Gaussian Mixture Model

Eric Bridgeford

August 8, 2016

Expectation-Maximization for Multivariate Gaussians

When using the Gaussian Mixture Model for modelling clusters of data points, we consider a sample that can be modelled of the form:

$$p(x) = \sum_c \pi_c N(x|\mu_c, \Sigma_c) \tag{1}$$

This model is a latend variable model, meaning our observed x is drawn from an unobserved z defined such that

$$p(z = c) = \pi_c \tag{2}$$

where π_c is the relative contribution into the joint distribution of the cth normal distrubution defined by:

$$p(x|z = c) = N(x|\mu_c, \Sigma_c) \tag{3}$$

We begin by first guessing our means and covariances of our clusters. These can be predicted through k-means or other clustering algorithms, and estimate our π vector often by just considering that each normal distribution is equally likely.

The EM algorithm for GMM is constructed as follows:

E Step

During the E-step, we consider the potential responsibilities of each cluster on our point. We consider the probability that our point is in a given cluster, and divide it by the total probability that a given point is in any cluster, rather naturally:

$$r_{ic} = \frac{\pi_c N(x|\mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} N(x|\mu_{c'}, \Sigma_{c'})} \tag{4}$$

M Step

During the M-step, we use our newly calculated responsibilities of each normal distribution to our observed data, and maximize our parameters by the following:

The cumulative responsibility of a given cluster c

$$m_c = \sum_i r_{ic} \tag{5}$$

The fraction of data in cluster c

$$\hat{\pi_c} = \frac{m_c}{\sum_c m_c} \tag{6}$$

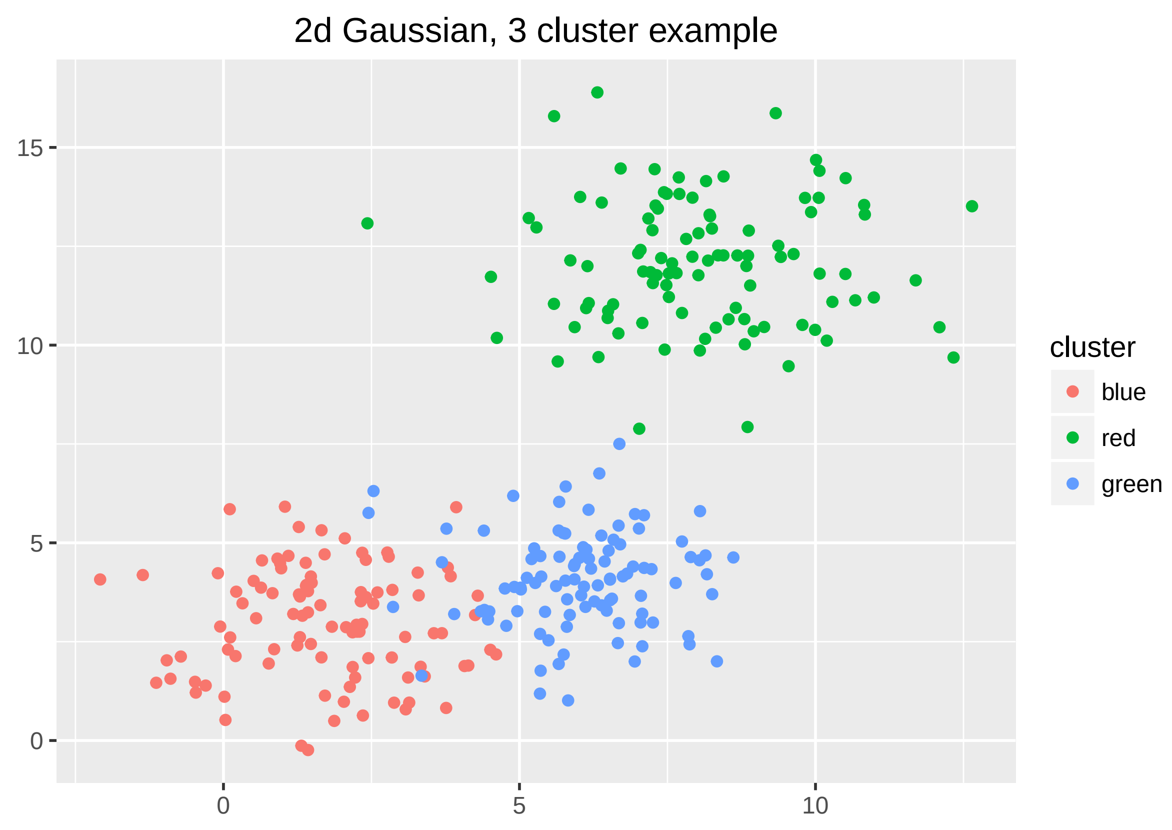
For updated means. The logic behind the mean maximization is that we consider how much a particular cluster "explains" our ith data point, so μ is more strongly impacted by data points that it explains more effectively.

