# Final Project: Structure in The APOGEE/Cannon Stars

Nick Konz, Bowen Gu, Roark Habegger

April 28, 2020

## 1 The APOGEE/Cannon Data Set

The raw APOGEE dataset is a set of spectra for over 263,000 stars in the Milky Way galaxy. Using this, SDSS provides intermediate datasets with stellar parameters and chemical abundances for all the observed stars. The spectra are averaged over multiple 'visits' (observations) to the star, so the processed data only provide one parameter set for each star. While these base abundances are derived using the APOGEE Stellar Parameters and Chemical Abundances Pipeline (ASPCAP), additional machine learning using the Cannon algorithm has produced more accurate abundances for many of the stars surveyed.

Ideally, we hoped to find some clustering and structure in the 20-dimension abundance space. This could correspond to physical clustering since stars with different birthplaces should have different chemical abundances. When this proved difficult and we were not finding any statistically-justified clustering, we got the radial velocities for each star in the dataset run through the Cannon algorithm. These velocites came from the original APOGEE dataset produced by the ASPCAP. These radial velocities add a significant amount of variation to the data set, but not enough to make a cluster number higher than 2 justified for the 21-dimensional samplings. To see how we parsed the Cannon data and got each star's radial velocity from the ASPCAP data, see the CreateCSV.ipynb file which also provides a plot of RA and DEC, illustrating the field of view for the Cannon dataset. These images also appear on the first page of the Figures PDF file.

## 2 K-means

$k$-means clustering[1] is a method of vector quantization, originally from signal processing, that aims to partition $n$ observations into $k$ clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. It is popular for cluster analysis in data mining.

We approach the problem via a specific instance of this $k$-mean unsupervised learning algorithm. To know how many clusters we are looking for, we split the data into two sections, the training set and the validation set. The results output by the training set are the predicted labels and the result output by the validation set are the true labels. Then we use cross validation to obtain accuracy scores using the Scikit-Learn tool box's built-in functions for different cluster numbers we input to the $k$-means algorithm. After that, we plotted the accuracy score as a function of cluster numbers and used the peak of the graph as the optimal number of clusters. Specifically, we used three different methods to partition the training set and the validation set.

**Note: For all "Fig"s mentioned in section 2.1 to 2.3, please refer to the corresponding image in "cannon_project_final.ipynb"**

### 2.1 Method I

In method I, we focused on all 20 different elements' chemical abundance of each star. We used 19(all except Fe) elements' chemical abundance as the training set and the rest 1(Fe) element's chemical abundance as the validation set. We started looking for 2 clusters and varied the number of clusters from 1 to 100 to plot the relation between accuracy score and number of clusters. The result is shown in Fig1.

[1] `https://en.wikipedia.org/wiki/K-means_clustering`

The peak of Fig1 appears when the cluster number is 5. We noticed that the plot's peak has a relatively high($> 0.7$) accuracy score, meaning 5 may be the optimal cluster number. As a result, we took a further look at how the members of these 5 groups distributed in physical space. After plotting the distribution of each cluster members in physical space, we got Fig2.

From Fig2, we can hardly see any physical boundaries. To take a closer look at it, we then plot the distribution of each group in five different figures and got Fig3 to Fig7.

Together with Fig2, Fig3 to Fig7 show that members of each group all mix with each other and there are no obvious physical boundaries between each group.

## 2.2 Method II

In method II, we still focused on all 20 different elements' chemical abundance of each star, but together with the RA and DEC information. This time we randomly picked 10 of the 20 elements(together with RA and DEC) as the training set and picked the other 10 elements(together with RA and DEC) as the validation set. For each combination of randomly picked training set and validation set, we look for different cluster number and plot the relation between accuracy score and number of clusters. Finally, we compared the peaks of the plots for each combination to decide the optimal number of clusters.

First, we looked for cluster number from 2 to 10. We tested a total of 552 randomly picked, non-repetitive different combinations of training and validation set. To determine the optimal number of clusters suggested by the algorithm, we plotted a histogram that counts the number of peaks appeared for each possible cluster number. We also plotted a histogram to show the distribution of the accuracy scores of the 552 tests. The results are shown in Fig8 and Fig9, respectively.

Also, a typical output of the algorithm for a certain combination of training and validation set is shown in Fig10.

According to Fig9 and Fig10, although most of the returned accuracy scores were relatively high($> 0.6$), most of them actually correspond to the 2-cluster condition(eg. Fig10). Also, from the histogram, the algorithm preferred 2 clusters about 60% of the time. Indeed, there were cases where the algorithm favored a large number of clusters(Fig11). However, these cases were few, and the accuracy scores were relatively low($< 0.5$) for each case.

