Course Work 1: DBScan++

Unsupervised Learning

Nikita Belooussov

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Barcelona, May 16, 2022

Contents

1 Description **1**

2 Method **1**
2.1 Clustering Procedures 1
2.1.1 DBSCAN++ 1
2.1.2 OPTICS 3
2.2 Datasets 4
2.2.1 Blobs and Moons 4
2.2.2 Iris Dataset 4
2.2.3 Pima Diabetes Dataset 5
2.2.4 Acoustic Extinguisher Fire Dataset (Fire Dataset) 5
2.3 Measurements 5
2.3.1 Adjusted Rand Score (ARS) 5
2.3.2 Calinski Harabasz Score (CHS) 6
2.3.3 Fowlkes Mallows Score (FMS) 6

3 Results **6**
3.1 Blobs 7
3.2 Moons 7
3.3 Iris 8
3.4 Pima Diabetes 9
3.5 Acoustic Extinguisher Fire Dataset 9

4 Conclusion **10**

1 Description

The purpose of this report is to go over the density based clustering procedure known as DBSCAN++ [5] and try to reproduce it. The report will first provide a description of how the procedure works, this will include a pseudo code description of the algorithm developed. The report will then go onto comparing the developed code to other commonly used clustering techniques. This is to measure how well the developed script performs to more professionally made ones. In order to compare the different techniques there will be four things that will be looked at. The first is the average time it took to go through the datasets for each protocol. The next will be adjusted rand score, fowlkes mallows score, and lastly the calinski harabasz score. A more in depth description of these measurements and the datasets used will be provided further in the report. In the end a model that is able to dp clustering on the data was produced.

2 Method

In this section of the report a lot of the objects used will be discussed. It will begin with the clustering procedures used in the report. The report mainly focuses on the DBSCAN++ procedure, but will also mention the OPTICS that was used as well. The report also uses normal DBSCAN but that will be covered in the DBSCAN++ section. After that the datasets that were analyzed will be discussed. Three of the datasets were ones that are based off of real data. They were chosen due to their sizes and the number of attributes that they have. As well as the experience of using them before. The other two were artificially generated samples that follow a certain pattern. Lastly, the measurements used will be discussed. As mentioned there were four total measurements taken. The time one won't be discussed since it should be naturally understandable what is happening through the results. The report used two external scores and one internal one. The external scores were the adjusted rand score and the Fowlkes Mallows score. The internal score was the Calinski Harabasz score.

2.1 Clustering Procedures

As mentioned before there were three clustering procedures used. The DBSCAN procedure will be discussed in the DBSCAN++ section found in Section 2.1.1. These all were chosen due to them all being fairly similar to each other in concept. Especially between DBSCAN and DBSCAN++. Since DBSCAN++ was an improvement to DBSCAN it would seem essential to compare it to the predecessor to see if any improvements have been made. The OPTICS procedure also uses the ideas in the DBSCAN in order to cluster, which is why it felt like it would be a good comparison. As mentioned before the report is mainly about DBSCAN++, as a result the OPTICS explanation will not be as thorough but a better explantion can be found in the original paper[2].

2.1.1 DBSCAN++

The DBSCAN++ is the main algorithm being studied in this report. As the name implies it is an improvement upon the DBSCAN procedure[3]. As stated in the Gan and Tao paper[4], the runtime of the DBSCAN procedure is $O(n^2)$ in certain scenarios. The DBSCAN++ is a procedure that is an attempt to help minimize the runtime of the procedure. The proposal to do this is by limiting the number of core points that are tested. By doing this, it is hoped that the runtime improves in poor conditions and is still able to perform at the same level as if this was not implemented.

Since the DBSCAN++ is an improvement on the DBSCAN procedure, it is important to understand how the DBSCAN procedure works. The main concept behind the DBSCAN procedure is that the data instances can be either core points, border points, or noise. This categorization is accomplished by seeing how many points are within a given area. As a result, besidesthe dataset, the DBSCAN procedure requires the number of points required to be inside an area to be considered a core point, and the radius that they need to fall into. The procedure then goes through the points in the dataset trying to see which points are which, with the points falling within the given distance being categorized as being in the same cluster. In Figure 1, the pseudo code of the DBSCAN procedure can be seen.

```
def DBSCAN(dataset,radius,minPts):
    unlabeled= array of unlabeled points
    clusterID=0
    while unlabeled has values and has not gone through all of the unlabeled points:
        toDo=[]
        chose value from unlabeled array and add to toDo list
        while toDo is not empty:
            get random point from toDo list and remove it from toDo list
            check if what type of point it is
            if core:
                remove point from unlabeled points
                add points within the radius to toDo list
                assign cluster ID to point
            if border and not the first:
                remove point from unlabeled points
                assign cluster ID to point
            if noise:
                remove point from unlabeled points
                assign to noise ID
        change clusterID
    Assign any left over points to noise
```

