Senior Thesis in Mathematics

The Expectation Maximization Algorithm & RNA-Sequencing

Author:
Maria Martinez

Advisor:
Dr. Johanna S. Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

May 5, 2017

Abstract

Next-generation sequencing (NGS) technologies, primarily RNA-seq technology has made great advancements in the study of gene expression. Gene expression helps us cluster genes based on how similar their gene expression are, and therefore we not only gain a better understanding on how and which of these genes are related to one another based on their gene expressions, but also we gain deeper knowledge on the purpose and function of these genes. Here we describe a clustering algorithm, Expectation Maximization, which allows us to improve the performance of these models compared to a heuristic approach such as the k-means algorithm.

Contents

Chapter 1

Introduction

When estimating a parameter from a population, say the mean, one of the most common statistical methods used is Maximization Likelihood Estimation (MLE). MLE is a powerful tool that gives estimates of the parameters that maximize the likelihood of your observed data occurring. In more simple words, if you have a dataset of heads and tails from flipping a coin 100 times, the MLE would give you a parameter that maximize the likelihood of the specific data occurring. Now this is a good example, but now imagine we were trying to solve a similar problem where we have lost the information on whether the flip came from population A or population B and it is important to know the probability of heads for both populations. Simply solving for the MLE is no longer straightforward given that we have more parameters to estimate. An algorithm that permits us to find the parameters that maximize our data with missing information is known as the Expectation Maximization Algorithm.

1.1 Expectation Maximization Algorithm

In this chapter we provide an explanation of the Expectation Maximization Algorithm, better known as the EM Algorithm, and give a derivation to detail some of the intuition behind the algorithm.

To fully grasp the EM Algorithm one has to understand when it is needed. To do so, we will first introduce an example where the EM Algorithm is not needed. Suppose we have i.i.d observations X_i^n from a Normal Distribution that represent the hours people spend watching Netflix. Assume the data

comes with an unknown mean μ and a known variance σ^2. In order to estimate our parameter, μ, or the average hours spent by the population watching Netflix, our best attempt is to find the MLE for that parameter. We can do this by finding the derivative of our likelihood function and solving for μ by setting the derivative equal to 0. In this particular Netflix example we rely on the normal distribution, however, data does not always come from a normal distribution and a more appropriate distribution can be used. In particular we focus later on the Gaussian Mixture Models (GMM) where data come from a variety of groups each with its own distribution. This leads us closer to what will be examined in this paper; for example, the average hours a population spends watching Netflix will look different if you consider there being two distinct normal distributions; one of a female population and one of males.

The previous Netflix example relied on the existence of only one parameter: an unknown μ. Now let's extend the example where there exist more than one unknown parameter. Suppose we have n random variables, X_i^n. Each belong to an individual person in our dataset and represent the amount of hours a person watched Netflix per week. The separate datasets are assumed to have the same variance σ^2 but different unknown means, μ_M and μ_F. The two different means come from the fact that we now have an unobserved label, C_i, that represents coming from the male or female populations. Because of this unknown label, we do not know which population each our X_i come from and therefore, we can not use standard MLE steps. The following Bernoulli distribution helps characterize the breakdown of the females and males:

$$p_{c_i}(c_i) = q^{\mathbb{1}(c_i=M)}(1-q)^{\mathbb{1}(c_i=F)} \tag{1.1}$$

Equation 1.1 uses the indicator function to provide the probability of c_i which is the probability of coming from either the female or male population. To simplify, we make $\pi_M = q$ and $\pi_F = 1 - q$. At first glance the product function gives the impression that we are working with multiple groups, but in our simple case provided we only have two possible groups (female or male). However, later on we will be working with more than one group.

$$p_{c_i}(c_i) = \prod_{c \in \{M,F\}} \pi_c^{\mathbb{1}(c_i=c)} \tag{1.2}$$

In equation 1.2 we assumed that the conditional distributions of each

class are Gaussian. In other words, the distribution of the length of time someone spends on Netflix, given that they are female is approximated with the normal distribution. The same applies to the male population. The difference is the parameter μ

1.1.1 Can one Find the MLE?

In unsupervised learning, unlike in supervised learning, there are no labels used when building a model; there is no training and testing your data. Instead you attempt to find patterns in a dataset that could provide insight about the data with respect to the group as well as with respect to the model. On the other hand, you have supervised learning where you use the labels provided in order to build a model. Unsupervised learning tends to be harder than supervised. Continuing with the Netflix example, the unknown labels represent the unsupervised learning nature of our problem. The gender of Y_i is the hidden information in the model building stage. The hidden information is what makes this problem harder to solve in closed form.

