Robby Sneiderman

Follow 40 Followers About

A Quick Introduction to the Expectation Maximization(EM) Algorithm

And how it can be applied to fit assumed distributions.

Robby Sneiderman Nov 10, 2020 · 8 min read

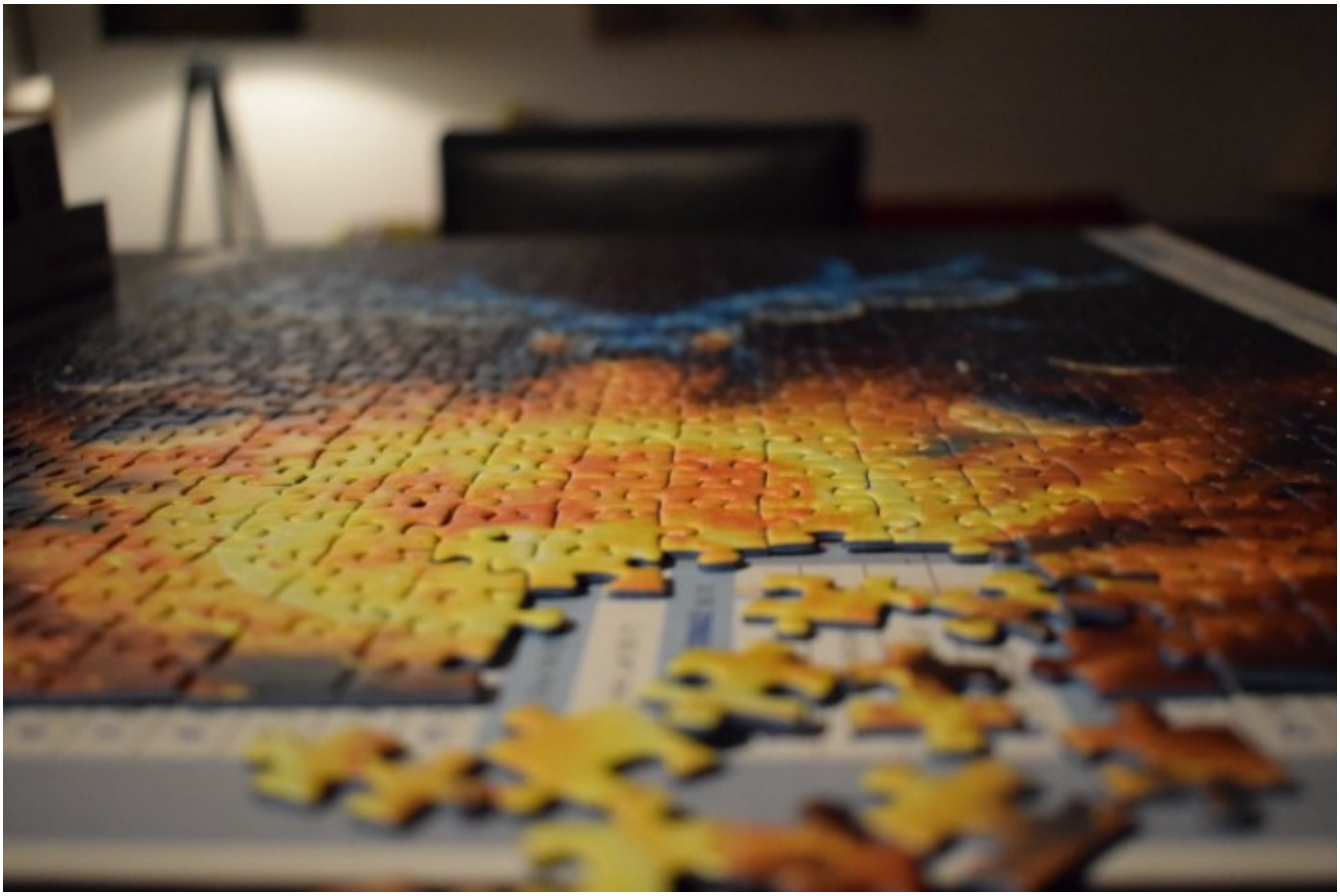


Image Citation: Mor Itam on Unsplash.

and walk through several numerical examples. By the end of this article, you should have a better idea of what the EM algorithm is, why it is useful, and how it can be implemented. The EM algorithm is often said to be used when there is 'missing data', but in general, it can also be viewed as an algorithm to use when there is 'latent' data.

