Clustering with Outliers in a Biased Animal Movement Database

Mei Li Go
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
m.l.go@student.utwente.nl

ABSTRACT

By putting accelerometers on horses, data was collected that can be classified into different behavioural patterns. Examples are "walking", "resting", "running-rider", and "running-natural". In order to study the behaviour of horses and investigate the structure of the data, the data has to be grouped according to these different behaviours. This research focused on grouping the horse data by means of clustering. Clustering is an unsupervised procedure to group data based on similarity. Unsupervised means that the clustering algorithm does not have classified (or labeled) data to train with. Finding the correct clustering algorithm is a challenge because the database is large, and the data is high-dimensional (21 dimensions). Furthermore, the data itself is biased. Most of the samples represent the horse trotting with a rider, and there are little samples available of other behaviour such as the horse shaking. Due to the small amount of samples for this behaviour, the algorithm could identify these samples as outliers (deviations from normal patterns) and remove them. In this research two algorithms were identified for clustering large and high-dimensional data: DBSCAN and OPTICS. The performance of the algorithms was evaluated using the V-measure. The algorithms were also assessed for biases towards clustering larger or smaller clusters as outliers, or clustering samples wrongly (False Negatives). After performing the tests, it was found that with the chosen parameter values, DBSCAN performed better. Although OPTICS had a far smaller percentage of False Negatives (21 percent per class on average compared to the 61 percent of DBSCAN), this could be explained by the high percentages of outliers that OPTICS had. DBSCAN was, in other words, better at identifying outliers. Furthermore, it had a higher V-measure (1 is the most desirable) with 0.512, whereas OPTICS had a V-measure value of 0.304. Further improvement of the performances can be achieved through extended parameter optimization.

Keywords

Clustering, large database, biased, outlier, animal activity recognition.

31th Twente Student Conference on IT Jul. 5th, 2019, Enschede, The Netherlands.

1. INTRODUCTION

In wildlife monitoring, it is hard to assigning sensory data to the corresponding activity. However, reliable animal activity recognition often require large amounts of categorized data to train with. For this research a horse database was used, of which the data was collected by attaching an accelerometer to the necks of 16 horses [9]. Similar to wildlife monitoring, each sample has to be assigned to the corresponding activity (or behaviour) before the data can be analysed. Examples of identified activities are "running", "running-rider", "walking", and "shaking". Manually assigning samples to classes is time consuming as a person would have to classify each sample individually. Clustering could offer a solution. Clustering is a process of grouping data items based on a measure of similarity [8]. Once the data has been grouped, a person can (manually) give names to the different groups, looking at only a few samples of each group. For the database this means that clustering provides a faster way of categorizing data, as well as enabling automatic data annotation. Clustering is also a good solution because it is an unsupervised process. Unsupervised means that it does not need to train with classified - or grouped - data [4], and therefore requires less resources. As this data set is imbalanced, clustering further helps the analysis by showing the balance of the data.

This paper focuses on strict partitioning clustering with outliers. Strict partitioning, also called hard clustering, means that a sample can only belong to one group [8]. This method is chosen as each horse activity should only be associated with one type of behaviour, for example either "running" or "walking". The data can also contain outliers. Outliers or anomalies are "significant deviations from behaviours that are normal patterns" [3]. These outliers should be removed from the data set, as they do not correctly represent the behaviour of the horse. The challenge of clustering this particular database is that small clusters which are not outliers, can be viewed as such. For example, out of 200 samples, only two can be the horse shaking and the remainder is the horse walking. It is important that the clustering algorithm does not identify these two samples of the horse shaking as outlier and therefore removes them. However, in the data set given no outliers were present. The classified data can nonetheless include outliers as it is clustered by humans. Thus the research focused on what the algorithms identified as outliers and if they favoured specific classes as outliers. Furthermore, the database used has 21 dimensions and almost 10,000 samples [9]. Therefore, the algorithms have to perform well on large and high-dimensional databases. To prevent further confusion the ground truth clusters, in other words the correct clusters, will henceforth be referred to as classes, and the results from the algorithms

shall be called clusters.

The research tried to try answer the following question:

Which clustering algorithm can best identify clusters in a biased data set while also identifying outliers in a data set with small groups of non-outliers?

In order to answer this question the following sub-questions were used for each algorithm:

1. *How many samples are clustered correctly?*
2. *For each class, what is the percentage of samples that is wrongfully clustered in another class (the percentage of False Negatives)?*
3. *Per class, what is the percentage of samples that is wrongly identified as outlier compared to:*
   (a) *The total amount of identified outliers?*
   (b) *To the class itself?*

This paper is structured as follows: Section 2 describes the literature research performed and the tools that were used. Section 3 explains which algorithms were chosen and why, after which the chosen evaluation methods will be described. The results are explained in Section 5. Possible improvements of the research are proposed in Section 6, followed by a conclusion.

2. METHODOLOGY

Different algorithms were compared by means of a literature review. The online Library of the University of Twente and Google Scholar were used to discover these algorithms. The algorithms that survey and review papers recommended were inspected further to find two well performing algorithms, that also had an online library in Python available. Two libraries were found to include code for both algorithms, which was desirable as using the same library for both algorithms would limit the likelihood of unreliable results due to different types of implementation. From the two libraries, the Scikit-learn library [15] was chosen as it was substantially faster and thus more efficient than the other library [13]. Furthermore, this library is well known and has been used over 60.000 times, reducing the possibility of mistakes in the code. After performing another literature research to find the fitting evaluation measurements for the clustering algorithms, it was discovered that both of these measurements were also available in the chosen library wherefore the library was used for this as well.

The algorithms were tested on a database containing 21 dimensions and 14 classes. Figure 1 is a visualization of the data using two of the 21 dimensions. In this, the 14 classes are indicated with different colours. Table 1 gives a visualization of all classes and the number of samples for each class. Furthermore, the database had almost 10,000 samples.