When we are not provided with the labels, our goal is find a way to maximize the likelihood in order to estimate the Gaussian parameter values including their label probabilities. This can be done by estimating the parameters μ_F and μ_M and the class membership of the observations *simultaneously*. To demonstrate that we need to find all of these parameters in order to successfully solve our problem, we will first attempt the standard way of finding the MLE by computing the joint density, logging the function and then differentiating in terms of either μ. The complete data is represented by the following joint density:

$$P_Y(y) = \prod_{i=1}^{n} (\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)) \tag{1.3}$$

We log the function:

$$\ln(P_Y(y)) = \sum_{i=1}^{n} \ln(\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)) \tag{1.4}$$

Next, we differentiate with respect to one of our μ's, in this case we choose μ_F. Differentiating gives us equation 1.5:

$$\sum_{i=1}^{n} \frac{1}{\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)} \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2) \left(\frac{y_i - \mu_F}{\sigma^2} \right) = 0 \tag{1.5}$$

Equation 1.5 cannot be used to solve for μ_F in closed form. It deals with too many terms at once such as exponentials and linear terms that we are unable to come up with a closed solution. Instead we must use another approach to solve equation 1.5. Knowing we cannot solve with regular MLE methods, we realize this is a much harder problem that can be resolved through the Expectation Maximization (EM) Algorithm.

1.1.2 Expectation Maximization as an Alternative

As seen above, having missing information in the labels prevents us from finding our MLE using calculus methods to differentiate. Instead we use alternative iterative methods such as EM Algorithm to take into consideration the unknown classes. Ideally, in our example, we want to find all the data values where $C_i = M$ to estimate μ_M and use the remaining points, where $C_i = F$, to estimate μ_F. To attempt this, first we look at the distribution of C_i. Bayes' Rule states that $\mathrm{P}(A|B) = \frac{P(B|A)P(A)}{P(B)}$. We use Bayes' rule to write:

$$P_{C_i|Y_i}(c_i|y_i) = \frac{P_{Y_i|C_i}(y_i|c_i)P_{C_i}(c_i)}{P_{Y_i}(y_i)} \tag{1.6}$$

In equation 1.6, we interpret $P_{C_i|Y_i}(c_i|y_i)$ as the probability of being in a certain group (in our case, male or female) given that you have a specific feature. For example, the probability of being female *given* that you spend 8 hours watching Netflix. This sounds like the opposite to what we are trying to find, but this calculation will help us find what we are truly looking for which is estimating μ_F.

$P_{C_i}(c_i)$ is defined as the unconditional probability of belonging to a certain group (e.g. male or female) and $P_{Y_i}(y_i)$ is the probability of spending 8 hours watching Netflix (regardless of gender) without the loss of generality. Equation 1.6 is just a generic form of the equation. Now using equation 1.3 to rewrite our denominator and equation 1.2, equation 1.6 can be rewritten for a better understanding of what is going on as seen below (notice we assumed

a normal distribution):

$$P_{C_i|Y_i}(c_i, y_i) = \frac{(\pi_c \mathcal{N}(y_i; \mu_c, \sigma^2))^{\mathbb{1}(c_i=c)}}{\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)} = q_{C_i}(c_i) \tag{1.7}$$

We rename this to simply $q_{C_i}(c_i)$ for notation purposes, but keep in mind that $q_{C_i}(c_i)$ is a function of *all* of the parameters. If we let $c_i = F$ for female group we get the following posterior probability:

$$P_{C_i|Y_i}(F|y_i) = \frac{\pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)}{\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i); \mu_F, \sigma^2)} = q_{C_i}(F) \tag{1.8}$$

In essence we are a step closer to solving one of our original problems; originally we were unable to solve for our parameters μ_M and μ_F because we did not know which distribution our points where coming from. Now we have $q_{C_i}(c_i)$, the probability of class given the observed data, to help us solve for the unknown as we will soon discuss. In equation 1.7 we found a simpler way of solving by rewriting the equation with q_{C_i}. This allows us to "solve" for μ_F without actually solving for it since we will be ignoring that q_{C_i} depends on μ_F.

In equation 1.9 we have a function that will help us estimate μ_F in an iterative way. This is the *maximization* step our algorithm.

$$\begin{aligned} \sum_{i=1}^{n} q_{C_i}(F) \frac{y_i - \mu_F}{\sigma^2} &= 0 \\ \frac{\sum_{i=1}^{n} q_{C_i}(F) y_i}{\sum_{i=1}^{n} q_{c_i}(F)} &= \hat{\mu_F} \end{aligned} \tag{1.9}$$

Note that μ_F is a weighted average of y_i. If we go back to our Netflix example, μ_F is the weighted average of the hours spent watching Netflix and each time spent is weighted by the probability of being female. We could find μ_M with the equivalent equation but instead with the probability of being male.

