RMIT University

Implementation of Clustering Models KMeans And DBSCAN Against Letter Recognition Dataset With Distortion

Name: Nguyễn Quang Minh

sID: s4061158

Email: s4061158@rmit.edu.vn

22/05/2025

Contents

Abstract

This report investigates the effectiveness of two unsupervised clustering models—KMeans and DBSCAN—on the Letter Recognition dataset originally compiled by Frey and Slate. The dataset consists of 20,000 character images from 26 English letters, represented by 16 numerical features derived from geometric and pixel-based properties. Each sample includes distortions in font, scale, and warping, which complicate clustering. KMeans and DBSCAN were evaluated through feature analysis, visual inspections, and clustering accuracy compared to true labels. KMeans achieved a maximum of 29.77% accuracy but showed limitations in handling overlapping and non-spherical clusters. DBSCAN demonstrated sensitivity to parameter tuning and struggled with noise and high-dimensional data. The results highlight both the challenges and possibilities of clustering distorted character data and underscore the need for dimensionality reduction or more robust methods in future work.

Introduction

The recognition of handwritten letters is a fundamental problem in pattern recognition and machine learning, utilizing the many capabilities of computers limits and human understanding of the problem with optical analysis, data gathering and usage... Thus in 1991, David J. Slate collected exactly 20,000 samples of black and white rectangular pixel displays which each contains a single letter of the English alphabet, ranging from 20 different fonts and distorted to be fed as data for machine learners.[1] With such a huge dataset, they decidedly identified and compiled data across 16 numerical attribute. The distortions applied ranges from font type, letter of alphabet, linear magnification, aspect ratio, and horizontal, vertical “warp”. [1] Making a distinction with black pixels being “on” and white pixels being “off”, said pixels are shuffled around with the distortions having a specified range of magnitudes which resulted in aftermaths that were, as they quote: “the resulting character images, although rather misshapen, were fairly recognizable to humans.” [1] Finally they were scanned throughout their box, resulting in said 16 attributes and, for use, also scaled linearly to a range of integer values from 0 to 15 which they think are adequate for the 26 letters. [1] These attributes collectively provide a rich numerical representation of each letter’s shape and structural features, enabling effective clustering and classification analyses. With their own use of “an adaptive classified system m similar in many respects to the approach pioneered by Holland”, they have achieved an accuracy with the model of somewhat a bit over 80%.

For my own report, I would be performing on the same dataset but making use of newer models, specifically the unsupervised clustering models K-means and DBSCAN from modern Python libraries pandas, sklearn and matplotlib. K-means partitions data in groups of pre-

defined clusters based on a provided parameter by minimizing the sum of squared distances between data points and their assigned cluster centroids, this however goes with the assumption that clusters are based off spherical or similarly shaped formations; DBSCAN, or Density-Based Spatial Clustering of Applications with Noise, on the other hand groups data points together based on their distances and specified range of data points required to form their clusters with core points and border points apart from noise, its performance is generally decided by the inputted neighborhood radius (eps) and minimum number of samples to form region clusters (min_samples). After preparing the data for my specific needs, from descriptive tabling and graphing capabilities provided by more modern computing processors, I hope to be able to find appropriate and optimal parameters, clusters, eps and more, for these models to perform at their highest potential and then be able to compare them within one another to derive common rules, advantages and disadvantages and room for discussion together with recommendations based on results.

Methodology

Research Goal

Firstly, as implied by the data science project, a research goal should be set. Hence, for the determined dataset and algorithms, it is decided to be the investigation on the effectiveness of unsupervised clustering techniques, K-Means and DBSCAN particularly and generally as a whole in regard to letter recognition. By use of a range of distorted letters, which are represented by geometric and statistical features, to train the model for generalized character groups the research hopes to evaluate how closely it aligns with its true letter categories. From its success, or failure, a general list of limitations, strengths and considerations should be derived surrounding their viability for pattern discovery, feature classification within the text recognition domain.

