Clustering of Amino Acids Profiles

Samuele Zoppi

University Of Zurich, Switzerland

samuele.zoppi@uzh.ch

1 Introduction

Classifying feed data can be very challenging. When a feed sample is analyzed, an amino acids profile, composed of the values of 18 essential amino acids, is built based on the chemical analyses. This process is time consuming, expensive and, therefore, results in various mistakes. For example, due to wrong setup of chemical analyses, the derived containment of some amino acid might be different from the true value. Another example are misclassified or unclassified feed types. In this case, based on the amino acids it is possible to correctly classify the feed type or even to determine a new one. In this project we will use clustering analyses techniques on amino acids profiles to detect outliers and correctly classify the feed data. In particular we will use DBSCAN and OPTICS which are efficient and accurate density based clustering techniques for high dimensional data. The project is divided into three tasks:

1. Learning of DBSCAN and OPTICS that are described in [3, 4].
2. Implementation of the above mentioned clustering techniques in any programming language and graphical visualization of the results.
3. Experimental evaluation and comparison of the above mentioned techniques on amino acids profiles.

2 The Data

An amino acids profile is composed of the values of 18 essential amino acids: LYS, MET, CYS, THR, TRY, ILE, LEU, PHE, TYR, VAL, ARG, HIS, ALA, ASP, GLU, GLY, PRO and SER. The amino acids profiles are stored with a unique identification number and the name of the feed type that they are supposed to come from. The dataset taken into consideration is composed of 393 amino acids profiles from about 80 different feed types, where each feed type generates at least one and up to 20 amino acids profiles.

Example:

ID	1	2
Feed type	Ackerbohnen	Lupine weiss
LYS	0.057011911	0.0446280992
MET	0.009897901	0.0068870523
CYS	0.01306523	0.0155647383
THR	0.027714123	0.0341597796
TRY	0.007601588	0.0074793388
ILE	0.03404878	0.041322314
LEU	0.062554735	0.0696969697
PHE	0.040383437	0.0365013774
TYR	0.029297787	0.0443526171
VAL	0.039591605	0.0385674931
ARG	0.076807713	0.1006887052
HIS	0.022963131	0.0210743802
ALA	0.039591605	0.0308539945
ASP	0.110856493	0.0982093664
GLU	0.114815654	0.1874655647
GLY	0.041967101	0.0373278237
PRO	0.034840612	0.0373278237
SER	0.035632444	0.046969697

Table 1: Example of two amino acids profiles

3 Clustering Algorithms

In order to find clusters in the feed data, we will use two different clustering algorithms. The first one, named DBSCAN, requires the input of two variables: the minimal number of points to build a cluster MinPts and ϵ (epsilon) as a sort of density parameter for the searched clusters. The second one, named OPTICS, is a further development of DBSCAN. In this case there is no parameter epsilon and thus the clusters can vary in density.

3.1 DBSCAN

In DBSCAN the clusters are defined using the idea of density reachability [4]. Given a minimum number of points MinPts and a distance ϵ (epsilon) the following definitions are given:

- **directly density-reachable**: a point Q is directly density-reachable from a point P, if the distance between Q and P is smaller than ϵ and if the $\epsilon - neighborhood$ of P contains at least MinPts points.
- **density-reachable**: a point Q is density-reachable from a point P, if

there is a chain of points $P_1,...,P_n$, $P_1=P$, $P_n=Q$ such that P_{i+1} is directly density-reachable from P_i.

- **density-connected**: a point Q is density-connected to a point P, if there is a point O such that both P and Q are density-reachable from O.
- **noise**: a point Q is a noise from the perspective of a point P, if Q is neither directly density-reachable nor density-reachable from P.

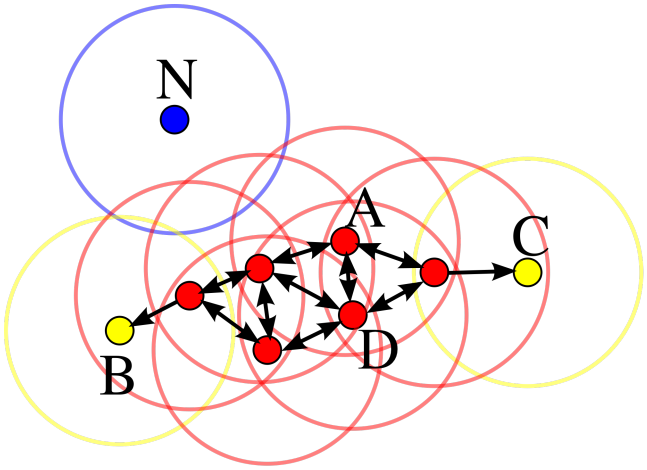


Figure 1: Point D is directly density-reachable from A, points B and C are density-reachable from A and density-connected among each other, point N is a noise [1].