Figure 1: DBSCAN pseudo code

As shown in Figure 1, the concept of the code is that clusters are built out of core points. If a point is a core point then the points are tested around it, to see if they are core points as well. This procedure continues until all of the core points that near the original one tested are identified. All of these points are classified as one cluster, and any point that falls withing the given range is also considered to be a part of the cluster and is labeled as a border point. This continues until all the points have been either classified or have been tested to be see if it is a core point. All other points are labeled as noise. It is important to note that it is possible that noise points are border points to each other, but since there aren't enough points within the given distance they won't be considered as a cluster. It is also possible that one point will change clusters depending on which cluster is tested first.

The proposed improvement from the DBSCAN++ procedure is to limit the points that can be initially chosen to be tested to see if they are core points. These points have two different ways of being determined. The first is to uniformly distribute them through out the dataset. The second in to try to place the points using the K-Center algorithm. This attempts to make it so that the distance between the points chosen and the rest of the points is minimized. In both cases the amount of points needs to be given to the procedure, so that it is able to do the calculations to determine which points are going to be tested as the initial core points. In Figure 2, the pseudo code can be seen once the initial points are given.

```
def DBSCAN(dataset,radius,minPts,initPoints):
    unlabeled= array of unlabeled points from dataset
    clusterID=0
    while initPoints has values and has not gone through all of the initPoints points:
        toDo=[]
        chose and remove value from initPoints array and add to toDo list
        while toDo is not empty:
            get random point from toDo list and remove it from toDo list
            check if what type of point it is
            if core:
                remove point from unlabeled points and the initPoints if in it
                add points within the radius to toDo list
                assign cluster ID to point
            if border and not the first:
                remove point from unlabeled points and the initPoints if in it
                assign cluster ID to point
            if noise:
                remove point from initPoints
                assign to noise ID
        change clusterID
    Assign any left over points to noise
```

Figure 2: DBSCAN pseudo code

As seen in Figure 2 and Figure 1, the differences between the two approaches is very minimal. The main thing is as mentioned the narrowing how many points are tested initially to be the core points. By doing this it is hoped that the points that are not reached are either noise, or in such small clusters, that it is not essential to the data. It is important to note that it is important to choose the correct number of points that are being tested, since if two few are chosen, it is possible that they will all be in the same cluster, especially when the points are chosen using the uniform method of choosing the initial points.

It is also important to note that the way the distance is measured can be set to many different equations. In this report,for both of the DBSCAN procedures, the Euclidean distance was used. The amount of initial data points will be changing between dataset to dataset. This is because all of the datasets have different sizes, and as a result it does not make sense to keep it the same between all of them. The method of finding the best settings for the radius being searched in and the minimum amount of points was determined by testing a range of both settings, and then taking the one that had the best of the three measurements for analyzing clustering. Since there was three different measurements being taken, if a certain configuration was better in two of the measurements and worse in one, it was deemed to be better. The measurements used will be discussed further in Section 2.3. Overall this is a brief overview of how the DBSCAN and DBSCAN++ procedures work.

2.1.2 OPTICS

The next procedure that was used was the OPTICS procedure[2]. As mentioned previously, this is procedure is similar to the DBSCAN procedure. This is that both use core points and have a core radius that they use. The differences in OPTICS comes from that it also has a reachability value that it is calculating. The reachability distance is calculated by calculating how far a point is from the core point. If a point is outside the core radius from the core point, then the true distance is assigned to the reachability value, otherwise the distance is the core distance. Through the use of these reachability distances a plot can be devloped as seen in Figure 3

