University of Tennessee

Clustering Algorithms and Portable Pixel Map Images

Kirolos Shahat

kshahat@vols.utk.edu

ECE 471: Pattern Recognition

Dr. Hairong Qi

04/12/18

Abstract

There are numerous images online that are often sent across servers whether through social media or through text message. Being able to compress these images while still preserving the visual information that they contain would help speed up transmission rates and lower wait times from either side. Thus, this inspired the study of clustering algorithms. Four were studied: K-Means, Winner-Take-All, Kohonen-Maps, and Mean-Shift to see optimal solutions and error rates for different parameters.

The result was that K-Means had the best performance and lowest error rate of all of the algorithms for all of the tested values. The error rates of the three classifiers which had a cluster number as a parameter was quadratic in error decrease rate while Mean-Shift had a linear error decrease rate with respect to bandwidth. There were issues with convergence for Winner-Take-All and Kohonen-Maps for low cluster values but they performed well for higher cluster values. Thus, it seems that they should not be used for low cluster values but more studies need to be done in this case to see if this problem was a potential outlier of performance.

Introduction

Images are carry crucial information which can often be memory intensive. To remedy this, clustering algorithms were studied in order to compress the images in a smaller set of colors so that it is possible to store less colors and convey the same amount of information through visualization. This could be quite useful for sending information over servers in a more efficient manner or potentially for storage constraints on systems. Thus, it has very practical applications in the study of compression.

There were four clustering algorithms which were used on this data set: K-Means, Winner-Take-All, Kohonen-Maps, and Mean-Shift. For the cases of K-Means, Winner-Take-All, and Kohonen-Maps we tested different cluster values to see how recognizable an image is based off of different cluster values as well as testing different bandwidth values for the case of Mean-Shift and compared them using error metrics based on a distance metric. We will analyze further the methods used in the technical approach to come.

Technical Approach

Our goal was to compress the given image given different cluster sizes and, for the case of Mean-Shift, different bandwidth values. The cluster sizes tested were: 2, 8, 16, 32, 64, 128, and 256 and the bandwidth values tested were: 0.1, 0.5, 1, 2, 3, 6, 10, 20, 30, 50, and 70. All of the code and results were implemented in python 3.6.5 The first clustering algorithm which we implemented was K-Means. The algorithm works as follows:

1. Initialize all of the clusters randomly, which we did using a uniform distribution.
2. Initialize all of the samples to be classified as a single cluster.
3. Iterate through all of the samples and compute the distance to each cluster and classify that sample to the cluster with the minimum distance based on a distance metric. The distance metric we used was Euclidean Distance given by:
   a. $\textit{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(a[i]-b[i])^2}$
4. Then, for all cluster centers, update their mean to be the mean of all the pixels that mapped to that cluster and repeat step three. However, if no pixels changed classes then that will be the cluster centers that we currently have.

The winner-Take-All and Kohonen-Maps algorithms varied only slightly to K-Means. For Winner-Take-All the following was added:

1. The initialization step was altered to initialize all distances to infinity prior to the first round of distance calculations.
2. For each pixel, when comparing all of the clusters and getting the distance we find the minimum cluster distance in all of the cluster points and if the pixel changes classes then we update it based off the equation:
    a. $Cluster[winner] = Cluster[winner] + \varepsilon * (X - Cluster[winner])$
        i. Where Cluster[winner] is the new cluster with the shortest distance, ε is the learning parameter, and X is the pixel sample in question. For ε we used 0.1 as our learning parameter.

And Kohonen-Maps had only one change from Winner-Take-All where it is as follows:

1. The cluster update formula is given as follows
    - For i = 1, …, n:

$$Cluster[i] = Cluster[i] + \Phi(k) * \varepsilon * (X - Cluster[i])$$

Where $\Phi(k)$ is a weighted update function which returns 1 if cluster[i] is the winning cluster center and otherwise returns a value strictly between 0 and 1. And finally, the last algorithm we used was Mean-Shift which is based on a kernel function. For Mean-Shift we were able to use the SciKit-Learn library in order to get the clusters based on a given bandwidth. The way we measure error is using Mean Squared Error. We will now begin analyzing the results of this experiment.