A cluster is a set of points which are all at least density-connected among each others. If a point, which is not yet in the cluster, is at least density-connected with any point of the cluster, then this point will also be part of the cluster. The algorithm used to implement DBSCAN is described in Algorithm 1 in the appendix.

The runtime complexity of the algorithm is $O(n^2)$. The major disadvantage of DBSCAN is, that - in order to achieve good results - the distance epsilon should be chosen appropriately to the density of the clusters. This approach does not permit to find clusters having very different densities.

3.2 OPTICS

OPTICS, as a further development of DBSCAN, eliminates the problem of choosing an appropriate distance epsilon as an input variable and thus permits to find clusters having very different densities. OPTICS is based on the concepts of core-distance and reachability-distance. Given a minimum number of points MinPts and a distance ϵ (epsilon):

- **core-distance**: the core-distance for a point P is undefined, if the $\epsilon - neighborhood$ of P contains less than MinPts points or is the distance between P and his MinPts-th distant point.
- **reachability-distance**: the reachability-distance of a point P from a point Q is undefined, if the $\epsilon - neighborhood$ of P contains less than MinPts points, otherwise the reachability-distance is the maximum between the core-distance of P and the distance between P and Q.

The parameter epsilon is, as mentioned, not strictly necessary. It can be set to a sufficiently large number so that it is never the case that the core-distance or the reachability-distance are undefined. In this case the runtime complexity of the algorithm (Algorithm 3) is $O(n^2)$, because every neighborhood is composed from the whole dataset.

The points are output into an ordered list following a particular order, storing the smallest reachability-distance for every point. The determination of the clusters in OPTICS is done visually. The reachability-distance of every point in the ordered list is visualized as a line in a diagram, like in Figure 2, where the y-axis reports the reachability-distance and the x-axis is the sequence of the points in the ordered list.

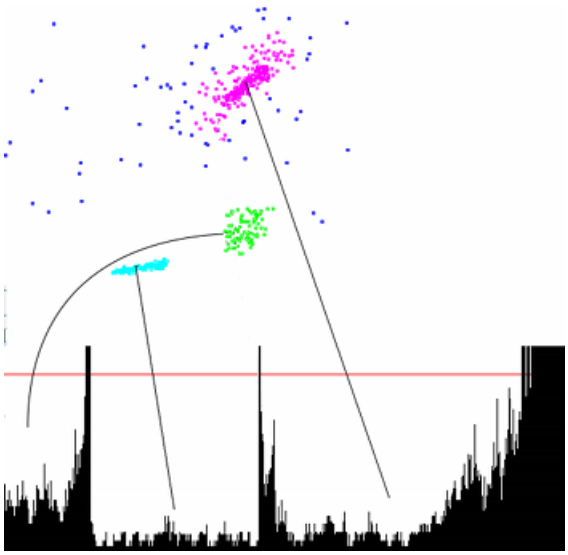


Figure 2: Visualization of the reachability-distances of the points in the ordered list [2]

The clusters can be recognized as the points forming a gap between the longest vertical lines (or longest reachability-distance values). In Figure 2, drawing a horizontal line and taking each gap between each crossed vertical line as a cluster, three different clusters can be recognized. The two dimensional points in

the top of the figure is the analyzed dataset, the three clusters are marked with different colors. If we move down the horizontal line, more clusters will appear. The choice of where to put the horizontal line is crucial for the definition of the clusters. All the points which have a reachability-distance above the horizontal line are considered as noises, which do not belong to any cluster. The visual determination of clusters can result in more work for the user, but this way (in contrast to DBSCAN) clusters having different densities can be classified.

4 Clusters Visualization Tool

In order to visualize the results of the clustering, we developed a tool which visualizes the clusters found. The tool only works on DBSCAN, because in order to find the clusters using OPTICS, manual work is needed.

The initial interface (Figure 3) asks the users to enter the values for the minimal number of points needed to build a cluster (MinPts) and a distance epsilon.

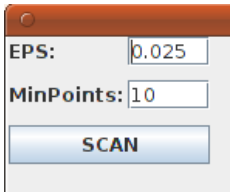


Figure 3: Input of the variables

After pressing on the SCAN button, the clusters found are displayed sorted by their dimension (Figure 4).

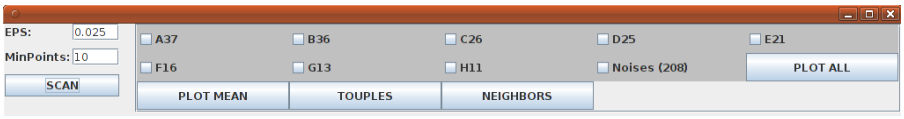


Figure 4: Visualization of the results

The number of points contained in each cluster is displayed as a part of the name of the cluster, e.g. "A36" contains 36 points. At this stage it is possible to select one or more clusters and launch one of the following operations:

- PLOT ALL: visualize all the points (i.e. amino acids profiles) contained in the selected clusters, where each cluster will be represented with a unique color. Figure 5.

