c omparison.html sphx - glr - auto - examples - cluster - plot - cluster - comparison - py$

structures (third and fifth rows). This phenomenon is due the K-means centroid-based definition which tends to create hyperspherical clusters. Furthermore, we can observe in six row, square dataset, how the K-means approach depends on the initialization parameters, in this case we said that data contain 3 clusters and the algorithm always find the number of clusters that has been given even though it is wrong. This effect is also observed in the Spectral clustering algorithm, which is also dependent on the initialization. However, in the other datasets the spectral algorithm in contrast with K-means, doesn't tend to find globular clusters thanks to deal with connectivity and not compactness.

Regarding the CURE clustering algorithm we can notice that many of the natural clusters are broken, such as the noisy concentric circles (first row), the noisy moons (second row) and the blobs of non-globular shape (fourth row). In this cases it ignores the inter-connectivity information of the clusters and tends to find globular structures such is clearly seen in the noisy moons dataset (second row). In addition, it is also sensitive to initialization parameters, as seen in the square data set (sixth row).



| Dataset | Algorithm | AMI | ARI | FM | CHI | DBI | Silhouette |
|---|---|---|---|---|---|---|---|
| noisy circles | k_means | 0 | 0 | 0,501 | 845,197 | 1,189 | 0,349 |
| | SpectralClustering | 1 | 1 | 1 | 0,001 | 989,794 | 0,114 |
| | CURE | 0,167 | 0,054 | 0,651 | 314,811 | 1,037 | 0,225 |
| | DBSCAN | 0,981 | 0,992 | 0,996 | 0,191 | 1062,752 | 0,082 |
| | OPTICS | 1 | 1 | 1 | 0,001 | 989,794 | 0,114 |
| | SNN | 1 | 1 | 1 | 0,001 | 989,794 | 0,114 |
| noisy moons | k_means | 0,385 | 0,484 | 0,742 | 2116,506 | 0,804 | 0,5 |
| | SpectralClustering | 1 | 1 | 1 | 1299,642 | 1,023 | 0,389 |
| | CURE | 0,474 | 0,437 | 0,735 | 1467,767 | 0,845 | 0,432 |
| | DBSCAN | 1 | 1 | 1 | 1299,642 | 1,023 | 0,389 |
| | OPTICS | 1 | 1 | 1 | 1299,642 | 1,023 | 0,389 |
| | SNN | 1 | 1 | 1 | 1299,642 | 1,023 | 0,389 |
| Varied | k_means | 0,814 | 0,833 | 0,889 | 4398,032 | 0,63 | 0,626 |
| | SpectralClustering | 0,915 | 0,941 | 0,961 | 4041,908 | 0,645 | 0,612 |
| | CURE | 0,667 | 0,549 | 0,758 | 1735,161 | 0,44 | 0,577 |
| | DBSCAN | 0,664 | 0,55 | 0,748 | 1631,775 | 2,222 | 0,532 |
| | OPTICS | 0,731 | 0,711 | 0,801 | 1352,45 | 1,972 | 0,407 |
| | SNN | 0,93 | 0,954 | 0,969 | 2691,777 | 1,163 | 0,566 |
| Aniso | k_means | 0,626 | 0,619 | 0,746 | 3636,857 | 0,697 | 0,511 |
| | SpectralClustering | 0,944 | 0,959 | 0,973 | 2851,126 | 0,803 | 0,479 |
| | CURE | 0,69 | 0,55 | 0,761 | 1789,788 | 0,485 | 0,505 |
| | DBSCAN | 0,955 | 0,975 | 0,983 | 1197,302 | 3,798 | 0,397 |
| | OPTICS | 0,918 | 0,945 | 0,964 | 1425,926 | 2,237 | 0,464 |
| | SNN | 0,996 | 0,998 | 0,999 | 2720,997 | 0,846 | 0,473 |
| Blobs | k_means | 1 | 1 | 1 | 34626,604 | 0,27 | 0,81 |
| | SpectralClustering | 1 | 1 | 1 | 34626,604 | 0,27 | 0,81 |
| | CURE | 1 | 1 | 1 | 34626,604 | 0,27 | 0,81 |
| | DBSCAN | 1 | 1 | 1 | 34626,604 | 0,27 | 0,81 |
| | OPTICS | 1 | 1 | 1 | 34626,604 | 0,27 | 0,81 |
| | SNN | 0,998 | 0,999 | 0,999 | 23076,074 | 0,809 | 0,455 |

Figure 3: External (adj. mutual info, adj. rand index and fowlkes mallows) and internal (calinski score, debies bouldin and silhouette) indices results for each clustering algorithm according to a given small data set.

DBSCAN, OPTICS and SNN clustering algorithms outperform in comparison with K-means, Spectral clustering and CURE algorithms. As we can observe in all dataset the right number of clusters is found. The main benefit of these clustering algorithm is that they are not sensitive to initialization parameters such the number of clusters that the data contain, instead, they find by their-selves this number. Also, these approaches they loose the "center" definition which usually tends to create globular structures, they work by finding clusters of high densities (DBSCAN and OPTICS) or different densities (SNN). Nevertheless, as seen in third and fourth row DBSCAN and OPTICS classify many data points located in the cluster boundaries as no belonging to any cluster whereas SNN don't.

