Implementing and Evaluating K-Means Clustering

Christopher Linscott
clinscott@oxy.edu
Occidental College

1 Abstract

The following tutorial report dives into two clustering algorithms, Fuzzy C Means and K Means, and implements them to test on examples with blobs of data points, image pixels, and datasets of images. Afterward, the K-Means algorithm is evaluated on an iris-flower image dataset using a confusion matrix, elbow method, classification report, and Davies-Bouldin Score as different metric. Finally, these metrics are reported and discussed. While the K-Means algorithm was found to perform well in clustering these images, there are many caveats to this result.

2 Methods

2.1 Approach/Framework

The general approach to grouping images or any form of data point is the idea of clustering, where the main goal is to "...[segment] a population into subgroups where members are more similar to each other... based on certain observed features" [2]. In the context of my COMPS project, being able to separate images means being able to separate them apart based on their differences in objects, shapes, or environment. The main goal is not to recognize the objects, but to recognize meaningful differences or similarities between two images based on their visual similarity. Therefore, unlike with clustering simple blobs or batches of data points, to cluster images requires "cluster[ing] pixels into different image regions, which are regions corresponding to individual objects, surfaces or natural parts of the objects" [8]. As this is an unsupervised version of machine learning (allowing for mainly testing), there is no response "class", or that the human needs to associate meaning with each cluster it generates.

The two clustering algorithms employed are Fuzzy C-Means and K-Means. With the help of tutorials [4, 9, 3, 1], I implemented both algorithms from scratch, utilized K-Means from the sklearn library on three different examples: batches of data points, the pixels of a given image, and batches of images. Before clustering each of these data formats, I preprocessed them using functions such as OpenCV's image read function to load images, creating arrays of data points of 5 dimensions to represent its position and the values for each color channel (red, green, blue). To cluster only on the colors of the pixels, numpy's reshape function was utilized to extract only the values from each color channel, creating three-dimensional data points. After performing sklearn's K-Means algorithm on these data points, by utilizing the algorithm's labels for each datapoint's cluster, I combined each cluster's data points into a single 2-D array, creating a 3-D array of k rows of 2-D arrays containing these data points; k represents the user-inputed number of clusters. The following arrays of data points were plotted using matplotlib, coloring pixels of the same cluster (i.e. in the same 2-D array) to show the results of the full clustering.

With clustering the images from the digits dataset (in particular), the digits beforehand were preprocessed to have their pixel values inverted by subtracting the pixel value from 255. These are primarily gray-scaled images, with the majority of the pixels being black or dark gray. By inverting the pixel values, the clustering algorithm will have an easier time clustering as it needs to only determine the dark pixels primarily.

2.2 Datasets

The datasets used by both my algorithms and my evaluation came from two of the seven toy datasets included in the sklearn library [5], the iris data set and the digits data set. The iris data set includes 150 different instances of iris flowers, with three different classes of iris flowers of 50 instances each; each class here refers to a type of iris plant. The digits data set includes 1797 instances of digits, or 8-bit images; the colors or pixel values range from 0 to 16. The digits data set has 10 different classes, each referring to a digit from 0 to 9.

2.3 Algorithms

A very well-known, partition-type clustering algorithm is K-Means. Partition-based clustering refers to a harder (meaning more strict) form of partitioning where every data point can only be in one cluster. The main goal of this algorithm is to create partitions of data points, by creating points

of relevance called clusters, which minimize the distance from any given data point to any given cluster. The distance metric utilized by K-Means is Euclidean distance where the distance from a data point x_i to a cluster x_j with d dimensions can be calculated by: $D_{i,j} = \sum_{l=1}^{d} \sqrt{|x_{il}^2 - x_{jl}^2|}$. In my own implementation (for learning, understanding, and documentation), I utilized Euclidean distance for only 2 dimensions via: $D_{i,j} = \sqrt{|x_j - x_i|^2 + |y_j - y_i|^2}$.
In combination with my own implementation, I utilized Lloyd's algorithm to compute and converge on centers of clusters, which is the same as for the library utilized in the tutorial, sklearn [6].

2.3.1 Pseudocode

Algorithm 1 K-Means: Lloyd's Algorithm

Initialize k clusters' positions
Select k number of initial centers (centroids) from datapoints (at random)
while Centroid positions change **do**
  for $i \rightarrow n$ data points **do**
    for $j \rightarrow k$ clusters **do**
      Calculate $D_{i,j}$ from data point i to cluster j
    end for
    Store closest cluster for each data point i
  end for

$$C_j = \sum_{x_i \in C_j} \frac{x_i}{n}$$

New centroid position is mean of all subsequent data point positions
end while

The soft (fuzzy) counterpart to the K-Means algorithm is the Fuzzy C-Means (FCM) algorithm. The FCM algorithm is "softer" or "fuzzier" as unlike K-Means, it allows for a data point to coexist in several different clusters with different values of membership (i.e. how much it relates to that given cluster). In comparison to the K-Means, the FCM algorithm is more popular for image segmentation because it's more computational efficient (O(n) versus O(knt) [7], applicable to multichannel data (i.e. colored images), and has the ability to model uncertainty in the data [8]. In the context of the high number of pixels being grouped upon, as well as the channels of color which apply to colored images, it's clear that an algorithm like Fuzzy C Means is a very good option for image segmentation.

