CAMIL: Clustering and Assembly with Multiple Instance Learning for Phenotype Prediction

Nathan LaPierre
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: nlapier2@gmu.edu

Mohammad Arifur Rahman
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: mrahma23@gmu.edu

Huzefa Rangwala
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: rangwala@cs.gmu.edu

***Abstract*—**

The recent advent of Metagenome-Wide Association Studies (MGWAS) has allowed for increased accuracy in the prediction of patient phenotype (disease), but has also presented big data challenges. Meanwhile, Multiple Instance Learning (MIL) is useful in the domain of bioinformatics because, in addition to classifying patient phenotype, it can also identify individual parts of the microbiome that are indicative of that phenotype, leading to better understanding of the disease. We demonstrate a novel, efficient, and effective MIL-based computational pipeline to predict patient phenotype from MGWAS data. Specifically, we use a Bag of Words method, which has been shown to be one of the most effective and efficient MIL methods. This involves assembly of the metagenomic sequence data, clustering of the assembled contigs, extracting features from the contigs, and using an SVM classifier to predict patient labels and identify the most relevant read clusters. With the exception of the given labels for the patients, this entire process is *de novo* (unsupervised). We use data from a well-known MGWAS study of patients with Type-2 Diabetes and show that our pipeline significantly outperforms the classifier used in that paper, as well as other common MIL methods. We call our pipeline “CAMIL”, which stands for Clustering and Assembly with Multiple Instance Learning.

I. Introduction

The human body contains one of the most dense and diverse microbial environments in the world. The human and microbial cells are collectively referred to as the *human microbiome* [1], [2]. Advances in biotechnology have allowed scientists to directly interrogate the human microbiome, particularly the development of high throughput sequencing technologies, which generate massive amounts of biological data. In recent years, improving capabilities in data science have allowed for the study of *metagenomics*, which involves sequencing the entire pool of microbial genomes at once. This data is usually gathered via *shotgun sequencing*, which generates a number of short *reads* containing bits of genomic data from the microbes of the host environment [3]. These reads, represented as strings of nucleotides, represent only small parts of the microbe’s full genome, and are not ordered in any way, which presents several challenges that will be discussed further in the paper.

Metagenomics has several advantages: (i) microbes are now understood to be the underlying cause of many human diseases and are also critical to many chemical processes and overall health [4]; (ii) it is believed that most microbes have not been laboratory-cultured and thus remain unknown [4]; (iii) whereas other methods such as 16S rRNA analysis are mainly useful for predicting the species of microbes (*phylogeny*), metagenomics contains other critical information from the microbial genomes that determine how these microbes function and affect diseases and chemical processes (*functional* information) [4]. Summarily, metagenomics allows us to view microbial data that is not accessible to us via traditional laboratory culturing and allows for both phylogenetic and functional profiling of those microbes. Thus, studying microbial metagenomics is an effective way to predict and model human disease, also known as clinical *phenotype*.

In this study, we develop an efficient classifier that predicts whether or not a patient has a disease based on their microbiome. We view this as a Multiple Instance Learning (MIL) problem, in which we have several *bags* of instances, and we have labels for the bags, but not for each instance within them. In this case, we have a patient (bag) and a label for each patient (whether or not they have a disease), but no labels for each patient’s sequence reads (instances). Specifically, we use Bag of Words (BoW) methods, discussed further in the Background section, which have been shown to be among the most effective and efficient MIL methods [11]. MIL has been studied in many contexts, but it has rarely if ever been studied in the context of predicting clinical phenotype based on metagenomic data from the microbiome. However, since datasets in this domain frequently have patient-level labels but almost never have instance-level labels, this is a well-suited domain for multiple instance learning.

We used data from a Metagenome-Wide Association Study (MGWAS) [32], which compares microbial metagenomic data between many patients with or without a given phenotype. MGWAS studies contain many expert-labeled patients and the metagenomic data associated with those patients, so they are useful for phenotype prediction but also cause many computational challenges. The data is very large (multiple terabytes) and high-dimensional (thousands of dimensions). Additionally, due to the nature of shotgun sequencing, most of the reads are not useful by themselves, and must first be assembled. *Assembly* is the process of combining pairs of reads in which the end of one read overlaps with the beginning of another, signifying that they are probably contiguous reads from the same genome. This process is repeated as much as possible to form long strings called *contigs*. Assembly also reduces the size and dimensionality of the data by discarding reads that cannot be assembled successfully. The reduction in data size also allows for clustering, which is not feasible for massive datasets. *Clustering* uses string similarity measures to group similar reads into "clusters", which is a way of identifying which species of microbe each read corresponds to. The alternative, "aligning" the reads with known genomes, is impractical for metagenomics, since many of the involved microbes have not yet had their genomes sequenced. From the clustering output, we extract feature vectors, which are then fed into a Support Vector Machine (SVM) classifier.

Thus, the entire pipeline consists of assembling the reads of each patient, combining the resulting contigs from each patient into one file, clustering the contigs, extracting features from the clustering output, and performing classification with the SVM. This process is explained in further detail in the methods section. We refer to the pipeline as "CAMIL", which stands for Clustering and Assembly with Multiple Instance Learning. We then compare the results of our method against the classifier used in the MGWAS study [32] from which we derived our data, as well as other popular MIL methods. We show that our classifier shows significantly improved performance. We have also released code for our pipeline on GitHub[1], under the open-source MIT license.