Figure 3 shows the scores of the external and internal indices for the noisy concentric circles, noisy moons, blobs of sparse globular shape, blobs of non-globular shapes and blobs of compacted globular shape data set. The SNN clustering algorithm show the best external index results for all the data sets except the compact globular blobs (but slightly close to 1), meaning, that it is the approach that has come closest to the real clusters. On the other hand, regarding the internal indices we cannot compare the results between data sets for the CHI and DBI scores as they are not normalized as the silhouette score. Furthermore, not always all these indexes are well accepted as we must choose the metric according to the type of data. For example, the results obtained for the noisy circles data set we will say that the silhouette is the good criterion of quality as it doesn't takes into account the dispersion and separation of the clusters. Here, cluster are near and the external circle is surrounding the data of the internal, which could lead to missinterpretations in CHI and DBI indices. The same occurs with the noisy moons dataset, the silhouette score is the one that performs better. Regarding the three blobs datasets, the best choice is the CHI score as here it is important to measure the dispersion parameter, and according to the best internal indices it obtains the highest values for the same clustering algorithms.

### 3.2.2 Complex shapes artificial datasets

In this section we have analyzed artificial dataset that are constituted of complex cluster shapes [2]: complex9, cure-t0-2000n-2D, cure-t1-2000n-2D and 3-spiral. The first was used in the SNN clsutering paper [1] to test the performance. The vast majority of datasets include are large number of data points in comparison with the previous section (except the 3-spiral dataset).

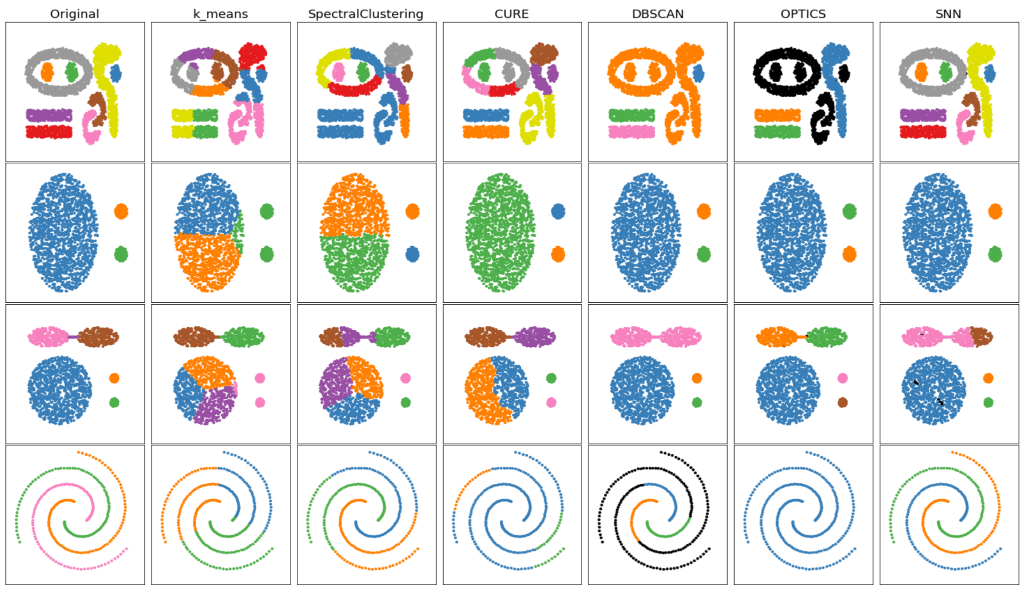


Figure 4: Comparison of different clustering algorithms in complex shapes artificial datasets.

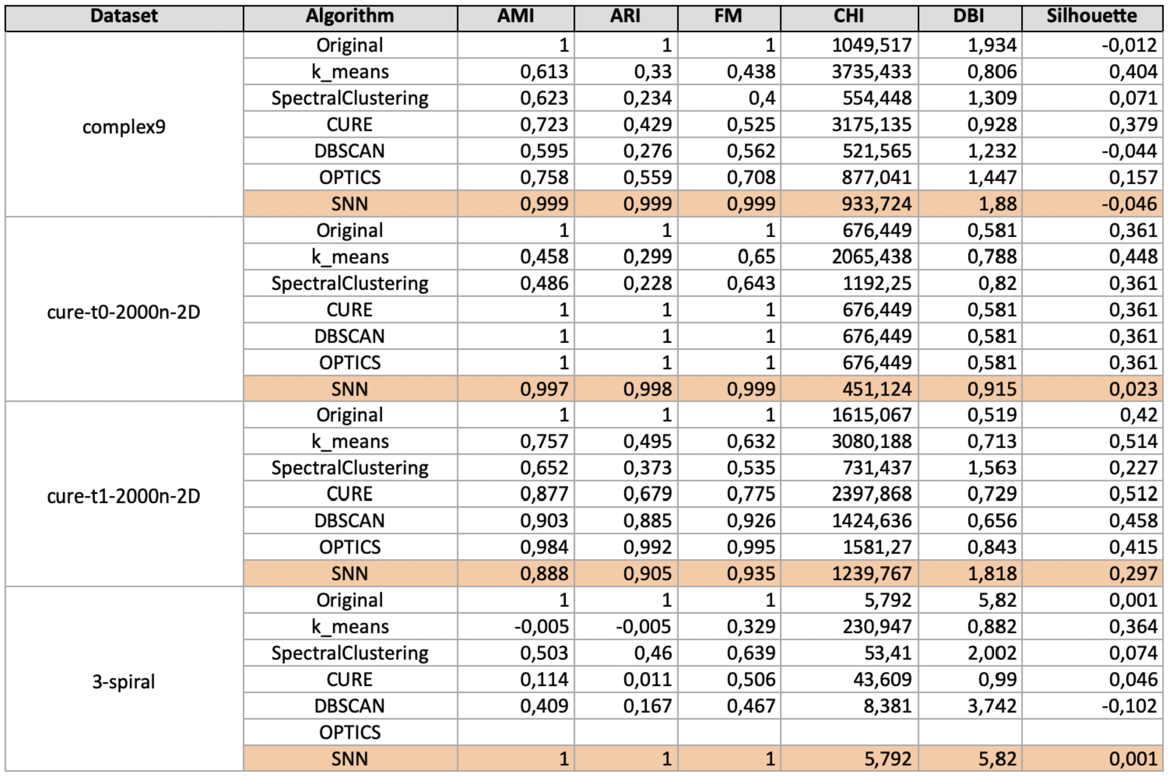
The most challenging dataset is the complex9, Figure 4 first row, as it contains a total of 9 clusters.

