CAMIL: Pipeline for Multiple Instance Learning of Clinical Phenotype Based on Metagenomic Data

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 Distance-based 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 data, clustering of the assembled contigs, extracting features from the contigs, and using a standard 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; (ii) it is believed that most microbes have not been laboratory-cultured and thus remain unknown; (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. This can be viewed 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 a Distance-based Bag of Words (D-BoW) method, discussed further in the Background section, which has been shown to be among the most effective and efficient MIL methods [11]. We also show that our pipeline can be used for Histogram-based Bag of Words (H-BoW) methods. 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), which compares microbial metagenomic data between many patients with or without a given phenotype. Because MGWAS studies contain many expert-labeled patients and the metagenomic data associated with those patients, they are very 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 finding 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, and combining them. 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. Most of the discarded reads would not be particularly useful anyway, since the assembly process could not establish their relationship to other reads and many of them are due to sequencing machine errors. 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 both time-inefficient and 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 standard 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 a single 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 classifier against the classifier used in the MGWAS study from which we derived our data. We show that our classifier shows significantly improved performance. This is discussed further in the Experimental Results section.

The rest of the paper is organized as follows. Section 2 presents relevant Background information on Multiple Instance Learning, Assembly, and Clustering. Section 3 presents the Methods we used in the creation of our pipeline. Section 4 presents Materials, such as dataset, hardware, and software descriptions. Section 5 presents our Results, and Section 6 presents our Conclusions.

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]. In this problem, we want to figure out which of a number of different molecules will bind to a target "binding site". Each molecule can assume a number of different 3-dimensional shapes, so even for molecules that are known to bind to the binding site, it is not necessarily known which formation of the molecule succeeds in binding. If even one shape of a given molecule binds to the 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 develops 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 classic methods follow the standard assumption, which is harmful in domains in which negative bags may contain some 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 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, and he showed that this paradigm was generally significantly 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.

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; (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]. 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 [11]. The distinguishing feature in D-BoW methods is that the values for the feature vector represent the instance that has the

smallest distance to the cluster center. Histogram-based Bag of Words (H-BoW) methods count the number of instances in a given cluster instead of keeping track of the closest-matching instance to that cluster. H-BoW methods were found to be among the more effective methods, but not quite as good as D-BoW [11].

As we have discussed, 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.

B. Assembly

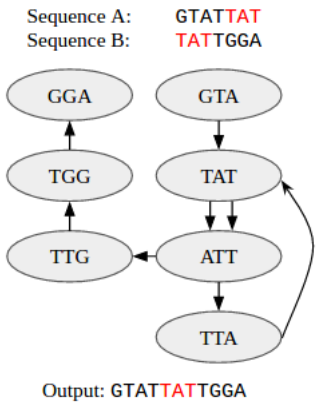

Fig. 1. This diagram illustrates a basic de Bruijn graph, as explained in the text. Notice the double arrow from TAT to ATT, allowing a path to go through that route twice. Thus, the longest contig that can be created by following a path is GTATTATTGGA, which is what we want.

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, because we wish 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 [28]. Even if full genomes cannot be assembled, combining reads into larger contigs can still make them much more useful for clustering and classifiers, 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 additionally poses its own set of challenges, due to very large datasets and lack of knowledge about how many species are present and in what relative abundances [29]. Thus, metagenome assembly is a relatively new and challenging field.

Some of the popular genome and metagenome assembly approaches include SOAPdenovo2 [26], IDBA-UD [27], Velvet [28], MetaVelvet [29], ABySS [30], and Ray Meta [31]. We used SOAPdenovo2 in our study, because it was the assembler used in the original MGWAS study [34] and because it has been shown to be one of the fastest assembly algorithms [27]. SOAPdenovo2 first constructs a type of directed graph called a *de Bruijn* graph that represents the overlaps between different sequences [25]. Reads are divided into strings of length K called *k-mers*; these k-mers are the nodes of the graph [28]. The choice of K is up to the user, and is important for having good assembly results. The k-mer nodes in the graph have an edge between them if a read contains those k-mers in order with an overlap of K-1 nucleotides, and the direction of the edge indicates in which order the k-mers appear [28]. SOAPdenovo2 then cleans this graph by removing nodes/sequences with few or no connections with other sequences, eliminating "tips" that represent likely sequencing machine errors, and removing redundant edges [25]. The contigs are then formed by combining reads according to the de Bruijn graph: each contig represents a directed path in the graph [28]. There are variations on how exactly de Bruijn graphs are represented and implemented, but Figure 1 illustrates the general process.

