Exploring Gene Expression using Naive Bayes Bayesian Network with Negative Binomial Local Models

Netta Shafir*, Shaked Amar†, Ela Falik and Prof. Nir Friedman‡

1 Abstract

Investigating gene behavior and their interactions, we shed light on the body's functionalities. Our project aimed to construct a Bayesian Network model called Naive-Bayes to capture the joint gene distribution within the human body. This model assumes gene conditional independence given a hidden cell state, akin to the problem of clustering samples based on their states. To characterize gene behavior, we employ the Negative Binomial distribution, a widely-used approach for modeling gene distributions. Additionally, we incorporate the sequencing depth of each sample into our model. This parameter, unique to each sample, significantly contributes to the variance across samples.

For parameter estimation, we employ the Expectation-Maximization (EM) algorithm and compare our outcomes with common models and clustering algorithms such as Gaussian Mixture Models (GMM) and K-means. Our results encompass both synthetic datasets generated based on our model and real data sourced from Friedman's lab.

2 Introduction

DNA consists of a sequence of letters that essentially acts as the body's instruction manual. Genes are specific segments within DNA sequence which are responsible for encoding individual proteins, which are complex molecules performing various functions in the body. Different types of cells can produce distinct sets of proteins, leading to variations in gene expression across cell types, despite each cell containing the same DNA. Our goal in this project is to comprehend cellular functions by studying gene interactions and estimating the **joint distribution** of genes. Analyzing this distribution within a single cell type aids in understanding its functions, while comparing gene distributions across cell types can highlight differences between them.

In genetic datasets, genes serve as features, and each sample represents their expression levels. The challenge arises because such datasets typically encompass tens of thousands of genes but only hundreds of samples, a scenario that cause a various probelms, jointly known as the "**curse of dimensionality**". In this situation, data points tend to become sparsely populated in the high-dimentional feature spaces, making it challenging for machine learning models to discern meaningful patterns while increasing the risk of overfitting. To tackle this issue, dimensionality reduction and feature selection can be employed. Alternatively, as in our project, statistical modeling based on domain knowledge and heuristics, which incorporate certain assumptions to address this challenge, can be effective.

In the context of genetic datasets, it is logical to assume that gene expression distributions change under different cellular conditions or **states**. For example, investigating gene expression in the liver involves considering various distributions, such as gene expression under unhealthy liver conditions or alcohol influence, as well as different cell lifecycle stages. We modeled gene expression using a Bayesian Network structure called **Naive-Bayes**, assuming conditional independence among genes given the mentioned state. These independencies simplify and enhance parameter estimation, despite the data's high dimensionality.

In our dataset, we do not directly observe any states. Instead, we treat the state as a **hidden** variable, transforming each dataset sample into a representation of the conditional gene distribution given a specific state. Hiding the state enables us to disregard their connection to particular clinical conditions, like cancer or cell lifecycle phases, and utilize them solely as a means to simplify the joint distribution of genes, merely indicating the number of states to be taken into account.

*netta.shafir@mail.huji.ac.il, ID 205915788
†shaked.amar@mail.huji.ac.il, ID 313234544
‡ela.fallik@mail.huji.ac.il, nir.friedman@mail.huji.ac.il, CS department

Because the state is hidden, direct **Maximum Likelihood Estimation** (MLE) for the joint distribution parameters is not feasible. Therefore, we employ the EM algorithm, well-suited for this task. After learning the parameters using **Expectation-Maximization** (EM) algorithm, they can be utilized for various queries, such as predicting the distribution of one gene given another, a valuable tool for biologists. Additionally, we can view our EM as a clustering algorithm that groups similar **samples** and then seeks to ascertain if the discovered states or clusters align with any biological interpretation. We provide an example of this in the Results section, and compare our algorithm to common clustering methods like K-means.

Naturally, the joint distribution varies when examining different numbers of states. We will present results addressing this matter in the Results section and revisit it in the Discussion section.

2.1 Background

2.1.1 Biological Background

DNA segments are marked by sequences of letters 'A', 'C', 'G', and 'T'. A gene is a DNA segment that is translated into a molecule named RNA, from which proteins are formed, and these proteins are responsible for all processes in the body. Today, DNA sequencing technologies are highly advanced and relatively inexpensive compared to the technology used to measure protein levels. Consequently, measuring the DNA segments that have been translated into proteins can provide valuable insights into the processes within the body.

There are special proteins linked to DNA known as histones, that play a fundamental role in DNA packaging. Various modifications can occur on histones, and one of these modifications serves as an indicator of gene expression or its potential to be expressed. Through the use of technique known as ChIP-seq [4], we capture histones with this specific modification, along with the associated DNA fragments, and sequence these DNA pieces. The DNA we obtain is aligned with the human genome, and we tally the number of DNA segments that correspond to each gene. This count is highly correlated with the level of expression of each gene.

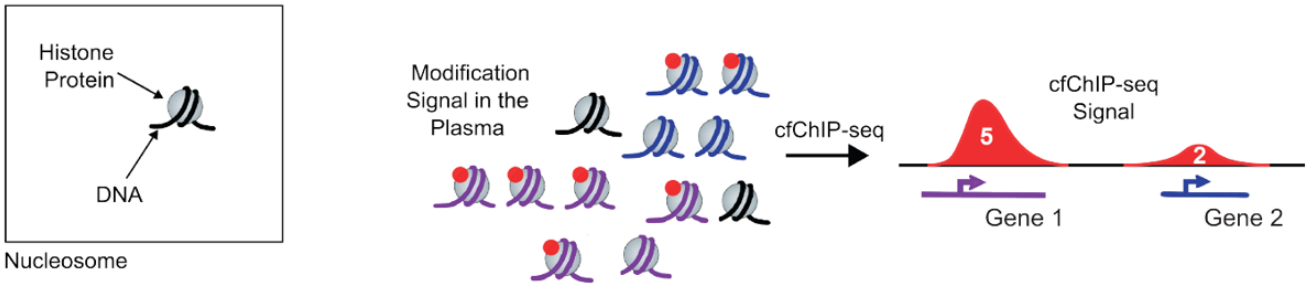


Figure 1: Histones, their modifications, and their utilization in the ChIP-seq technique.

2.1.2 Bayesian Networks

Bayesian Networks are part of family of models known as **Probabilistic Graphical Models**. Bayesian Networks represents a set of variables via a DAG (Directed Acyclic Graph). The directionality in the graph naturally describes **dependency** between the random vaiables. The basic assumption of Bayesian Networks is that a random variable in the graph is independant of its non-decandants, given its parents. That way, the joint distribution of the grpah factorized into a few **local models**, or **CPD's**, each contains only a subset of the all of the random variables. This factorization makes the task of parameter estimation more easy, by decreasing the general amount of parameters of the joint distribution.

One of the simplest Bayesian Networks called **Naive-Bayes.** In this model, we look on the set of random variables $\mathcal{X} = \{C, X_1, \ldots, X_N\}$. The variable C is discrete and called the class, and we assume that

$$\forall i, j \qquad X_i \perp X_j | C \tag{1}$$

That's mean that the X's are independence when C is observed (although they might be dependant withouts observing it). With this asumption, we get the factorization of the joint distribution:

$$p(C, X_1, \ldots, X_N) \overset{(A)}{=} p(C) \prod_{i=1}^{N} p(X_i | C, X_1, \ldots, X_{i-1}) \overset{(B)}{=} p(C) \prod_{i=1}^{N} p(X_i | C) \tag{2}$$

where (A) is from the chain rule, and (B) is from the independence assumption in (1).

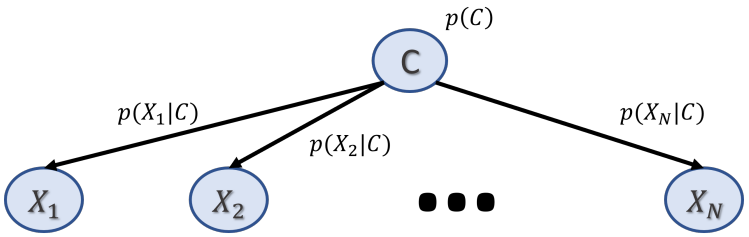


Figure 2: The graphical representation of the Naive-Bayes model. The factors $p(X_i|C)$, as well as the prior distribution $p(C)$, are the CPDs.

Utilizing the Naive Bayes model can significantly decrease the number of parameters necessary to describe the joint distribution $p(C, X_1, \ldots, X_N)$. When assuming that $C, X_1, \ldots, X_N$ follow a categorical distribution with K categories, without the Naive Bayes assumption, we would require $O\left(K^N\right)$ parameters, whereas with it, we only need $O(K(N+1))$ parameters.

When we possess samples from all variables $C, X_1, \ldots, X_N$, estimating the parameters for each CPD is straightforward. However, in numerous scenarios, we work with a dataset comprising samples from $X_1, \ldots, X_N$, while C is missing. In this situation, we lack the assignment of each sample to its corresponding state, rendering the task of parameter estimation more challenging. Nonetheless, various iterative algorithms, such as the EM algorithm, can be applied in such cases.

2.1.3 EM algorithm

The Expectation-Maximization (EM) algorithm [1] is a powerful iterative optimization technique widely used in statistics and machine learning to estimate parameters in probabilistic models when dealing with incomplete or missing data. The algorithm consists of two main steps in each iteration: the E-step (Expectation step) and the M-step (Maximization step). In the E-step, it compute the expectation of the complete-data log-likelihood, or **complete log-likelihood**, w.r.t the posterior of the latent variables. In the M-step, it updates the parameter estimates to maximize the expected log-likelihood obtained in the E-step.

When denoting by X as a set of observed variables, by Z a set of hidden variables, and by θ the vector of unknown parameters of the complete log-likelihood $\ell(\theta; X, Z) = \log p(X, Z; \theta)$, then in the $t+1$ iteration, the E-step defines

$$Q\left(\theta \mid \theta^{(t)}\right) = \mathbb{E}_{Z|X;\theta^{(t)}}\left[\ell(\theta; X, Z)\right] \tag{3}$$

and the in the M-step we calculates

$$\theta^{(t+1)} = \text{argmax}_\theta Q\left(\theta \mid \theta^{(t)}\right) \tag{4}$$

and this alternation between estimating the missing data and updating the parameters continues and is promised to convergence to the local maxima of the **marginal log-likelihood** $\ell(\theta; X) = \log p(X; \theta)$ - the likelihood of the observed variables. In particular, it holds that the marginal log-likelihood cannot decrease during the iterations, namely

$$\ell\left(\theta^{(t+1)}; X\right) \geq \ell\left(\theta^{(t)}; X\right) \tag{5}$$

This approach is particularly valuable when dealing with scenarios where data points do not have clear, known cluster memberships. The EM algorithm effectively uncovers these latent or hidden cluster assignments.

In practice, computing $\mathbb{E}_{Z|X;\theta^{(t)}}\left[\ell(\theta; X, Z)\right]$ in the E-step demands caluculating the expected value of the **sufficient statistics**. Sufficient statistics are a condensed set of statistics (functions of the data, or data summaries) that contain all the necessary information for making statistical inferences about a parameter, as the data itself. In other words, if you have sufficient statistics, you can perform any parameter estimation using only them, and get completely equivalent results as using the full dataset. The primary benefit of using sufficient statistics instead of the original data is higer computational efficiency. You also get a nicer form of the likelihood, so that it depends on the sufficient statistics instead of the data. In the E-step, after calculating the expected sufficient statistics (ESS), we plug them in the likelihood instead of the sufficient statistics.

