Color Image Compression Using Unsupervised Learning

PROJECT 4

Joshua Chavarria | ECE471 | April 24, 2017
jchavarr@vols.utk.edu

Abstract:

This report covers the development and implementation of various unsupervised learning clustering techniques used to compress the colors of a provided image and note trends in picture quality and color display. The motivation for this project is that many display devices allow only a limited number of colors to be displayed simultaneously, meaning that though there can potentially be thousands of colors in one picture, only a certain amount of them are chosen to be displayed for the purpose of saving storage space or display in restricted display devices. The experiments in project 4 contribute directly to the concept of having to convert true color images into indexed color images to be shows on screen. The topics covered later in this report will include methods of unsupervised learning, specifically clustering, and how this kind of pattern recognition can be applied to carry out the task of compressing colors in an image. It was found that with each of the three techniques used in this project, different cluster numbers had very different effects on the provided image. For k-means, the differences between images were subtle compared to using the other two. Comparisons will be provided later in the results section of this paper.

Introduction:

Unsupervised Learning:

Throughout most of the semester we've covered many supervised learning methods. Some pattern recognition systems are trained using one set of data referred to as the training set, and are tested with another set of data typically labeled as the testing set for accuracy. The use of this labeled training data is known as "supervised learning" in which the system is tasked with using this data to make inferences and comparing its results with that of the desired output values. [1] Unsupervised learning, though, is the most applicable subfield of machine learning as it does not require any labels in the dataset. Because the world is full of unlabeled data, unsupervised learning can be implemented in just about anything, from the facial recognition software now implemented in phones to voice recognition, and much more. There are many methods to unsupervised learning but the most common one that will be covered here is known as clustering.

Clustering Background:

As previously mentioned, a basic task of unsupervised learning is to classify a data set into two or more classes based on a similarity measure applied to the data and without referencing any a priori information on how the classifications must be carried out. Any nonuniform data contains an underlying structure due its own heterogeneity and identifying the structure by grouping related elements is known as data classification, or as it relates to this project, clustering. In brief, clustering is an unsupervised machine learning task that automatically divides the data into clusters of similar items, usually defined by some similarity measure. Before covering some of the most popular clustering techniques, it is important to note that clustering is somewhat different from the classification, prediction, and pattern recognition tasks that we have examined throughout the semester. For most of those, the results are typically a model that relates features to an outcome or to other features. Clustering on the other hand creates new dat. Seeing as clustering is referred to as unsupervised classification, unlabeled examples are given a cluster label that has been inferred entirely from the relationships between data. Clustering is a field of study that has seen a lot of activity in the recent years. Originally proposed to aid in the field of archaeology for classifying findings, the applications of clustering are now more often used to aid in the development of pattern recognition software and various other implementations of artificial neural networks. There are now various algorithms used throughout various fields for describing and finding clusters from data.

Unfortunately, because of the several algorithms and their differing applications, there is no single definition for cluster that is considered universal. Some might take into account distance more than similarity of vertices, while others might be more reliant on adjacency than other methods. Although this discrepancy can be a hindrance when trying to determine a good cluster, there are certain desirable cluster properties that are commonly looked for in many of these fields. For example, the field of graph clustering tends to prefer that clusters should be intuitively connected; each cluster should consist of at least one path connecting a pair of vertices within it. It is generally agreed upon that a subset of vertices which together make a cluster denser is preferred over others. Aside from these properties, there exist several common similarity based measures for identifying clusters that should be considered when applying any sort of clustering algorithm. [2] Approaches for identifying any cluster can be classified in two separate approaches. One may either compute some value for the vertices and then classify the vertices into clusters based on the previously obtained values, or one can compute a fitness measure over the possible clusters and choose the clusters that display the most desirable properties for optimizing the measure shown. There are many clustering algorithms based on similarities between its vertices, so the higher the similarity, the more likely those vertices will be compressed into clusters. Computing these similarities are not, in all cases, simple. Instead of similarity, we can also use distance measures, which happens to be related to what we've been doing so far in class. For example, for points in an n-dimensional Euclidean space, possible distance measures for two data points can include Euclidean and Manhattan distance. As was already mentioned, clustering algorithms all have different definitions of what they consider "similar" and how they go about their clustering.