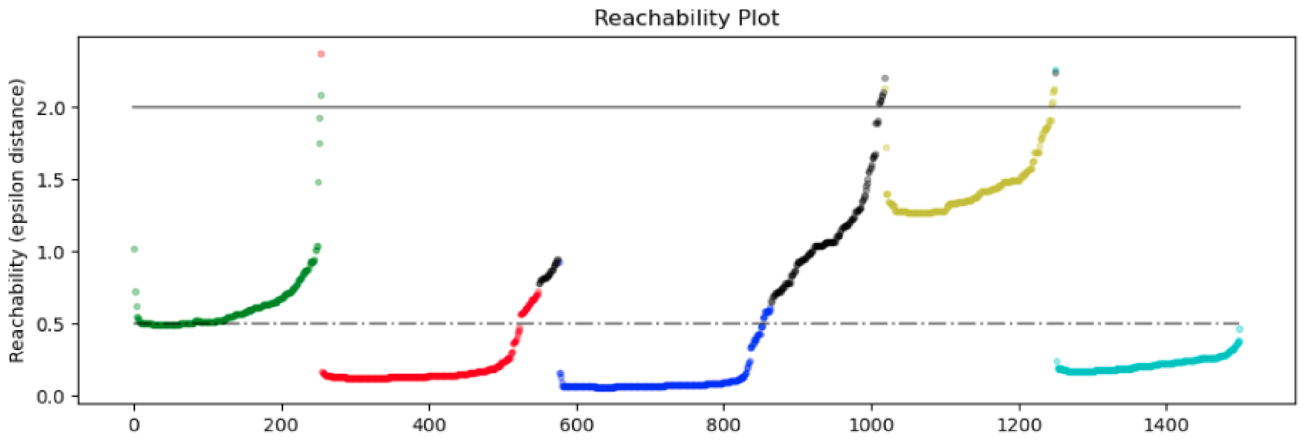


Figure 3: OPTICS Reachability Plot[1]

In Figure 3 the reachability plot from the sklearn website[1] can be seen. This is what the optics procedure produces after it calculates all of the reachability distances. The valleys that occur in the reachability values are then classified as clusters. As mentioned before the main reason for choosing this procedure was due to its similarity to DBSCAN and DBSCAN++.

2.2 Datasets

In order to test out the procedures selected a total of 5 datasets were chosen. These were chosen due to their sizes and number of attributes. Since all of the procedures for clustering chosen are not able to work with string values, the string values were one hot encoded. Although ordinal encoding could have been done, it was felt that one hot encoding would be better to avoid making any relation between the possible data values. The data was also normalized for the three actual data sets to try to simplify the range of the core radius being searched for. As mentioned before there were two data sets that were artificially created. These being the blobs and moons data set. These were created in order to quickly check if the clustering procedures appear to be working. The real datasets were the Iris[7], Pima Diabetes[8], and Acoustic Extinguisher Fire Datasets[6]. These were to see how the clustering procedures would perform on more real data.

2.2.1 Blobs and Moons

The blobs and moons dataset were both created using the sklearn's functions with similar names. In both cases the random state was set, so that across all of the states, the same instances would be tested allowing for a more accurate comparison. The settings across the functions were also set to be the same. The blobs function will create a specified amount of blobs around given points in a grid. The total amount of points is also given. The standard deviations can also be changed for the different blobs, allowing for some blobs to be harder to clusterize than others. In this report, three different blobs were made and can be seen in Figure 4 in the results section 3.1.

The moons dataset will produce two moon shaped clusters using a given amount of points. In this dataset noise can be added to try to distort the moons and make it harder to clusterize the two moon shaped objects. This dataset was also used as a quick check to see if the procedures chosen are able to clusterize the data correctly and everything was set up correctly. The moons that are being produced can be seen in Figure 5. These two datasets are useful as a quick check on the procedures, since it is can be easily displayed and as a result it is simple to see how the clusters change when different parameters are being changed.

2.2.2 Iris Dataset

The Iris dataset is a relatively small data set. It only has 150 instances, with three classes, and four features. The data set was ordered by its class. Although it is not essential to shuffle the data for these procedures, it was as a precaution. The dataset does have an equal amount of instances for each label. As a result, it is unlikely that a label will be missed due to the points being selected to be tested not being inside the label. Besides the class feature, the rest are numerical values so as a result no one hot encoding was done on this dataset. This dataset was used to test how the algorithm performs on a smaller data set and a method of quickly checking that everything appeared to work. Biased off of previous experience two of the labels are more similar to each other, and the third label can be easily separated out. As a result, it may happen that the procedures chosen may not be able to separate the two more closely related labels.