Despite its utility, the EM algorithm has certain disadvantages. One primary limitation is its **sensitivity to initialization**. As mentioned above, convergence to the global optimum is not guaranteed, and the final solution depends on the initial parameter values. Additionally, the EM algorithm can be **computationally intensive**, especially when dealing with large

datasets or complex models. Each iteration involves calculating expectations over the missing data, which might require integrating over a high-dimensional space.

Moreover, EM algorithm suffers from a special challenges in the case of a **Mixure Model**, which is a probabilistic model that represents a population as a combination of several subpopulations or components, each characterized by its own probability distribution, and its commonly used for modeling complex data with hidden structures or clusters. One phenomenon of the EM algorithm in this context is called "**label switching**". Label switching refers to the fact that the components of the mixture model are interchangeable in terms of their order. During the EM algorithm's optimization process, the algorithm can easily encounter multiple equivalent solutions where the components have swapped labels, but the likelihood remains unchanged. This occurs because the EM algorithm cannot distinguish between the different components in terms of labeling. This phenomenon is also a private case of a wider phenomanon called **non-identifiablity**, where two different model gives the same likelihood and therefore both fit to the data the same way.

2.2 Data

Our data comes from a public data source named **BLUEPRINT** [2]. The data publishers utilized the ChIP-seq method to capture histones bound to DNA fragments with the H3K4me3 modification. They applied this method to various cell types, each of which had a different number of samples. For each sample, Friedman's laboratory counted the DNA fragments that fell within each gene and calculated the sequencing depth of that sample. In total, we obtained a count matrix in which each row represents a sample, and each column represents a gene. Along with each sample, we have information about the cell type on which the method was performed and the sequencing depth of the sample. For our algorithm's purposes, we focused on three types of cells: T cells, B cells, and NK cells, all of which are lymphocytes. Specifically, we included 23 samples of T-cells, 36 samples of B-cells, and 3 samples of NK-cells.

2.3 Related Work

Our work was inspired by the article "Context-Specific Bayesian Clustering for Gene Expression Data", by Barash et al. [3], that modeled gene expression with a variant of Naive Bayes Model, which called **Context-Specific Clustering**. A brief description of the model suggest in the article appears in appendix B - Context-Specific Clustering.

3 Methods

3.1 Statistical Methods and Consideration

3.1.1 Probabilistic Model of Our Data

Having N genes, we denote by $X_1, \ldots, X_N$ the gene expression levels, and we note by C the hidden state. Let's denote by $\overrightarrow{c} = (c[1], \ldots, c[M])$ the (imaginary) observed values of $C \sim \text{categorical}(\theta_C)$ (when $\theta_C \in \mathbb{R}^K$, and $\theta_C[k] = p(C = k)$[1]), and our dataset $\mathcal{D}_M = [x[1], \ldots, x[M]]^T \in \mathbb{R}^{M \times N}$ of M samples, where $x[m] \in \mathbb{R}^N$. For start, we assume that $\{x_i[m]\}_{m=1}^M$ are i.i.d samples from $p(X_i \mid C = c[m])$. Later on, we will give up on that assumption.

The marginal log-likelihood of the observed X's is then

$$\ell(\theta; \mathcal{D}_M) = \sum_{m=1}^{M} \log \left(\sum_{k=1}^{K} \theta_C[k] \prod_{i=1}^{N} p(X_i = x_i[m] \mid C = k) \right) \tag{6}$$

and the complete log-likelihood is

$$\ell(\theta; \mathcal{D}_M, \overrightarrow{c}) = \sum_{k=1}^{K} \sum_{m=1}^{M} \left[\mathbb{1}_{\{c[m]=k\}} \log \theta_C[k] + \sum_{i=1}^{N} \log p(X_i = x_i[m] \mid C = k) \right] \tag{7}$$

Algebric steps are shown in appendix A - Mathematical Formulations.

[1] By $\theta_C[k]$ we mean the k'th component of θ_C. This notation remains consistent throughout the remainder of this paper.

Modeling the distribution of the CPD's - For completing the formulation, we model the CPD's, i.e. the distributions $\{p(X_i \mid C=k)\}_{i,k}$. The regular Naive-Bayes model offers the CPD's $X_i|\{C=k\} \sim \text{categorical}(\theta_{i,k})$. We chose the **Negative Binomial** (NB), which is a common modeling for gene expression in the Biology [5]. If $X \sim \text{NB}(r,p)$ then

$$p(X=x) = \binom{x+r-1}{x} p^r (1-p)^x \tag{8}$$

Here we take the extended definition of the NB distribution, so that the r parameter can get positive real values (and not just integers) so we get that $\binom{x+r-1}{x} = \frac{\Gamma(x+r)}{\Gamma(x+1)\Gamma(r)}$ When $\Gamma(x)$ is the gamma function, which for any natural number holds $\Gamma(n) = (n-1)!$.

The NB distribution has a few advantages over the categorical distribution. First, it only demands $O(NK)$ parametrs. In the Categorical CPD, if we denote $|\theta_{i,k}| = n_{i,k}$, then the number of parameters is $O(NK \max\{n_{i,k}\})$, while $\max\{n_{i,k}\}$ can be very large: A natural choice is to take $n_{i,k} = D$ for every i and k, where D is maximum value we observe in $\mathcal{D}_M$ for all values of i and k, and this value can be quite large. This large number of parameters is also might leed to overfitting.

Second advantage lies in the **predictive distribution** $p(X[M+1]|\mathcal{D}_M)$- the distribution of the $M+1$ sample, given a training dataset $\mathcal{D}_M$ of M samples. In the NB case, the predictive distribution is well defined for any positive discrete new sample and give some probability mass to values larger than seen in the train set, while the categorical case doesn't allow to generalize to a new datapoint which have a gene ith value larger that $\max\{n_{i,k}\}$[2].

In our project, we assume that every gene have its own NB distribution, given the state of the cell, namely $X_i|\{C=k\} \sim \text{NB}(r_{i,k}, p_{i,k})$.

Now we have a complete formulation for the complete log-likelihood, which is

$$\ell(\theta; \mathcal{D}_M, \overrightarrow{c}) = \sum_{k=1}^{K} \left[\left(\sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \right) \log \theta_C[k] \right. \tag{9}$$

$$+ \sum_{i=1}^{N} \left[\sum_{d=0}^{\max X_i} \left(\sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \mathbb{1}_{\{x_i[m]=d\}} \right) \log \Gamma(d + r_{i,k}) \right. \tag{10}$$

$$+ \left(\sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \right) (r_{i,k} \log p_{ik} - \log \Gamma(r_{i,k})) \tag{11}$$

$$+ \left(\sum_{m=1}^{M} x_i[m] \cdot \mathbb{1}_{\{c[m]=k\}} \right) \log(1-p_{ik}) \tag{12}$$

$$+ const(\theta) \tag{13}$$

Sufficient Statistics - As mentioned before, finding MLE in the case of missing data demands defining the **sufficient statistics** of the likelihood. We now define the statistics

$$S_C[k] := \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \tag{14}$$

$$S_{sum}[i,k] := \sum_{m=1}^{M} x_i[m] \cdot \mathbb{1}_{\{c[m]=k\}} \tag{15}$$

$$S_{count}[i,k,d] := \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \mathbb{1}_{\{x_i[m]=d\}} \tag{16}$$

when $S_C[k]$ counts the number of samples whose state is k, $S_{sum}[i,k]$ sum the values of the i'th variable whose sample's state is k, and $S_{count}[i,k,d]$ counts the number of sample whose state is k and their i'th variable equals d. With our new

[2] i.e. the probability of a gene to get any value which is larger than the length of the vector parameter of the gene's CPD in the categorical case, is not well defined.

definitions, our complete log-likelihood can be written as

$$\ell\left(\theta;\{S_C[k]\}_k,\{S_{sum}[i,k]\}_{i,k},\{S_{count}[i,k,d]\}_{i,k,d}\right)=\sum_{k=1}^{K}[S_C[k]\cdot\log\theta_C[k] \tag{17}$$

$$+\sum_{i=1}^{N}\left[\sum_{d=0}^{\max X_i}S_{count}[i,k,d]\cdot\log\Gamma(d+r_{i,k}) \tag{18}$$

$$+S_C[k]\cdot(r_{i,k}\log p_{ik}-\log\Gamma(r_{i,k})) \tag{19}$$

$$+S_{sum}[i,k]\cdot\log(1-p_{ik})]] \tag{20}$$

$$+const(\theta) \tag{21}$$

And we can see that we do not need the data anymore to describe the log-likelihood. This means that our statistics are indeed sufficient.

We now want to find the MLE for θ. In the case of missing data, we don't have the sufficient statistics, and so we need to execute the EM algorithm, and in the E-step calculating the expectations getting the ESS. But for a start, let's assume we have a complete dataset, i.e. have the sufficient statistics.

3.1.2 Maximum Likelihood Estimation of Complete Dataset

Given the sufficient statistics, our optimization problem is:

$$\max_{\theta}\ell\left(\theta;\{S_C[k]\}_k,\{S_{sum}[i,k]\}_{i,k},\{S_{count}[i,k,d]\}_{i,k,d}\right) \tag{22}$$

$$s.t.\quad\sum_{k=1}^{K}\theta_C[k]=1 \tag{23}$$

$$\forall i,k\quad p_{i,k}\in(0,1) \tag{24}$$

$$\forall i,k\quad r_{i,k}>0 \tag{25}$$

We can solve this use Lagrange multipliers: We Differentiate the Lagrangian $\mathcal{L}$ w.r.t each parameter, and equate to zero to find the maximum[3].

Differentiating the Lagrangian w.r.t. $\hat{\theta}_k^C$, we get

$$\hat{\theta}_C[k]=\frac{S_C[k]}{M} \tag{26}$$

And differentiating w.r.t. $p_{i,k}$ and $r_{i,k}$, we get

$$\hat{p}_{i,k}=\frac{r_{i,k}\cdot S_C[k]}{S_{sum}[i,k]+r_{i,k}\cdot S_C[k]} \tag{27}$$

$$\frac{\partial}{\partial r_{i,k}}\mathcal{L}=\left[\sum_{d=0}^{\max X_i}S_{count}[i,k,d]\cdot\psi(d+r)\right]-S_C[k]\cdot\psi(r)+S_C[k]\cdot\log p_{i,k} \tag{28}$$

Where ψ is the Digamma function, which its defination is $\psi(x)\coloneqq\frac{d}{dx}\log\Gamma(x)$. One can observe that $p_{i,k}$ has a closed form solution, while $r_{i,k}$ doesn't has. Nevertheless, we can yet solve this equation system numerically with Gradient-based solvers like Newton-Rapshon.

But in reality, we do not have the sufficient statistics because $\vec{c}$ is missing, so we cannot compute the MLE directly. Instead, we will use the EM algorithm.