$$\hat{\mu_c} = \frac{1}{m_c} \sum_i r_{ic} x_i \tag{7}$$

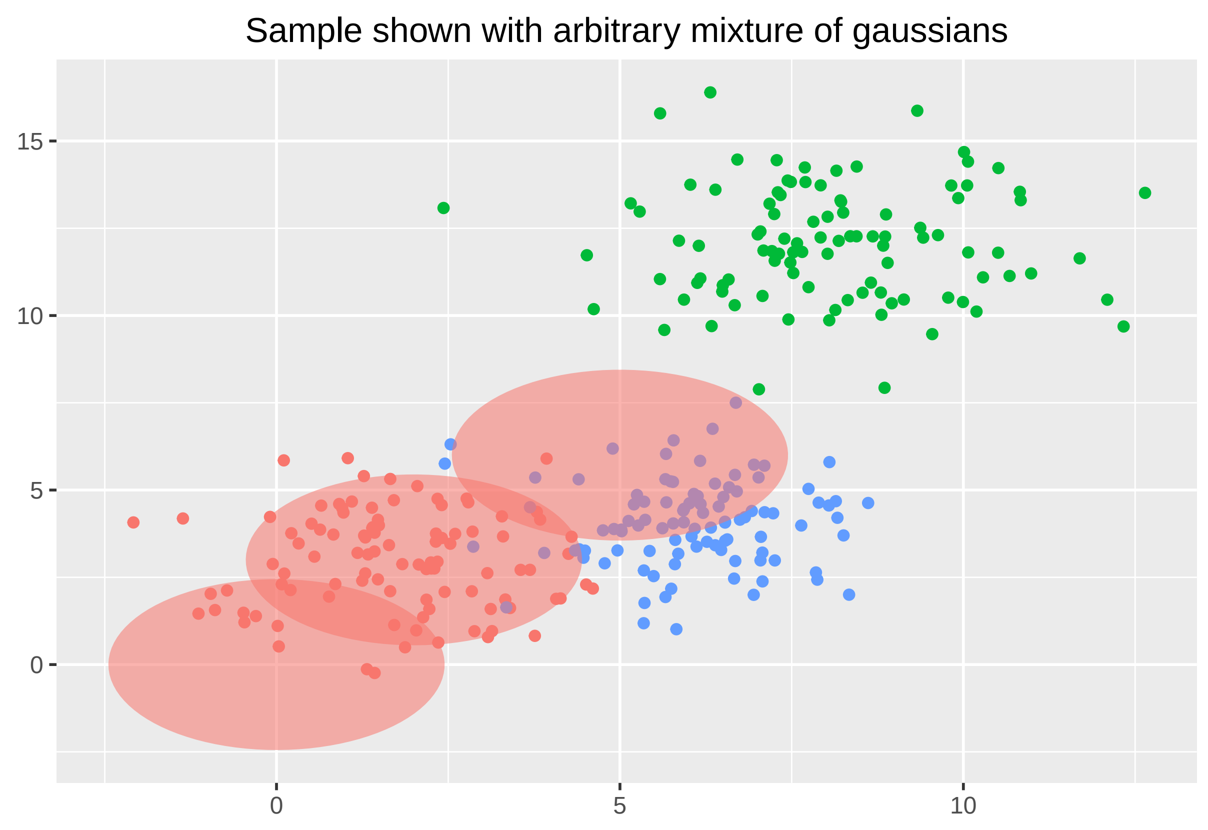
For updating the covariances. The logi here is again the same; if a point is explained well by a particular distribution, its covariance is more strongly considered than if it isn't explained by that distribution.