2.2.3 Pima Diabetes Dataset

The Pima Diabetes dataset has 768 instance with 8 attributes and two classes. This data set's order already seemed to be randomized, but in case it was not it was shuffled anyway. The data set again had only numerical values for the features, besides the string for the label. As a result, again no one hot encoding was done on this data set. It is about is about 2 to 1 of "tested_negative" label to "tested_positive". The dataset is slightly unbalanced with its labels, but there are still more than enough points to be able to have initial points in both labels. This dataset was used to further test algorithm after it was shown to work with the smaller dataset.

2.2.4 Acoustic Extinguisher Fire Dataset (Fire Dataset)

Lastly is the Acoustic Extinguisher Fire dataset, which may be referred to as the Fire dataset in the future. This was by far the largest dataset out of the ones tested. There are 17,442 instances in this data set, with six attributes and two classes. One of the attributes does contain string values, and as a result needed to be one hot encoded. There were four values for the string values. This means that the dataset ended up having nine attributes instead of six. Since the string attribute was dropped, but replaced by the one hot encoded ones. As mentioned previously, this was done to since the procedures chosen would not be able to calculate the distances between the instances if they contain string values. This dataset also seemed to have some order to it. As a result the shuffling was again done as a precaution although it should not be necessary. The dataset was also pretty much balanced between the two labels, so again there should be no reason to believe that one of the labels was missing from the initial points. This data set was used to help show how the algorithm operates on large amounts of data and make sure that it could handle data sets with string values.

2.3 Measurements

As mentioned previously there were four different measurements taken when evaluating the procedures. The first is time, which was just done by taking the average of who long it each procedure takes to attempt to clusterize the data given to it. A further description of this measurement is not needed, the other three are the Adjusted Rand score, Fowlkes Mallows score, and lastly the Calinski Harabasz. These scores were chosen to have a mix of interior and exterior criteria.

2.3.1 Adjusted Rand Score (ARS)

The first measurement being used it the Adjusted Rand score. This is an external index and is an extension of the Rand score as the name implies. The equation for the rand score can be seen in Equation 1.

$$RI = \frac{(a+b)}{a+b+c+d}$$

a- the number of pairs of elements that share the same subset in both datasets

b- the number of pairs of elements that are in different subsets in both datasets

c- the number of pairs of elements in that is in the same subset in data set X and is in the different subset in the dataset Y

d- the number of pairs of elements in that is in the different subset in data set X and is in the same subset in the dataset Y

(1)

The RI is calculating how much of the data is in the same cluster as the correct clustering. So that if the point is in in the correct cluster then it will help provide a higher score. If the instance is labeled incorrectly then it will lower the score, due to increasing the lower part of the equation. This will be due to it making either the c or d value being higher. The highest value it is able to achieve is 1 and the lowest is zero. The ARI is then created by adjusting the RI index for "chance". The ARI is then adjust using the equation seen in Equation 2

$$ARI = \frac{RI - RI_{Expected}}{max(RI) - RIExpected} \tag{2}$$

The ARI can now range from -1 to 1. If the index is 1 then it means that the clusters match perfectly, 0 being that the clusters are completely random. A negative score means that the clustering have less in common than if the were guessed at random. Since this seems like a fairly common method of measuring performance for clustering exercises this was one of the chosen metrics.

2.3.2 Calinski Harabasz Score (CHS)

The Calinsk Harabasz score is an internal score that is used to judge the procedures being used. The Calinsk Harabsz score is also known as the Variance ratio criterion. The simplified explanation for this index is that it is dividing the separation of clusters by the cohesion of clusters. The separation of clusters in this case means how different the cluster is to other clusters, and the cohesion is how similar the cluster is to other clusters. The complete equation can be seen in Equation 3

$$CH = \frac{\frac{\sum_{i=0}^{K} |C_i| \times \|\mu_i - \mu\|^2}{K-1}}{\frac{\sum_{k=1}^{K} \sum_{i=0}^{|C_i|} \|x_i - \mu\|^2}{N-K}} \tag{3}$$

As seen, this data set does not use another dataset to compare. As a result, it may be more helpful for when using datasets, where the initial labels are not known. There is also no limit to this index. In general it is considered that the higher the value is considered to be better. It is suggested to look for a peak or steep elbow to determine what the appropriate setting is when using this measurement. In order to easier automate this measurement, it was decided to just to consider that the highest value would be considered as the best setting.