[2]https://github.com/milaan9/Clustering-Datasets

SNN clustering algorithm obtains almost the same results as the ground truth. Non of the other clustering algorithms are able to find such complex cluster shapes. The results are also noticeable in the external evaluation metrics, Figure 5, where SNN almost reach the highest possible score in AMI, ARI and FM scores. The problem that we find is that non of the internal clustering are good criteria to evaluate the algorithms as results do not have many sense taking into account the external indices results, SNN should obtain the best values. This phenomenon it is also observed in the next two datasets.

The cure-t0-2000n-2D dataset acquired better clusters in those algorithms where usually do not tens to find globular shapes (CURE, DBSCAN, OPTICS and SNN). But, for the cure-t1-2000n-2D dataset, where 3 more clusters are included (two ellipsoids with a bar that connects them), non of the algorithms it is able to find them. However, in this case CURE obtained the highest external indices (AMI, ARI and FM).

About the 3-spiral dataset we can see how robust the SNN clustering is. It outperforms in comparison with the other algorithms by finding perfectly the three spirals. SNN is the most robust algorithm when it comes to build concentric clusters of challenging shapes and densities, the external and internal indices are the same as the ground truth. It is demonstrate how K-means and spectral clustering tend to find globular clusters. In addition, we can observe that DBSCAN fails when data points within the circles start to be mores distant.



| Dataset | Algorithm | AMI | ARI | FM | CHI | DBI | Silhouette |
|---|---|---|---|---|---|---|---|
| complex9 | Original | 1 | 1 | 1 | 1049,517 | 1,934 | -0,012 |
| | k_means | 0,613 | 0,33 | 0,438 | 3735,433 | 0,806 | 0,404 |
| | SpectralClustering | 0,623 | 0,234 | 0,4 | 554,448 | 1,309 | 0,071 |
| | CURE | 0,723 | 0,429 | 0,525 | 3175,135 | 0,928 | 0,379 |
| | DBSCAN | 0,595 | 0,276 | 0,562 | 521,565 | 1,232 | -0,044 |
| | OPTICS | 0,758 | 0,559 | 0,708 | 877,041 | 1,447 | 0,157 |
| | SNN | 0,999 | 0,999 | 0,999 | 933,724 | 1,88 | -0,046 |
| cure-t0-2000n-2D | Original | 1 | 1 | 1 | 676,449 | 0,581 | 0,361 |
| | k_means | 0,458 | 0,299 | 0,65 | 2065,438 | 0,788 | 0,448 |
| | SpectralClustering | 0,486 | 0,228 | 0,643 | 1192,25 | 0,82 | 0,361 |
| | CURE | 1 | 1 | 1 | 676,449 | 0,581 | 0,361 |
| | DBSCAN | 1 | 1 | 1 | 676,449 | 0,581 | 0,361 |
| | OPTICS | 1 | 1 | 1 | 676,449 | 0,581 | 0,361 |
| | SNN | 0,997 | 0,998 | 0,999 | 451,124 | 0,915 | 0,023 |
| cure-t1-2000n-2D | Original | 1 | 1 | 1 | 1615,067 | 0,519 | 0,42 |
| | k_means | 0,757 | 0,495 | 0,632 | 3080,188 | 0,713 | 0,514 |
| | SpectralClustering | 0,652 | 0,373 | 0,535 | 731,437 | 1,563 | 0,227 |
| | CURE | 0,877 | 0,679 | 0,775 | 2397,868 | 0,729 | 0,512 |
| | DBSCAN | 0,903 | 0,885 | 0,926 | 1424,636 | 0,656 | 0,458 |
| | OPTICS | 0,984 | 0,992 | 0,995 | 1581,27 | 0,843 | 0,415 |
| | SNN | 0,888 | 0,905 | 0,935 | 1239,767 | 1,818 | 0,297 |
| 3-spiral | Original | 1 | 1 | 1 | 5,792 | 5,82 | 0,001 |
| | k_means | -0,005 | -0,005 | 0,329 | 230,947 | 0,882 | 0,364 |
| | SpectralClustering | 0,503 | 0,46 | 0,639 | 53,41 | 2,002 | 0,074 |
| | CURE | 0,114 | 0,011 | 0,506 | 43,609 | 0,99 | 0,046 |
| | DBSCAN | 0,409 | 0,167 | 0,467 | 8,381 | 3,742 | -0,102 |
| | OPTICS | | | | | | |
| | SNN | 1 | 1 | 1 | 5,792 | 5,82 | 0,001 |

Figure 5: External (AMI, ARI and FM) and internal (CHI,DBI and silhouette) indices results for each clustering algorithm for complex artificial datasets.

