Multiple Instance Learning of Clinical Phenotype Based on Metagenome-Wide Association Study

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 metagenome 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*. 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.

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 [4]. 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 [23]. 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 domains 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. This data is very large (hundreds of gigabytes) 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. 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 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 discusses 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. 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 [18]. 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 [18]. 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 all negative instances. 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 [18]. In the late 1990s and early 2000s, a number of different approaches were developed for the original MIL problem, such as Diverse Density (DD) [19], EM-DD [20], MI-SVM [21], and MILES [22]. A recent review of MIL by Amores created a taxonomy of these various methods and compared their effectiveness for classification [23].

However, since 2010, there has been increased interest in different formulations of the MIL problem. For instance, the problem of "key instance detection" [24] 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 [25]. 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.

However, this method treats a group label simply as an aggregation of instance labels. Amores, in his taxonomy, called this type of method "Instance-based", and found this paradigm to be ineffective [23]. Amores identifies two other paradigms, Bag-Space and Embedded-Space. An example of the latter 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 [23]. 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 [23]. 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. Despite these recent developments in MIL, we have not found any literature that specifically applies MIL to classifying patient phenotype based on metagenomic data.

B. Assembly

The *assembly* problem involves combining overlapping short reads into longer sequences called *contigs*.

C. Clustering

The *clustering* problem in this context involves grouping input short reads such that reads within a group are similar to each other. This process may lead to groups that are organism-specific and is unsupervised in nature. The clustering process does not attempt to provide an automated labelling of the input reads. The clusters/groups obtained from an input metagenome sample are referred to as Operational Taxonomic Units (OTUs), and the number of OTUs gives an approximation of species diversity in a sample [9], [10], [11]. These approaches are not constrained due to the absence of a complete coverage in taxonomic databases. Some environmental samples contain microbial organisms that have never been cultured in a laboratory, and thus those organisms do not exist in genomic databases. As such, clustering of sequence reads has several advantages: (i) it can lead to an improved metagenome assembly, (ii) it can be used for computing species diversity metrics [12] and (iii) the reduced computational complexity within several work-flows that analyze only cluster representatives, instead of individual sequences within a sample.

CD-HIT [13], UCLUST [8], CROP [14], MC-MinH [15] and MC-LSH [12] are some of the popular metagenome sequence clustering approaches used for binning. UCLUST, MC-MinH and MC-LSH are greedy approaches that achieve computational efficiency by using either hash-based indexing or matching of gap-less sequences called seeds (instead of expensive sequence alignment) and followup with an incremental clustering approach that does not involve comparing all pairs of input sequences. 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 [35], [36].

UCLUST [8] follows a greedy, iterative clustering approach. As a first step, this approach identifies exact matches of fixed length between sequence pairs known as seeds. These seeds are then extended by allowing for a few mismatches

and/or gaps between the aligned pairs. Seeds scoring above a certain threshold are chosen as high segment pairs (HSPs) and used for further processing. As such, the step eliminates a lot of pairwise comparisons. Then UCLUST follows an incremental approach for assigning the input sequences to the different bins. The clustering solution is initialized as an empty list. Sequences are then incrementally added to the clusters existing within the list. Each unassigned input sequence is compared to the cluster representatives within the list using the fast indexing search technique that uses the HSPs. If a match is found with one of the cluster representatives, then that sequence is assigned to that particular cluster; otherwise, the input sequence forms a new cluster. Matches are computed by comparing two reads, character by character, and dividing the number of characters at the same position in both reads that match by the length of the longer of two reads. UCLUST ensures that the cluster representatives are sequences with the largest length.

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 1 on page 4.

B. Assembly with SOAPdenovo2

For our assembly step, we used SOAPdenovo2 [32], 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[39]. 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 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 [8], which has been shown to be among the simplest, fastest, and most effective clustering methods [35], [36]. 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 1.

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 [23].