2.3.3 Fowlkes Mallows Score (FMS)

The Fowlkes Mallows score is another external score that was used to judge the system. This was chosen to help break ties if the happened between the other two metrics used. It was also decided that in the case of the report external scores are more helpful, due to them being able to use the accurate labels for the datasets. The equation can be seen in in equation 4.

$$FM = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} \tag{4}$$

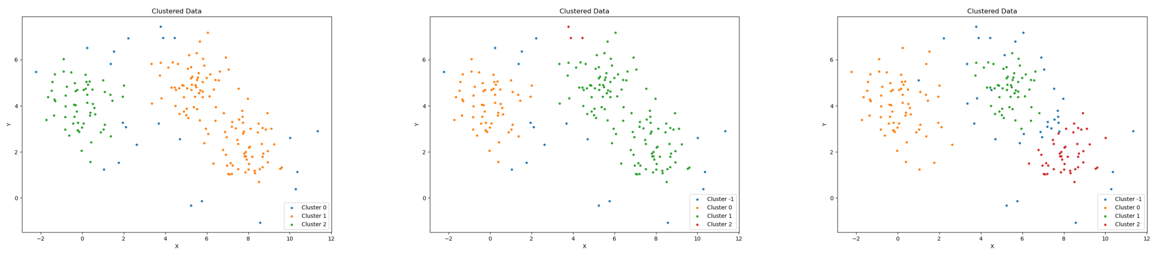
The equation 4 pretty much shows that what it is doing is getting the square root of the precision and recall. The value for TP,FP,and FN are calculated the same way as a, c, and d in Equation 1 respectively. It could be possible that since similar values are being measured for both Adjusted Rand and Fowlkes Mallows, that the bias will be heavily towards these two measuring methods when finding the best set up. As mentioned previously, this is issue was overlooked due to it being considered that the external scores are preferable to the internal ones. So biasing towards them is preferred.

3 Results

The results of the report will be broken down into the data sets that provided them. It will be mainly focused on the performance of the best determined set up, the timing of the function, and seeing if it was able to predict the correct number of clusters. The timing will be used to see how well the created script compares to ones that were are provided with the libraries. It is expected that the results between the DBSCAN++ and DBSCAN should be similar, since the core function between the two are similar. The correct number on labels is also important, since it is important to see if the algorithim was able to find the correct number of labels for each dataset. It is important to note that when talking about the results the noise label will not be counted as a label. Since the real datasets have more than 3 features, it was decided not to plot them, and use the lessons learned from the first two datasets to try and interpret what is happening with the results.

3.1 Blobs

The first dataset that was tested was the artificially created blobs dataset. Both DBSCANs had slight issues with getting the clustering correctly. The main issue is that both procedures merged two separate blobs into one large one. This is likely because the two blobs are not very separated from each other and that there were enough instances between the two blobs for the DBSCAN to be able to cross the gap between them with the core points. It is likely that a increasing the restriction would be able to separate the two blobs, this could either be done by decreasing the core radius or increasing the amount of points that are needed to be inside the core distance. The plot results for the Blobs dataset can be seen in Figure 4



(a) DBSCAN++ (b) DBSCAN (c) OPTICS

Figure 4: Blobs Plot Results

Interestingly OPTICS was able to differentiate the two blobs from each other. This is due to it having a different calculation for what the clusters are. The DBSCAN was did say that there are three label as seen in Table 1 and Figure 4b. Although it seems like it labeled a couple of outlier points from one of the clusters as the cluster. It is likely that the reason that the DBSCAN++ would also have labeled these spots as a seperate cluster, and that the only reason that it hadn't was because none of these spots were chosen to be tested as the core points. Looking at the the measured results seen in Table 1, the DBSCAN and DBSCAN++ performed very similarly. The slight increase in performance for DBSCAN, is likely due to the small start of the third cluster.

