Jeremy Midvidy, jam658
EECS 349, Fall 2017
Assignment 6 PDF

Problem One:

Initial Parameters:

mu1 = [10 , 30]
mu2 = [-20, -2, 50]

sig1 = [20 , 20]
sig2 = [20, 20, 20]

w1 = [0.5, 0.5] #random weights that add up to one
w2 = [1/3 , 1/3, 1/3]

its = 25

I used plt.hist() to look through the data for x1 and x2, and it looked like x1 had 2 gaussians and x2 and 3 gaussians. Mu1 and mu2 refer to my initial estimates for the mean of each component of each GMM per x1 and x2. Sig1, Sig2, w1, and w2 were my initial variance and weight parameters, respectively.

When I ran my code, it returned with final optimal parameters:

Class 1
mu = [9.7748859235799497, 29.582587182910004]
sigma^2 = [21.92280456278392, 9.7837696131844893]
w = [0.5976546303844265, 0.40234536961557466]

Class 2
mu = [-24.822751728500094, -5.0601582830661158, 49.624444719527126]
sigma^2 = [7.9473354088778425, 23.322661812346212, 100.02433750443187]
w = [0.20364945853348632, 0.49884302378965145, 0.29750751767686234]

These parameters look pretty good and I am satisfied with my code's result.

Problem Two:

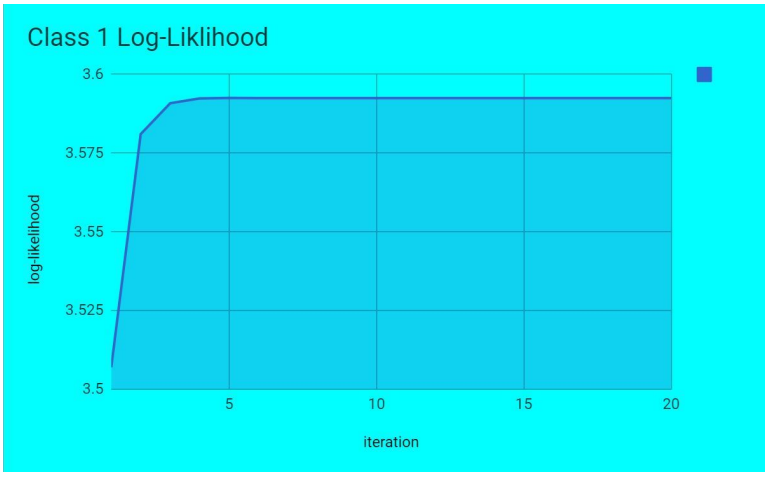
I think my programmed converged pretty quickly. After each iteration, I log-transformed the sum of the probability of each gaussian component. I did this for the first 20 iterations as specified by the assignment, and for each class, was returned with the log-likelihood {L} list:

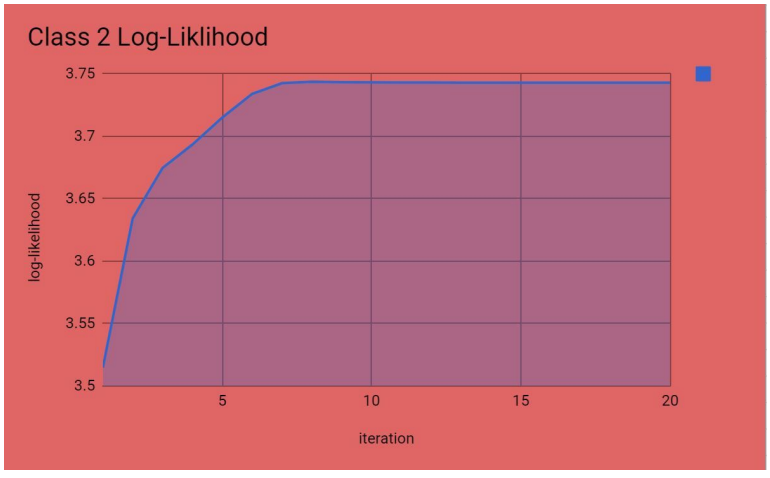
Class 1:
[3.506989756860647, 3.581066221460361, 3.5908553809900488, 3.592375761985201, 3.5925314052249506, 3.5925130487144217, 3.5924927748181, 3.5924828110229794, 3.5924786160390463, 3.5924769363000473, 3.592476275939507, 3.592476018136472, 3.5924759177614893, 3.5924758787215008, 3.5924758635433847, 3.592475857643308, 3.5924758553499543, 3.592475854458554, 3.5924758541120765, 3.5924758539774064]

Class 2:
[3.51475279257795, 3.634317098766968, 3.6746582222609927, 3.6934234472060887, 3.7150195618473543, 3.733996770502769, 3.7425418316626504, 3.7436758433507134, 3.7433794735573533, 3.743165611959898, 3.7430778884226026, 3.7430460494121496, 3.74303491074102, 3.743031060484418, 3.7430297349529424, 3.7430292792388298, 3.7430291226393555, 3.743029068834937, 3.743029050349851, 3.7430290439992224]

As you can see in the data, with Class 1, it converged to 3.5293 after 4 iterations. And with Class 2, it converged to 3.742542 after 7 iterations.

Here are what these probabilities look like graphically:





Each graph clearly shows convergence, when the plot of the log-likelihood levels off over the iterations. In Class 1, this occurs at iteration=4 and in Class 2 this occurs at iteration = 7.

