Characterizing Densities from Semi-Analytical Models of Galaxy Formation

Alex Ji
Statistics
Email: alexji@stanford.edu

Amir Kavousian
Civil Engineering
Email: amirk@stanford.edu

Steve Lesser
Computer Science
Email: sklesser@stanford.edu

I. Introduction

The naive way to model galaxy formation is a computationally intensive process involving large N-body simulations of hydrogen atoms interacting in a very complex manner. This is not feasible for modeling a large number of galaxies in a variety of environments, which is needed to compare theoretical models of galaxy formation to empirical observations from galaxy surveys and to discern which physics processes affect what galaxy properties. Astrophysicsts have dealt with these issues by developing semi-analytical models (SAMs) where the important stages of galaxy formation are filled in with parameterized analytic "recipes" describing the underlying physics. The SAM takes input parameters and outputs a population of galaxies. To summarize a population of galaxies, astrophysicsts usually do something similar to histogramming the various galaxy properties. For example, one can histogram the log-mass of a population of galaxies, which is called the "stellar mass function". The height of each bin is then a summary statistic of the galaxy population. (There are standardized ways of normalizing these counts in a given bin width.)

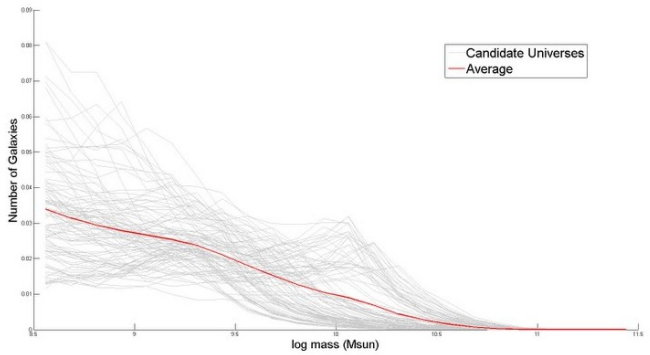

Fig. 1. Various stellar mass functions from the same semi-analytical model showing the variance of the SMF despite all corresponding to the same luminosity function.

One can use a procedure like the Metropolis algorithm to find the parameters that fit a particular histogram, also called a data set. Once we fit parameters to a particular data set, we are interested in the predictions of the SAM for another data set. We can take the posterior distribution of SAM parameters from fitting to the first data set and sample from this posterior distribution, running the parameters through the SAM to create a set of possible universes, each with their own stellar mass function.

In this project, we have taken a sample of 1000 $z = 1$ stellar mass functions, each with 24 log mass bins, and each predicting an identical $z = 0$ K-band luminosity function. These stellar mass functions populate a 24-dimensional space, where each dimension is the height of the stellar mass function in a given mass bin. Our goal is to model the density of this distribution. Once we have a density, we can calculate the probability of the true universe occuring in this model. The ability to validate a SAM in different data sets is extremely valuable, as it can further constrain SAM parameters and suggest where a SAM incorrectly models galaxy formation physics.

II. Dimensionality Reduction

The Stellar Mass Function (SMF) data is highly correlated in some dimensions (see figure 2). To reduce the dimensionality of the model and to enable better visualization of the data, we performed Principal Components Analysis (PCA).

The results of our PCA show that using only the first three principal components, we can explain 90% of the variance in SMF data (see figure 3).

A. Details of Principal Components Analysis

The calculation is done by a singular value decomposition (SVD) of the centered and scaled data matrix. Compared with using eigenvalues of the covariance matrix, the SVD method generally offers more numerical stability and accuracy. The SVD of the $N \times p$ matrix X is given by:

$$X = UDV^T$$

Where U and V are $N \times p$ and $p \times p$ orthogonal matrices with the columns of U spanning the column space of X, and the columns of V spanning the row space. D is a $p \times p$ diagonal matrix, with diagonal entries $d_1 \geq d_2 \geq ... \geq d_2 \geq 0$ singular values of X. The eigenvectors v_j (columns of V) are principal component directions of X. The first principal component direction v_1 has the property that $Z_1 = Xv_1$ has the largest sample variance amongst all normalized linear combinations of the columns of X. The subsequent principal components are calcualted in the same way [5]. We used the first three principal components to fit our models as three components

Fig. 2. Histograms of Stellar Mass Density data; and correlations between different density distributions.

