Shomik Ghose, Austin Tao

ML4FG Final Report: Modeling Colorectal Cancer Gene Expression Distributions using Mixture Models

Abstract

We implemented mixture models, an unsupervised machine learning technique, to best fit continuous gene expression distributions. Throughout our research, we both applied existing packages (for Gaussian and student-t mixture models) and developed our own implementations (for Gaussian and Shifted Asymmetric Laplace mixture models) to fit our data. Ultimately, we found that while our own implementations were moderately effective, a weighted average of Gaussian and student-t mixture models using the existing packages (but using a novel method to analytically determine weights) were best at fitting the distributions. We also developed our own model selection metrics, as we found that standard metrics such as Bayesian Information Criterion (BIC) did not truly identify which models best fit the distribution, especially for non-normal distributions.

1. Introduction

In genomics, the process of modeling continuous gene expression distributions is an ongoing research problem. While such distributions are often assumed to be normal, this assumption is often false. For example, distributions of cancer cells tend to display non-normal distributions, including those that are bimodal or unimodal but with a non-normal tail. Inaccurately assuming that these distributions are normal can have deleterious effects on analyses based off of the distributions; for example, the de Torrenté et al. paper demonstrated that blanket assumptions of normality on prognostic marker genes for cancer lowered the performance of survival analysis models. Thus, we aimed to develop a method to best fit gene expression distributions, particularly accounting for non-normality.

We used a microarray phenotype dataset gathered by Dr. Walter Bodmer, a researcher at Oxford. The dataset was sourced from a Github repository detailing the GMMchi package[1], a modeling package that Professor Bodmer helped develop and that we later attempted to apply to our data. The dataset consists of a table, where each row corresponds to a specific gene and each column corresponds to a colorectal cancer cell line. Each (row, column) entry represents the base-2-log of the gene expression for the cell line.

For our research, we experimented with several different variants of mixture models as best fits for our distributions. Briefly, mixture models are unsupervised learning clustering models that assume a given set of data follows a specific type of probability distribution, then attempt to find the parameters for the ideal distribution of that type that best fits the data. First, we used existing packages, including Gaussian mixture models through the sklearn.mixture[2] package and student-t mixture models through the smm package[3]. Next, we used our own approach, where we took a weighted average of the two model types. Lastly, we implemented our own mixture models from scratch, developing both Gaussian and Shifted Asymmetric Laplace (SAL) mixture models. In particular, our SAL implementation was novel, as it had never before been achieved in any existing research.

2. Methods

2.1 Data

Data was downloaded from the GMMchi repository referenced in Section 1 (Introduction). Upon initial inspection, the data contained 54,702 total row entries (genes), each including logarithmic gene expression values for 78 unique colorectal cancer cell lines. There were 21,077 unique genes represented in the dataset; 33,625 of the row indices were duplicates. The GMMchi researchers confirmed that every single gene expression distribution could be approximated by either a unimodal, unimodal with non-normal tail, or bimodal probability distribution.

---

[1] https://github.com/jeffliu6068/GMMchi

[2] https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture.score_samples

[3] https://t-student-mixture-models.readthedocs.io/en/latest/index.html#

2.2 Preprocessing

Our first step in preprocessing was to convert duplicate rows (rows with the same index) into unique entries. We chose to convert these rows into unique entries instead of removing them because even though these rows represented the same genes, the expression values included in them were different; this is likely because different iterations of a microarray would naturally yield altered results. We chose to handle duplicates by relabeling them with an underscore corresponding to the number of the duplicate (i.e. three rows corresponding to a gene named A would have their indices relabeled as A, A_1, and A_2).

Figure 1: Example of duplicate genes post-processing. The row index corresponding to the second appearance of CDH1 has been relabeled as CDH1_1. The value in row 1, column 1, 8.84, represents log_2(the expression value of gene CDH1 in the C10 cell line).