Data Retrieval

Now begins the exercise of data retrieval, the dataset is first obtained from download and picked between 4 other possible datasets to be chosen from. Within it lies the dataset appropriately contained in text format, “letter-recognition.data” and separated by commas for different columns and a new line for different rows. In addition, a separate “letter-recognition.names” file is provided for insight on metadata of the dataset regarding attribute information, class distribution and more. Links regarding the dataset link on the internet is provided and the original dataset author and original use for reporting by P. W. Frey and D. J. Slate. It is specified that the data would contain no missing attribute values, and from the listings of the class distribution an almost equal representation is applied for all classes within 20000 instances. The columns are divided into 16 numeric attributes (17 if counting letter label also) and can be generally detailed as follows: Each attribute captures distinct geometric and statistical properties of the letter's pixel distribution within a minimal

bounding box enclosing all active (“on”) pixels: horizontal and vertical positions of the bounding box center, measured respectively from the left edge and the bottom of the image; width and height of the bounding box in pixels, the total count of “on” pixels representing the letter; normalized mean horizontal and vertical positions of all “on” pixels relative to the bounding box center, capturing pixel distribution biases (e.g., left-heaviness); horizontal and vertical pixel variance measures, reflecting how spread out pixels are within the bounding box; correlation attributes measuring the directional tendencies of pixel arrangements, including diagonal orientations; higher-order moments capturing the relationship between pixel positions and their variances, which provide finer detail on shape and structure; edge count features that quantify the number and position of pixel edges detected by scanning horizontally and vertically, differentiating complex shapes like “W” or “M” from simpler letters; positional sums of edges, highlighting regions within the bounding box where edges are concentrated (e.g., edges near the top for the letter “Y”).

Data Preparation

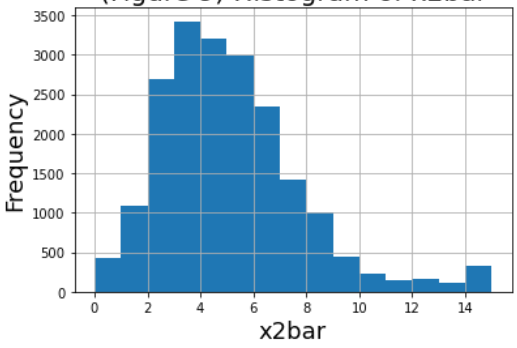
Though in the files it has specified the metadata of the dataset, after retrieval of data it could still be corrupted and/or false so the process of data recognition is required. After download, the directory for the dataset is provided and inputted for the file_path variable in the Jupyter Notebook file. Since column names weren’t specified directly within the data file, it is appropriately listed and attached to the csv translation process to get DataFrame class letter_recognition variable to be used. A basic check on .columns proves success of the column name integration, .shape for total number of samples of 20000 and attributes for 17. The .dtypes correctly returns integer values for all attributes, the letters are uniquely all within 26 letters on value_count. During describe(), as mentioned by the original authors, the numerical data of each column is normalized in a minimum of 0 and maximum of 15.[1] with no outliers outside the range. A search for null values using dropna=False is done successfully without any found.

Data Exploration

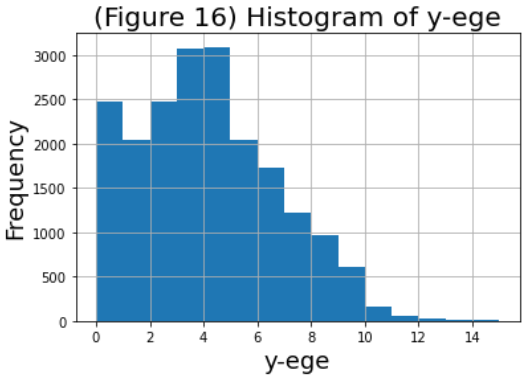
Single Attributes

Coming to data exploration, the choice of graphing options and statistic patterns to look out for will be solely dependant on how the data is represented, formatted and used. Taking into account of the letter recognition dataset, it contains only one categorical label attribute, and all others are ratio numerical attributes, normalized within a 0-15 range with 0 representing the non-existence of that attribute in the sample. Hence it is justified to use histograms, to measure the frequency distribution of the numerical attributes within a 1 step interval from 0-15, and boxplots, for how the samples distribution is spread out and tendency along with any outliers.