Each algorithm was tested multiple times on the database, with varying parameters in order to discover a result that would correspond to the ground truth the most and (therefore) had the highest scores on the evaluation measurements. For each combination of parameters, the following data was visualized in a table: the V-measure, the percentage of False Negatives, the percentage of outliers per class, the percentage of outliers compared to the total amount of outliers, and the distribution of the result clusters over the classes. In order to test the algorithms, the examples in the online library were copied and adjusted to cluster the horse data. Furthermore, additional code was written in order to make the data suitable for the coding of

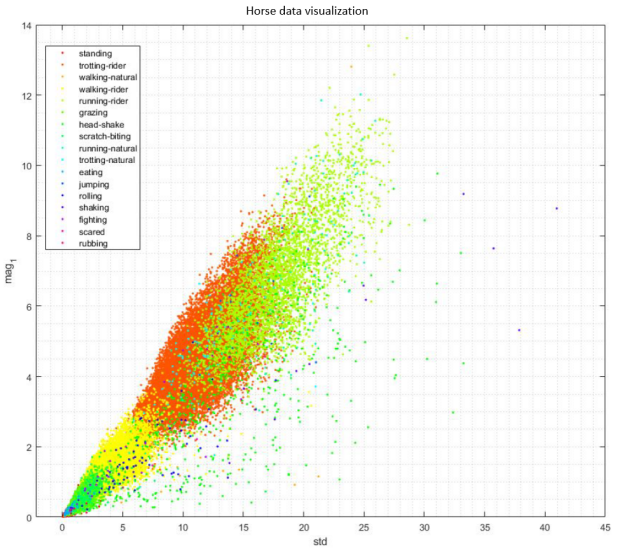


Figure 1. Visualization of the horse database based on two of the dimensions

the algorithm (the library required the data as an array). As only the V-measure was available in the online library, code was written to generate the tables and calculate the False Negatives and outlier percentages as well.

class	Number of samples
walking-rider	35425
trotting-rider	25688
grazing	18062
standing	5297
running-rider	3934
walking-natural	3609
head-shake	619
scratch-biting	285
running-natural	102
trotting-natural	94
rolling	67
eating	48
fighting	31
shaking	21

Table 1. Number of samples per class

3. ALGORITHMS

3.1 Background

To properly understand the choice of algorithm that will be made in this research, an understanding of clustering algorithms themselves is necessary before discussing specific algorithms. There are different types of algorithms based on the way they cluster. First there are hierarchical algorithms. Once a point is clustered according to such an algorithm, it is not re-evaluated and results in a tree-like structure. This can be done bottom-up or top-down. Bottom-up means that each sample starts as its own cluster and clusters are then merged, top-down means that all samples belong to the same cluster in the beginning and are then split into multiple clusters [10], [12]. Density-based cluster approaches group data based on high-density, in other words when many samples (how many depends on the parameters) are close together ('close' is also defined by a parameter). In a grid-based approach the algorithm distributes the samples into a fixed number

of cells, which has a form of a grid. Finally, subspace algorithms try to find all clusters in all available subspaces [8], [4], [14]. The algorithms that were considered for this research are explained below.

3.2 BIRCH

BIRCH, or Balanced Iterative Reducing and Clustering using Hierarchies, is an algorithm designed for large databases using the hierarchical approach. This means that once a point is clustered, it is not re-evaluated and results in a tree-like structure. Its' I/O (Input/Output) costs are linear and its' memory usage is efficient as the first step of the algorithm is to generate a summary of the data set. This summary reflects the natural closeness of the data. It reduces memory requirements and becomes scalable by using the Clustering Feature (CF) Tree. A CF tree is a height-balanced tree with two parameters, the branching factor B and the threshold T. A non-leaf node, in other words a node that has no "children", contains maximum B entries. A leaf node must satisfy the threshold requirement T. It's diameter or radius has to be less than T. If the tree is smaller, than the T is larger [19], [20].
The CF is defined as:

$$\mathbf{CF} = (n, LS, SS)$$

where n is the number of samples in a cluster, LS the linear sum of the n samples, and SS the square sum of the N samples. The time complexity of BIRCH is:

$$O(n)$$

where n again is the number of samples.

However, BIRCH does not scale well with high dimensions, and therefore cannot be used for this research. One of the successors of BIRCH, called CURE (Clustering Using REpresentatives) is less sensitive to outliers and more efficient than BIRCH [6]. However, although this algorithm performs well on both large as well as high dimensional databases, it does not do hard clustering. In other words, a sample can belong to multiple clusters. Therefore, it is not suited for this research.

3.3 CLIQUE

This grid- and density-based algorithm works bottom-up and with monotonicity [14], [1]. It looks at the density of a group of samples S in k-dimensional space. S can only be a cluster if it is also dense in every (k-1)-dimensional space. The algorithm starts by identifying subspaces that contain clusters by looking at the density of the samples, and then identifies the clusters. It also produces a minimal DNF (Disjunctive Normal Form) expression describing each cluster.

The algorithm is scalable, although the memory usage has to be managed carefully as a buffer of the database records is stored by the algorithm. This memory can be overrun by the candidates that the algorithm produces. The runtime of the algorithm is:

$$O(c^k + nk)$$

where c is the number of given dimensions (parameter value), k the number of clusters, and n the number of samples.

However, similar to CURE this algorithm does not do hard clustering, and was therefore not taken into further consideration.

3.4 DBSCAN

There are several requirements for a clustering algorithm according to [5]: minimal knowledge of the data domain to determine the algorithm's input values, discovery of clusters with an arbitrary shape, and efficiency on large databases. Based on these requirements, the density-based approach DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is build. The algorithm has two parameters:

- **Eps**: the minimum distance between two samples. If the distance between samples is lower than this value, the two samples are neighbours;
- **MinPts**: the minimum number of samples that needs to be within the distance Eps of a single sample in order for that group of samples to become a cluster.