Second, we looked for cluster number from 21 to 30. This time, we tested a total of 173 randomly picked, non-repetitive different combinations of training and validation set. We also plotted a histogram that counts the number of peaks for each possible cluster number and a histogram that shows the distribution of accuracy scores. The results are shown in Fig12 and Fig13, respectively.

Also, a typical output of the algorithm for a certain combination of training and validation set is shown in Fig14.

From Fig12, we noticed that unlike the case of cluster number from 2 to 10, here the distribution is almost uniform. We also noticed that Fig13 implies that the overall accuracy score of the tests is much lower than the 2 to 10 cluster case suggests (Fig9). Also, judging from Fig 14, the peak is almost the same height as many other points, meaning it is not necessarily a better choice than the other cluster numbers, not to mention that all the accuracy scores are really low ($< 0.1$).

## 2.3 Method III

In method III, we added the radial velocity information of each star, which was obtained from the Gaia data. This time we decided to split the chemical abundance information and the physical space information to see whether they agree with each other. To achieve this goal, we used all 20 different elements' chemical abundance as the training set and the RA, DEC, and radial velocity information as the validation set. As for the cluster number, we still looked for 21 clusters and increased the number of clusters to 30 to plot the relation between accuracy score and number of clusters. The result is shown in Fig15.

Fig15 shows extremely low accuracy scores for all cluster numbers($< 0.06$). This means that even if the stars can be partition into 21 to 30 clusters, their chemical abundance data are too consistent and do not vary much, making it almost impossible to find any physical boundaries from clusters in abundance space.

# 3 Gaussian Mixture Modelling and UMAP

## 3.1 Gaussian Mixture Models

A Gaussian Mixture Model, or GMM, is an unsupervised learning technique that aims to determine clustering within unlabeled datasets. Nick's portion of the project was to use GMMs and UMAP (a method to be discussed shortly) to try to find clustering within the Cannon dataset. Given some $d-$dimensional dataset $\{\boldsymbol{x}_i\}$ with $N$ datapoints, the basic idea behind GMMs is to fit a mixture of some $K$ $d-$dimensional Gaussian distributions to the dataset. Specifically, this corresponds to maximizing a likelihood function of the form $\mathcal{L}\left(\{\boldsymbol{x}_i\} \,|\, \{w_k\}, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}\right) = \sum_{k=1}^{K} w_k \mathcal{N}\left(\{\boldsymbol{x}_i\} \,|\, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}\right)$, where the parameters being fit are $\{w_k\}$, the set of weights $w_k \in \mathbb{R}$ (corresponding to the amount of the dataset found within the $k^{\text{th}}$ cluster) for each of the $K$ Gaussians denoted by $\mathcal{N}$; $\{\boldsymbol{\mu}_k\}$, the set of means $\boldsymbol{\mu}_k \in \mathbb{R}^d$ of the Gaussians; and $\{\boldsymbol{\Sigma}_k\}$, the set of covariance matrices $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$ of the Gaussians [2]. In practice (as in the case of `sklearn.mixture.GaussianMixture`, this fitting is usually done with the EM, or Expectation Maximization algorithm.

One improvement of GMMs over other clustering methods, e.g. $K$-means, is that due to GMMs having explicit likelihood functions, we can define a metric that compares the relative fitness of models with different values for $K$, the number of clusters. One method that we use is the Bayesian Information Criterion, or BIC, which is a log-likelihood function that is *penalized* by the number of model parameters (as well as the number of datapoints). Specifically, the BIC is defined as $\text{BIC} = \log\mathcal{L} - \frac{M}{2}\log N$, where $M$ is the total number of free model parameters [2]. We also use the Akaike information criterion, or AIC, which is defined similarly (but from a frequentist, rather than Bayesian approach). In practice, we use the AIC and BIC to determine the lowest $K$ where a nontrivial/noticeable local maximum of the criteria appears. Then we take this to be the optimum $K$.

### 3.1.1 GMM Results

We used GMMs to attack this dataset in a number of ways. To begin, we wanted to examine all possible correlations and clusterings between individual features of the dataset. To do so, for each pair of features—e.g. Calcium abundance `CA_H` vs. Nitrogen abundance `N_H`, which is the example that we show in Figure 3.1.1 of the notebook—we ran GMMs with $K = 1, 2 \ldots, 9$ on the two-dimensional dataset of each pair of features, and plotted the BIC and AIC of these GMMs vs. $K$ (Fig. 3.1.1). We then used the method described in the previous paragraph to determine the optimum $K$ from this plot, and ran and plotted a GMM with that $K$ on the dataset, including a negative log-likelihood contour plot (Fig. 3.1.1). We ran this on all possible pairs and although we saw some correlations, mostly between chemical abundances, it was difficult to glean anything about the dataset as a whole. As such, while interesting, this analysis didn't provide any useful information.

