Computer Vision Homework 4 Writeup

Lucas Laird

3. April 2018

1 Clustering Algorithms and Features

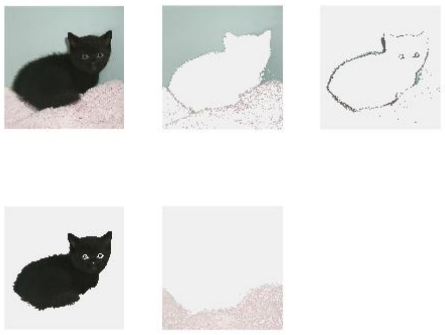
K means clustering: K means clustering is a simple and straightforward clustering algorithm that seeks to clump clusters together around k different centroids. The centroids are initially chosen at random among all of the data points and then iteratively grown by finding a point closest to a cluster and adding it into that cluster. The benefits are this algorithm is quick and easy. The problem is that it creates circular, contiguous clusters in the features chosen which is not always the best option.

Hierarchical agglomerative clustering: Hierarchical agglomerative clustering(HAC) works by considering a large number of pre-selected clusters and tries to merge them to lower the number of clusters. Typically, the HAC algorithm starts by considering every single data point its own cluster and then merges, but in principle this could work if there were already chosen clusters. To merge clusters, we find the two clusters with the lowest pairwise distance(euclidean, etc.) between their centroids and combine the two. We choose to eliminate the cluster with the larger index/label by convention and move all of the points into the one with the smaller index/name.

Feature Vectors: Choosing feature vectors is a decision made outside of the clustering. Since the clustering relies on things like distance to determine similarity between points, the vectors need to have meaning with that regard. Further, each feature should be normalized so that all of them have equal weight in the distance metric to avoid one particular feature meaning more. Normalizing a feature can be done the same way one would normalize statistical data, simply subtract the mean of the feature over the data set and divide by the standard deviation.
For instance, a good feature vector for a clustering algorithm on an image would be to use the rgb pixel values as well as the position in the image since it is likely that things close together and similar in color are of the same cluster. rgb pixel values however might be larger than the size of the image or vice-versa in which case the color or the distance would get unequal weight on the clustering, so we normalize them. A bad feature vector might be simply position data for an image since the clustering would just cluster over the image uniformly with no regard for the actual image being clustered.

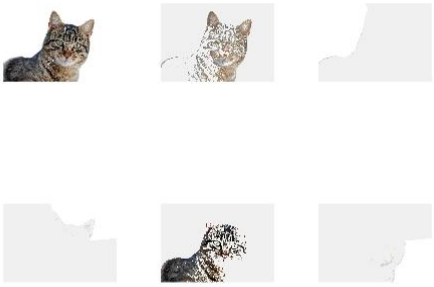
2 Clustering Examples

Successful Clustering



kmeans, 5 clusters, Color only, Normalized

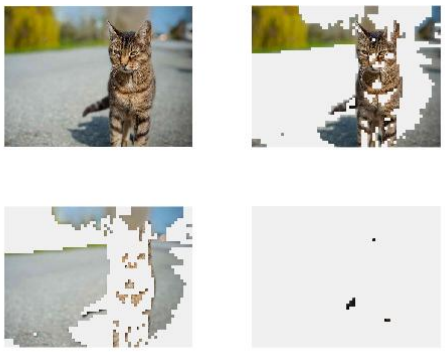
hac, 5 clusters, color + Position, Normalized



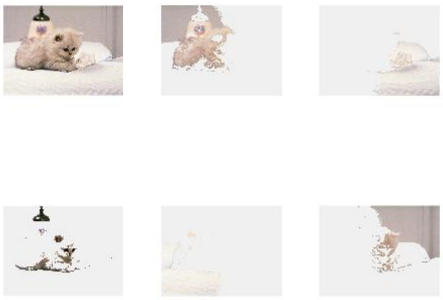
kmeans,5 clusters, color + Position, Normalized

These three were successful primarily because of the image itself, not the clustering method or data. Since the cats are very distinct from the background and in the black cats cases, almost entirely the same color, they are easy to cluster.

