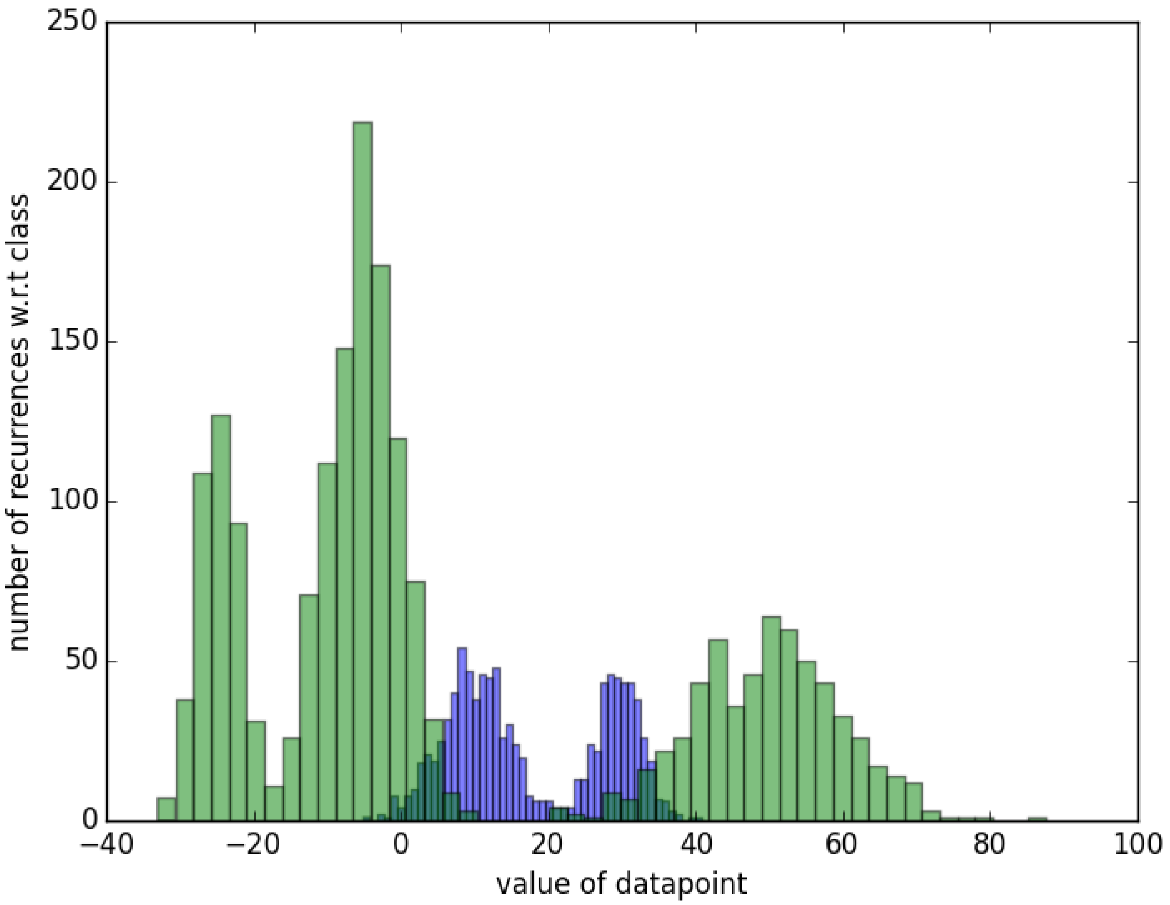
EECS 349 HW 6
Marc Gyongyosi

Problem 1
a) gmmest.py is included in the submission
b) For the first model (1), I choose two gaussians since the distribution of the data points indicates the presence of 2 subpopulations/sources. I initialize the parameters based on visual inspection of this histogram. Class 1 is blue, class 2 is green.



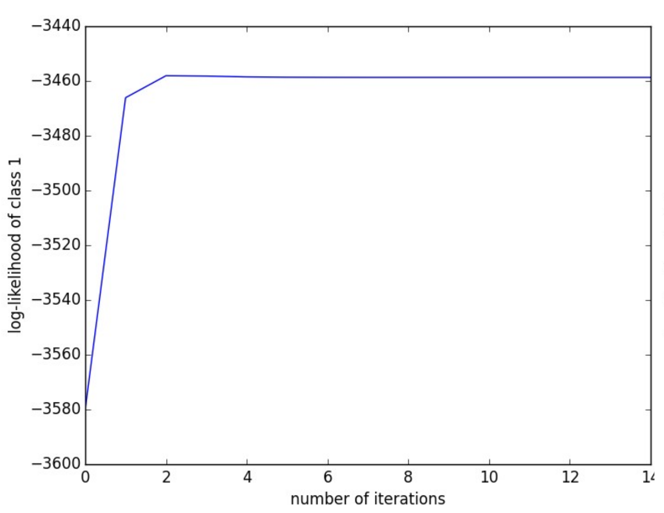
Model 1:

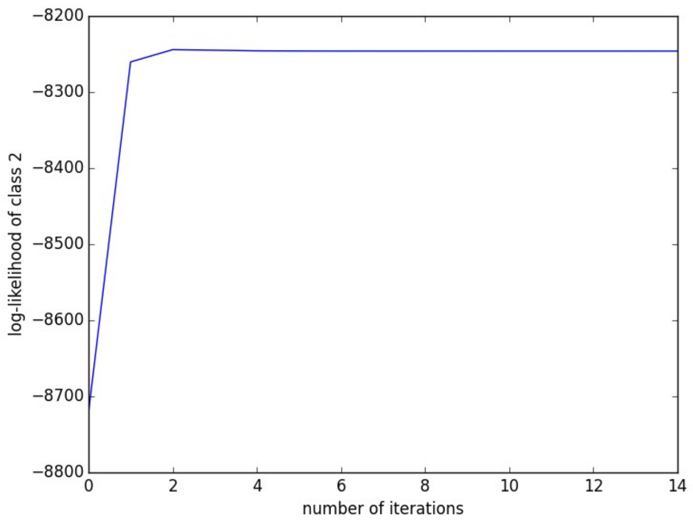
mu_init = [11,30] ### based on histogram plot
sigmasq_init = [1,1]
wt_init = [.6,.4]
its = 15

Model 2:

mu_init = [-25,-5,48] ### based on histogram plot
sigmasq_init = [1,1,1]
wt_init = [.2,.4,.4]
its = 15

This is the log-likelihood for the first and second class:



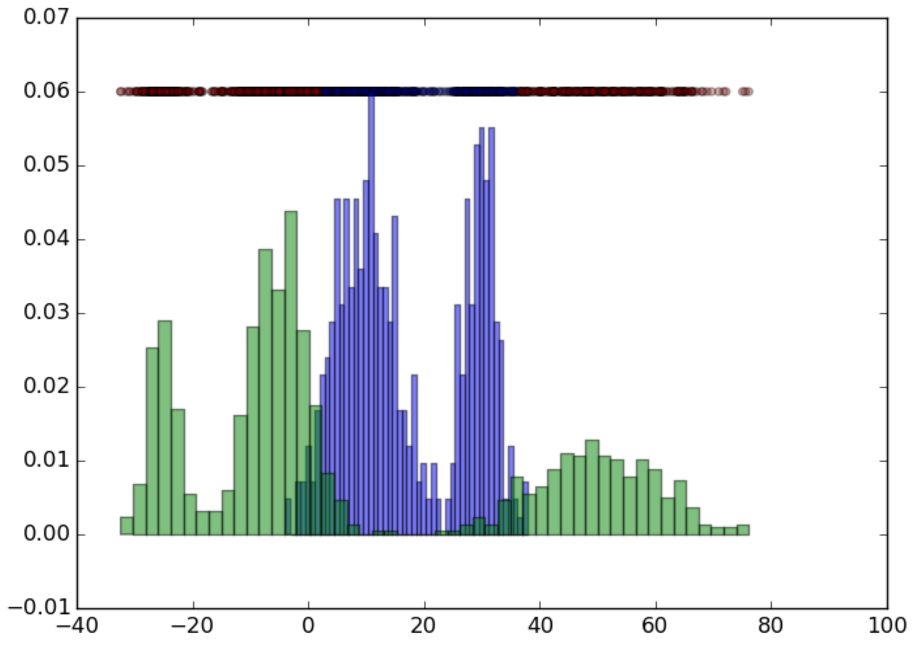


I think my program has converged, because the log likelihood is flat and the parameters of the returned models vary only slightly if I increase/decrease the iterations. The parameters for its = 15 are:

MODEL 1:
- MEAN: [9.7748860193066864, 29.582587234395586]
- VARIANCE: [21.922805620624587, 9.7837692541886128]
- WEIGHTS: [0.59765463497206051, 0.40234536694319667]

MODEL2:
- MEAN: [-24.822751701961966, -5.0601582724403276, 49.624444719515857]
- VARIANCE: [7.9473355670936909, 23.322661683389818, 100.02433750486372]
- WEIGHTS: [0.2036494593462945, 0.49884302339803449, 0.29750751767694433]