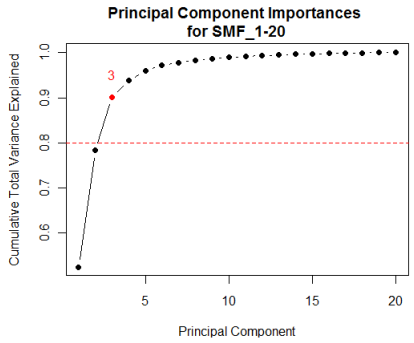

Fig. 3. PCA results on SMF data: the eigenvalues drop quickly as more principal components are added; as a result, using only the first three principal components, we are able to explain 90% of the variability in SMF data.

explained 90% of the variance in our data. Note, this assumes our observed stellar mass function has variability similar to predictions from the SAM.

III. Distribution Fitting Methods

Our goal is to parameterize the probability density of our Semi Analytical Model. Since the distribution has unclear structure we used unsupervised learning for density estimation.

A. K-Means Clustering

One unsupervised method we investigated was k-means clustering. We did not know the number of clusters we should use so we tried many different cluster sizes and evaluated their effectiveness using the p-value of chi-squared independence test between training data and test data. To perform k-means clustering we treated each stellar mass function as a single 3-dimensional point using PCA components. We then performed standard k-means clustering on a subset of all of our available generated points. We tried various ratios of splitting up our data into training and test sets and settled on a random 70% of our points being used for training and the remaining 30% used for testing. To predict density values for the test set we clustered the test points using the trained clusters and let the density be proportional to the number of points in a cluster. Once clusters were found we planned to fit continuous distributions to each individual cluster.

We were interested in whether our k-means clusters were stable predictors of the probability density function. To determine this we used a chi-squared test on the ratio of the size of each cluster to the total size of the either the training set or the test set. Let k be the number of clusters, M be the test set size, N be the training set size, m_i be the fraction of test points found in cluster i, and n_i be the fraction of training points found in cluster i

$$X^2 = \sum_{i=1}^{k} \frac{(Nm_i - Mn_i)^2}{n_i NM}$$

We then took the P-value of our chi-squared value as a performance metric of the k-means as a good fit or bad fit where the P-value is defined as

$$P = 1 - P(X^2 <= X^2_{k-1})$$

For each candidate cluster size k we ran k-means for 100-1000 iterations of random separations of training and test data. We then histogramed the resultant P-values of the chi-squared performance metric as seen in III-A. We can see there is a wide variance in most of the p-value distributions implying that the k-means clustering is highly dependent on which points are split into the training set versus the test set. We can see when k is 14 the p-values reach essentially their maximum average and have relatively few iterations with P-values less than 0.4.

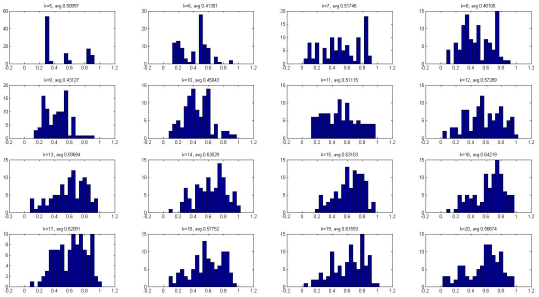

Fig. 4. Histograms of the chi-squared P-values for repeatedly running k-means clustering with different numbers of clusters. Cluster size starts at 5 on the top left and increases to the right continuing again at the beginning of each row. Due to the high variance in p-values we determined k-means was not an acceptable method for density estimation.

However, even with the best k-values we think k-means is not stable enough to definitely show a strong and stable probability density model hence we do not consider it to be appropiate as a final modeling tool.

B. Kernel Density Estimation

In order to have a definitely accurate distribution we implemented kernel density estimation. However, this method is slow and inefficient due to requiring the complete training dataset to produce a single density value as opposed to requiring a smaller number of parameters such as in mixture of gaussians. It also generally overfits the density.