### 3.2.3 Different density artificial datasets

The main novel benefit of the SNN clustering algorithm is its capacity to outperform in clusters of different densities. Here, it has been studied the clustering results of the algorithm in four artificial data sets with varying density: triangle1, triangle2, st900 and compound. For these datasets, we have to focus on the internal indices and the the CHI index as our interest is to analyse the dispersion parameter.

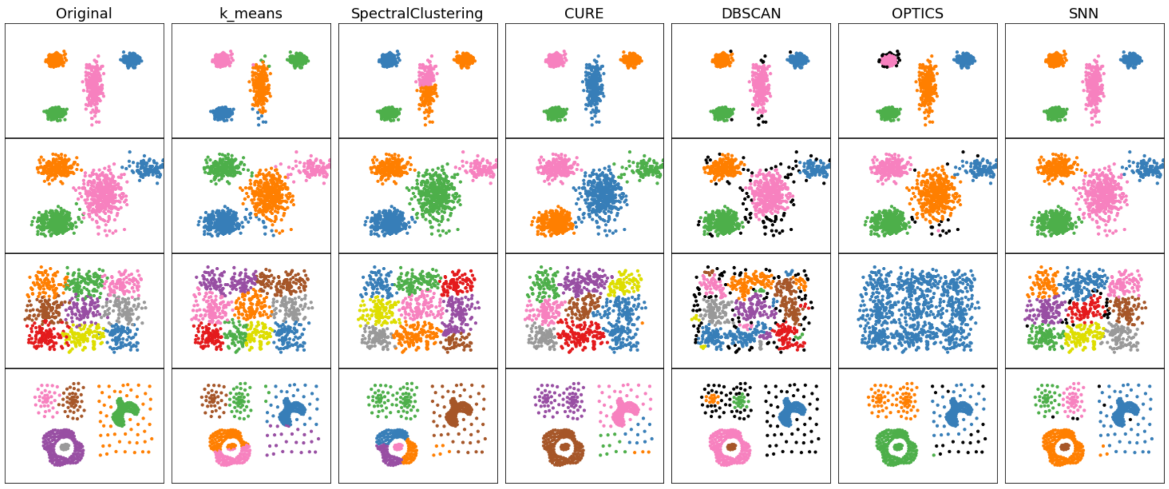
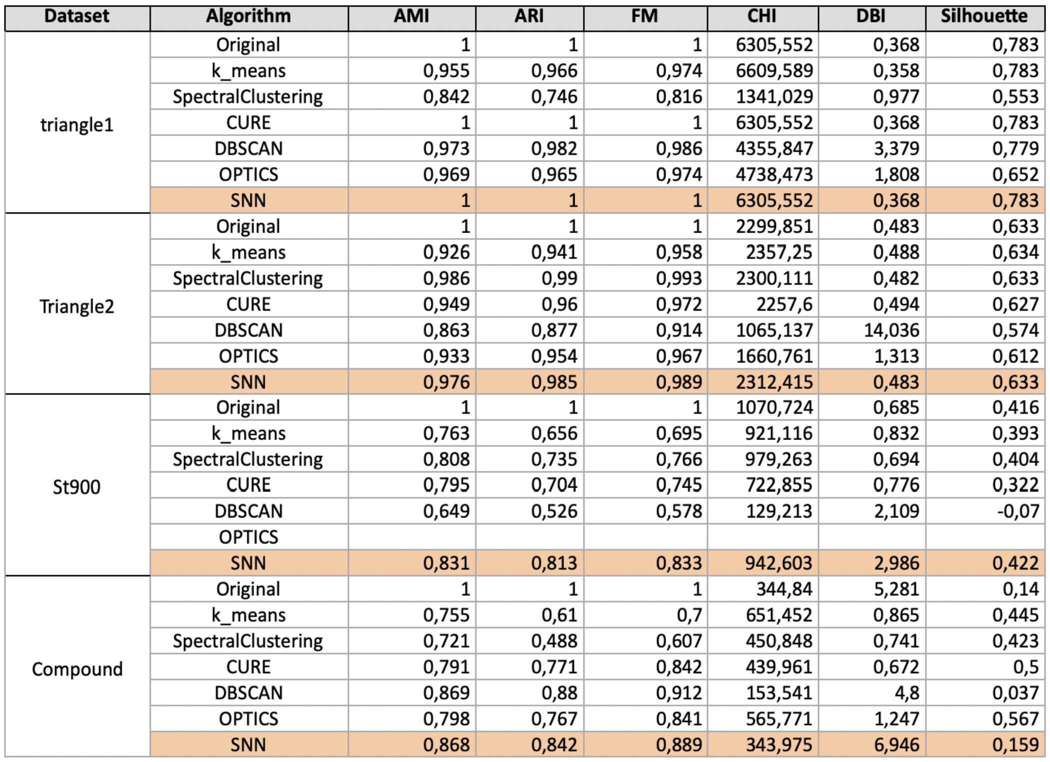


Figure 6: Comparison of different clustering algorithms in artificial datasets where clusters have different densities. From top to bottom rows: triangle1, triangle2, st900 and compound datasets.

