## Implementation Description

I implemented the EM algorithm based on Andrew Ng's notes for GMMs and EM. My flow is: generate some data, run EM and gather statistics, plot results. I defined 7 experiments, each checking an aspect of the EM algorithm. All the experiment setups are documented in the top of the .m file. Each experiment is defined by a combination of two sets of parameters that are used: parameters for data generation, and parameters for EM initialization. I further specify on the experiments in the Experiments section. Next, I will briefly describe each part of my flow.

In order to generate data, I used the prior cluster probabilities and *mvnrnd* to sample from the two gaussians a complete set of points. Next, I initialize the EM parameters with guesses, which are defined in each experiment specification. Based on these guesses I initialize the likelihood function.

Next, I run the EM algorithm. For the E-step, I construct using Bayes's law, two vectors $v_1$ and $v_2$ of posterior cluster probabilites, where $v_j[i] = p(z^{(i)} = j|x^{(i)}; \theta)$ where $x^{(i)}$ is the $i$th point, $z^{(i)}$ is a cluster index of the $i$th point, and $\theta$ are the current parameters. In the M-Step, I use these posterior vectors to maximize the likelihood of the parameters, based on the formulas, using only matrix computations (no for loops). The stopping crieterion is whether the likelihood improvement is smaller than some $\epsilon$. In addition, during the algorithm run, I track the progress of the parameters estimations, and save them in arrays, to be plotted later.

## What have I learned

By reading carefully the notes, I finally learned why EM works... Also, Andrew's explanation about viewing it as coordinate ascent, gave an interesting way to look at it. A few points that I learned follows.

Initially, I didn't understand what is being initialized. From some reason, I had the impression that we initialize hard cluster assignments and then start the algorithm. Now I understand that although this method could work, we usually only initialize the parameters, and let the E-step set the "soft" cluster assignments.

I learned what is exactly the stopping criterion. I wasn't sure whether it's likelihood itself that is being checked for convergence, or the lower bound on it that we keep building. Although both of them could probably work, I used the real likelihood, as it was easier to compute (and is the actual thing that we care about).

In addition, I saw that bad initialization can lead to very bad results. In contrast, many not-so-good initializations works surprisingly well. More on that in the Experiments section.

I also learned that "cluster1" and "cluster2" are just arbitrary names, as many times "red" points were explained accurately by "blue" classifications. Retrospectively, it makes sense...

## Experiments

(I am $\frac{1}{2}$ page beyond the limit because I ran some more experiments.)
In the experiments, I tested the effects of:

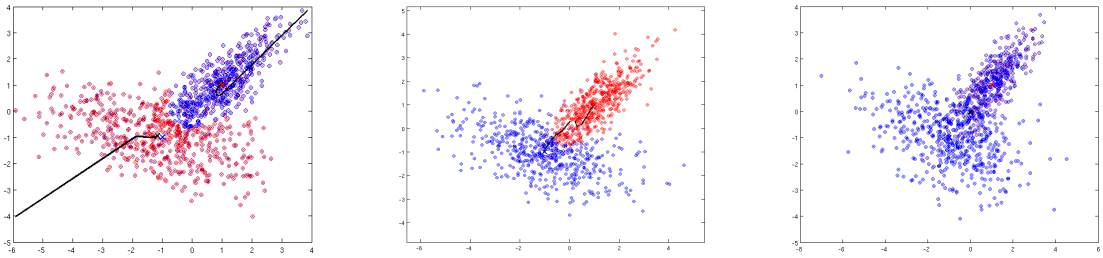
- Changing the mean initializations in the algorithm.
- Changing the distance between the two Gaussians.

- Changing the probability distribution of the hidden variables, the one that controls from which cluster to generate the next data point ($\phi$ here).

Covariance guesses were always initialized to I, and $\phi$ guesses were always initialized to 0.5.