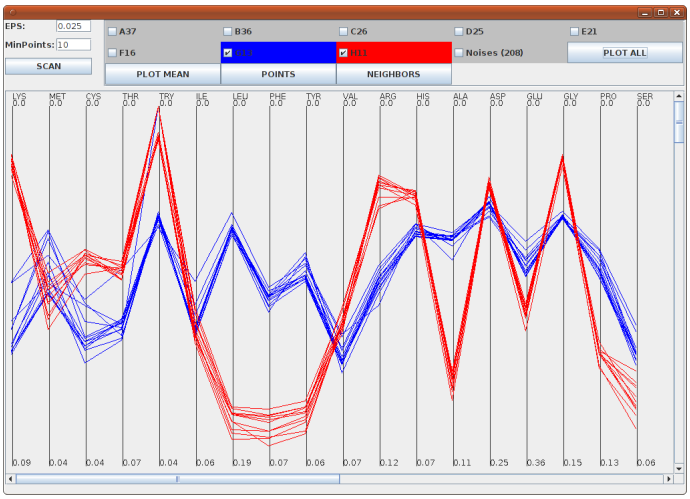


Figure 5: Visualization of the points of two clusters

For each amino acid type a vertical axis is drawn and the scale for the range of values is adjusted. The points on the vertical axis are joined with a line for every amino acids profile.

- PLOT MEAN: visualize the mean value over all points in a cluster for the selected clusters. Figure 6.

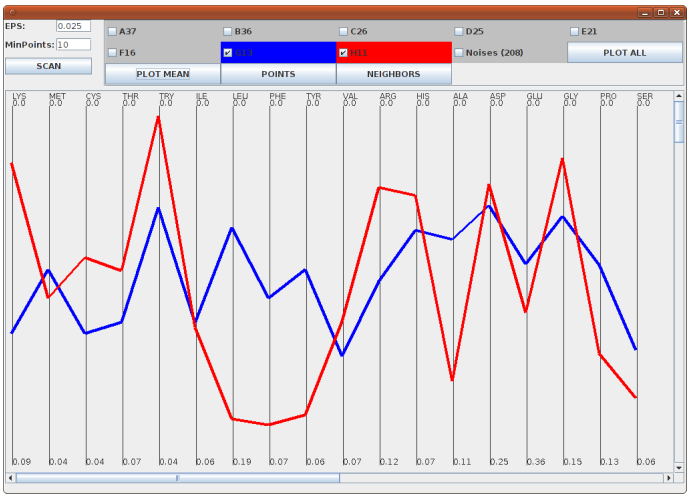


Figure 6: Visualization of the mean values for two clusters

- POINTS: output the points contained in the selected clusters, each point contains an amino acids profile, a unique identification number and the feed type for the amino acids profile. Figure 7.

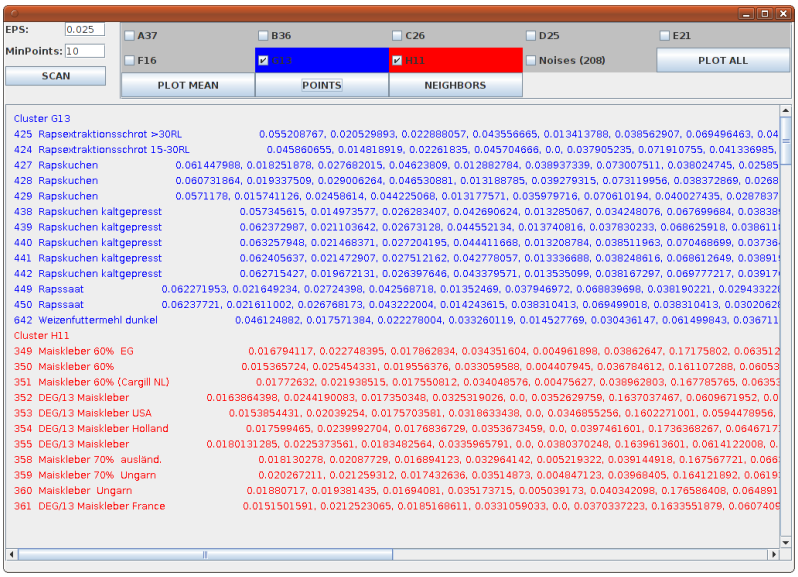


Figure 7: Output of the points contained into two clusters

At the first place in the output there is the identification number of the point, then the name of the feed type - where the amino acids profile should come from - is stated and after that the list of the values of the amino acids profile are reported. The values of the amino acids are printed out following the order of the graphical visualization (PLOT ALL).