	C10	C106	C125PM	C32	C70	C75	C80
CDH1	8.84476	8.43063	9.05031	9.41713	8.56102	8.34133	10.6095
CDH1_1	11.83090	13.22360	12.34470	11.83150	11.90950	12.01860	13.2421

2 rows × 78 columns

Our second step in preprocessing was to handle null values because the fit() methods of the packages we were using would return errors if nan values were present in the data. However, we could not simply set null values to 0 or impute them to the mean of the other values because this would skew the gene expression distributions. Thus, we decided to drop null values from the data. We ensured this would not drop entire rows because we only dropped null values after we had already isolated a specific row to run analysis on, so this processing step was something we had to repeat each time we ran an analysis.

2.3 Existing Packages

Initially, we attempted to run GMMChi, an existing package of academic code optimized to fit both normal and non-normal distributions by leveraging Gaussian mixture modeling along with a chi-square fit to handle non-normal tails, on our data. Following GMMchi, we ran a selection of three methods (Gaussian mixture, student-t mixture, weighted average of Gaussian and student-t mixtures) we felt would generate good fits.

We used the scikit-learn package for Gaussian mixture models and the smm package for student-t mixture models. For the weighted average mixture model, we selected our weights through a novel analytical process based off of our Adjusted Least Squares (ALS) metric described in section 2.5. We calculated two "running scores", one for the Gaussian and one for the student-t mixture model, across the dataset. For each datapoint, we calculated the ALS at that point for each of the two types of model fits; we then chose the lower of the two ALS values (i.e. $min(ALS_{Gaussian}, ALS_{Student-t})$), as this would indicate a better "fit" for that point, and added that value to the corresponding running score. Lastly, we normalized the two running scores to add up to 1 and took the scores as our weights. This essentially allowed us to determine how much of the dataset each type of mixture model was a good fit for, and then use the goodness of that fit to weight the contribution of each model to our weighted average.

2.4 Our Implementation

Following the fits using existing packages, we built our own Gaussian and SAL mixture models from scratch, implementing the Expectation-Maximization (EM) algorithm. We adjusted two parameters (mean and variance) for Gaussian EM and three parameters (skewness, mean, variance) for SAL EM. The EM algorithm consists of

calculating the expected log-likelihood, maximizing this expectation with respect to the model parameters, and repeating until some convergence criteria is met. We show the final mathematical results for our two different EM algorithms below:

a. *Gaussian EM*[4]

Let the parameters for our model be designated as $\Theta = \{\pi, \mu, \Sigma\}$, where π is a mixing coefficient, μ is the mean, and Σ is the covariance matrix. Denote $\gamma(z_{nk})$ as follows,

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \mu_j, \Sigma_j)}$$

The expectation step is accomplished by calculating

$$\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z} \mid \theta^*)] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) [\log(\pi_k) + \log \mathcal{N}(\mathbf{x}_n \mid \mu_k, \Sigma_k)]$$

and the maximization step involves computing

$$\theta^* = \arg\max_{\theta} \mathbb{E}[\log p(\mathbf{X}, \mathbf{Z} \mid \theta^*)]$$

b. *Shifted Asymmetric Laplace EM*

The math for the SAL EM algorithm is considerably more complicated compared to the Gaussian case. We present a condensed summary of the algorithm's math: let $x_1, \ldots, x_n$ be the observed data, $\tau_1, \ldots, \tau_n$ be component membership labels where each label is an indicator vector, $W = [w_1, w_2, \ldots, w_i]$ is a vector of random variables from an exponential distribution with mean 1, $h(w_i)$ be the PDF of w_i, and $\{\mu, \boldsymbol{\alpha}, \Sigma\}$ be mean, skewness, and covariance, respectively. In the expectation step, we calculate the expectation of the likelihood function given by

$$\mathcal{L} = \prod_{i=1}^{n} \prod_{g=1}^{G} \left[\pi_g \phi \left(\mathbf{x}_i \mid \boldsymbol{\mu}_g + w_i \boldsymbol{\alpha}_g, w_i \boldsymbol{\Sigma}_g\right) h\left(w_i\right)\right]^{\tau_{ig}}$$

where

$$\phi \sim \mathcal{N}(\boldsymbol{\mu}_g + w_i \boldsymbol{\alpha}_g, w_i \boldsymbol{\Sigma}_g)$$