General Idea:

- We maximise parameters given observed data, that is, we want to find what the parameters should be to result in generating the data we observe with the highest probability.

Maximum Likelihood Estimation:

The EM algorithm is a method of *maximum likelihood estimation*, thus we briefly review what that means.

Suppose we have a random sample of **N** observations;

$$y_1, y_2, \cdots, y_N$$

Figure 1: A random sample of size N.

With a proposed density function parameterised by theta;

$$\Pr(y|\theta)$$

Figure 2: The observations y belong to some distribution with parameter theta.

Then the log-likelihood for that sample is;

$$L(\theta) = \sum_{i=1}^{N} \log \Pr(y_i|\theta)$$

Figure 3: For a given parameter theta we can calculate the log likelihood of that sample. it will tell us how likely it is we observed the sample if the parameter of the model were theta.

The EM algorithm works by considering the *conditional expectation* of the log-likelihood. In particular, if we form the expected log-likelihood conditional on the

$$Q(\theta|\theta^{(t)}) = E[\log L(\theta|Y)|x, \theta^{(t)}]$$

Figure 4: The Expected log-likelihood given the observations and the parameters theta.

Which is equivalent to;

$$Q(\theta|\theta^{(t)}) = E[\log f_Y(y|\theta)|x, \theta^{(t)}]$$

Figure 5: An equivalent form.

Which equivalently, can be viewed over all possible missing data.

$$Q(\theta|\theta^{(t)}) = \int [\log f_Y(y|\theta)] f_{z|x}(z|x, \theta^{(t)}) dz$$

Figure 6: An equivalent form considered over a latent variable z.

Then, the EM algorithm works by calculating the conditional expectation (E step) and maximising it (M step).

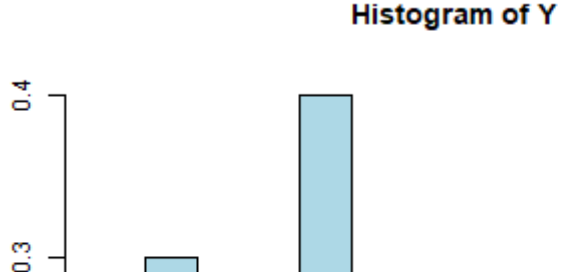
Example 1:

Suppose we were given a random set of 20 observations, **Y** as seen below, and we wanted to fit some distribution to the data.

```
# Consider the toy dataset of 20 observations
Y=c(-0.39,0.12,0.94,1.67,1.76,2.44,3.72,4.28,4.92,5.53,0.06,0.48,1.01,1.68,1.80,3.25,4.12,4.
```

dist.R hosted with ❤ by GitHub view raw

Making a histogram of our 20 observations appears to show two normal distributions.



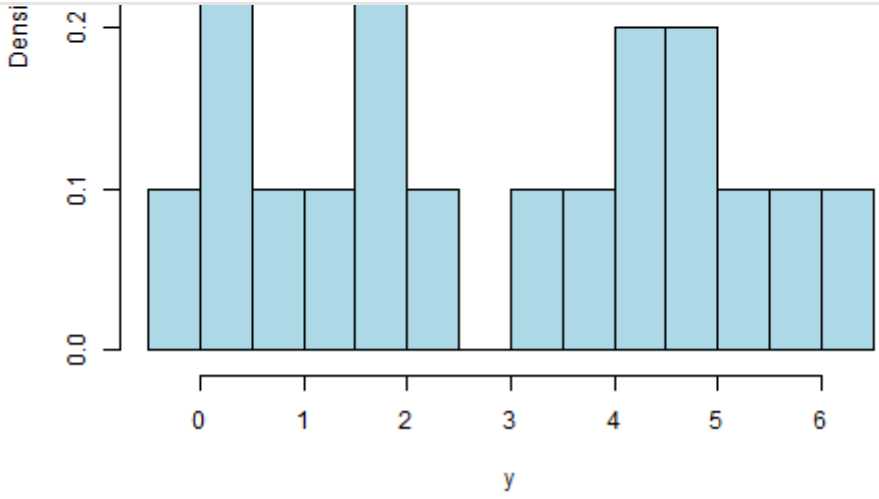


Figure 7: Histogram of our observed data points.