Figure 8: Ranking list of the closest neighbors of two clusters

- NEIGHBORS: output the closest clusters for the selected clusters, i.e. a ranking list that starts with the closest cluster and ends with the most faraway. A close cluster means, that it contains amino acids profiles having values close to the selected clusters. Figure 8.

4.1 Setting MinPts and Epsilon

For an initial test MinPts should be chosen really small (like 2 or 3), epsilon can be chosen randomly as a positive number. Both parameters will be adjusted by running some experiments. The goal is to cluster the amino acids profiles belonging to the same feed type into a single cluster, where each cluster should be homogeneous. A cluster containing amino acids profiles from the same feed type is called homogeneous. By contrast, if a cluster contains amino acids profiles belonging to different feed types it is called heterogeneous. An example of a good cluster is reported in Figure 9, which contains 16 amino acids profiles derived from fish flour (Table 2) and is homogeneous.

ID	**Feed type**
105	F.Poisson 999 Danemark
107	F.Poisson 999 Danemark
108	F.Poisson72% Norvege
109	F.Poisson 999 Danemark
110	F.Poisson72% Norvege
111	F.Poisson72% Islande
112	F.Poisson 999 Danemark
114	F.Poisson 999 Danemark beh A
115	F.Poisson 999 Danemark
106	F.Poisson72% Norvege
113	F.Poisson 999 Danemark beh A
116	Fischmehl 70/72
118	Fischmehl 70/72
103	Fischmehl islaendisch
117	Fischmehl 70/72
102	Fischmehl 64%

Table 2: Amino acids profiles of a good cluster

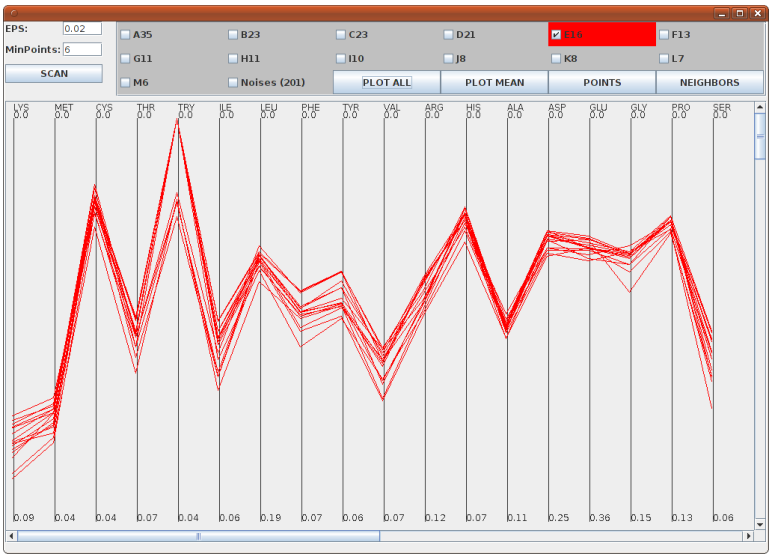


Figure 9: Example of a good cluster

In order to find a good epsilon, we can run a first test - setting MinPts to 2 - and then look at the results of the clustering. At this point the following statements can be made:

- If we have just one or few clusters with very many points in it and practically no noises, it means that epsilon is too big and we should choose a smaller one.
- If we have many noises, epsilon is probably too small.
- If the clusters are heterogeneous, epsilon is probably too big.

The consequences of changing epsilon are described in the experimental evaluation section of this document - in this section we just give the users of the clustering tool a guide how to achieve good clustering, without going into technical details. By running some experiments, a good value for epsilon should be found. After this, if we want to cut the clusters that are too small, we can simply increase MinPts until a value we consider as good.

Example of setting a good value for epsilon:

In Table 3 the results for four different experiments with MinPts set to 2 are reported. If we choose epsilon>=0.1, we would end up with a single cluster containing all the points, so epsilon should be smaller. If epsilon=0.04, we would have more clusters and few noises, but the clusters would still be too heterogeneous, so epsilon should be decreased. Choosing epsilon=0.01, we would have homogeneous clusters but this time with too many noises, so epsilon should be bigger. With epsilon=0.02 we have homogeneous clusters, a good number of

them and not too many noises, so epsilon=0.02 should be a good value for this dataset.

epsilon	MinPts	#clusters	#noises	homogeneous
0.01	**2**	49	220	YES
0.02	**2**	53	77	YES
0.04	**2**	18	14	NO
0.1	**2**	1	0	NO

Table 3: Four experiments using different pairs of epsilon and MinPts

4.2 Finding Outliers and Missclassified Points

In order to find the outliers, we have to analyze the noises found by the clustering algorithm. Looking at the noises we can distinguish and analyze these three cases:

- If a noise-point is the only point in the dataset belonging to its feed type, then it is clear that it cannot belong to any cluster, because to build a cluster we need at least two points deriving from the same feed type. Such points cannot be considered as outliers.
- If a noise-point has been classified as a noise while a cluster containing points from the same feed type exists, then this is a good indication that the noise-point could be an outlier.
- If a group of noise-points having the same feed type is found and no cluster of such feed type exists, then it is probable that these points have not built a cluster because they have a too small density. If a bigger epsilon is chosen, the points would be clustered together, so they should not be considered as outliers.