Note that to solve for $q_{C_i}(c_i)$ in equation 1.7 we need μ_F, but to solve μ_F in equation 1.9 we need $q_{C_i}(c_i)$ that is that in order to solve for the parameters, we need to find the posterior probabilities, $q_{C_i}(c_i)$, and in order to find the posteriors we need the parameters, μ_F and μ_M. This circular way of solving for the unknown leads us back to the EM Algorithm. Through the 2-step iteration we are able to fix one and solve for the other and vice

versa. We initialize the missing parameters randomly and iterate the steps until convergence. To summarize, the algorithm goes as follows:

Algorithm 1 Expectation Maximization Algorithm

1: *Initialization:* Randomly choose values for your parameters.
2: *Expectation:* The E step where you re-calculate probabilities of membership to every distribution.
3: *Maximization:* The M where you re-calculate parameters by maximizing the likelihoods
4: Iterate between steps 2-3 until convergence

So far we have found a way to solve for our parameters, but we are left wondering if this will give us the parameters that maximize the likelihood of our data. For that, we will derive the EM algorithm to show how it optimizes our parameters. We also will explore finding an intuitive way to initialize that randomly assigns group membership.

Chapter 2

Derivation of the Expectation Maximization

From the the previous chapter, it seems amazing that by modifying a few things we are able to iteratively solve the likelihood maximization problem for our parameters. However, as all good algorithms go, we have technically not shown that our algorithm give us an optimal solution. The next question to answer is: how do the previous equations actually help us find the MAXIMUM LIKELIHOOD ESTIMATE?

First, we will set up our problem as before. Suppose we have an observed random variable Y and hidden variable C. We know that the distribution of Y depends on C and that we are interested in finding the unknown parameter θ which comes from the distributions of C and Y. Now, in our example from before we stated that $Y = Y_1, \ldots, Y_n$ indicated the time individuals spent watching Netflix and $C = C_1, \ldots, C_n$ were the hidden gender labels of each individual respectively. Also, θ was the vector of all our missing parameters, previously μ_F and μ_M, the average hours spent on Netflix for each gender.

2.1 Derivation

Our ultimate goal is to maximize the log likelihood of our observed data:

$$\log(\mathrm{P}_Y(y;\theta))$$

By the definition of marginal probability, we are able to rewrite the log likelihood as $\log\left(\sum_c P_{Y,C}(y,c;\theta)\right)$. We rewrote the log likelihood of our observed data (or having spent X amount of hours on Netflix) as the sum of

women and men who watch X hours since we have those two possible groups. From here, we multiply by 1 by multiplying by $q_c(c)/q_c(c) = 1$. Next we turn the equation into an expectation problem. Recall that we defined $q_c(c)$ to be the posterior probability (i.e. conditional on Y) distribution updated with the data.

$$\begin{aligned}
\log P_Y(y;\theta) &= \log\left(\sum_c \frac{q_C(c)}{q_C(c)} \times P_Y(y;\theta)\right) && \text{Multiplied by 1} \\
&= \log\left(\sum_c q_C(c)\frac{P_{Y,C}(y,c;\theta)}{q_C(c)}\right) && \text{Marginalized with respect to C} \\
&= \log\left(\mathbb{E}_{q_c}\left[\frac{P_{Y,C}(y,c;\theta)}{q_C(c)}\right]\right) && \text{Made into an expectation wrt } q_C(c)
\end{aligned}$$

Now we introduce a theorem better known as Jensen's Inequality that permits us to put a lower bound to estimate the above equation. It also simplifies our math since instead of logging an expected value, we do the opposite.

Theorem 2.1. *Jensen's Inequality* states:

$$\log\left(\mathbb{E}[X]\right) \geq \mathbb{E}\left[\log(X)\right]$$

Using Jensen's inequality we rewrite the above equation as:

$$\log\left(\mathbb{E}_{q_c}\left[\frac{P_{Y,C}(y,c;\theta)}{q_C(c)}\right]\right) \geq \mathbb{E}_{q_c}\left[\log\left(\frac{P_{Y,C}(y,c;\theta)}{q_C(c)}\right)\right] \tag{2.1}$$

From here on we work with the right side of the inequality since we have been provided with a lower bound to our original log equation that we aimed to maximize. Using log rules, we rewrite the right side of inequality 2.1 as:

$$\mathbb{E}_{q_c}\left[\log\left(\frac{P_{Y,C}(y,c;\theta)}{q_C(c)}\right)\right] = \mathbb{E}_{q_c}[\log(p_{Y,C}(y,c;\theta))] - \mathbb{E}_{q_c}[\log(q_C(C))] \tag{2.2}$$