- **Eucledian distance – most frequently used**

$$D(A,B)=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\ldots+(a_n-b_n)^2}$$

- **Manhattan (or city block) distance**

$$D(A,B)=|a_1-b_1|+|a_2-b_2|+\ldots+|a_n-b_n|$$

- **Minkowski distance – generalization of both Eucledian and Manhattan**

$$D(A,B)=(|a_1-b_1|^q+|a_2-b_2|^q+\ldots+|a_n-b_n|^q)^{1/q}$$

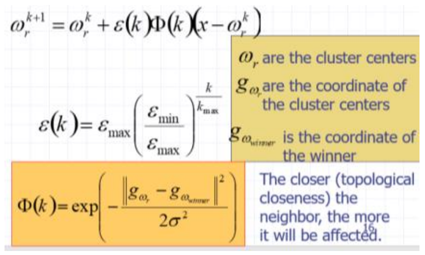
***q* – positive integer**

Clustering Algorithms:

For this project three clustering methods were implemented. The first is what is called **K-means**, which is known to be one of the simplest unsupervised learning algorithms that can solve the clustering problem. The procedure follows a simple way to classify a given data set through a certain number of clusters fixed a-priori. The main idea to behind this method is to define k-centroids, one for each cluster. These individual centroids should be preferably placed as far away as possible from each other. The next step is to take each point belonging to a specific data set and connect it to the closest centroid and continue doing this for all points which will ultimately create new clusters. K-means is finished when clusters are no longer changing and at this point the metric to be minimized can be calculated from the separation of objects into groups. Though very simple, the downsides to K-means reside in its results, which do not always find the most optimal configuration, and its sensitivity to initial randomly selected cluster centers. K-means is also computationally expensive since it involves finding the distance from each example to each cluster center at each iteration. The next algorithm implemented for this experiments is referred to as the "**Winner-Take-All**" approach to clustering, which is an iterative K-means algorithm. Again, most unsupervised neural networks rely on algorithms to compute and compare distances, determine the "winner", and use them to adapt the weights. Like K-means, you begin with an arbitrary set of cluster centers, or centroids. For each sample, you continue to find the nearest cluster center, which in this case will be labeled as the winner. The newest winner is then calculated by adding the old winner cluster with a learning parameter, which usually are small values on the order of 0.01 (shown in the following formula). At the end of each epoch check if the stopping criteria is satisfied. If so, winner-take-all clustering has finished.

$$\omega_\alpha^{new} = \omega_\alpha^{old} + \varepsilon(\mathbf{x} - \omega_\alpha^{old})$$

The last method that was applied to compressing the colors of the image is called **self-organizing feature maps (SOM)** which is an extension of the winner-take-all algorithm. SOMs extend competitive learning in many ways. During learning, not only the weights of the winner are update but also the weight of its neighbors. [3] A neighboring function defines the size and shape of the neighborhood around it. As the learning progresses, both the size of the neighborhood and the learning rate begin to increase. The following formula was pulled from the lecture notes and serves as the basis to how the winning cluster center and its neighbors are trained.



$$\omega_r^{k+1} = \omega_r^k + \varepsilon(k)\Phi(k)(x - \omega_r^k)$$

ω_r are the cluster centers

g_{ω_r} are the coordinate of the cluster centers

$$\varepsilon(k) = \varepsilon_{\max}\left(\frac{\varepsilon_{\min}}{\varepsilon_{\max}}\right)^{\frac{k}{k_{\max}}}$$

$g_{\omega_{winner}}$ is the coordinate of the winner

$$\Phi(k) = \exp\left(-\frac{\left\|g_{\omega_r} - g_{\omega_{winner}}\right\|^2}{2\sigma^2}\right)$$

The closer (topological closeness) the neighbor, the more it will be affected.

16

When comparing both K-means and SOM, they both define a mapping from a higher dimensional space to a lower dimensional space, from the number of examples to the number of clusters. They both use more units to represent regions with the higher density and require that the number of clusters is specified in advance. On one hand, K-means does not learn topology while SOM does. For SOM, similar inputs are mapped to neighbors and topologically close units have similar input vectors mapped on them. On the other hand, for K-means each input example determines the adaptation of 1 winner, whereas SOM adapts the winner and other units as well.