Fuzzy C-Means, in a very similar fashion to K-Means, will utilize a Euclidean distance metric. However, FCM utilizes membership values of the data points, where every data point has a set of k membership values adding up to 1, with k being the number of clusters. These membership values will be randomly initialized for each data point, and tweaked depending on the distance metric from a data point to a cluster. The centers of the cluster will now be tweaked based on the mean of the data points as well as their membership values. In other words, the centroid's data point will be most affected by data points which have a membership value high for that cluster. Mathematically, this is all represented by first calculating the centroid positions by computing

$$C_j = \frac{\sum_i^n (m_{ij})^f * dp_i}{\sum_i^n (m_{ij})^f}$$

with n data points dp, a cluster j and a given membership value of a data point i for a cluster j m_{ij}. The new f value represents the fuzzifier, or what will convert a hard output into a soft output (i.e. less hard clustering); a value of 1 equates to K-Means, whereas infinity equates to all values being equally in the same cluster. After computing the new centroid values, the data point's membership values will be equal to the euclidean distance from a data point i to cluster j, divided by the total euclidean distance to all clusters from the same data point. This is calculated as follows:

$$m_{ij} = \frac{1}{\sum_{i=1} k(\frac{|dp_i - c_j|}{|dp_i - c_k|})^{\frac{1}{f-1}}}$$

To converge on membership values and centroid positions, we had the following psuedocode:

2.3.2 Pseudocode

3 Evaluation

Evaluation of the K-Means algorithm was performed using "simple" images, or images with objects of few, distinct colors, and the iris and digits datasets proposed in the Datasets section of this paper. A confusion matrix, elbow method, classification report, and Davies-Bouldin Score were all utilized to generate different metrics of accuracy of the clustering algorithms.

The first proposed method was a confusion matrix, where given the ground truth and the predictions, it will plot the numbers of correct and incorrect predictions in a matrix, with the correct predictions along the main diagonal (top left corner to bottom right corner) and the incorrect predictions on either side of the main diagonal; false negatives were on the top right side and false positive on the bottom left side. Given the class names (i.e. what digits or flowers), the Confusion Matrix can specifically tell which classes the algorithm struggled to cluster.

The second proposed method was Davie Boulden's Score, which can assess the strength of a clustering algorithm without knowing the ground truth. The score is

defined as the ”average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances” [5]. The minimum and best possible score is zero, with lower values indicating better clustering.

The third proposed method was the Classification Report, which when given the ground truth and prediction, can quickly calculate and give information about accuracy, precision, recall, and f1-score of the clustering algorithm by computing various ratios/formulas as shown where TP means the number of ”true” predictions where the model predicts the correct class, TN means the number of true ”false” predictions where the model predicts the image is not a given class, FN refers to the number of false negatives where the model falsely predicts an image is not a given class (but it truly is), and a FP refers to the number of false positives where the model says incorrectly that an image is of a certain class when it's in truth another one.

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2 * P * R}{P + R}$$

The final method, the ”Elbow” Method evaluates the best number of clusters for clustering the given data; this is very useful for clustering algorithms such as FCM or K-Means where the number of clusters must be given. The main idea behind the Elbow Method is plotting the WCSS (Within Cluster Sum Square), which is the sum of the squared distance between each point and the centroid in the cluster; the squared distance can be calculated by summing and square rooting the Euclidean distance differences in every dimension 0 to d for every data point in the cluster j:

$$\sqrt{\sum_{x_i \in c_j} (x_i - c_j)^2}$$

As you plot more clusters, WCSS will decrease at a slower rate, and have diminishing returns. Once the graph begins to have a second bend, much like an elbow, where the elbow is formed tells you the best number of clusters.

3.1 Results

After performing the following evaluation methods on the K-Means algorithm with the Iris Dataset, we obtained the following results:

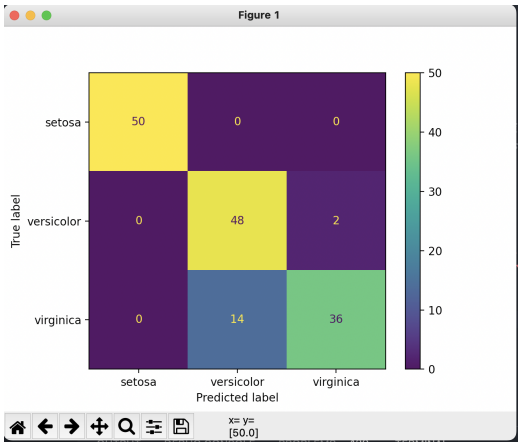

Figure 1. The Confusion Matrix generated after clustering upon 150 flowers of three different possible classes, with the K-Means algorithm.