II. Background

A. Multiple Instance Learning

The Multiple Instance Learning (MIL) problem was first described by Dietterich in the context of drug activity prediction [5]. Each molecule can assume a number of different 3-dimensional shapes (conformations), so even for molecules that are known to bind to the binding site, it is not necessarily known which conformation of the molecule succeeds in binding. If even one shape of a given molecule binds to a binding site, it is considered a "good" molecule [5]. Thus, in the original formation of MIL, a bag is classified as positive if one or more instances within it is positive, while a negative bag contains only negative instances. This is commonly referred to as the "standard multiple instance assumption" [11]. In the original paper, Dietterich developed a solution based on axis-parallel rectangles to solve this problem, and the MIL approach was shown to be significantly more effective than a standard supervised learning approach [5]. In the late 1990s and early 2000s, a number of different approaches were developed for the original MIL problem, such as Diverse Density (DD) [6], EM-DD [7], MI-SVM [8], sbMIL [9], and MILES [10]. A recent review of MIL by Amores created a taxonomy of these various methods and compared their effectiveness for classification [11]. Most of these methods follow the standard assumption, which is not useful in real domains in which negative bags may contain some proportion of positive instances.

More recently, there has been increased interest in different formulations of the MIL problem. For instance, the problem of "key instance detection" [12] revolves around finding the instances that contribute the most to bag labels. A recent study by Kotzias et al. focused on a formulation of the MIL problem in which bags with negative labels can contain some positive instances, and developed a general cost function for determining individual instance labels from group labels [13]. This is significant in metagenomics because, while some diseases are caused by a single pathogen, many arise from a combination of many factors, and even patients that are healthy may contain small amounts of pathogens that are normally associated with disease. In contrast with the standard assumption, this can be referred to as the "collective" assumption [11]. Additionally, it is helpful to discover which microbes and which functional attributes of those microbes lead to disease, making instance level information significant in this domain.

While some of the above methods follow the standard assumption and others follow the collective assumption, they all treat bag labels simply as aggregations of instance labels and thus focus only on comparisons between individual instances and not entire bags or groups of instances. For methods following the standard assumption, the aggregation function is simply an OR function: if any of the instances in a bag are positive, the entire bag is positive. For methods following the collective assumption, the aggregation function is often based on an averaging of instance labels. Regardless of the problem assumption, methods that rely only on comparisons between individual instances are referred to as "Instance Space" by Amores; this paradigm was generally less effective than the two other paradigms, "Bag Space" and "Embedded Space" [11]. Bag Space methods define a distance or kernel function that determine the similarity between bags, while Embedded Space methods map bags into feature vectors which can then be used for classifiers [11]. Embedded Space methods can be further divided into two subcategories: methods that simply aggregate information about all instances in a bag without differentiating them, and "Vocabulary-based" methods that group certain similar instances together and then use those groups to form the feature vector [11]. We use a vocabulary-based method in this paper, because having information about groups of similar sequence reads can be biologically important, as explained further in the clustering section.

The Multiple Instance paradigm fits phenotype prediction well, since we have a set of labeled patients containing unlabeled sequence reads, and we would like to predict both the patient phenotype and which reads are indicative of that phenotype. Despite the recent developments in MIL and its potential utility in phenotype prediction, we have not found any literature that specifically applies MIL to classifying patient phenotype based on metagenomic data. We present our MIL-based feature extraction method in section III-D.

B. Assembly

The *assembly* problem involves combining overlapping short reads (usually less than 1000 base pairs) into longer sequences called *contigs* (often tens of thousands of base pairs). For instance, if one read ends with the same relatively large nucleotide string that another read starts with, the reads are likely to be overlapping fragments from the same genome, and can thus be combined into one contig. This can be done either *de novo* (in an unsupervised manner) or by referencing sequences against known contigs. We focus on de novo assembly, in order to keep our pipeline as unsupervised as possible.

One of the main purposes of assembly is to determine the whole genomes of microbial species, the vast majority of which have not or cannot be laboratory cultured, from sequencing reads [27]. Even if complete genomes cannot be assembled,

[1] https://github.com/nlapier2/multiInstanceLearning

combining reads into larger contigs can still make them much more useful for clustering and classification, because the contigs will contain more phylogenetic and functional information than short reads. This is because short reads of less than 1000 base pairs constitute only a tiny fraction of microbial genomes, which are usually hundreds of thousands to millions of base pairs, making it difficult to ascertain much about the phylogeny of individual reads. Many modern sequence reads are produced by Next-Generation and High-Throughput Sequencers, which usually produce these short reads. Metagenomics poses its own set of challenges, due to large datasets and lack of knowledge about how many species are present and in what abundances [28]. Thus, metagenome assembly is a new and challenging field. Some popular single genome and metagenome assembly approaches include SOAPdenovo2 [25], IDBA-UD [26], Velvet [27], MetaVelvet [28], and Ray Meta [29].

C. Clustering

The *clustering* problem in this context involves grouping input short sequences (reads or contigs) such that sequences within a group are similar to each other. The clusters obtained from this process are referred to as Operational Taxonomic Units (OTUs). OTUs represent a group of equivalent or similar organisms. Accordingly, the number of OTUs in a sample gives an approximation of the species diversity in that sample [17], [18], [19]. In addition to approximating species diversity, clustering has several other key advantages. Because clustering is always de novo (unsupervised), it is not limited by the species that are covered in taxonomic databases. This is important because it is believed that most micro-organisms that reside in the human body have not been laboratory cultured [4]. Clustering also reduces computational costs by allowing analyses to operate on entire clusters instead of on each read/contig. Finally, clustering helps the classification process by allowing feature vectors to be built at the OTU level, instead of using individual short reads. UCLUST [15], CD-HIT [16], mothur [17], DOTUR [18], CROP [20], and MC-MinH [21] are some of the popular sequence clustering approaches.