3.1.3 Expectation Maximization

E-step - In the E-step, we need to calculate the expectations of the log-likelihood using the parameters of the last iteration. In fact, this only requires to calculate the **expected sufficient statistics (ESS)**. In our case, to calculate the ESS is sufficient

[3]Actually, we omit the two last constrains for getting an easier problem, and write the Lagrangian only with the first constraint. Afterwards, we see that the solutions we get for $p_{i,k}$ and $r_{i,k}$ keep their original constraints, so we know for sure that this is indeed the optimal solution for the original problem

to calculate the expectations of $\mathbb{1}_{\{c[m]=k\}}$, which are

$$\begin{aligned}\mathbb{E}_{C|\mathcal{D}_M;\theta^{(t)}}\left[\mathbb{1}_{\{c[m]=k\}}\right] &= p\left(C\left[m\right]=k \mid \mathcal{D}_M, \overrightarrow{\eta};\theta^{(t)}\right) \\ &= p\left(C=k \mid X=x\left[m\right],\eta\left[m\right];\theta^{(t)}\right) \\ &= \frac{p\left(X=x\left[m\right] \mid C=k,\eta\left[m\right];\theta^{(t)}\right)p\left(C=k;\theta^{(t)}\right)}{\sum_{k'}p\left(X=x\left[m\right] \mid C=k',\eta\left[m\right];\theta^{(t)}\right)p\left(C=k';\theta^{(t)}\right)}\end{aligned}$$

and with $\theta^{(t)}$ we have a closed form for all the distributions above. The ESS is then

$$\mathbb{E}_{C|\mathcal{D}_M;\theta^{(t)}}\left[S_C\left[k\right]\right] = \sum_{m=1}^{M}\mathbb{E}_{C|\mathcal{D}_M,\overrightarrow{\eta};\theta^{(t)}}\left[\mathbb{1}_{\{c[m]=k\}}\right] \tag{29}$$

$$\mathbb{E}_{C|\mathcal{D}_M;\theta^{(t)}}\left[S_{sum}\left[i,k\right]\right] = \sum_{m=1}^{M}x_i\cdot\mathbb{E}_{C|\mathcal{D}_M,\overrightarrow{\eta};\theta^{(t)}}\left[\mathbb{1}_{\{c[m]=k\}}\right] \tag{30}$$

$$\mathbb{E}_{C|\mathcal{D}_M;\theta^{(t)}}\left[S_{count}\left[i,k,d\right]\right] = \sum_{m:x_i[m]=d}\mathbb{E}_{C|\mathcal{D}_M,\overrightarrow{\eta};\theta^{(t)}}\left[\mathbb{1}_{\{c[m]=k\}}\right] \tag{31}$$

M-step - In the M-step, we just need to execute the MLE procedure described in the last section, using the ESS instead of the sufficient statistics. Namely:

$$\left[\hat{\theta}_C\left[k\right]\right]^{(t+1)} = \frac{\mathbb{E}\left[S_C\left[k\right]\right]}{M} \tag{32}$$

$$\hat{r}_{i,k}^{(t+1)}=\text{NewtonRaphson}\left(\left[\sum_{d=0}^{\max X_i}\mathbb{E}[S_{count}[i,k,d]]\cdot\psi(d+r)\right]-\mathbb{E}[S_C[k]]\cdot\psi(r)+\mathbb{E}[S_C[k]]\cdot\log\left(\frac{r_{i,k}\cdot\mathbb{E}[S_C[k]]}{\mathbb{E}[S_{sum}[i,k]]+r_{i,k}\cdot\mathbb{E}[S_C[k]]}\right)\right) \tag{33}$$

$$\hat{p}_{i,k}^{(t+1)} = \frac{\hat{r}_{i,k}^{(t+1)}\cdot\mathbb{E}\left[S_C\left[k\right]\right]}{\mathbb{E}\left[S_{sum}\left[i,k\right]\right]+\hat{r}_{i,k}^{(t+1)}\cdot\mathbb{E}\left[S_C\left[k\right]\right]} \tag{34}$$

We ulternate between the E-step and the M-step until convergence of the marginal log-likelihhod.

3.1.4 Sequencing Depth

When sequencing genetic material like RNA or DNA, the **sequencing depth** is a parameter represents how deep is the sequencing of each sample. In our dataset $\mathcal{D}_M$, every sample has a different sequencing depth, which is estimates using the pipeline described in **Sadeh R. et al.** [4]. The sequencing depth is a major source of variation between samples and therefore can bias the parameter estimation of our model.

For addressing this issue, we first denote by $\overrightarrow{\eta} = (\eta\left[1\right],\ldots,\eta\left[M\right])$ the vector of the sequencing depth of all the samples in our dataset $\mathcal{D}_M$. We still want to remain with the Negative Binomial modeling. Let's recall the the NB distribution can be represented as a Gamma-Poisson mixure [6]. Namely, if $X\sim\text{NB}\left(r,p\right)$, then it is equivalent that $X|\lambda\sim\text{Poisson}\left(\lambda\right)$ when λ is a random variable with the prior distribution $\lambda\sim\text{Gamma}\left(\alpha,\beta\right)$. Now, giving a constant $\eta>0$, let's consider the variable $\widetilde{X}$, such that $\widetilde{X}|\lambda\sim\text{Poisson}\left(\eta\cdot\lambda\right)$. Then it can be shown that

$$\widetilde{X}\sim\text{NB}\left(r,\frac{p}{p+\eta-p\eta}\right) \tag{35}$$

with respect to the parameters (r,p) of the variable X.

In our project, we are still assummig that $X_i|\left\{C=k\right\}\sim\text{NB}\left(r_{i,k},p_{i,k}\right)$, but now we believe that the gene expression we are observing have slighly different distribuiton, such that

$$X_i\left[m\right]\mid\left\{C=k\right\}\sim\text{NB}\left(r_{i,k},\frac{p_{i,k}}{p_{i,k}+\eta\left[m\right]-p_{i,k}\eta\left[m\right]}\right) \tag{36}$$

That way, taking the sequencing depth into considerations in our analysis, we get that the samples in $\mathcal{D}_M$ **are not i.i.d anymore**: They sampled independently, but they are not identically distributed.

The complete log-likelihood of the data changes a bit: It is similar to the likelihoon in Eq. (17), but have another term which is

$$-\sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \cdot (r_{i,k} + x_i[m]) \cdot \log(p_{ik} + \eta[m] - p_{ik}\eta[m]) \tag{37}$$

One can observe that if we place $\eta[m] = 1$, this term gets nullified and we return to the log-likelihood we had before. Another important change is that the statistics we defined before are not sufficient anymore: Defining sufficient statistics requires the ability of bringing the likelihood into a form in which the data and the learned parameters are located in separate factors, and it is clear that the term in Eq. (37) cannot be brought into that form. Consequently, we get that our sufficient statistics is the data itself, which is the trivial sufficient statistic. Furthermore, differentiating the lagrangian of the MLE w.r.t. $r_{i,k}$ and $p_{i,k}$ we get

$$\frac{\partial}{\partial p_{i,k}}\mathcal{L} = \frac{r}{p} \cdot S_C[k] - \frac{1}{1-p} S_{sum}[i,k] - \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \cdot (r + x_i[m]) \cdot \frac{1-\eta[m]}{p + \eta[m] - p\eta[m]} \tag{38}$$

$$\frac{\partial}{\partial r_{i,k}}\mathcal{L} = \sum_{d=0}^{\max X_i} S_{count}[i,k,d] \cdot \psi(d+r) - S_C[k] \cdot (\log p - \psi(r)) - \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \cdot \log(p + \eta[m] - p \cdot \eta[m]) \tag{39}$$

And one can observer that after equating to zero, the equation system we get does not have a closed form solution for neither $r_{i,k}$ nor $p_{i,k}$, and the solution is needed to be found numerically. This is not a dead end of course, but it makes the optimization harder than before, what will slow down the M-step of our EM algorithm.

3.2 Model Training Details

3.2.1 Variation of EM

As mentioned above, solving the equation system in Eq. (38)-(39) was challenging, for it does not possess a straightforward closed-form solution. What complicates matters is that even numerical solutions have proven to be problematic in certain cases, when sometimes it has no solution at all. Therefore, we used a variation of EM in which we do not necessarily maximizing the expected complete log-likelihood in the M-step, but instead trying to find parameters that increases its value. By that, we also got an improvement in the marginal log-likelihood - our goal in each iteration of the EM - because the expected log-likelihood is known to be a lower bound of the marginal log-likelihood.

In our algorithm, we explored two distinct solvers. One approach aimed at maximizing the complete log-likelihood, while the other sought to find a root for its gradient[4]. We attempted both methods and used the results from one of them when the other failed. If both of them were failed, we assessed whether the output parameters led to improvements in the expected log-likelihood compared to the previous iteration. If so, we adopted them. If no improvement was observed, we retained the parameters from the prior iteration, as they ensured the same value in the expected complete log-likelihood as the previous iteration for their respective conditional probability distributions.

3.2.2 Handeling the label switching on simulated data

As mentioned earlier, the EM algorithm is susceptible to a phenomenon called "label switching". We encountered this challenge when applying the algorithm to simulated data because we observed that it sometimes converged to the correct values but with different orders of states. This presented an issue when evaluating the algorithm's success, particularly when assessing the convergence of parameters to the correct values. Even if we ran the algorithm multiple times with different initializations and it converged successfully each time, the parameters for a certain state could appear in different columns in the output parameter matrix.

We addressed this problem by identifying the permutation π that maximizes the inner product between the output vector $\hat{\theta}_C$ and the vector $\tilde{\theta}_C$, when $\hat{\theta}_C$ represents the real MLE estimator for θ_C [5], and $\tilde{\theta}_C$ is the output of the EM algorith. That is to say

$$\pi = \text{argmax}_{\pi'} \sum_{k} \hat{\theta}_C[k] \cdot \tilde{\theta}_C[\pi'(k)] \tag{40}$$

Where π is also the permutation of the components of $\tilde{\theta}_C$ that maximizes the correlation between $\hat{\theta}_C$ and the permutated $\tilde{\theta}_C$. This problem is known as **linear sum assignment**.

[4]Both solvers are of 'sklearn' library.

[5]Which in fact is the proportions of each state in the empirical data distribution.

We found that this method did nicely in aligning the output with the order of the desired parameters, in the cases of successfull run of the algorithm. Yet, in the of partial seccess of the algorithm, i.e. when part of the states identified properly and the other did not, it can easily miss aligning the states that recovered properly in their correct index.

3.3 Preprocessing

In the data preprocessing, a series of crucial steps were implemented. Initially, samples with sequencing depths below 1, deemed illegal for our model, were removed, with only a single sample exhibiting sequencing depth which is lower that 1. Subsequently, the data was divided into training and testing sets, with 49 samples allocated for training and 12 for testing. To enhance the robustness of the analysis, only genes present in the **RefSeq** [8] [9] database were retained, as these genes are associated with more reliable annotations. RefSeq is a curated repository of genes, containing reference sequences for various organisms, and ensuring the accuracy and consistency of genomic information. Additionally, feature selection was conducted using K-means clustering with three clusters on the train data, after we standardazed the data by dividing each sample with its sequncing depth. We defined a criterion to identify genes with significant expression differences across clusters, in the manner that those genes have high expression in one cluster and low expression in the other two clusters. Around 130 genes was considered as significant to any of the clusters w.r.t our criterion. Then, those significant genes were assessed based on their fold change between the largest and second-largest mean within each cluster, with the top 10 genes chosen from each cluster, resulting in a selection of 30 genes for the further analysis.

