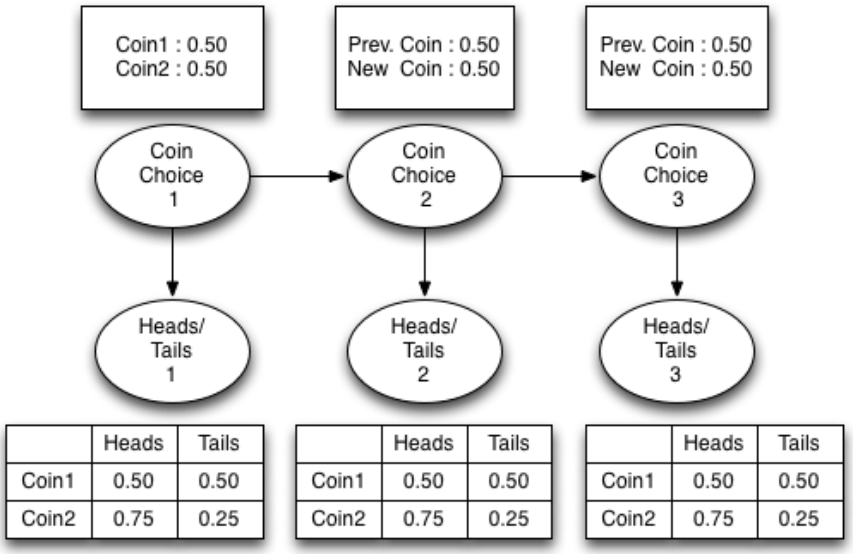
Jacob Klingler
AI2 HW#5
3/21/13

Answers to Questions

1.

a)



	Heads	Tails
Coin1	0.50	0.50
Coin2	0.75	0.25

	Heads	Tails
Coin1	0.50	0.50
Coin2	0.75	0.25

	Heads	Tails
Coin1	0.50	0.50
Coin2	0.75	0.25

b) Flip 1:

Initial Choice (0.50) * Tails (0.25) = 0.125

Flip 2:

Keep Choice (0.50) * Prev. Choice (0.50) * Tails (0.25) = 0.0625
Replace Choice (0.50) * Pick (0.50) * Tails (0.25) = 0.0625
0.0625 + 0.0625 = 0.125

Flip 3:

Keep Choice (0.50) * Prev. Choice (0.50) * Tails (0.25) = 0.0625
Replace Choice (0.50) * Pick (0.50) * Tails (0.25) = 0.0625
0.0625 + 0.0625 = 0.125

c) Assuming "the same coin" means either Coin1 was used for both flips or Coin2 was used for both flips:

Coin1:

Initial Choice (0.50) * Heads (0.50) = 0.25
Keep Choice (0.50) * Prev. Choice (0.50) * Tails (0.50) = 0.125

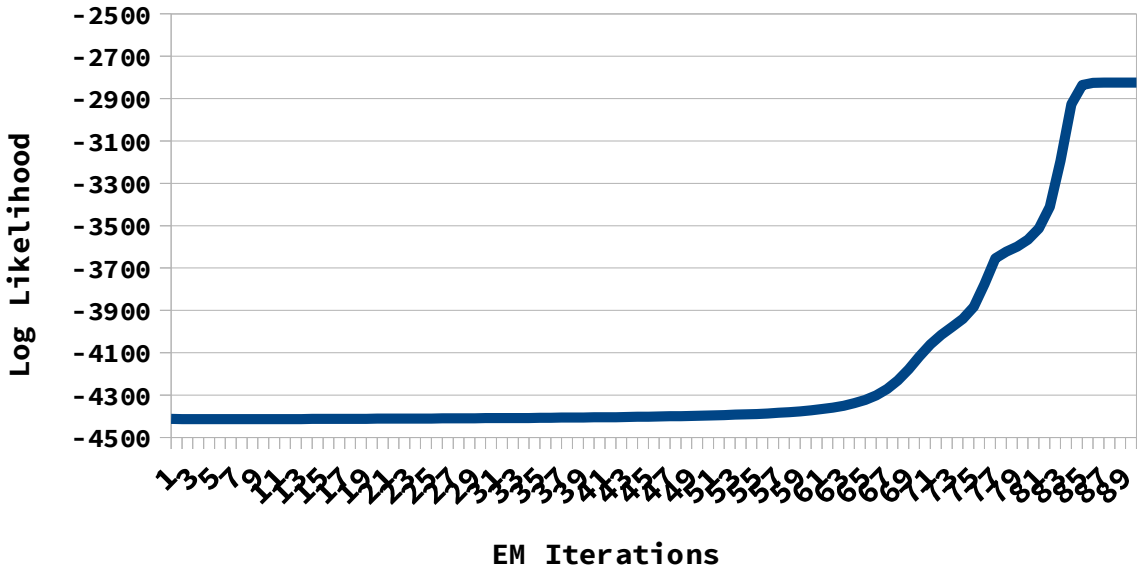
Coin2:

Initial Choice (0.50) * Heads (0.75) = 0.375
Keep Choice (0.50) * Prev. Choice (0.50) * Tails (0.25) = 0.0625

Probability that the same coin was used = 0.25+0.125+0.375+0.0625 = 0.8125

2.

a) EM Iterations vs. Log Likelihood, Data Set 1, 3 Gaussians



The GMM process finds the three component distributions with these parameters:

```
C1 - mean: -1.12938341500     stdev: 2.9309261906      weight: 0.33342375320
C2 - mean: 12.0808746780      stdev: 1.0680248081      weight: 0.33324257114
C3 - mean: 25.0820790861      stdev: 2.0673865793      weight: 0.33333367565
```

After reaching a certain number of iterations, the log likelihood of the data starts to level off.

b)

Running the GMM on the same data, but looking for two component distributions finds this:

```
C1 - mean: 5.97132630036      stdev: 7.466379106       weight: 0.68610447930
C2 - mean: 25.1915441189      stdev: 1.966828103       weight: 0.31389552069
```

With a final log likelihood of `-3268.55`, which is significantly lower than the final log likelihood for three gaussians (`-2850`)

Running the GMM on the same data, but looking for four component distributions finds this:

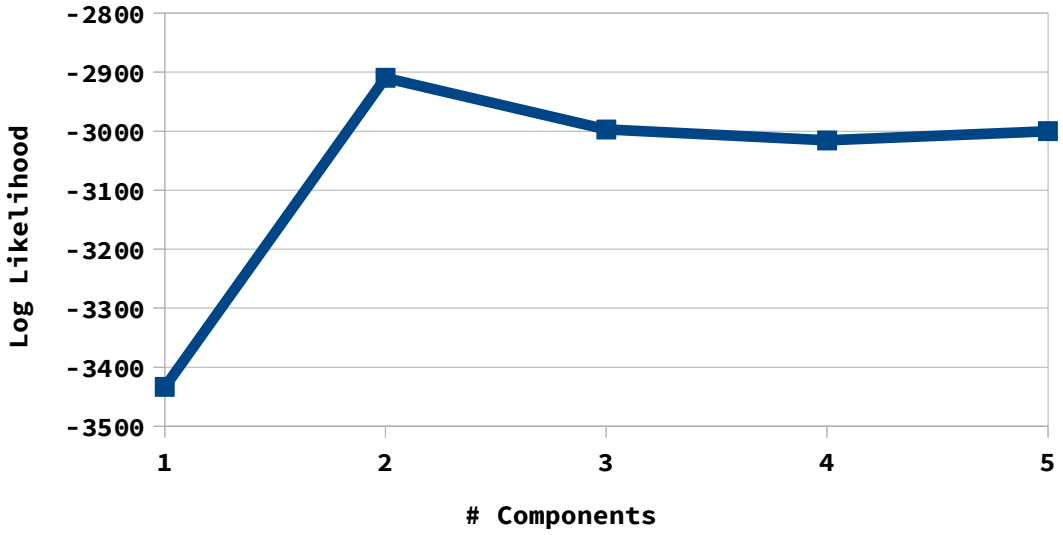
```
C1 - mean: -1.3982013747      stdev: 2.39545491716      weight: 0.16930447490
C2 - mean: -0.8441893504      stdev: 3.40049636139      weight: 0.16438964003
C3 - mean: 12.0837698536      stdev: 1.06646676582      weight: 0.33297220983
C4 - mean: 25.0820887373      stdev: 2.06738552910      weight: 0.33333367521
```