C. Clustering

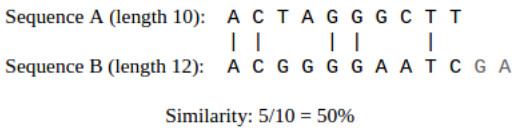

Fig. 2. This diagram illustrates the similarity computation between two sequences in UCLUST. Terminal characters are excluded, so the length of the reads is 10. In the first 10 characters of each string, there are 5 matching characters that are in the same position in the string. Thus, the similarity is 5/10 = 50%.

The *clustering* problem in this context involves grouping input short reads such that reads 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 detail is important, because it is believed that most microorganisms 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. Finally, clustering helps the classification process by allowing feature vectors to be built at the OTU level, instead of using individual short reads that contain little biological information.

UCLUST [15], CD-HIT [16], mothur [17], DOTUR [18], CROP [21], MC-MinH [22], and MC-LSH [20] are some of

the popular nucleotide sequence clustering approaches. 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 [23], [24].

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]. Thus, each *centroid* defines the center of a cluster, and the distance T defines the radius of the cluster, such that any point that has a similarity of greater than T to the centroid is within the radius and is thus part of the cluster. UCLUST is a heuristic algorithm that has several optimizations to improve speed, thus condition (i) above is not always guaranteed [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]. This is illustrated in figure 2.

III. Methods

A. Overview

As mentioned in the Introduction, our 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 reads that could not be assembled were unlikely to be useful anyway, since they could not be assembled with other reads into useful contigs. 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 applied our feature extraction method, which uses the D-BoW method of measuring the closest distance from a patient's reads to each cluster center. Finally, we used a standard SVM 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 the results section. Aside from the patient labels, this process is entirely de novo, and does not consult any external databases. An illustration of this pipeline can be seen in Figure 3 on page 5.

B. Assembly with SOAPdenovo2

For our assembly step, we used SOAPdenovo2 [26], which is a de novo metagenome assembly tool. This tool was also used for assembly in the MGWAS study that our data came from. We use this method in order to compare ourselves with the MGWAS study, and because Soapdenovo2 has been proven to be one of the fastest assemblers[27]. 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 [34]. 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 used UCLUST [15], which has been shown to be among the simplest, fastest, and most effective clustering methods [23], [24]. Since our contigs were not ordered, we used the usersort option, and we set the sequence match id 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. Again, this value of 40% was found to be the best value based on our experiments with this dataset.

D. Feature Extraction

Our feature extraction operation followed the D-BoW method. For each patient, we initialized a vector with length equal to the number of clusters. The value for each place in that patient's feature vector would then be set to the closest percentage match (0.4 to 1) of the reads from that patient to the relevant cluster seed. For instance, if there are three clusters of reads, and patient A has the cluster seed for cluster 1, no reads from cluster 2, and a read with 70% match to cluster 3, their feature vector would be A = [1.0 0.0 0.7]. This is illustrated in Figure 3. We implemented this feature extraction method in Python.

E. Classification and Evaluation Metrics

We performed classification with a standard SVM classifier using the generated feature vectors; in this case, we used svm-light [33]. The choice of classifier is not very important for D-BoW methods [11].

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:

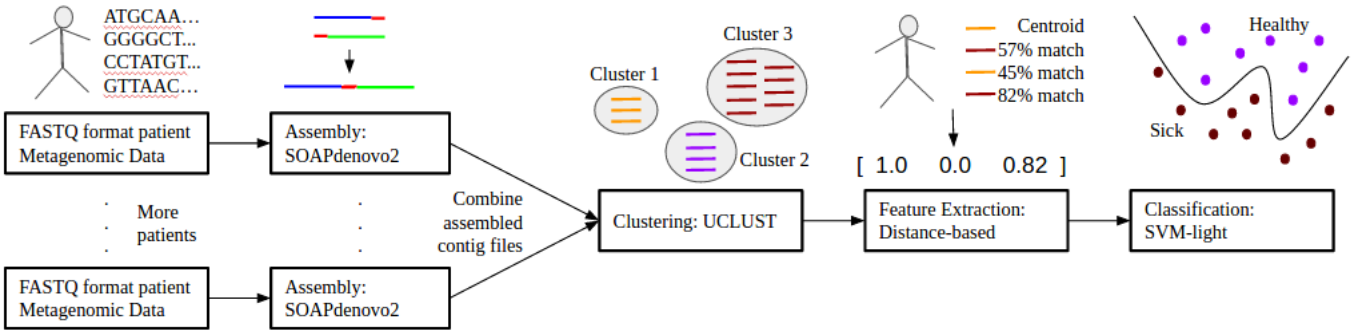

Fig. 3. This diagram illustrates the entire pipeline. Patient files with FASTQ metagenomic reads are individually assembled using SOAPdenovo2, then combined into one file and clustered with UCLUST. We extract features according to the D-BoW method, and classify the patients using the feature vectors with svm-light.

