Laboratory exercise 6: Segmentation

Santiago Enrique Cortés
Bárbara & Frick
Av. Calle 82 # 10 18A - 12, Bogotá, Colombia
santiago.cortes@barbara.net.co

Juan Felipe Cerón Uribe
Universidad de los Andes
Cra 1 # 18A - 12, Bogotá, Colombia
jf.ceron10@uniandes.edu.co

1 Introduction

In this exercise, we experimented with several algorithms that seek to divide an image into relevant semantic segments. We compared the performance of our algorithms to a given *true* segmentation via the mutual information score and discussed the tuning and results of each of the algorithms.

2 Dataset

The data used was a subset of the Berkeley segmentation Data set and Benchmark. IT consists in 12000 hand-labeled segmentations of 1000 corel dataset images from 30 persons. Half the segmentations were obtained from presenting the subject with a color image; the other half from presenting a grey scale representation.

3 Methodolgy

We applied different clustering algorithms to segment the pixels in each picture. We experimented with different color spaces (RGB, HSV and L*a*b) as features for each pixel. Finally, we experimented with augmenting the features of each pixel by its coordinates in the image, which favors the formation of spatially cohesive segments. We now describe each of the applied algorithms.

3.1 Gaussian mixture model

In this algorithm, we assume each of the points to have been picked from one of k Gaussian distributions (the i-th) with probability p_i, where $\sum_k p_i = 1$. Such a distribution can be described by:

$$P(X \in A) = \sum_{i=1}^{k} p_i P_{norm(\mu_i,\sigma_i)}(X \in A | class = i). \quad (1)$$

We then find the parameters μ_i, σ_i and p_i that maximize the likelihood of the observed set of points. Each Gaussian represents a segment of the image. Lastly, we assign each point to the Gaussian which it is most likely to have come from given the fitted model.

3.2 k-means

The standard heuristic algorithm to the NP-complex problem of k-means consists of an initialization, followed by the iterated repetition of an assignment and an update step until convergence is reached:

- **Initialization:** k different points in $\mathbb{R}^d$ (if the data is d-dimensional) are chosen at random. They are called centroids.
- **Assignment:** Each point in the data is assigned to the *cluster* of its nearest centroid. Ties should be rare.
- **Update:** The centroid of each cluster is recalculated from its points.

3.3 Hierarchical (agglomerative) clustering

The agglomerative approximation was the use of a hierarchical clustering using the average linkage between clusters. That is, the definition of the distance between sets is the average of all the distances between a pair of elements each one blogging to a different cluster. In order to reduce the computational time a k-means with 15 clusters was made over the image before, then changing each pixel for it respective centroid the agglomeration was executed.

3.4 Watershed

This algorithm is better explained by a flooding analogy. For this purpose, we conceive a gray-scale image as the topological surface where each point's height represents that pixel's intensity. The algorithm proceeds by:

1. Make a hole in several regional minima.
2. Gradually submerge the surface in water.
3. Whenever two lakes meet, construct a damn to prevent them from mixing. The final set of lakes are considered the segments of the image.

Following a recommendation from a scikit-image example [1], we carried this process out on a gradient representation of each image. We obtained this representation by averaging the gradient representation of the images in each of the channels. Markers (where we poke holes in the surface) were chosen by taking connected sets of low gradient pixels. We did not take into account spatial dimensions for this algorithm, as they (and their gradients) draw trivial topological surfaces.

4 Preprocessing

The first pre-segmentation process practiced in the images was a normalization of the pixels' features. Specifically, we divided each feature by its maximum value. Otherwise, distance metrics would be dominated by features with the largest ranges.

For the agglomerative algorithm, it was necessary to first re-scale the images. Typically, these algorithms begin by calculating an inter-point distance between each pair of points. Because the points in this case are each of an image's pixels, the inter-point distance matrix (of size *#pixels*^{2}) did not fit in memory, and when it did it was too slow to calculate. To cope with this problem, we first applied k-means clustering to the images with large k, and then applied hierarchical clustering over the set of centroids. This can be seen as a way of re-scaling the image down to k pixels.

5 Parameter tuning

One tuning parameter, which is ubiquitous across almost all clustering algorithms, is the number k of clusters to be consolidated. We now present a few ways in which we might choose it.

1. We might use domain knowledge of how many objects are present in the dataset's images to fix this parameter.

2. If we have an annotated training set, we could obtain a list of the number of segments in each of the images. If the numbers have a low standard deviation, we could fix k as the integer closest to their mean. We could also cluster the images in search for a clustering in which images in one cluster have a similar number of segments. Then, for a test image, we could associate it with one of those clusters, and use the cluster's mean number of segments as parameter k for the segmentation of the image.

3. There is a famous way to pick k in the k-means algorithm. It consists of graphing out the internal sum of square errors between points and their clusters' centroids (after performing the algorithm) against the parameter k, as shown in figure 1. When the sum of square errors ceases to decrease fast, this is a sign that we are near the appropriate k. We can use this method for any clustering algorithm, since we can calculate centroids regardless of how we grouped the points.

