Intelligent Automation and Soft Computing, 2015
http://dx.doi.org/10.1080/10798587.2015.1073887

LEVERAGING CLUSTERING TECHNIQUES TO FACILITATE METAGENOMIC ANALYSIS

DAMIEN ENNIS, SERGIU DASCALU, AND FREDERICK C. HARRIS JR.
Department of Computer Science and Engineering, University of Nevada, Reno, NV, 89557, United States

ABSTRACT—Machine learning clustering algorithms provide excellent methods for conducting metagenomic analysis with efficiency. This study uses two machine learning algorithms, the self-organizing map and the K-means algorithms, to cluster data from an environmental sample collected from a hot springs habitat and to provide a visual analysis of that data. A data processing pipeline is described that uses the clustering algorithms to identify which reference genomes should be included for further analysis in determining possible organisms that are present in a metagenomic sample. The clustering revealed probable candidates for additional analysis, including a thermophilic, anaerobic bacterium, which is likely to be found in a hot springs environment and serves to validate the functionality of these tools. The machine learning techniques discussed here can serve as a launching point for elucidating protein sequences that could serve as possible reference comparisons to a specific metagenomic sample and lead to further study.

Key Words: Machine learning; Metagenomics; Self-organizing map; K-means; Clustering

1. INTRODUCTION

Metagenomics, a relatively new field within biology, involves the classification of DNA sequence fragments from environmental samples consisting of diverse microbial populations. Research in metagenomics focuses on both the classification of organism types and the functions of sequences that are present in environmental samples. The latter method essentially treats the metagenome as a single genomic item, which can provide information about metabolic processes within the habitat (Handelsman, Rondon, Brady, Clardy, & Goodman, 1998). Because metagenomics research involves vast amounts of data, its analysis calls for the use of tools that can facilitate the process with efficiency. This type of large-scale analysis of genomic data falls within the interdisciplinary field of bioinformatics, which is a merging of molecular biology, probability, and computer science. Clustering techniques within the field of machine learning provide excellent methods for analyzing metagenomic data.

A number of approaches have been used to analyze DNA sequences in metagenomic data, including sequence similarity searches, sequence composition methods, and phylogenetic methods (Bazinet & Cummings, 2012). A Basic Local Alignment Search Tool (BLAST) search provides one type of similarity comparison, wherein a query sequence is compared against a library of known sequences in order to find similar sequences that fall within a designated threshold limit (Altschup, Gish, Miller, Myers, & Lipman, 1990). An example of a sequence composition method is a Markov model, which is a probabilistic-based model of the composition of the likely DNA sequence. A Markov chain model assigns the current nucleotide base a probability based on the composition of the preceding nucleotides (Kelley & Salzberg, 2010). Phylogenetic techniques use an evolutionary relationship model to locate where the query sequence

*Correspondence should be addressed to Damien Ennis, Truckee Meadows Community College, 7000 Dandini Blvd, Vista 200C, Reno, NV 89512, dennis@tmcc.edu most likely fits. Applications may even employ a combination of these techniques to perform sequence classification.

Several studies have investigated methods for better facilitating the analysis of metagenomic data; these methods generally focus on either identifying genomic sequences or elucidating the functionalities present within a sample (Culligan, Sleator, Marchesi, & Hill, 2014; Chistoserdova, 2009). Ghosh, Mohammed, Rajasingh, Chadaram, and Mande (2011) developed a method for rapidly determining sequences in a metagenomics habitat. Their technique, called HabiSign, focuses on finding habitat-specific oligonucleotide usage patterns to differentiate between metagenomes, which have variations at phenotypic, species, and biome levels. Abubucker et al. (2012) developed a method, which uses short DNA reads to analyze both the presence and abundance of microbial communities within the host environment. Their system, HMP Unified Metabolic Analysis Network (HUMAnN), profiles the metabolic potential of microbial communities by using a series of steps that involve gene and pathway-level quantification, noise reduction, and smoothing of data.

