# Chris McCormick

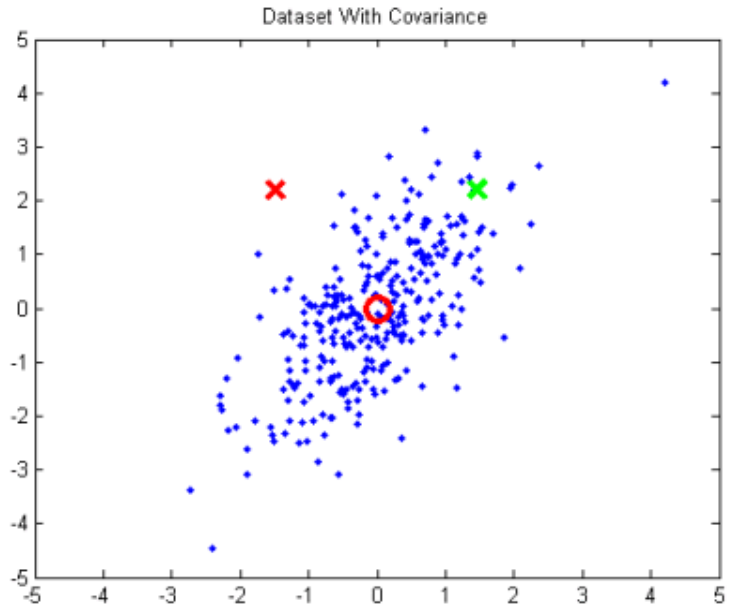
## Computer Vision and Machine Learning Projects and Tutorials

# Gaussian Mixture Models Tutorial and MATLAB Code

August 4, 2014 · by Chris McCormick · in Tutorials. ·
You can think of building a Gaussian Mixture Model as a type of clustering algorithm. Using an iterative technique called Expectation Maximization, the process and result is very similar to k-means clustering. The difference is that the clusters are assumed to each have an independent Gaussian distribution, each with their own mean and covariance matrix.

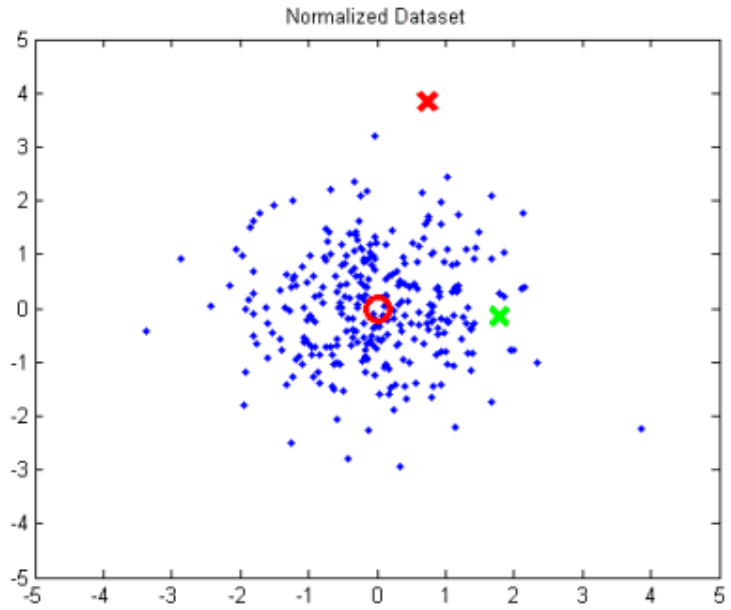
## Comparison To K-Means Clustering

When performing k-means clustering, you assign points to clusters using the straight Euclidean distance. The Euclidean distance is a poor metric, however, when the cluster contains significant covariance. In the below example, we have a group of points exhibiting some correlation. The red and green x's are equidistant from the cluster mean using the Euclidean distance, but we can see intuitively that the red X doesn't match the statistics of this cluster near as well as the green X.




(https://chrisjmccormick.files.wordpress.com/2014/07/datasetwithcovariance.png)

If you were to take these points and normalize them to remove the covariance (using a process called whitening), the green X becomes much closer to the mean than the red X.