(Figure 9) Histogram of x2bar



(Figure 16) Histogram of y-ege



Firstly the categorical letter attribute is graphed with a box plot in Figure 1, and we can quickly see that the distribution is mostly even and around ~800-750 for all letters, with the maximum difference between U and H being about 79 which is negligible compared to the total of 20000 samples and average 800 of all letters and so it is safely considered that all is being equally represented.

From Figure 2 we can see that the histogram shows the x-box of samples has a high tendency to be within the low 3-5 range with an entire 25-75% of the samples lying within said range, when referencing Figure 18. It can prove that the letters horizontal centers from the dataset is mostly clustered near the image center.

Coming to Figure 5, it seems that the height of letters can almost be separated binarily and most other values may be accounted for by the distortions applied, with the 2 peaks reaching over 3500 and other values all under 2750. It may be caused by a distinction of letters tend towards a binary status, perhaps grouping of taller ones like I,T,L,H,Y… and shorter C,O,S,E…

How onpix is represented from Figure 6 could be due to changes in some radical distortions like font styles taking more space for the higher ends and distortions in general tend to add more onpix than take, but in general doesn't deviate that much with 50% lying within a 3-value range, from Figure 18. Such is to say the letters have a generally even number of on-pixels represented.

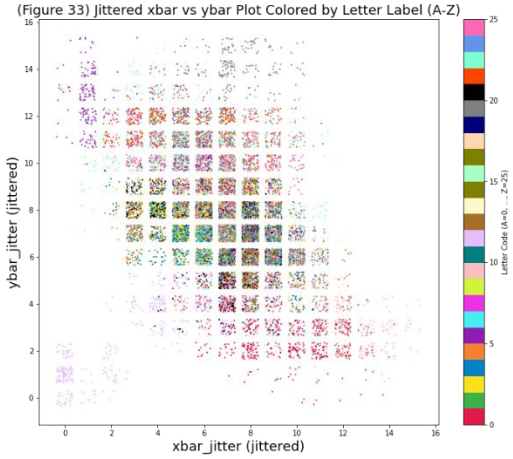
Figure 7 together with Figure 8 are almost identical with both having a singular peak in the center value, almost over 2000 samples apart from its second highest peak. We can infer from this that almost all letters are naturally centered in both directions horizontally and vertically. A difference between the two is that y-bar seem to have a couple more samples that are on the positive, around the values 10-15 enough to make push its 75 percentile from 2 to 9. It might be from the appearance of letters such as L,P,F…

X2bar represents the mean squared horizontal distance of "on" pixels relative to the center of the bounding box. Deriving from it would be the horizontal spread of pixels. From Figure 9 it is shown to be right-skewed, as most letters aren't very furtherly spread in the horizontal direction but a some such as M, W and H can and accounts for the the right-skewed part. Similarly can be said with y2bar in Figure 10, that there are not much variation except from a moderate vertical pixel spread tending towards the middle and an even smaller skew