Table 1: Results for Blobs

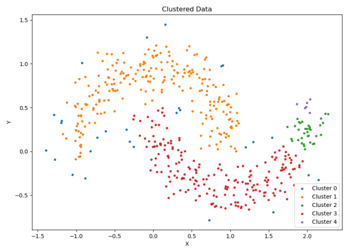
	#of Labels	**ARS**	**FMS**	**CHS**	**Avg. Time**
DBSCAN++	2	0.474	0.684	278.83	0.552
DBSCAN	3	0.477	0.685	185.05	0.004
OPTICS	3	0.711	0.800	427.31	0.251

Table 1 also shows that OPTICS performed much better in every one of the measured results. This is definitely seen in Figure 4c. Although without the prior knowledge of that there were three clusters, it may be hard to tell that this in fact is the better clustering, and the two DBSCANS may have been thought to be preferred. The fastest method was the DBSCAN. This is likely due to it being less complex than OPTICS, and due to it being programmed better than DBSCAN++. The fact that it is faster for the Blobs dataset, will shows that it is likely to be faster in every other dataset as well. Since as the data starts becoming more complex and having more instances, the longer it will take the other methods to finish their tasks.

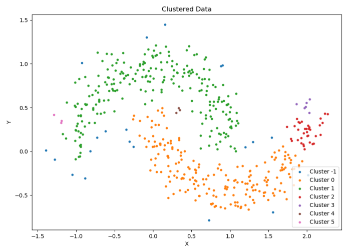
Overall the blobs dataset showed that the two DBSCAN procedures may have issues with identifying clusters that may be overlapping. In order to be able to avoid this a lot of fine tuning is likely required. The OPTICS procedure seemed to have less issues for this. The DBSCAN was also shown to be significantly faster than the other two procedures.

3.2 Moons

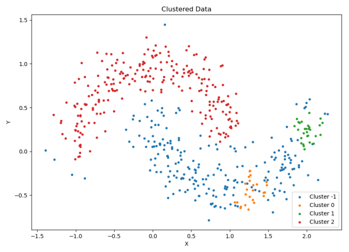
The next dataset that was tested to make sure the clustering procedures are working correctly was the created moon dataset. Interestingly all of the procedures tested had issues with getting the correct number of labels, and it is not exactly clear why it happened. Looking at Figure 5, it may be due to the fact that the two clusters are a lot closer to each other. As a result, the natural variation within in the clusters causes them to be considered to be another cluster, in order to avoid clustering the two larger clusters as one large cluster.



(a) DBSCAN++



(b) DBSCAN



(c) OPTICS

Figure 5: Moons Results

The most surprising thing looking at Figure 5, of how poorly it seems that OPTICS performed. It is important to note that the -1 label is pointing out noise and not a cluster. This means that in Table 2, the results being shown for the OPTICS will be artificially much higher. Since a more accurate calculation would have considered these points as False Positive in some parts of the calculations. Due to it being considered as a True Positive it has artificially raised the result scores.

Table 2: Results for Moons

	#of Labels	**ARS**	**FMS**	**CHS**	**Avg. Time**
DBSCAN++	4	0.730	0.854	98.96	2.438
DBSCAN	6	0.730	0.854	69.52	0.007
OPTICS	3	0.695	0.836	120.92	0.667

Although the OPTICS did have a boost in it scores, since it seems to be performing too well for what is seen in Figure 5c, the two other procedures were shown to perform better than it. The DBSCAN++ is shown to be performing slightly better than DBSCAN. It is not clear as to why this would be, it is the internal measurement instead of one of the external measurements. This means that means that the reason is likely that the clusters are more different to each other and as a result affecting the results. Looking at the clustering seen in Figure 5b, it is seen that again there is a small outlier of instances that were clustered together for the DBSCAN procedure, that do not appear in the DBSCAN++ procedure. This is again likely due to the points required to create this tiny clusters were not included in the points that were tested for core points in the DBSCAN++ procedure. Since these points are not real clusters it does not hurt the performance of the DBSCAN procedure. The procedure starts to perform better in the Calinski Harabasz Score, because it has more clusters that are similar to each other.

Another interesting note is that the OPTICS procedure outperforms the other two procedures in the CHS measurement. Since it is due to the measurement of the variance of a data point to its own cluster compared to others, it may be that the smaller cluster had less variance in them thus allowing for a higher score. The adjustment for the noise data, would likely greatly affect the result for this measurement.

The results for the Moon dataset showed that for this dataset, the DBSCAN++ was the best performer in terms of the measured resutls. It was again the slowest procedure out of the ones tested. All of the procedures had issues with giving the correct number of labels. The likely reason being that due to the two clusters being closer together, causing it harder to separate the two. In order to try to do it it was required to create more smaller clusters within the overall larger cluster.