The border samples do not fulfil the MinPts requirement, but are in the neighbourhood of a core samples. Thus they will be in a cluster. This also means that DBSCAN detects outliers, because those are not in the neighbourhood of a core samples [18]. Furthermore, DBSCAN is prone to errors when clusters are not well separated due to the algorithm clustering based on density. Another disadvantage is the high run time, namely:

$$O(n * log(n))$$

where n is the number of samples.

3.5 OPTICS

The OPTICS (Ordering Points To Identify the Clustering Structure) algorithm can be viewed as an extension of DBSCAN [2]. It uses the same parameters and principle of the algorithm, only here the Eps parameter is optional and mainly used as a way to reduce computational complexity (if no value is specified, Eps = infinity). OPTICS makes an ordering of the database, which is then used to cluster, and also produces the core distance and reachability-distance for each sample.

The algorithm starts with the samples that have the smallest distances to each other, and hierarchically works up from there. It uses the MinPts parameter to both define the neighbourhoods, as well as to calculate the distances for each sample. Furthermore, OPTICS has the same run time as DBSCAN.

3.6 K-means

The k-means clustering algorithm is computationally attractive and simple to implement, and therefore also a popular clustering algorithm. It starts by assigning random coordinates as centroid points. The amount of centroid points is given by a parameter defining the number of clusters. The data space is also divided by the same parameter. The algorithm then looks at where the centre (middle) of each cluster is, and moves the centroid point to this position. These two steps are repeated until the found centroid points cannot be improved further. During this process, the clusters are also adjusted. Because the centroid points change their location in the data space, samples may be closer to another centroid point than before the change. Thus these samples will then belong to the cluster of the other centroid point.

This approach is very sensitive to outliers, because it does not identify them. As the centroid point is determined by the location of the middle (or average) of the samples, the outliers influence where the centroid point is [8].

The time complexity of the algorithm is:

$$O(nkd)$$

where n is the number of samples, k the number of clusters, and d the time it takes to compute the distance between two points [14].

3.7 Chosen algorithms

Based on the recommendations of the survey papers [14], [8], and [10], and the availability of libraries for the algorithms, DBSCAN and OPTICS were chosen. K-means was not used as it is very sensitive to outliers as it does not detect nor remove them. CLIQUE was designed for finding accurate clusters in large and high dimensional databases [1], but does not do hard clustering and does therefore not conform with the requirements of the research. The same is the case for CURE. DBSCAN has a high, but still feasible run time. It can also handle large databases as well as high dimensionalities, and detects outliers, which is important for the given data set. OPTICS is DBSCAN's extension, also detects outliers, and makes an ordering of the data set from which the data could possibly be clustered better than with DBSCAN. Therefore this algorithm was also chosen. Other algorithms that were considered as they were recommended by most survey papers were PROCLUS, MAFIA, and ENCLUS. However, due to time limitations they were not further considered.

4. EVALUATION METHODS

There are three types of performance metrics possible to evaluate algorithms with:

a. *Internal measures*: these evaluate the structure without the use of external knowledge [11]. No knowledge of the ground truth (the reality, real groups to which the samples belong) is necessary.
b. *External measures*: uses external knowledge such as the ground truth to validate the clustering algorithm.
c. *Relative measures*: comparing the result of a clustering algorithm with results of other clustering algorithms.

Using internal measures is difficult as an understanding of the structure of the data is needed in order to evaluate the cluster compactness and separation. The criterion measure also has to be derived from the data itself [14]. Due these difficulties, the use of internal measures did not fit within the time scope of this research. Because there is already classified data available, the algorithms in this research will be evaluated using external measures. Relative measures were also used as the algorithms will be compared with each other.

The following external measures were used:

a. **F-measure**: this is a popular measure which is calculated by precision (P) and recall (R), and of which the ideal result is 1. The formula is as follows [17], [14]:

$$F = \frac{2PR}{P+R}$$

In order to calculate the precision and recall True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) are used. Precision evaluates which percentage of the cluster is correct, and recall validates which percentage of samples that should be in a certain class actually is in it. They are calculated as follows:

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

The disadvantage of the F-measure is that it does not take outliers into account, and for the calculation needs every class to have clusters assigned to it.

b. **V-measure**: this evaluates the algorithm using homogeneity and completeness [16]. Homogeneity means that a cluster only contains sample members of a single class. A data set satisfies completeness when all samples of a class are in a single class. Increasing one often decreases the other, as a guaranteed maximum homogeneity is reached when each sample is assigned to their own cluster, whereas completeness is maximized when all samples are assigned to a single class. The perfect V-measure value is 1. The V-measure is calculated as follows:

$$V_\beta = \frac{(1+\beta) * h * c}{(\beta * h) + c}$$

where h is homogeneity and c is completeness. The calculations of h and c can be found in [16]. If β is smaller then one, homogeneity is weighted more strongly, if it is greater then one then completeness is weighted more strongly. For this research, the value of β was one.

The V-measure is used next to the F-measure because of several reasons. The first reason is that, before the F-measure is executed, a post-processing step in which each cluster is assigned to a class has to be done. This means that the measure also calculates the goodness of the cluster-class matching, and therefore an identical post-processing algorithm has to be used for each test. Secondly, the F-measure has a "problem of matching". In calculating the similarity between a clustering result and the ground truth, the F-measure only considers the contributions from those clusters that are matched to a target class. This is a problem, as two significantly different clustering outcomes can result in identical scores [7]. The F-measure does also not take into account outliers. Furthermore, the V-measure evaluates the structure of the clusters, whereas the F-measure evaluates the correctness of the sample assignment (in other words, which samples are in the correct cluster).