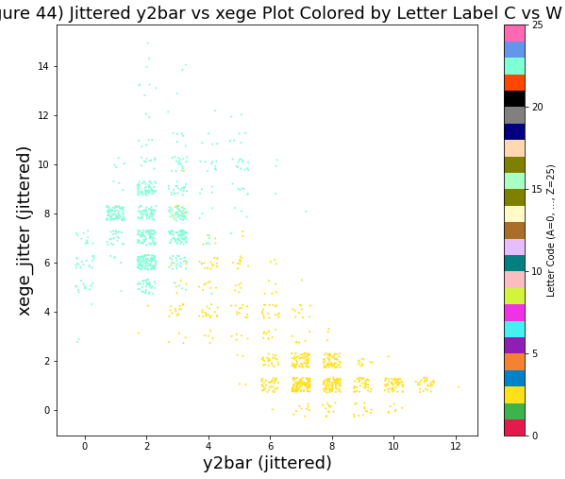
perhaps from the smaller amount of possible letters representing them.
Because the data is scaled form 0-15, thus no correlation or 0 correlation would lie within the 7 value. Thus, most letters wouldn't have any diagonal correlation, as proven by Figure 11 with its peak reaching over 5500 samples. Though there is a second smaller peak that makes it bimodal, at around the more positive 8-11 value range with 50 to 75 percentile. This means there are still several letters with positively correlated diagonal onpixel placements.
With using squared horizontal to vertical pixel position, Figure 12 indicates the letter's x2ybr maintain moderately spatial relationship between its horizontal spread vertically. The tails spreading both sides from the peak is mostly similar, possibly from symmetricals such as A and V or T and L.
Though x-ege can be only simply the count of edges, but I think from the distortions possible like font style and warping of the opposite direction of the edge being counted, alters pixel transitions and thus can significantly make distance between some samples. Such as shown in Figure 14 the right skew has a very miniscule size, though regardless mostly trends toward range 2 and 3 which may correspond to the average xege being 2 by design of the letters.
Yege may encounter a similar situation, as shown in Figure 16, however the peak is more dispersed to all its neighbouring values. A possible reason for this may be that along the columns, edges of some letters has more edge count instead of just mostly 2 like xege, like X, K…

Pairs of Attributes

(Figure 33) Jittered xbar vs ybar Plot Colored by Letter Label (A-Z)



(Figure 44) Jittered y2bar vs xege Plot Colored by Letter Label C vs W



To be able to describe the pattern of 20000 samples while they are all scaled within a small range of 0-15 in a pairing between attributes, I have decided to use a graphing method using a scatter plot but with the difference being use of having the points also added, or subtracted a random amount within a range of 0.35 of the original value (0-15) to be able to somewhat both isolate the samples and make sense the density of samples within that value and also isolate the density of different labels in said value. This is only possible from the samples being in strictly whole number values instead of also including decimal values. The differentiation of labels is done, though not highly effective, by 26 different color values (0-

25) associating with the 26 different letters (A-Z). Though it may not be very effective when plotting all 20000 samples, but it would in my use for them during quarantine of only certain samples to test my hypothesis. The quarantine of certain samples would be done mostly in pairs or above instead of each having its own plot to save space, as I would only be using at most 4-5 labels for each pairing hypothetical, so it won't matter as much, and as to better visualize their placing within the hypothesis.

The first simple hypothesis I came up with was between the relationship of width and onpix. Since the increasing of width would imply more pixels to present for it to extend to said width, there should be mostly a positive correlation between the two. Though it should not always be the case as there could be wide letters that are sparse due to variation in distortion types (such as font style). Plotting out Figure 19, it is safe to say that it within expectation as a clear positive slope seems to form. When specifically isolating W vs I and M vs J (Figure 20,21), I and J seems clear to be on the very low values while M and W spreads out on the higher value ends. Yet there seems to exist high dispersion but small densities on very high values, which could be explained by more radical distortions. Nevertheless all has the same positive correlation seen.

Though the amount of edges vertically can vary in a wide range of numbers, their position relative horizontally won't see much diversity as many letters either extend fully or many has a common vertical bar on the left corner. It is quite evidently true from the clump formed on the center-left of Figure 22, but in the context of specific samples it can separate B and F (23).