III. Methods

A. Overview

Our proposed pipeline involves a number of steps, which serve a variety of purposes. For each patient file, we assembled the sequence reads, which served the dual purpose of generating larger contigs that contain more functional biological information and reducing the dataset size by discarding reads that could not be assembled. The clustering step assigns the contigs to certain clusters, which represent functionally similar microbes, and thus establish classes of instances that can be used as features for the classifier. We then developed a vocabulary-based feature extraction method, discussed further in subsection III-D. Using the extracted feature vectors, we trained an SVM-based classifier to predict patient phenotype, and used several metrics to assess its accuracy. We used the SVM's decision boundary to infer information about which clusters of instances were most or least indicative of the phenotype, discussed further in subsection III-E. Aside from the patient labels, this process is entirely de novo, and does not consult any external databases. An illustration of the pipeline is shown in Figure 1 on page 4.

B. Assembly with SOAPdenovo2

For our assembly step, we used SOAPdenovo2, because it was the assembler used in the MGWAS study [32] that we compare our results with and because it has been shown to be one of the fastest assembly algorithms [26]. It should be noted that SOAPdenovo2 was not originally intended for metagenome assembly, but is often tuned for that application, as was done by us and Qin et al. [32]. We tested a number of different combinations of parameters, and found that the best results came when we cut reads off after 100 base pairs (reads were 180 base pairs long originally) and used a k-mer size of 51. The average insert size was set to 350, in accordance with the reported average insert size from the MGWAS study that we used data from [32]. The patient files needed to be assembled separately, in order to avoid assembling reads from different patients together. Conversely, all contigs need to be in one file for clustering, to avoid inconsistent cluster assignments between different patients. Thus, we combined the contigs from each assembled patient file into a single for clustering.

C. Clustering with UCLUST

We use UCLUST within our study, which is one of the most widely used and cited metagenome clustering methods and has been shown to be amongst the most effective in terms of speed and accuracy in benchmarking studies [22], [23]. UCLUST seeks to ensure that, for some similarity T, the following conditions hold: (i) all cluster centroids have a similarity of less than T to each other; and (ii) all points in a cluster have a similarity of greater than T to the cluster centroid [15]. UCLUST proceeds in a greedy, iterative manner. The first sequence in the input file becomes a new cluster centroid. For each new sequence in the file, it is compared with each of the existing cluster centroids in order. As soon as it is compared with a centroid that it has a similarity of greater than T with, it becomes part of that cluster. If the read is not similar enough with any of the existing cluster centroids, it becomes the centroid of a new cluster. The similarity measure T is defined as a string similarity between the two nucleotide sequences that counts the number of character placements that they have in common and then divides that number by the length of the reads, with terminal characters excluded [15].

Since our contigs were not ordered, we used the usersort option, and we set the sequence match threshold to 40%, which means that two reads needed to have 40% of the same nucleotides to be in the same cluster. For instance, between two strings of length 100, at least 40 places in each of those strings would have to contain the same nucleotide (represented as A, T, G, or C). New contigs that did not match at least 40% to any of the existing contigs would form the seed of a new cluster. This value of 40% was found to be the best value based on our experiments with this dataset. Other values tried, including 50%, 75%, and 90%, led to many clusters with very few reads per cluster.

D. Feature Extraction and Classification

We used a "vocabulary-based" feature extraction method. An example of Vocabulary-based methods are Bag of Words (BoW) methods, which involve the following three-step process: (i) Cluster the instances to create classes of instances;

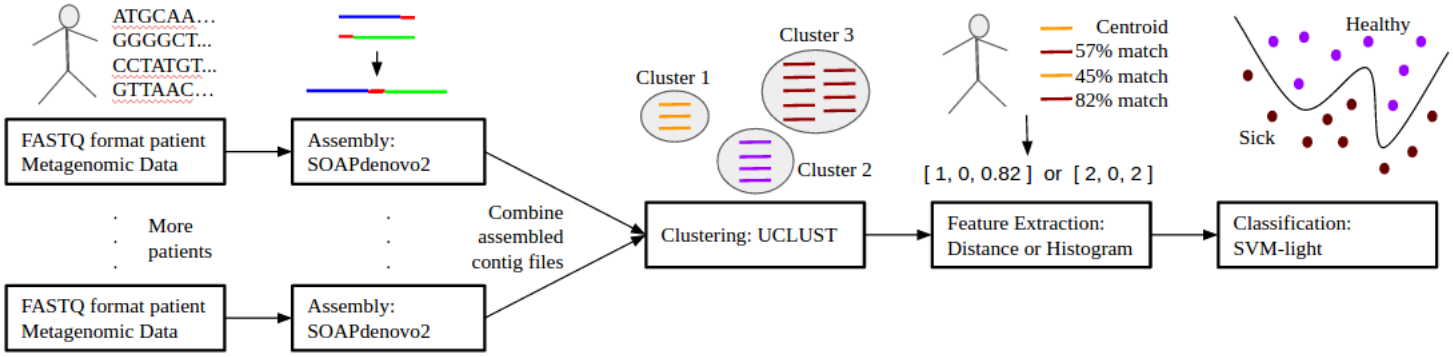
ATGCAA…
GGGGCT…
CCTATGT...
GTTAAC…
FASTQ format patient Metagenomic Data
Assembly: SOAPdenovo2
More patients
FASTQ format patient Metagenomic Data
Assembly: SOAPdenovo2
Combine assembled contig files
Cluster 1
Cluster 3
Cluster 2
Clustering: UCLUST
Centroid
57% match
45% match
82% match
[1, 0, 0.82] or [2, 0, 2]
Feature Extraction: Distance or Histogram
Healthy
Sick
Classification: SVM-light

Fig. 1. This diagram illustrates the entire CAMIL pipeline. Patient files with FASTQ metagenomic reads are individually assembled using SOAPdenovo2, then combined into one file and clustered with UCLUST. We extract features according to either the D-BoW or H-BoW method, and classify the patients using the feature vectors with svm-light.