Clustering methods are also used to facilitate metagenomic analysis (Bazinet & Cummings, 2012; Li, Fu, Niu, Wu, & Wooley, 2012). Clustering is the process of grouping “like” pieces of data into one of several bins, which is often based on overall similarity (Euclidian distance or some other metric), but other qualities could also serve as the guideline for this binning process. The goal of a clustering algorithm is to group raw (unlabeled) data on the basis of some underlying features that are not readily apparent. Each cluster represents a collection of like items (data) with similar features, whereas some implied dissimilarity exists between items in different clusters. This can be used to directly elucidate new patterns in complex data, but may also serve to reduce the scope of subsequent analysis (Aggarwal & Reddy, 2013; Murphy, 2012; Liao et al., 2013). One clustering method used for metagenomic analysis is CD-HIT, which orders genomic sequences by length and then assigns the longest ones as cluster seeds. It then compares the reads against the existing clusters by using a greedy incremental algorithm, a type of algorithm that makes the local optimal choice at each decision point (Li & Godzik, 2006). CD-HIT is less computationally intensive than several other methods that compare every sequence against all the others present in the sample. UCLUST, another greedy algorithm-based method, differs from CD-HIT in that it uses heuristics to facilitate faster sequence comparisons (Edgar, 2010).

The field of machine learning offers clustering techniques that have not yet been widely applied in metagenomics research. Machine learning clustering techniques use computer algorithms that adapt to the data presented, and they can be categorized into two varieties: Supervised and unsupervised. A supervised algorithm relies upon previously evaluated data to create a classification system, whereas an unsupervised algorithm can be used to group unknown data into collections of similar items (Marsland, 2011). Thus, machine learning techniques can effectively cluster extremely large quantities of data with some or no training, a quality which makes them well-suited to metagenomic analysis.

Two types of unsupervised algorithms that offer great potential for metagenomic analysis are the self-organizing map (SOM) and the K-means algorithms. Both techniques have been successfully applied in other fields but have had only limited application in metagenomics. Of note, however, is a study by Weber et al. (2011), which employed a machine learning algorithm in the analysis of five metagenomes of medium complexity. The authors created an application, called TaxSOM that uses a self-organizing map to plot unknown genomic signatures against known types in order to provide taxonomical classification.

The work described in this paper applies the SOM and K-means algorithms to the analysis of metagenomic data collected from a sample from the Great Boiling Spring habitat in the state of Nevada in the United States. The typical approach to metagenomic analysis centers on taxonomical classification, generally proceeding in one of two directions: Either by comparing the complete sequences or by focusing on the 16sRNA genes present in the sample, which comprise the DNA sequence that specifies the small ribosomal subunit and are often used for species identification (Ghosh et al., 2011). The approach taken here uses a variation of the former. This work attempts to identify possible candidates to serve as reference genomes for use in further analysis. A reference genome is a documented collection of nucleotide sequences and protein products from a specific species, often documented from multiple samples. These sequences, which are often annotated by domain experts, can serve as points of comparison to unknown or novel sequences.

This paper attempts to answer the question: Can the use of machine learning clustering techniques provide a guideline as to which reference genomes should be included for further analysis in determining possible organisms that are present in a metagenomic sample? This paper does not attempt to directly taxonomically classify organisms present in the sample, but rather "pre-bins" the data so as to provide a method to streamline future analysis. The unsupervised techniques discussed here can serve as a launching point for elucidating protein sequences that could serve as possible reference comparisons to a specific metagenomic sample and lead to further study, possibly by using supervised learning or even domain-expert analysis.

2. CLUSTERING TECHNIQUES

Because unsupervised machine learning algorithms like the SOM and the K-means are especially adept at clustering large datasets, they are well-suited for use in the field of metagenomics. An SOM synthesizes complex data that consist of many features, clusters the data on the basis of these similar features, and produces a visualization of the clusters in the form of a map. The resulting map output presents a simplified view of the data in a regular grid or lattice of nodes. Input data items with similar features are placed in close proximity on the grid, thus allowing viewers to easily discern items that cluster together (Kohonen, 1998, 2001). The K-means algorithm employs a pre-determined number of clusters (i.e., bins) and iteratively assigns data to the cluster that most closely matches itself. While techniques exist for mitigating the need to pre-select the number of clusters (Lai, 2005), the K-means algorithm was used in this study, because it provided a way to efficiently categorize the data. The clusters themselves are adapted to match the data as it is assigned. The result of applying K-means is that all inputted data vectors are assigned to a specific bin (Hartigan, 1975; Jain, 2010). Because of their similar domain applicability, the SOM and the K-means algorithms both provide a suitable approach to the problem addressed here, and each provides a natural validation of the performance of the other.