It appears that there are two separate distributions. How could we model this distribution? We could use a mixture model; that is a combination of two normal distributions where;

$$Y_1 \sim N(\mu_1, \sigma_1^2)$$
$$Y_2 \sim N(\mu_2, \sigma_2^2)$$
$$Y = (1 - \Delta)Y_1 + \Delta Y_2$$
$$\Delta \in \{0, 1\}, \Pr(\Delta = 1) = \pi$$

Figure 8: Our proposed possible mixture distribution. We have two normal distributions. We assume that each observation either belongs to the first model, or the second (with a probability , pi, known as the mixing value).

The density of $\mathbf{Y}$ is thus a combination of two normal densities;

$$g_Y(y) = (1 - \pi)\phi_{\theta_1}(y) + \pi\phi_{\theta_2}(y)$$

Figure 9: The density function of our proposed mixture model.

And the log-likelihood of the mixture is the sum of the individual log likelihoods of two normal distributions. This function alone would be very difficult to simply maximize by taking the derivative. It contains log terms with an inner term containing sums.

$$\sum^{N}$$

Figure 10: Maximising this likelihood directly would be extremely difficult as it contains the sum.

Given an observation from **Y**, we can calculate the comparative probability that the observation belongs to one model compared to the other.

$$\gamma_i(\theta) = E(\Delta_i|\theta, \mathbf{Z}) = Pr(\Delta_i = 1|\theta, \mathbf{Z})$$

Figure 11. The 'Responsibility'. The expected value of delta given that we observed the data point.

In the case of a simple Gaussian mixture, the EM algorithm can be greatly simplified. Take a look at the algorithm (provided via *The Elements of Statistical Learning , 2nd edition*). Essentially, we will weight the observation and compare the probability that it belongs to our second model, versus the probability that it belongs to our first model. This is known as the *'responsibility'*. We then update our parameters, the mean and variance for both, and finally we update our guess at the mixing parameter.

Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).
2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \quad i = 1, 2, \ldots, N. \qquad (8.42)$$

3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

   and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.

Figure 12: The EM algorithm is greatly simplified for Gaussian mixture models. Citation: The Elements of Statistical Learning, 2nd Edition. Chapter 8.

random observation). For the variance we use the observed mean to form the common variance estimate.

```
N=length(Y)
y=as.matrix(Y,nrow=N)
# Get Basic summary statistics
# Initalize first guess
# Mixing parameter pi
# Use random sample for random mu, estimator for variance
ybar=mean(Y)
pi=0.5
mu1=sample(Y,1)
mu2=sample(Y,1)
var1=sum(((Y-ybar)^2)/N)
var2=sum(((Y-ybar)^2)/N)
#assign initial guess
init_guess=c(pi,mu1,mu2,var1,var2)
```

initalize.R hosted with ❤ by GitHub view raw

Now that we have an initial guess, we can iteratively compute the responsibility and update the parameters. We can also compute the log-likelihood at each generation using the formula from Figure 10.

```
makeEM=function(Y,parameters,iter){
  #Responsiblity
  responsibility=matrix(0,nrow=N)
  for(j in 1:iter){
  for(i in 1:N){
  model2=parameters[1]* dnorm(Y[i],mean=parameters[3],sd=sqrt(parameters[5]))
  model1=(1-parameters[1])*dnorm(Y[i],mean=parameters[2],sd=sqrt(parameters[4]))
  responsibility[i]=model2/(model1+model2)
}
  R=responsibility
  mu1=sum((1-R)*Y)/sum(1-R)
  mu2=sum(R*Y)/sum(R)
  var1=sum((1-R)*(Y-mu1)^2)/sum(1-R)
  var2=sum(R*(Y-mu2)^2)/sum(R)
  pi=sum(R)/N
  parameters=c(pi,mu1,mu2,var1,var2)
}
  return(parameters)
}
```