The maximization step procedure is identical to the Gaussian case, except the updates are far more complex. For the update equations, refer to Franczak et al.[5]

2.5 Model Fitting and Selection

For each individual row (gene), we conducted two types of tests. First, we conducted an "intra-model" test, where we determined which parameters (number of components, covariance matrix type, convergence threshold, random initialization method, distribution of weights for the weighted average model, etc…) would cause each type of model to best fit the data; in particular, figuring out the optimal number of components (i.e. 1 v. 2) was important to our ability to model the distribution. For example, if we had a unimodal distribution with a non-normal tail, fitting with a 1 component model would largely ignore the tail while a 2 component model would account for the tail; thus,

[4] https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95
[5] https://arxiv.org/abs/1207.1727

determining which model was better could give us information on the importance of the tail to the general data distribution.

Second, we conducted an "inter-model" test where we figured out which one of our three methods (Gaussian, student-t, weighted average) would best fit the distribution.

To graph our fit, we first produced a histogram of the gene expression distribution, with the x-axis containing the base-2-log of the gene expression values and the y-axis representing the frequency (as a ratio instead of as a count) of the number of cell line samples that fall into each expression value bucket. We then placed a curve representing the log-likelihood of each sample (based on the mixture model we just fit to the data) onto the histogram. Lastly, since our primary goal with the fit was to match shape (not exact values), we normalized the values of the curve to match the y-axis values of the histogram so that the two could be evaluated on the same scale.

In both types of tests, we aimed to measure the "best fit" using the Bayesian information criterion (BIC).

$$\mathrm{BIC} = k \log(n) - 2 \log L_{M,G}(x \mid \hat{\theta})$$

where $L_{M,G}$ is the maximized likelihood function of a model M with G number of components, n is the sample size, and k is the number of estimated parameters. In particular, since BIC incorporates a penalty term that increases proportionally with the increase of parameters, this metric is effective at countering overfitting. Following with past research papers, we used a lower BIC to denote a "better fit".

We also developed two of our own metrics to measure model fit, both based on quantifying the difference between the histogram of the gene expression distribution and the mixture model curve that was fit to the distribution.

The first metric, which we titled "approximate least squares" (ALS) is as follows:

$$\mathrm{ALS} = \sum_{i=1}^{n} (y_i - b_j)^2$$

Where y_i is the i-th value out of the n data points in the mixture model curve we fit to the distribution and b_j is the value corresponding to the histogram bin where the i-th x-value (i.e. the value that the likelihood score y_i is based off of), lies. In other words, ALS essentially applies a sum of least squares to see how much the fitted curve differs from the actual histogram bars.

The second metric, which we titled "area under difference" (AUD) is as follows:

$$AUD = AU_{\text{curve}} - AU_{\text{histogram}}$$

where

$$\mathrm{AU} = \sum_{i=1}^{n-1} y_i (x_{i+1} - x_i)$$

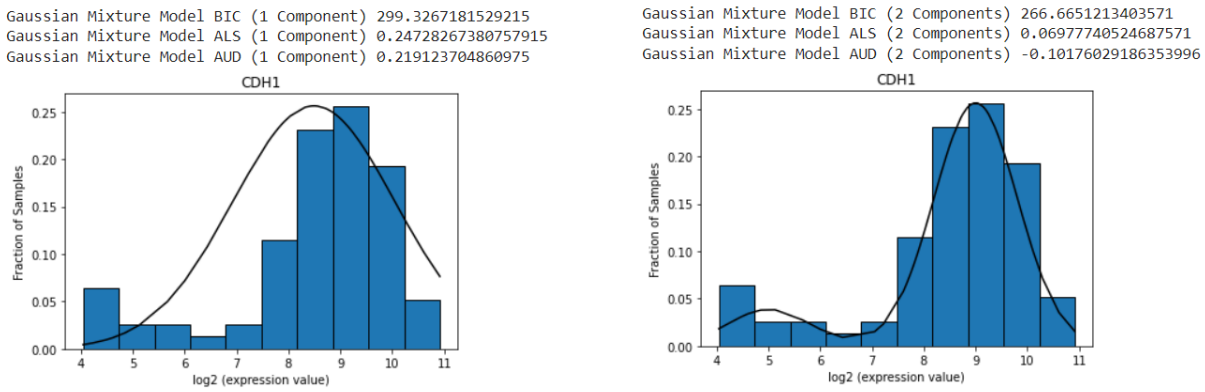
Here, x_i is the i-th value in the gene expression distribution and y_i is the i-th value in the curve fitted to the distribution (or corresponding to the i-th bin in the histogram). In short, AUD finds the difference between the area under the histogram and the area under the curve (estimated using a rectangular area approximation).