2.1 Self-Organizing Map

An SOM produces an output that serves to reduce the dimensionality of input data. As such, the most prevalent configuration is that of a two-dimensional map, which is easily human-readable. Unlike some other types of artificial neural networks, (which require known input-to-expected-output pairings to serve as the basis for training and parameterization of the network), an SOM is a competitive learning algorithm which does not use pre-existing information to aid in the determination of outputs. Instead, the algorithm is used to find similarities between different inputs, and the data itself serves to guide the structure and weights of the set of connections of neurons. These similarities are visualized through the resulting map, a common configuration of which is a rectangular grid of neurons (Kohonen, 2001).

Each neuron has an associated weight vector, which has the same dimensionality as the input data. The neuron weight vectors are adjusted by the collected numerical data from the input points. Data which are close (Euclidian distance) in the original dimensionality will be placed in close proximity within the completed map. Each neuron is connected to the input layer through its associated weight vector. Weight vectors are initialized with random values within a specified range (Kohonen, 2001). The N-dimensional weight vector is defined as follows:

$$w_i = [w_1, w_2, \ldots, w_n] \tag{1}$$

The SOM is executed in three steps: Data acquisition and normalization, training the SOM on the supplied data, and then gaining insight regarding the data from the trained SOM. The training process is iterative, with the assignment of input vectors resulting in adjustments to the map. During the training phase an input vector is selected at random from the training dataset and presented to the map of neurons (Kohonen, 2001). The neuron whose weight vector most closely matches the input data as determined by Euclidian distance is selected as the winning neuron. This is defined as:

$$\text{winning neuron } c = \arg\min\|x - w_i\| \tag{2}$$

Once the winning neuron has been selected, the weight vectors of the winning neuron and the neurons that reside in its neighborhood are adjusted to more closely match the current input. The smaller the Euclidian distance is between the input vector and winning neuron, the larger the change to its current weight values:

$$w_i(t+1) = w_i(t) + \alpha(t)h_{ci}(t)[x(t) - w_i(t)] \tag{3}$$

where $x(t)$ is the input vector randomly drawn from the input set at time t; $\alpha(t)$ is the learning rate function; and $h_{ci}(t)$ is the neighborhood function centered on the winning neuron at time t.

Both the learning rate represented by the size of the change to the weight vectors and the neighborhood radius decrease over the number of iterations, which is represented through a Gaussian function:

$$h_{ci}(t) = \exp\left[\frac{-\|r_c - r_i\|^2}{2\sigma(t)^2}\right] \tag{4}$$

where $\sigma(t)$ is the width of the Gaussian kernel; and $-\|r_c - r_i\|^2$ is the distance between the winning neuron c and the neuron i with r_c and r_i representing the two-dimensional positions of neurons c and i on the SOM grid. Vectors continue to be chosen randomly from the training data, and the steps above are repeated until changes to the weight values in a given iteration are below a specific threshold (Kohonen, 2001; Marsland, 2011).

2.2 K-Means

The K-means algorithm employs an approach that is similar to an SOM in regard to the purpose of clustering like data. Like an SOM, the K-means algorithm is also executed in three main steps. The first step is initialization, which involves choosing the number of clusters k. Some variations of the algorithm include dynamic selection of the number of clusters. When employing the core algorithm, Hartigan (1975) suggests running the algorithm with differing values for k and performing an analysis of the variance to determine the optimal number of clusters. Once the number of clusters has been determined, k random vectors are chosen with the same dimensionality and value ranges as the input data. These values are then assigned to serve as the cluster centers u_j. The second step is learning, which involves an iterative process similar to that used with an SOM, whereby each input data item is compared against the cluster centers and then assigned to the one that is closest:

$$d_i = \min_j d(x_i, \mu_j) \tag{5}$$

As a new data item is assigned to a particular cluster, that cluster center point is recalculated as the mean of all currently assigned data items, according to the following equation:

$$\mu_j = \frac{1}{N_j}\sum_{i=1}^{N_j} x_i \tag{6}$$

The process continues until such time as the center values do not update beyond a certain threshold. In the final step, the trained algorithm is used to bin the remaining collection of data after the cluster center points have stabilized (Hartigan, 1975).