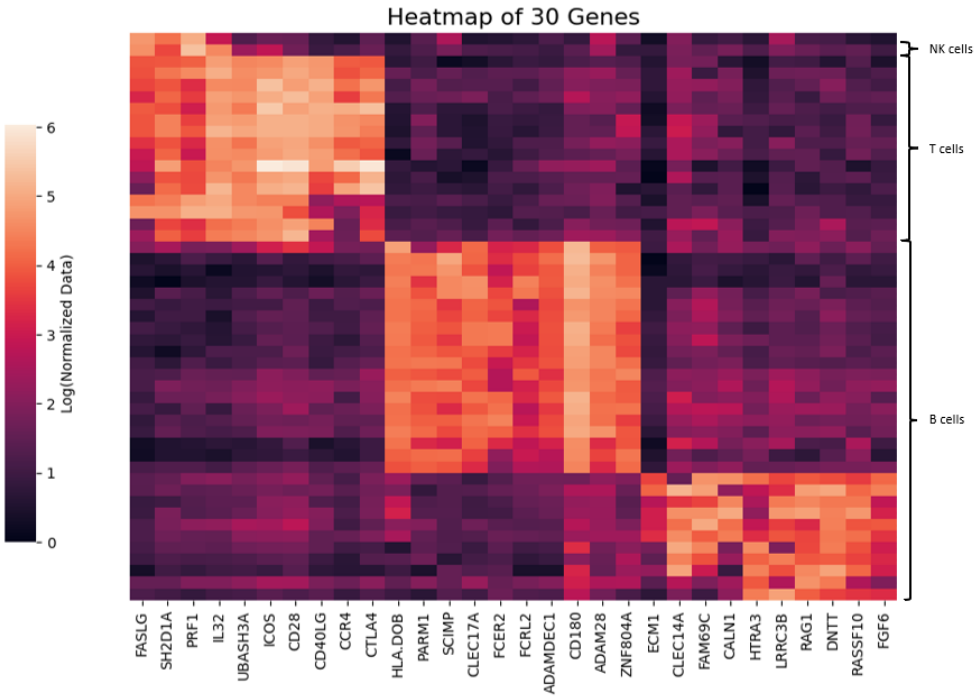


Figure 3: A hierarchically-clustered heatmap that show the expression of the selected genes in the train samples, in a logarithmic scale. We can see that the data is easily devided to 3 clusters, for we chose the genes that were significant in separate 3 clusters of K-means.

4 Results

4.1 Initializations for the EM algorithm on Synthetic data

An issue highlighted earlier is the sensitivity of the EM algorithm to its initialization. This sensitivity can lead to convergence towards local maxima instead of the global maximum, and this problem becomes more pronounced as the model complexity increases. To address this challenge, we explored different initialization strategies, most of which involved running the EM algorithm multiple times with varying initializations and selecting the parameter values that yielded the highest likelihood.

1. **A completely random initialization -** We attempted a random initialization approach, where each conditional probability distribution (CPD) was initialized with independently chosen parameters. However, we found that these parameters often failed to converge to the true values and, instead, tended to become stuck in a loop without convergence.

2. **An identical initialization for each state -** We geenrated a random r and p for each gene, and copied it for all of the states. We also introduced random variations in θ_C to break complete symmetry between the states (that necessarily yields a symmetric output for each state as well). This approach improved convergence to the correct parameters, although it took a relatively long time to converge, with many iterations showing minimal parameter changes. This was less than ideal since the EM algorithm typically experiences the most significant improvements in the likelihood during the initial iterations.

3. **An identical-noised initialization** - Like the last initialization, but also introducing small, normally distributed noise to r and p. This method achieved better results, converging to the correct parameters at a similar rate as the identical initialization approach, but with significantly faster convergence.

4. **GMM initialization** - The best initialization method we found, was to initiate our EM with the output of a GMM Mixure Model. Indeed, the GMM initialization not only yielded correct results more consistently across multiple runs compared to the identical-but-noised initialization as described in the last section[6], but also started closer to the correct parameters, resulting in quicker convergence (typically around 3 iterations). Additionally, the efficient implementations of GMM in libraries such as sklearn, which are often written in lower-level languages than Python, significantly accelerated the execution of our algorithm[7].

[6] In fact, the GMM operates as an EM algorithm with random initialization, implying that theoretically, its results should exhibit some inherent variability, which, in turn, would introduce variance into our algorithm's output. However, in practice, the GMM solver consistently converges to nearly identical results across most runs, resulting in a corresponding reduction in the variance of our algorithm's output.

[7] Here are the specific details of this GMM-based initialization: First, we standardized the data by dividing each sample by its sequencing depth and applied the GMM solver to this standardized dataset. We used the prior output of the GMM as θ_C and estimated r and p using the method of moments with the means and variances obtained from the GMM. In some cases, these estimations yielded illegal values, with negative $r_{i,k}$ or values outside the range $(0, 1)$ for $p_{i,k}$. To address this, we ran another GMM on the original data and replaced the illegal values with the outputs of this second GMM. In most instances, this rectified the issue, but if any illegal $r_{i,k}$ or $p_{i,k}$ values persisted, we assigned random estimations to them. Notably, any other GMM-based initialization strategy, apart from this one, produced inferior results compared to the identical-noised initialization.

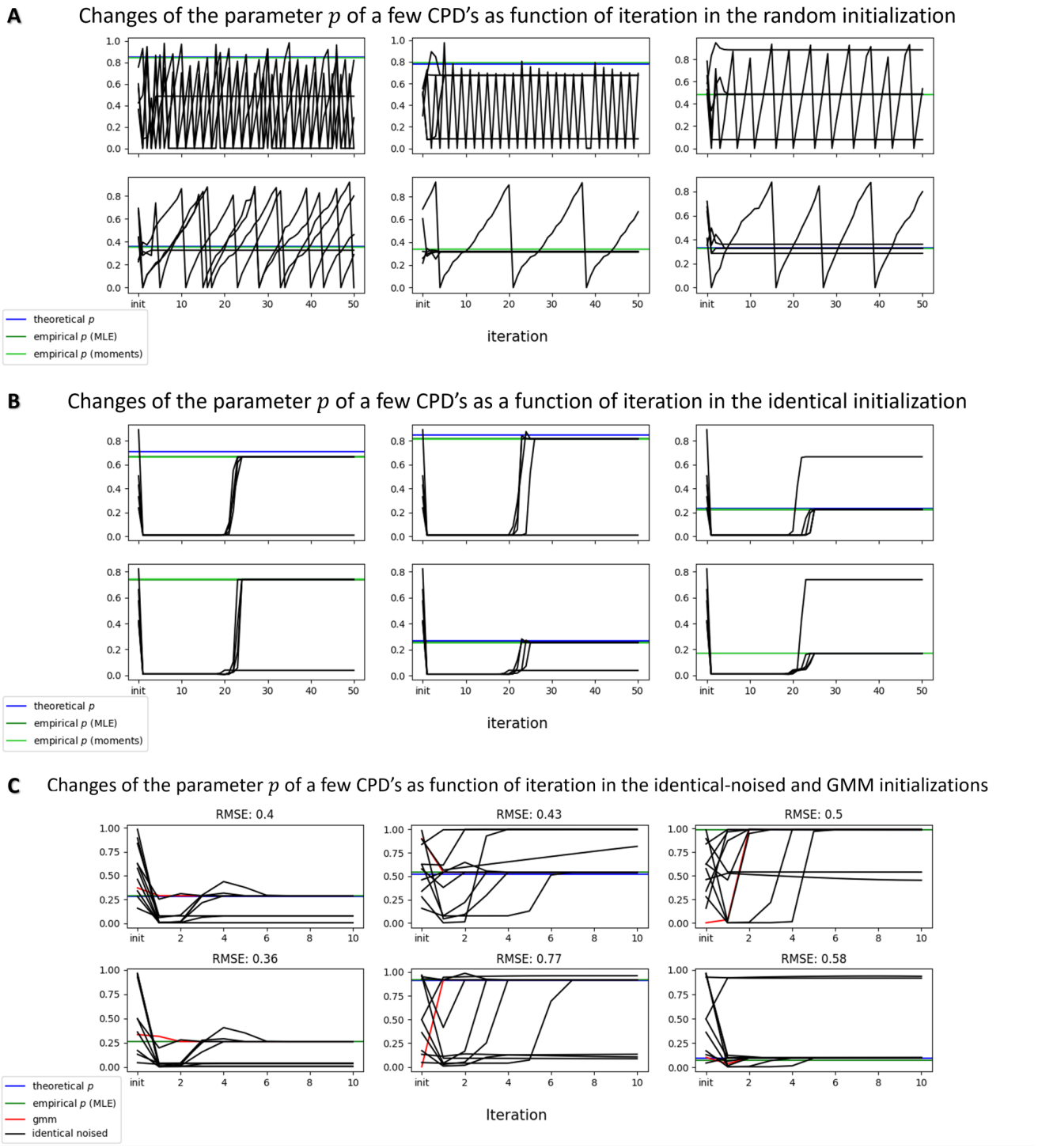


Figure 4: The change in parameters for the different initialization methods as a function of iteration of the EM, each black line represents a seperate run with different initialization under the strategy shown in the graph. The blue and the dark green horizontal lines represents the parameters the data was generated from, and the MLE for the parameters using the sufficient statistics of the empirical distribution of the data, respectively, when the EM should converge to the MLE. **Plot A** - We can see that the random strategy didn't converged most of the times, **Plot B** - identical strategy converged to the right parameters, but it took many iterations, **Plot C** - the identical-noised strategy aventually converged most of the runs, but the GMM strategy converged faster, and it also converged almost every time (although it cannot be observed from the plot). Above each subplot there is the Root-mean-square deviation (RMSE) of the identical-noised stratery w.r.t. the real MLE parameters in green.

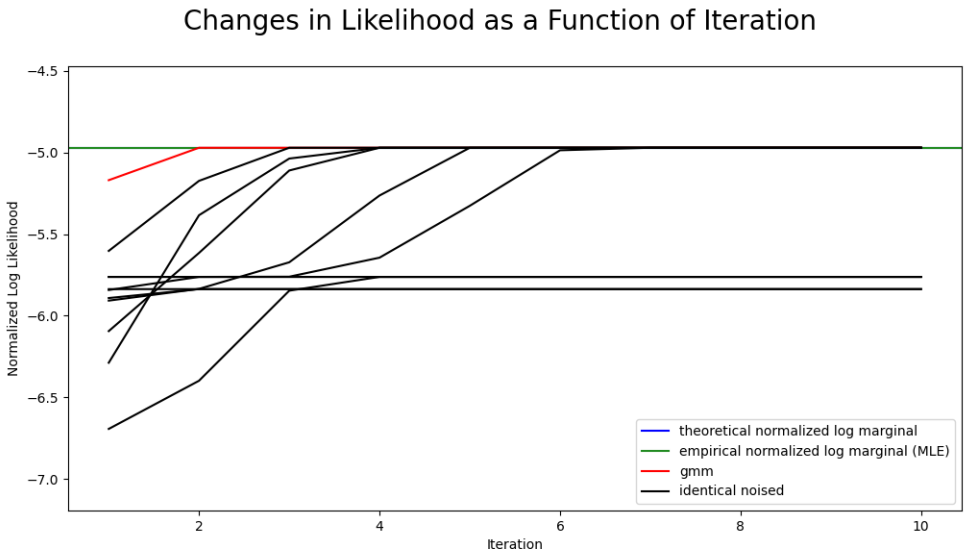


Figure 5: The change in likelihood for two "good" initialization methods as a function of iteration of the EM.

4.2 Accuracy In Predicting Cell Type

Referring to our earlier feature selection strategy, we based our gene selection process on a K-means clustering approach, aiming to leverage the presence of three distinct cell types within our dataset. Recognizing the potential significance of cell type as a robust signal in our data, we incorporated it into our feature selection methodology. Initially, we filtered genes that met our significance criteria and sought to validate their biological relevance to these specific cell types. To accomplish this, we utilized **Enrichr**, a gene analysis tool [10], to determine the association between the selected genes and cell types based on over-expression patterns compared to diverse gene datasets. Our analysis revealed strong correlations between the expression profiles of certain gene clusters and NK-cells, T-cells, and B-cells. The complete analysis appears in the 'Resources' section.