Example:
There are feed types like gastro soup, oat glume or goat whey, that have amino acids profiles, which vary quite a lot from measurement to measurement. Thus, the amino acids profiles of these feeds are often classified as noises. Indeed, if we increase epsilon, at a certain point a cluster for each of this feed type will be created and they will be clustered together.

In order to find misclassified points, we have to look at the clusters. If a cluster contains one or few points, that are not coming from the same or a close feed type like the majority of the points in the cluster, then this can be the case that these points have been misclassified, i.e. their feed type is wrongly classified and their real feed type is probably the one of the majority of the points in the cluster. If we do not know, whether two feed types are close or not, we can try to decrease epsilon and see if the points (belonging to different feed types) remain in the same cluster or not. If they remain in the same cluster,

then the feed types are either really close or there is some misclassified point in the cluster. A special case would be, if a point is at the same time misclassified and an outlier or the only one belonging to its feed type. In this case, it is not possible to recognize it as misclassified, because it would be in the noises group.

Example:

If we find a cluster, where flay seeds and sunflower seeds are together, it is because they have a very similar amino acids profile and not because some of them have been misclassified. But if we find for instance some soya beans clustered with many maize gluten, then we can be pretty sure that the amino acids profile from soya beans has been misclassified.

5 Experimental Evaluation

In this section we want to test the efficacy of the two algorithms deriving clusters and classifying noises. The analyzed dataset is composed of 393 18-dimensional points. The dataset contains amino acids profiles from about 80 different feed types, which each generates one up to 20 amino acids profiles.

5.1 DBSCAN

The efficacy of DBSCAN depends on how good the two input variables MinPts and epsilon are set. In order to do this, we ran 15 different experiments with 15 different pairs of MinPts and epsilon. The results are reported in Table 4:

epsilon	**MinPts**	**#clusters**	**#points/ cluster**	**#noises**
0.01	**2**	49	3.5	220
0.01	**5**	9	8.4	317
0.01	**10**	1	17	376
0.02	**2**	53	5.9	77
0.02	**5**	14	14.5	190
0.02	**10**	7	19.1	259
0.03	**2**	34	10.6	32
0.03	**5**	11	26.4	103
0.03	**10**	5	46.4	161
0.04	**2**	18	21	14
0.04	**5**	7	48.8	51
0.04	**10**	3	102.3	86
0.1	**2**	1	393	0
0.1	**5**	1	392	1
0.1	**10**	1	391	2

Table 4: DBSCAN experiments with different pairs of MinPts and epsilon

For each pair of epsilon and MinPts we kept track of the number of clusters found, the average number of points per cluster and the number of noises found. The results can be generalized into the following rules:

	$MinPts \nearrow$	$MinPts \searrow$	$epsilon \nearrow$	$epsilon \searrow$
#clusters	$\searrow$	$\nearrow$	$\searrow$	$\nearrow$
#points/cluster	$\nearrow$	$\searrow$	$\nearrow$	$\searrow$
#noises	$\nearrow$	$\searrow$	$\searrow$	$\nearrow$

Table 5: Effects of changing MinPts and epsilon

If the minimal number of points to build a cluster (MinPts) increases, the number of clusters will decrease, because some small clusters will not be in the results anymore, while the average number of points per cluster will increase, due to the absence of the small clusters. All this will increase the number of noises found. The inverse logic can be applied, if the minimal number of points to build a cluster decreases. If epsilon increases, the number of clusters will decrease, because some clusters will be condensed in only one, since the maximum allowed distance for a point to belong to a cluster (epsilon) increases. This way we have less clusters which contain more points. The same argumentation also applies to the number of noises, which decreases, since some of them will now be part of a cluster due to the increased distance epsilon. When epsilon decreases, the opposite happens.