The triangle1 and triangle2, 6 first and second row respectively, show the same clusters but data is more sparse in triangle2. By comparing the clusters results we can see how the CURE and the SNN clustering algorithms are the ones that achieve the best results, also proven in the indices displayed in Figure 7.

In the case of the st900 dataset, Figure 6 third row, we can see that all algorithms except OPTICS achieve to find the 9 clusters. However, the evaluation results in Figure 7 say that SNN clustering algorithm is the most robust when dealing with clusters of low density.

It is interesting, to know how the algorithms behave in a challenging datasets of varying cluster densities. In Figure 6 fourth row, we can observe the compound data set. The first column show how the data is composed of 6 clusters, where the most difficult to classify seems to be the orange one, which contains a dense cluster inside. The clustering results show how any of the algorithms is able to find this cluster, neither SNN clsutering. Nonetheless, the SNN clustering and DBSCAN approaches are the only ones that correctly find the other 5 clusters. If we observe the evaluation results, Figure 7, DBSCAN performs better (higher AMI, ARI and FW scores) as for the most challenging cluster (color orange in the original plot) does not classify any point while SNN does.



| Dataset | Algorithm | AMI | ARI | FM | CHI | DBI | Silhouette |
|---|---|---|---|---|---|---|---|
| triangle1 | Original | 1 | 1 | 1 | 6305,552 | 0,368 | 0,783 |
| | k_means | 0,955 | 0,966 | 0,974 | 6609,589 | 0,358 | 0,783 |
| | SpectralClustering | 0,842 | 0,746 | 0,816 | 1341,029 | 0,977 | 0,553 |
| | CURE | 1 | 1 | 1 | 6305,552 | 0,368 | 0,783 |
| | DBSCAN | 0,973 | 0,982 | 0,986 | 4355,847 | 3,379 | 0,779 |
| | OPTICS | 0,969 | 0,965 | 0,974 | 4738,473 | 1,808 | 0,652 |
| | SNN | 1 | 1 | 1 | 6305,552 | 0,368 | 0,783 |
| Triangle2 | Original | 1 | 1 | 1 | 2299,851 | 0,483 | 0,633 |
| | k_means | 0,926 | 0,941 | 0,958 | 2357,25 | 0,488 | 0,634 |
| | SpectralClustering | 0,986 | 0,99 | 0,993 | 2300,111 | 0,482 | 0,633 |
| | CURE | 0,949 | 0,96 | 0,972 | 2257,6 | 0,494 | 0,627 |
| | DBSCAN | 0,863 | 0,877 | 0,914 | 1065,137 | 14,036 | 0,574 |
| | OPTICS | 0,933 | 0,954 | 0,967 | 1660,761 | 1,313 | 0,612 |
| | SNN | 0,976 | 0,985 | 0,989 | 2312,415 | 0,483 | 0,633 |
| St900 | Original | 1 | 1 | 1 | 1070,724 | 0,685 | 0,416 |
| | k_means | 0,763 | 0,656 | 0,695 | 921,116 | 0,832 | 0,393 |
| | SpectralClustering | 0,808 | 0,735 | 0,766 | 979,263 | 0,694 | 0,404 |
| | CURE | 0,795 | 0,704 | 0,745 | 722,855 | 0,776 | 0,322 |
| | DBSCAN | 0,649 | 0,526 | 0,578 | 129,213 | 2,109 | -0,07 |
| | OPTICS | | | | | | |
| | SNN | 0,831 | 0,813 | 0,833 | 942,603 | 2,986 | 0,422 |
| Compound | Original | 1 | 1 | 1 | 344,84 | 5,281 | 0,14 |
| | k_means | 0,755 | 0,61 | 0,7 | 651,452 | 0,865 | 0,445 |
| | SpectralClustering | 0,721 | 0,488 | 0,607 | 450,848 | 0,741 | 0,423 |
| | CURE | 0,791 | 0,771 | 0,842 | 439,961 | 0,672 | 0,5 |
| | DBSCAN | 0,869 | 0,88 | 0,912 | 153,541 | 4,8 | 0,037 |
| | OPTICS | 0,798 | 0,767 | 0,841 | 565,771 | 1,247 | 0,567 |
| | SNN | 0,868 | 0,842 | 0,889 | 343,975 | 6,946 | 0,159 |

Figure 7: External (AMI, ARI and FM) and internal (CHI,DBI and silhouette) indices results for each clustering algorithm for artificial datasets of different density.

## 3.3 Real world data sets results

### 3.3.1 Small dataset - Iris

The Iris data base [3] is a small multivariate data set created by the British scientist R. Fisher. The dataset contains 150 numerical instances under five attributes: petal length, petal width, septal width, septal length and species ("class" attribute"). It has 3 classes of 50 instances each ("Iris-virginica", "Iris-setosa" and "Iris-versicolor"). This data set is often used for testing out machine learning algorithms.

### 3.3.2 Medium dataset - Breast cancer

The breast cancer data base [4] represents characteristics of cell nuclei in images from a breast mass. The dataset contains 569 numerical instances under 30 attributes where instances can are binary classified. This data set is more complex as the number of dimensions is much higher than in the iris dataset, here the SNN clustering algorithm is tested under high dimensional problems.

