Comparison of Compositional Approaches to Taxonomic Binning for Metagenomics

Ryan Nelson
M.S. Computational Biology
Carnegie Mellon University
Pittsburgh, PA 15213
ryann@andrew.cmu.edu

Patrick Kimball
M.S. Computational Biology
Carnegie Mellon University
Pittsburgh, PA 15213
pkimbal2@andrew.cmu.edu

May 13, 2021

Abstract

Taxonomic binning is a technique for making sense of metagenomics data. Supervised machine learning classification represents one compositional approach to taxonomic binning. Here we shown a modification to existing methods for generating fragment datasets from metagenomics data. We also compare generative and discriminative classification approaches on two small datasets and find no real difference in performance between classification strategies.

1 Introduction

1.1 Problem

Metagenomics is the analysis of the genetic sequences discovered through broad-stroke sampling of an environment. One of the goals of metagenomics is to be able to reconstruct the full genome of the organisms found during sampling. In order to accomplish this, collected samples must first be classified into clades, or groups, based on taxonomy. This process is known as taxonomic binning, and the goal is to do this as efficiently and accurately as possible. Compositional approaches to taxonomic binning leverage machine learning to classify genetic material based on the sequences they contain. Common supervised techniques like Naïve Bayes and SVM have been implemented successfully for taxonomic binning[1], and many advanced techniques have been used as well.

One of the computational challenges associated with these approaches is handling the dimensions of the dataset. With metagenomics data, the length of sequences in a single dataset can range over multiple orders of magnitude, making it difficult to encode all samples in the same way for processing. The current "sampling" strategy for standardizing sequences to a specific length produces a massive number of samples in order to achieve sufficient coverage of the original sequence. Encoding these samples exponentially increases the dimension of each sample. This process makes infeasible any techniques which cannot handle high-dimensional, high-volume data.

1.2 Motivation

One question which should be considered for metagenomics is whether generative or discriminative models are better for taxonomic binning. Each model type has advantages and disadvantages, and the direction of future method development could be guided toward one direction or the other if a significant difference in performance was found. The other motivation for this work is to develop an understanding of the process needed to prepare metagenomics data for use in supervised machine learning.

1.3 Method Overview

This project compares the performance of generative and discriminative supervised machine learning classification techniques on a sample metagenomics dataset. Techniques representative of both categories are used to classify species based on sequence. Following Vervier et al. (2015), performance is assessed using average recall to reduce bias introduced through class imbalance.

The discriminative models chosen for comparison are logistic regression, SVM, and random forest classifier. Logistic regression and SVM represents a style of classification based on a linear hyperplane. Random forest represents a style of classification which also considers nonlinear relationships in the data. The generative model chosen for comparison is Naïve Bayes classifier. Logistic regression and Naïve Bayes are implemented from scratch for the project. `scikit-learn` is used for the other models as well as for advanced logistic regression approaches involving L1 and L2 penalties and class weights.

1.4 Dataset

The datasets used in this work are derived from the small training dataset constructed by Vervier et al. for the paper, "Large-scale machine learning for metagenomics sequence classification"[1]. The original dataset "train_small-db.fasta" is found at `http://cbio.ensmp.fr/largescalemetagenomics`. It consists of 1564 nucleotide sequences from 193 species in .fasta format along with matching taxonomy data. Sequence length ranges from 1308 bp to 9.8 million bp. Due to hardware limitations, two smaller datasets were generated from this original dataset to be used for this work. One dataset, known hereafter as the 2000-lengths dataset, selects all sequences of length 2000 or shorter. It consists of 5 sequences and 5 species. The second dataset, known hereafter as the 3000-lengths datasets, selects all sequences of length 3000 or shorter. It consists of 20 sequences and 12 species.

