Multiple-Instance Learning via Embedded Instance Selection (MILES) for Phenotype Prediction

Kyle Jackson and Robert Truong

December 19, 2016

Abstract

We extend the work of N. LaPierre et al. in [1] by using the same Metagenome-Wide Association Study dataset and apply Multiple Instance Learning to predict phenotype. We use the same assembly method from [1], but pursue a separate method for feature extraction and phenotype classification. We show that the Multiple-Instance Learning via Embedded Instance Selection (MILES) framework from Y. Chen et al. [2] has some promising features, but due to the heuristics involved in developing the pipeline for this dataset, MILES did not perform as well as the CAMIL pipeline outlined in [1].

1. Introduction

Studying the human microbiome has become an interesting area of research for the data mining and bioinformatics community due to the large amounts of data generated and the potential for classifying phenotype (disease). In [1], the authors provide an introduction that explains why metagenomics, which involves sequencing the entire microbial genome of an organism at once, is appropriate for big data techniques.[1] We use metagenome-wide association study (MGWAS) data from [3] which investigated the functional role of known gut microbiota in type 2 diabetes (T2D) Han Chinese patients. The data are also expert-labeled.

In this study, we use a Multiple Instance Learning (MIL) framework known as Multiple-Instance Learning via Embedded Instance Selection (MILES) from [2] to predict whether or not a patient has T2D based on their intestinal microbiota. MIL is a supervised learning problem where there are bags of instances, with each bag being labeled positive or negative, while the instances in each bag are unlabeled. In the case of this study, each patient is a bag and each instance within a bag is a genomic sequence (e.g. ATCCGCA…). We use MILES since [1] compares a variety of MIL methods to the CAMIL pipeline, but does not try MILES. MILES also fits within the "collective" assumption, as opposed to the standard assumption, necessary for this problem. These assumptions and our motivations will be discussed further in the literature review portion.

Our study constructs something similar to the CAMIL pipeline. Given that the data is large and contains only sequence reads, which are snippets of genomes as opposed to whole genomes, we use an

[1] We assume that readers have read or are at least familiar with [1]. We assume this because this study is a close extension of [1] and uses the exact same data and methods except for the feature extraction and classification methods.

assembly step to reduce the size of the data and construct longer sequences that are more representative of a genome. Assembly includes overlapping reads and matching ends, which signifies that the reads likely come from the same genome [1]. We then randomly sample without replacement from each patient's reads to cut down on the amount of data further. We then apply the MILES framework, where we map each bag, or patient file, "...into a feature space defined by the instances in the training bags via…" using the Jaccard similarity index [2].We then use 1-norm Support Vector Machine (SVM) to classify the patients.

This paper is organized as follows. Section 2 will review our problem statement and notations. Section 3 will include our literature review, which includes a more thorough introduction to MIL, our motivation for this study, and assumptions. Section 4 will discuss our application of the MILES framework to this problem. Section 5 includes our dataset details, our evaluation metrics, and results. Section 6 includes our conclusion.

2. Problem Statement

Our goal is to classify patient phenotype based on intestinal microbiota data. Although we use the same assembled sequences from [1], we extend it further by applying instance based feature mapping for feature extraction and 1-norm SVM, as opposed to using clustering. The application of the MILES framework to genomic data tests how instance space methods fair on large datasets and provides another method for [1] to compare CAMIL to.

3. Literature Review

The multiple instance problem was first introduced in the context of drug activity prediction by Dietterich et al. in 1997 [4]. In [4], the authors show that tackling the multiple instance problem to determine what feature vectors are responsible for the classification of a bag provides significantly better results than not doing so. Particular to drug activity prediction, drugs are molecules that bind to larger protein molecules [4]. Each molecule can take on a variety of shapes, usually small variations in the molecule structure, with some forms being able to bind and others not [4]. This leads to a number of feature vectors for one molecule. If one of those shapes binds to a target site, that molecule is classified as binding [4]. This follows the "standard" assumption, which states that a bag is classified as positive if it contains a positive instance, or molecule shape that binds, otherwise the bag is classified as negative [1][4][7].

