Machine Learning

Lab 6: k-Means

1 Overview

“Despite its drawbacks, k-means remains the most widely used partitional clustering algorithm in practice.”

“Top 10 algorithms in data mining” by X. Wu et al. DOI 10.1007/s10115-007-0114-2

The k-means algorithm can be used to find clusters in a data set using only an initial guess of the number of clusters. By alternating between two steps, the algorithm gradually learns more reasonable centers for each cluster until a steady state is reached. This is known as the expectation-maximization algorithm, and in this lab you will write the code which solves this problem then apply it to three sets of data: two sets are collections of points in the plane, and the third involves grouping similarly colored pixels in an image.

2 Provided Resources

- `pointclouds.m` - Function which generates the data for the first data set: five point clouds consisting of 1000 points in the plane.
- `pointrings.m` - Function which generates the second data set of points in the plane composed of five reasonably well connected rings.
- `rgb2im.m` - Function which converts RGB color values in $[0,1]^3$ to an image with the provided dimensions.
- `im2rgb.m` - Loads an image file and returns it as RGB color triples for each pixel in the image, plus the original image data and the image dimensions.
- `km.m` - Function to be completed which performs k-means data clustering on arbitrary dimensional point clouds (in this lab, two and three dimensional).
- `plane_small.png` - An image which can be used in this lab.
- `mountains_small.png` - Another image which can be used in this lab.

3 Guide

1. Complete the function `km.m` which performs the k-means iterations on an arbitrary collection of points in $\mathbb{R}^d$. If the data is denoted by x_n and we are trying to find K clusters with centers μ_k, in each iteration your code should assign each observation to the nearest centroid:

$$t_n \leftarrow \arg\min_k \|\mu_k - x_n\|_2, \quad \forall n$$

and then re-learn the centroids based on an average of assigned data points:

$$\mu_k \leftarrow \frac{\sum_{n|t_n=k} x_n}{\sum_{n|t_n=k} 1}, \quad \forall k$$

where the index n is always assumed to range over the number of observations and k over the number K of clusters being learned. The initial parameters for each cluster should be chosen by randomly selecting data points as centroids, and the algorithm should run until convergence (that is, data points no longer change assignment).

2. Examine the data generated by `pointclouds.m`. In particular, make sure to look at a scatter plot of the data for example by using `scatter(X(1,:),X(2,:),1,Y)`. Finally, use the k-means function to classify the data generated by `pointclouds.m` into five clusters.

3. Generate and turn in the classification result of your code for three separate runs of the k-means algorithm that lead to different results. This should be in the form of three scatter plots with the class assignments visually distinguished.

4. Do the same for the `pointrings.m` dataset, again producing three distinct classification result figures.

5. Which dataset do you believe k-means performed a better job clustering, on average? Why do you believe this is the case?

6. Use the function `im2rgb.m` to read the pixel RGB values of either of the provided images (or pick an image of your own):

   ```
   [X,I,dims] = im2rgb('plane_small.png')
   ```

   Perform a k-means clustering of these color values (`X`) in $\mathbb{R}^3$ using ten clusters.

7. Generate and turn in the image produced by setting each pixel color to the centroid of the class it belongs to. This can be accomplished with the command:

   ```
   imshow(rgb2im(mu(:,labels), dims))
   ```

   where `mu` contains the learned centroid values, `labels` contains the learned class assignments, and `dims` is the image dimensions from `im2rgb`. Show the original image, for comparison:

   ```
   imshow(I)
   ```

4 Extra

1. Suppose that we use the above method to compress an image with N pixels using 16 classes. Assuming nothing else about the image, how many bits of information will we need to store it?

2. Quite evidently, k-means clustering depends on the initial guess for the cluster centroids. For this lab we simply selected data points at random but in general this can be unreliable. What alternative method of initialization do you believe could improve on this?