PROBLEM SET 6

Notes

This problem set adds up to 140 points, which means that you should do problem 1 and any 2 out of the remaining 3. The remaining problem will be added to the subsequent problem set a week later. So, if you want a break, do all of them and have an easy week later :-).

1. AGGLOMERATIVE CLUSTERING (20 pts)

Implement Minimal and Maximal algorithms. Experiment with data drawn from two Gaussians with different covariances -

\[
\mathcal{N}_1 = \left[ \begin{array}{cc} 0 & 0.3 \\ 0 & 3 \end{array} \right], \quad \mathcal{N}_2 = \left[ \begin{array}{cc} 4 & 0.3 \\ 2 & 0 \end{array} \right]
\]

(1) follow the framework for agglomerative clustering algorithms given in class;
(2) write a function \texttt{dmin()} and \texttt{dmax()} that for two clusters of data returns a distance between them. \texttt{dmin} returns the distance between two closest points of the clusters, \texttt{dmax} - a distance between the farthest points; \textit{Hint: you can use an array of cluster labels and just use subsets of your data: \texttt{X(:, find(Labels==1))}, without having to store them in separate matrices.}

Change covariances at will and run both algorithms on the resulting data. Can you tell what sort of clusters each of the algorithms prefers?

2. DIVISIVE CLUSTERING (40 pts)

Implement an algorithm for Divisive clustering. You can use this algorithm to produce a good initialization for your subsequent K-means.

(1) Start with all your data being in one cluster \((K = 1)\);
(2) Iterate until \(K = 10\);
(3) Find the cluster with the highest variance (compute the covariance, find the largest eigenvalue, split the cluster along the corresponding eigenvector);
3. K-means (40 pts)

3.1. **Random data.** Implement it. Use a set of data that you generate from 3 Gaussian sources (1000 points from each source): $\mathcal{N}([0; 0], \text{eye}(2))$, $\mathcal{N}([9; 2], \text{eye}(2))$ and $\mathcal{N}([3; 1], \text{eye}(2))$ (use the `sample_gauss()` function that was posted for one of the previous problem sets).

You need to make sure that your algorithm is sufficiently fast. So, NO LOOPS OVER DATA POINTS. Well, you’ll probably have some loops still, but you HAVE TO use matrices instead of loops as much as possible if you want to see the results before the end of term.

Try it with different number of clusters starting with different initial configurations. Compute the quantization error for each and plot it for $K = 2...10$:

$$E = \sum_{k=1}^{K} \sum_{n=1}^{N} \left\| x_n^{(k)} - t_k \right\|^2$$

Submit the plot of quantization error and a plot of the data points plotted in different colors or shapes for different classes for $K = 4$.

3.2. **Color quantization.** Cluster an image in the color space.

1. Turn it into a set of 3D vectors - RGB for every pixel;
2. Apply the clustering algorithm;
3. Replace each pixel with the value of its nearest mean;
4. Submit a picture and the number of clusters that you used.

Your clusters will not necessarily be all contiguous. Again, try to not use loops.

3.3. **Image segmentation.** Now add positions to color. Take the $3 \times N$ array of color values and add to them two more rows of $X$ and $Y$ positions of these pixels. Obviously you can just reshape two matrices that you can get with `meshgrid(...)`. Then replace each pixel in the image with its cluster number. Submit.

4. EM for Gaussian mixture (40 pts)

Implement it and repeat the last two experiments on the image. Use `plot_gauss(...)` function distributed with the problem set if you’d like to plot your Gaussians while debugging.

*Note: Mixture model provides you with the membership function - the posterior: $p(j|x_n)$, where $j$ is the cluster label. That means that your cluster assignment is “soft”, that is, no cluster owns a data point exclusively, as in K-means. You have to take “argmax” of the posterior in order to determine the “hard” cluster label for each data points to plot it: $[\text{junk, j}] = \max(p_{k\text{ given } j})$*