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

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

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:

$$F1{-}Score = 2{*}(Precision{*}Recall)/(Precision{+}Recall). \quad (4)$$

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 versus False Positive Rate, given by:

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

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

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 [34]. 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 the largest among those studies. The full dataset used in this study contains 368 patients [34], but only 218 of them were labeled and available online. Each patient file was downloaded from NCBI[1] and converted to FASTQ format using the SRA toolkit[2]. The labels were found on the EBI website[3]. The total size of these 218 FASTQ files was 2.3 terabytes, with an average size of 10.82 gigabytes per patient file. Qin et al. develop a simple T2D classifier using a minimum redundancymaximum relevance (mRMR) method [35] for feature selection and an SVM for classification, based on the R packages "sideChannelAttack" and "1071", respectively [34]. They achieved an AUC-ROC of 0.81 by training a classifier on 345 of the patients and using the remaining 23 as a test set [34]. While we did not have access to all of the same patients that they had, we were able to replicate their method and test it on the subset of data that was publicly available. The study also called for more extensive testing of gut microbiota classifiers [34].

B. 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] [26], UCLUST[6] [15], and svm-light[7] [33] their respective websites and compiled on the Argo platform.

V. Experimental Results

A. Experimental Setup

The 218 publicly-available and labeled patients were downloaded from NCBI and converted to FASTQ format using the SRA toolkit, and labeled according to the labels on EBI, as described in the Materials section. The data was split evenly into training and test sets, with each containing 109 patients. The training set had 54 T2D patients and 55 control patients, while the test set had 53 T2D patients and 56 control patients. Each read was assembled with SOAPdenovo2, with the k-mer length set to 51, reads cut off after 100 base pairs (original length of 180 base pairs), and the average insert size set to 350 in accordance with the reported average insert size reported in the MGWAS study [34]. Assembly for each file took 7-30 minutes, depending on the file size, and each patient was assembled in parallel. Combining the files into one file took 12 minutes. Assembly was used for all of the classification

[1]http://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP011011
[2]http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software
[3]https://www.ebi.ac.uk/metagenomics/projects/SRP011011
[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/

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 instances instead of clusters. In order to do this, sequence reads were represented as counts of k-mers. k-mers are nucleotide strings of length k. Since there are four possible nucleotides, the number of possible k-mers is 4^k. For instance, if k=3, there are 4^3 = 64 possible k-mers, which are AAA, AAC, AAG, AAT, ACA, ACC, ..., TTT. For a string "ATACGATA", the count for the k-mers is 2 for ATA, 1 for TAC, ACG, CGA, and GAT, and 0 for everything else. We wrote a script to represent the reads as vectors representing the k-mer counts in the string, and these vectors were used by the other methods. From experimental validation, we found a k-mer value of 3 to be the most effective. Converting the reads from their original string form to k-mer vectors took 3 hours, 17 minutes, and 55 seconds.

1) CAMIL - Our Pipeline: We implemented two different versions of the pipeline: 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 10 hours and 51 minutes. Without the assembly step reducing the size of the data, clustering would not have been feasible. Feature extraction and SVM-light classification took 10 minutes and 17 seconds to run for D-BoW, versus 5 minutes and 14 seconds for H-BoW.

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 assumption. For the implementation of these methods, we used an open-source Python implementation by Doran [14], which is available on GitHub[8].

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 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.

4) Original MGWAS Paper: The methods in the original paper are neither MIL-based, nor are they entirely de novo apart from patient labels. The authors first performed de novo assembly with SOAPdenovo2 [26] and then use a tool called MetaGeneMark [36], [37] for de novo prediction of genes from the assembled contigs [34]. They then combined these genes with an existing gene catalog, MetaHIT [38], and carried out taxonomic assignment and functional annotation of the genes using the KEGG [39] and eggNOG [40] databases, as well as 2,890 other reference genomes [34]. 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) [35] 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 [34]. 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. We manually computed the accuracy and F1 score with an ideally-chosen decision boundary based on the results the authors provided in their supplementary tables.