In Figure 8 the clustering results of the iris (first row) and brest cancer (second row) datasets are displayed. We can notice how the K-means, spectral clustering and SNN clustering algorithms are the only data sets that achieve to find the true number of clusters. However, if we observe the external

[3] `https://archive.ics.uci.edu/ml/datasets/Iris`
[4] `http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29`

and internal indices for these three algorithms, Figure 9, it is clearly how SNN achieves results more similar to the ground truth for external indices (adjusted mutual information, adjusted rand index and Fowlkes Mallows) even though in the plot some of the point aren't classified. The same happens with the Silhouette score, which is the one that gives a good quality measure for these datasets, SNN achieves the lowest value.

Regarding the CURE, DBSCAN and OPTICS it is clearly that nor of them have a good performance in real world data sets as it can be either seen in the clustering results (Figure 8) or the evaluation metrics (Figure 9).

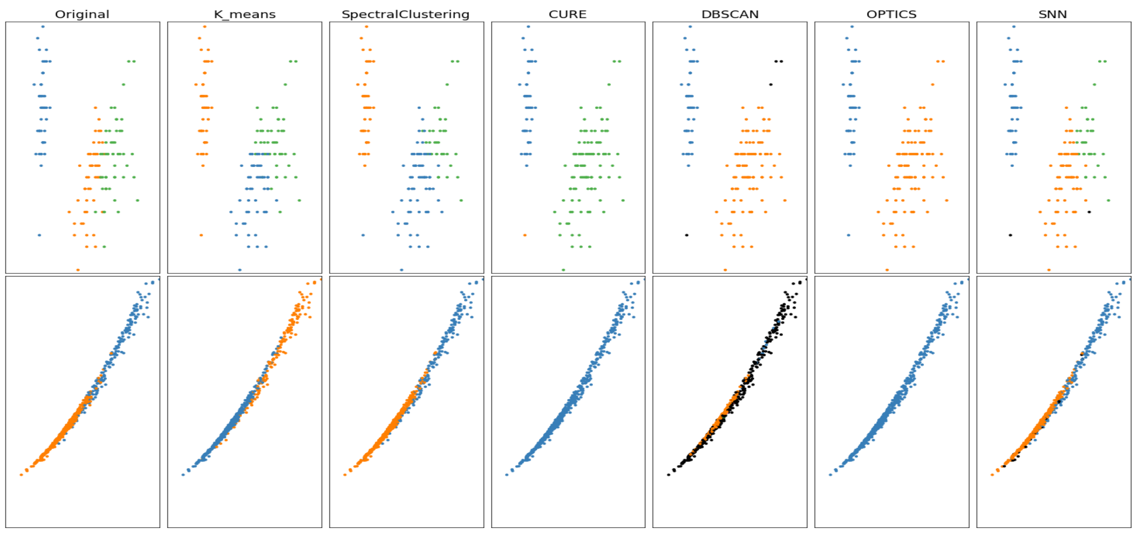
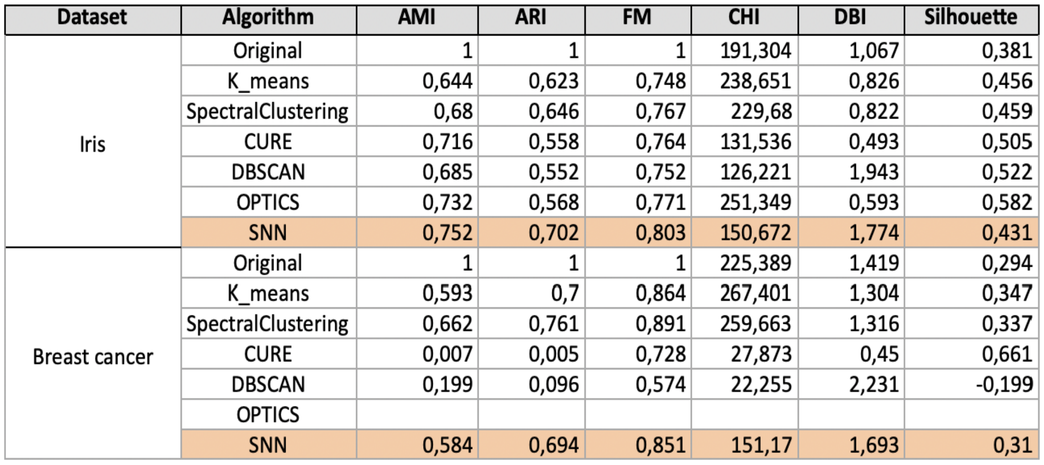


Figure 8: Comparison of different clustering algorithms in real world data sets. Iris dataset: first row. Breast Cancer dataset: second row.