To compare the algorithms with each other, the following the relative measurements were used:

a. For each class, what is the percentage of samples that is wrongfully clustered in another class (the percentage of False Negatives (FN%))?

$$FN\% = \frac{FN}{total\ samples\ of\ class}$$

b. Per class, what is the percentage of samples that is wrongly identified as outlier compared to:

(a) The total amount of identified outliers (Outlier Percentage of Class from Total amount of Outlier (**OPCTO**))?

$$OPCTO = \frac{identified\ outliers\ of\ class}{total\ amount\ of\ identified\ outliers}$$

(b) To the class itself?

$$Outlier\,\% = \frac{identified\ outliers\ of\ class}{total\ samples\ of\ class}$$

4.1 Parameter selection

To select the values of MinPts that would be used for testing, the number of samples of each class were measured (Table 1). Since the results should also include the smallest cluster, the four lowest values were chosen (21, 31, 48, and 67). In addition, two values below 21 - 10 and 15 - were chosen to asses if this would benefit the algorithm in identifying the smaller clusters.

In order to determine values for the Eps parameter, the distances between the centres (the average of all the samples of a class) of the classes were calculated. This was done using the L2 matrix, of which the results are shown in Appendix A. The L2 matrix, also called the Euclidean matrix, calculates the straight-line distance between two points (in this case the centres). The calculated distances were then processed into a histogram (Figure 2).

Most distances were within the range of 4,690. The algorithm should be able to identify all clusters, and therefore the distances chosen were near and including the smallest distance calculated. Based on the histogram and the distribution of the distances, the following values for Eps were chosen: 15, 25, 50, 200, 1190, and 1400. Again, a value smaller than the minimum distance was chosen to measure if this would be beneficial in identifying the smaller clusters.

Because both algorithms require the same parameters, both were tested with the same parameter values.

4.2 Identifying cluster classes

Before being able to evaluate the results of the algorithms, for each resulting cluster the class - or ground trurth cluster - it belonged to had to be identified. In order to do so, the absolute number of samples per class was evaluated for each resulting cluster. For instance, "walking-naturally" has 70 samples and "running-rider" has 10 samples in the cluster. The class with the highest number of samples was chosen as the class of the resulting cluster, in the example this would be "walking-naturally".

5. EVALUATION RESULTS

5.1 V-measure

For DBSCAN, the highest score achieved was 0.512 (Appendix C.1). This was done with an Eps value of 25 and an MinPts value of 15. In general, DBSCAN achieved higher scores with combinations of low Eps values (below 200). OPTICS' highest value was 0.304 (Appendix C.2). Interestingly, its' highest values were with parameters with which DBSCAN reached its' lowest V-measure values, namely with a high Eps value and low MinPts value. In other words, for DBSCAN the V-measure increased the lower the Eps value and the higher the MinPts value, whereas with OPTICS the V-measure then decreased.

Appendices C.3 and C.4 offer further insight into the calculation of the V-measure. Next to the V-measure, homogeneity and completeness are also displayed in the graph. As for both graphs completeness consistently has a higher value than homogeneity, it is clear that both algorithms had a tendency towards clustering according to completeness. This means that the algorithms were more prone to cluster all samples into a single cluster, which Table 2 confirms. The table shows how many clusters were assigned to each class, excluding the classes that did not get any clusters assigned to them (as the number of clusters assigned is zero for these classes). Table 2 confirms the tendency towards completeness as, from the 14 classes, only two classes for DBSCAN and five classes for OPTICS had clusters assigned to them. DBSCAN also had a higher tendency to cluster according to completeness. This higher value for completeness is one of the reasons why the V-measure of DBSCAN has a higher value. The higher V-measure can also be explained by the fact that DBSCAN generated substantially less clusters than OPTICS (Table 2), making the results correspond more to the ground truth. Another explanation of the V-measure values can be given when looked at the percentage of samples that are correctly clustered (Appendix D). This was done by adding up the percentages of outliers and False Negatives, and subtracting this number from 100 (as the total of each class is 100 percent). All classes that have clusters assigned to them were above 85 percent correctly clustered for DBSCAN, whereas for OPTICS the maximum was 62 percent. Finally, the V-measure can be explained when looked at the number of outliers, which will be discussed later (in Section 5.4). To conclude, with a V-measure value of 0.208 higher than OPTICS, DBSCAN clustered more correctly according to the V-measure.

Class	DBSCAN	OPTICS
Trotting-rider	87	1022
Walking-rider	1	287
Running-rider	0	139
Grazing	0	57
Standing	0	25

Table 2. Number of clusters assigned to each class

For further discussion of the results, only the results of generated by the parameter values with the highest V-measure shall be discussed as these were the best results. For OPTICS, these are two parameter combinations (Eps = 1190 and Eps 1400 both combined with MinPts = 15). However, the results measured where identical and therefore only the results with parameters Eps = 1400 and MinPts = 15 will be discussed.

5.2 F-measure

In order to use the F-measure, there always has to be a value higher than zero for True Positives, False Positives, and False Negatives (otherwise either the precision or recall formula will have a division by zero). However, in every result of both algorithms, there were classes with no True Positives and no False Negatives (all samples had been identified as outliers or classes had no clusters assigned to them as is clear from Table 2). Therefore, the resulting F-measures were not reliable and thus not used to evaluate the algorithms with.

5.3 Percentage of False Negatives

In this area, OPTICS outperformed DBSCAN by a large margin. The average percentage of False Negatives equaled 21.25 percent for OPTICS, whereas for DBSCAN it was 63.71 percent (Appendix D). When looking at the percentages of False Negatives per class, OPTICS had low percentages for all classes where DBSCAN had a percentage above 99. All classes that had this high percentage for DBSCAN, are dense. They had a standard deviation far lower than the average, as well as their minimum and maximum distance between two samples being significantly below the average (Appendix B). DBSCAN, in other words, was not able to correctly identify dense classes. All classes that did have clusters assigned to them had low percentages of False Negatives (for both algorithms). The limited number of classes that had clusters assigned to them also explains the high(er) percentages of those classes that had no clusters assigned to them. A reason that larger classes such as "walking-natural" (with 3,609 samples) had a high percentage of outliers for DBSCAN, can be that most of those samples have been assigned to the "walking-rider" class. The distance between these two classes is also far below the average (Appendix A). In conclusion, OPTICS has less False Negatives than DBSCAN, which is supported by the distribution of clusters over the classes (Table 2).