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:

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

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

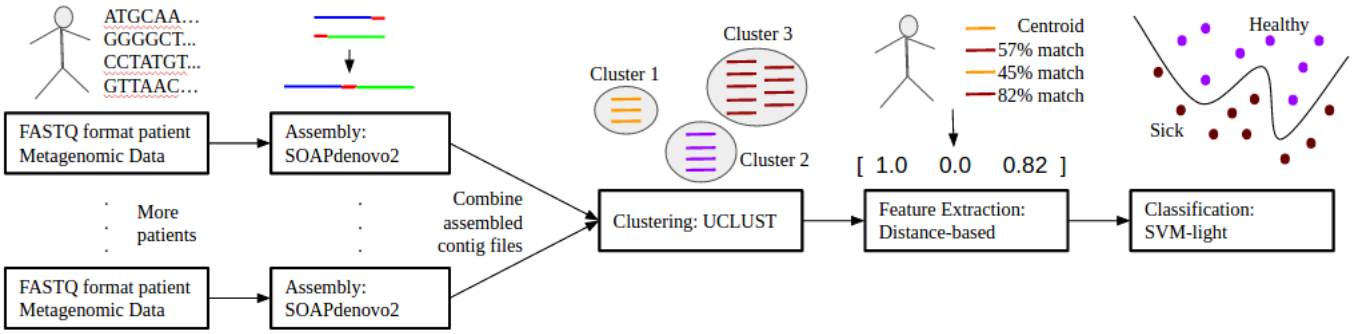

Fig. 1. This diagram illustrates the entire pipeline. Patient files with FASTQ metagenomic reads are individually assembled using SOAPdenovo2, then combined into one file and clustered with UCLUST. We extract features according to the D-BoW method, and classify the patients using the feature vectors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

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 [31]. The total dataset used in this study contains 368 patients [31], 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]. Each patient file was several gigabytes in size, with the total dataset size being over 600 gigabytes. Qin et al. develop a simple T2D classifier using a minimum redundancymaximum relevance (mRMR) method [30] for feature selection and an SVM for classification, based on the R packages "sideChannelAttack" and "e10171", respectively [31]. 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 [31]. 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 [31].

B. Software and Hardware Details

We used the ARGO computing cluster available at George Mason University [40]. 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) [40]. Source codes for SOAPdenovo2[4] [32], UCLUST[5] [8], and svm-light[6] [33] their respective websites and compiled on the Argo platform.

V. Experimental Results

A. Data, Software, and Hardware Used

Our main dataset comes from a GWAS of Type 2 Diabetes (T2D) in Chinese adults, published by Qin et al [31]. The data is publicly available on the National Center for Biotechnology Information (NCBI) [37] website using the accession number SRP011011. The full dataset contains over 200 patients, but for our study we used a subset of 60 patients due to lengthy download and format conversion times. Of these 60 patients, 30 had T2D and 30 were controls. For the purpose of testing our cluster sampling methods, we also used a smaller dataset obtained from Dr. Patrick Gillevet of GMU, which is not publicly available. It contains 904 patients, 239 of whom have encephalopathy and 665 of whom are controls, but is much smaller because it uses 16S rRNA instead of whole genome data. The encephalopathy dataset contains about 1.3 million reads, while the T2D dataset contains about 1.3 billion reads.

As previously mentioned, we use the publicly-available UCLUST [8] software for clustering. For classification, we use Support Vector Machines (SVMs) [17], one of the most powerful and versatile binary classifiers used in myriad applications. For a binary classification problem, the SVM framework maximizes the separation between the two classes. In addition to the ubiquity and effectiveness of SVMs, they are well established in both supervised and multiple instance learning, making it easy to compare the two approaches. For our standard supervised SVM, we use svm-light [33], a popular and publicly available implementation. For multiple instance learning, we use the python implementation by G. Doran and S. Ray ("MISVM") [38] for its ease of use and public availability.

B. Efficient Unsupervised Clustering using UCLUST

From early experimental results, UCLUST can still be computationally infeasible for very large datasets such as the

[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]SOAPdenovo2: http://soap.genomics.org.cn/soapdenovo.html
[5]UCLUST: http://www.drive5.com/uclust/downloads1_2_22q.html
[6]svm-light: http://svmlight.joachims.org/

one we will be using, despite its generally excellent speed. For our T2D dataset, UCLUST was only able to cluster 2.8% of the reads within 12 hours, at which point ARGO killed the batch job. UCLUST slows down as more clusters are formed, as each new read needs to be compared to more existing clusters. Thus, as the number of clusters grows, the execution time of UCLUST increases rapidly. The only way to feasibly complete the clustering phase in a reasonable amount of time was to cluster the patients individually in parallel, instead of clustering the reads of all patients together. However, this leads to the problem of having mismatched clusterings between patients, that is, cluster X from patient A may represent the same OTU as cluster Y from patient B. The solution is to compare the clusters from different patients and map them to each other, and then to combine clusters from different patients that represent the same OTU and output a single clustering output file for all patients. However, comparing every read from each cluster to every read from all other clusters is also computationally infeasible. Thus, we developed methods to sample representative reads from each patient and use them to create the consensus clustering. We explored a number of different methods, but two of them in particular were able to achieve significant speed improvements without significantly harming classification results. For each of them, we used a parameter "alpha", which was used to select 1 out of every alpha reads. For instance, if alpha=10, 1/10 of reads would be used.