Following feature selection, we evaluated our algorithm's ability to classify samples according to their known cell types. We employed our EM algorithm on the training data, making predictions for each sample by maximizing its posterior probability. Subsequently, we compared its performance with four variations of K-means clustering: two after feature selection and two before. Each pair of K-means models included one trained on standardized data, where each sample's sequencing depth was factored in, and one trained on the original training data.

As anticipated, cell type emerged as a robust signal, and all clustering algorithms outperformed random guessing, even without feature selection. Among the variations of K-means, the primary enhancement stemmed from data standardization, with feature selection contributing a more moderate improvement, likely by augmenting the signal-to-noise ratio. Notably, our algorithm yielded the best results, approaching near-perfect accuracy rates of 0.97 on the training set and 0.91 on the test set.

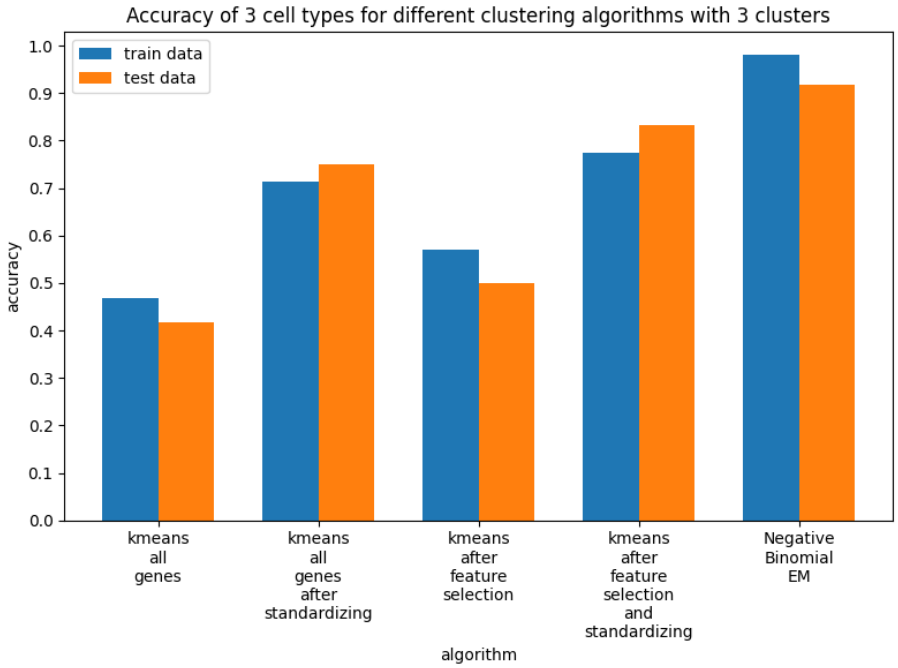


Figure 6: Accuracy of Predicting the cell type of a sample.

4.3 Comparison to Other Models

As previously mentioned, our model belongs to the class of mixture models, wherein a probability distribution is expressed as a composite of multiple component distributions. We employ algorithms like the Expectation-Maximization (EM) to estimate the parameters of these component distributions. This approach serves as a clustering method, associating each data point with its most probable cluster. A widely used mixture model is the Gaussian Mixture Model (GMM).

There exists a connection between the Negative Binomial distribution and the Normal distribution. Specifically, the Negative Binomial distribution NB (r, p) can be interpreted as the sum of r independent and identically distributed Geometric random variables with a parameter p. As r becomes large, following the Central Limit Theorem (CLT), this sum approximates a normal distribution. Consequently, our Negative Binomial mixture can be approximated by a GMM. Thus, we aimed to compare our algorithm to other EM algorithms that model data using some form of GMM.

We explore three types of GMMs: one with different covariance for each cluster, another with different diagonal covariance for each cluster, and a third GMM in which all variables have the same variance across all states.

The first type has $K\left(N^2+N\right)+K-1$ parameters, the second has $2NK+K-1$ parameters, and the third has $N(K+1)+K-1$ parameters. As the model complexity increases with more parameters, a fair comparison to our algorithm is with the second GMM, as it has the same number of parameters. The parameters of the third GMM are learned using the K-means algorithm, which is a variant of the EM algorithm, assuming a GMM with each Gaussian having a spherical shape with the same variance [7]. The comparison was based on the log-likelihood of each model with respect to the parameters learned by the different EM algorithms.

We first compared these four algorithms on synthetic data which was sampled according to our model, assuming a constant sequencing depth. Our algorithm outperformed all other algorithms in this scenario, which is nontrivial since the regular GMM has more parameters. This suggests that our algorithm is better at analyzing data distributed according to our model compared to other popular algorithms.

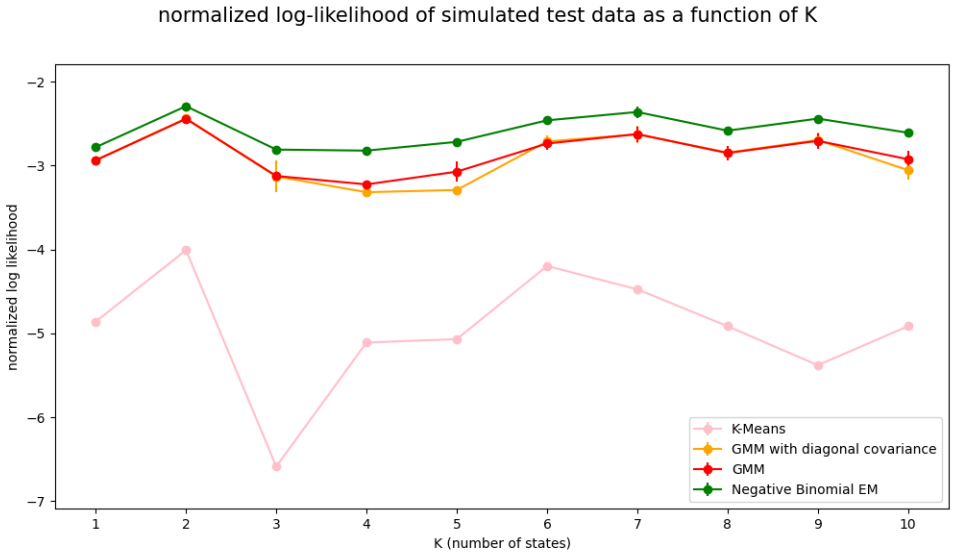


Figure 7: The models comparison on synthetic test data. For each K, we generated different random parameters, generated train and test datasets with those parameters according to our model, ran the different algorithms on the train dataset with a few initializations and calculated test log-likelihood for each algorithm.

We also conducted this comparison on real training and test datasets. This time, we ran our algorithm twice: once with a constant sequencing depth and once considering the real sequencing depth. Several observations were made:

1. The regular GMM exhibited clear overfitting, with the train log-likelihood increasing while the test log-likelihood decreased sharply.

2. Our clustering algorithm performed better than K-means and the diagonal-covariance GMM on the test set, indicating that our model provides a superior data description, making our algorithm more practical on this kind of dataset than other similar popular algorithms. However, considering sequencing depth did not lead to improved performance on either the train or test datasets.

3. Typically, all clustering algorithms should show an upward trend in the train log-likelihood as K (the number of clusters) increases, as greater complexity allows for a better model fit. However, our EM algorithm displayed a deviation from this trend around $K = 6$, suggesting room for improvement in our initialization strategy for real data, which we will address in the Discussion section.

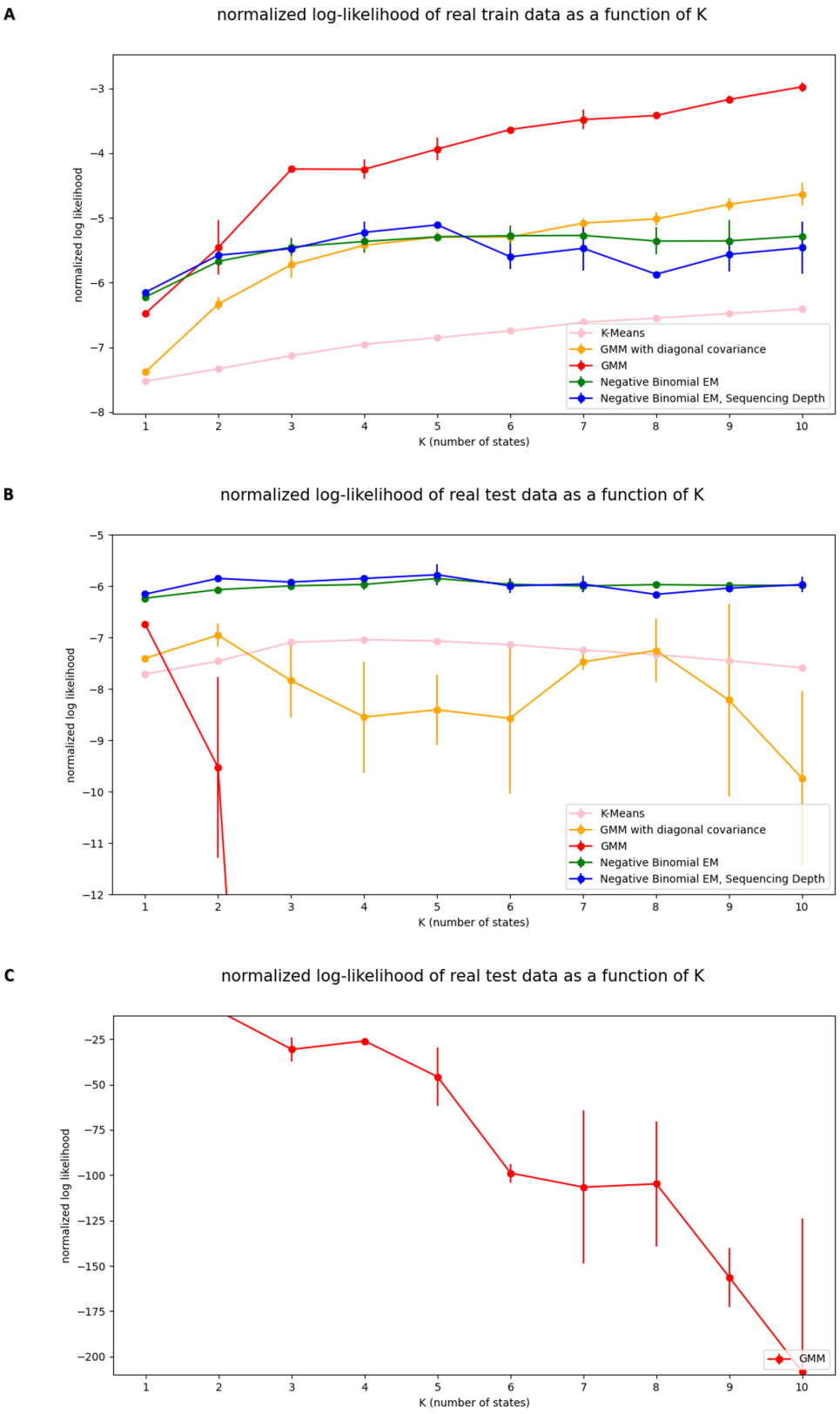


Figure 8: The models comparison on the real data. The performances of the GMM on the test data is shown in a seperate plot.