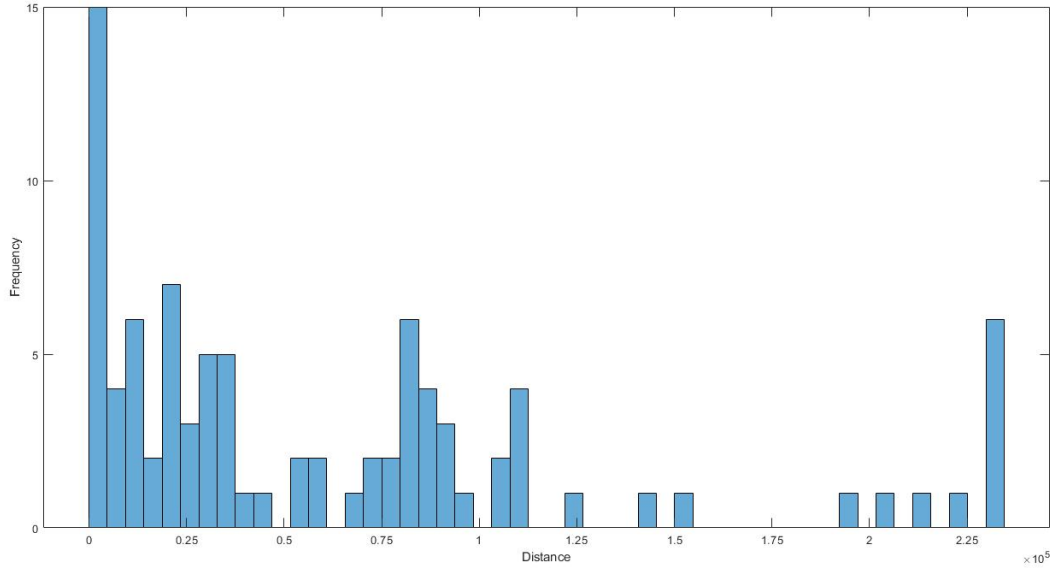


Figure 2. Distance between classes calculated with the L2 matrix

5.4 Outlier percentages

5.4.1 Percentages per class

Where a class had a high or low percentage of False Negatives, it had the opposite value for the percentage of outliers (Appendix F), making the two percentages together 100 percent. The only exceptions where the classes that did have clustered assigned to them. As the rest of the classes did not have any clusters assigned to them, their samples were either False Negatives or outliers. This resulted in outlier percentages higher than 30 for every class with the OPTICS algorithm. DBSCAN had better percentages, at highest reaching 82 percent. Since DBSCAN had high percentages of False Negatives, generally the outlier percentages were low (below 15 percent).

When comparing the outlier percentages with the percentage of False Negatives, it can be further explained why DBSCAN had a high percentage of False Negatives for dense classes. The algorithm did cluster the samples, but in the wrong class as only two classes have clusters assigned to them. OPTICS, on the other hand, did not recognise the dense classes at all, and identified all their samples as outliers. Furthermore, where DBSCAN had low percentages of both False Negatives and outliers for the classes that do have clusters assigned to them, this is not the case for OPTICS. In OPTICS the class "walking-rider", for example, has 287 clusters assigned to it, but still an outlier percentage of 86 percent.

Given this information and with an average of 23 percent outliers per class, DBSCAN is better at identifying outliers then OPTICS, which had an average of 69 percent.

5.4.2 OPCTO

For both algorithms, the larger classes made up most outliers (Appendix G). This is due to the fact that these classes have more samples. So even though the percentage of identified outliers may be lower than with another, smaller, class, the number of samples can still be higher. For example for OPTICS, "running-rider" has an outlier percentage of 30 and "eating" a percentage of 95. From the total amount of outliers, "running-rider" takes 1.76 percent and "eating" 0.07. This is due to the fact that "running rider" has significantly more samples than "eating". However, based on these results, it can be concluded that both algorithms still cluster many samples of classes that do have clusters assigned to them as outlier.

In this area, both algorithms performed equally, and both had the highest percentages of outliers with the classes that had the biggest number of samples.

6. DISCUSSION

The fact that every class had outliers as well as a high amount of clusters identified with both algorithms, can be due to the data being gathered from 16 horses. These horses may have different walking, running, and trotting patterns to mention a few examples. Therefore, it is possible that each horse requires their own classes as their patterns for the same behaviour may differ significantly. This would mean that instead of 14 there would be 224 (14 times 16) classes. Another cause that could have had influenced the number of outliers can be that the categorization - which is done by persons - missed classes that the algorithms did identify. Since the ground truth has been determined by a person, human error could also have increased the number False Negatives as well as the number of outliers. Finally, the shape of the classes can be a cause for the algorithms to be inaccurate. For example, if a small class has a large distance to other classes, the algorithm could identify its' samples more easily as outliers. The distances within the cluster itself could also be a cause for identification as outlier or assignment to or with a different class. Moreover, samples of a class could have been in the same area as another class. An example of this would be with "walking-rider" and "walking-naturally".

A reason for the high number of False Negatives, can be the manner in which the clusters were assigned to the classes. The classes were assigned based on which class had the highest amount of samples in the cluster. This could have resulted in a cluster containing all samples of a class, and still be classified differently due to another class

having a higher number of samples in the cluster. This approach was still chosen, however, as the algorithm should be hard-clustering, in other words a sample can only belong to one cluster. Furthermore, the algorithms cluster based on similarity. Therefore, it is likely that the centre of the resulting cluster is in or near the class with the highest absolute number of samples. A possibility for future research, however, would be not to use hard clustering. In such a case, a sample can belong to multiple clusters or can have different percentages of similarity for each class, adding up to 100 percent total per sample. Which of these two approaches is chosen depends on the algorithm.