Once we have run the algorithm for 30 iterations, we can see what a mixed distribution with our expected parameters would look like;

```r
#library to plot mixed distributions
library(distr)
myMix <- UnivarMixingDistribution(Norm(mean=mu1, sd=sqrt(var1)), Norm(mean=mu2, sd=sqrt(
rmyMix <- r(myMix)
# Sample a million random variates
x <- rmyMix(1000000)
# Histogram of mixture
hist(x[x>-0.5 & x<6.5], breaks=50, col="red", main="",xlab='y',freq=FALSE,ylab='Density'
```

makemix.R hosted with ❤ by GitHub view raw

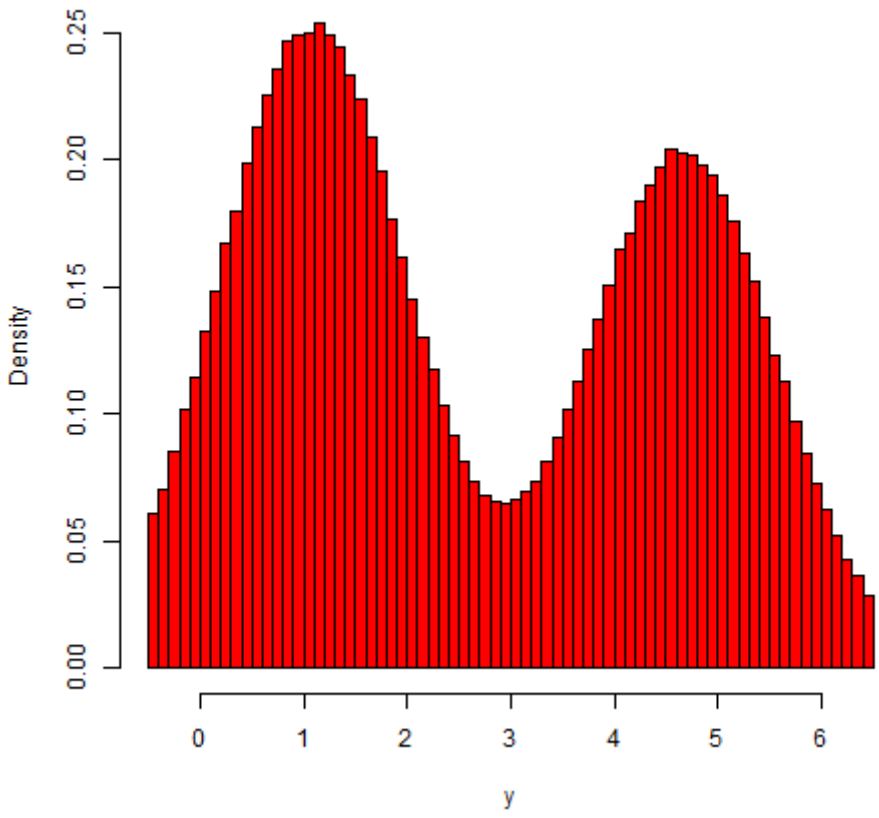


Figure 13 : The histogram of our mixture model, which appears to fit the initial observations quite well.

We can also plot the log-likelihood (evaluated per each iteration).