(ii) for each bag, map the clusters of instances in that bag to a feature vector; and (iii) use a standard classifier that uses the feature vectors to predict group labels [11]. Step (i) is covered by our assembly and clustering process, while step (iii) is covered by performing classification with a standard SVM classifier based on the extracted feature vectors. In this case, we used svm-light [31]. Below, we describe our feature selection methods for step (ii), which we implemented in Python, as well as the rationale for using these methods.

Amores found the Distance-based Bag of Words (D-BoW) method to be the second most effective of all tested methods, and the most effective one that was also time-efficient (linear, rather than quadratic, in the number of bags and number of instances per bag) [11]. H-BoW methods were found by Amores to be somewhat less effective than D-BoW methods on average, but performed the best out of all algorithms on several datasets, indicating that this method performs very well on some real world problems [11]. Thus, we tested our pipeline using both of these feature extraction methods.

Either way, the input is a set of clusters for each patient. The D-BoW method creates a feature vector based on the contig for each cluster that was the closest match to the cluster seed. For instance, say Patient A's reads include the centroid of cluster 1, another contig that has a 45% match to the centroid of cluster 1, no contigs from cluster 2, and two contigs that match to the centroid of cluster 3, one with a 57% match and one with an 82% match. The string match percentage is determined by UCLUST, as described in the previous subsection. Then, D-BoW would extract the feature vector [1, 0, 0.82], indicating the contigs for Patient A that match most closely to the cluster centroid for each cluster. The H-BoW method, instead of using the closest match to each cluster, counts the number of contigs for a patient that belong to each cluster. For the above example, the H-BoW method would extract the feature vector [2, 0, 2], since Patient A has 2 representatives from clusters 1 and 3, but no representatives from cluster 2. This example is illustrated in part of Figure 1.

E. Deriving Instance "Labels"

One of the benefits of using Multiple Instance Learning methods is that we can attempt to discover instance "labels". In fact, we did not attempt to apply static, unchanging labels to individual reads or clusters, since organisms are affected by their interactions with each other. For instance, a patient with X amount of microbe A or X amount of microbe B may be healthy, but with X/2 amount of microbe A and X/2 amount of microbe B they may be sick. This simple example is illustrated in Figure 2. We can infer from the SVM decision boundary which clusters appear to be most relevant to the disease diagnosis. Since feature vectors are multiplied by the weight vector of the decision boundary to determine the label of the patient, we can assume that clusters with the highest weights in the weight vector are most relevant to the disease diagnosis. For instance, if the *i*th scalar in the weight vector is has the highest value of any of the weights, then cluster *i* is likely to play a major role in the disease pathology. Similarly, the most negative weights in the weight vector indicate clusters whose presence in a patient indicates that they likely do not have the disease. Because the data is metagenomic, the clusters represent both phylogenetic and functional similarity, so identifying the most relevant clusters can help discover more about the pathology of the disease. For Type 2 Diabetes, which is a complex phenotype and a disease that is both common and deadly, this is potentially quite valuable.

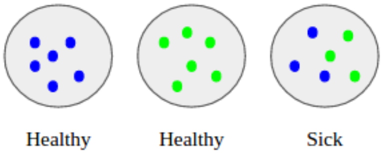

Fig. 2. This diagram illustrates why static instance labels are not sufficient for phenotype prediction. A patient with 6 of the blue microbe or 6 of the green microbe may be healthy, while a patient with 3 of each is sick. Static instance labels cannot capture this relationship. This is also explained by Amores [11].

IV. Materials

A. Dataset Description

We used data from a well-known Metagenome-Wide Association Study by Qin et al. of Type 2 Diabetes (T2D) in Chinese patients [32]. This study was chosen because it is one of the only MGWAS studies that made its data available online and labeled the phenotype of the patients, and is one of the largest among those studies. Additionally, the authors called for more extensive testing of gut microbiota classifiers [32].

The full dataset used in this study contains 367 patients [32]. Each patient file was downloaded from NCBI[2] and converted to FASTQ format using the SRA toolkit[3]. The labels were found in the paper's Supplementary Tables [32]. The total size of these 367 FASTQ files was 3.29 terabytes, with an average size of 8.97 gigabytes per patient file. Out of the 367 patients, 182 were diabetic and 185 were healthy controls.

B. Evaluation Metrics

We can assess the success of our classifier in several ways. The simplest measure, accuracy, measures the percentage of instances that are classified correctly, represented by

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (1)$$

where TP, TN, FP and FN represents true positives, true negatives, false positives and false negatives respectively.