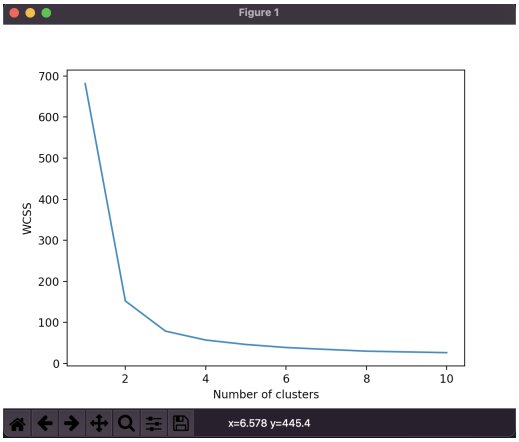

Figure 2. The WCSS versus Number Of Clusters (for the Elbow Method). The best number of clusters should be 3.

	precision	recall	f1-score
0	1.00	1.00	1.00
1	0.77	0.96	0.86
2	0.95	0.72	0.82
accuracy			0.89
macro avg	0.91	0.89	0.89
weighted avg	0.91	0.89	0.89

Figure 3. The classification report of the clustering by K-Means on the Iris Dataset. Each class was evaluated on its precision, recall, and f1-score. The overall accuracy is summed up.

As well as, the generated Davie-Boulden's Score was 0.66, which is very strong considering its proximity to 0.

4 Discussion

The results we got above are predictable and expected. As we clustered upon the three different possible classes, the "elbow" formed around k = 3 clusters; this is the appropriate and expected number of clusters to work best roughly (since there are 3 classes of flowers). As well as, the classification report, in combination with our confusion matrix, show the different scores across the different classes. As the first class (zero) was predictedly entirely correctly, the scores will all be ones, as the true positives and true negative will be equal to the total number of TP, TN, FN, FP since the false positives and false negatives are effectively zero. However, for the second and third classes, the precision and recall are opposite, as the number of false positives (14) for class 1 (which lower the precision) corresponds to the number of false negatives for class 2 (which lowers the recall). As well as, the number of false negatives for class 1 (2) corresponds to the number of true positive for class 2.Therefore, the values of precision and recall will be similar, but opposite for these two classes (lowering their f1 score). Overall, the number of false positives and negatives (16) is relatively low in comparison to the entire 150 total instances, making the accuracy relatively high (0.9) as expected.

However, these results are hard to reproduce, not because the clustering algorithm is incompetent, but due to the fact that clusters have no corresponding class to them. Therefore, the matching of "clusters" and classes is entirely random, and to reproduce these results takes a few reruns of the clustering algorithm. This makes sense, since some of these evaluation methods are better for classification algorithms

To go further, the algorithm used came from the sk-learn library as opposed to my own implementation, as it cluster data points of higher dimensions (¿ 2). Therefore, this doesn't evaluate the validity of my original implementations entirely (even if they're modeled after sklearn's algorithm)

As well as, this doesn't evaluate the strength of the clustering algorithm entirely. While this shows that K-Means is proficient, it doesn't show any comparisons to other clustering algorithms such as FCM or DBSCAN for example. Not only does this affect the results of the Classification Report, but this affects the Confusion Matrix produced, as the mismatch of class appears as False Positives and Negatives.

To go further, while the classification report calculates the individual scores of precision, recall, and f1-score for classes, there is no overall score for all of them together.

References

[1] Aktaş, Yağmur Çiğdem. *Image Segmentation with Clustering*. en. Sept. 2021. URL: https://towardsdatascience.com/image-segmentation-with-clustering-b4bbc98f2ee6 (visited on 05/01/2022).

[2] C3AI. "Clustering". In: (). URL: https://c3.ai/glossary/data-science/clustering/.

[3] Jordan, Jeremy. *Evaluating image segmentation models.* en. May 2018. URL: https://www.jeremyjordan.me/evaluating-image-segmentation-models/ (visited on 04/29/2022).

[4] Khushijain. *K-Means Clustering: Python Implementation from Scratch*. en. July 2021. URL: https://medium.com/nerd-for-tech/k-means-python-implementation-from-scratch-8400f30b8e5c (visited on 05/01/2022).

[5] sk-learn (last). *7.1. Toy datasets*. en. URL: https://scikit-learn/stable/datasets/toy_dataset.html (visited on 05/01/2022).

[6] sk-learn (last). *sklearn.cluster.KMeans*. en. URL: https://scikit-learn/stable/modules/generated/sklearn.cluster.KMeans.html (visited on 05/01/2022).

[7] Mittal, Himanshu et al. "A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets". en. In: *Multimedia Tools and Applications* (Feb. 2021). ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-021-10594-9. URL: http://link.springer.com/10.1007/s11042-021-10594-9 (visited on 04/26/2022).

[8] Roy, Payel et al. "Image Segmentation Using Rough Set Theory: A Review". In: *International Journal of Rough Sets and Data Analysis* 1 (Sept. 2014), pp. 62–74. DOI: 10.4018/ijrsda.2014070105.

[9] The Academician. *48. Fuzzy C Means (FCM) using simple example and Python*. May 2020. URL: https://www.youtube.com/watch?v=FA-hJBu5Bkc (visited on 05/01/2022).