3. METHODS

This study was conducted in three phases. First, an algorithm was used to cluster metagenomic reads. The clusters are composed of similar sequences, most likely from related genes. These clusters, in turn, can be used to ascertain what functionalities are present within the given sample habitat by using the consensus sequences within the clusters to perform a BLASTX similarity search (BLASTX compares a nucleotide query sequence translated in all reading frames against a protein sequence database; http://blast.ncbi.nlm.nih.gov/Blast.cgi). This is similar to the goal of the HUMAnN system (Abubucker et al., 2012); however, the pipeline described here does not focus on functionality. In the second phase, the calculated midpoints of the clusters were used as the basis for BLASTN searches, which allowed for the quick identification of likely genomes present (BLASTN compares a nucleotide query sequence against a nucleotide sequence database; http://blast.ncbi.nlm.nih.gov/Blast.cgi). In the third phase, a visual representation of the clustered data + BLAST results was produced, which demonstrates the frequency of each projected organism. These phases together serve as a pipeline that allows one to quickly identify good choices to use as reference genomes for further analysis. Figure 1 provides a flow chart representing the process.

3.1 Data Used

Any collection of mostly non-ambiguous FASTA sequences could be used in the first phase of this process. FASTA, which is a short-hand representation of "fast-all," is a text format for both nucleotide (FAST-N) and protein (FAST-P) data, which includes a descriptive line and the single-letter code representation of either nucleotide or protein sequences. The DNA sequence data that is used in this study is comprised of nucleotides, whose bases are represented by a single letter (A, adenine; G, guanine; T, thymine; C, cytosine; N represents a wildcard). This data consisted of the Great Basin Boiling Spring, Nevada habitat sample, collected on December 1, 2008. The sample data were obtained from the Integrated Microbial Genomes with Microbiome Samples website (https://img.jgi.doe.gov/cgi-bin/m/main.cgi). The sample is

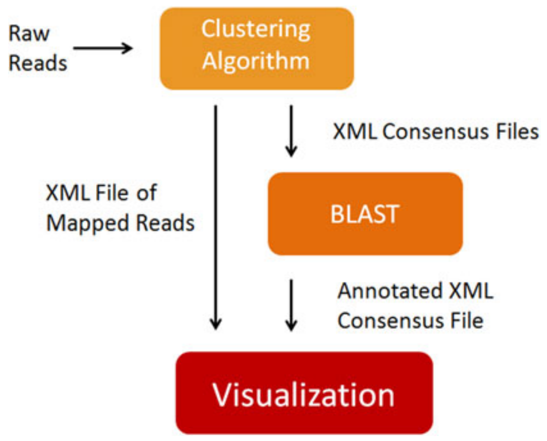


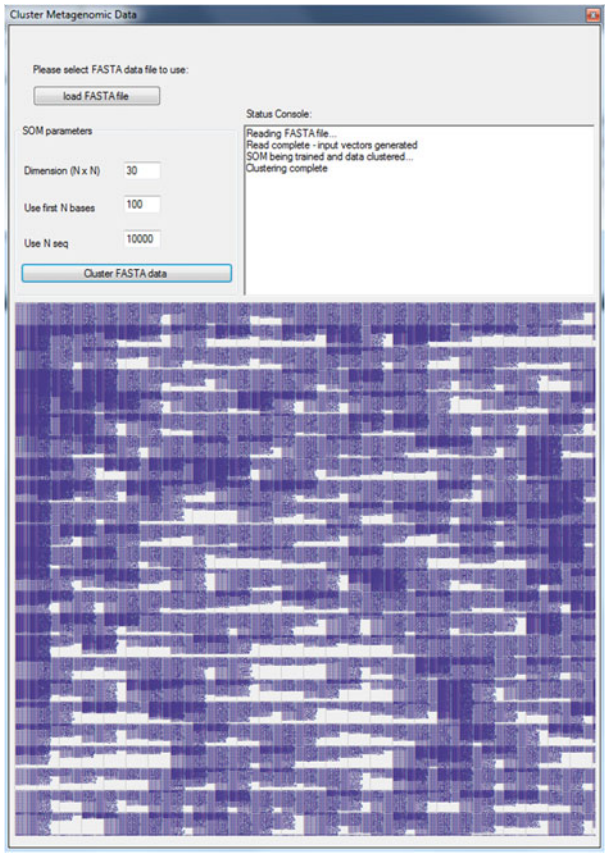
Figure 1. Diagram Showing the Project Pipeline.

from an aquatic thermal springs ecosystem with temperature ranges as high as 90°C. The sequencing technology used on the project was Roche 454 Titanium. The estimated size of the data is 6,283,876 kilobases.