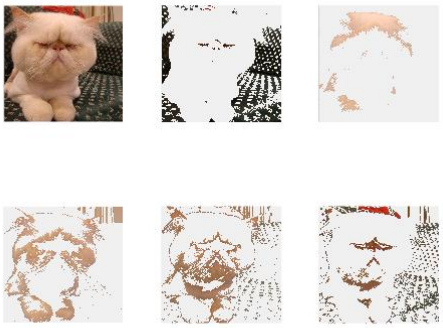
Unsuccessful clustering



hac,3 clusters, color only, Normalized



kmeans,5 clusters, color + Position, Normalized



kmeans,5 clusters, color only, Normalized

These three examples were unsuccessful. The first image uses HAC clustering with 3 clusters and no position data. The second image uses kmeans, both position and color, and 5 clusters. The last one uses kmeans with 5 clusters but has no position data. The unsuccessful images are more interesting than the successful. The first image was unsuccessful despite the cat being fairly distinct from the background colors because position data was not taken into account causing parts of the cat to be clustered with the trees. The second image was unsuccessful because the cat is very similar in color to the background. Maybe a different feature vector or transformation would help distinguish the cat. The last image was unsuccessful because the cat has several parts of its face that are quite red, causing them to be lumped together with the background. Otherwise the clustering would have been successful.

3 Composite Images

Kmeans, 5 clusters, Color + Position, Normalized

hac,10 clusters, Color+ Position, Normalized

As is shown in the images, the clustering works best with cats whose colors are more distinct from the background. The left image was a very good separation and the black cat on the right image was also very good. The white cat however was split into many separate parts that included white background.

4 Accuracy

Clustering Method	k	Position?	Color?	Normalize?	Accuracy	MaxPixels
kmeans	5	Y	Y	Y	0.8818	50000
kmeans	11	Y	Y	Y	0.9154	50000
hac	5	Y	Y	Y	0.8622	3000
hac	11	Y	Y	Y	0.9099	3000
kmeans	5	N	Y	Y	0.8945	50000
hac	5	N	Y	Y	0.8653	3000
kmeans	5	Y	Y	N	0.9307	50000
hac	5	Y	Y	N	0.8667	3000
kmeans	5	N	Y	N	0.931	50000
hac	5	N	Y	N	0.8666	3000
kmeans	5	Y	Y	Y	0.9149	3000

Impacts of parameters on clustering performance
It seems like kmeans clustering performs better across the board than HAC. I think this is because the limitation of connected clusters is not an issue in this case as the cats are all connected in real life.

More clusters translates into better accuracy for both kmeans and HAC which is expected since more clusters means more granularity in the final clustering.
Position in the image was a red-herring it seems as both HAC and kmeans performed better without it, both normalized and not.
Normalizing also seemed largely unimportant as both kmeans and hac performed similarly although slightly better without normalizing.
Pixel count didn't lower accuracy as I thought it would, instead it increased it for kmeans. This could be because of the filtering of noise within the images. Some cats have very complex fur which lowering pixel count would help remove that problem.

Conclusion
The parameters impacts on performance are discussed above but their impacts on computation can be important as well. Lowering the maximum pixel amount drastically reduces the computation time as both of the algorithms runtimes increase with the number of data points. The impact is much more pronounced for HAC than for kmeans though. Changing the cluster amount is a small runtime increase for kmeans but is actually the opposite for hac. This is because kmeans builds up the k clusters whereas hac builds down to them. Changing the features and the normalization has a negligible impact on computation time.
The performance of the algorithm also heavily depends on the image itself. Combining the information from the Clustering Examples in section 2 and the data here, we can see that the properties of kmeans are appropriate for clustering connected objects like cats. The features chosen however have a big impact on performance here. If the picture of the cat has a background very similar to the color of the cat, neither of the algorithms will do well in clustering the cat out. The other issue to consider is that some of the cats have extremely diverse colors on the fur coats. In this case, not using position data can lead to parts of the cat being combined with the background.
The properties of a good image are primarily: the cat is of a similar color throughout and that uniform color is very different from the background.