3.3 Iris

The Iris dataset was the first dataset that was tested that was an actual dataset. The results seen in Table 3 show that the all of the datasets were able to perform fairly well. Interestingly, the external and internal results are reversed. This may be due to the shape of the clusters being different, or the space it was taking up. Also again none of the datasets were able to predict the correct number of labels. This was slightly expected from previous experiences since it was known that two of the labels seemed to be very similar to each other, and as a result it would be hard to separate them from each other. On first instinct, the DBSCAN again created two small outlier groups as has happened in the first two datasets tested, but this seems to be dis proven by the fact that the ARS and FMS values are not similar to each other. As a result, it is more likely that there was four different clusters created by the DBSCAN.

Table 3: Results for Iris

	#of Labels	ARS	FMS	CHS	Avg Time
DBSCAN++	2	0.716	0.811	170.99	0.157
DBSCAN	4	0.444	0.601	2404.35	0.006
OPTICS	8	0.401	0.572	5757.09	0.172

The discrepancy in the internal results of CHS is hard to explain. The previous explanation that was discussed is that the clusters created by the DBSCAN and OPTICS with smaller groups creates clusters with smaller variance to each other. As a result it produced a much higher result. The fact that it does not follow the trend of the two internal measurements, may mean that for this dataset it is a poor choice of measurement, due to the clusters having a high variance to each other.

Lastly, interestingly in this data set the OPTICS procedure performed slower than the DBSCAN++ procedure. It is hard to determine what would be the cause of this. It is possible that the computer at the time of the OPTICS test was performing other intensive tasks, and as a result performed the OPTICS task slower than it normally would. Although it is unlikely due to when looking at previous datasets the difference between the two procedures is significant, and is unlikely that the computer explanation would work. It may also be due to that the Iris dataset, being more complex than the two others being tested. As a result more computations were requried and slowed down the OPTICS dataset enough to take longer than the DBSCAN++.

The overall results seem to show that the most accurate results are produced by the DBSCAN++ procedure. This may be due to it not creating cluster of outliers, like DBSCAN seems to have a tendency to do. It was also shown that the CHS seems to be a bad measurement for this dataset, since it goes against the trend shown in the other two measured results. The fastest procedure again has been shown to be DBSCAN.

3.4 Pima Diabetes

The next data set is the Pima dataset, this dataset is larger than the Iris one and from previous experience was harder to separate the labels from each other. As a a result, it is not surprising to see that the procedures are doing worse than compared to the other datasets. The two other procedures were not able to produce the correct number of labels, with saying that the data set contains 49 labels, when it only contained 2. The DBSCAN++ seems to get the correct number of labels. The FMS value for the DBSCAN ++ also seems to be showing that the accuracy is not very bad. Although the ARS shows differently for all of the procedures. Showing that pretty much for all of the algorithms tested the clustering is pretty much random.

Table 4: Results for Pima Diabetes

	#of Labels	ARS	FMS	CHS	Avg Time
DBSCAN++	2	0.086	0.640	38.14	0.815
DBSCAN	49	0.058	0.477	10312.25	0.021
OPTICS	49	0.030	0.484	25403.55	1.003

The CHS in this data set also again seems to be going against the the results shown for the other two metrics. It is hard to tell what is happening overall in this dataset. The best procedure seems to be DBSCAN++ due to it being the best in two of the metrics measured and that it was able to predict the correct number of labels, but the ARS seems to show that even for this procedure the clusters are random.

The DBSCAN++ method again has been shown to be faster than the OPTICS method. This may be showing that for more complex datasets the DBSCAN++ is a much faster method than the OPTICS method. As always the DBSCAN method is fastest method available. Overall it is hard to make any conclusions from this dataset. It seems that the DBSCAN++ is the best, but it is hard to say how well the data it clustered is.

3.5 Acoustic Extinguisher Fire Dataset

The last data set is the Acoustic Extinguisher Fire Dataset, which is by far the largest dataset. In previous expierence, this dataset was able to be labeled very accurately, so it was expected to be easy to seperate the data into clusters. Looking at the results, this does not seem to be the case. None of the of the procedures were able to predict near the correct number of labels of 2. Also all of the metric values being pretty much the lowest out of all the models tested. This may be due to this being the only dataset where one hot encoding was applied. As a result, the distance between these features makes it difficult to calculate well. Since if a feature is not important to the labeling, it will be hard to cluster since there is already a large distance between them when compared to the other values.