(https://chrisjmccormick.files.wordpress.com/2014/07/datasetnormalized.png)

The Gaussian Mixture Models approach will take cluster covariance into account when forming the clusters.

Another important difference with k-means is that standard k-means performs a hard assignment of data points to clusters–each point is assigned to the closest cluster. With Gaussian Mixture Models, what we will end up is a collection of independent Gaussian distributions, and so for each data point, we will have a probability that it belongs to each of these distributions / clusters.

## Expectation Maximization

For GMMs, we will find the clusters using a technique called "Expectation Maximization". This is an iterative technique that feels a lot like the iterative approach used in k-means clustering.

In the "Expectation" step, we will calculate the probability that each data point belongs to each cluster (using our current estimated mean vectors and covariance matrices). This seems analogous to the cluster assignment step in k-means.

In the "Maximization" step, we'll re-calculate the cluster means and covariances based on the probabilities calculated in the expectation step. This seems analogous to the cluster movement step in k-means.

## Initialization

To kickstart the EM algorithm, we'll randomly select data points to use as the initial means, and we'll set the covariance matrix for each cluster to be equal to the covariance of the full training set. Also, we'll give each cluster equal "prior probability". A cluster's "prior probability" is just the fraction of the dataset that belongs to each cluster. We'll start by assuming the dataset is equally divided between the clusters.

# Expectation

In the "Expectation" step, we calculate the probability that each data point belongs to each cluster.

We'll need the equation for the probability density function of a multivariate Gaussian. A multivariate Gaussian ("multivariate" just means multiple input variables) is more complex because there is the possibility for the different variables to have different variances, and even for there to be correlation between the variables. These properties are captured by the covariance matrix.

$$g_j(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)}$$

| Symbol | Meaning |
|---|---|
| $g_j(x)$ | The PDF of the multivariate Gaussian for cluster j; the probability of this Gaussian producing the input x |
| $j$ | Cluster number |
| $x$ | The input vector (a column vector) |
| $n$ | The input vector length |
| $\Sigma_j$ | The n x n covariance matrix for cluster j |
| $\lvert\Sigma_j\rvert$ | The determinant of the covariance matrix |
| $\Sigma_j^{-1}$ | The inverse of the covariance matrix |

(https://chrisjmccormick.files.wordpress.com/2014/08/multivariategaussian_eq.png)

The probability that example point i belongs to cluster j can be calculated using the following:

$$w_j^{(i)} = \frac{g_j(x)\phi_j}{\sum_{l=1}^{k} g_l(x)\phi_l}$$

| Symbol | Meaning |
|---|---|
| $w_j^{(i)}$ | The probability that example i belongs to cluster j |
| $g_j(x)$ | The multivariate Gaussian for cluster j |
| $\phi_j$ | The "prior probability" of cluster j (the fraction of the dataset belonging to cluster j) |
| $k$ | The number of clusters |

(https://chrisjmccormick.files.wordpress.com/2014/08/membershipprobability_eq.png)

We'll apply this equation to every example and every cluster, giving us a matrix with one row per example and one column per cluster.

# Maximization

You can gain some useful intuition about the maximization equations if you're familiar with the equation for taking a weighted average. To find the average value of a set of $m$ values, where you have a weight $w$ defined for each of the values, you can use the following equation:

$$\bar{y} = \frac{\sum_{i=1}^{m}(w_i y_i)}{\sum_{i=1}^{m} w_i}$$

(https://chrisjmccormick.files.wordpress.com/2014/02/weightedaverage1.png)

With this in mind, the update rules for the maximization step are below. I've copied these from the lecture notes on GMMs (http://cs229.stanford.edu/notes/cs229-notes7b.pdf) for Stanford's CS229 course on machine learning (those lecture notes are a great reference, by the way).

$$\begin{aligned}
\phi_j &:= \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)}, \\
\mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}
\end{aligned}$$

(https://chrisjmccormick.files.wordpress.com/2014/08/maximizationequations1.png)

The equation for mean (mu) of cluster j is just the average of all data points in the training set, with each example weighted by its probability of belonging to cluster j.