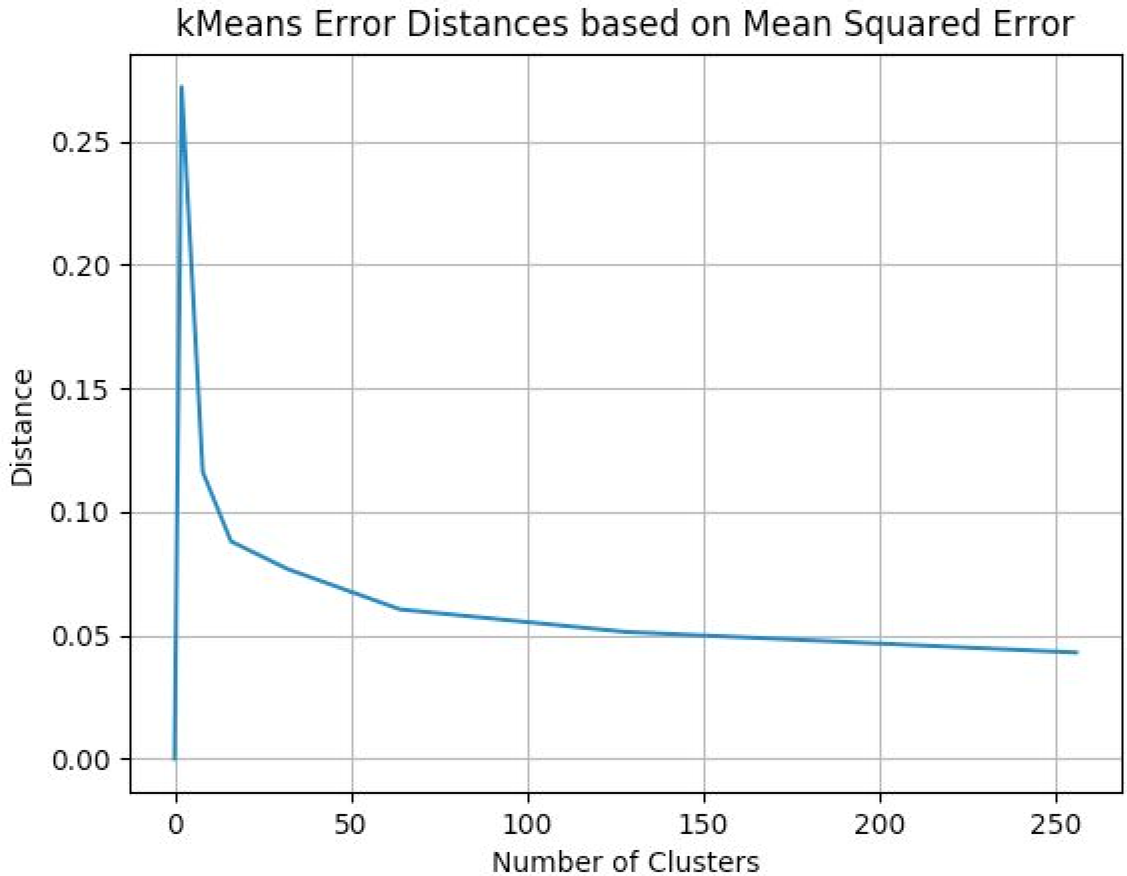
Results

First, we will analyze the results of K-Means clustering. K-Means seemed to be the baseline that the other clustering algorithms needed to match up with. K-Means performed very well and even with simply having a K value of 2 the convergence was very quick as well as accurate. The image still contained a lot of information as shown below:

Clusters	2	16	32	64	128	256
Image						

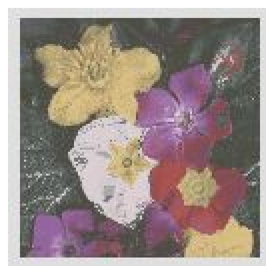
Even at simply having 16 color choices the information needed is still there and thus conserves on so many color values which is an astounding result. The error, as one would expect, peeks at

2 and then drops as more clusters are added and is 0 for the original image as shown below:

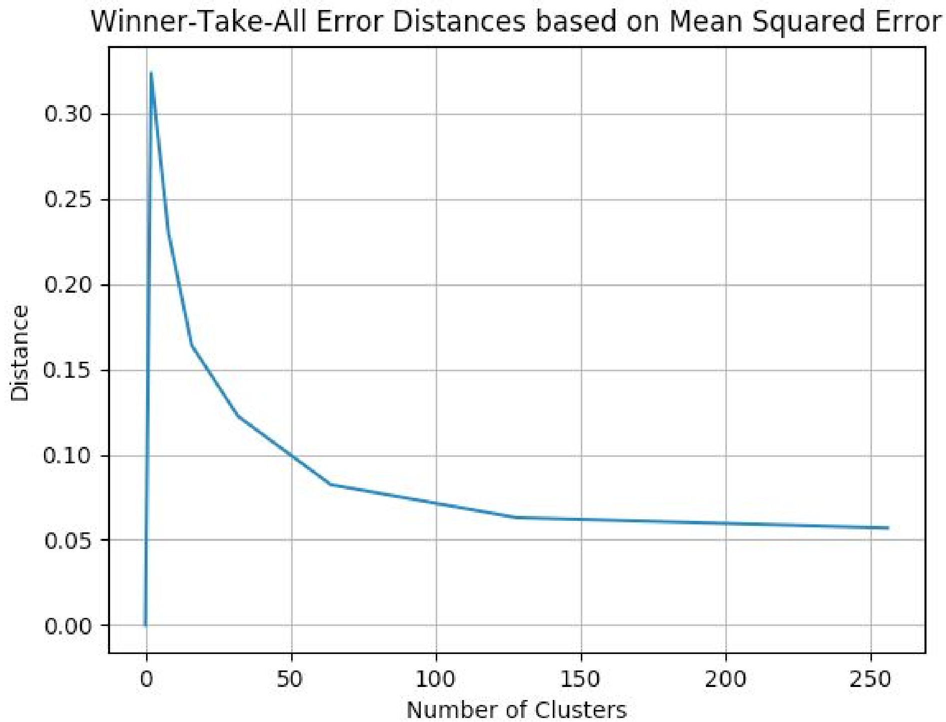


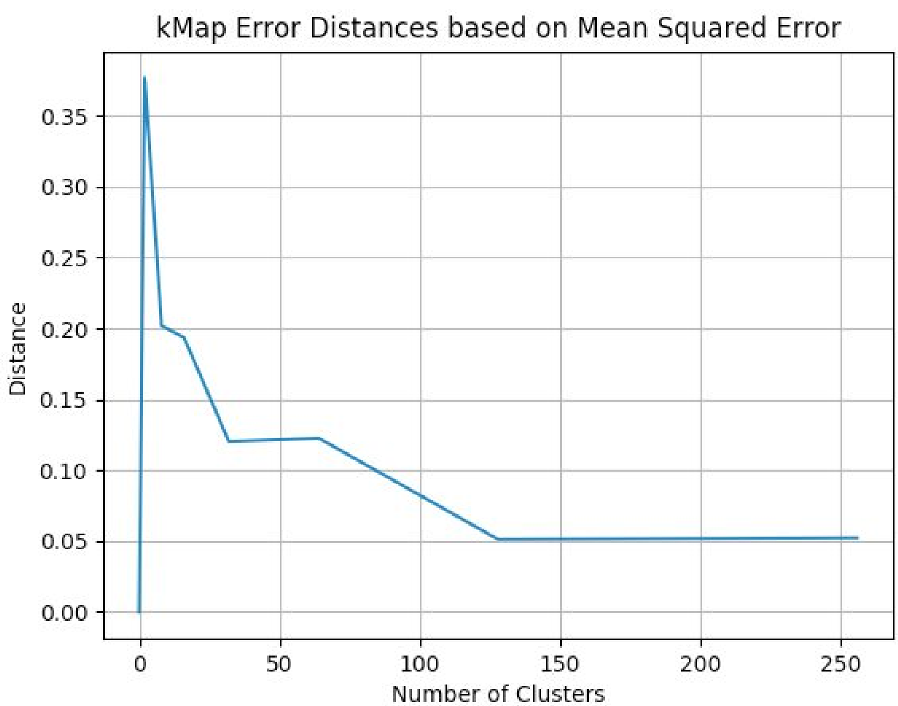
The results of Winner-Take-All and Kohonen-Maps were as follows:

K	Winner-Take-All	Kohonen-Maps
2		

16		
32		
64		
128		
256		

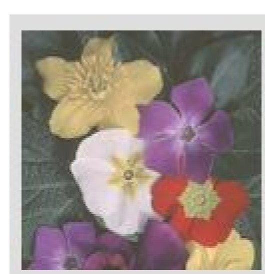
This table shows that these two algorithms took much longer to converge to the full information that K-Means achieved at K=16 which is fairly surprising. The result graphs are as follows:





Which shows that Kohonen Maps has a higher error rate at smaller cluster values but seems to have a better error rate for higher cluster values for this data set.

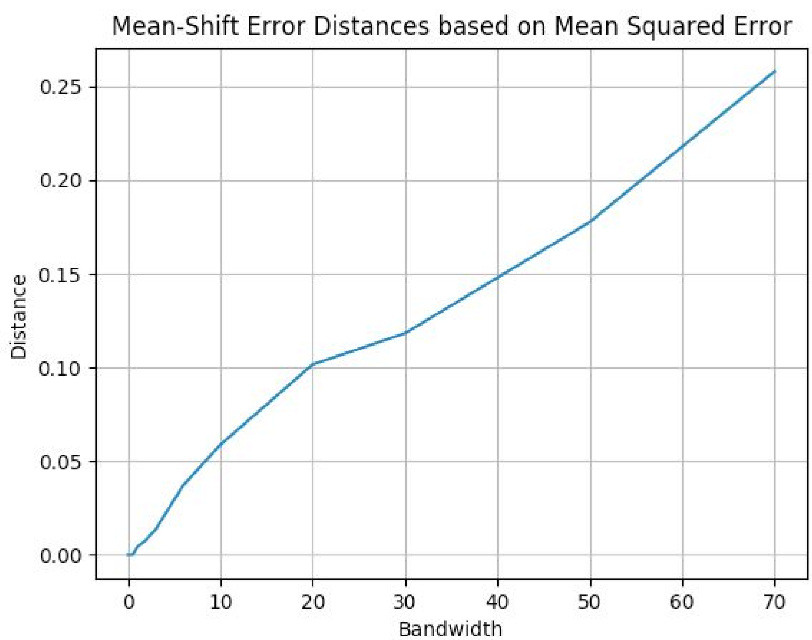
Finally, The results of Mean-Shift is as follows:

Bandwidth	Image
0.1	
0.5	
1	
2	

3	
6	
10	
20	
30	

50	
70	

Which quite clearly shows that for bandwidth higher than 6 the quality begins to drop dramatically and that for a bandwidth equal to 70 the quality is equivalent to only having two colors. The graphed error is as follows:



Which is interestingly linear in error based on the bandwidth. We will now discuss these results in more detail.

Discussion

As shown in the error graphs in the results, K-Means is the fastest one to converge to the best human recognizable image. This is likely due to the fact that there is no online learning in the middle of an epoch, learning is done after each epoch but with Winner-Take-All and Kohonen-Maps they learn within their epochs. Thus, Kohonen-Maps is the slowest to converge because of all of the input that each sample has in each epoch. Thus, it seems not very good for samples to have input other than after total classification because the more input they have within epochs the more noise introduced and the longer it takes to converge, based on this case study. For the case of Mean-Shift the Bandwidth error decrease is linear rather than quadratic in the other three clustering algorithms which shows that the parameter searching would take longer than for the cases of K-Means, Winner-Take-All, and Kohonen Maps. Therefore, based on this study it seems best to use K-Means to find the optimal clustering and if one were to use Mean-Shift the optimal bandwidth would be a value of approximately one.

Conclusion

We tested four methods of clustering: K-Means, Winner-Take-All, Kohonen-Maps, and Mean-Shift. For the first three clustering algorithms, to get the best visual image based on low cluster inputs one would need to use either K-Means or Mean-Shift with a bandwidth equal to approximately one. Kohonen-Maps and Winner-Take-All took the longest to converge to a good image, likely due to the updating that is done online. Moving forward, one would ideally look to optimize the run-time of K-Means or search for better way to update the clusters online for Winner-Take-All or Kohonen-Maps.