$$\hat{\Sigma_c} = \frac{1}{m_c} \sum_i r_{ic}(x_i)(x_i - \hat{\mu_c})^T (x_i - \hat{\mu_c}) \tag{8}$$

```r
# 3 cluster example
require(MASS)
require('abind')
source('gmm.R')
require('ellipse')
require('ggplot2')
require(latex2exp)
draw_mus <- array(0, dim=c(2,3))
draw_covs <- array(0, dim=c(2,2,3))
draw_mus[,1] <- c(2,3)
draw_mus[,2] <- c(8,12)
draw_mus[,3] <- c(6, 4)
draw_covs[,,1] <- 2*diag(2)
draw_covs[,,2] <- 3*diag(2)
draw_covs[,,3] <- 1.5*diag(2)
colors <- c('blue', 'red', 'green')

data <- data.frame(x=c(), y=c(), cluster=c())
x <- array(0, dim=c(100, 2, 3))
for (i in 1:3) {
  x[,,i] <- mvrnorm(n=100, draw_mus[,i], draw_covs[,,i])
}
truth_mus <- array(0, dim=c(2,3))
truth_covs <- array(0, dim=c(2,2,3))
for (i in 1:3) {
  truth_mus[,i] <- apply(x[,,i], 2, mean)
  truth_covs[,,i] <- cov(x[,,i])
  data <- rbind(data, data.frame(x[,,i], cluster=colors[i]))
}

x <- abind(x[,,1], x[,,2], x[,,3], along=1)

ggplot(data = data, aes(x = X1, y = X2, color=cluster)) +
  geom_point() +
  ylab("") +
  xlab("") +
  ggtitle("2d Gaussian, 3 cluster example")
```



```
pi_vec <- array(c(.333, .333, .333))
test_mus <- array(0, dim=c(2,3))
test_covs <- array(0, dim=c(2,2,3))
test_mus[,1] <- c(0,0)
test_mus[,2] <- c(5,6)
test_mus[,3] <- c(2,3)
test_covs <- sapply(1:3, function(x) diag(2), simplify='array')

fig <- ggplot() + geom_point(data = data, aes(x = X1, y = X2, color=cluster))
for (i in 1:3) {
  ellipse <- data.frame(ellipse(test_covs[,,i], centre=test_mus[,i]))
  fig <- fig + geom_polygon(data=ellipse, aes(x=x, y=y, fill='black', alpha=0.1, group=i))
}
fig <- fig +
  xlab('') +
  ylab('') +
  ggtitle('Sample shown with arbitrary mixture of gaussians') +
  theme(legend.position = 'none')
fig
```



```r
model <- gmm(x, test_mus, test_covs, pi_vec, tol=0.001)
model$means
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.773056  8.07928 6.133732
## [2,] 3.054514 12.06917 4.009289
```

```r
truth_mus
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.715711  8.08057 6.029628
## [2,] 2.970989 12.06831 4.054378
```

```r
model$covs
```