Kernel density estimation places a small kernel, in our case a multivariate gaussian, on each data point and then computes an average probability based on existing data and a bandwidth matrix H. Let $\{x_1, x_2, ..., x_n\}$ be an independent random sample drawn from $f(x)$. The general form of the kernel estimator of $f(x)$ is

$$f_H(x) = \frac{1}{n}\sum_{i=1}^{n} K_H(x - x_i)$$

where $K_H(x) = |H|^{\frac{-1}{2}} K(H^{\frac{-1}{2}}x)$, $K(x)$ is a multivariate gaussian function, and H is a symmetric positive definite $d \times d$ dimensional matrix known as the bandwidth matrix. After principal component analysis, the dimensionality of our data is 3. We used a diagonal bandwidth matrix and computed the values on the diagonal based on the assumption that data was observed approximately sampled from a multivariate normal density. This is a strong assumption our data does not satisfy, but it still gives a reasonable result. The ith diagonal of H is calculated described by [**?**]zhang04) as

$$h_i = \sigma_i \frac{4}{(d+2)n}^{1/(d+4)}$$

where σ_i is the standard deviation of the ith variate and d is the dimensionality of the data. The probability density can be seen in the level contours of figure 7.

C. Mixture of Gaussians

As our third density estimation method, we decided to use mixture of gaussians (abbreviated GM). We used the first 3 principle components to fit this model. We used the MATLAB function `gmdistribution.fit` to fit the mixture of gaussians, which gives us a density function. To choose k, we decided to compare the log-likelihoods on a hold-out test set (30% of the data). In figure 5, we plot the log likelihood of the test set and the training set. We determine the number of clusters by finding an elbow in the likelihood. In our data, this occurs at $k = 7$.

One might expect that when you have reached an optimal k, adding an additional gaussian component will not significantly change the resulting distribution. To test this hypothesis, we needed a way to compare two different distributions. We decided to use Kullback-Leibler divergence (KL divergence). The KL divergence between two distributions $P(X)$ and $Q(X)$ is defined as follows:

$$KL(P \parallel Q) = \int P(x_1, x_2, x_3) log \frac{P(x_1, x_2, x_3)}{Q(x_1, x_2, x_3)} dx_1 dx_2 dx_3$$

Numerically calculating the three-dimensional integral with MATLAB `quad` proved incredibly time consuming, so we decided to use Monte Carlo integration with 10000 points and report the error. This method introduces some error because a uniform sample of a distribution with many features may not adequately sample all the features, however it makes the numerical calculation tractable.

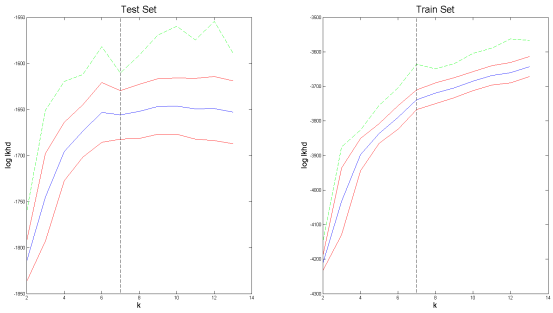

Fig. 5. Left: log-likelihood of test set. Right: log-likelihood of training set. Blue line represents the mean log likelihood over 100 runs of mixture of gaussians. Red lines represent +/- one standard deviation from the mean. Green line represents maximum log likelihood. Note that at $k = 7$ (vertical dotted line), the log likelihood for the test set begins to flatten, so we choose $k = 7$ for our model.

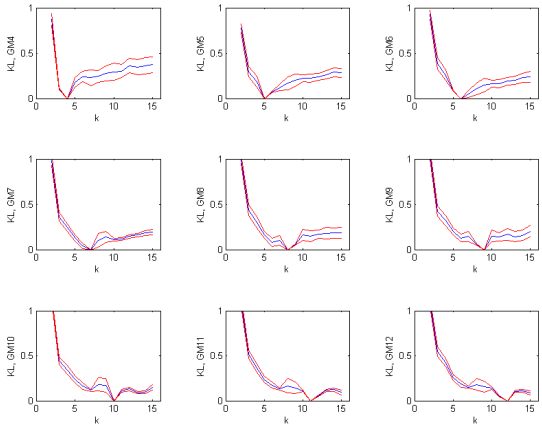

Fig. 6. The results of the KL divergence to compare GMs with different number of clusters. Each plot shows the results of the KL divergence of one specific k GM model (indicated on the y-axis) against models with $k = 1, 2, 3, ..., 15$ clusters. As the plots show, the KL divergence bottoms out around $k = 7$ and stays that way for higher k. This means increasing k does not significantly change the probability distribution and confirms $k = 7$ is a good number of clusters for our GM.