The next analysis that we performed with GMMs was higher dimensional; we created three datasets from the master: $C$, the set of all abundance and equatorial coordinate data; $V$, the set of all abundance and velocity data, and $P = C \cup V$. We then performed a similar analysis as in the previous paragraph to determine potential clustering within these datasets (performing GMMs with $K = 1 \ldots 35$ on all three of them, and plotting BIC and AIC vs. $K$). The results are given in Fig. 3.1.2 of the notebook; we expected to determine some sort of abundance grouping within position and/or velocity space, but as shown, we didn't find any nontrivial local maximum in the BIC/AIC that would indicate clustering for any of the three datasets.

## 3.2 UMAP

UMAP[2] is a robust and easy-to-use dimensionality reduction method meant to help visualize clustering of high-dimensional datasets by embedding high-dimensional data into 2D space [3]. We only experimented with UMAP for our project, by attempting to apply GMMs to UMAP's outputs (with various parameters chosen for UMAP), in order to find potential clusters within the 2D embedding that could indicate higher dimensional clusters/classes or correlations.

[2] `https://umap-learn.readthedocs.io/en/latest/`

[3] The mathematical details of UMAP are beyond the scope of this writeup, so it is a "black box" tool from our perspective.

Shown in Fig. 3.2.1 of the notebook, we first attempted to run UMAP on the entire dataset, to try to find structure that would indicate high-dimensional groupings (using UMAP parameters that best supported the formation of large scale structure). As shown, however, we did not see any noticeable clusterings; most of the data was within one large structure, with only a few sparse outside shapes. Next, as shown in Fig. 3.2.2, we did the same, but with the datasets $C$, $V$ and $P$ of the previous section; unfortunately, this didn't give any useful information either, as no noticeable clustering appeared, even after running UMAP on a variety of parameters.

# 4 DBSCAN

The Density-Based Spatial Cluster Algorithm with Noise (DBSCAN) provided by Scikit-Learn is an unsupervised learning algorithm, where 'noise' is data points that do not belong to any cluster. DBSCAN can identify clusters while not forcing every data point to be a member of some cluster, particularly when it does not make sense for that point to be a part of the nearby cluster (large distance from core point).

The first step in performing the algorithm is to determine a proper $\varepsilon$ parameter. This parameter is a radius in the parameter space which marks the furthest apart two points can be and still be in the same cluster. Scikit Learn's demo of DBSCAN algorithm suggests doing a Nearest Neighbor algorithm for varying radii and finding the kink in the cluster size curve. Doing this for the abundances, we found a radius of 1.075 was the optimal limiting radius. This is shown in Fig1 and Fig2 in the DBSCAN portion of the notebook. This is a significant portion of the abundance space distributions, which are shown in plots against the overall metallicity of each star (Fig3,4,5,6 in DBSCAN section).

Using the $\varepsilon$ parameter determined by the nearest neighbors algorithm, the DBSCAN algorithm suggests a single cluster regardless of the amount of data withheld from the algorithm. In an attempt to find some clustering, we ran the DBSCAN algorithm again on the radial velocities of each sample in addition to the abundances, making a 21-dimension feature space. The algorithm returns the same result, a singular cluster, when the velocities are normalized to range from $-3$ to 3 (this was chosen so it would be similar in order of magnitude to the abundance values).

If the velocities are not normalized and left in $\mathrm{km\,s^{-1}}$, then the algorithm identifies seven clusters that clearly define the galactic regions rotating toward and away from the Earth. However, this seems to be relatively independent of the abundances, suggesting the algorithm is just identifying different clusters in terms of their radial velocity (see Fig7).

# 5 Final Conclusions

In this project, we tried to determine whether there are physical boundaries which can be determined by clustering in abundance space. However, we failed to get a convincing result. This is partly because we could not determine how many clusters we should find based on the chemical abundance data. Although we approached the problem using different clustering algorithms, the results from all three methods were dissatisfying (we consider UMAP and GMM as a singular algorithm). In fact, the algorithms seemed to provide a unanimous assertion that there is only a single cluster in the abundance space of the Cannon dataset. Due to the high, and relatively uniform, density of that cluster in abundance space, it is difficult for the algorithms to divide it into smaller clusters that might more accurately describe clustering in physical space.

For the complete picture of our work, we have GitHub repository nickk124/stellardatamining. It can be found at the following link: https://github.com/nickk124/stellardatamining

# References


[1] Cannon Data Web Page
`https://www.sdss.org/dr14/irspec/the-cannon/`

[2] Murphy, Kevin P (2012) *Machine Learning: a Probabilistic Perspective*, MIT press.