The nature of squaring values and then normalizing them within a small discrete range (0-15) should lead to a sizeable amount of samples on a tall hill-like spread in the middle values while some bulges on the extreme can be found. From Figure 24 it seems to be the case, and some distinct clusters, possibly from the same letters, appear on extreme low and high combinations. I think an important distinction can be used is for O and X as they are both very symmetrical and other combinations may align them very similarly, x2bar and y2bar seems to fairly well distinguish them (Figure 25).

Considering x2ybr and xege, though there aren't many variations in the amount of x edges, but with the combination of x2ybr it can separate letters with similar x edges but more complex, different configurations on direction and distribution of said edges. Such can be said for inside A vs V Figure 28, and the further sides of Figure 26 shows singularities also. Akin to that, its inverse xy2br and yege also shows a similar trend, where even though most yege is similar but some distinctions can be made when together with xy2br Figure 29. See also Figure 30,31,32 for some quite effective clustering of quite similar y edges but distinct xy2br letters.

Most letters are symmetrical in both x and y averages, though when during somewhat little more dislocated from the center, it only tends to go on one axis rather than both, like F, P, L, W… From Figure 33 it is justified by a negative correlation, though surprising is when isolating L it seems to be wildly different in distributions, both in concentrated and not concentrated ones, perhaps from the varying different distortions of font styles (Figure 34).

In practice, see Figure 35 and 36 for performance on X vs A and L vs T.
Pairing x2bar with xybar would allow us to explore how diagonal oriented letters may have varied pixel positions horizontally. Though this pairing is proven to be not very effective in consistency, just from looking at the overall Figure 37. Inserting pairings of letters does generate a slightly comprehensive separation between C and I from Figure 40, but along with other Figures 38, 39 it is insanely inconsistent with clustering of the same letter showing in very different places. The Figure in 41 shows an even spread on multiple values for S. Perhaps only the slightest distortion, ie stretching, font… can make huge differences on variation values like x2 and diagonal like xy.
Though between xbar and yegvx I wouldn't consider there being much correlation, as I thought where pixels are concentrated horizontally shouldn't indicate whether edges on the vertical appears more horizontally themselves. Yet from Figure 42 there shows a positive trend in the middle range, while the outliers of that range are accounted by specific letters only.
However I am certain xbar and yege shouldn't have any feedback between the two, and on Figure 43 it was proven true, with a wide spread along both axes, but no clear linear or curved correlation is immediately apparent. Even on a sample like H of Figure 44 no direct pattern comes up.
For y2bar xege, there may be a negative correlation as if y pixels are more varied, then it wouldn't "make enough space" for edges in the x direction, or the lines required to reach further direction in y would take up the amount of edges in x. Figure 43 makes an arc hints at said correlation with possible distinctions to be made with C vs W (Figure 44).

Data Modelling

On to data modelling, though the task has been provided with labels and can be treated to be used with classification, in my report I would want to utilize the 2 clustering models KMeans and DBSCAN for the task. This is because from what I have seen during the Data Exploration phase, effective separation from just pairing of attributes are possible from visualization, moreover with many meaningful attributes with a total of 16 when combined together should be able to fairly recognize with certain accuracy 26 categories of letters. It is decided all 16 attributes will be used for clustering, as they all prove to contain meaningful description on a natural level the characteristics of letters and its relationship between pixels, position, variability and edges. From the Clustering category, the 2 models KMeans and DBSCAN is picked.