3. Results

3.1 Existing Packages

We found that GMMchi contained several errors in its codebase, and even after multiple alterations to the existing code, we were unable to run GMMchi on our data. Thus, we moved on to fitting the distributions using the three methods we had brainstormed; these consisted of applying 2 existing packages as well as a third method of our own creation extending the first two packages.

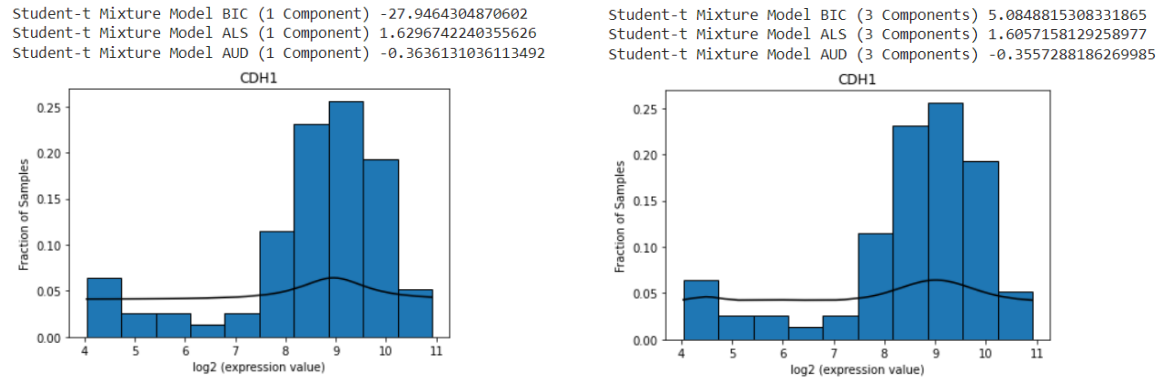
Figure 2 ("Intra-Model" Gaussian Test): This figure shows us the fit of a 1 component v. 2 component Gaussian mixture model on a selected gene. As shown, the distribution appears to be bimodal, which is confirmed by the fact that the 2 component mixture model has a lower BIC, ALS, and abs(AUD).