A possible way to improve the performance of the algorithms of this research, would be reducing the number of dimensionalities. With high dimensional databases there is a "curse of dimensionality", which means that as the dimensionality increases, the distances between samples become more similar. This makes it harder to cluster the database with distance measures [14]. Therefore, dimensionality reduction could improve the clustering of the algorithm as distances are less similar. However, valuable information could be lost when decreasing the number of dimensions, making the clustering inaccurate. Therefore, the dimensions that will be removed (or used) have to be selected carefully.

It was also clear from the V-measure, that OPTICS started to perform better with a higher value for the Eps parameter. Therefore, it can be explored how well OPTICS performs with higher Eps values. DBSCAN, performed best with the combination of Eps = 25 and MinPts = 15, and thus experiments with more similar values can be performed to improve the performance. In addition, other algorithms could be tested. This research only analyzed the performance of two algorithms due to time constraints.

7. CONCLUSION

This aim of this research was to determine a clustering algorithm that can correctly identify clusters while simultaneously determining outliers. The data set was challenging since it was biased and contained small groups of non-outliers. DBSCAN has produced the best results as it had a smaller number of outliers and a higher V-measure value. However, both algorithms did not manage to identify the correct number of classes nor all the classes themselves (the ground truth). OPTICS shows more potential with higher Eps values then used in this research, whereas with DBSCAN there should be experiments with different small parameter values. Both of the algorithms produced high numbers of outliers or had large numbers of False Negatives. Mainly small classes and similar classes (such as "walking-naturally" and "walking-rider") were sensitive to being clustered incorrectly. Neither of the algorithms showed a biased towards a class in identifying outliers.

The results of this research show that clustering with outliers in a practical data set remains challenging. DBSCAN performed better in this research, however OPTICS shows potential with higher Eps values. Better results for both algorithms can be explored by choosing different values for the parameters as suggested above, and different algorithms can also be tested as this research only analysed two clustering algorithms.

8. REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 94–105, New York, NY, USA, 1998. ACM. event-place: Seattle, Washington, USA.

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points to Identify the Clustering Structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 49–60, New York, NY, USA, 1999. ACM. event-place: Philadelphia, Pennsylvania, USA.

[3] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin. A fast and noise resilient cluster-based anomaly detection. *Pattern Analysis and Applications*, 20(1):183–199, Feb. 2017.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, Inc., Canada, 2 edition, 2001.

[5] M. Ester, H.-P. Kriegel, X. Xu, and J. Sander. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Vol. 96:6, 1996.

[6] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. *Information Systems*, 26(1):35–58, Mar. 2001.

[7] G. Hripcsak and A. S. Rothschild. Agreement, the F-Measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, May 2005.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Comput. Surv.*, 31(3):264–323, Sept. 1999.

[9] J. W. Kamminga. Dataset: IMU Movement Data of Horses. June 2019.

[10] H.-P. Kriegel, P. KrÃűger, and A. Zimek. Clustering High-dimensional Data: A Survey on Subspace Clustering, Pattern-based Clustering, and Correlation Clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1:1–1:58, Mar. 2009.

[11] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of Internal Clustering Validation Measures. In *2010 IEEE International Conference on Data Mining*, pages 911–916, Dec. 2010.

[12] M. Mittal, L. M. Goyal, D. J. Hemanth, and J. K. Sethi. Clustering approaches for high-dimensional databases: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1300, 2019.

[13] A. Novikov. PyClustering: Data Mining Library. *Journal of Open Source Software*, 4(36):1230, Apr. 2019.

[14] D. Pandove, S. Goel, and R. Rani. Systematic Review of Clustering High-Dimensional and Large Datasets. *ACM Trans. Knowl. Discov. Data*, 12(2):16:1–16:68, Jan. 2018.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] A. Rosenberg and J. Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 410–420, Prague, June

2007.

[17] Y. Sasaki. The truth of the F-measure. page 5, Oct. 2007.

[18] E. Schubert, J. Sander, M. Ester, H.-P. Kriegel, and Xiaowei Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, 42(3):1–21, July 2017.

[19] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. ACM. event-place: Montreal, Quebec, Canada.

[20] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, June 1997.