We computed GM models for $k = 2$ to $k = 15$ and calculated the KL divergence between all models, plotted in figure 6. Note that when the GM has $k \geq 7$, the KL divergence is flat for $k \geq 7$, suggesting that the probability distribution does not change significantly after a 7-component model. The

probability density can be seen in the level contours of figure 7.

D. Dirichlet Process

Using the Dirichlet process as a prior for the number and relative weighting of gaussians in a gaussian mixture model allows us to have an arbitrary number of gaussians in our prior for the distribution we want to estimate. This allows us to avoid the process of tuning the number of clusters [3]. Markov Chain Monte Carlo (MCMC) methods can be used to find this Dirichlet mixture model. We used algorithm 7 from Neal's paper [4] with prior distributions for new gaussians set according to [2], and we quickly describe the algorithm here.

The state of the Markov chain is determined by a cluster assignment for each data point and the mean and covariance matrix parameters for all gaussians containing a data point. To step the Markov chain, the algorithm essentially does two things. First, it redistributes the points into different clusters using Metropolis-like probabilities. With some probability, points can be put into a randomly initialized new cluster. Second, we use a maximum-likelihood estimate to update the mean and covariance matrix of each cluster.

We encountered an underspecified part of these Markov chain algorithms. The Mixture of Gaussians algorithm does fuzzy clustering, so it gives the probability of each data point being in a given gaussian. However, all MCMC algorithms we found for the Dirichlet process gaussian mixture model (DPGMM) used hard clustering, since the state of the Markov chain is determined by the cluster label of each point. When a cluster has less than $p = 3$ (number of dimensions) points in it, then the maximum likelihood covariance estimate is singular. To sidestep this problem, we simply did not reestimate the covariance matrix when there were too few points in the cluster.

Initializing the Markov chain with k = 7 and a distribution given by a gaussian mixture fit, we ran the MCMC while checking whether the distributions given by the last three Markov states were significantly different as determined by KL divergence. However our results showed that our Markov chain converged whenever the Monte Carlo integration error was large, rendering the results unreliable. Given the time constraints, we decided to run the Monte Carlo simulation for 100 times and fit our model based on that. The probability density can be seen in the level contours of figure 7.

IV. Conclusion and Next steps

We have done a survey of useful methods for density estimation. We investigated a simple k-means model, a kernel density estimation model, a Gaussian mixture model, and a Dirichlet process mixture model.

For this particular data, we found that a mixture of Gaussians model with $k = 7$ works fairly well. Slightly increasing k does not significantly change the distribution, as seen by KL divergence. Additionally, the Dirichlet process Gaussian mixture model provides an automatic way of determining k. We are fairly confident that our Gaussian mixture with $k = 7$ accurately estimates the density, and we can compute values from the density very quickly.

Now that we have a density, the next step is to consider the observed stellar mass function from our universe to this probability density. We can find the magnitude of the density for the observed universe, and in future work we will find a way to interpret that number to constrain the underlying semi-analytical model.

Thank you to Richard Socher for his advice on exploring Dirichlet processes for density estimation, Dr. Yu Lu for providing the data and useful discussions, and the rest of the CS 229 course staff for their general advice and encouragement.

References

[1] X. Zhang, M. King, R. Hyndman. *Bandwidth Selection for Multivariate Kernel Density Estimation using MCMC*, 2004.

[2] C. Rasmussen. *The Infinite Gaussian Mixture Model*, Advances in Neural Information Processing Systems 12, MIT Press, 2000.

[3] Y. Teh. *Dirichlet Process*, Encyclopedia of Machine Learning, Springer, 2010.

[4] R. Neal. *Markov Chain Sampling Methods for Dirichlet Process Mixture Models*, Technical Report No. 9815, Department of Statistics, University of Toronto, 1998.

[5] T. Hastie, R. Tibshirani, Robert. *The Elements of Statistical Learning*, Springer, 2011.

[6] Y. Lu, H. Mo, M. Weinberg, N. Katz. *A Bayesian approach to the semi-analytic model of galaxy formation: methodology*, Monthly Notices of the Royal Astronomical Society 416, 2011.