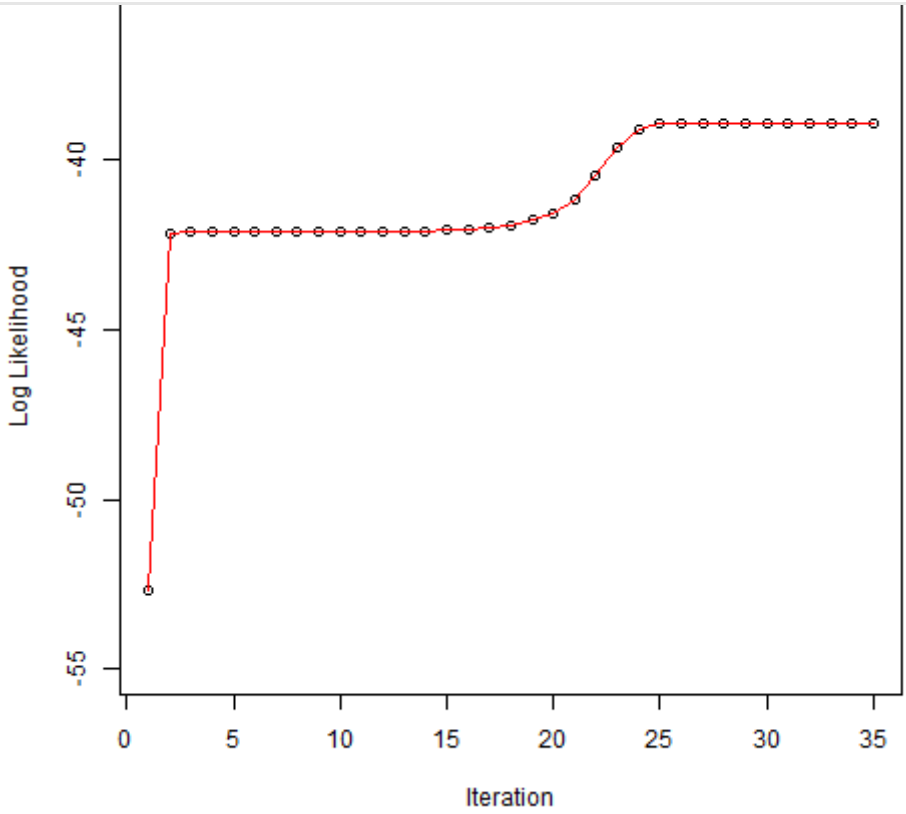
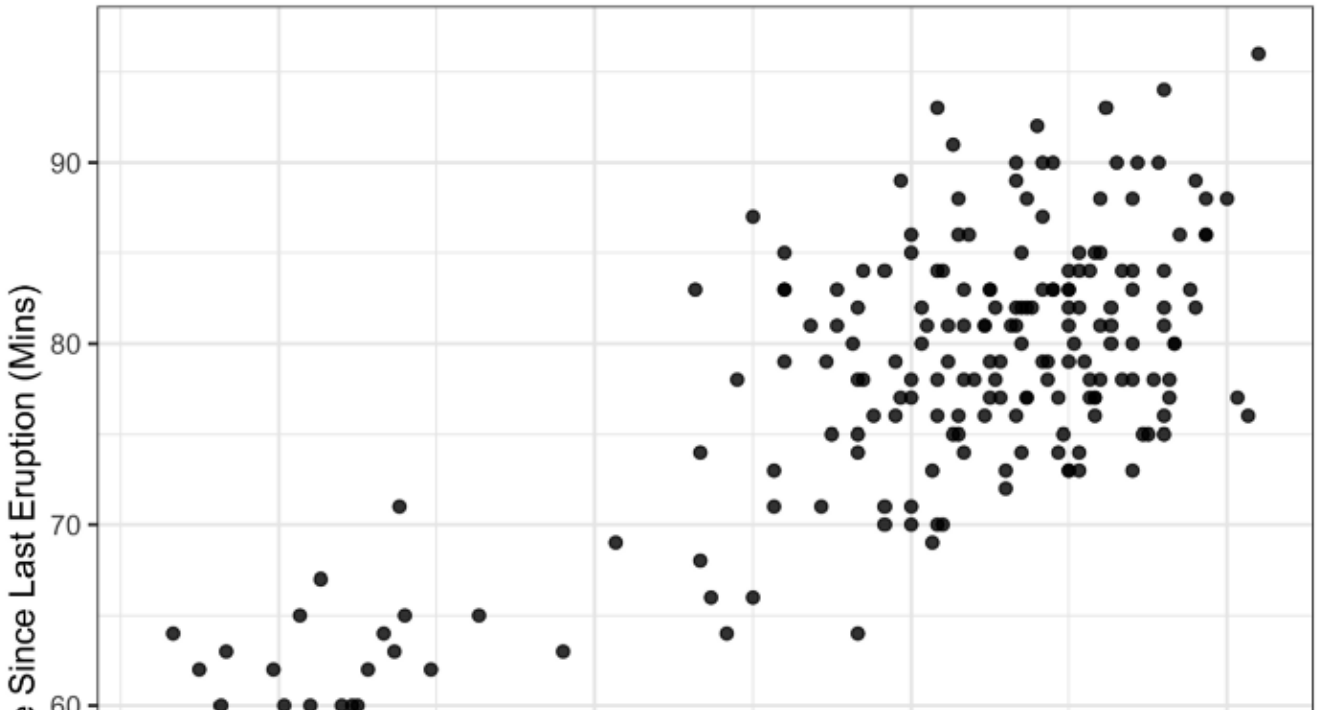


Figure 14: The log-likelihood per each iteration, in general it increases, but it never decreases.