Problem Three

Program / Command line output

Problem Four

After creating the classifier, I ran all of the data, and then compared the classified data to the data split by class, and these were the accuracy results:

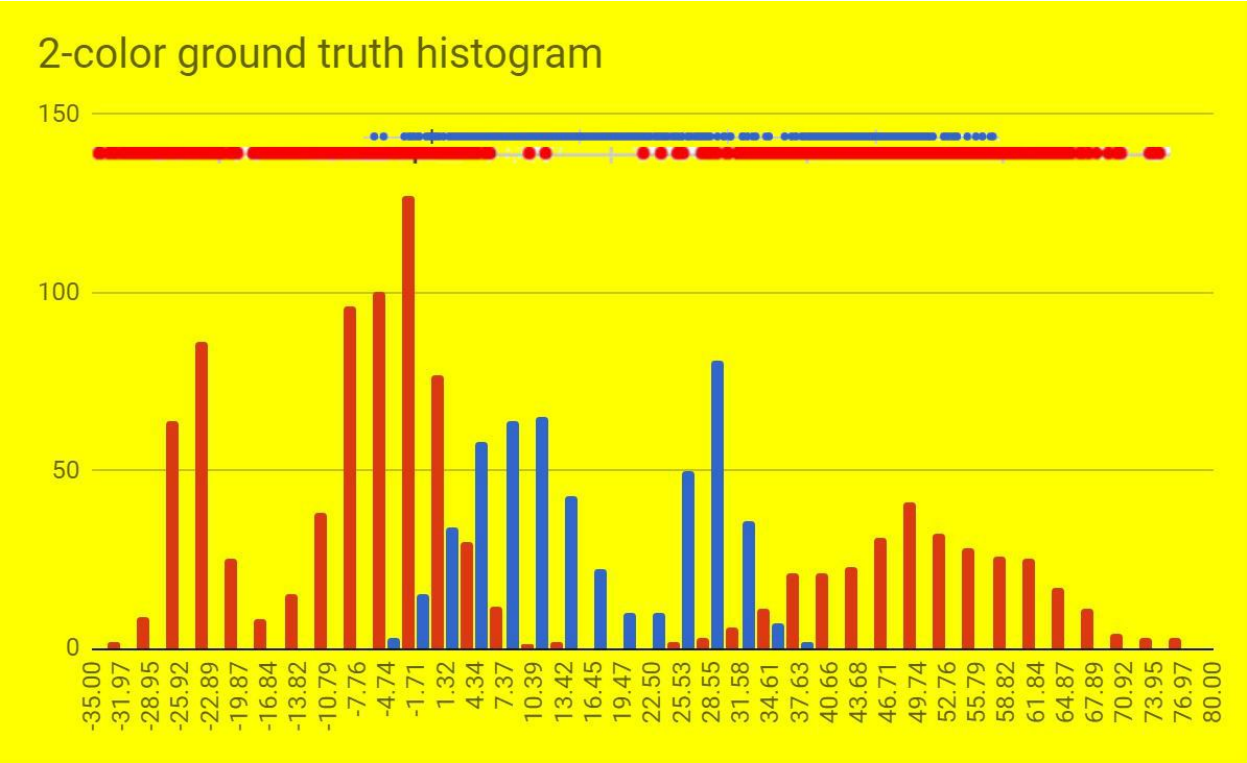
Class 1 was classified 92.6% correctly.
Class 2 was classified 66.2% correctly.

I am pretty happy with these results because Class 1 was relatively easy to classify - two gaussians, centered at around 10 and 30 each, each with relatively similar variance and weights. This indicates that points that should have been classified as Class 1 would have been pretty easy to classify as Class 1 because the gaussian components in the mixture model were pretty evenly spread, weighted, and thus would most likely produce a high probability of classifying points that it should have classified.

Class 2's performance was a little bit lower, and this makes logical sense. It had three gaussian components, centered -25, -5, and about 49. This makes it a little harder to classify points that it should have classified because, for example, if the data point x= -10 would have been difficult for this GMM to say it would have predicted. With weights from the first two gaussian components, it would have been split on whether to classify it on one of the two centered about -25 and -5. Then it would produce an extremely low probability of being classified by the component centered around 49. So since it has two components relatively closer together (-25 and -5), each probability would end up being lower than if it just had one gaussian centered around, say (-15). This is true because gaussians -25 and -5 have greater individual variance than just one gaussian would have had, so it it would decrease the overall probability of summing prob(g@-25) and prob(g@-5). So the overall complexity and of GMM2 means it should have a harder time classifying data than GMM1.

And that's what this data shows. GMM1 was really good at classifying data points it should have classified, and GMM2 was pretty good at classifying data points it should have classified, at 92.6% and 66.2% respectively.

This can be seen graphically. In this histogram below, Class 1 is in red and Class 2 is in blue (ground truth). The lines at the top show the distribution of classifications by my GMM predictions. This shows how accurate Class 1 data points were classified (much of the red bar has red dots above it, with space where much of the blue bars are) and also how accurate Class 2 data points were classified (much of the blue bar has blue dots above it, even though some red bars have blue above, indicating misclassification.)



Problem 5:

Yes, you would be able to use a closed-form solution to find the parameters for GMM that maximize the probability of the data without using EM. Since the gaussian that generated each data point *is known*, we can used the partial derivative of the all the gaussian parameters in order to calculate optimal parameters. We can use a closed form solution because this equation:

to 0.

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \left(\frac{-(x_i - \mu)^2}{2\sigma^2} - \log\sigma \right)$$

Will work when given the known gaussian for each data point. This means we don't have to use EM to figure out the optimal parameters because we can just use this equation (partial derivative of mu, sig^2) in order to find the optimal parameters.

Problem 6:

Yes, in this case, we must use EM. Since we don't know which gaussians generated each point (we actually don't even know a single gaussian), we can't be sure that applying the equation above would work. If we aren't given the gaussians, we can't be sure that the gaussian parameters per data point well represent the optimal parameters. Furthermore, since we don't even know *any* gaussian parameters, we must use EM to find the optimal gaussian parameters (much like this problem set.)