As [1] explains, there are a number of methods to address the original MIL problem. While many of those solutions follow the standard assumption, these methods do not work in domains where negative bags contain positive instances. Recent work on labeling key instances based on their influence on the bag label has opened up the MIL problem to domains that do not follow the standard assumption [2][8]. This is important for the domain of metagenomics, where healthy patients may have pathogens within them or a combination of microbes can make a patient sick. As opposed to the standard assumption, MIL in metagenomics takes on the "collective" assumption [1][7].

The authors of [1] construct a novel MIL solution for metagenomics using data from a metagenome wide association study [3]. The data consists of shotgun sequence reads obtained from a patient's stool with multiple patient files of Han Chinese individuals [3]. Each file is labeled positive or negative depending on whether the patient has T2D [3]. Although the authors of [3] developed a simple classifier, they called for more extensive work on classifying patients. Since these metagenomic reads are only snippets of information of the intestinal microbiota and not the full genomes, [1] used an assembly method to join reads with similar ends to form longer reads, called contigs. This reduces the amount of noise and cuts down on the size of the data, making for a better classifier. [1] then uses clustering via UCLUST [9] to group the contigs into groups called Operational Taxonomic Units (OTUs), which can be thought of as a group of equivalent or similar organisms [1]. This is followed by feature extraction via a bag of words approach and a similarity and distance metric [1]. Finally, [1] uses a standard SVM classifier using the generated features. The authors also note that the choice of the classifier is not that important to this problem.

This study extends the work from [1] by applying the MILES framework from [2] to the same dataset. MILES follows the collective assumption described earlier by looking for key features that contribute to a bag's label [2]. MILES improves upon the Diverse-Density (DD) [10] framework, which is restrictive because it always seeks for one and only one key feature [2]. MILES also falls within the what is referred to as an "instance space" method [1]. The authors of [2] apply MILES to domains where the collective assumption are necessary, such as the drug activity prediction task, image categorization, and object class recognition [2]. This makes MILES are particular good choice for the MGWAS dataset, given that CAMIL [1] is not compared to it.

4. Methods and Techniques

Our project begins after the assembly step done by [1] using SOAPdenovo2 [5]. The goal of the project was not to redo the assembly step. [1] explains the reasoning behind choosing SOAPdenovo2 against other methods as well as a thorough discussion on assembly. Given this, we will briefly discuss how the assembly step works. It is done to reduce the number of reads, which are bits of genomes, by combining reads to form contigs. Contigs are more representative of genomes and provide better results for classification as it reduces the amount of noise in the data. The actual process of assembly includes overlaying reads and if there are enough matching base pairs, as determined by the user, then those reads are likely the same genome and are combined to form a contig.

Once receiving the already assembled data from the authors of [1], we reduced the data size by further stripping unnecessary artifacts and identifiers. We took the data out of the FASTA format and into simple text documents, which reduced the size of the data by over a half.

We use the MILES framework as follows:

1. We combined the instances from all the bags into an indexed pool. The goal is to transform each bag into a vector with each component corresponding to an instance.

2. For each bag and each instance, we calculated a maximum similarity, using an program called Mash [6], which provides the framework for calculating the Jaccard index between two genomes or read sequences in an efficient manner.

    a. In particular, we used the sourmash[2] implementation, which is written in Python and can compare two sequence strings, as opposed to requiring a comparison between two FASTA files. In addition, unlike Mash, it can be called within the program as a package.

    b. Mash works by using Min-Hashing. It takes a window of size *k*, and slides this window across the sequence. It hashes each subsequence the window lies on, but only preserves the *n* sequences with the lowest hashing function evaluations. The Mash paper refers to this as a sketch.

    c. It then performs the Jaccard indexing between two sketches by performing the magnitude of their intersection divided by the magnitude of their union. The Jaccard index of the sketches is an unbiased estimator for the true Jaccard index taken over all the windows, but is much more space-efficient to calculate.

    d. The original MILES paper handled data in the form of Euclidean vectors, and searched for the minimum Euclidean distance between an instance over every instance in a given bag. Calling that minimum distance *d,* it then calculates e^(-(d/σ)^2) with σ as a predetermined parameter. The paper remarked that this could be viewed as a similarity measure, so we elected to just substitute the Jaccard index here.