Table 5: Results for Acoustic Extinguisher Fire

	#of Labels	**ARS**	**FMS**	**CHS**	**Avg Time**
DBSCAN++	237	0.002	0.256	120.93	283.79
DBSCAN	1139	0.025	0.370	1177.72	0.891
OPTICS	6	0.003	0.106	6629.89	57.498

When looking at the ARS values measured in Table 5 it is seen that pretty much for all of the procedures the clustering seems to be completely random. None of the procedures scored highly in the FMS measurements either. Although CHS seems to be performing well, looking at the other datasets with actual data it is the lowest. The best algorithm does appear to be DBSCAN with it scoring the highest in ARS and FMS.

Overall this dataset was clearly the hardest for the algorithms to handle. As theorized it may be due to the one hot encoding of the datasets, and this causing a biasing towards those features. The best procedure seems to be DBSCAN according to the metrics measured, but it has way too many labels. The best procedure according to the labels was OPTICS, but it performed the worst according to the two external metrics. So maybe focusing the CHS metric would have been better, but this did not seem to be the case for the other two real datasets.

4 Conclusion

Overall, the report had a lot of surprising results. It was expected that the DBSCAN and DBSCAN++ algorithms would perform similarly, due to them having a very similar method of clusterizing data. It seems that the improvements that DBSCAN++ had is that is tends not to make clusters out of small clusters that can be located in the outlier data. This seemed to happen a couple several times with the DBSCAN method. It may also be surprising that the DBSCAN method is faster than the DBSCAN++ method, since according to the paper the DBSCAN++ was created to help decrease the runtime, it is likely that the reason for it taking longer is due to inefficiencies in the code produced. It is likely if the DBSCAN part of the DBSCAN++ was taken from the code and modified to be purely a DBSCAN method it would be slower than the DBSCAN++ tested. It is also possible that the DBSCAN function called ends up using some other langauge than Python that is better at doing these types of calculations, thus making it significantly faster.

It was also interesting to see that in pretty much all of the datasets tested the OPTICS function performed the worse, and was at times slower than the created DBSCAN++ model. It is hard to tell what was the cause of this. It seems that the more groups the OPTICS model created the longer it took, so it is likely something to do with the calculations it is doing for each cluster. It does seem that for accuracy the DBSCAN++ method would be preferred to the OPTICS method.

All of the methods tested were shown to perform worse for the real data, than the produced data. This is not surprising since the real data is more likely to be intermixed. Although, the very poor performance for the Acoustic Extinguisher Fire dataset was very surprising. As mentioned in the results section, this may be due to the one hot encoding. Although for the Pima Diabetes dataset, the ARS values were also low. So it is hard to determine if this is the reason for it.

Lastly, it seems that the CHS measurement seemed to be the worst one to use. It often went against the trend in the other two measurements. This is likely due to it being a external measurement of the performance. Although, it still should be expected that it stays closer to the trend most of the time and not differ from it every time as it seems to have happened in this report.

Overall, the report shows a successful build of the DBSCAN++ model, although it is more slower than the DBSCAN model it was tested with. It was able to perform better in most of the datasets that it was tested in than the DBSCAN model, due to it not creating smaller outlier clusters, and it was shown to perform better and generally be faster than the OPTICS model in the real datasets. The more complex datasets seemed to show a problem for this model, and it seems that it is better to avoid datasets were one hot encoding would be necessary.

In conclusion, the DBSCAN++ procedure is a useful one to use when the DBSCAN model can be used, especially if it can be coded correctly so that it is efficient.

References

[1] Demo of optics clustering algorithm.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, page 49–60, New York, NY, USA, 1999. Association for Computing Machinery.

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[4] Junhao Gan and Yufei Tao. Dbscan revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 519–530, New York, NY, USA, 2015. Association for Computing Machinery.

[5] Jennifer Jang and Heinrich Jiang. Dbscan++: Towards fast and scalable density clustering, 2018.

[6] Murat KOKLU. Acoustic extinguisher fire dataset, Apr 2022.

[7] UCI Machine Learning. Iris species, Sep 2016.

[8] UCI Machine Learning. Pima indians diabetes database, Oct 2016.