4.4 Hyper-Parameter Selection using BIC

A widely-used regularization technique for many models is the Bayesian Information Criterion (BIC) [11]. This criterion is mathematically defined as

$$BIC = \mathcal{K} \cdot \ln(M) - 2 \cdot \ln\left(\hat{\mathcal{L}}\right) \tag{41}$$

Where $\mathcal{K}$ represents the total number of parameters in the model, M is the sample size, and $\hat{\mathcal{L}} = p\left(\mathcal{D}_M; \hat{\theta}_{MLE}\right)$ stands for the log-likelihood achieved by the model on the training dataset $\mathcal{D}_M$ using the maximum likelihood estimator $\hat{\theta}_{MLE}$. As the model's complexity increases, the second term decreases, while the first term - the panelty - increases. Consequently, this formula assigns the lowest value to the best-fitting model.

When we generated a synthetic dataset with $K = 5$ and ran the algorithm with different K values, the BIC exhibited a minimum at $K = 5$, suggesting that this approach performs well with our algorithm and data following our model.

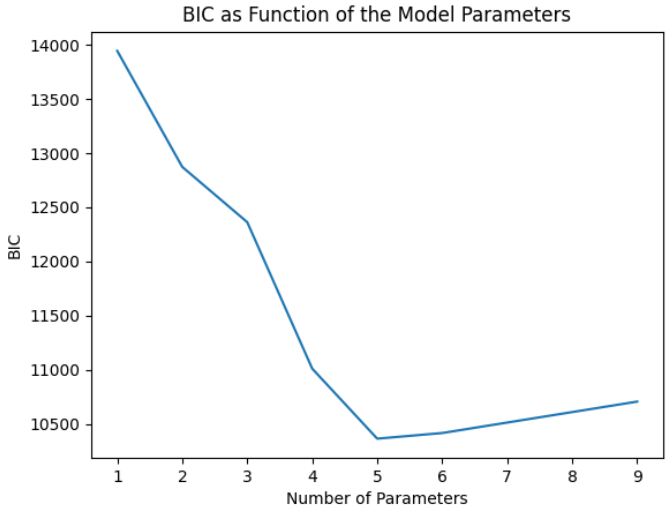


Figure 9: BIC plot on simulated data with $K = 5$, with sequencing depth.

Subsequently, we conducted the same analysis on the actual training data, once with a constant sequencing depth and once with the real sequencing depth. With constant sequencing depth, the graph displayed a minimum at $K = 3$, possibly indicating a strong signal of 3 distinct cell types. In contrast, with the real sequencing depth, the graph showed a minimum at $K = 5$, which is a reasonable result. There is a minor issue with the fact that the second graph displays a local minimum at $K = 2$. BIC is composed of two components: one is a monotonically increasing penalty factor, and the other incorporates the likelihood, which should exhibit a monotonically decreasing trend. Consequently, the typical behavior of the BIC graph is characterized by a single global minimum. However, situations may arise where a plateau appears in the likelihood factor, resulting in the presence of local minima. Nevertheless, it's important to consider the possibility that this observed local minimum is primarily due to a decrease in likelihood between certain values of K, as evident in the clustering comparison plot. As mentioned before, this does not imply a flaw in our algorithm but rather a suboptimal initialization strategy for this plot since the algorithm runs independently for each K and may converge to a local maximum in some cases. We will delve into this issue further in the Discussion section.

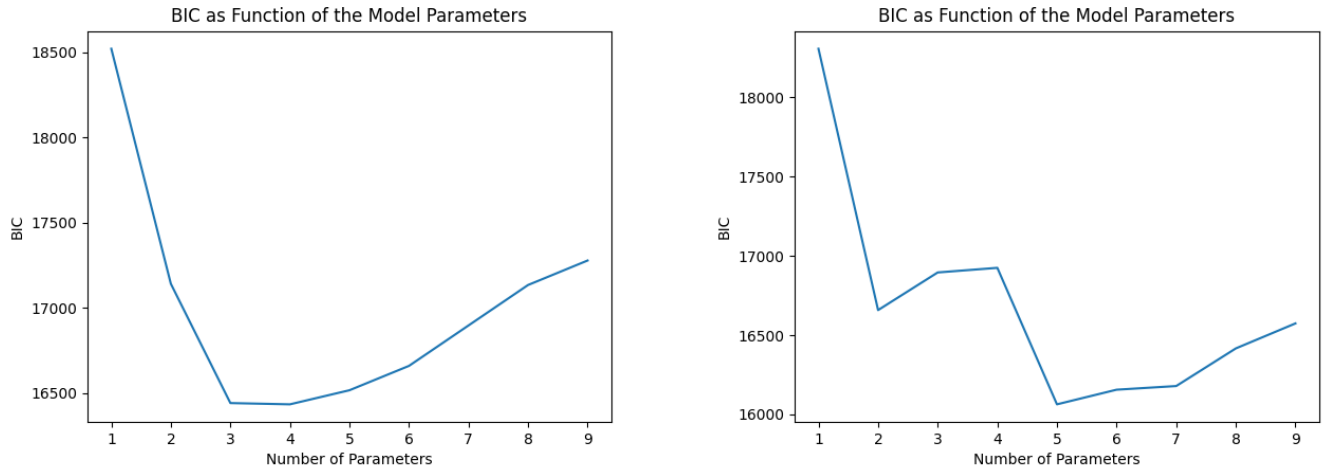


Figure 10: BIC plot on the real train data. Left plot - constant sequencing depth, Right plot - considering the real sequencting depth.

5 Discussion

5.1 Summary

In our project, our aim was to create a tool to explore the relationships between genes through their joint distribution. To achieve this goal, we developed a clustering algorithm centered on estimating the Maximum Likelihood Estimators (MLE) for parameters in a mixture model featuring Negative Binomial Conditional Probability Distributions (CPD). We employed the Naive Bayes model, which assumes that gene behaviors are conditionally independent given an underlying cell state. This assumption, although stringent, substantially reduced the parameters governing the joint distribution. This reduction addressed the challenge of high-dimensionality in our data, commonly referred to as the "curse of dimensionality." As the statistician George Box famously said, "All models are wrong, but some are useful."

Our model also took into account sequencing depth, a parameter influencing the depth of sample sequencing, which can introduce significant variance across samples. This approach had not been previously explored for modeling gene expression. In particular, in Barash et al. [3] we reviewed before, the CPD's are modeled with categorical distribution, and the sequencing depth is not addressed at all.

To estimate the parameters of our model, we utilized the Expectation-Maximization (EM) algorithm. We subsequently presented the performance results of our algorithm, demonstrating its effectiveness through various simulations on synthetic data generated from the distribution our model assumes, as well as on real gene expression data.

5.2 Conclusions Based on the Results

Our algorithm exhibits superior performance in clustering genetic data compared to similar algorithms such as K-means and diagonal GMM. This superiority is evident in its higher accuracy in identifying cell types and its test log-likelihood on real genetic datasets. Furthermore, it demonstrates a lower susceptibility to overfitting when compared to more complex models like the regular GMM. it is worth noting that the diagonal GMM, a model with the same number of parameters as the one at the core of our algorithm, also appears to suffer from overfitting, as suggested by the test log-likelihood plot in the clustering comparison section, where a sharp decline begins at $K = 8$.

However, it is plausible that our algorithm may also experience overfitting, although this cannot be definitively inferred from the log-likelihood plot. This is because the log-likelihood on the training data did not improve as K increased, suggesting that our algorithm might converge to a local maximum. We will explore the possibility of implementing a more intelligent initialization strategy for this plot in the Future Work section.

As previously mentioned, it is widely acknowledged that sequencing depth plays a pivotal role in introducing variance between samples, thereby complicating parameter estimation for any model. This assertion finds support in the accuracy plot, where the standardization of data by normalizing each sample according to its sequencing depth significantly improved the ability to predict cell types across various K-means algorithm variations.

However, when examining the log-likelihood plots, it appears that accounting for sequencing depth within our algorithm did not have a pronounced impact on the log-likelihood. Several factors could contribute to this observation. One possibility is that our dataset may not have been sufficiently large, and employing larger training and test datasets could potentially reveal the advantageous effects of a model that addresses this issue. Another explanation relates to the concern raised in the preceding paragraph, wherein our EM algorithm may not be converging to the global maximum. It is conceivable that achieving global convergence could illuminate the disparities between the two variations of our algorithm.

5.3 Limitations

A significant constraint of our algorithm is its scalability issue concerning higher dimensions, specifically when dealing with a greater number of genes. In theory, the runtime complexity of our algorithm is $O(NK)$, meaning it scales linearly with both the number of states and the number of genes. However, in practical application, when running it on datasets containing 100 genes or more, it experiences substantial slowdowns, diminishing its appeal as a commonly used tool. Consequently, we had to implement some form of feature selection to obtain a manageable dataset for our algorithm.

Furthermore, our model grapples with a minor technical challenge. Specifically, if θ_C contains a zero component or if there exist $p_{i,k}$ values of 0 or 1, certain factors such as $\log \theta_C[k]$, $\log p_{i,k}$ and $\log(1-p_{i,k})$ within the likelihood calculation yield illegal (NaN) values, consequently rendering the entire likelihood calculation invalid. We attempted to mitigate this issue by introducing a small epsilon value (addition or subtraction) to these variables when encountering this problem. However, this adjustment led to another complication where the marginal likelihood occasionally decreased between iterations. This occurred because we selected values for θ_C and $p_{i,k}$ heuristically, without a guaranteed improvement in the likelihood. A possible solution is to write the likelihood function a bit differently: Recalling that the NB distribution is a mixure of Gamma and Poisson, we can write the likelihood using this mixture, so our parameters would be α and β of the Gamma distribution, from which we could calculate r and p of the NB, which are a one-to-one function of α and β. The estimation of that parameters of Gamma distribution might be easier that the NB distribution.

5.4 Future Work

Firstly, it is essential to ensure that when we execute the algorithm with various K values, as demonstrated in the clustering comparison, we consistently observe an increase in log-likelihood on the training dataset. In practice, we have encountered situations where this doesn't occur, indicating that our algorithm may converge to local maxima. To address this challenge, one potential solution to explore involves taking the output parameters of the algorithm for each K, dividing the largest cluster into two equal clusters, estimating the parameters for these new clusters, and then initializing the algorithm with $K+1$ using this modified initialization. This approach is expected to yield an improved likelihood compared to the output from the previous algorithm run.

Another avenue for future work involves running the algorithm without feature selection on robust hardware infrastructure like a large CPU cluster to evaluate the results. While this may entail longer training times, the compelling results obtained could enable biologists to utilize the pre-trained model in their research without the need for retraining.

Lastly, we need to address the fundamental question that accompanies every clustering algorithm: How do we choose K, i.e., the number of clusters or hidden states for our model? One approach we've considered is a simple regularization of the BIC criterion, but it has its limitations. This issue extends to model selection in general. For our model, we might explore more advanced models, such as Naive-Bayes with missing edges or Context-Specific independence, similar to Barash et al.'s work. To address these, we would require more sophisticated regularization techniques of Bayesian learning, involving what's known as **structure learning**. In this advanced concept, we assume a prior distribution over the graph's structure, penalizing large numbers of states and the presence of edges. Additionally, given a structure $\mathcal{M}$, we assume a prior distribution over $\theta \mid \mathcal{M}$, and subsequently employ the EM algorithm to find a Maximum A Posteriori (MAP) estimator for the structure and its parameters concurrently.

