CSE 5522
Roee Ebenstein

HW3

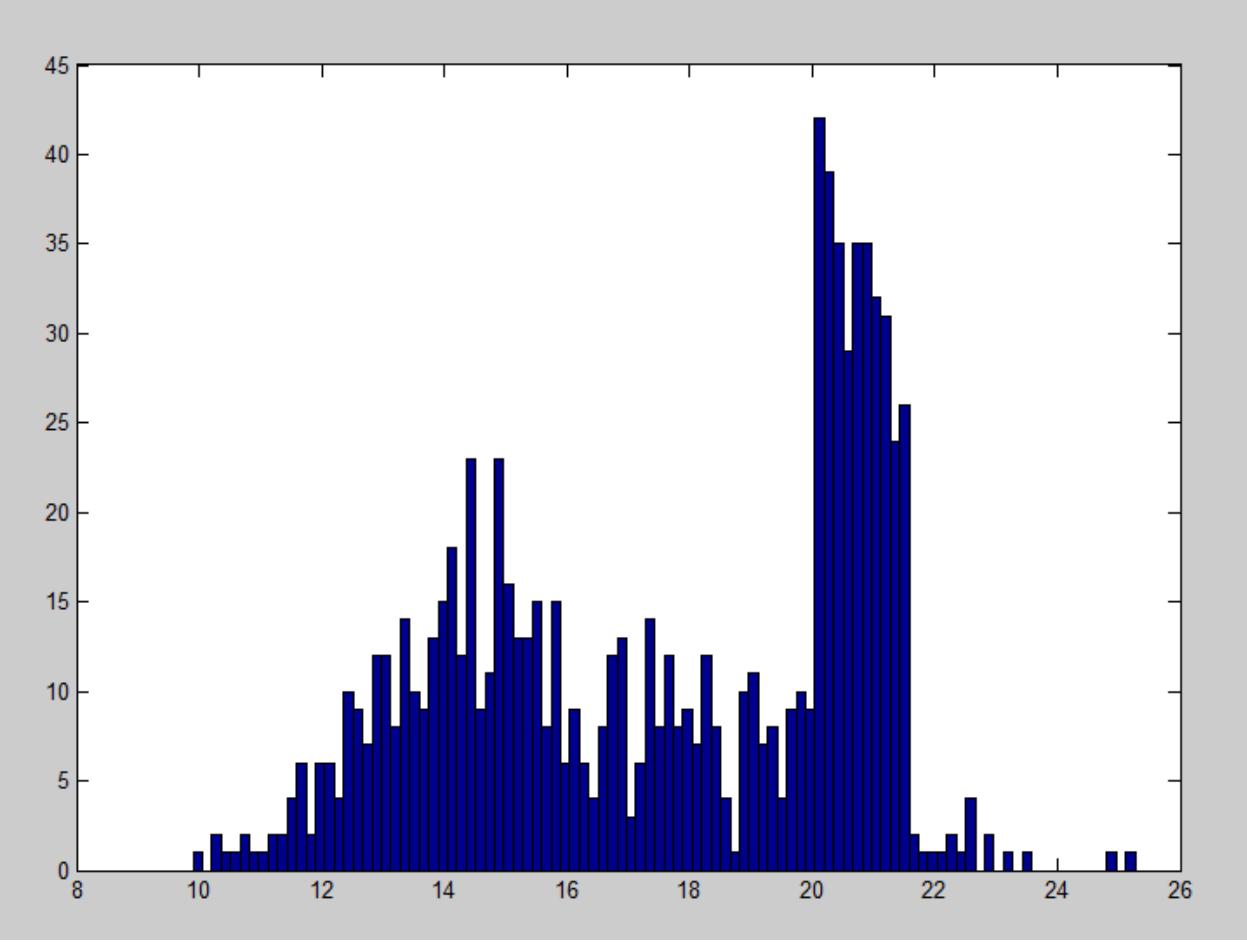
Q1

In this question I thought about the best way to pick the best initial guess.
Instead of using random number (and after consulting, and getting approved by the professor), I used "an educated guess" – I sorted the numbers, built 3 equal sized cluster, for each calculated the mean and standard variation, and used that as the initial guess.
The advantage it has – the final result will always be consistent, the downside is that if the clusters are not nearly the same size, there's no random search that will allow other hits.

The Maximization call is divided to a section that's within the main loop, and a section that is in an extracted method.