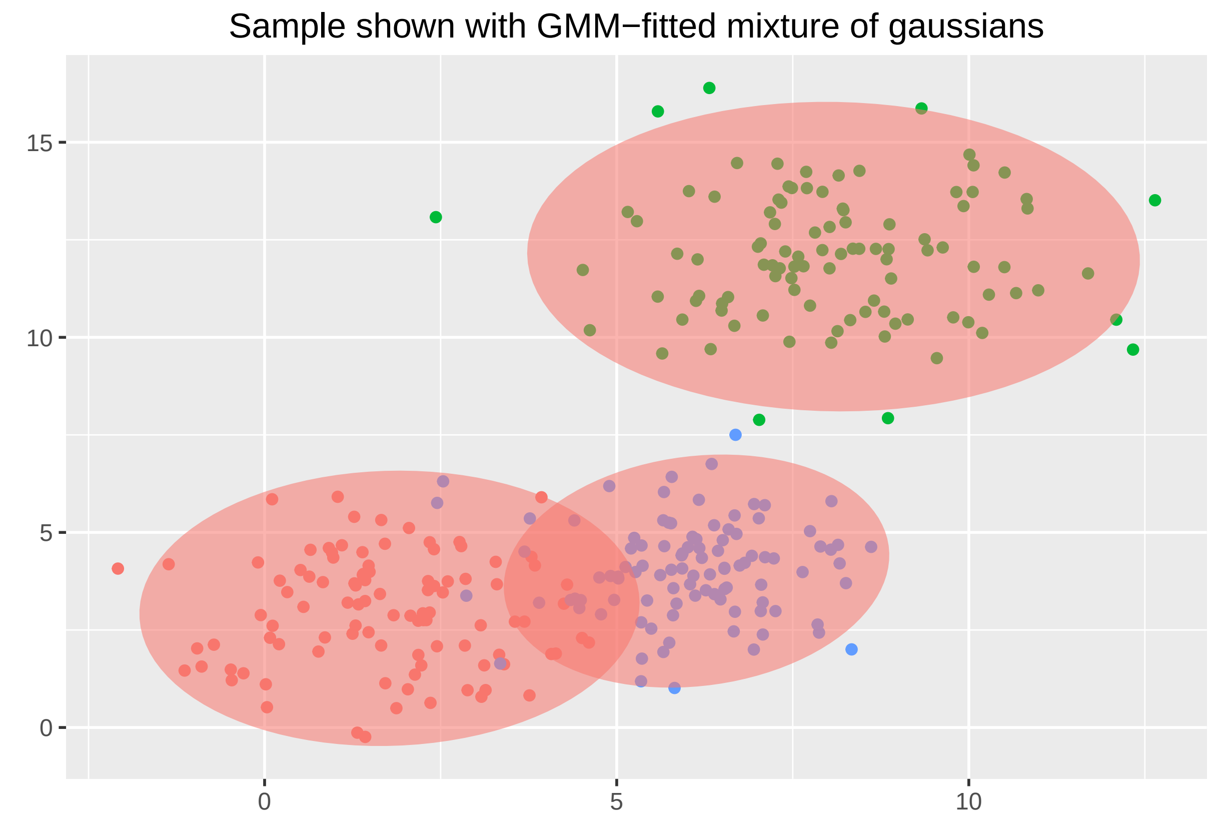
```
## , , 1
##
##            [,1]       [,2]
## [1,] 2.10523746 0.08247066
## [2,] 0.08247066 2.07893135
##
## , , 2
##
##             [,1]        [,2]
## [1,]  3.16000545 -0.07207134
## [2,] -0.07207134  2.62670380
##
## , , 3
```

```
##
##          [,1]     [,2]
## [1,] 1.250902 0.185948
## [2,] 0.185948 1.489236
```

```
truth_covs
```

```
## , , 1
##
##             [,1]        [,2]
## [1,]  2.03626892 -0.03082649
## [2,] -0.03082649  1.95471833
##
## , , 2
##
##             [,1]        [,2]
## [1,]  3.18765152 -0.08032492
## [2,] -0.08032492  2.64819383
##
## , , 3
##
##            [,1]       [,2]
## [1,] 1.56253242 0.01834039
## [2,] 0.01834039 1.51662543
```

```
fig <- ggplot() + geom_point(data = data, aes(x = X1, y = X2, color=cluster))
for (i in 1:3) {
  ellipse <- data.frame(ellipse(model$covs[,,i], centre=model$means[,i]))
  fig <- fig + geom_polygon(data=ellipse, aes(x=x, y=y, fill='black', alpha=0.1, group=i))
}
fig <- fig +
  xlab('') +
  ylab('') +
  ggtitle('Sample shown with GMM-fitted mixture of gaussians') +
  theme(legend.position = 'none')
fig
```



As we can see above, we have 3 normal distributions that we take our data points from. Before running gmm, the distributions are arbitrarily centered, and after gmm, it is clear that we have visually found an effective fir for the data. Note that we are able to effectively model the difficult bottom two clusters; even though the data largely overlaps, we can find the two unqiue clusters with relative precision.