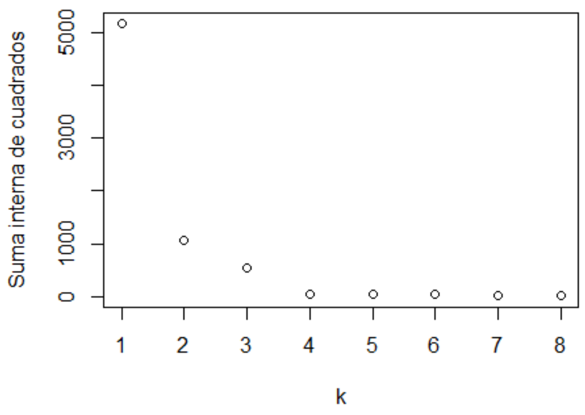

Figure 1. Graph-based approach to selecting k in the k-means algorithm.

4. Drawing inspiration from the previous method, we could perform the clustering algorithm for a long sequence of consecutive k, and measure (and graph out) the mutual information between consecutive clusters. When this distance metric begins to become very small between consecutive k, we can take this as a sign that a clustering with more clusters is simply catching a new, small, irrelevant cluster, and thus we are near a good k.

6 Evaluation methodology

We used the mutual information metric to measure the difference between the true segmentation and the one resulting from each of the algorithms. For two clusterings $U = \{U_i\}$ and $V = \{V_i\}$, it can be defined by

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_i|}. \qquad (2)$$

Roughly speaking, this is a measure of the amount of information given by knowing the cluster of a point in one of the clusterings about its cluster in the other. This has become a popular distance metric between clusterings both because it is constant under label permutations (like swapping clusters 1 and 2) and because of the intuitive appeal of information-theoretical quantities.

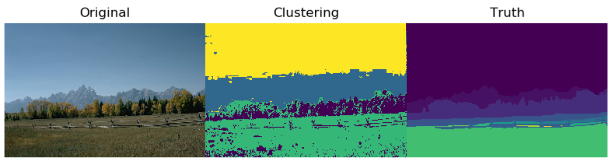

Figure 2. An example of a segmentation with gmm.
Mutual information: 0.7973998355087308 Mutual information between truth and uniformly random k-clustering (for comparison): 0.0003009176324996838

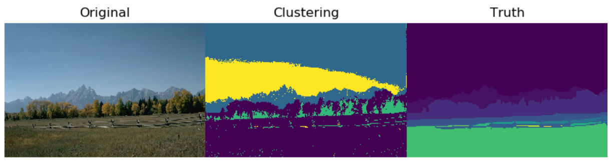

Figure 3. An example of a segmentation with Hierarchical.
Mutual information: 0.7086781181034963 Mutual information between truth and uniformly random k-clustering (for comparison): 0.0002691360551027467

7 Results

The methods gmm, kmeans and hierarchcial perform quite similar. They perform very well in images that the colour was a good indicator of a segment. In contrast watersheds was more robust when the scale of colours was not as variate as the segments. Some examples are shown, the metric to evaluate the clustering is the mutual information, this metric is explained below. There a re few examples of how kmeans don t work as well when there is no an important color change between segments. Another interesting finding is the performance obtained by this methods when using the position for segmentation also. In general using the position do not give a better performance, it can maybe be for the fact that even between close points, a great change in contrast can occur. In spite of being a rather obvious observation, it is worth mentioning that the execution of a Gaussian mixture model and a hierarchical cluster is rather expensive. In particular for this last one, the computation of a distance matrix can be rather expensive for high resolution images. The watersheds algorithm did a remarkable performance when an adequate number of region minima was choosen. For more detailed results the images in this report have as their caption their mutual information score.

References

[1] Markers for watershed transform. http://scikit-image.org/docs/dev/auto_examples/segmentation/plot_marked_watershed.html. Accessed: 2010-09-30. 1

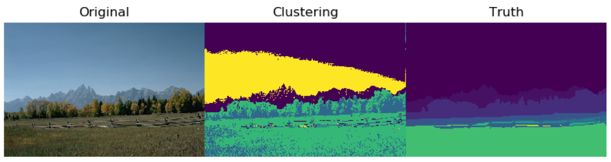

Figure 4. An example of a segmentation with gmm.
Mutual information: 0.67600114441049 Mutual information between truth and uniformly random k-clustering (for comparison): 0.00035498840821610717

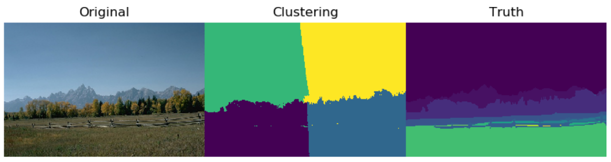

Figure 5. An example of a segmentation with knn and using spatial information
Mutual information (more is better): 0.665712350076357 Mutual information between truth and uniformly random k-clustering (for comparison): 0.00028057074915159265

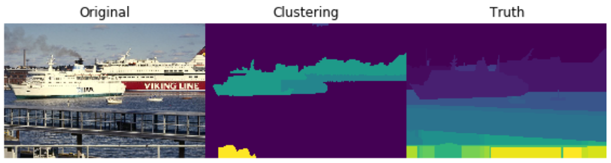

Figure 6. An example of a segmentation with watersheds