Implementing and Evaluating K-Means Clustering

Christopher Linscott
clinscott@oxy.edu
Occidental College

1 Abstract

The following report dives into an approach to clustering images together: based on purely pixels. After performing both Fuzzy C-Means and K-Means on select color images, it was found that these methods were not useful in creating meaningful clusters, given only pixels as data points.

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" [6]. 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, 7, 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 datapoint 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: $D_{i,j} = \sum_{l=1}^{d} \sqrt{|x_{il}^2 - x_{jl}^2|}$. In my own im-

plementation (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.

2.3.1 Pseudocode

Algorithm 1 K-Means: Lloyd's Algorithm

```
Initialize k clusters to be at initial positions of (0, ..., 0).
Select k number of initial centers (centroids) from datapoints (at random)
while Centroid positions don't change do
    for n datapoints do
        for k clusters do
Calculate distance from data point n to cluster k

        end for
Store closest cluster for each data point n
    end for
New centroid position = 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 datapoint 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) (Source)), applicable to multichannel data (i.e. colored images), and has the ability to model uncertainty in the data (Source). 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, the data points now have an additional set of k membership values, k being the number of clusters. FCM utilizes membership values of the data points, where every datapoint 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 datapoint, and tweaked depending on the distance metric from a datapoint 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 datapoint will be most affected by data points which have a membership high for that cluster. Mathematically, this is all represented by first calculating the centroid positions by computing

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

with n datapoints dp, a cluster j and a given membership value of a datapoint 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 datapoint i to cluster j, divided by the total euclidean distance to all clusters from the same datapoint. This is calculated as follows:

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

. In psuedocode, we'd have something like:

2.3.2 Pseudocode

Algorithm 2 Fuzzy C-Means

```

$f \leftarrow 2$ Initialize k clusters to be at initial positions of (0, ..., 0).
Select k number of initial centers (centroids) from datapoints (at random)
while Centroid positions don't change do
    for n datapoints do
        for k clusters do
Calculate distance from data point n to cluster k

        end for
Store closest cluster for each data point n
    end for
New centroid position = mean of all subsequent data point positions
end while
```

3 Evaluation

Evaluation of these clustering algorithms were performed using "simple" images, or images with objects of few, distinct colors, versus more complex images where the colors of objects often overlapped. The first proposed method for comparing K-Means and Fuzzy C-Means is utilizing a confusion matrix and computing accuracy, precision, and recall to compare on select images. A secondary method of utilizing Intersection of Union (IoU), which is simply the "ratio of the number of common pixels between X and Y to the total number of pixels in X and Y" (Mittal), mathematically calculated by

$$\frac{|X \cap Y|}{|X| + |Y|}$$

To test and obtain these values, we utilized the BLANK dataset as a ground truth to compare the algorithms with each other and the ground truth, the "ground truth" being human-segmented images.

4 Discussion

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] 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.

[7] 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).