Our end goal is to optimize the likelihood with respect to our parameters, θ. Equation 2.2 allows us to do just that (it actually is the maximization step). So by maximizing 2.2 we maximize our log likelihood. By rewriting,

we see that only the term $\mathbb{E}_{q_c}[\log(p_{Y,C}(y,c;\theta))]$ depends on θ and therefore we can ignore the other term. We eliminate that term and optimize equation 2.3.

$$\hat{\theta} \leftarrow \underset{\theta}{\operatorname{argmax}}(\mathbb{E}_{q_c}\left[\log(P_{Y,C}(y,C;\theta))\right]) \tag{2.3}$$

Now if we assume to know $q_C(c)$, we can find $\hat{\theta}$. Recall from equation 1.9, we defined $\hat{\mu_M}$ as:

$$\hat{\mu_M} = \frac{\sum_{i=1}^{n} q_{C_i}(M) y_i}{\sum_{i=1}^{n} q_{c_i}(M)}$$

We show the above equation is true an it is the parameter that optimizes our likelihood (2.3) with the following derivation.

$$\begin{aligned}
0 =& \frac{\partial}{\partial \mu_M} \sum_{i=1}^{n} \sum_{c \in \{M,F\}} \mathbb{E}_{q_c}[\mathbb{1}(C_i = c)] \left[\log(P_{Y,C}(y,C;\theta))\right] \\
=& \sum_{i=1}^{n} \frac{\partial}{\partial \mu_M} \left[\mathbb{E}_{q_c}[\mathbb{1}(C_i = F) \frac{(y_i - \mu_F)^2}{2\sigma^2} + \mathbb{E}_{q_c}[\mathbb{1}(C_i = M)] \frac{(y_i - \mu_M)^2}{2\sigma^2} \right] \\
=& \sum_{i=1}^{n} \frac{\partial}{\partial \mu_M} \left[0 + \mathbb{E}_{q_c}[\mathbb{1}(C_i = M)] \frac{(y_i - \mu_M)^2}{2\sigma^2} \right] \\
=& \sum_{i=1}^{n} q_{C_i}(M) \frac{\partial}{\partial \mu_M} \frac{(y_i - \mu_M)^2}{2\sigma^2} \\
=& \sum_{i=1}^{n} q_{C_i}(M) \frac{2(y_i - \mu_M)}{2\sigma^2} (-1) \\
=& \sum_{i=1}^{n} q_{C_i}(M)(y_i - \mu_M) = \frac{0 \times \sigma^2}{-1} = 0 \\
\hat{\mu_M} =& \frac{\sum q_{C_i}(M) y_i}{\sum q_{C_i}(M)}
\end{aligned} \tag{2.4}$$

These steps have lead us to confirm that equation 2.3 will optimize our parameters. This is the derivation of our *maximization* step of the EM Algorithm. Through mathematical means we have found a way to optimize our parameters. As stated previously the EM Algorithm is a 2-step iterative algorithm, so we also need to derive the *expectation* step. In the *expectation* step we aim to find the $q_C(c)$ that will optimize the likelihood. Fortunately,

with a different variation we are able to find it. We start with the same Jensen's Inequality we first stated above.

$$\begin{aligned}\mathbb{E}_{q_c}\left[\log\left(\frac{P_{Y,C}(y,c;\theta)}{q_C(c)}\right)\right] &= \mathbb{E}_{q_c}\left[\log\left(\frac{P_Y(y;\theta)P_{C|Y}(c|y;\theta)}{q_C(c)}\right)\right] \\ &= \log\left(P_Y(y;\theta)\right) - \mathbb{E}_{q_C}\left[\log\left(\frac{q_C(c)}{P_{C|Y}(c|y;\theta)}\right)\right]\end{aligned} \tag{2.5}$$

First we rewrote the inside of our log equation using the definition of conditional probability $P(A\&B) = P(B) \times P(A|B)$. In the second step, we rewrote the equation using log rules. Recall that we are trying to find the $q_C(c)$ that will maximize our likelihood.

When we maximize with respect to $q_C(c)$, we derive the following and optimized and are left with our *expectation* step, where we estimate the probability of each observation belonging to every cluster conditionally on our current parameter estimates, whereas in the maximization step we estimated the parameters given the current cluster membership probabilities.

On the right side of our equation 2.5 we have

$$\log(P_Y(y;\theta)) - \mathbb{E}_{q_c}\left[\log\left(\frac{q_c(c)}{P_{C|Y}(c|y;\theta)}\right)\right] \tag{2.6}$$

Using KL divergence [Sridharan,], we know that the second part of equation 2.5, $\frac{q_c(c)}{P_{C|Y}(c|y;\theta)}$ will always be greater than 1, therefore the log of it will be at least 0. In order to maximize the likelihood, we want to maximize equation 2.6 in respect to $q_C(c)$; this will happen when the log in the second part of the equation is equal to zero (any other possible value will reduce the likelihood). To make this happen we make $q_C(c)$ equal to $P_{C|Y}(c|y;\theta)$ so that the fraction is equal to 1 and the log of one is zero. Now the likelihood is equal to $\log(P_Y(y;\theta)) - 0$. We've found the $q_C(c)$ that will maximize our likelihood. We've successfully derived the expectation step or better known as the *E-step* by optimizing $q_C(c)$ in order to be able to do expectations by of our parameters.