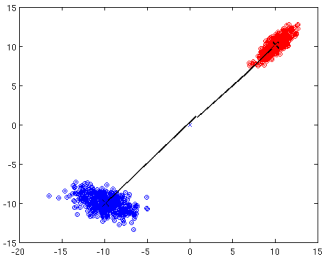
For each experiment, running on 1000 points, I check the number of iterations ran and the percent of correctly assigned points. I also plot the generated points in red and blue 'x's, according to which Gaussian they come from, and again the same points in red and blue 'o's to show how they were classified. I also show in black the mean converging to its final value (a black 'x'), on the same plot. For some experiments I ploted the convergence of the covariance parameters, and the convergence of the cluster-choice probabilities ($\phi$).

**Changing the mean initializations (partially-overlapping Gaussians)** Here, for two partially-overlapping Gaussians, I ran three experiments, each initializing the means differently: on the boundaries of the data, then close to the center but on two different points, and finally identically on the same point on the center of data. The results are as follows **(Please zoom-in to see the details)**.

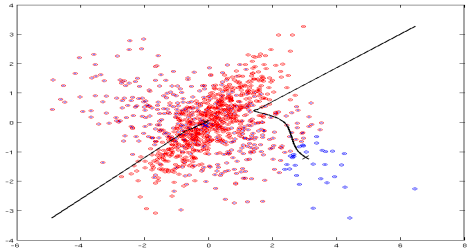
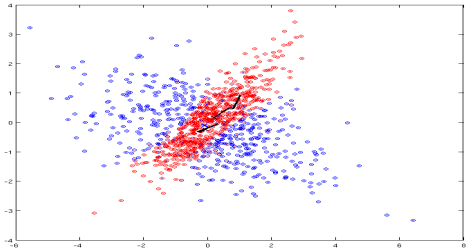


We see that when the means are initializing far from each other (left), or even close but not identically initialized (middle), and even though the Gaussians overlap, the results are pretty good: in the left figure #iterations=28, accuracy=93.5% and in the middle #iterations=32, accuracy=91.3%. However, on the right one, where the two means where initialized identically, all the points were assigned to the same cluster, achieving a baseline (bad) accuracy of around 50% (already a local maximum).

**Far Gaussians: easy classification** Here I tested the algorithm with two far Gaussians. Even-though the means were initialized near each other, at the center, it took 6 iterations to achieve perfect classification of 100% **(Please zoom-in to see the details)**.

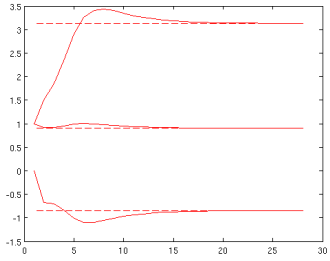
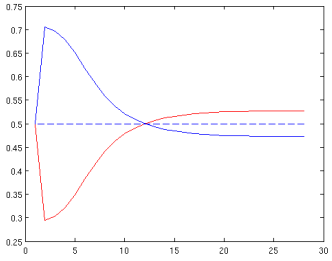
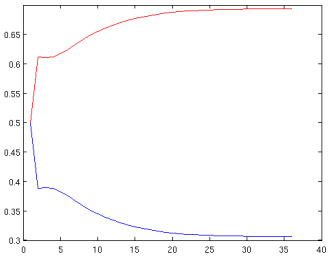


**Significantly overlapping Gaussians: classification depends on initialization** Here I generated two highly overlapping Gaussians. In the first experiment I initialized the means far from each other, on the boundaries of data, and in the second one I initialized both of them close to the center of data at two different points **(Please zoom-in to see the details)**.

We see that here, initializing the means far from each other (left figure) didn't help to achieve convergence: after 131 iterations it converged to 54% accuracy. On the other hand initializing the means close to each other (right figure), which is also close to their real values, achieved impressive results: 39 iterations and 79% accuracy on two highly overlapping Gaussians.

**Other convergence results** Finally I show some graphs for the other parameters' convergence, which were very similar for all runs.

The left figure shows the convergence of the 3 covariance matrix entries to their real values. The other two figures shows the convergence of the $\phi$ values (the probabilities of choosing Gaussian1 or Gaussian2 while generating data) to their values, in case they are both 0.5 (middle) and in case they are 0.3 and 0.7 (right).