| Dataset | Algorithm | AMI | ARI | FM | CHI | DBI | Silhouette |
|---|---|---|---|---|---|---|---|
| Iris | Original | 1 | 1 | 1 | 191,304 | 1,067 | 0,381 |
| | K_means | 0,644 | 0,623 | 0,748 | 238,651 | 0,826 | 0,456 |
| | SpectralClustering | 0,68 | 0,646 | 0,767 | 229,68 | 0,822 | 0,459 |
| | CURE | 0,716 | 0,558 | 0,764 | 131,536 | 0,493 | 0,505 |
| | DBSCAN | 0,685 | 0,552 | 0,752 | 126,221 | 1,943 | 0,522 |
| | OPTICS | 0,732 | 0,568 | 0,771 | 251,349 | 0,593 | 0,582 |
| | SNN | 0,752 | 0,702 | 0,803 | 150,672 | 1,774 | 0,431 |
| Breast cancer | Original | 1 | 1 | 1 | 225,389 | 1,419 | 0,294 |
| | K_means | 0,593 | 0,7 | 0,864 | 267,401 | 1,304 | 0,347 |
| | SpectralClustering | 0,662 | 0,761 | 0,891 | 259,663 | 1,316 | 0,337 |
| | CURE | 0,007 | 0,005 | 0,728 | 27,873 | 0,45 | 0,661 |
| | DBSCAN | 0,199 | 0,096 | 0,574 | 22,255 | 2,231 | -0,199 |
| | OPTICS | | | | | | |
| | SNN | 0,584 | 0,694 | 0,851 | 151,17 | 1,693 | 0,31 |

Figure 9: External (AMI, ARI and FM) and internal (CHI,DBI and silhouette) indices results for each clustering algorithm for small and medium real world.

### 3.3.3 Large dataset - KDD CUP '99 Network Intrusion

In this section we present experimental results from the KDD CUP '99 data set which is widely used for benchmarking network intrusion detection system. Also, it has been studied in by L. Ertöz et al. [1] to test the performance of the SNN clustering algorithm in comparison with the K-means approach.

The dataset contain TCP sessions whose 42 attributes include TCP connection properties, network traffic statistics using a 2 second window and other attributes extracted from TCP packets. The attack sessions can take 5 possible values: normal, u2r, dos, r2l or probe. The original dataset is very large, therefore, we sampled the data by picking 1000 samples per class, this sampling resulted in a dataset of 6878 samples.

The data was clustered using K-means and SNN clustering algorithm with K=300. Tables 1 and 2 show the predicted instances and F1 scores of the clusters for K-means and SNN respectively. Here, the F1 score has been computed as it gives a better measure of the incorrectly classified cases than accuracy. In those table we can see that the F1 score for the rare case, u2r (user to root attack), neither of the algorithms is able to detect any correct sample. The clusters that the SNN algorithms found for dos and probe classes are superior to K-means clusters, where K-means doesn't detect anything for r2l class. However, the SNN algorithm fails in the normal and probe classes, as the F1 score is much worse than the F1 score in K-means.

The size of the largest K-means clusters is 412 and the largest cluster obtained with the SNN algorithm is of 712. Even though, SNN detected 6 clusters in the output, the largest cluster is much bigger than the largest K-means cluster, results consistent with L. Ertöz et al. work [1]. In particular, for the rare case, u2r (user to root attack) K-means does not detect any data point correctly whereas SNN picks up 280 out of 986. The clusters that the SNN algorithm found are always superior except fr the probe class.

| Class Attack | Total | Predicted | Correct | Incorrect | F1 score |
|---|---|---|---|---|---|
| dos | 1000 | 2011 | 190 | 1821 | 0.13 |
| normal | 1000 | 741 | 186 | 555 | 0.21 |
| probe | 1000 | 1362 | 412 | 950 | 0.35 |
| r2l | 1000 | 601 | 0 | 601 | 0.00 |
| u2r | 1000 | 285 | 0 | 285 | 0.00 |

Table 1: Results of the K-means clustering algorithm for the KDD CUP '99 Network Intrusion dataset

| Class Attack | Total | Predicted | Correct | Incorrect | F1 score |
|---|---|---|---|---|---|
| dos | 1000 | 1321 | 710 | 611 | 0.61 |
| normal | 1000 | 1477 | 626 | 851 | 0.51 |
| probe | 1000 | 350 | 35 | 315 | 0.05 |
| r2l | 1000 | 494 | 159 | 335 | 0.21 |
| u2r | 1000 | 986 | 280 | 706 | 0.28 |

Table 2: Results of the SNN clustering algorithm for the KDD CUP '99 Network Intrusion dataset

In general, by observing the accuracy and F1 macro and weighted averages, Table 3, the SNN clustering algorithm performs better than K-means in highly dimensional data. Here no external indices have been computed because as we're dealing with very high dimensional data such measures are sometimes non-trivial to interpret. It has been demonstrated by N. Tomasev and M. Radovanovic [8] that

Silhouette score is a robust and effective clustering quality criteria for evaluating high dimensional data, so is the measure that has been used in addition of the F1 score averages. We can see that Silhouette index in the case of SNN algorithm is much closer to 0 than for the K-means approach. However, although the results are better in SNN,21¡ both algorithms are not good alternatives for this type of dataset.

| Algorithm | Accuracy (%) | F1 macro average (%) | F1 weighted average (%) | Silhouette |
|---|---|---|---|---|
| K-means | 16 | 14 | 14 | 0.725 |
| SNN | 36 | 28 | 33 | 0.288 |

Table 3: Accuracy, F1 score macro average and F1 score weighted average results for the KDD CUP '99 Network Intrusion dataset

## 4 Discussion and limitations

In this laboratory work it has been implemented the SNN clustering technique described by L.Ertöz et al. [1] that can find clusters of varying sizes, shapes and densities. The algorithm has been compared with other well known clustering techniques (K-means, spectral clustering, CURE, DBSCAN and OPTICS) to show its performance under the described clusters.