So what did we do? Recall that the goal was to maximize the likelihood of our given data. We could not use simple methods such as solving the derivative with respect to θ and q_c simultaneously. So instead we had to iterated between finding the max of the likelihood with respect to θ and based

on those results maximized with respect to $q_c(c)$ and vice versa. It is important to acknowledge this iteration does not guarantee a global maximum for the likelihood, but we rest assured that after every iteration our likelihood can only increase (will eventually reach a threshold which might not be the global maximum). To aid with this, the next chapter talks about a method to choose the starting values to help improve the results.

Chapter 3

RNA-Seq

So far we have used a simple example of measuring the average numbers of hours spent watching Netflix by two populations: female and male. However, the EM Algorithm can be and is commonly used when there are more than 2 groups. In this paper we will explore a more realistic real-life example with more than 2 groups. We will be working with RNA-seq data and be calling the groups, clusters.

3.1 RNA-Seq Clustering

RNA-Seq is a relatively new technology that is part of Next-Generation Sequencing (NGS) technology and has become a competitor to the more commonly known technology, microarrays. In essence, RNA-seq allows for the study of cluster analysis where genes are grouped based on similar gene expressions. These groups become clusters that allow scientist to better comprehend genes and their functions. In the past the technology has been used to understand photosynthesis([Xu et al., 2015]).

To analyze such data, the heuristic method, k-means is commonly used for its simplicity, however in this paper we explore the use of the model-based algorithm, Expectation Maximization Algorithm. Model based algorithms have been known to outperform heuristic ones. As Ka Yee Yeung puts it, in model based clustering, we assume that the data comes from a "finite mixture of underlying probability distributions" [Yeung et al., 2001]. Using probabilistic models to approximate the distribution of these clusters permits us to better analyze these groups. It is common amongst biologist and statis-

ticians to use the Poisson distribution, however more research has been done looking into a different distribution: the negative binomial [Si et al., 2013].

3.2 Negative Binomial Distribution

The Negative Binomial Distribution model for gene clustering was first proposed by Robinson and Smyth (2007). In numerous biology experiments that deal with RNA-seq count data, the Poisson distribution is used to model the distribution of the data [Marioni et al., 2008, Bullard et al., 2010]. The data collected in biology like RNA-seq count data (counts means the number of times a specific event happens) is most of the time not too complicated and one-parameter distributions like Poisson are often used. The Poisson distribution makes the consequential assumption that the mean and variance are equal. However, with RNA Seq data, the variance is not always determined by the mean and a one-parameter distributions is not appropriate. In most cases, which is our case with our RNA-seq count data, the observed data are over-dispersed, meaning the sample variance is larger than predicted. Instead we choose to use the negative binomial (NB) distribution [Robinson and Smyth, 2007]. Not taking into consideration the over-dispersion, then the model can create biased estimates [Wang et al., 1996]. Therefore, the Negative Binomial distribution is an essential discrete probability model in biology.

One standard way of parameterizing the negative binomial distribution is:

$$f(k; r, p) = \binom{r-1}{k-1} p^k (1-p)^{r-k}$$

Where r is the number of Bernoulli trials until k successes and p is the probability of success. However, to better fit our problem of clustering data, Robinson and Smyth reformulated the probability function of the NB distribution using parameters for the mean and variance instead of parameterizing as Bernoulli trials distribution as follows:

$$f(y : \mu, \phi) = P(Y = y) = \frac{\Gamma(y + \phi^{-1})}{\Gamma(\phi^{-1})\Gamma(y+1)} \left(\frac{1}{1+\mu\phi}\right)^{\phi^{-1}} \left(\frac{\mu}{\phi^{-1}+\mu}\right)^{y} \quad (3.1)$$

.