In our case we cannot say anything about the number of noises expected, but we know that we have about 80 feed types, which can have only a couple (in some cases also just one) of amino acids profiles. This means that it is reasonable to set MinPts to 2. For epsilon we chose epsilon = 0.02, because this is the value that returns the biggest number of clusters, namely 53 from the approximately 80 different feed types that we have. In addition choosing epsilon = 0.02 results in homogeneous clusters. Choosing a bigger epsilon would lead us to have less but bigger and more heterogeneous clusters, which is not good since clusters should be of the same feed type. If instead we had chosen a smaller epsilon, we would have less clusters with less points per cluster and much more noises. In this case a big part of the noises can be rightly classified when epsilon is bigger. In case we had chosen a bigger MinPts, the number of clusters would be smaller, because the small - but rightly classified - clusters would have been put into the noises group.

The biggest cluster for MinPts = 2 and epsilon = 0.02 contains 35 points, the second and the third ones contain 23 points and so on until the last 21 clusters that contain only 2 points each (see Table 6).

#points/cluster	#clusters
35	1
23	2
22	1
17	1
16	1
13	1
12	1
11	1
8	3
7	1
6	1
5	1
4	9
3	8
2	21

Table 6: Clusters found for MinPts=2, epsilon = 0.02

If we look at the amino acids profiles, that are clustered together, we can notice that practically all the clusters are homogeneous, i.e. they contain exactly the same type of feed or very close ones. For instance, the biggest cluster contains only feed types having an elevated content of proteins, like soya beans, soya cake, field beans or peas, which are not of the same feed type but from very close ones. This kind of clusters - with different but close feed types - can be found in the five biggest clusters. Starting from the sixth cluster (having 16 points), all the clusters are composed of a single or maximal two different but close feed types.

Looking more carefully at all the clusters found, brings us to the conclusion that no feed type of any amino acids profile seams to be misclassified. The 77 noises found with this setup are to be attributed to one of the following cases:

1. The point is an outlier or is affected by mistakes in the measurements.
   Example: in the noises, there is an amino acid profile from sun flower seeds, which has not been clustered with a big group of other sun flower seeds profiles. In this case there were either mistakes in the measurement of the amino acids values or the point is an outlier.

2. Values for certain amino acids are missing.
   Example: in some amino acids profiles, there are missing values for some amino acids, which precludes their clustering.

3. The point is the only one of its feed type, so is it impossible to build a cluster.
   Example: there is an amino acids profile derived from tapioca root that is the only one deriving from this kind of feed type, in this case it is impossible to build a cluster.

4. There are feed types, which are very heterogeneous, i.e. they have very varying amino acids profiles. In this cases the algorithm does not manage to cluster the points of such types of feed because the density of the points is smaller than the one allowed by the parameter epsilon. This drawback of DBSCAN is managed by OPTICS automatically.

Example: in the noises we found four amino acids profiles deriving from goat whey. But if we increase epsilon to 0.04, we will find a cluster containing the goat whey's profiles.

After analyzing all the noises we identified the following number of points for each case described above, Table 7:

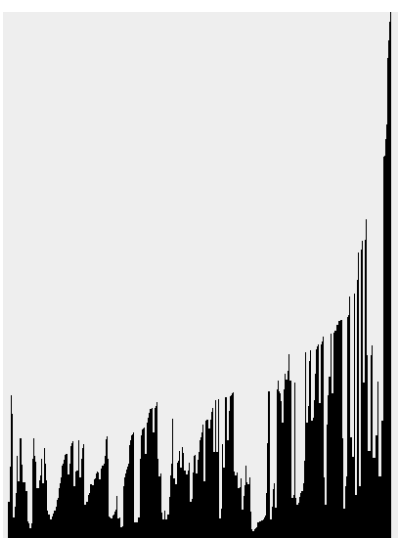
Case #	# of such points
1	11
2	13
3	13
4	40

Table 7: Number of points per case

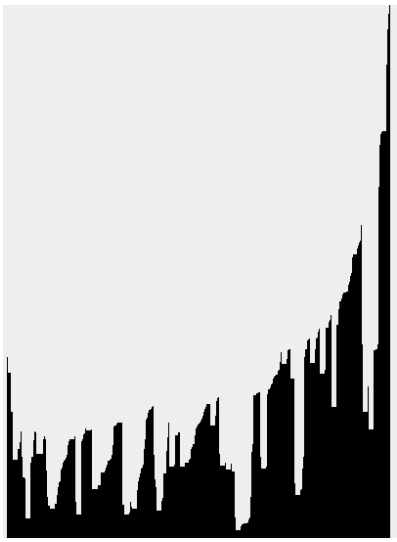
The number of outliers seams to be quite small, maybe because the dataset we received from the researcher was previously cleaned up from outliers. In conclusion we can say, that with DBSCAN it is possible to achieve good results by running various experiments. A problem remains in the cases where the dataset has large differences in densities, because in this case it is not possible to choose a pair of epsilon and MinPts that is appropriate for all the clusters. However this seams not to be a major issue using our dataset: from the 393 amino acids profiles, only about 40 could not be clustered because of this.