3. This was followed by the classification step, where we used 1-norm SVM[3]. SVM is a process which attempts to divide two classes by a hyperplane that maximizes the distance over which the two classes are separated. In the cases where the two classes are not strictly linearly separable, it will couple a soft-margin term that also seeks to minimize the amount the model line misclassifies the point by also seeking to minimize the distance the line is from those points. The problem essentially boils down to trying to minimize:

    $$\text{average hinge loss} + \lambda * ||w||^2$$

    Where w is the size of the margin that the hyperplane separates the two classes by, and hinge loss is defined to be 0 if the point is on the correct side of the line, and proportional to the distance to the margin if it's on the wrong side. This is typically solvable by quadratic programming. What 1-norm SVM does is it replaces ||w||^2 with ||w||_1, calculated using the 1-norm or Manhattan

[2] https://github.com/dib-lab/sourmash
[3] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Distance. This just takes the absolute values of each component of w and adds them, instead of squaring w's components like normal SVM does. This system is then solvable with linear programming instead, which is computationally cheaper to calculate.

5. Discussion and Results

5.1 Datasets

Our data comes from a metagenome-wide association study (MGWAS) which investigated the functional role of known gut microbiota in type 2 diabetes (T2D) Han Chinese patients [3]. The data is available online[4] and provides expert labels in the supplementary tables of [3]. Since matching the labels to the sample IDs in the data took a considerable amount of effort, the authors of [1] provided us with the labels[5].

The data consisted of 367 patient files in the FASTQ format, with each patient file containing millions of lines of reads. At the initial download, the data was 3.29 TB, although when we received the data from [1], the size of the data was approximately 25.2 GB. Of the 367 patients, 182 had T2D and 185 were healthy controls. After we finished preprocessing the data after receiving it from the authors of [1], we were able to shrink the data further to about 10 GB.

Due to the time constraints of this study, we had to not only sample the number of sequences from each patient, but also the number of patients we examined. Table 1 shows the difference between the data used in [1] and this study.

Table 1: Statistics on data used in CAMIL [1] and this study

	Patient Files (*n*)	**Positives (T2D)**	**Negatives (con)**	**# of Total Sequences**
CAMIL	367	182	185	~56 million
This Study	183	95	88	27,582

While we would have preferred to analyze the same data as CAMIL used; however, the time needed to map the bags into the feature space was not feasible with the original data size. This topic will be discussed in section 6.

5.2 Evaluation Metrics

To evaluate our results from classification using a 1-norm SVM, we used the same methods from [1], namely accuracy, F1 score, and Area Under Curve of the Receiver Operating Characteristic (AUC-ROC).

[4] https://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP008047&go=go, https://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP011011&go=go
[5] https://github.com/kjacks21/MILES-python

While many supervised learning problems deal with largely imbalanced classes, this dataset is relatively balanced. This means using accuracy as a metric is appropriate, followed with the two other methods. In addition to these metrics, we used 5-fold cross-validation[6] to ensure our classifier was not overfitting to the training bags.

5.3 Experimental Results

The results for our study are seen in Table 2 below:

Table 2: Results

	Accuracy	**F1 Score**	**AUC-ROC**
Results	55.79	42.79	52.68

The results were disappointing, but also expected. Even though the metrics from Table 2 are markedly worse than CAMIL's performance [1], we were above 50% accuracy and AUC-ROC after averaging the results from our k-fold cross validation. There were a number of reasons why we have these results, as we will discuss below.

The majority of the MILES framework, including the preprocessing steps such as sequence assembly, involve heuristics. While we did not re-do the assembly step as done in [1], it came to our attention that the parameters used were likely not optimal, such as the number of k-mers. This is especially important because the sequences are not full genomes, relatively small, and are not being compared to a known database of pathogens. The choice for the value of k-mers also arose at the step for calculating the similarity between sequences in the instance mapping step of MILES. In particular, a larger value for k-mer provides more specificity, while smaller k-mers provide more sensitivity [6]. Although we did vary the k-mer parameter slightly when using Mash, we started at a high k-mer value and lowered it until we started getting values greater than 0, meaning that there was some similarity between sequences.

When calculating the similarity between sequences for the instance mapping stepped, we only tried two measures: jaccard similarity index and hamming distance. Ideally, we would have tried a couple others, but due to time constraints we could not. Given how new Mash [6] was, using it all the way through to the classification step was a good use case and learning experience.