6 Resources

1. **Code -** The code for our project can be found in `https://github.cs.huji.ac.il/nettashaf/final_project`. All the code is located under the directory 'EM_implementations'. The file 'main.py' runs our algorithm on a simulated data, the file 'Preprocessing.py' execute all the preprocessing on the real data including the train-test split, and the file 'plots.py' executes all the plots in this report. In addition, in the head of the file 'utils.py' there is all the configurations and the hyperparameters of the algorithm. The file 'requirements.txt' contains all the packages in our environmet, in which all the code run properly.
   For implementing the categorical case, we got some insperation from that repository:
   `https://github.com/diningphil/Multinomial_Mixture_Model/tree/efbfcf68b9296688bd5482eb9630872ed0e64750`

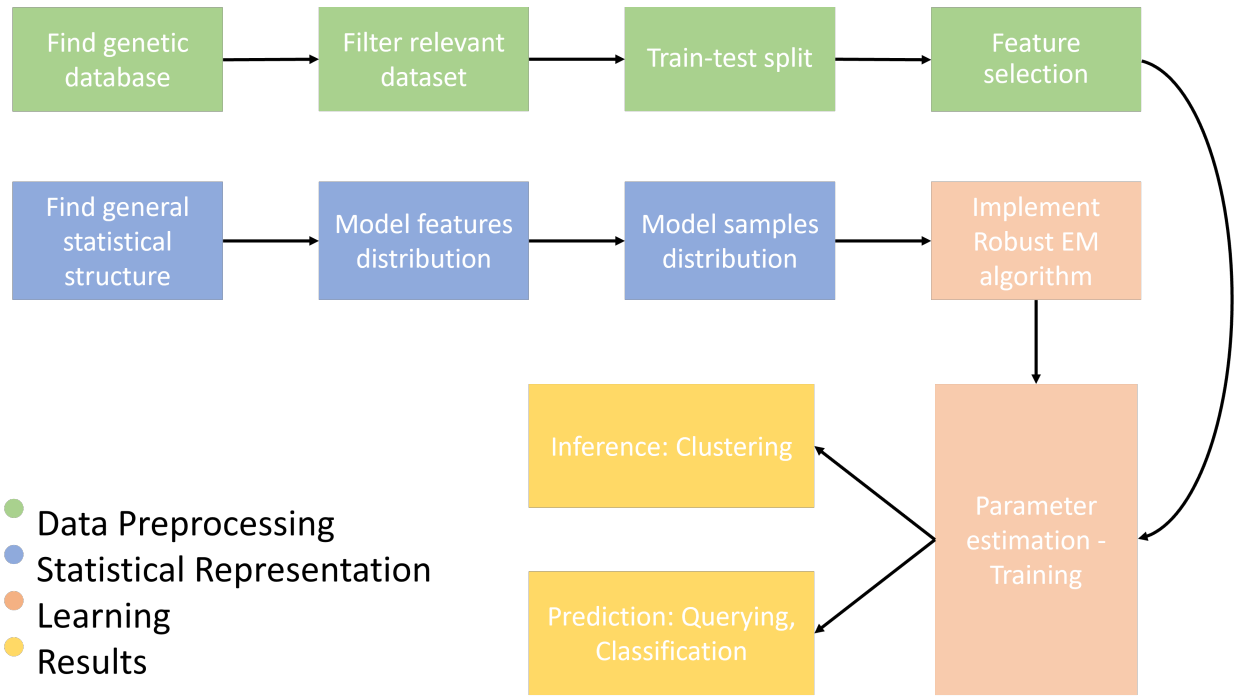
2. **Data -** The data was come from the **BLUEPRINT** project, which is a open source of genetic data:
   `https://projects.ensembl.org/blueprint/`.
   The relevant dataset for our project can be found in our repository, under the directory 'Data'. The raw data that have been taken from BLUEPRING, and contains more cells than were not suitable for our needs, was too large to be saved in the repository, or to be submitted. The file 'preprocessing.R' extracts from the raw the dataset of our project, and save it in the three files 'data.csv', 'seq_depth.csv' and 'tags.csv'.

3. **Enrichr -** The analysis of the significance cells of the genes in the three clusters that used us for feature selection.
   The cluster that related to NK-cells and T-cells:
   `https://maayanlab.cloud/Enrichr/enrich?dataset=d85b4f88310b64e6c235e059a2b6cfb0`
   The cluster that related to B-cells:
   `https://maayanlab.cloud/Enrichr/enrich?dataset=75aacaf921125b73b60bcefa00610b9e`
   Another cluster that doesn't seem to be related to neither of them:
   `https://maayanlab.cloud/Enrichr/enrich?dataset=c696ab59adeca3c1ca28ce947a557b82`
   On the upper bar, go to 'Cell Types' for watch the different databases, and the different significant.

4. **RefSeq** - The RefSeq project, from which we took the list of genes for our initial feature selection, can be found in:
   `https://genome.ucsc.edu/cgi-bin/hgTables?db=hg19&hgta_group=genes&hgta_track=refSeqComposite&hgta_table=refGene&hgta_doSchema=d`

7 Work Pipeline

In this section, we will describe the stages of our work in this project. We only explain briefly about every stage, because all the information has been discussed alreday earlier in this report.



7.1 Data Preprocessing

This stage involved the processing of actual genetic data. Our model and algorithm are adaptable for various types of genetic data that serve as indicators for protein expression. Therefore, the initial step is to locate a suitable database for the specific data type, such as DNA sequencing (as in our case) or RNA sequencing (as an alternative example). Following that, we refined our dataset by selecting relevant samples. We have done it with lymphocytes, whereas liver cells or multiple cell types could be chosen in different scenarios. Subsequently, we partitioned our dataset into training and testing subsets and conduct feature selection to focus on specific segments of the entire genome. In our case, we utilize the RefSeq database and employ a significance criterion based on K-means clustering for this purpose.

7.2 Statistical Representation

In this stage, we focus on constructing a statistical model for the data. Initially, we established our overarching model as Naive-Bayes, and subsequently, we characterized the CPDs as Negative Binomial, although there are several alternative options available for this choice. Following that, we outlined the distribution of the samples within our dataset, taking into account that they are not uniformly distributed but influenced by their respective sequencing depths.

7.3 Learning

Here we implemented he EM algorithm for learning the parameters of our model, and ran it on our train dataset. This is the training stage.

7.4 Results

Once we've successfully trained our model, it becomes a valuable tool for performing inference on the training data, or prediction on the test data or on future data. In the realm of inference, the primary objective is clustering, where we assign each sample to a particular state by selecting the state that maximizes the posterior probability based on the learned parameters. Alternatively, we can perform a soft clustering by examining the posterior probabilities and assessing the likelihood of each sample belonging to each state. This approach is similar to what we undertook in the clustering comparison section, where we evaluated the overall likelihood of the training samples under our model as compared to other models after training.

In the context of prediction, one task involves classification, where we associate a data point with one of the states learned during training. This process is similar to what we conducted in the accuracy comparison. Additionally, within our algorithm, there's another option for prediction: leveraging the predictive distribution of the trained model to address various queries concerning gene relationships. For example, if someone observe the genes X_i and X_j, she could easily get the marginal distribution of the gene X_k using the factorization:

$$p\left(X_k \mid X_i = x_i, X_j = x_j\right) = \frac{p\left(X_k, X_i = x_i, X_j = x_j\right)}{p\left(X_i = x_i, X_j = x_j\right)} = \frac{\sum_k p\left(C = k\right) p\left(X_k \mid C = k\right) p\left(X_i = x_i \mid C = k\right) p\left(X_j = x_j \mid C = k\right)}{\sum_k p\left(C = k\right) p\left(X_i = x_i \mid C = k\right) p\left(X_j = x_j \mid C = k\right)}$$

where all the factors in the last formula can be calculated easily with the parameters we learned.

References

[1] Dempster, A. P., et al. "Maximum Likelihood from Incomplete Data via the EM Algorithm." Journal of the Royal Statistical Society. Series B (Methodological), vol. 39, no. 1, 1977, pp. 1–38. JSTOR, http://www.jstor.org/stable/2984875.

[2] Martens JH, Stunnenberg HG. BLUEPRINT: mapping human blood cell epigenomes. Haematologica. 2013 Oct;98(10):1487-9. doi: 10.3324/haematol.2013.094243. PMID: 24091925; PMCID: PMC3789449.

[3] Yoseph Barash and Nir Friedman. Context-Specific Bayesian Clustering for Gene Expression Data. Journal of Computational Biology.Apr 2002.169-191.http://doi.org/10.1089/10665270252935403

[4] Sadeh, R., Sharkia, I., Fialkoff, G. et al. Author Correction: ChIP-seq of plasma cell-free nucleosomes identifies gene expression programs of the cells of origin. Nat Biotechnol 39, 642 (2021). https://doi.org/10.1038/s41587-021-00853-3

[5] Lipp, Jesse, "WHY SEQUENCING DATA IS MODELED AS NEGATIVE BINOMIAL", Wordpress, BIORAMLE, https://bioramble.wordpress.com/2016/01/30/why-sequencing-data-is-modeled-as-negative-binomial/, January 30, 2016

[6] Zhou M, Li L, Dunson D, Carin L. Lognormal and Gamma Mixed Negative Binomial Regression. Proc Int Conf Mach Learn. 2012;2012:1343-1350. PMID: 25279391; PMCID: PMC4180062.

[7] Eva Patel, Dharmender Singh Kushwaha, Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model, Procedia Computer Science, Volume 171, 2020, Pages 158-167, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.04.017.

[8] Pruitt KD, Brown GR, Hiatt SM, Thibaud-Nissen F, Astashyn A, Ermolaeva O, Farrell CM, Hart J, Landrum MJ, McGarvey KM et al. RefSeq: an update on mammalian reference sequences. Nucleic Acids Res. 2014 Jan;42(Database issue):D756-63. PMID: 24259432; PMC: PMC3965018

[9] Tatusova T, DiCuccio M, Badretdin A, Chetvernin V, Nawrocki EP, Zaslavsky L, Lomsadze A, Pruitt KD, Borodovsky M, Ostell J. NCBI prokaryotic genome annotation pipeline. Nucleic Acids Res. 2016 Aug 19;44(14):6614-24 PubMed PubMedCentral

[10] Xie Z, Bailey A, Kuleshov MV, Clarke DJB., Evangelista JE, Jenkins SL, Lachmann A, Wojciechowicz ML, Kropiwnicki E, Jagodnik KM, Jeon M, & Ma'ayan A. Gene set knowledge discovery with Enrichr. Current Protocols, 1, e90. 2021. doi: 10.1002/cpz1.90

[11] Schwarz, Gideon. "Estimating the Dimension of a Model." The Annals of Statistics, vol. 6, no. 2, 1978, pp. 461–64. JSTOR, http://www.jstor.org/stable/2958889. Accessed 26 Sept. 2023.