3.2 Phase One: Clustering Tool

C# software applications were developed, which are capable of selecting a FASTA-formatted reads file and clustering the reads either within an SOM or via the K-means algorithm. The SOM program shows a small grid representation of the clustering (which serves as the first visualization of the clustered sequences) and outputs an XML file listing each read and its assigned location on the map. After the initial clustering run, the results were carried forward in a simple XML format, which is illustrated in Appendix A. A second XML file was generated, which displays the neuron weight sequences, as well as the consensus sequence of the neurons with the highest number of mapped sequences. The XML output files were used for the last two parts of this project. A screenshot of the application in action is shown in Figure 2.

A selectable number of encoded partial sequences were used as input vectors to both the map and K-means algorithms. Sequence length was designated by a configurable parameter. Clustering was undertaken with varying sequence count and lengths, but ultimately the authors used a 100-base sequence length, concluding that this length could allow for efficient processing. The neuron with the closest weight



Dimension (N x N) 30
Use first N bases 100
Use N seq 10000

```
Reading FASTA file...
Read complete - input vectors generated
SOM being trained and data clustered...
Clustering complete
```

Figure 2. 100-base Read SOM with a 30 × 30 Grid of Neurons.

vector to the input is assigned that particular input vector and has its weights updated to more closely match the assigned input vector. Other nodes within the winning neuron's neighborhood have their weights updated as well to a lesser amount. This continues to happen in an iterative process until the weight updates fall below an assigned threshold, at which point the map is essentially complete with similar sequences being assigned in close proximity to each other.

It was necessary to employ an encoding scheme that avoided prejudicing the clustering algorithms. A naive encoding scheme could result in the middle two nucleotide values appearing closer to each other. This issue was avoided by encoding each base as four values (e.g., A = 1,0,0,0), which essentially represents each nucleotide at an equal distance from each other in three-dimensional space. However, this did increase the encoded input vector by a factor of four from the single-character representation to the four-character vector. Using this encoding scheme, the clustering results were promising based on the close similarity of the decoded neuron weight sequences versus the consensus sequences built from all of the sequences assigned to that particular neuron. See Table I for a comparison of the calculated consensus sequence against the cluster center weight vector.

The table demonstrates a comparison of a sample neuron weight sequence vs. the consensus sequence generated from the collection of reads assigned to that neuron. The sequences' close similarity (based on a character-by-character analysis) serves to validate the process.

3.3 Phase Two: Manual BLAST Results

The next step involved processing the XML results file that was generated from the clustering algorithms to find likely represented genomes (partial). The consensus XML file with the neuron weight sequences was used as the basis for this phase. Manual BLASTN queries were performed using the microbial genome database at NCBI (National Center for Biotechnology Information). The standard settings were used with the exception of reducing the word size down to seven based on the short length of the sequences. Promising results from the top several "center" sequences were manually added to the consensus XML file (as shown in Appendix A), while weight sequences that produced no acceptable matches were removed. Sequences that produced no significant BLAST results were not carried forward and included in the XML files to avoid unnecessary increases in file sizes, since this data would not be included in the final visual representation.

3.4 Phase Three: Integration of the Combined Information from Phases One and Two

The updated XML file, including the labeled BLASTN results, was fed to a Python application, using the Matplotlib library. The results represented the frequency of the likely genomes, as well as the overall clustered map of sequences. A visualization resembling a heat plot was used to show the frequency of sequences at each position on the map, with the labeled sequences provided by the BLAST results serving as guide locations.

The Python script expects two XML files matching the project's proprietary formats as input. The first portion of the script reads in and parses the two XML files, generating a count of assigned sequences at each

Table I. Comparison of a Sample Neuron Weight Sequence vs. the Consensus Sequence.

Weights Sequence	Consensus Sequence
AAAGAGAAAAGATTAAAAAGAAAATA AAAAAAAAAGAAATGAATTAAGAGAA AATGAGAATGGTGAAGAATAAGAATA TTAAAAGATATAAAAAAAAATA	AAAGAGATAAGAGAAATAAATAAAGC TAAAAAGAAGGAAATGGTTATAAGAA AATAAGAATGAAGAAGAGAAAGTTTA TAAAAAGCTATAGAAAAAAATA

neuron location (*x,y*). In addition, it also parses the updating consensus XML file for the names of the likely reference sequences which will serve as labels within the frequency map.