In equation 3.1 we let Y be an NB random variable with mean μ and

dispersion ϕ. Variance is parametrized as $\mu + \phi\mu^2$. Y has a distribution NB(μ, ϕ). In order to derive results we use the R package, `MBCluster.Seq` [Si et al., 2013]. The following code provided is a small part of an internal function, `est.nb.mu.mle.one`, used to demonstrate how the reformulation of the distribution is used to find the MLE of our data:

```
for (i in 1:nU) {
        m = c * mu0 * exp(u[i])
        lglk = rowSums(lgamma(n + 1/v) - lgamma(n + 1) -
               lgamma(1/v) - log(1 + m * v)/v -
               log(1 + 1/m/v) * n)
        id[lglk > lglk0] = i
        lglk0[lglk > lglk0] = lglk[lglk > lglk0]
}
```

In this function, `est.nb.mu.mle.one`, μ_0 is initially estimated (not seen above) as the sum of the rows divided by the sum of sum of all our clusters centers. Then we calculate the MLE. In this instance the MLE is calculated by a for loop as seen in the line that states: `i in 1:nU`. The code is going through many iterations to compare different values of μ to see which value maximizes the likelihood. We do this numerical maximization because calculus methods do not permit us to find the derivative of functions of gamma functions. In a similar way, function, `est.nb.v.QL.one`, is able to estimate the dispersion.

```
for (i in 1:nU) {
        v = v0 * exp(u[i])
        Qlglk = n * log(n/mu) - (n + 1/v) *
                log((n + 1/v)/(mu + 1/v))
        d = abs(2 * rowSums(Qlglk) - (nJ - 1))
        id[d < d0] = i
        d0[d < d0] = d[d < d0]
}
v = v0 * exp(u[id])
```

It is important to note again that we could not just find the derivative because of the gamma function values in the NB distribution. The MLE of data with a NB distribution, can only be estimated through iterative

methods. Even when there is no missing data, EM algorithm has been used to estimate MLE of data with negative binomial distribution [Adamidis, 1999].

Using the Negative Binomial Distribution in our data with g groups/clusters and equation 1.4 leads us to the following complete data likelihood.

$$P_Y(y) = \prod_{i=1}^{n} \sum_{j=1}^{g} \pi_j \left(\frac{\Gamma(y_i + \phi_j^{-1})}{\Gamma(\phi_j^{-1})\Gamma(y_i + 1)} \right) \left(\frac{1}{1 + \mu\phi} \right)^{\phi^{-1}} \left(\frac{\mu}{\phi^{-1} + \mu} \right)^{y} \quad (3.2)$$

Here i indexes the observed data value and j indexes the group/cluster.

3.3 Model Based Clustering

We model our data as coming from a mixture of probability distributions as stated earlier. Each distribution is one cluster/group. Let's start with the assumption that there exists K clusters and each cluster has a center μ_k with dispersion ϕ_k. Also assume independence amongst these genes since it would prove too difficult to model correlation amongst the genes with no prior knowledge on the relationship between the genes.

Another difficulty that comes with the expectation maximization algorithm is its lack of a guaranteed global solution. Instead, similar to another commonly used clustering algorithm, the k-means algorithm, it only guarantees a local max. To combat this we provide a better initialization step such that the final solution is hopefully closer to the global max, if not the actual global max.

3.3.1 K-Means Clustering & Initialization

A widely used clustering algorithm is the k-means clustering algorithm. In essence, k-means clustering algorithm is a heuristic approach to the NP-hard problem of partitioning n observations into k clusters such that the square distance between points in the same cluster is minimized. K-means is actually similar to the EM Algorithm; the EM algorithm on Gaussian data is the generalized version of k-means algorithm.

One big difference between the two algorithms is k-means uses hard clustering, but the probabilistic nature of EM makes it use soft clustering instead; in place of assigning every data-point to exactly one cluster in hard clustering, soft clustering is where a data-point is assigned to multiple clusters each

with the probability of belonging to that specific cluster. The algorithm to k-means is simpler than the EM. It follows:

Algorithm 2 K-Means Clustering

1: Pick k random points to act as your k cluster centers.
2: For the remaining $n-k$ points use Euclidean distance to cluster the point with the nearest cluster center.
3: Calculate k new centers by averaging the points in each cluster.
4: Reclassify all n points based on new cluster centers.
5: Recalculate new centers based on new classifications.
6: Repeat steps 4-5 until none of the points are reclassified.

The Expectation Maximization Algorithm is a generalized version of the k-means clustering algorithm giving us more flexibility. This flexibility includes the ability to classify non-normal data using other measuring methods that do not rely on Euclidean distance. However, the algorithms do not differ much in their initialization step. They begin with k arbitrary cluster centers that are usually chosen at random by giving all the points a uniform probability of being chosen. These k initial cluster centers are the reason why these algorithms find local maximums instead of the global max. Considering how important this step is, we aim to find stochastic processes that will ideally lead us closer to a global max.

To reduce the risk of being cemented at a local minimum instead of finding the global minimum, it is important to consider your initialization. Through careful initialization, not only are the end results positively impacted, but the algorithms speed can increase [Fraley and Raftery, 2002]. Fraley and Raftery also proposed using the Bayesian Information Criterion (BIC) to determine the number of groups/clusters present .

