Shaun Howard 9/9/2015
EECS 484 PS2 smh150

PS2: Garment Clustering

Code Additions:

The files I created or edited are the following:

Kmeans_clustering.m

I turned this into a function that returns the success_rate. I changed the number of clusters and passes of clustering. I removed some print-outs to console. I added a round to cluster_attributes(closestClust) in the loop to test for consistency because attribute values were appearing as floating point numbers or integers with a decimal and several zeroes. This helped round them to the nearest integer to compare with the actual shirt attribute values.

Find_closest_cluster.m

I messed around with the distance metric used to find the closest cluster. I tried the Euclidean, Chebychev, Minkowski, and CityBlock metrics.

Test_using_clusters_incomplete.m

I renamed the file to validate_clusters.m. I then turned it into a function that returns the success_rate. I added the ability to find the closest cluster when testing for consistency. Now the program will find the attribute of the closest cluster to each pattern in order to determine if it was classified within a correct cluster.

Calculate_average_success_rate.m

I created this script to calculate the average success rate for kmeans_clustering and validate_clusters after 100 iterations. After each iteration of kmeans_clustering, it will clear the function kmeans_clustering. It will also do this for validate_clusters. These values are printed out to the console near the end of the program's operation.

Experiments:

I tested the success rate of various numbers of clusters. I tested each combination with 100 trials and 10 passes, recording the average success rates per 100 trials. My results follow:

Distance Metric	Data Type	7 Clusters	40 Clusters	200 Clusters
Euclidean	Training	.3528	.9486	1
	Validation	.28	.9307	1

I really tried to test clustering with more distance metrics like Chebychev or Minkowski or even CityBlock, but they were too slow for me to even sit through a few iterations of. Hence, I managed to run 2 training and validation trials for each with varying amounts of clusters and 10 passes, just in order to demonstrate how they compare from a quick glance. My results follow:

Distance Metric	Data Type	7 Clusters	40 Clusters	200 Clusters
Chebychev	Training	.2381	.9372	1
	Validation	.238	.6363	.714286
Minkowski	Training	.3571	.9156	1
	Validation	.362	.857	1
CityBlock	Training	.4048	.9264	1
	Validation	.238	.632	.701

I experimented more with the number of passes in my next evaluation. I kept a steady 40 clusters and 100 trials and stuck with the Euclidean distance. My results follow:

Distance Metric	Data Type	5 Passes	10 Passes	40 Passes	100 Passes
Euclidean	Training	.9458	.9486	0.9530	0.9461
	Validation	.9134	.9307	0.93939	0.987013

Discussion:

A few good observations can be derived from the distance metric and clustering experimentation results. One observation is that the Minkowski distance worked the best on the fewest amount of clusters. When the number of clusters grew, however, all of the distance metrics besides Euclidean had divergent training and validation data success rates. I thought, initially when testing with the Euclidean distance, that the clustering algorithm was too good to be true. Now, I am seeing that maybe the distance metric does play an important role in the clustering and classification of patterns. It could have been the amount of trials and randomness in cluster seeds that threw off the results, but it still seems like the Euclidean distance is by far the winner for this type of problem.

In terms of performance, I noticed that the distance metrics besides Euclidean distance were faster for fewer amounts of clusters, but slower on larger amounts of clusters. The Euclidean distance, however, was fast no matter the amount of clusters thrown at it. This was another observation I made while testing the different distance metrics. The function I used for calculating the distances (besides Euclidean) was the pdist2() function, which finds the pairwise distance of the two input vectors. This is fed the two vectors and the distance metric type to use.

I believe maybe the pairwise distance function was not designed to be used with vectors, and hence, why it is slower. However, in the case of clustering, it seems that such a function would basically be useless if it is not optimized for vector/matrix operations. Hence, it may have been an unfair comparison, but Euclidean distance still prevails.

I would like to think of these classification algorithms in the way humans think. If humans prefer to use something that is quick and gets the job done, then the computer program might also use that method. Usually, by applying iterations to simple algorithms like the Euclidean distance, a computer can produce a very similar result to that of a human with true brainpower. Sometimes the simplest ideas provide astonishingly satisfying results. The provided solution for classifying shirts with Euclidean distance has a fairly accurate classification success rate and should be fine to use on a new prototype robot which sorts laundry.

In terms of goodness of classification, we can see that increasing both the number of clusters and the number of passes strengthened our chances of properly classifying laundry by color. This was true for both the training and validation phases. As we can notice, however, the training data performed poorer on 100 passes than it did on 40 passes, yet validation data performed much better. There seems to be a trend in these results. It seems as if the training data success rate is staying around an average value (~.95) while the validation success rate is increasing. It seems as if validation data should have a 100 percent success rate if the number of passes was increased over a certain threshold greater than 100.

I am unsure of whether this is a good trait or not for a clustering program to have. One downside of increasing values is that the computer is doing more work, and hence, using more resources. This can be very problematic on mobile devices or even robotic computer systems that must maintain a certain power usage to remain efficient and practical to use. I think that for some problems in which an estimate solution is suitable, then the simplest and quickest approaches should be taken to solve that problem. Considering this during design would benefit both the designers and users because the system will be reliable, consistent, and easier to maintain.

Involving the distance metrics, there is no comparison in terms of efficiency to the Euclidean distance. In terms of tweaked values, an acceptable minimum threshold for the number of clusters is around 40 and for the number of passes is around 10. Either of these values could be increased to promote a higher success rate, but this would cause performance drawbacks in a larger system. However, for this problem, setting the number of clusters to the number of patterns leads to a 100 percent success rate. Even setting the number of clusters to a value approaching the limit of the number of patterns available will lead to a near 100 percent accuracy for this program. As I tested, 200 clusters with 10 passes led to 100 percent success rate. This tells me that perhaps the data set that we are analyzing is too small to determine if this clustering method really works well or that clustering is perfectly applied to this problem. I still have to decide which one to believe most, but after analyzing the distance metrics, I believe that this domain may just be a great application for clustering.