Objectives:

For this project, a 120 x 120 full-color image was provided (flowers.ppm). Each pixel of the image has three components, which are the colors red, green, and blue. Each component is an 8-bit unsigned char and thus there are 2^24 different colors that can be displayed. Certain computers, though, can only display up to 2^8 different colors. Having learned that, the objective of this project is to find the best 2^8 colors that can approximate the original full-color image.

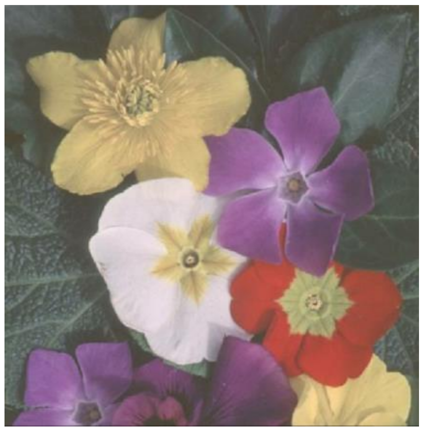
Technical Approach:

The majority of this experiment was carried out via C++ code that I wrote specifically to utilize three functions, one for each of the previously mentioned clustering algorithms. The program implements K-Means, Winner-Take-All, and SOM to develop a transformed image from the original one provided. Other alterations that contributed to the output image of each relied on changing the number of acceptable clusters and specifying the total number of colors. We know that cluster number for the three algorithms need to be previously specified before carrying out the tasks. Once I wrote the functions and tested the program to make sure that it can successfully read in and output.ppm files (using readimage() and writeimage() functions), I tested it extensively. The steps of each algorithm were covered in the introduction of this report, so I simply used that as pseudocode. The first task was to use k-means algorithm to find 256 colors that best represent the original full-color image. The result of this K-means was then compared to the original and some metrics were designed to illustrate the performance gain/loss. The rest of the tasks involved repeating the previous two by implementing winner-take-all and SOM. Though not necessary, I implemented SOM to take advantage of the **extra credit** opportunity. Lastly, I tested each of these with varying cluster numbers and compared their outputs as well. The results of my experiments can be seen in the next section where I will also be discussing the differences and trends that I noticed with each.

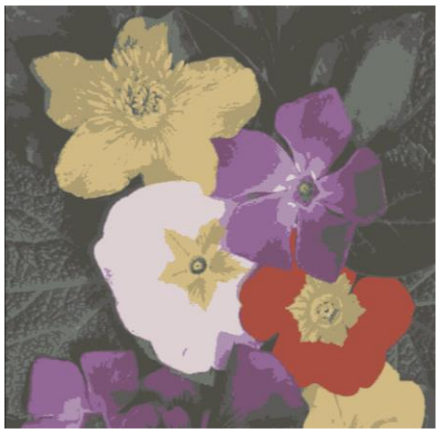
Experiments and Results:

K-MEANS

Original Image



K-means (256 colors)