The decision to reduce both the number of sequences and patient files through random sampling without replacement was both regrettable and necessary. Even with access to the ARGO cluster[7] at Mason, a supercomputer, we were not able to generate the output files necessary for our analysis due to a number of technical issues and general unfamiliarity with using the cluster. We spent about two weeks troubleshooting and trying to run parallel jobs on the cluster to map the bags to the instance space, but were unsuccessful. Even though we discovered a major bug in our code a day before this report's deadline, we had to make the decision to shrink everything to run on a local machine a few days prior.

[6] http://scikit-learn.org/stable/modules/cross_validation.html
[7] http://wiki.orc.gmu.edu/index.php/About_ARGO

This meant a substantial loss of information for training the classifier and testing it. This would be resolved given a longer timeline for this project.

While the time required to train the classifier was marginal, mapping the instance space on our laptop took 27 hours and 25 minutes with no parallelization. Getting the mapping done in a reasonable time was one of the hardest parts of this project and one that we would improve upon in the future.

6. Conclusion

While our results were not great and we did not have the opportunity to exploit the entire dataset, it was still a great learning experience. Prior to this project, neither of the authors here had heard of multiple instance learning. Now, given both our literature review on the subject and the application of it to a large dataset, we feel more comfortable with the topic and where it can be appropriately applied. We also learned quite a bit about bioinformatics, particularly the different kinds of data generated from organisms and how to process the data. We also enjoyed doing this project because of the demand for bioinformaticians and how interesting the topic and its applications are. Although we are not experts in the field, we now have a solid foundation to build on and can engage in some discourse on the matter.

Another useful thing we learned was how to interface with the ARGO cluster, but more generally clusters for computing. This knowledge will be useful for future projects at school but also in industry, where the large amounts of data are likely run on the cloud.

Finally, the act of engaging in a more rigorous project from a research perspective, as opposed to doing a Kaggle competition and using pre-made libraries for everything, provided us insight into the world of research in computer science. We became better at reading papers, identifying useful information, and how to better read method and algorithm explanations. The latter part we spent a particularly large amount of time on, discussing what our interpretations are, how we can go about implementing the methods, and what liberties we could take to make the methods better for our use case.

6.1 Directions for Future Work

This project was only the first step in applying MILES to this dataset. In the future, we would also optimize the assembly step. We would also look into a number of different similarity measures and implementations for our instance mapping step. Further, we would more thoroughly explore instance labeling as specified by the MILES framework [2]. It is also clear that some sort of preprocessing would need to be done to make this program tractable using MILES, but obviously random sampling at the extent we did was not the appropriate direction to go. Perhaps random sampling at 1/100th or a different larger proportion might have made the resulting sampled sets more representative of each patient as a whole, or we could have looked into the literature further to see if techniques for pulling out or generating representative instances from a bag of sequences existed. Finally, we would get multi-processing jobs running successfully on the ARGO cluster, which would have greatly reduced our compute time needed.

References

[1] N. LaPierre et al., “CAMIL: Clustering and Assembly with Multiple Instance Learning for Phenotype Prediction,” in *IEEE International Conference on Bioinformatics & Biomedicine,* Shenzhen, China, 2016.

[2] Y. Chen et al., “MILES: Multiple-instance learning via embedded instance selection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, no. 12, pp. 1931 – 1947, 2006.

[3] J. Qin et al., "A metagenome-wide association study of gut microbiota in type 2 diabetes," *Nature,* vol. 490, no. 7418, 55-60, 2012.

[4] Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. Artificial intelligence, 89(1), 31-71.

[5] Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., ... & Tang, J. (2012). SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, *1*(1), 1.

[6] Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., & Phillippy, A. M. (2016). Mash: fast genome and metagenome distance estimation using MinHash. *bioRxiv*, 029827.

[7] Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. Artificial Intelligence, 201, 81-105.

[8] Liu, G., Wu, J., & Zhou, Z. H. (2012). Key Instance Detection in Multi-Instance Learning. ACML, 25, 253-268.

[9] R. Edgar, “Search and clustering order of magnitude faster than blast,” *Bioinformatics*, vol. 26, no. 19, pp. 2460-2461, 2010.

[10] Maron, O., & Lozano-Pérez, T. (1998). A framework for multiple-instance learning. Advances in neural information processing systems, 570-576.