APPENDIX

A. L2 Matrix

	standing	trotting-rider	walking-natu	walking-rider	running-rider	grazing	head-shake	scratch-biting	running-natu	trotting-nat	eating	rolling	shaking	fighting
standing	0.00	29,800.53	1,456.84	1,982.67	89,269.54	25.46	82,826.48	1,223.74	109,231.62	37,506.35	30.06	11,201.36	234,412.18	22,920.55
trotting-rider	29,800.53	0.00	28,343.76	27,817.89	59,469.02	29,822.31	53,025.96	28,576.87	79,431.10	7,705.82	29,772.73	18,599.18	204,611.66	6,879.98
walking-natural	1,456.84	28,343.76	0.00	525.87	87,812.78	1,478.55	81,369.72	233.14	107,774.86	36,049.58	1,428.97	9,744.59	232,955.42	21,463.78
walking-rider	1,982.67	27,817.89	525.87	0.00	87,286.91	2,004.42	80,843.85	758.98	107,248.99	35,523.72	1,954.83	9,218.72	232,429.55	20,937.91
running-rider	89,269.54	59,469.02	87,812.78	87,286.91	0.00	89,291.33	6,443.07	88,045.88	19,962.08	51,763.19	89,241.74	78,068.19	145,142.64	66,349.00
grazing	25.46	29,822.31	1,478.55	2,004.42	89,291.33	0.00	82,848.27	1,245.45	109,253.41	37,528.13	49.63	11,223.14	234,433.97	22,942.33
head-shake	82,826.48	53,025.96	81,369.72	80,843.85	6,443.07	82,848.27	0.00	81,602.82	26,405.14	45,320.14	82,798.69	71,625.14	151,585.70	59,905.94
scratch-biting	1,223.74	28,576.87	233.14	758.98	88,045.88	1,245.45	81,602.82	0.00	108,007.96	36,282.69	1,195.87	9,977.69	233,188.52	21,696.89
running-natural	109,231.62	79,431.10	107,774.86	107,248.99	19,962.08	109,253.41	26,405.14	108,007.96	0.00	71,725.27	109,203.82	98,030.27	125,180.56	86,311.08
trotting-natural	37,506.35	7,705.82	36,049.58	35,523.72	51,763.19	37,528.13	45,320.14	36,282.69	71,725.27	0.00	37,478.55	26,305.00	196,905.84	14,585.80
eating	30.06	29,772.73	1,428.97	1,954.83	89,241.74	49.63	82,798.69	1,195.87	109,203.82	37,478.55	0.00	11,173.55	234,384.38	22,892.75
rolling	11,201.36	18,599.18	9,744.59	9,218.72	78,068.19	11,223.14	71,625.14	9,977.69	98,030.27	26,305.00	11,173.55	0.00	223,210.84	11,719.20
shaking	234,412.18	204,611.66	232,955.42	232,429.55	145,142.64	234,433.97	151,585.70	233,188.52	125,180.56	196,905.84	234,384.38	223,210.84	0.00	211,491.64
fighting	22,920.55	6,879.98	21,463.78	20,937.91	66,349.00	22,942.33	59,905.94	21,696.89	86,311.08	14,585.80	22,892.75	11,719.20	211,491.64	0.00
Min	25.46	6,879.98	233.14	525.87	6,443.07	25.46	6,443.07	233.14	19,962.08	7,705.82	30.06	9,218.72	125,180.56	6,879.98
Max	234,412.18	204,611.66	232,955.42	232,429.55	145,142.64	234,433.97	151,585.70	233,188.52	125,180.56	196,905.84	234,384.38	223,210.84	234,433.97	211,491.64
Mean	47,837.49	46,450.52	46,972.14	46,810.33	73,703.49	47,857.41	69,738.53	47,079.73	89,058.94	48,821.55	47,815.81	45,392.07	204,610.22	43,628.95
Overall average														
Min	13,556.17													
Max	206,740.49													
Mean	64,698.37													

B. Density of clusters

	minimum	maximum	mean	std	var
standing	74.70	271.82	77.10	38.31	1,467.32
trotting-rider	6,354.51	24,418.56	1,496.29	3,470.75	12046087.25
walking-natural	330.22	1,525.94	145.85	1,916.93	3,674,601.80
walking-rider	457.00	2,179.98	171.05	524.20	274,788.94
running-rider	16,347.19	62,875.82	4,328.59	8,791.57	77,291,783.04
grazing	97.39	371.75	75.52	73.68	5,428.14
head-shake	26,493.58	131,312.54	4,021.84	19,711.47	38,8542,123.27
scratch-biting	231.41	1,451.26	135.00	819.43	671,468.11
running-natural	26,970.18	96,710.51	5,279.30	12,732.22	162,109,454.73
trotting-natural	6,554.40	34,461.11	1,863.31	5,373.33	28,872,635.43
eating	53.86	277.99	77.85	131.17	17,206.36
rolling	2,749.46	17,158.07	611.09	2,968.86	8,814,108.22
shaking	34,837.91	551,088.55	11,240.24	83,025.91	6,893,302,162.38
fighting	3,499.30	42,379.71	1,168.39	7,407.31	54,868,184.14
Average	8,932.22	69,034.54	2,192.24	10,498.94	545,035,107.08

C. V-measure

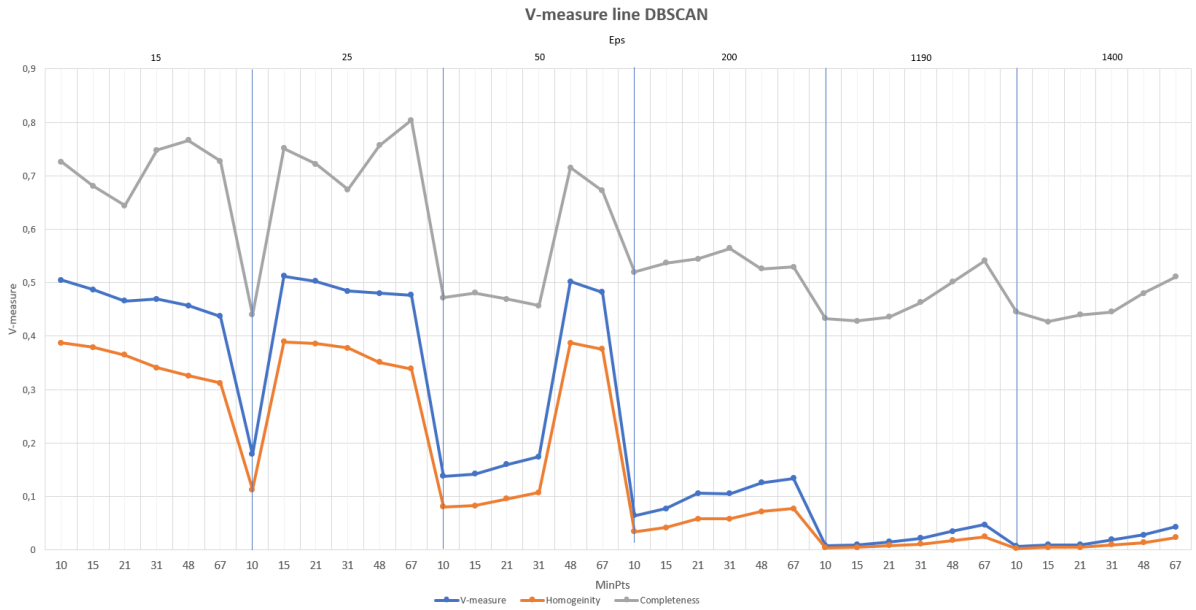
1. DBSCAN

	MinPts					
		10	**15**	**21**	**31**	**48**	**67**
Eps	**15**	0.505	0.487	0.466	0.469	0.457	0.437
	25	0.179	0.512	0.503	0.484	0.48	0.477
	50	0.138	0.142	0.16	0.174	0.502	0.482
	200	0.064	0.077	0.106	0.105	0.126	0.134
	1190	0.008	0.01	0.015	0.022	0.035	0.047
	1400	0.007	0.01	0.01	0.019	0.028	0.043