Arthur and Vassilvitskii proposed a better initialization step in 2007. This method, is closely related to the k-means clustering algorithm and therefore is adequately is referred to as the K-Means++ clustering algorithm [Arthur and Vassilvitskii, 2007]. K-means++ only refers to the initialization part. The initialization is shown below in Algorithm 3:

Algorithm 3 K-Means++ Clustering

1: Choose one random cluster center from all the genes
2: For the remaining genes calculate $\mathrm{D}(x)$, the distance between the point and the nearest cluster centers available.
3: Choose the next cluster from the remaining genes based on the weighted probabilities that are proportional to $\mathrm{D}(X)$.
4: Repeat 2-3 until you have k clusters.

Using the K-means++ clustering algorithm initialization step for our initial part of the EM algorithm, we now have an algorithmic start instead of a random start. This permits us to perform better and be a step closer to a global maximum. However, note that $\mathrm{D}(x)$ is Euclidean distance, so initialization works well when using EM Algorithm with Gaussian data. However, other distributions that do not rely on Euclidean distance such as Negative Binomial would not work as well.

Chapter 4

Conclusion

We derived the Expectation Maximization algorithm and used it to calculate clusters of gene expression data. This data was represented having a negative binomial distribution specifically to count for over-dispersion.

4.1 Applications

Through the package `MBCluster.Seq` [Si et al., 2013], simulations can be run where you give your normalized data and the number of clusters,k, to the `Cluster.RNASeq` function and using a negative binomial and the EM algorithm, the function is able to cluster all the genes into k clusters. `Cluster.RNASeq` is able to reproduce the clusters in a vector, however, we are able to better represent this data through some plots as shown below in figures 4.1,4.2 and 4.3

The `Hybrid.Treeplot` and `Hybrid.Tree` functions permit us to have a visual representation of our clusters. In the figures shown below we inputted the same data into all three plots, however we used different number of clusters where $k = 5$, 10 and 20. In the x-axis we have the 1000 genes of the sample data set with RNA-seq expressions for all of the genes, 4 treatment and 2 replicates (dataset obtained from the `MBCluster.Seq` R package). In the y-axis there are the four different treatments performed. Below the x-axis we are able to see the different k clusters.