Accuracy as an evaluation metric can be biased if one of the classes (positive or negative) has a larger number of examples than the other. Precision measures the percentage of positive predictions that were correct, whereas recall measures the percentage of positive examples that were correctly predicted (or retrieved). We can represent Precision and Recall as:

$$Precision = TP/(TP + FP). \quad (2)$$

$$Recall = TP/(TP + FN). \quad (3)$$

The F1 score captures the trade-offs between precision and recall in a single metric and is the harmonic mean of precision and recall, given by:

$$F1 = 2 * (Precision * Recall)/(Precision + Recall). \quad (4)$$

Finally, we also use the Area Under Curve of the Receiver Operating Characteristic (AUC-ROC), which measures the performance of the classifier as the decision boundary threshold is moved. The SVM classifier generally predicts a group label to be negative if the predicted label for that group was less than 0 and predicts a group label to be positive otherwise. The AUC-ROC measures the performance of the classifier as the threshold is varied to more or less than 0. In effect, it measures how far off incorrect predictions were from being correct. AUC-ROC plots True Positive Rate (TPR) versus False Positive Rate (FPR), given by:

$$TPR = TP/(TP + FN). \quad (5)$$

$$FPR = FP/(FP + TN). \quad (6)$$

C. Software and Hardware Details

We used the ARGO computing cluster available at George Mason University[4]. The clustering and classification phases were run on one of the compute nodes available on the cluster. The cluster is configured with 35 Dell C8220 Compute Nodes, each with dual Intel Xeon E5-2670 (2.60GHz) 8 core CPUs, with 64 GB RAM. (Total Cores 528 and 1056 total threads, RAM>2TB). Source codes for SOAPdenovo2[5] [25], UCLUST[6] [15], and svm-light[7] [31] were downloaded from their respective websites and compiled on the ARGO platform. The source code for our implementations of the H-BoW and D-BoW feature extraction methods and GICF are available on GitHub[8] under the open-source MIT license.

V. Experimental Results

A. Experimental Setup

Each bag of reads was assembled with SOAPdenovo2, which took 7-30 minutes per file, depending on the file size. The assembly of the patient files is embarrassingly parallel, so the assembly of each file could be done at the same time. Combining the files into one file took 2 minutes and 35 seconds. Assembly was used for all of the classification methods tested, because the combining of individual reads and reduction in total data size made classification feasible.

B. Methods Tested

This section provides an overview of the different methods that we compared our pipeline to. Our pipeline uses clustering, whereas the other methods do not. Instead, those methods directly compare individual reads instead of clusters. In order to do this, sequence reads were represented as counts of k-mers. We wrote a script to represent the reads/contigs as vectors representing the k-mer counts in the string, and these feature vectors were used by the other methods. We tested possible k values from 1 to 6; since the number of possible k-mers is exponential in k, higher values for k quickly become impractical in both run time and memory usage. From experimental validation, we found a k-mer value of 3 to be the most effective. Converting the reads from their string form to k-mer vectors with k=3 took 24 hours, 8 minutes, 49 seconds.

1) CAMIL - Our Pipeline: We implemented two different versions of the pipeline in Python: one that uses D-BoW feature extraction, and one that uses H-BoW feature extraction. These results are denoted as "CAMIL D-BoW" and "CAMIL H-BoW", respectively, in the results tables and graphs. Clustering for the pipeline with UCLUST took 14 hours, 7 minutes, 44 seconds. All 367 patients were clustered together in order to keep the features consistent.

2) MISVM and sbMIL: MISVM [8] and sbMIL [9] are two of the classic Multiple Instance Learning algorithms that fall into what Amores calls the "Instance Space" (IS) methods, in that they only use "local" information based on comparisons between individual instances and treat bag labels as aggregations of instance labels [11]. Additionally, both of these methods follow the standard MIL assumption that bags with negative labels contain only negative instances, whereas positive bags contain one or more positive instances [11]. sbMIL specifically assumes that positive bags contain few positive instances [9]. We include these algorithms as an example of many of the early MIL algorithms, which usually fell into the IS paradigm and used the standard MIL

[2] http://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP008047, http://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP011011

[3] http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software

[4] http://orc.gmu.edu/research-computing/argo-cluster/argo-hardware-specs/

[5] SOAPdenovo2: http://soap.genomics.org.cn/soapdenovo.html

[6] UCLUST: http://www.drive5.com/uclust/downloads1_2_22q.html

[7] svm-light: http://svmlight.joachims.org/

[8] https://github.com/nlapier2/multiInstanceLearning

assumption. For the implementation of these methods, we used an open-source Python implementation by Doran [14], which is available on GitHub[9].

3) GICF: The Group-Instance Cost Function (GICF) is a method proposed by Kotzias et al. that learns instance labels in addition to group labels [13]. The cost function uses a kernel that measures similarity between instances and a penalty on the difference between instance labels to generate instance labels [13]. It then sets the group label to be the average instance label of all instances in that group, using a penalty on the difference between the predicted group label and the actual group label [13]. Ideally, this would cause instances that are similar to each other to have similar predicted labels, and predicted group labels to correspond closely to reality. Unlike MISVM and sbMIL, GICF explicitly does not hold the standard MIL assumption, instead favoring the collective assumption. However, because this method compares only individual instances and not entire bags, and treats bag labels simply as aggregations of instance labels, GICF is still an Instance Space method. GICF is a generalized cost function, but the authors also use a specific version of it in their paper with squared loss for bag and instance level errors and logistic regression for classification [13]; this is the specific version that we implemented in Python. Like Kotzias et al. [13], we used mini-batch stochastic gradient descent with momentum to train the classifier and linear grid search to pick the parameters.