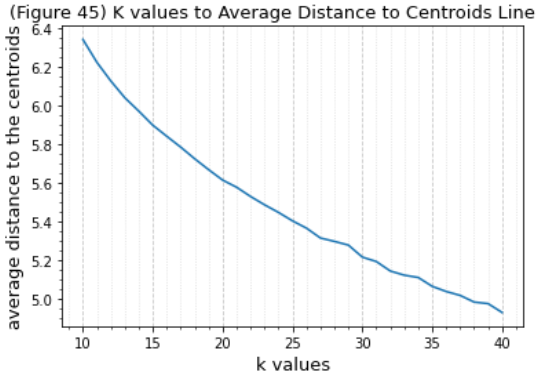
Results

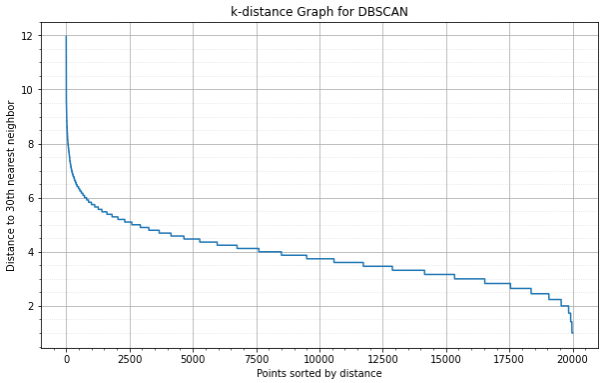
KMeans

KMeans uses minimal parameters but quite effective and simple method of clustering data points, with an input of the number of clusters and sometimes the number of iterations on

different centroid seeds and max iterations of centroid positioning, KMeans works by first initializing “centroids” and an area it works around then calculates the distances between it and data points, grouping them based on the nearest distance then decides on whether to move the centroids and repeats the process. This effectively finds clusters of data purely by distance of the points to the initialized centroids and is not much more complex than that, thus its popularity. However it is important to note its disadvantages may arise from the specific letter recognition dataset, as during exploration it can be seen some clusters aren’t very spherical in shape and varies, some distortions causes radical noisy data and outliers along with the dataset’s method of scaling from 0-15 may disrupt its non-linearity.
Though it should be from domain knowledge that we would be testing on 26 letters, a cluster number to average distance to centroid graph is still utilized to make sure. I have chosen the wide range from 10-40, and resulted in Figure 45. It resulted in very linear, only miniscule curve can be seen and may prove that KMeans will have a difficult time clustering correctly based on the data given. Though a slight elbow can be seen during the 26-28 mark of the k values. Now KMeans is ran 3 times with the cluster of 26,27,28 for each, getting the appropriate amount of clusters. The results are filled in a separate dataframe, containing the cluster index, the target labels and its counter on where the points of the cluster belongs to. A confusion matrix is set up inside the code with IPython, though is limited by the width of the rows so a separate csv export is prepared as “KMeans_confusion_matrix.csv”. Finally a cluster to letter conversion is done based on most popular true label inside each cluster, and the accuracy is calculated by the mean of all the cluster to true label conversion calculation.

(Figure 45) K values to Average Distance to Centroids Line Plot





With a setting of random state=13 and 26 clusters, it had a 28.46% accuracy, 27 clusters with 29.77% and 28 with 29.69%. Tweakings in other smaller KMeans variables such as max_iter going up to 600 and n_init up to 20 did not change the accuracy metric and so discussion on it is omitted. Looking further into the confusion matrix, it is understandable on where KMeans is unable to separate very similar letters characteristics such as a 225-185-107 split on cluster index 2 with F-P-T at the 26 clusters confusion matrix, or 108-109-109 for C-G-Z at index 14 for 27 clusters. However it can separate well very distinct characteristics like a total 100% 325 samples going from L for index 8 or 584 of A at index 24 at 26 clusters. A further analysis is done on the counting of where clusters lie on the letters of the alphabet and which letters are unrepresented by majority by any cluster at all. Letters unrepresented by

majority that appears on all 3 cluster numbers includes T,H,K,E,S,R, as the number of clusters increase from over 26 B appears, and others are included also.