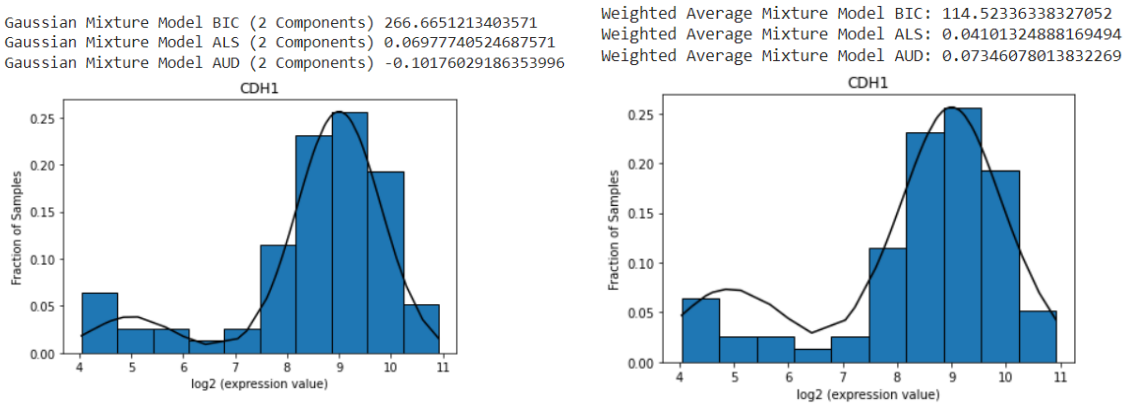


In general, Gaussian mixture models seemed to work well in identifying and fitting the data. Across the entire dataset, the best fit Gaussian mixture model for each gene had an average BIC of 75.4.

Next, we tested the student-t and weighted average models. However, these other types of mixture models tended to show more inconsistent results.

Figure 3 ("Inter Model" Test): The "inter model" test across methods, as well as an additional intra-model test for student-t models, yielded questions surrounding the validity of the BIC measure as a best fit. First off, the 3 component student t model seems to be a better fit than the 1 component student t, as the 3 component model accounts for the non-normal tail; however, the 3 component student t has a noticeably higher BIC (5) than the 1-component student t (-27). A similar pattern exists for the "inter model test". We see that even though the Gaussian and weighted average models visually appear to be better fits than the student-t model, their BIC's are much larger. This shows, contrary to what was expected, that a lower BIC does not always truly represent the best fit. By comparison, our own metrics (ALS and AUD) were generally lower for the better-fitting models, meaning they may be more effective differentiators.



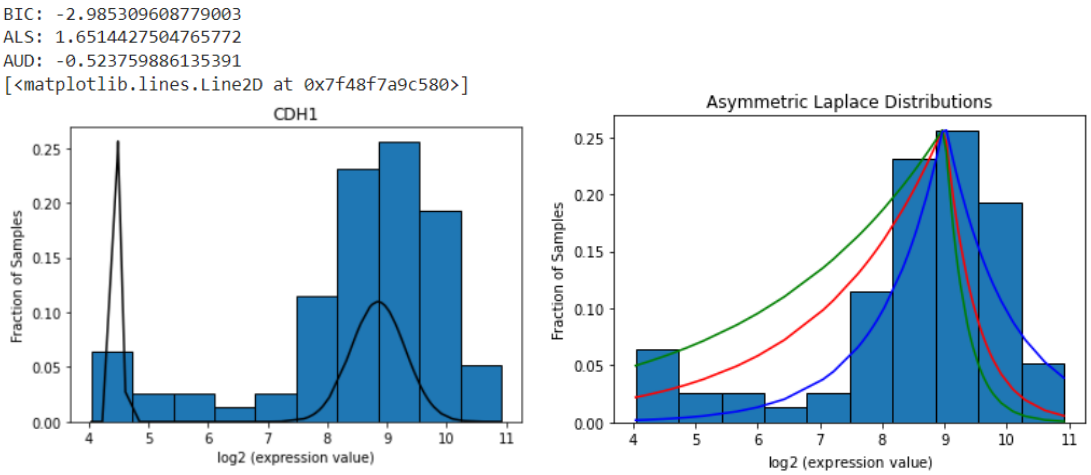


3.2 Our Implementation

We found that our own implementation of Gaussian mixture models was also generally able to detect non-normal shapes. However, our implementation was highly dependent on model initialization; even a small change in the initial mean and variance of the distributions we were using could lead to significant differences in the eventual fit. Additionally, we found that our own implementations were not able to account for the difference in peaks between both modes, as shown by visual evidence (see Figure 4) as well as the high ALS and AUD. While the blended implementation was better at accounting for peaks than the Gaussian implementation (supporting the observation that partially using a student-t mixture to model the distributions could better account for non-normal tails), it was still not as good as the existing packages at fitting the distribution.