4) Original MGWAS Paper: The methods used by Qin et al. [32] are neither MIL-based, nor are they entirely de novo apart from patient labels. The authors first performed de novo assembly with SOAPdenovo2 [25] and then used a tool called MetaGeneMark [34], [35] for de novo prediction of genes from the assembled contigs [32]. They then combined these genes with an existing gene catalog, MetaHIT [36], and carried out taxonomic assignment and functional annotation of the genes using the KEGG [37] and eggNOG [38] databases, as well as 2,890 other reference genomes [32]. The authors defined gene markers by mapping the sequence reads from the MGWAS dataset to the updated gene catalog. They identified the 50 most important gene markers with the minimum redundancy - maximum relevance (mRMR) [33] method, using the "sideChannelAttack" R package and then used these 50 gene markers for SVM classification of T2D phenotype, using the "e1071" R package for the SVM [32]. Thus, the method in the original paper first applies de novo assembly and gene prediction methods, but then uses a number of references to identify the gene markers to be used in classification. From their results, the authors generated an Area Under Curve - Receiver Operating Characteristic (AUC-ROC) graph. The authors did not provide a learned decision boundary in their supplementary tables, only the predicted values for each patient, so we manually computed the accuracy and F1 score with an optimally-chosen decision boundary.

C. Results For Bag/Patient Labels

Qin et al. use 344 patients as a training set and 23 as a test set. However, when comparing to other MIL algorithms, we wished to have a more balanced training vs. test set split. We put 184 patients in the training set and 183 in the test set. Since this used a different training set than the one in the MGWAS paper, we do not compare our results to theirs here. Table I shows the comparison of results between CAMIL, GICF, MISVM, and sbMIL, while Table II compares the classification time and memory usage. CAMIL methods significantly outperform GICF, with the H-BoW variant of CAMIL slightly outperforming the D-BoW variant. We attempted to improve on GICF's results by selecting only one read from each cluster for each patient and discarding the other reads. This significantly reduced the computation time and improved results, showing how important the clustering step is. These results are listed as "GICF-Cluster" in the tables above. Even with this step, CAMIL was faster and more effective than GICF-Cluster, demonstrating the effectiveness of our feature extraction method. We performed Leave One Out Cross Validation for the Accuracy and F1-Score metrics (denoted in the table as CV Acc. and CV F1, respectively). CAMIL's cross validation results were slightly worse than its test set results but still significantly outperformed GICF. Cross validation results for GICF and GICF-Cluster were infeasible to calculate due to extremely long computaion time. CAMIL took much less time than GICF for two main reasons: (i) GICF requires representing each read as an array of length 64 (for k-mer length 3), while CAMIL reduces the data size with clustering and feature extraction; (ii) GICF requires expensive pairwise comparisons between each pair of instances in a mini-batch. MISVM and sbMIL require computing a kernel matrix of size N*N, where N = number of instances. Since this dataset had millions of instances, these methods crashed with memory errors, and are shown as "—" in the tables. We attempted to resolve these issues by applying the same methods that we used for GICF-Cluster, but still received memory errors.

TABLE I. PERFORMANCE WITH EVEN TRAIN/TEST SPLIT.

Method	Accuracy	F1-Score	AUC-ROC	CV Acc.	CV F1
MISVM	—	—	—	—	—
sbMIL	—	—	—	—	—
GICF	63.04	68.33	66.19	—	—
GICF-Cluster	79.31	82.35	82.38	—	—
CAMIL D-BoW	86.34	87.18	95.93	82.07	83.07
CAMIL H-BoW	**90.71**	**89.70**	**97.63**	**89.13**	**88.09**

TABLE II. CLASSIFICATION TIME AND MEMORY USAGE WITH EVEN TRAIN/TEST SPLIT.

Method	Classification Time	Memory Usage
MISVM	—	Memory Error
sbMIL	—	Memory Error
GICF	8 hours, 44 mins, 27 secs	2.646 GB
GICF-Cluster	24 minutes, 51 secs	**500.07 MB**
CAMIL D-BoW	5 minutes, 33 seconds	545.293 MB
CAMIL H-BoW	**5 minutes, 25 seconds**	546.297 MB

TABLE III. PERFORMANCE ON SUBSET OF INSTANCES WITH EVEN TRAIN/TEST SPLIT.

Method	Accuracy	F1-Score	AUC-ROC
MISVM	50.8	—	48.47
sbMIL	50.8	—	48.47
GICF	64.67	66.95	65.56
CAMIL D-BoW	**84.15**	**85.13**	**90.86**
CAMIL H-BoW	74.32	68.46	83.49

We wanted to compare MISVM and sbMIL to GICF and CAMIL, so we used a subset of the instances for each patient so that MISVM and sbMIL would not crash. They could only

[9] https://github.com/garydoranjr/misvm

be run on 0.1% of the reads for each patient, and even then took over 32.5 GB of memory. The results are shown in Table III. MISVM and sbMIL only achieved 50.8% accuracy. The F1 score could not be computed because there were no True or False Positives. GICF and CAMIL, while experiencing a performance drop due to the massive information loss, still performed much better, with the CAMIL methods outperforming GICF again. CAMIL D-BoW outperformed CAMIL H-BoW this time, because the distance calculations are less affected by having fewer reads than the histogram method.

MISVM and sbMIL performed the worst overall. This makes sense, as they make the standard MIL assumption, which is not helpful in the context of phenotype prediction, in which even healthy patients can host a small number of pathogens. Additionally, they are instance space methods that do not leverage bag-level information. The performance of these two methods serves to illustrate why many of the classic MIL algorithms with standard assumptions will not be effective in this domain. GICF performs better than MISVM and sbMIL, which makes sense given the fact that it follows the collective assumption. It also has the benefit of calculating instance labels, which we explore further in the next section. However, GICF is still an instance space method, so it makes sense that CAMIL outperformed it.