Old Faithful:

Another common example used to illustrate the EM algorithm is in its application to the old faithful dataset. This dataset consists of time in minutes between each eruption (y-axis) versus the duration of the eruption (x-axis).



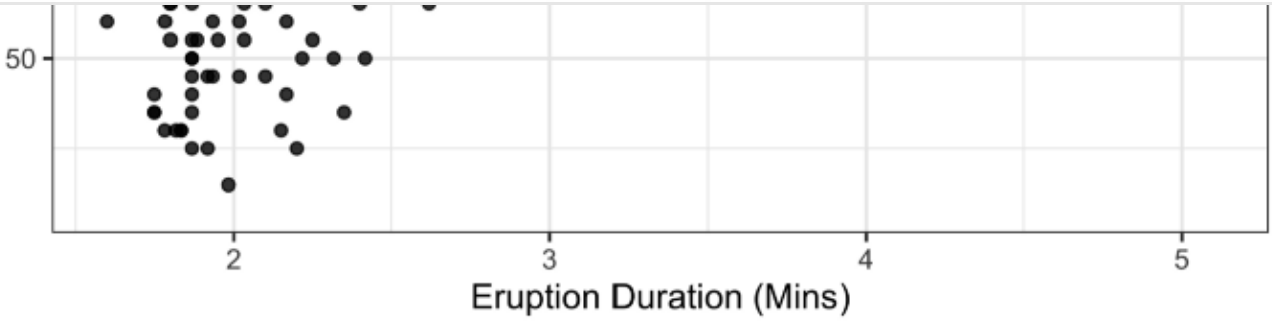


Figure 15: It looks like there are two clusters of points. In general, there is a clear trend; the longer we wait between each eruptions seems to result in longer duration eruptions.

We can obtain better estimates of the clusters by *assuming there are two multivariate normal distributions*. Once we naively make an assumption of a starting distribution, we use the density of the multivariate normal to preform the EM Algorithm.

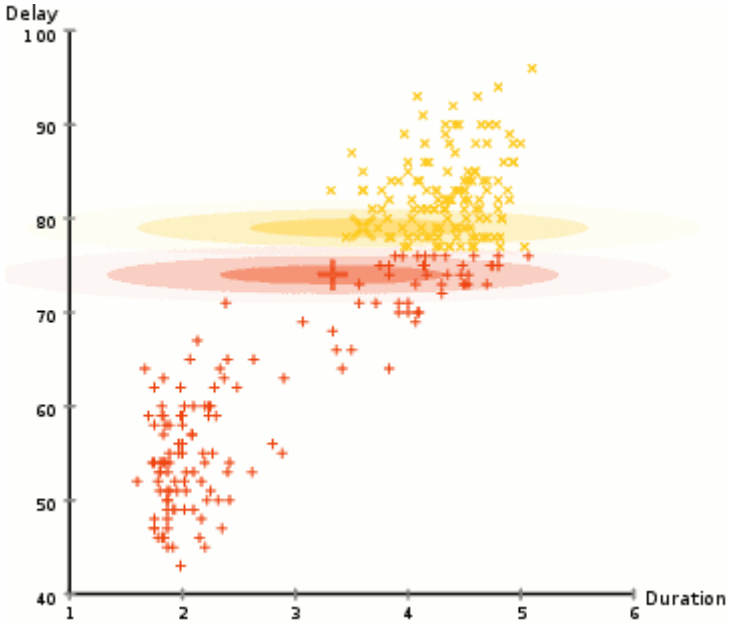


Figure 16: EM algorithm running on the old faithful dataset with two multivariate normal distributions. Gif Citation: Creative Commons: Chire, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

The above animation illustrates the steps taken. We start off with two naive and badly fit multivariate normal distributions. Each step of the EM algorithm updates the parameters according to which distribution is more likely. In the end we have two well fitting multivariate normal.

Peppered Moths:

C is dominant to I and T, and T is recessive to I. There are three distinct *phenotypes* of these moths (that is , we can actually observe three different types).

***CC, CI* and *CT* result in a solid black moth (*carbonaria*).**

TT* results in an light moth (*typica).

***IT* and *II* result in a intermediate coloured moth (*insularia*).**

Suppose we capture a large number of these months, say **N** of them, and we can then count how many of each phenotype we see. Hence, denote **N=nC+nI+nT**

While we can see the colour of the moth, we can't actually see the underlying genotype. How can we get an idea of the distribution of alleles? Firstly, note that since only TT results in a light moth, we can simplify the problem. We know that there are **nT** moths with genotype TT.