5.2 OPTICS

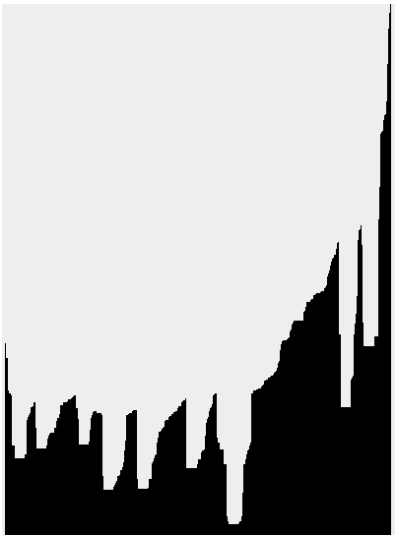
With OPTICS we have only one variable to set: the minimum number of points to build a cluster (MinPts). Figure 10 reports the reachability-distance diagram using three different MinPts.



(a) MinPts = 2



(b) MinPts = 5



(c) MinPts = 10

Figure 10: Reachability-distance diagrams for three different MinPts values

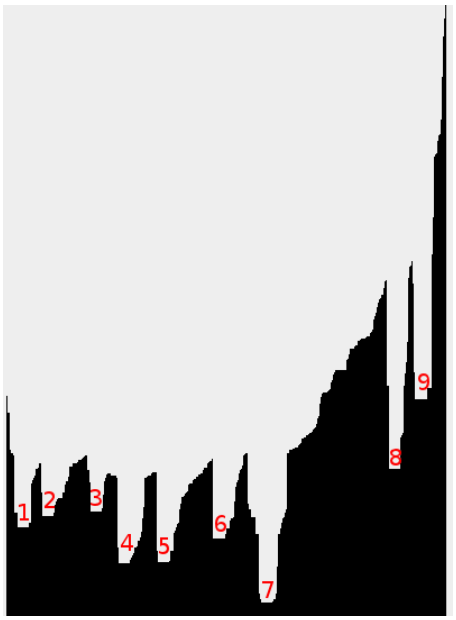


Figure 11: The nine clusters found with MinPts = 10

Position in the list	ID	Feed type
6	5	Ackerbohnen
7	653	Weizenkeime
8	**102**	**Fischmehl 64%**
9	**105**	**F.poisson 999 Danemark**
10	**117**	**Fischmehl 70/72**
11	**118**	**Fischmehl 70/72**
12	**103**	**Fischmehl islaendisch**
13	**106**	**F.poisson 72% Norvege**
14	**108**	**F.poisson 72% Norvege**
15	**109**	**F.poisson 999 Danemark**
16	**110**	**F.poisson 72% Norvege**
17	**111**	**F.poisson 72% Islande**
18	**112**	**F.poisson 999 Danemark**
19	**113**	**F.poisson 999 Danemark beh B**
20	**114**	**F.poisson 999 Danemark beh A**
21	**116**	**Fischmehl 70/72**
22	**107**	**F.poisson 999 Danemark**
23	**115**	**F.poisson 999 Danemark**
24	35	Levure de bière

Table 8: First part of the ordered list for MinPts = 10

If we take MinPts=10, we can recognize nine gaps in the figure that represent nine clusters, as shown in Figure 11. Looking at the ordered list of points created by the OPTICS algorithm, we can find out that the first cluster found contains fish flour, as shown in Table 8, starting from position 8 to position 23 in the ordered list. After having recognized all the clusters, all the unclustered points are classified as noises, which can be analyzed exactly like we did at the end of Section 5.1.

Taking a smaller MinPts, like in Figure 10 a) or b), it becomes much more difficult to recognize the gaps and classify them correctly as clusters. For instance taking MinPts = 2, which is the value that we chose for DBSCAN, it is almost impossible to visually recognize and classify each cluster. But since we also have the ordered list with the names of the feed type for every amino acids profile, we could also sequentially look at the list and cut out the different clusters every time a new feed type appears. In any case, both solutions are not practicable for big datasets where a small MinPts is required, because this would take far too much time and we would not be sure to end up with a good result. A good use of OPTICS with a small MinPts is to use it as a complement to DBSCAN, i.e. to cluster points that have a different density and for this reason have not been clustered by DBSCAN.

Example:

After running DBSCAN we could run OPTICS using the noises of DBSCAN as the dataset, so that the clusters having different densities - which were not found by DBSCAN - can be found by OPTICS.

5.3 Comparison

The main advantage of DBSCAN over OPTICS is that DBSCAN automatically recognizes the clusters, while OPTICS needs manual work to build them, which can be more or less sustainable depending on the value of MinPts that has been chosen. When using OPTICS, the user does not need to find a good value for epsilon and the algorithm can also identify clusters very different in density. However, the experiments run with our dataset showed, that also with DBSCAN it is not so difficult to set a good value for epsilon and that the cases of clusters with very different densities is not big.