Cluster # = 128

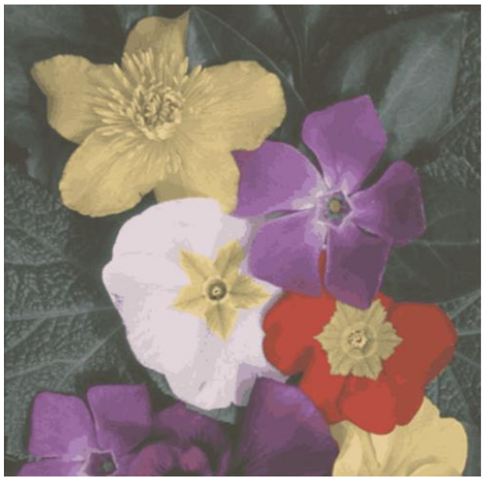
Cluster# = 64

Cluster# = 32

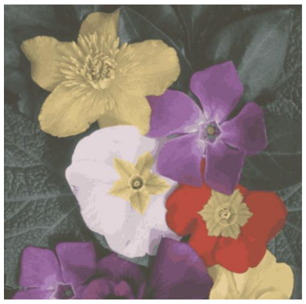
K-means is computationally expensive since it involves finding the distance from each example to each cluster center at each iteration. During my experiment, I noticed this, as many of the above images took quite a while to develop. For lower cluster numbers, K-means obviously finished a little faster but it still took over 100 iterations for any of them to finish. The first task was to represent the original image with only 256 colors with K-means. This resulted in a color image that seemed much flatter than the original. It can be easily seen that many of the flowers no longer have shadows or darker shades to indicate depth. It is also easier to see where the clusters formed and from the color boundaries especially on the purple flower. For the varying cluster number experiments, the results were pretty like the original image. Differences from the images produced by cluster numbers 128, 64, and 32 were very subtle. By looking closely, the image produced by cluster number being 32 is not as smooth as the original when looking at the red flower. Again, most of the results were subtle but I figured that drastic differences would not be easily apparent to the naked eye. More experiments with different values might have produced more defined clusters and boundaries.

WINNER-TAKE-ALL:

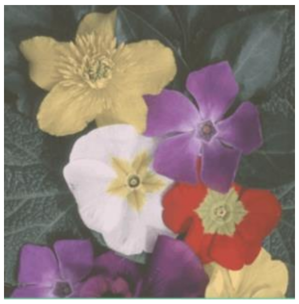
Winner-Take-All (256 Colors)



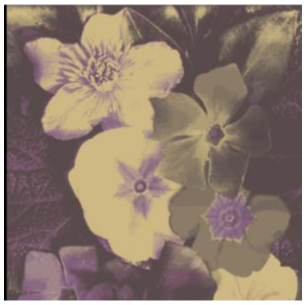
Cluster # = 128



Cluster# = 64



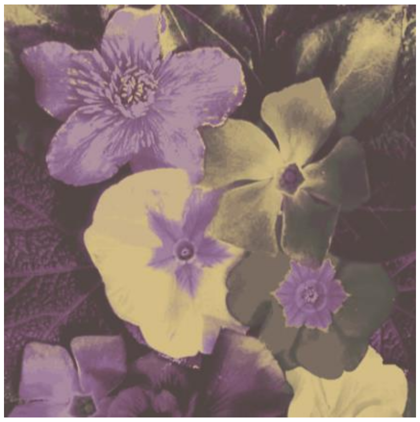
Cluster# = 32



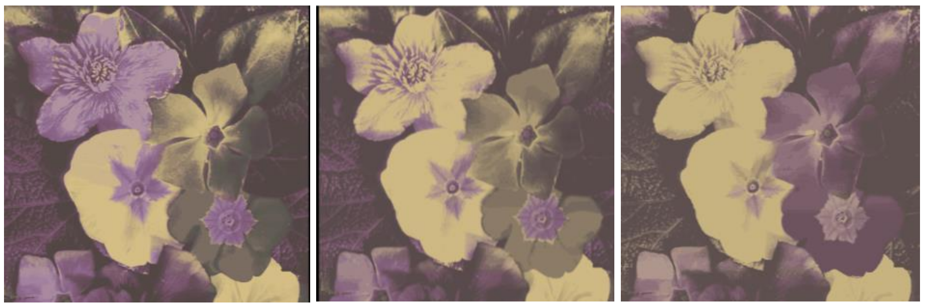
Winner-Take-All can be understood as an advanced version of k-means in which neigbors are updated with the use of a learning curve. The results for this experiment were very similar to the K-means one, especially when using 256 colors. Winner-Take-All was able to accomplish the image color clustering a bit quicker than its counterpart and ultimatelt had almost identical results. As with the previous algorithm, testing the method with cluster numbers 128 and 64 resulted in an image that was flatter in comparison to the original. The one outstanding image was a result of having a cluster number of 32. For some reason, this image came out unlike the rest. Referring to the above image, the colors seemed to have been clustered into different shades of yellow and purple. This might be due to many factors including learning rate or simply he effects of cluster number within the system. As will be seen next, this image was closely related to the output I got from implementing Kohonen maps, making the results of Winner-Take-All reside in an area between K-means and SOM. Another reason might be that the clustering didn't have enough time to optimally complete an image representing the colors. Having a cluster number of 32 and a learning rate set at .01 might have not been enough to display more colors and instead chose to primary ones. Another interesting thing I noted about the last image, was that the two colors are practically contrasts to eachother. The centers of the white and red flowers were turned purple while most everything else turned into a hue of yellow. At this point I can only keep speculating as to why I got this result, but for the most part I agree with the other images Winner-Take-All produced. Compared to the exmaples Dr.Qi provided, I would say that they seem pretty accurate and tell me that my implementation of Winner-Take-All is mostly correct.