Appendix - Mathematical Formulation

Complete log-likelihood of the data

For the Naive-Bayes model, with any CPD's

$$\begin{aligned}
\ell\left(\theta; \mathcal{D}_M, \overrightarrow{c}\right) &= \sum_{m=1}^{M} \log p\left(X = x\left[m\right], C = c\left[m\right]\right) \\
&= \sum_{m=1}^{M} \log\left(p\left(C\left[m\right] = c\left[m\right]\right) p\left(X = x\left[m\right] \mid C = c\left[m\right]\right)\right) \\
&= \sum_{m=1}^{M} \log p\left(C = c\left[m\right]\right) + \sum_{i=1}^{N} \log p\left(X_i = x_i\left[m\right] \mid C = c\left[m\right]\right) \\
&= \sum_{k=1}^{K} \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \log p\left(C = k\right) + \sum_{i=1}^{N} \log p\left(X_i = x_i\left[m\right] \mid C = k\right) \\
&= \sum_{k=1}^{K} \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \log \theta_C\left[k\right] + \sum_{i=1}^{N} \log p\left(X_i = x_i\left[m\right] \mid C = k\right)
\end{aligned}$$

having $X_i | \left\{C = k\right\} \sim \text{NB}\left(r_{i,k}, p_{i,k}\right)$ and $X_i\left[m\right] | \left\{C = k\right\}, \vec{\eta} \sim \text{NB}\left(r_{i,k}, \frac{p_{i,k}}{p_{i,k} + \eta[m] - p_{i,k}\eta[m]}\right)$, it holds that

$$\begin{aligned}
\log p\left(X_i\left[m\right] = x \mid C = k\right) = &\log\Gamma\left(x_i + r_{i,k}\right) - \log\Gamma\left(x_i + 1\right) - \log\Gamma\left(r_{i,k}\right) \\
&+ r_{i,k} \cdot \log\left(\frac{p_{i,k}}{p_{i,k} + \eta\left[m\right] - p_{i,k}\eta\left[m\right]}\right) + x_i \cdot \log\left(1 - \frac{p_{i,k}}{p_{i,k} + \eta\left[m\right] - p_{i,k}\eta\left[m\right]}\right)
\end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned}
\log p\left(X_i\left[m\right] = x | C = k\right) = &\log\Gamma\left(x_i + r_{i,k}\right) - \log\Gamma\left(x_i + 1\right) - \log\Gamma\left(r_{i,k}\right) \\
&+ r_{i,k} \cdot \log\left(\frac{p_{i,k}}{p_{i,k} + \eta\left[m\right] - p_{i,k}\eta\left[m\right]}\right) + x_i \cdot \log\left(\frac{\eta\left[m\right] \cdot \left(1 - p_{i,k}\right)}{p_{i,k} + \eta\left[m\right] - p_{i,k}\eta\left[m\right]}\right)
\end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned}
\log p\left(X_i\left[m\right] = x | C = k\right) = &\log\Gamma\left(x_i + r_{i,k}\right) - \log\Gamma\left(x_i + 1\right) - \log\Gamma\left(r_{i,k}\right) \\
&+ r_{i,k} \cdot \left[\log p_{ik} - \log\left(p_{ik} + \eta\left[m\right] - p_{ik}\eta\left[m\right]\right)\right] \\
&+ x_i \cdot \left[\log\left(\eta\left[m\right]\right) + \log\left(1 - p_{ik}\right) - \log\left(p_{ik} + \eta\left[m\right] - p_{ik}\eta\left[m\right]\right)\right]
\end{aligned}$$

and consenquently, now when $\theta = \left\{\theta^C, \left\{r_{i,k}\right\}_{i,k}, \left\{p_{i,k}\right\}_{i,k}\right\}$, it holds that

$$\begin{aligned}
\ell\left(\theta; \mathcal{D}_M, \overrightarrow{c}, \overrightarrow{\eta}\right) = &\sum_{k=1}^{K} \sum_{m=1}^{M} \mathbb{1}_{\{c[m]=k\}} \log \theta_C\left[k\right] + \sum_{i=1}^{N} \log\Gamma\left(x_i + r_{i,k}\right) - \log\Gamma\left(x_i + 1\right) - \log\Gamma\left(r_{i,k}\right) \\
&+ r_{i,k} \cdot \left[\log p_{ik} - \log\left(p_{ik} + \eta\left[m\right] - p_{ik}\eta\left[m\right]\right)\right] \\
&+ x_i \cdot \left[\log \eta\left[m\right] + \log\left(1 - p_{ik}\right) - \log\left(p_{ik} + \eta\left[m\right] - p_{ik}\eta\left[m\right]\right)\right]
\end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned}\ell\left(\theta;\mathcal{D}_M,\overrightarrow{c},\overrightarrow{\eta}\right)=&\sum_{k=1}^{K}\left[\left(\sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\right)\log\theta_C\left[k\right]\right.\\&+\sum_{i=1}^{N}\left[\sum_{d=0}^{\max X_i}\left(\sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\mathbb{1}_{\{x_i[m]=d\}}\right)\log\Gamma\left(d+r_{i,k}\right)\right.\\&+\left(\sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\right)\left(r_{i,k}\log p_{ik}-\log\Gamma\left(r_{i,k}\right)\right)\\&+\left(\sum_{m=1}^{M}x_i\left[m\right]\cdot\mathbb{1}_{\{c[m]=k\}}\right)\log\left(1-p_{ik}\right)\\&\left.\left.-\sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\cdot\left(r_{i,k}+x_i\left[m\right]\right)\cdot\log\left(p_{ik}+\eta\left[m\right]-p_{ik}\eta\left[m\right]\right)\right]\right]\\&+const\left(\theta\right)\end{aligned}$$

and we can notice that then $\eta\left[m\right]=1$ for every m, then we get the complete log-likelihood of the regular NB case without sequencing depth.

Finding MLE $\hat{\theta}^C$ usign Lagrange multipliers

Our Lagrangian is

$$\mathcal{L}=\ell-\lambda\left(\sum_{k=1}^{K}\theta_k^C-1\right)$$

Differentiating it with respect to θ_k^C, we get

$$\frac{\partial}{\partial\theta_k^C}\mathcal{L}=S_C\left[k\right]\frac{1}{\theta_C\left[k\right]}-\lambda\overset{!}{=}0\Leftrightarrow\hat{\theta}_C\left[k\right]=\frac{S_C\left[k\right]}{\lambda}$$

Differentiating with respect to λ, we get

$$\sum_{k=1}^{K}\hat{\theta}_C\left[k\right]=1\Rightarrow\sum_{k=1}^{K}\frac{S_C\left[k\right]}{\lambda}=1\Leftrightarrow\hat{\lambda}=\sum_{k=1}^{K}S_C\left[k\right]=M$$

Finally we get

$$\hat{\theta}_C\left[k\right]=\frac{S_C\left[k\right]}{M}$$

Sequencing depth modeling

If $X|\lambda\sim\text{Poisson}\left(\lambda\right)$ when $\lambda\sim\text{Gamma}\left(\alpha,\beta\right)$, it holds that $X\sim\text{NB}\left(r,p\right)$ when $r=\alpha$ and

$$\beta=\frac{1}{1-p}\Leftrightarrow p=\frac{\beta}{\beta+1}$$

Now we have another variable $\widetilde{X}$, such that $\widetilde{X}|\lambda\sim\text{Poisson}\left(\eta\cdot\lambda\right)$. Recall the following propery of Gamma distribution: If $\lambda\sim\text{Gamma}\left(\alpha,\beta\right)$ then for any $\eta>0$ it hold that $\eta\cdot\lambda\sim\text{Gamma}\left(\alpha,\frac{\beta}{\eta}\right)$. Therefore, it hold that $\widetilde{X}\sim\text{NB}\left(r_\eta,p_\eta\right)$, when $r_\eta=\alpha$ and $p_\eta=\frac{\frac{\beta}{\eta}}{\frac{\beta}{\eta}+1}$. Now, it holds that:

$$p_\eta=\frac{\frac{\beta}{\eta}}{\frac{\beta}{\eta}+1}=\frac{\beta}{\beta+\eta}=\frac{\frac{1}{1-p}}{\frac{1}{1-p}+\eta}=\frac{p}{p+\left(1-p\right)\eta}=\frac{p}{p+\eta-p\eta}$$

So in the terms of the parameters $\left(r,p\right)$ of the variable X, it means that

$$\widetilde{X}\sim\text{NB}\left(r,\frac{p}{p+\eta-p\eta}\right)$$

Second derivative of the likelihood

After taking the sequencing depth into considerations, in each CPD we need to find a root for the gradient both r and p simultaneously with solvers like Newton-Raphson, which requires the second derivatives. If so:

$$\frac{\partial^2}{\partial^2 p_{i,k}}\ell = -\frac{r}{p^2}S_C\left[k\right] - \frac{1}{\left(1-p\right)^2}S_{sum}\left[i,k\right] + \sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\cdot\left(r+x_i\left[m\right]\right)\cdot\left(\frac{1-\eta\left[m\right]}{p+\eta\left[m\right]-p\eta\left[m\right]}\right)^2 \tag{42}$$

$$\frac{\partial^2}{\partial^2 r_{i,k}}\ell = \sum_{d=0}^{\max X_i} S_{count}\left[i,k,d\right]\cdot\psi^{(1)}\left(d+r\right) - S_C\left[k\right]\cdot\psi^{(1)}\left(r\right) \tag{43}$$

$$\frac{\partial^2}{\partial p_{i,k}\partial r_{i,k}}\ell = \frac{\partial^2}{\partial r_{i,k}\partial p_{i,k}}\ell = \frac{1}{p}\cdot S_C\left[k\right] - \sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\cdot\frac{1-\eta\left[m\right]}{p+\eta\left[m\right]-p\eta\left[m\right]} \tag{44}$$

$$= \sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\cdot\left(\frac{1}{p} - \frac{1-\eta\left[m\right]}{p+\eta\left[m\right]-p\eta\left[m\right]}\right) \tag{45}$$

$$= \frac{1}{p}\sum_{m=1}^{M}\mathbb{1}_{\{c[m]=k\}}\cdot\frac{\eta\left[m\right]}{p+\eta\left[m\right]-p\eta\left[m\right]} \tag{46}$$

Where $\psi^{(1)}$ is te first order polygamma function, which define as $\psi^{(1)} := \frac{\partial^2}{\partial^2 x}\log\Gamma\left(x\right)$. The iteration step of Newton-Raphson is

$$\binom{r_{i,k}}{p_{i,k}}^{(t)} = \binom{r_{i,k}}{p_{i,k}}^{(t-1)} - \begin{bmatrix} \frac{\partial^2}{\partial^2 r_{i,k}}\ell & \frac{\partial^2}{\partial p_{i,k}\partial r_{i,k}}\ell \\ \frac{\partial^2}{\partial r_{i,k}\partial p_{i,k}}\ell & \frac{\partial^2}{\partial^2 p_{i,k}}\ell \end{bmatrix}^{-1}\begin{bmatrix} \frac{\partial}{\partial r_{i,k}}\ell \\ \frac{\partial}{\partial p_{i,k}}\ell \end{bmatrix} \tag{47}$$

Appendix B - Context-Specific Clustering

In the article "Context-Specific Bayesian Clustering for Gene Expression Data" [3], each sample in the data refers to a particular gene, and the features are a series of experiments. The features are denoted as a series of random variables $X_1, ..., X_N$ and they are assumed to be independent given a hidden random variable C. In the article, the authors present a model called Context-Specific-Independence Network, which suggests that each X_i has a group $\mathcal{L}_i \subseteq \text{Supp}\left(C\right)$, so that the distribution of X_i depends on $k \in C$ if $k \in \mathcal{L}_i$, and else gets a default probability. Formally:

$$p\left(X_i|C=k\right) = \begin{cases} p\left(X_i|C=k\right) & k \in \mathcal{L}_i \\ p_{def}\left(X_i\right) & \text{otherwise} \end{cases}$$

The intuition behind this is that each variable X_i might only be dependant in a subgroup of states. The value of K (the size of $\text{Supp}\left(C\right)$), and the groups $\mathcal{L}_i$ are the model's hyper-parameters. The article suggests the idea of structure learning, in which we study the model's structures and hyper-parameters together with it's parameters. The article assume a prior over K, over $\mathcal{L}_i$, and also over the $p\left(\theta \mid K, \mathcal{L}_i\right)$, when θ represents all the parameters of the model. Then, the article uses the EM algorithm to learn the MAP (Maximum aposteriori) estimators for all the parameters and hyper-parameters.