With a final log likelihood of `-3023.56`, which is still lower than the final log likelihood for three gaussians.

Since we know the data were generated by three gaussians, it makes sense that the GMM would struggle fitting two or four gaussians to the data. In fitting two, it has to force one cluster to cover what would have been generated by two clusters, and in fitting four, it has to split what would have been one cluster into two.

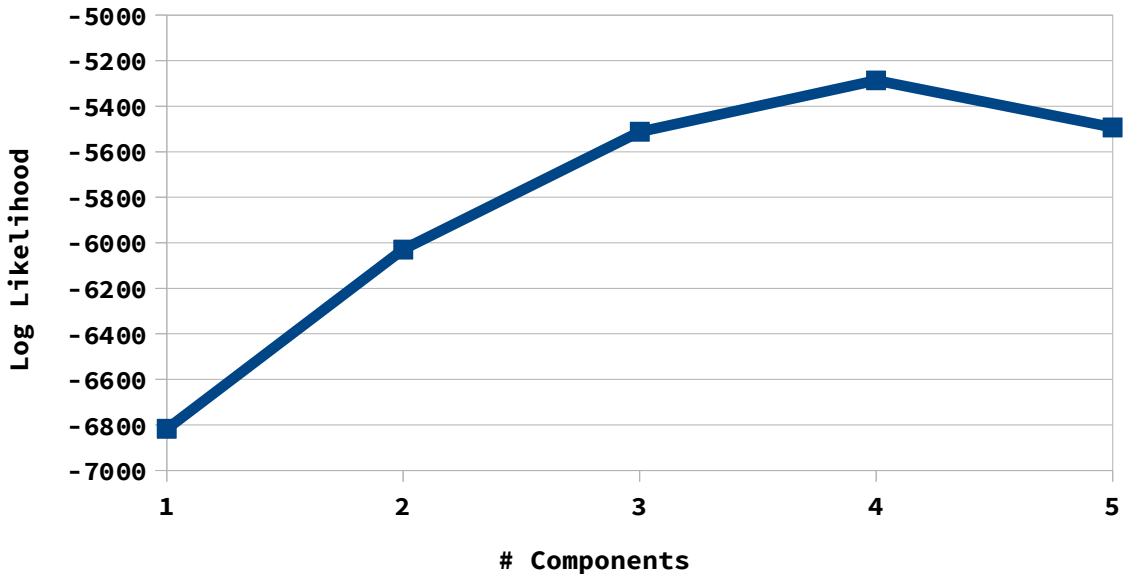
c)

Gaussian Components vs. Log Likelihood, Data Set 2



For the second data set, the number of components peaks the log likelihood at two, and depresses it slightly afterwards.

Gaussian Components vs. Log Likelihood, Data Set 3



For data set three, the log likelihood is maximized at four gaussian components, then tapers off again.

Would it be useful to try and find the “best” number of clusters by increasing them until the log likelihood gets worse? Possibly. It would make sense to pick the smallest number of clusters (Ockham's Razor) with the highest log likelihood for the data. It would help to know at least something about the data to make an educated guess at the number of clusters, though.

Note: the reason my graphs only include the first five components is that my program underflows the Java Double class, leading to errors like components with -Infinity standard deviations, etc. I could have rewritten the program using BigDecimal, but I would have had to find a third-party library supporting square roots of BigDecimals, as this is not supported in the standard library. (Probably should've used Python instead...)