We refer to one of our methods as "seed re-clustering" (UCLUST refers to cluster centroids as "seeds"). In this method, the reads corresponding to the cluster seeds from each patient are copied into a temporary file, and then those reads are clustered. This runs quickly, since clusters in this setting tend to have hundreds to thousands of reads, so the number of cluster seeds is relatively small. We then construct a map, which maps each seed from its original cluster number to its new cluster number from the seed clustering. We go back to the individual patient clustering output, and select 1 out of every alpha lines to be placed in the consensus clustering file. As we write the lines to the consensus file, we use the map we developed to change the cluster number for the line from the original clustering number to the new consensus cluster number. We refer to the other method as "two round clustering". We select one out of every alpha reads from the clustering output of each patient, place them in a single file, and run UCLUST on this smaller subset to arrive at a consensus clustering. The main difference between two round clustering and seed re-clustering is that here we select a distribution of reads from the original clustering to determine the consensus clustering, instead of cluster seeds/centroids.

We sought to empirically evaluate how these methods affected clustering runtime and classification accuracy, since feature vectors are generated from the clustering output and thus classification results are affected by the clustering methods. Without use of these sampling methods, UCLUST didn't finish within ARGO's 12 hour time limit on the T2D dataset, so we couldn't evaluate the classification results on that dataset and compare them to the classification results with the clustering sample methods. In order to evaluate these methods, we used the aforementioned Encephalopathy dataset, which is much smaller. We measured the time it took to perform UCLUST on all patients at once, as well as the time taken by each of our two sampling methods. We then used the respective clustering outputs to generate feature vectors and run svm-light classification (as described in the next section) and compared the results, which are discussed in the "Experimental Results" section. For UCLUST, default settings were used, except the threshold for placing two sequences in the same cluster was set to 40% match (–id 0.40), and the –usersort option was used.

C. Supervised and Multiple Instance Classification using Support Vector Machines (SVMs)

For our formulation we use SVMs to make clinical phenotype prediction for each patient sample based on their input metagenome sequence samples represented with their clustering or taxa membership features. With our clustering approach, we have a set of reads that have been assigned to clusters based on UCLUST and our techniques. We now generate feature vectors from each patient from this clustering output. We follow a different process of feature vector generation for svm-light and MISVM. For each method, we read through the consensus clustering output from UCLUST line-by-line, forming feature vectors as we go. UCLUST output lines provide the cluster number, the percentage match to the cluster seed, and the patient that read comes from.

For svm-light, we have one feature vector per patient. We use each cluster/OTU as a feature, meaning that the presence of various microbes are the features for the patient. Each number (feature) in the vector corresponds to a cluster/OTU, and the value for that number corresponds to how closely the read from the patient matched the cluster seed. For each input sequence, we use the percentage match to its cluster seed as the value for that feature and average the sequence match if multiple reads from the metagenome samples are assigned to the same cluster by UCLUST. Each sequence read includes a label for the patient that it came from, so each read only affects the feature vector corresponding to the patient that it came from. For example, say patient A has a read with 95% match to OTU 2, a read that is the cluster seed of OTU 3, a read that is a 93% match to OTU 2, a read that is a 97% match to OTU 5, and no reads for OTU 1 or 4. Patient A's feature vector would then look like: [0 0.94 1 0 0.97]. We first initialize the patient feature vectors to all 0s and then iterate through the UCLUST output lines and update the feature vectors as we go. For MISVM, patients are bags, reads for patients are instances, and each read/instance has a feature vector. We iterate through the UCLUST output. For each read, we add that read as an instance to the relevant patient bag. The feature vector for that read contains one number: the cluster number that the read belongs to. We compared the runtime and accuracy of the two implementations, which both used a linear kernel. In both cases, the training and test set each contained 30 patients, and each contained 15 patients with type 2 diabetes and 15 control (healthy) patients. The results are discussed in the "Experimental Results" section.