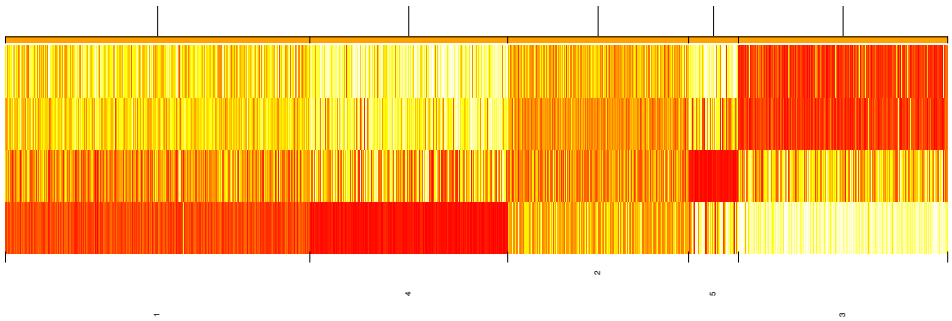

Figure 4.1: Hybrid Tree K = 5

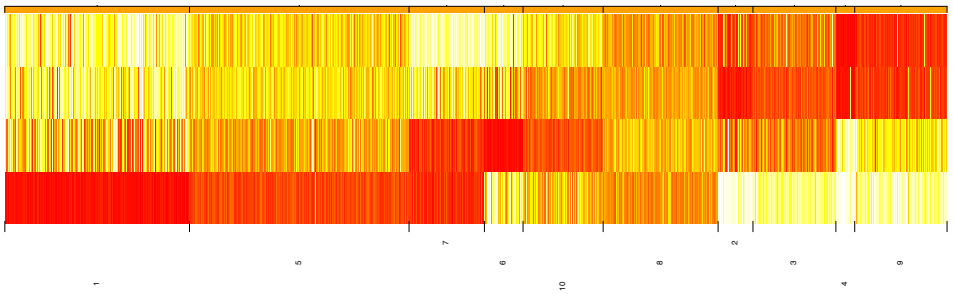

Figure 4.2: Hybrid Tree K = 10

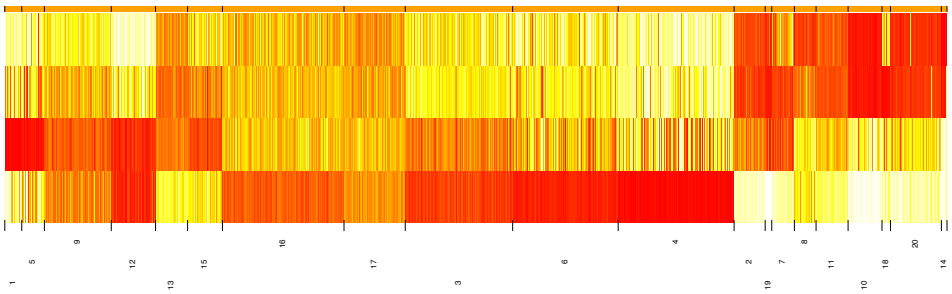

Figure 4.3: Hybrid Tree K = 20

Each different k number of clusters, gave us different results since the same amount of genes had to be separated in more buckets each time. In order to see how much difference existed between the k values we calculate the adjusted rand index. The adjusted rand index gives us a number between 0 and 1; the higher the number the more the two different k's in question agreed on which cluster to assign the genes. When the rand index is 1, then the two groups agreed perfectly. We did not expect a perfect 1 in our data, since that would be impossible, but we were more interested in where this number lied. The adjusted rand differs from the normal rand index by assuming a hyper-geometric distribution to take into account randomness [Yeung and Ruzzo, 2001]. The rand index is typically higher in value than the adjusted rand index.

Table 4.1: Adjusted Rand Index

Average Adjusted Rand Index after 10 Iterations			
k	**5 v 10**	**5 vs 20**	**10 v 20**
Adjusted Rand Index	0.446	0.307	0.494

Throughout this paper, we talked about the Expectation Maximization Algorithm as a clustering algorithm through the mixture of NB models. The EM algorithm was used with a K-means++ initialization to reduce our risk of a local maximum. Through this algorithm we are able to cluster RNA-seq data more efficiently than heuristic methods such as the regular k-means algorithm.

Bibliography

[Adamidis, 1999] Adamidis, K. (1999). Theory and methods: An em algorithm for estimating negative binomial parameters. *Australian and New Zealand Journal of Statistics*, 41:213–221.

[Arthur and Vassilvitskii, 2007] Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035.

[Bullard et al., 2010] Bullard, J. H., Purdom, E., Hansen, K. D., and Dudoit, S. (2010). Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments. *BMC Bioinformatics*, 11.

[Do and Batzoglou, 2008] Do, C. B. and Batzoglou, S. (2008). What is the expectation maximization algorithm? (8).

[Fraley and Raftery, 2002] Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis and desity estimation. *Journal of the American Statistical Association*, pages 611–631.

[Marioni et al., 2008] Marioni, J. C., Mason, C. E., Maine, S. M., Stephens, M., and Gilad, Y. (2008). Rna-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18:1509–1517.

[McLachlan and Basford, 1988] McLachlan, G. J. and Basford, K. E. (1988). *Mixture Models: Inference and Applications to Clustering*, volume 84. Marcel Dekker.

[Robinson and Smyth, 2007] Robinson, M. D. and Smyth, G. K. (2007). Small-sample estimation of negative binomial dispersion, with applications to sage data. *Biostatistics*, pages 321–332.

[Si et al., 2013] Si, Y., Liu, P., Li, P., and Brutnell, T. P. (2013). Model-based clustering for rna-seq data.

[Sridharan,] Sridharan, R. Gaussian mixture models and the em algorithm.

[Wang et al., 1996] Wang, P., Puterman, M. L., Cockburn, I., and Le, N. (1996). Mixed poisson regression models with covariate dependent rates. *Biometrics*, 52(2):381–400.

[Wong et al., 2017] Wong, G. T., Bonocora, R. P., Schep, A. N., Beeler, S. M., Fong, A. J. L., Shull, L. M., Batachari, L. E., Dillon, M., Evans, C., Becker, C. J., Bush, E. C., Hardin, J., Wade, J. T., and Stoebel, D. M. (2017). Genome-wide transcriptional response to varying rpos levels in escherichia coli k-12. *Journal of Bacteriol*, 199.

[Xu et al., 2015] Xu, L., Cruz, J. A., Savage, L. J., Kramer, D. M., and Chen, J. (2015). Plan photosynthesis pehnomics data quality control. *Bioinformatics*, pages 1796–1804.

[Yeung et al., 2001] Yeung, K. Y., Fraley, C., Murua, A., Raftery, A. E., and Ruzzo, W. L. (2001). Model-based clustering and data transformations for gene expression data. Technical report, University of Washington.

[Yeung and Ruzzo, 2001] Yeung, K. Y. and Ruzzo, W. L. (2001). Details of the adjusted rand index and clustering algorithms supplement to the paper "an empirical study on principal component analysis for clustering gene expression data" (to appear in bioinformatics).