2 Methods

2.1 Classification Dataset Preparation

The sequences that make up metagenomics data do not have a standardized length. Supervised machine learning requires that all records have the same dimension, so a process must be created to construct a standard-length dataset while mitigating the information lost versus having the full sequence available. This work developed two processes for dataset preparation, both closely following the approach used by Vervier et al. (2015). The general strategy of both for both is illustrated in Figure 1. There are two main steps in each approach: fragment generation and fragment encoding.

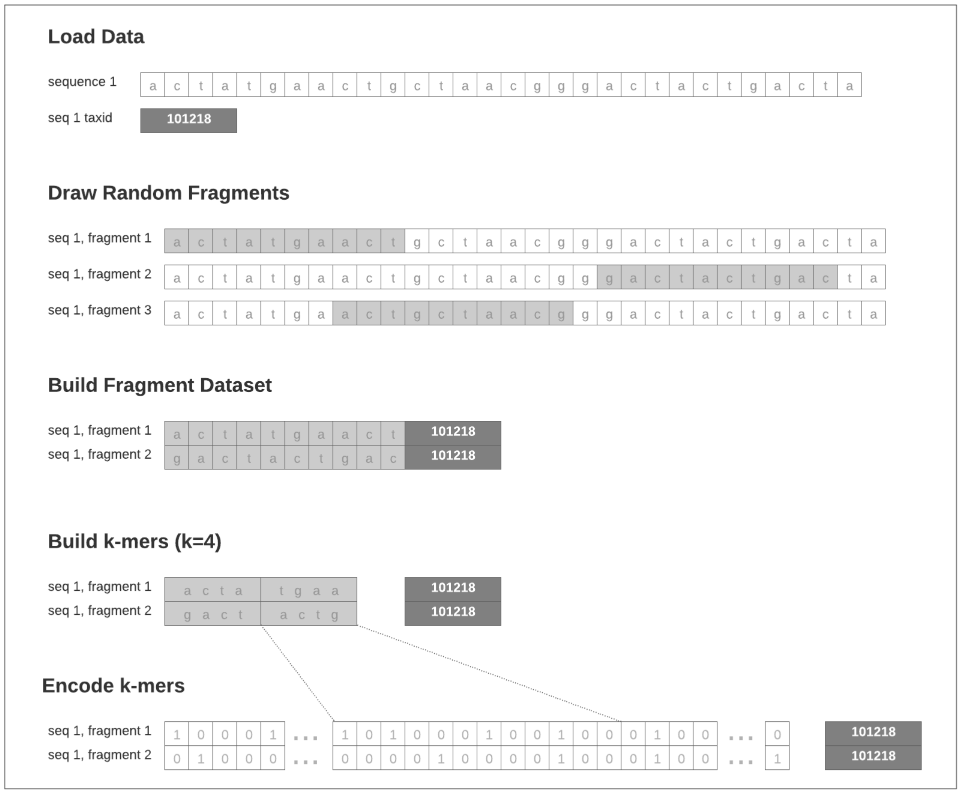

Figure 1: Dataset Preparation Steps

2.1.1 Fragment generation

In fragment generation, a number of fragments of a predetermined length are selected at random for each sequence. The predetermined length, also known as the sample length, defines the length of each fragment, and the hyperparameter is standardized across all sequences. The number of fragments n selected for a given sequence is calculated based on the original sequence length S, the coverage hyperparameter C, and the sample length L using Equation 1. This formula differs slightly from Vervier et al. (2015) because this formula always rounds up the number of required fragments. Coverage defines the desired number of times that a given base pair in the sequence should be present among the fragments generated for that sequence.

$$n_{frag} = \left\lceil \frac{S * C}{L} \right\rceil \tag{1}$$

As fragments are drawn for a sequence, a check is performed to determine whether the selected fragment is valid. Valid fragments consist only of the four standard DNA nucleotides, and invalid fragments are discarded. Unlike the approach in Vervier et al. (2015), if a given fragment is invalid, a new random fragment is generated to replace it. After fragments have been drawn for the sequence, the taxid (species) for the sequence is appended as the final dimension for each fragment array. Fragment data for each sequence is written to disk as separate binary files to allow for parallelization of the fragment generation process.