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. Finally, we have shown the effectiveness of Multiple Instance Learning methods within metagenomics and phenotype prediction, particularly Distance-based Bag of Words methods. 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 labels and validate those labels against clinically verified labels.

ACKNOWLEDGMENT

The authors would like to extend our gratitude towards the Office of Research Computing at George Mason University.

REFERENCES

[1] P. J. Turnbaugh, R. E. Ley, M. Hamady, C. M. Fraser-Liggett, R. Knight, and J. I. Gordon, "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, R. E. Ley, J. L. Sonnenburg, D. A. Peterson, and J. I. Gordon, "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, R. Crea, and P. H. Seeburg, "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] E. K. Costello, C. L. Lauber, M. Hamady, N. Fierer, J. I. Gordon, and R. Knight, "Bacterial Community Variation in Human Body Habitats Across Space and Time," *Science*, vol. 326, no. 5960, pp. 1694–1697, 2009. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/326/5960/1694

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

[7] C. R. Woese and G. E. Fox, "Phylogenetic structure of the prokaryotic domain: the primary kingdoms." *Proc Natl Acad Sci U S A*, vol. 74, no. 11, pp. 5088–5090, 1977.

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

[9] P. Schloss, S. Westcott, T. Ryabin, J. Hall, M. Hartmann, E. Hollister, R. Lesniewski, B. Oakley, D. Parks, C. Robinson *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.

[10] 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.

[11] Y. Sun, Y. Cai, L. Liu, F. Yu, M. Farrell, W. McKendree, and W. Farmerie, "Esprit: estimating species richness using large collections of 16s rrna pyrosequences," *Nucleic Acids Research*, vol. 37, no. 10, pp. e76–e76, 2009.

[12] Z. Rasheed, H. Rangwala, and D. Barbara, "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%.

[13] 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

[14] X. Hao, R. Jiang, and T. Chen, "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

[15] 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.

[16] D. R. Cox, "The Regression Analysis of Binary Sequences," *Journal of the Royal Statistical Society*, vol. 20, no. 2, pp. 215–242, 1958.

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

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

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

[20] 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.

[21] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Advances in neural information processing systems*. Vancouver, BC, Canada: NIPS, 2002.

[22] Y. Chen, J. Bi, and J. Wang, "MILES: Multiple-instance learning via embedded instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 1931–1947, 2006.

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

[24] G. Liu, J. Wu, and Z. Zhou, "Key Instance Detection in Multi-Instance Learning," in *Asian Conference on Machine Learning (ACML)*. Singapore: ACML, November 2012.

[25] D. Kotzias, M. Denil, N .De Freitas, and P. Smyth, "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.

[26] P. Li, A. Shrivastava, J. Moore, and A. Konig, "Hashing algorithms for large-scale learning," in *Advances in neural information processing systems*. Granada, Spain: NIPS, December 2011.

[27] P. Li and C. Konig, "b-Bit minwise hashing," in *ACM International World Wide Web Conference (WWW)*. Raleigh, NC: WWW, April 2010.

[28] A. Broder, S. Glassman, M. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, no. 8, pp. 1157–1166, 1997.

[29] A. Broder, "On the resemblance and containment of documents," in *IEEE International Conference on Compression and Complexity of Sequences (SEQUENCES)*. Positano, Solerno, Italy: SEQUENCES, June 1997.

[30] H. Peng, F. Long, and C. Ding, "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.

[31] 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.

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

[33] Svmlight support vector machine. [Online]. Available: http://svmlight.joachims.org/

[34] Weka3: Data Mining Software in Java. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[35] M. Bonder, S. Abeln, E. Zaura, and B. Brandt, "Comparing clustering and pre-processing in taxonomy analysis," *Bioinformatics*, vol. 28, no. 22, pp. 2891–2897, 2012.

[36] 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.

[37] National Center for Biotechnology Information. [Online]. Available: http://www.ncbi.nlm.nih.gov/

[38] 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, 2013. Code available online at: https://github.com/garydoranjr/misvm/

[39] Y. Peng, H. C. M. Leung, S. M. Yiu, and F. Y. L. Chin, "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.

[40] On-campus research computing. [Online]. Available: http://orc.gmu.edu/research-computing/argo-cluster/argo-hardware-specs/