Similary, the equation for the covariance matrix is the same as the equation you would use to estimate the covariance of a dataset, except that the contribution of each example is again weighted by the probability that it belongs to cluster j.

The prior probability of cluster j, denoted as phi, is calculated as the average probability that a data point belongs to cluster j.

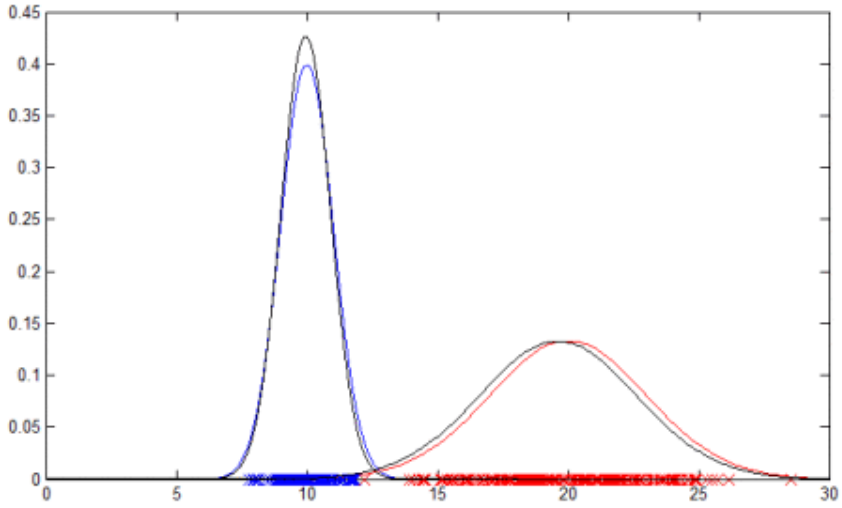
## MATLAB Example Code

I've implemented Expectation Maximization for both a 1D and a 2D example. Run 'GMMExample_1D.m' and 'GMMExample_2D.m', respectively. The 1D example is easier to follow, but the 2D example can be extended to n-dimensional data.

GMM Example Code (https://dl.dropboxusercontent.com/u/94180423/GMM_Examples_v2014_08_04.zip)

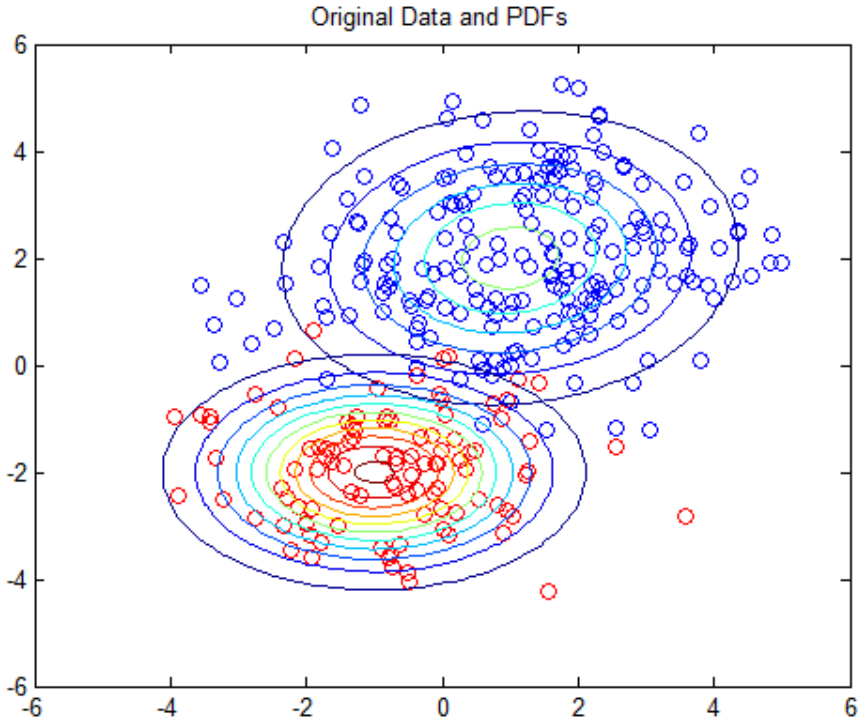
If you are simply interested in using GMMs and don't care how they're implemented, you might consider using the vlfeat implementation, which includes a nice tutorial here (http://www.vlfeat.org/overview/gmm.html). Or if you are using Octave, there may be an open-source version of Matlab's 'fitgmdist' function from their Statistics Toolbox.