KOHONEN MAPS (SOM):

SOM (256 Colors)

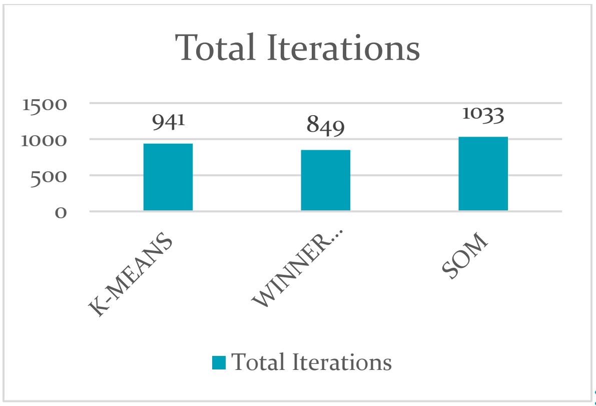


Cluster # = 128 Cluster# = 64 Cluster# = 32



My SOM implementation, done for **extra credit,** had some very interesting results on the provided image. Compared to the original, the image for cluster numbers 128, 64, and 32 seemed to primarily display two colors. It can be easily inferred from the above images that the colors purple and yellow make up most of the clusters in each image. As stated earlier, Kohonen maps can be thought of as an extension of the previous two clustering methods. When experimenting with this method, I had to make sure to test the program out with different learning rates. The most interesting results came from using a learning rate of around .1. Running SOP on 128 clusters resulted in what seemed to be a reversal of images. Compared to the original, the yellow flower in this image has been converted to purple. The next two images produced were more consistent in that the light flowers of the original image were output as a light yellow, whereas the flowers of opposite contrast, specifically the red and purple, were depicted in a darker shade of yellow with certain clusters of purple at the center of each flower. The last image is similar in that the light colors are depicted as a light yellow, but different in that the darker colors are purple with yellow centers. The last image was very similar to one of that resulted from the Winner-Take-All clustering algorithm tested previously. Clearly, this method has the most drastic impact on the clustering of colors. The similarity measure used topological distance of neighbors, and in this case, the closest neighbors of the winning clusters were merged together to form a more optimal one. Any discrepancies with the results of my SOM implementation may be due to learning rate, but my program accurately applies the steps of SOM and implements them accordingly to the original image. Overall I believe that the reason for these outputs is that Kohonen maps in general tend not to worry as much about accuracy instead of topological structure in their clustering.

VISUAL METRIC:



The above chart simply shows the amount of iterations that each clustering method had overall when displaying an image of 256 colors, as specified in Task 2 of the project outline. Winner-Take-All seemed to accomplish the task quicker, while producing the same results as K-means. SOM took a bit longer but that is probably because it is frantically searching or topological structure within the full color image.

Conclusion:

. I feel confident in my results for this project. When comparing the images using the provided test image code of Dr.Qi, it seems that my imaged were in fact pretty accurate representations of the overall effects of K-means, Winner-Take-All, and SOM on the original images, as well as the change in cluster numbers. This was the most enjoyable project we have had all semester and felt challenged in designing an unsupervised learning machine learning system. This project further opened my eyes to the applications of unsupervised learning that I had never noticed in the displays that surround us every day. Being able to, in a way, visualize these clustering algorithms and noticing cluster and boundaries of color was what I found most interesting. As for any possible discrepancies, that could be due to factors of normalization or simple that different values would have provided better results.

References

[1] American Journal Of Intelligent Systems 201, Vinita Dutt, Vikas Chaudhry, and Imran Khan. "Pattern Recognition: An Overview." *American Journal of Intelligent Systems 2012* (2012): n. pag. Bhagwant University. Web.

[2] http://www.inf.ed.ac.uk/teaching/courses/pmr/docs/ul.pdf

[3] http://www.lsi.upc.edu/~bejar/amlt/articulos/Graph%20Clustering03.pdf