4. RESULTS AND DISCUSSION

The clustering phase of the pipeline was executed several times with various sequence lengths (65, 100, 120), numbers of reads (5,000; 10,000), and map sizes (30×30, 33×33) for the SOM, as well as different cluster counts for the K-means algorithm. These different implementations resulted in similar clusters. For example, for the SOM runs, the 65-base sequence length on a 30×30 map produced 11 significant clusters—that is, areas of the map with a very large number of assigned reads. The 100-base sequence length on a 30×30 map also produced the same large groupings. The SOM map size provided a more granular mapping simply, because the number of neurons was substantially larger than the cluster counts employed with the K-means algorithm. The BLASTN comparisons were made using varying sequence lengths, but in the end the visualization phase was initiated with only the 100-base results, which can be seen in Table II. The 100-base results represented a significant enough length to enable the differentiation of distinct sequences while still allowing for efficient processing. Although the clustering results were consistent across all parameters, the 100-base results were selected to create a succinct visualization that could facilitate later analysis.

Table II shows selected neuron weight vectors, which were used as the BLAST query sequence generated from the trained SOM, the most similar result as returned by BLAST, the associated E-value (the expected chance of randomly seeing a match to the query sequence), and the percent of the query sequence that matched the top result. Table II shows the possible organisms present in the metagenomic sample, which could be used as reference sequences and warrant further investigation. In particular, the

Table II. 100-base Sequences BLASTN Results.

Neuron Weight Sequence	Top BLAST Result	E-Value	% Identity
TTTTTGAGAAAAAGAAAAAGAAGATAAAAAA AAAAAAAAAGAATGAAGATAGAAGAAGAAT TGAGAGGAGTAAGGGAAAGGGTTAAGAAAG TTAATAAAA	NC_018664.1 Clostridium acidurici 9a chromosome, complete genome	9e-07	88%
TTTCTTTTTTGAACTTTACCTTTTGCTACTTT GGAACTTTTTAAAACACCTAGAGCATTAA CAACAAACCCTTCTACTGTTTCTTTTAAAT CTTCTACCT	BAFA01000036.1 Staphylococcus aureus PM1 DNA, contig: PM1contig00036	0.019	89%
TTTGGAAGAGATTTGAAAGAAGAAAGATA GAAAAAACTTAGGAACTAAAGAAATAAGA GCGCATAAGAGAAACAAGAAAAAAAAAAG AGATTTTTTGGAA	AKXG02000035.1 Leptospira interrogans serovar Grippotyphosa	0.068	89%
TTTTGAGGAAAATAGAAAGGTAAAAGGAA AAATGTAGTAATTAGTTAAAGAAAAAAGA AAGGGTGTGAGAAAAAAAATAAAAAAAA TAGAAAATATGGAA	ADEJ01000237.1 Clostridium difficile 6534 contig_237, whole genome shotgun sequence	5e-04	83%
GTAAAGAATGTTTTTAGATATACGAAG AAAAATTAGTAAAAAAAAAAAAAAA AAAAAAAAAATTATAAAAAAAAAAA AAAAAAAAGAAATTAAAAAAGCAAA	AJUD01000013.1 Thermoanaerobacter siderophilus SR4	6e-15	88%

Table III. Top K-means 100-base Sequence Result.

Neuron Weight Sequence	Top BLAST Result	E-Value	% Identity
TTATAAAAGATATGAAGATAGGTTAGGAG ATTTTTGAGAAGGCTAAAACAGTAGAATT TTCTTAAGGATTATCAGAAAAATAAAAA AAAAAGGAATTTAA	AJUD01000013.1 Thermoanaerobacter siderophilus SR4	0.004	78%

Thermoanaerobacter siderophilus result is especially promising, in that it is a thermophilic, anaerobic, spore-forming bacterium that has been found in hydrothermal vents. This genome is a likely candidate for this environment and therefore can be used as a reference genome. Other labeled results include *Clostridium acidurici* and *Clostridium difficilei*, which are bacteria that are often found in soil and are capable of producing spores that allow them to survive in extreme conditions. Two additional types of bacteria, *Staphylococcus aureus*, and *Leptospira interrogans*, were also identified.