The results are:
Cluster 0: Mean=13.814842767295605 , std=1.2357315737844061 , number of objects in the cluster=318
Cluster 1: Mean=17.70244239631337 , std=1.447179690895959 , number of objects in the cluster=217
Cluster 2: Mean=20.74846575342464 , std=0.5667879718127016 , number of objects in the cluster=365

As can be seen on the data plot – this is a pretty good result:



CSE 5522
Roee Ebenstein

Just for fun, I tried running it with random mean value and standard variation,
Results were sometime similar sometimes far away (dependent how close the guesses were):
Cluster 0: Mean=13.826593750000006 , std=1.2407434605755279 , number of objects in the cluster=320
Cluster 1: Mean=20.71745501285346 , std=0.7225362008929627 , number of objects in the cluster=389
Cluster 2: Mean=17.403874345549745 , std=1.0078112019450534 , number of objects in the cluster=191

Cluster 0: Mean=15.461433891992556 , std=2.460674536179873 , number of objects in the cluster=537
Cluster 1: Mean=20.222774566473984 , std=0.26467853998113683 , number of objects in the cluster=173
Cluster 2: Mean=21.08636842105262 , std=0.27982507891939173 , number of objects in the cluster=190

Q2

I decided to try and increase the number of Gaussians.
The code I wrote is generic enough for it to be just a change of a parameter (since I planned to try to do it either way…)

My hypothesis is that for 4 Gaussians it will find the small noise around 25 as a peak – and will create a mean after the last actual peak (maybe split it to two different peaks surrounding the last,20.7, peak),
For 5 Gaussians it will in addition split the 13.8 Gaussian to two (the first peak): something around 12 and something around 15,
And for 6, all of the above, and will split the last peak (17.5) to some peak between 17.5 to 15, and another peak around 19.
All of that is being guessed by watching the graph on Q1.

Evaluation:
For 4 Gaussian – I would expected the end of the data to be found as a peak:

For 4 Gaussians

Found results for the 1D EM problem:

Cluster 0: Mean=13.938377581120953 , std=1.2899070507658248 , number of objects in the cluster=339

Cluster 1: Mean=17.7436507936508 , std=1.0113074489796876 , number of objects in the cluster=189

Cluster 2: Mean=20.242202380952378 , std=0.24300799907282628 , number of objects in the cluster=168

Cluster 3: Mean=21.217598039215673 , std=0.6063411768478646 , number of objects in the cluster=204

What I expected actually been found, but instead of being way around 25, it gave a value of 21.2.
I consider it as a success since I found the right area, the specific number I recognized was wrong though.

For 5 Gaussians:

For 5 Gaussians

Found results for the 1D EM problem:

Cluster 0: Mean=12.886094674556224 , std=0.9291168501400902 , number of objects in the cluster=169

Cluster 1: Mean=15.08236559139786 , std=0.5867055788343956 , number of objects in the cluster=186

Cluster 2: Mean=17.94308988764045 , std=0.9550580915380733 , number of objects in the cluster=178

Cluster 3: Mean=20.32489583333332 , std=0.25705258375665524 , number of objects in the cluster=192

Cluster 4: Mean=21.305257142857133 , std=0.6118933774001175 , number of objects in the cluster=175

I suspected the first peak would be split to two, while the earlier peak will remain – and that what happened.

And last, for 6 Gaussians:

For 6 Gaussians

Found results for the 1D EM problem:

Cluster 0: Mean=12.659500000000005 , std=0.8611772050927897 , number of objects in the cluster=140

Cluster 1: Mean=14.7235393258427 , std=0.5188279261882538 , number of objects in the cluster=178

Cluster 2: Mean=16.786929133858273 , std=0.6601831192051958 , number of objects in the cluster=127

Cluster 3: Mean=18.872755102040813 , std=0.5944209678029241 , number of objects in the cluster=98

Cluster 4: Mean=20.35983695652171 , std=0.22914888729221225 , number of objects in the cluster=184

Cluster 5: Mean=21.31179190751444 , std=0.6123768934142526 , number of objects in the cluster=173

This one also responded as I expected.

My hypothesis have been confirmed.
(I am actually surprised a bit – I thought there are too many things that could split first, my intuition said, that first the edge that looks funny in the graph would shine out, than the largest peak, and afterwards the other peak).

Q3

These are the results:

Found results for the 1D K-Means problem:

Cluster 0: Mean=13.594321428571435 , number of objects in the cluster=280

Cluster 1: Mean=16.829708737864085 , number of objects in the cluster=206

Cluster 2: Mean=20.614516908212547 , number of objects in the cluster=414

As can be noticed the results are not far from 13.81(number of objects in the cluster=318), 17.7 (num =217), 20.74(365).
Although the results are not too far away – given the information of each original group had 300 samples in it, the error in number of samples in k-means is 20+94+114 = 228, while for EM is 18+83+65=166.
If we measure by it – EM is better.