In conclusion, the use of DBSCAN is more comfortable when having very small clusters in the dataset, while OPTICS is good for finding bigger clusters.

6 Conclusions

Clustering amino acids profiles with DBSCAN and OPTICS can be very helpful to classify them according to their feed type. However, certain manual work and an accurate analysis of the outcome of the algorithms is always needed to achieve good results. Using DBSCAN, the most effort is put into finding a good value for epsilon, while using OPTICS, the most effort is put into visually deriving the clusters from the reachability-distances diagram. After testing both algorithms,

we noticed that the approach of DBSCAN makes the clustering of small clusters more comfortable and thus is a better solution for our case. In any case, the use of OPTICS remains helpful for further analyses of the results of DBSCAN. The tool we developed to visualize and analyze the results of the clustering algorithms, is still a prototype and probably needs some improvements, like an easy way to input new datasets.

This work also gave us the possibility to learn more about the characteristics of the data. As a first important result we can now say, that it is very rare to find misclassified or unclassified feed types (we did not find any). As a second result, we noticed, that most of the clusters are very small - about 80% of them contain less than ten points. Feed types generating only one amino acids profile are also contained in the dataset (about 13 on 393 profiles). The number of outliers is to consider as small (only 11), probably because the dataset was previously cleaned up from them. These are the results of the analysis of a single dataset containing 393 profiles. It would be interesting to run more experiments on other datasets and see if these results can be validated.

References

[1] http://en.wikipedia.org/wiki/DBSCAN (15.01.2012)

[2] http://en.wikipedia.org/wiki/OPTICS_algorithm (15.01.2012)

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander. OPTICS: ordering points to identify the clustering structure. In SIGMOD 1999. New York, USA. 49-60. (1999)

[4] M. Ester, H.-P. Kriegel et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In KDD 1996. Portland, USA. 226-231. (1996)

Appendix

Algorithm 1 DBSCAN(D, eps, MinPts)

input: a set of points D, a distance eps, the minimal number of points to build a cluster MinPts

```
1. Noises; //the list of noises
2. Clusters; //the list of clusters
3. for each not clustered point P in dataset D
4.     N = neighborhood(P,eps);
5.     if (N.size()<MinPts) then
6.         Noises.add(P);
7.     else
8.         C = newCluster();
9.         expandCluster(D, P, N, eps, MinPts, C, Clusters);
10. end for
```

Algorithm 2 expandCluster(D, P, N, eps,MinPts, C, Clusters)

input: a set of points D, a point P, the neighborhood N of P, a distance eps, the minimal number of points to build a cluster MinPts, a cluster C, a list of clusters Clusters

```
1. C.add(P)
2. for each point P2 in N
3.     N2 = neighborhood(P2,eps);
4.     if (N2.size()>=minPts) then
5.         N = mergeGroups(N,N2);
6.     if (P2 is not clustered) then
7.         C.add(P2);
8.         Noises.remove(P2); //in the case P2 was put in the noises list
9. end for
10. Clusters.add(C);
```

Algorithm 3 OPTICS(D, MinPts)

input: a set of points D, the minimal number of points to build a cluster MinPts

```
1. OrderedList; the ordered list of clusters//
2. eps = MAXINT; //a distance eps
3. for each unvisited point P in dataset D
4.     set P as visited;
5.     N = neighborhood(P,eps);
6.     OrderedList.add(P);
7.     Seeds = empty priority queue;
8.     if (coreDistance(N,P,eps,MinPts) != UNDEFINED) then
9.         update(N,P,Seeds,eps,MinPts);
10.        while (Seeds.size()>0)
11.            P2 = Seeds.remove(); //Remove the head of the queue
12.            set P2 as visited;
13.            N2 = neighborhood(P2,eps);
14.            OrderedList.add(P2);
15.            if (coreDistance(N,P,eps,MinPts) != UNDEFINED) then
16.                update(N2,P2,Seeds,eps,MinPts);
17. end for
```

Algorithm 4 update(N,P,Seeds,eps,MinPts)

input: the neighborhood N of P, a point P, a priority queue Seeds, a distance eps, the minimal number of points to build a cluster MinPts

```
1.  CoreDist = coreDistance(N, P, eps, MinPts);
2.  for each unvisited point O in N
3.      NewReachabilityDist = max(CoreDist, distance(O,P));
4.      if (O.reachabilityDist == UNDEFINED) then
5.          O.reachabilityDist = NewReachabilityDist;
6.          Seeds.add(O);
7.      else if (NewReachabilityDist < O.reachabilityDist) then
8.          O.reachabilityDist = NewReachabilityDist;
9.          Seeds.remove(O);
10.         Seeds.add(O);
11. end for
```