C. Results For Bag/Patient Labels

TABLE I. COMPARISON OF THE PERFORMANCE OF VARIOUS METHODS ON THE MGWAS DATASET.

Method	Accuracy	F1-Score	AUC-ROC
MISVM	50.00	50.00	50.00
sbMIL	50.00	50.00	50.00
GICF	60.00	71.43	66.00
Original	82.61	80.02	81.00
CAMIL H-BoW	91.74	91.43	95.00
CAMIL D-BoW	**92.66**	**92.31**	**96.00**

Table I shows the comparison of classification performance between the various methods described above. CAMIL methods significantly outperform the other methods, with the D-BoW variant of CAMIL slightly outperforming the H-BoW variant.

MISVM and sbMIL had the worst results, faring no better than a random guess. This makes sense, as they make the standard MIL assumption, which doesn't make sense 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

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

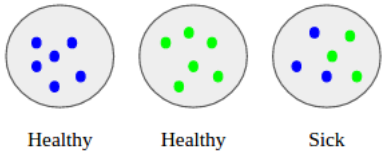

Fig. 4. This diagram illustrates why static instance labels are not sufficient for phenotype prediction. A patient with 6 of the blue microbe or 6 of the green microbe may be healthy, while a patient with 3 of each is sick. Static instance labels cannot capture this relationship. This is also explained by Amores in his MIL taxonomy [11].

an instance space method, so it makes sense that it performs worse than CAMIL.

The method used in the original paper is the only one tested here that is not an MIL method, and the only one that is not entirely unsupervised apart from the patient labels. Given that the methods from this paper were not de novo, it makes sense that they would outperform many of the unsupervised MIL methods. However, CAMIL still significantly outperformed the results reported in the original paper. We believe that this is due to the following reasons: (i) the clustering process puts similar contigs into groups that are useful features for the classifier, and (ii) instead of attempting to select the most significant features before performing classification, we allow the classifier itself to determine the most significant features from the entire pool of features.

D. Deriving Instance "Labels"

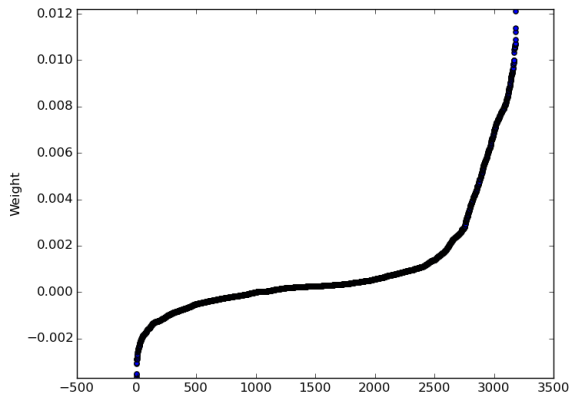

Fig. 5. This diagram illustrates the distribution of the instance weights assigned by CAMIL D-BoW. The Y-Axis shows the weights, while the X-Axis shows the ranking of the clusters by weight. Clearly, there are a relatively small number of clusters that have disproportionately large weights, while there is a smaller number of clusters with smaller weights.

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 is illustrated in Figure 4. This example is very simplified, but it explains why static instance labels are insufficient.

However, 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 Ith 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.

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, and subsequently identifies significant clusters based on the classification results. We printed the 25 highest-weighted and 25 lowest-weighted clusters, out of 3188 total clusters. The highest-weighted cluster had a weight of 0.012109, while the lowest-weighted cluster had a weight of -0.003668. The average of the 25 highest cluster weights was 0.010402, while it was -0.002753 for the lowest 25. The average of all 3188 cluster weights was 0.001003, and the median was 0.000254. These results indicate that the highest weights were much larger than the lowest weights and skewed the average weight to be much higher than the median weight. 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 aren't 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. The instance weights determined by CAMIL are shown in Figure 5.

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. 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 Distance-based 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 better assembly and clustering methods,

application of different multiple instance learning methods (other than D-BoW), 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.

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] Z. Rasheed et al., "Lsh-div: Species diversity estimation using locality sensitive hashing," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. Philadelphia, USA: IEEE, October 2012, pp. 1–6, acceptance Rate: 59/299 = 19.93%.

[21] 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

[22] 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.

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

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] J. T. Simpson et al., "ABySS: A parallel assembler for short read sequence data," *Genome Research*, pp. 1117–1123, 2009. doi: 10.1101/gr.089532.108.

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

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

[33] 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

[34] 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.

[35] 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.

[36] 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.

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

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

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

[40] 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.