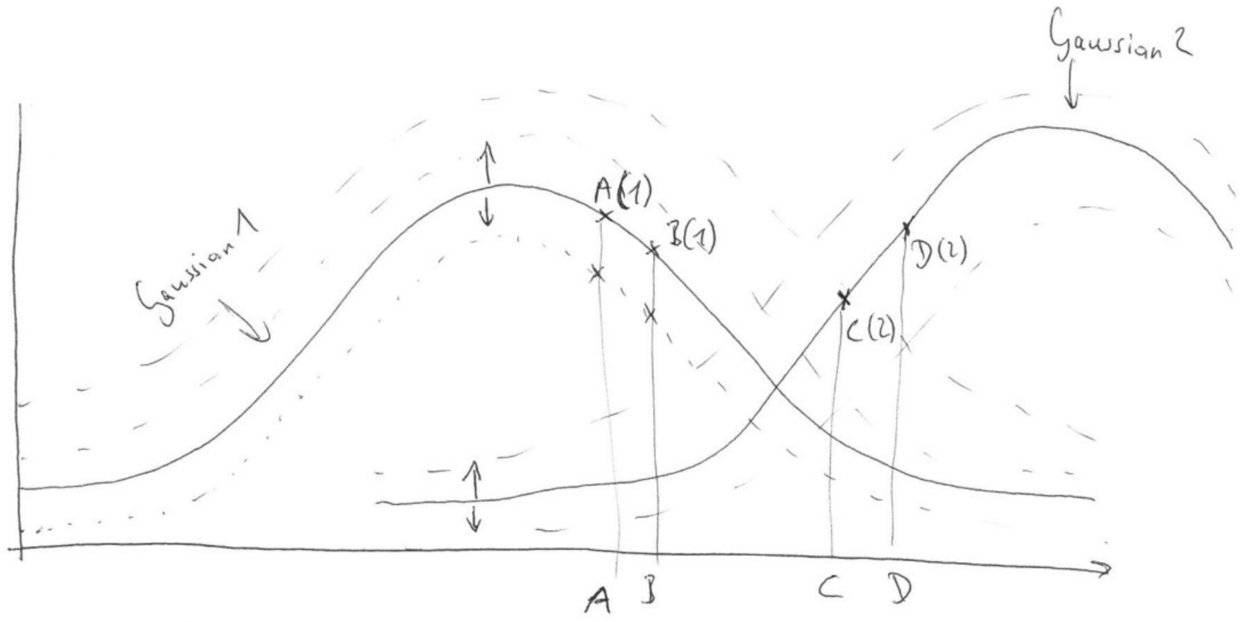
C) The function is implemented in gmmclassify.py. It returns an array corresponding to the GMM classificiations of the provided data. The output is of size len(X). The classifier has about 6.5% error rate. The requested plot is shown. Red Circles indicate class 2, blue ones class 1.



The prior probability on the training data for class 1 is 33%. On the training data just using this would give us 66% accuracy, or 33% error rate (significantly more than the GMM classifier). On the testing data, there are 1500 instances, 500 of them are class 1, 1000 of them class 2. This means the prior probability for class 1 is the same as on the training data. Thus GMM outperforms prior probability quite significantly.

Problem 2:

a) I don't think a closed form solution exists, because even if I can accurately define “where each Gaussian should be dominant” by looking at the points associated with it, a finite number of Gaussians is not enough to determine what the exact P(xi|Gn) (i.e. probability of xn given that it is Gaussian n) is. To visualize this, let's assume this scenario where I have two Gaussians. Each one of them contributes to a number of points on them, however, we don't know the exact probability values that we should have and such we can shift the Gaussians up and down as long as we don't change the relative magnitude (i.e. Gaussian 1 is bigger at points A,B than Gaussian 2). This allows for infinite possible Gaussians, if there are more Gaussians in the system then that will lead to more constraints on which “levels” the Gaussians can be on or in between, but still since we are dealing with real numbers there is inifinitely many in such a constrained interval. This probability in this “region” defined by the constraints of all the other Gaussians is what we are maximizing with the EM algorithm, it doesn't have a closed form solution.



$\rightarrow$ as long as $G1 > G2$ @ A,B and $G1 < G2$ @ C,D

$\Rightarrow$ Solution

$\Rightarrow$ infinitely many solutions

NO CLOSED FORM

b) One of the fundamental assumptions for the EM algorithm is that each point was only created by a single Gaussian. Thus, if we don't know which one each point belongs to, we would have to do some kind of clustering first to give every data point a Gaussian before using EM. You cannot use Expectation Maximization if you're data can belong to multiple classes, because then you can't maximimze within margins (as described before) since the margins are not defined.