The 1D example will output a plot showing the original data points and their PDFs in blue and red. The PDFs estimated by the EM algorithm are plotted in black for comparison.

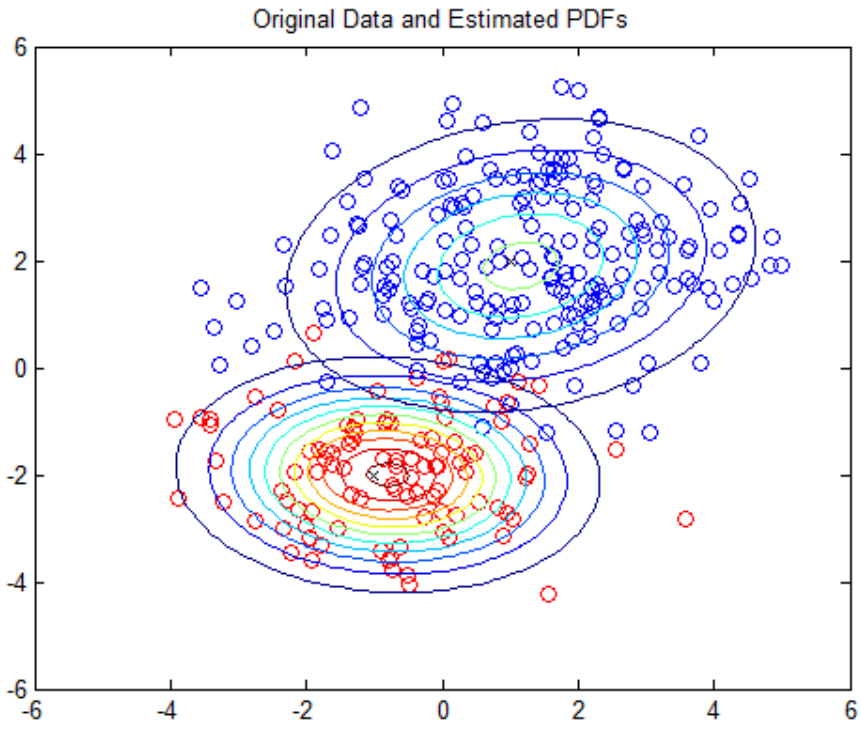



(https://chrisjmccormick.files.wordpress.com/2014/08/1d_example.png)

The 2D example is based on Matlab's own GMM tutorial here (http://www.mathworks.com/help/stats/gaussian-mixture-models.html), but without any dependency on the Statistics Toolbox. The 2D example plots the PDFs using contour plots; you should see one plot of the original PDFs and another showing the estimated PDFs.




(https://chrisjmccormick.files.wordpress.com/2014/08/2d_example_origdata1.png)




(https://chrisjmccormick.files.wordpress.com/2014/08/2d_example_estpdfs.png)



Tags: Clustering, Covariance, Expectation Maximization, Gaussian Mixture Models, Machine Learning, MATLAB, Multivariate Gaussian, Octave, Stanford CS229, Statistics, Unsupervised Learning

# 10 responses to “Gaussian Mixture Models Tutorial and MATLAB Code”

1. *Faraz Khan* September 17, 2014 at 4:21 pm · · Reply →
   Thank you. That was very helpful.

   Can you please also touch on the concept of UBM in GMM. Thank you

2. Pingback: How to give label for cluster from GMM iteration?·

3. *Babacar* September 4, 2015 at 8:32 pm · · Reply →
   Thank you guy. Great

4. *Jane* September 8, 2015 at 1:21 am · · Reply →
   It’s helpful for understanding from k-means to GMMs.