What is seen:

$$x = (n_C, n_I, n_T)$$

Figure 17: The Observed data, the three phenotypes. We can only see the three types of moths.

What is 'missing' or latent:

The Genotype (there are six possible);

$$G = (n_{CC}, n_{CI}, n_{CT}, n_{II}, n_{IT}, n_{TT})$$

Figure 18: In reality, there are 6 underlying genotypes.

If the alleles were distributed with probability PC , PI and PT respectively, then the probability of each genotype could be represented by Gp.

$$Gp = (P_C^2, 2P_CP_I, 2P_CP_T, P_I^2, 2P_IP_T, P_T^2)$$

This distribution can be viewed as a *multinomial* distribution, with each allele being chosen with probability according Gp.

Then proceeding as before, to apply the EM algorithm, we need the log- likelihood of the multinomial distribution

The probability density function (pdf) of the multinomial distribution in this case is thus;

$$f_{Y|p} = \frac{n!}{(n_{CC}!\bullet n_{CI}!\bullet n_{CT}!\bullet n_{II}!\bullet n_{IT}!\bullet n_{TT}!)}(P_C^2)^{n_{CC}}(2P_CP_I)^{n_{CI}}(2P_CP_T)^{n_{CT}}(P_I^2)^{n_{II}}(2P_IP_T)^{n_{IT}}(P_T^2)^{n_T}$$

Figure 20: The pdf of the multinomial distribution.

And the conditional log likelihood is;

$$\log f_{Y|p} = \log \frac{n!}{(n_{CC}!\bullet n_{CI}!\bullet n_{CT}!\bullet n_{II}!\bullet n_{IT}!\bullet n_{TT}!)} + n_{CC}\log(P_C^2) + n_{CI}\log(2P_CP_I) + n_{CT}\log(2P_CP_T) + n_{II}\log(P_I^2) + n_{IT}\log(2P_IP_T) + n_T\log(P_T^2)$$

Figure 21: The conditional log-likelihood.

Above is the conditional log-likelihood, and we need to compute the expected value conditional on what we observe.

Expectation Step:

The expected number of CC alleles will be proportional to the number of *carbonaria* counted and the probability of being CC, by the probability of appearing *carbnoaria.* By the same logic we can obtain all the conditional expectations. These are then replaced in the above formula and we take the derivative, set it to zero to solve for the update steps.

$$E[N_{CC}|n_C, n_I, n_T, p^{(t)}] = n_{CC}^{(t)} = n_C * \frac{(P_C^{(t)})^2}{(P_C^{(t)})^2 + 2P_C^{(t)}P_I^{(t)} + 2P_C^{(t)}P_T^{(t)}}$$

Figure 22: The Expected number of genotypes of the form CC.

Figure 23: The Expected number of genotypes of the form CI.

$$E[N_{CT}|n_C, n_I, n_T, p^{(t)}] = n_{CI}^{(t)} = n_C * \frac{(2 * P_C^{(t)} P_I^{(t)})}{(P_C^{(t)})^2 + 2P_C^{(t)} P_I^{(t)} + 2P_C^{(t)} P_T^{(t)}}$$

Figure 24: The Expected number of genotypes of the form CI.

$$E[N_{II}|n_C, n_I, n_T, p^{(t)}] = n_{II}^{(t)} = n_I * \frac{(P_I^{(t)})^2}{(P_I^{(t)})^2 + 2P_I^{(t)} P_T^{(t)}}$$

Figure 25: The Expected number of genotypes of the form II.

$$E[N_{IT}|n_C, n_I, n_T, p^{(t)}] = n_{IT}^{(t)} = n_I * \frac{2 * P_I^{(t)} P_T^{(t)}}{(P_I^{(t)})^2 + 2P_I^{(t)} P_T^{(t)}}$$

Figure 26: The Expected number of genotypes of the form IT.

Note, we already know the expected number of TT genotypes, as we can actually observe it(the number of light moths).