The difference between the two approaches developed in this work involves the design of the data array for the fragments. In the first approach, fragments are stored in memory as string arrays in which the entire fragment is a single column. In the second approach, fragments are stored in memory as character arrays, and each letter has its own column. The first approach was found to be faster and to result in smaller binary output files, but introduced additional complexity during encoding. As a result, the second approach was used for the remainder of the work.

2.1.2 Fragment encoding

After fragments have been generated for each sequence, the data must be encoded for use in classification. Following the approach from Vervier et al. (2015), fragments are subdivided into k-mers of a desired length and encoded using One-Hot Encoding. The k-mers are grouped starting at the first position in each fragment. Only full-length k-mers are retained; the remaining portion of the fragment is discarded.

Differences in the two fragment generation approaches developed in this work require different approaches for fragment encoding. For the first fragment generation approach in which fragments are stored as single strings, each fragment is individually processed into k-mers using a complex mapping process. For the second fragment generation approach in which fragments are stored as individual characters in an array, all fragments can be grouped into k-mers in parallel using vectorized NumPy operations. The second approach was used for all model tests.

2.2 Linear Regression

Linear regression classifies binary data using a linear hyperplane. The linear regression model developed for this work implements gradient descent following Mitchell (1997). Both gradient descent and L2 penalized gradient descent were implemented for comparison purposes. Multiclass linear regression was implemented using a one-vs-all strategy. Additionally, all code was implemented to handle sparse matrices natively to ensure that large metagenomics datasets could fit into memory on limited hardware (i.e. a laptop).

2.3 SVM

SVM classifies binary data using a linear hyperplane. Two different `scikit-learn` versions of SVM were used in testing: SVC (rbf kernel) and LinearSVC. Each uses a variety of predetermined regularization parameters C to train multiple models. Multiclass SVC was handled using a one-vs-one scheme and multiclass LinearSVC was handled using a one-vs-all scheme.

2.4 Random Forest

Random Forest classifiers generate a number of random decision trees of predetermined length and collectively uses these decision trees as a committee to predict sample class. Hyperparameters can be set for both maximum depth of the trees and number of estimators. This approach is natively multiclass.

2.5 Naïve Bayes

Naïve Bayes uses data to generate distributions for each class, and uses these distributions to predict class labels. The version of Naïve Bayes implemented in this work was designed to handle sparse matrices to ensure data fit in memory. Given a training set of one-hot encoded sequences and taxids for those sequences, the taxid probabilities are calculated and the training dataset is split based on taxid. For predictions, the most probable taxid was calculated given the input test sequence using the Naïve Bayes formula.

3 Results

Models were tested for performance against both 2000-length and 3000-length datasets, with hyperparameters for sequence length, k-mer size, and coverage explored using a grid search. Results were generated by running each configuration 5 times and averaging the scores. Results for the 3000-length dataset were very similar to the 2000-length dataset. For the purposes of comparing generative and discriminative models, the results of the 2000-length dataset plots were sufficient, so the 3000-length plots were excluded from this section.

3.1 Logistic Regression

For the implemented version of logistic regression (Figure 2), as coverage improved, model performance improved. Performance also improved dramatically as the k-mer value is increased from 1 to 4, then dropped off slightly for larger values of k. Sample length of 200 achieved the best results. Tests were also performed using an L2 penalty (not shown), but no real difference in performance was found. For the 3000-length dataset (not shown), the implemented version produced less predictable results for the hyperparameters. This result is attributed to the model having difficulty correctly classifying twice as many species and was not seen in the `scikit-learn` version for 3000-length tests.