In regards to our own implementation of SAL, we were able to successfully implement the initialization of multiple SAL distributions, with these initializations forming a fairly accurate first fit. Moreover, we found that the SAL EM algorithm was able to improve in performance with additional iterations. However, the dimensionality of the observed data began posing an issue in our implementation; specifically, taking the matrix inverse with numpy.linalg.inv() turned out to be a bottleneck point for our algorithm. Usually, the operation takes $O(n^3)$ time for most basic implementations, and at best gets improved to around $O(n^{2.37})$. Further testing is needed to improve runtime complexity and rectify this discrepancy.

Figure 4 (Our implementation of Gaussian mixture models (left) and SAL mixture models (right)):



3.3 Conclusions

Ultimately, we concluded the best method to fit the distributions was by using the weighted average of the existing Gaussian and student-t mixture model implementations, as the weighted average method was able to most effectively account for both normal and non-normal distributions. Since the weights were determined through our own novel procedure, this combined the best of existing methods as well as our own research.

4. Discussion

Our research's primary challenge resulted when implementing our own mixture models, mainly concerning the difficulty of initializing our parameters, which is something the existing packages automatically did. We found that if we did not initialize our parameters to a hyper-specific range, our model would converge too fast and result in misshapen distributions; we were ultimately not able to determine this range automatically, and had to hard-code it for each new distribution we modeled. Additionally, translating complex equations into functional code proved especially challenging when working with the Shifted Asymmetric Laplace distribution.

For next steps, we seek to continue to improve our mixture model implementation, including adding automatic initialization of parameters, rectifying the EM algorithm runtime issue in the SAL implementation, and improving performance as compared to the existing packages. We would also like to add additional distribution mixture models (e.g. Noncentralized Beta Distribution), especially those that are not currently implemented elsewhere (similar to our novel implementation of Shifted Asymmetric Laplace). Another long-term future direction can also be generalizing this to multidimensional mixture models, of which few implementations currently exist.

Additionally, beyond just working to find the best fitting distribution to the gene expression data, we would like to actually test the effects of having a better-fitting distribution and demonstrate how it is desirable. Our assumption is that a more optimal fit is ideal, as noted in several papers, "We showed how incorporating the shape of the expression distribution provided a means to identify genes with prognostic value [...]" (Torrente et al.). However, we do not have conclusive results confirming this assumption is correct. If possible, we would like to prove the validity of this assumption and show the benefits of having an optimized fit. One way we could do this is by applying a supervised model on top of our unsupervised fit. For example, with a dataset of cancer gene expressions, we could employ a method similar to the Torrente et. al paper where we would determine the shape of the gene expression distributions and then apply a classifier model to them to identify prognostic marker genes.

Appendix

Github Link (to project code): shogho0/ml4fgfinalproject (github.com)

References

Franczak BC., Browne RP., McNicholas PD.. "Mixtures of Shifted Asymmetric Laplace Distributions." IEEE Trans Pattern Anal Mach Intell, 36(6):1149-57, 2014. https://doi.org/10.1109/TPAMI.2013.216.

Liu, Ta-Chun & Kalugin, Peter & Wilding, Jennifer & Bodmer, Walter. "GMMchi: gene expression clustering using Gaussian mixture modeling." *BMC Bioinformatics* 23, 2022. https://doi.org/10.1186/s12859-022-05006-0.

de Torrenté, L., Zimmerman, S., Suzuki, M. *et al.* "The shape of gene expression distributions matter: how incorporating distribution shape improves the interpretation of cancer transcriptomic data." *BMC Bioinformatics* 21 (Suppl 21), 562, 2020. https://doi.org/10.1186/s12859-020-03892-w

Miin-Shen Yang, Chien-Yo Lai, Chih-Ying Lin. "A robust EM clustering algorithm for Gaussian mixture models." *Pattern Recognition*, Volume 45, Issue 11, Pages 3950-3961, ISSN 0031-3203, 2012. https://doi.org/10.1016/j.patcog.2012.04.031.

S. J. Roberts, D. Husmeier, I. Rezek and W. Penny. "Bayesian approaches to Gaussian mixture modeling." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1133-1142, 1998. https://doi.org/10.1109/34.730550.

Pedregosa *et al.* "Scikit-learn: Machine Learning in Python." JMLR 12, pp. 2825-2830, 2011.