Maximization Step:

We can take the derivative of the log-likelihood to obtain the maximization step.

$$\frac{d \log f_{Y|p}}{d_{P_C}} = \frac{2n_{CC}^{(t)} + n_{CI}^{(t)} + n_{CT}^{(t)}}{P_C} - \frac{2n_{IT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{1 - P_C - P_I}$$

Figure 27: Maximising the expected log likelihood with respect to Pc.

$$\frac{d \log f_{Y|p}}{d_{P_I}} = \frac{2n_{II}^{(t)} + n_{IT}^{(t)} + n_{CI}^{(t)}}{P_I} - \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{1 - P_C - P_I}$$

Simplifying ,and noting that $P_C+P_I+P_T=1$.

$$P_C^{(t+1)} = \frac{2n_{CC}^{(t)} + n_{CI}^{(t)} + n_{CT}^{(t)}}{2n}$$

$$P_I^{(t+1)} = \frac{2n_{II}^{(t)} + n_{IT}^{(t)} + n_{CI}^{(t)}}{2n}$$

$$P_T^{(t+1)} = \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{2n}$$

Figure 29: The updates for each iteration. After solving for each parameter.

We can now use these steps to form the algorithm and iteratively update each step.

```
# We observe X, we guess an initial P
X = c(74, 196, 341)
P = c(1/3,1/3,1/3)

expectationMaximization=function(x,p,iter){
for(i in 1:iter){
# Expectation Step
    n_CC = (x[1]*(p[1]^2))/((p[1]^2)+2*p[1]*p[2]+2*p[1]*p[3])
    n_CI = (2*x[1]*p[1]*p[2])/((p[1]^2)+2*p[1]*p[2]+2*p[1]*p[3])
    n_CT = (2*x[1]*p[1]*p[3])/((p[1]^2)+2*p[1]*p[2]+2*p[1]*p[3])
    n_II = (x[2]*(p[2]^2))/((p[2]^2)+2*p[2]*p[3])
    n_IT = (2*x[2]*p[2]*p[3])/((p[2]^2)+2*p[2]*p[3])
    n= c(n_CC,n_CI,n_CT,n_II,n_IT,x[3])
# Maximization Step
    p_C = (2*n[1]+n[2]+n[3])/(2*sum(x))
    p_I = (2*n[4]+n[5]+n[2])/(2*sum(x))
    p_T = (2*n[6]+n[3]+n[5])/(2*sum(x))
    p_total = c(p_C,p_I,p_T)
    p=p_total
}
    return(p)
}
expectationMaximization(X,P,40)
[1] 0.06251023 0.19042788 0.74706189
```

emmoth.R hosted with ❤ by GitHub view raw

Given an initial observation of 74 black moths, 170 intermediate and 321 light moths, we can use the EM algorithm to get an idea of the probability distribution of each allele. From the initial observations, we expect that the allele C will be quite rare. The T allele is extremely common, and the I allele is also quite rare. This is because for there to be so many light moths, there must be an overwhelming presence of the T allele. After applying the algorithm we estimate that the probability of having a C is only 0.0625, of I is 0.1904 and T is 0.747, which makes sense.

Convergence:

It can be proven that the EM algorithm does converge. That is, that each step does not decrease the log-likelihood. Hence at a minimum it makes no change, but it will in general increase the log-likelihood and converge at the maximum.

Summary and Conclusion:

The EM Algorithm is a powerful tool that is used when there is underlying or latent data. It is thus crucial that data scientists and statisticians are at least familiar with the algorithm. The EM algorithm is used extensively in industry, EM is frequently used for clustering in also is used in computer vision.

The Baum-Welch algorithm, a form of the EM algorithm is also used extensively in Natural langue processing where it deals with hidden Markov models.

The EM algorithm is also widely used in for medical image reconstruction, especially in PET scans and XRays.

Sources:

[1] Dempster, Laird and Rubin (1977) J Royal Statistical Society (B) Maximum Likelihood from Incomplete Data via the EM Algorithm.

[2]Computational Statistics, 2nd edition. Geof H Givens, J A. Hoeing (2012).

[3] The Elements of Statistical Learning, II Edition (2009).Hastie, Tibshirani and Friedman.

[4]Geoffrey J. McLachlan, Thriyambakam Krishnan (2007). The EM Algorithm and Extensions

[6] Current trends in medical image registration and fusion (2016). Egyptian Informatics Journal

Github:

https://github.com/Robby955/ExpectationMaximization

Expectation Maximization Machine Learning Algorithms Statistics Data Science