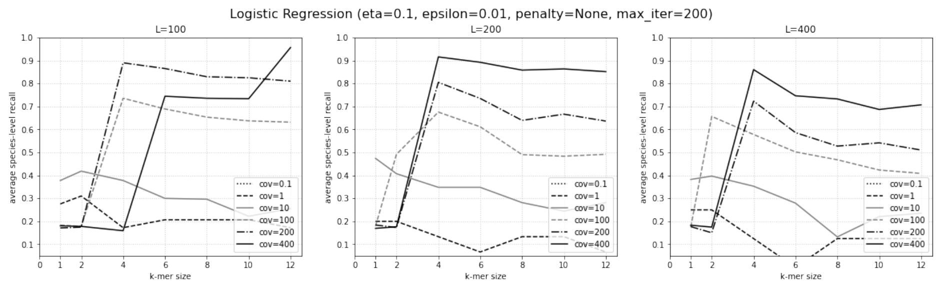

Figure 2: Logistic Regression 2000-lengths dataset results

For the `scikit-learn` version of logistic regression, multiple hyperparameters were tested. The results of using an L1 penalty are shown in Figure 3. The effects of increasing coverage and k-mer size had similar effects on this model, with an even more significant drop-off in performance for larger k values. The performance for this model peaked using a sample length of 100.

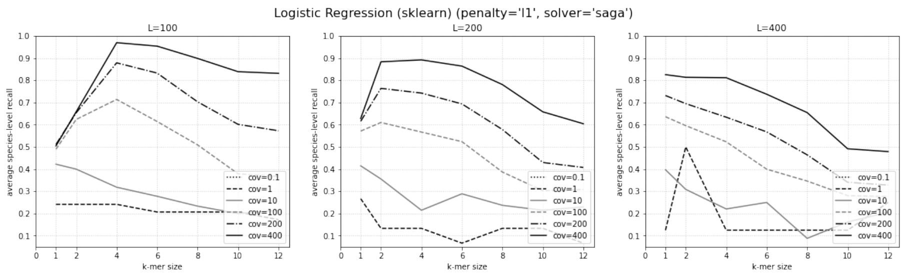

Figure 3: Logistic Regression (sklearn) 2000-lengths dataset results

Compared against L2 penalty (not shown), both L1 and L2 penalties resulted in identical peak performance. The L2 version experienced less degradation as k was increased. Additional studies were performed to determine whether using balanced class weights to compensate for class imbalance made a difference in model performance (not shown). For the small datasets tested, this was not found to significantly impact performance.

3.2 SVM

For sufficiently large coverage, LinearSVC results (Figure 4) show a peak performance with k-mer size of 4 and a slight performance decrease as the k increases. The SVC model produced significantly worse results compared with the LinearSVC model. Regarding hyperparameter tuning, altering the C-value for LinearSVC (not shown) had no effect on performance but resulted in better performance for the SVC model when C-value was greater than or equal to 1.

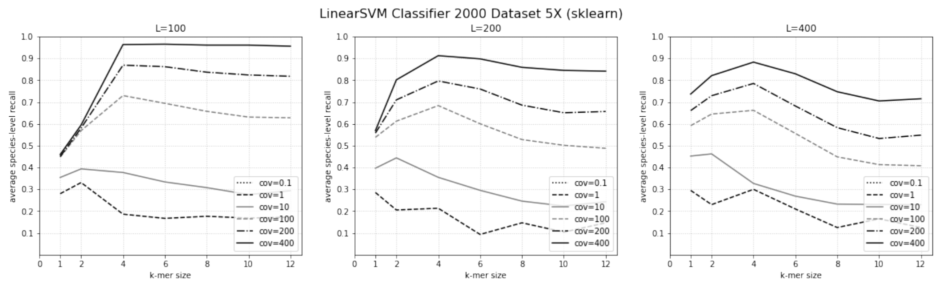

Figure 4: LinearSVC 2000-lengths dataset results

3.3 Random Forest

The results of testing using the Random Forest classifier are shown in Figure 5. Hyperparameters were tuned via grid search (not shown). Both max_depth and n_estimators had positive correlation with model performance. Results show that increasing coverage had similar improvement to that found in the other models. Smaller sample length also resulted in better performance. In contrast to the linear models, this model had high scores for very small k-mer sizes, with a sharp decrease as k was increased.