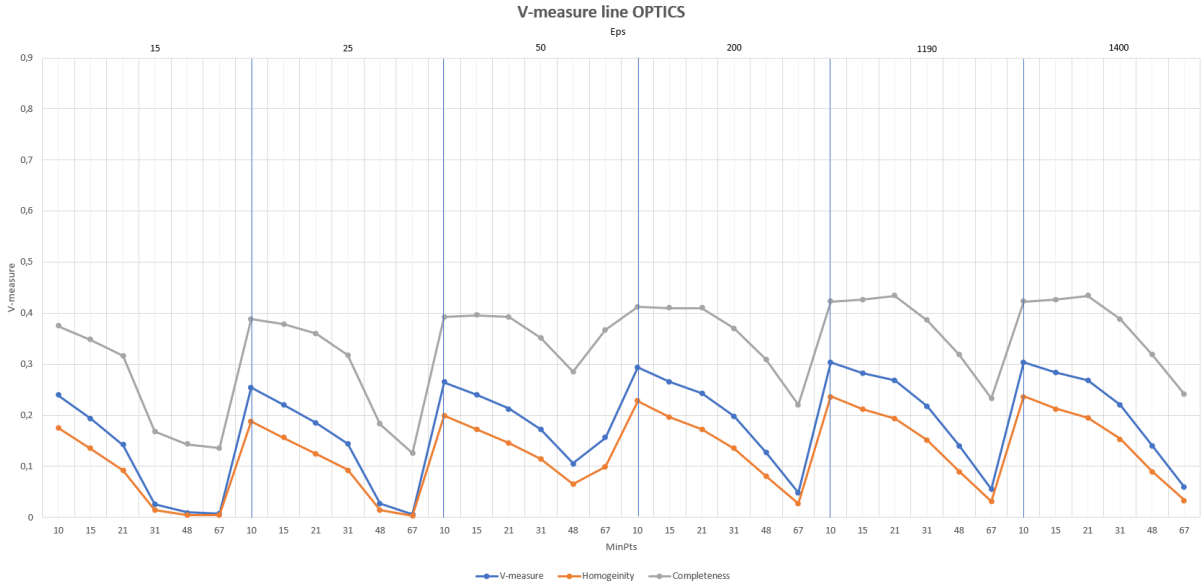
2. OPTICS

	MinPts					
		10	**15**	**21**	**31**	**48**	**67**
Eps	**15**	0.239	0.194	0.142	0.026	0.01	0.008
	25	0.254	0.221	0.185	0.144	0.027	0.006
	50	0.265	0.24	0.213	0.173	0.106	0.156
	200	0.294	0.266	0.243	0.198	0.128	0.048
	1190	0.304	0.283	0.269	0.218	0.141	0.056
	1400	0.304	0.284	0.269	0.221	0.141	0.059

3. DBSCAN V-measure graph



4. OPTICS V-measure graph



D. Percentage clustered correctly

Class	DBSCAN	OPTICS
standing	0	7.47
trotting-rider	87.36	61.9
walking-rider	98.08	12.73
running-rider	0	43.69
grazing	0	4.33

E. False Negatives

1. DBSCAN

Eps	MinPts					
	10	**15**	**21**	**31**	**48**	**67**
15	61.21	57.58	53.19	44.63	43.32	42.5
25	71.19	63.71	61.59	57.59	45.78	44.46
50	75.55	74.25	71.99	69.4	60.87	57.17
200	83.99	81.49	78.77	76.77	75.04	73.11
1190	89.57	89.26	88.44	87.3	85.27	82.96
1400	89.6	89.3	88.93	87.82	86.52	84.23

2. OPTICS

Eps	MinPts					
	10	**15**	**21**	**31**	**48**	**67**
15	11.02	8.94	5.62	1.43	0.6	0.4
25	12.19	10.09	9.4	8.77	1.01	0.4
50	13.86	13.03	11.94	10.16	6.09	8.58
200	18.59	16.48	15.72	12.44	7.61	3.5
1190	21.25	19.73	20.16	15.16	9.47	3.55
1400	21.25	20.02	20.39	15.7	9.27	4.53

3. False Negatives for best V-measures

Class	DBSCAN	OPTICS
standing	99.64	0.4
trotting-rider	2.87	2.06
walking-natural	99.75	7.82
walking-rider	1.59	0.99
running-rider	17.82	26.26
grazing	99.82	0.32
head-shake	46.79	39.76
scratch-biting	99.65	2.45
running-natural	20.58	62.72
trotting-natural	77.64	60.42
eating	100	4.16
rolling	91.05	17.88
shaking	47.61	23.8
fighting	87.11	48.45

F. Outlier percentages per cluster

Class	DBSCAN	OPTICS
standing	0.36	92.13
trotting-rider	9.77	36.04
walking-natural	0.25	92.57
walking-rider	0.33	86.28
running-rider	82.28	30.05
grazing	0.18	95.35
head-shake	53.15	59.94
scratch-biting	0.35	97.54
running-natural	79.41	37.25
trotting-natural	22.34	39.36
eating	0	95.83
rolling	8.96	82.09
shaking	52.38	76.19
fighting	12.9	51.61
Average	23.05	69.45

G. OPCTO

Class	DBSCAN	OPTICS
standing	0.3	7.25
trotting-rider	39.35	13.75
walking-natural	0.14	4.96
walking-rider	1.82	45.41
running-rider	50.77	1.76
grazing	0.52	25.59
head-shake	5.16	0.55
scratch-biting	0.02	0.41
running-natural	1.27	0.06
trotting-natural	0.33	0.05
eating	0	0.07
rolling	0.09	0.08
shaking	0.17	0.02
fighting	0.06	0.02