TABLE IV. PERFORMANCE WITH 23 PATIENT TEST SET.

Method	Accuracy	F1-Score	AUC-ROC
mRMR + SVM	80.00	81.20	82.30
CAMIL D-BoW	91.62	91.89	**98.03**
CAMIL H-BoW	**95.59**	**95.20**	97.93

Table IV compares CAMIL to the method used by Qin et al., mRMR + SVM. We initially tested CAMIL on the same 23 patient test set that Qin et al. used, for which CAMIL H-BoW had 100% accuracy and AUC, while mRMR + SVM had 0.81 AUC as reported by Qin et al. [32]. We know CAMIL is not 100% accurate, so we validated it by averaging the results of 10 independent trials in which we selected 23 patients randomly out of the 367 to serve as the test set, with the other 344 of the training set. Table IV shows that CAMIL significantly outperformed mRMR + SVM on these trials, with the H-BoW variant of CAMIL slightly outperforming the D-BoW variant. Leave-one-out cross validation generally yielded similar results to the test set results, except precision was sometimes lower.

Unlike the other MIL methods, Qin et al. use reference genomes to inform their classifier, so it makes sense that their method performs better than the MIL methods that only use de novo techniques. However, CAMIL still significantly outperformed the results reported in the MGWAS paper. We believe that the primary reason for this is that Qin et al. relied on alignments of sequences to reference genomes and attempted to select the 50 most significant genes for the phenotype before building the classifier. There are two primary problems with this approach: (i) many microbes found in the gut do not exist in reference databases and would thus be unusable for their classifier, and (ii) by only using 50 genes to inform the classifier, a lot of potentially valuable data is left out. CAMIL avoids these issues by using as much data as can be assembled and not relying on reference databases. The tradeoff is that we don't know exactly what genes are being used by the classifier to form the decision boundary. We also believe that the clustering process of putting similar contigs into groups forms useful features for the classifier.

D. *Cluster-level "Labels"*

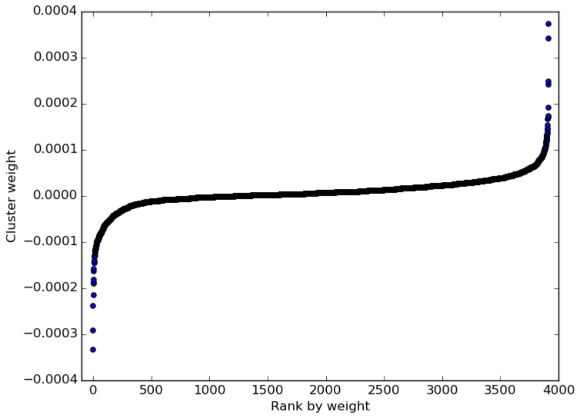

Fig. 3. This diagram illustrates the distribution of the instance weights assigned by CAMIL H-BoW. The Y-Axis shows the weights, while the X-Axis shows the ranking of the clusters by weight. Clearly, there are a relatively small number of clusters that have disproportionately large weights or small weights, while the vast majority of the 3918 clusters have weights close to 0.

In the original MGWAS paper, the authors identify 50 important gene markers with mRMR that are used for their classifier. Conversely, CAMIL uses all of the data to train and test the classifier, resulting in 3918 clusters, and subsequently identifies significant clusters based on the classification results. Figure 3 is a visual display of the cluster weights determined by CAMIL, using the 344 patient training set and 23 patient test set. Clearly, there are a few clusters with disproportionately high or low weights, while most clusters have weights near 0. Concretely, the lowest cluster weight is -0.000333, the highest is 0.000374, the mean is 0.000008, and the median is 0.000006. Intuitively, this appears to make sense, as there should be a relatively small number of key clusters whose presence is actually indicative of type 2 diabetes, while most other clusters are not particularly relevant in this case and whose weights are just noise. Thus, the weights obtained by this method appear to be plausible. In contrast, the labels obtained by GICF were barely differentiated from each other at all.

VI. CONCLUSION

We have demonstrated an effective and efficient computational pipeline for classifying patient phenotype based on metagenomic data. We have demonstrated that even relatively simple de novo assembly and clustering methods, when used within this pipeline, lead to significantly better performance results than the standard classifier used in the original Metagenome-Wide Association Study and other common MIL methods. We would like to emphasize that, while we used a particular type 2 diabetes dataset in this paper, the methods described here are general-purpose and could be used for any metagenomic dataset. We have shown how to infer the most important OTUs in the disease pathology by using the SVM decision boundary and discussed the clinical importance of this ability. More generally, we have shown the effectiveness of Multiple Instance Learning methods within metagenomics and phenotype prediction, particularly Bag of Words methods. CAMIL is a relatively simple and easy to implement pipeline that has both shown strong results and significant potential for even further improvement. Future work could revolve around improving individual parts of the pipeline, such as using better assembly and clustering methods, application of different multiple instance learning methods (other than Bag of Words), and further attempts to generate more specific instance level or cluster level information and validate that information against the known pathology of various diseases.

Acknowledgment

The authors would like to extend our gratitude towards the Office of Research Computing at George Mason University. This work was supported by NSF grant 1252318.

References

[1] P. J. Turnbaugh et al., "The human microbiome project." *Nature*, vol. 449, no. 7164, pp. 804–810, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1038/nature06244

[2] F. Backhed et al., "Host-Bacterial mutualism in the human intestine," *Science*, vol. 307, no. 5717, pp. 1915–1920, 2005. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/307/5717/1915

[3] J. Messing et al., "A system for shotgun DNA sequencing," *Nucleic Acids Research*, vol. 9, no. 2, pp. 309–321, 1981.