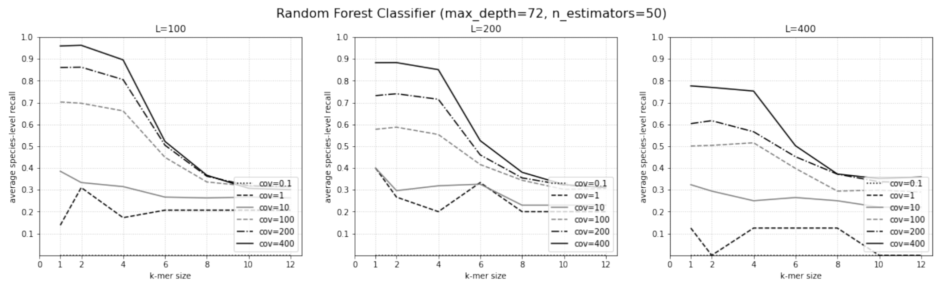

Figure 5: Random Forest 2000-lengths dataset results

3.4 Naïve Bayes

The Naïve Bayes classifier (Figure 6) showed relatively poor performance with k-mers smaller than size 4, but performance sharply increased when k-mer size was increased to 6 or 8. As the k-mer size increased beyond this, the performance was steady with only a slight decrease.

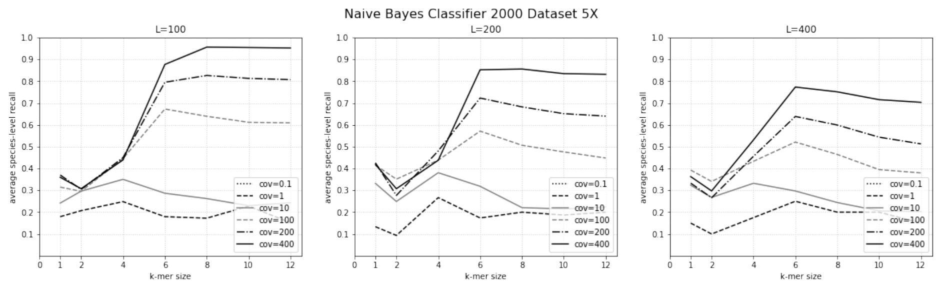

Figure 6: Naïve Bayes 2000-lengths dataset results

4 Conclusions

4.1 Summary of Findings

Comparing the overall results between our generative and discriminative models, no clear evidence was found that one type of method was advantaged over the other in terms of classification ability. Generative models were found to run much more efficiently, which makes sense given that the distributions used to predict classes came from summing values within the data, not from following an intensive optimization process. In contrast, as the dataset size grew larger than a handful of sequences, discriminative models took a very long time to perform predictions and quickly reached performance limits.

Regarding the different approaches to generating standardized datasets, small datasets had relatively balanced classes. For small datasets, classification accuracy is very high using the current strategy, but degrades as the number of sequences increases. At this point, it cannot be determined whether class imbalance due to sampling strategy has a significant impact on model performance.

4.2 Future Work

A next step for this project is solving the imbalanced dataset issue the current sampling method creates. This could possibly be done by grouping our sequences based on length before we generate our fragments or by generating additional fragments for smaller datasets to compensate. Another next step would be to check model performance on larger datasets. It is possible that there are relationships and conclusions that cannot be observed in a small dataset.

References

[1] Vervier, K., Mahé, P., Tournoud, M., Veyrieras, J. B., and Vert, J. P. (2016). Large-scale machine learning for metagenomics sequence classification. Bioinformatics (Oxford, England), 32(7), 1023–1032. https://doi.org/10.1093/bioinformatics/btv683

[2] Mitchell, T. M. (1997). Machine Learning. New York: McGraw-Hill.