It has been demonstrated that the SNN algorithm can handle artificial data that contains complex clusters, varying densities and even in the presence of noise, and it can automatically determine the number of clusters. Thus, the algorithm is a robust and good alternative to many others clustering techniques that have limitations such as specifying in advance the number of clusters (K-means and spectral clustering).

In particular, the SNN algorithm outperforms in small artificial datasets as it is able to cluster almost perfectly all data point is their true clusters. This phenomenon is reflected in the external and internal evaluation indices (Figure 3). Both SNN and spectral clustering surpass the performance of traditional K-means, when clusters do not follow a globular shape. Spectral clustering it is preferred if we care about the computational time, as it is much faster than SNN, but, SNN do not requires the number of cluster. So, if you already know how many clusters data has the best option would be Spectral clustering, otherwise, SNN would be the best option.

In addition, for artificial datasets containing complex clustering shapes and more large data, the SNN algorithm does the best clustering in challenging shapes such as complex9 and 3-spiral data, while other algorithms fail. However, sometimes, such as in cure-t1-2000n-2D dataset, SNN is not able to detect correctly horizontal clusters that have its boundaries in contact, in this case OPTICS behaves better.

In the case of varying density data, SNN algorithm can find the clusters that represent uniform regions with respect to their surroundings in low and medium density (triangle1, triangle2 and st900. On the the other hand, when clusters of low and medium densities are combined with high density and well defined boundaries, compound dataset, SNN fails in finding clusters that share space in common and are of high and low density. Under this circumstances, DBSCAN is more cautious and do not classify those low density data points when their cluster are not well defined while SNN does.

Real world datasets are not the best option to show the good properties of SNN but, in any case, it's the type of data that the algorithm will deal in real clustering works. It has been shown how the algorithm is able to find the true clusters in small and medium real datasets, Iris and Breast Cancer, in fact, it exceeds the performance of other algorithms that fail in classifying data points within the boundaries of two close clusters. It is important to highlight that the Breast Cancer dataset contains up to 30 attributes, which is a case of high dimensional data, but, it has a binary classification which it makes easier to classify. A more challenging dataset was tested, the KDD CUP '99 Network Intrusion data, that contains 42 attributes and 5 different attack classes. In this case, the K-means algorithm has been compared against the SNN approach. Even though, the clustering results are not good for both techniques, SNN acquired better results beacuse in general clusters are more large and robust than in the case of K-means.

The main drawback of the SNN algorithm is the computational time. Among all the studied clustering algorithms SNN is the slowest one as the computation of the SNN graph to find the K nearest neighbors of all pairwise similarities is highly costly. Furthermore, in order to find the right number of clusters the K nearest neighbor, Eps and MinPts parameters need to be tuned. Thus, taking into account the time limitation, when it comes to work with large data sets, the parameters tuning process is time consuming.

## 5 Conclusion and future work

The results displayed in this laboratory work show that SNN clustering algorithm is a robust alternative as it performs good in clusters of different sizes, shapes and densities in noisy and high dimensional artificial and real world data. While other algorithms tend to find globular clusters or do not achieve in finding clusters of low density, SNN approach, in the majority of the studied dataset, SNN obtained the best results. Nonetheless, the algorithm it has a high computational time, so, depending on the circumstances of the problem maybe other algorithms such as DBSCAN or Spectral clustering are preferred.

Future work could be related in study deeply the properties of the SNN algorithm by testing it in more types of datasets and comparing its behaviour with other algorithms such as Chameleon [5] (hierarchical clustering) that obtained good performance in low dimensional data. Another important goal of future research could be to investigate a way to find out a cheaper way of computing the sheared nearest neighbors of pairwise points.

## 6 Running the code

### 6.1 Installing pre-requisites

You need to have installed the **Python 3.7** version in your computer and the following Python libraries:

- **numpy**
- **pandas**
- **sklearn**
- **pyclustering**

## 6.2 Run

In order to execute the code you need to follow the next instructions:

1. Download the `PW1-optB-URL-JanaReventos` ZIP file
2. Open a terminal shell in the `PW1-optB-URL-JanaReventos` folder file
3. Execute the clustering algorithm comparison with the following commands:
    - `>> python3 main_artificial.py`
    - `>> python3 main_real.py`
4. Results will be displayed in the console or saved in the `Results` folder in `.csv` and `.xlsx` formats.

## References

[1] Steinbach M. Kumar V. Ertöz, L. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. *In Proceedings of the 2003 SIAM international conference on data mining*, pages 47–58, (2003, May).

[2] Sander Xu Ester, Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

[3] Rastogi R. Shim K. Guha, S. Cure: an efficient clustering algorithm for large databases. *ACM Sigmod record*, pages 73–84, 1998.

[4] Raymond Austin Jarvis and Edward A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on computers 100.11*, pages 1025–1034., 1973.

[5] Han E. H. Kumar V. Karypis, G. Chameleon: Hierarchical clustering using dynamic modeling. 1999.

[6] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory.*, page 129–137, 1982.

[7] et al. ROMANO, Simone. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research,*, pages 4635–4666, 2016.

[8] Radovanović M. Tomašev, N. Clustering evaluation in high-dimensional data. *In Unsupervised Learning Algorithms*, pages 71–107, 2016.