DBSCAN

Onto DBSCAN, its results turn out to be not much different. Experimentation is conducted by alteration of 2 parameters: eps and min_samples. Increasing and decreasing each is the key factor for finding the right amount to effectively cluster labels correctly while minimizing noise. It can help by plotting out a k-distance graph, where it is sorted by points and the distance is graphed based on it. An “elbow point” is generally sought after for it is where higher amount of clusters begin to be less effective. For the letter recognition dataset specifically, flat lines occasionally reoccurs for the graph, a theory could because of its strict 0-15 scale normalization and nonpresence of decimal values. The general rule is using 2 times the number of attributes for a start for k values and thus after finding the elbow value it should be a local optimum. However, realistically when tested for clusters between 28-32, and “elbow” points around 6-7 eps, it creates large clusters while not really eliminating fully noise. Testing with lower min_samples values of around 5-12 and eps of 3-5 generates usually the similar amount of clusters as the number of labels, 24-32 and relatively low nose. However on accuracy inspection it is seen that it is very low of 21-25%. More and more ranging values for the 2 parameters seems to follow the same problem, either high accuracy and low noise but a huge amount of clusters (40+) or low-medium accuracy and moderate amount of clusters(26-32) but very high noise (4000+). In regards to most similar cluster amount to class label and low noise, eps=4 and min_samples=6 provided with 29 clusters and 367 noise but at 10% accuracy, eps =3.7 and min_samples=25 gave 29% accuracy and 27 clusters but at the cost of 6115 noise points. Many more examples can be given on the tradeoff present within DBSCAN.

Discussion

In regards to KMeans, it is safe to say it performed badly on the letter recognition dataset. This may be due to the many similarities in nature by the letter themselves to have similar patterns in edges, strokes… But it can also be from the possible distortions applied to the letters that may give radically different values on its attributes and thus straying from a possible spherical groupings of data samples that KMeans is meant for. From the start of graphing of k values to its average distances to centroids being linear it is the indication that KMeans won’t perform well, possibly from groupings where the points are mixed instead of separate enough to significantly change the rate of decrease of average distance to centroid as the number of centroids get higher. Another insight is how letters that are commonly unrepresented (T,H,K,E,S,R) despite some increase in cluster number appears, which indicates these specific letters has their points spread among clusters and not enough to make a majority vote on any of them to be considered.

While KMeans is effective at finding broadly defined spherical clusters, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) was employed as an alternative to handle non-spherical clusters and noisy data. Unlike KMeans, DBSCAN does not require predefining the number of clusters, which is advantageous for problems without much domain knowledge. Instead, it relies on two parameters: eps: the radius within which to search for neighbors and min_samples: the minimum number of points required to form a dense region (core point). On further inspection, when taking apart what data points and true labels lies in which cluster, most of them are stuck within either the 1st or 2nd cluster, this is part of the disadvantages of DBSCAN as it not entirely deterministic and may rely on which cluster is processed first itself. Also, with the amount of dimensions considered the "curse of dimensionality" affects DBSCAN and its reliance on distance calculations quite considerably. Sensitivity to parameter selection is another problem with DBSCAN on this particular dataset, with the large variation of points and low range of 0-15. Thus it is to derive that cluster purity vs coverage is a core trade off in DBSCAN with this dataset.

Conclusion

This report explored the use of KMeans and DBSCAN for clustering the letter recognition dataset. KMeans, while simple and efficient, struggled with non-spherical clusters and overlapping letter features. Its best accuracy reached 29.77%, but certain letters like T, H, K, and R consistently lacked clear representation across clusters.

DBSCAN, in contrast, does not require a predefined number of clusters and handles noise and outliers effectively. It was able to detect some fine-grained structure in the dataset, particularly when parameters were tightly tuned. However, its performance was highly sensitive to the choice of eps and min_samples, often resulting in either excessive fragmentation or overly broad clusters.

Overall, while DBSCAN showed slightly better alignment with true labels, both methods faced challenges due to the dataset's high dimensionality, tight feature range, and visual similarity across letters. Future improvements could include dimensionality reduction or advanced embedding techniques before clustering.

References

[1] P. W. Frey and D. J. Slate (Machine Learning Vol 6 #2 March 91):"Letter Recognition Using Holland-style Adaptive Classifiers".