[4] J. Handelsman, "Metagenomics: Application of Genomics to Uncultured Microorganisms," *Microbiology and Molecular Biology Reviews*, vol. 68, no. 4, pp. 669–685, 2004.

[5] T.G. Dietterich, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1, pp. 31–71, 1997.

[6] O. Maron and T. Lozano-Perez, "A framework for multiple-instance learning," in *Advances in neural information processing systems*. Denver, CO: NIPS, July 1998.

[7] Q. Zhang and S. Goldman, "EM-DD: An improved multiple-instance learning technique," in *Advances in neural information processing systems*. Vancouver, BC, Canada: NIPS, 2001.

[8] S. Andrews et al., "Support vector machines for multiple-instance learning," in *Advances in neural information processing systems*. Vancouver, BC, Canada: NIPS, 2002.

[9] R. Bunescu and R. Mooney, "Multiple instance learning for sparse positive bags," in *International Conference on Machine Learning*. Corvallis, Oregon: ICML, 2007.

[10] 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.

[11] J. Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artificial Intelligence*, vol. 201, no. 1, pp. 81–105, 2013.

[12] G. Liu et al., "Key Instance Detection in Multi-Instance Learning," in *Asian Conference on Machine Learning (ACML)*. Singapore: ACML, November 2012.

[13] D. Kotzias et al., "From group to individual labels using deep features," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. Sydney, Australia: SIGKDD, August 2015.

[14] G. Doran and S. Ray, "A Theoretical and Empirical Analysis of Support Vector Machine Methods for Multiple-Instance Classification," *Machine Learning*, vol. 97, no. 1, pp. 79–102, 2014.

[15] R. Edgar, "Search and clustering orders of magnitude faster than blast," *Bioinformatics*, vol. 26, no. 19, pp. 2460–2461, 2010.

[16] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/22/13/1658.abstract

[17] P. Schloss et al., "Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities," *Applied and environmental microbiology*, vol. 75, no. 23, pp. 7537–7541, 2009.

[18] P. Schloss and J. Handelsman, "Introducing dotur, a computer program for defining operational taxonomic units and estimating species richness," *Applied and environmental microbiology*, vol. 71, no. 3, pp. 1501–1506, 2005.

[19] Y. Sun et al., "Esprit: estimating species richness using large collections of 16s rrna pyrosequences," *Nucleic Acids Research*, vol. 37, no. 10, pp. e76–e76, 2009.

[20] X. Hao et al., "Clustering 16s rrna for otu prediction: a method of unsupervised bayesian clustering," *Bioinformatics*, vol. 27, no. 5, pp. 611–618, 2011. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/27/5/611.abstract

[21] Z. Rasheed and H. Rangwala, "Mc-minh: Metagenome clustering using minwise based hashing," in *SIAM International Conference in Data Mining (SDM)*. Austin, TX: SIAM, May 2013.

[22] M. Bonder et al., "Comparing clustering and pre-processing in taxonomy analysis," *Bioinformatics*, vol. 28, no. 22, pp. 2891–2897, 2012.

[23] Y. Sun et al., "A large-scale benchmark study of existing algorithms for taxonomy-independent microbial community analysis," *Bioinformatics*, vol. 13, no. 1, pp. 107–121, 2011.

[24] R. Li et al., "De novo assembly of human genomes with massively parallel short read sequencing," *Genome Research*, pp. 265–272, 2010. doi: 10.1101/gr.097261.109.

[25] R. Luo et al., "SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler," *GigaScience*, vol. 1, no. 1, pp. 1–6, 2012.

[26] Y. Peng et al., "IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth," *Bioinformatics*, vol. 28, no. 11, pp. 1420–1428, 2012.

[27] D. R. Zerbino and E. Birney, "Velvet: Algorithms for de novo short read assembly using de Bruijn graphs," *Genome Research*, pp. 821–829, 2008. doi: 10.1101/gr.074492.107.

[28] T. Namiki et al., "MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads," *Nucleic Acids Research*, vol. 40, no. 20, pp. e155, 2012. doi: 10.1093/nar/gks678.

[29] S. Boisvert et al., "Ray Meta: scalable de novo metagenome assembly and profiling," *Genome Research*, 2012. doi: 10.1186/gb-2012-13-12-r122.

[30] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[31] T. Joachims, "Making Large-Scale SVM Learning Practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf and C. Burges and A. Smola, Ed. Cambridge, MA: MIT Press, 1999., pp. 41-56

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

[33] H. Peng et al., "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226-1238, 2005.

[34] W. Zhu et al., "Ab initio gene identification in metagenomic sequences," *Nucleic Acids Research*, vol. 38, no. 12, pp. e132, 2010. doi: 10.1093/nar/gkq275.

[35] J. Besemer and M. Borodovsky, "Heuristic approach to deriving models for gene finding," *Nucleic Acids Research*, vol. 27, no. 19, pp. 3911–3920, 1999.

[36] J. Qin et al., "A human gut microbial gene catalogue established by metagenomic sequencing," *Nature*, vol. 464, no. 7285, pp. 59–65, 2010.

[37] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes," *Nucleic Acids Research*, vol. 28, no. 1, pp. 27–30, 2000.

[38] S. Powell et al., "eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges," *Nucleic Acids Research*, vol. 40, no. D1, pp. D284–D289, 2012. doi: 10.1093/nar/gkr1060.