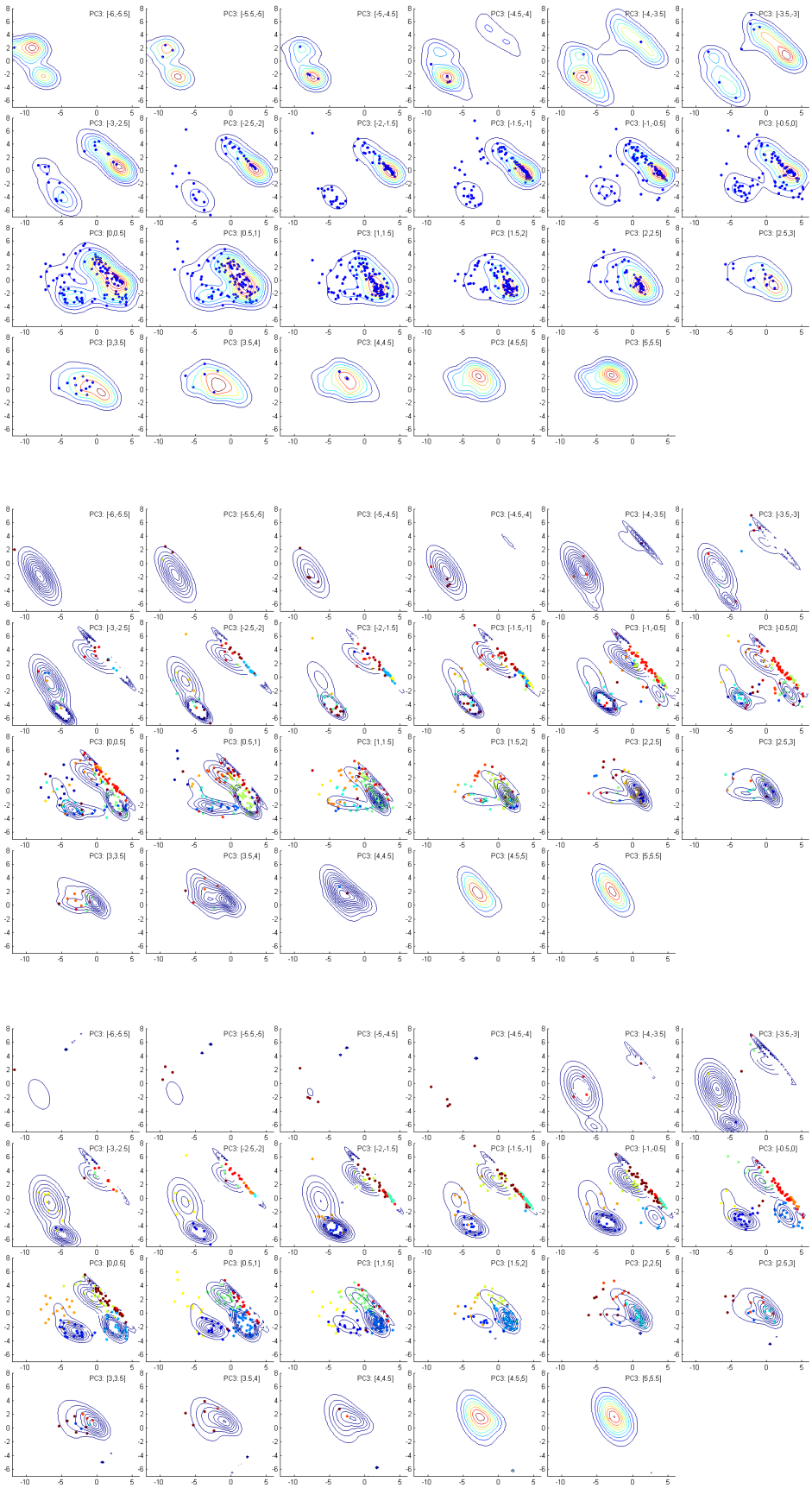

Fig. 7. Level contours of variuos density functions including from top to bottom: kernel density estimation, mixture of Gaussians with $k = 7$, and Dirichlet process. Point colors represent different Gaussian components. Note the similarity to kernel density estimation with both mixture of Gaussians and Dirichlet process despite them containing many fewer parameters and being much more efficient.