A demonstration of the efficacy of either of these clustering techniques can be seen in the comparison of the consensus sequence from the largest cluster of the K-means algorithm with the results of the SOM. The largest cluster of sequences in the 100-base K-means run produced the same top result as the consensus weight sequence from the SOM application. The consistent top result between both clustering techniques served to validate that phase of the pipeline. Table III shows that the top BLAST result for the K-means algorithm was also *Thermoanaerobacter siderophilus* SR4. However, the identity percentage and E-value are less significant than the SOM results.

Figure 3 shows the results of the final phase of the data processing pipeline and provides an illustration of the clustering of the sample sequences. Color positions on the map represent the abundance of sequences similar to that particular cluster center. Several cluster centers are labeled with the corresponding top BLAST results to give an investigator a visual overview of the abundance and possible types of sequences

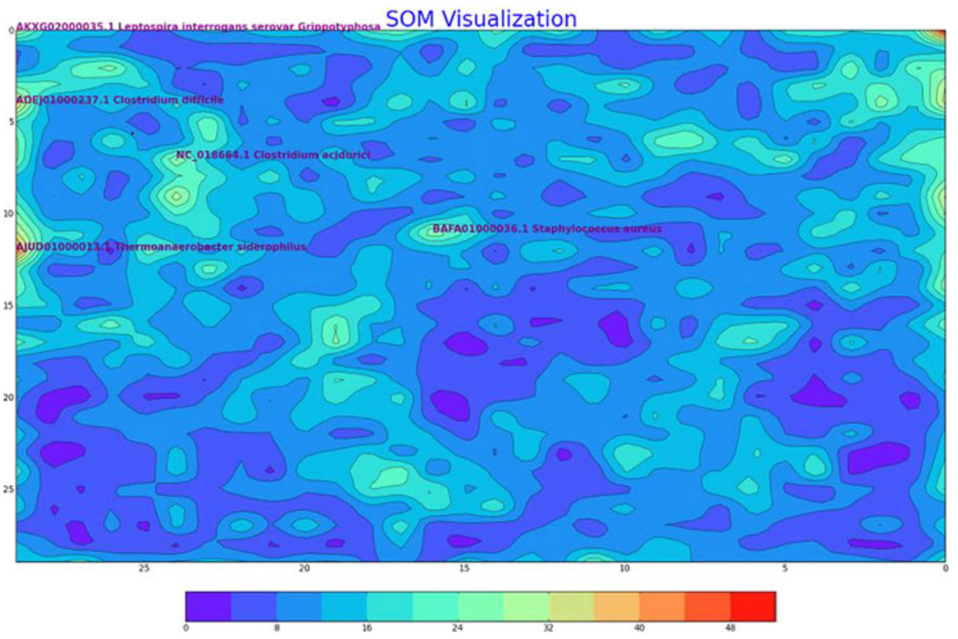


Figure 3. SOM Data Visualization with BLASTN Results Labeled. The key shows that color positions on the map represent the number of sequences assigned to that location. Clusters are shown in the light blue, yellow, and red areas.

present in the sample. This representation demonstrates the applicability of the system and shows the benefit of refining and proceeding with this clustering technique.

The previously discussed CD-HIT application (Li & Godzik, 2006; Li, 2014; Edgar, 2010) was employed to attempt to validate the techniques used here. The metagenomic data used in the development of the described pipeline was subjected to the CD-HIT-454 application for clustering. These results were fed to the consensus building program, CD-HIT-CONSENSUS, which creates a set of consensus sequences from the clusters provided by CD-HIT-454. However, this provided a limited point of comparison as little reduction in data was found when evaluating mapped reads against reference genomes. CD-HIT produced significantly fewer clusters using the pipeline data than did the authors' method and could not serve as a functional point of comparison.

5. CONCLUSION

This paper attempts to solve a problem similar to that addressed by the TaxSOM application, a tool that trains an SOM on the basis of known genomic signatures and then maps metagenomic reads to the pre-calculated map (Weber et al., 2011). The pipeline presented here uses the metagenomic reads themselves to train the clustering technique as a first step. The metagenomic data are then clustered using the trained algorithm—either the SOM or the K-means algorithm. This technique is both applicable and advantageous because it involves no initial preparation, attempting to reduce the data from the raw reads without reference to an outside source for comparison. The clustered sequences can then be used to reduce the amount of data to carry forward for downstream analysis.

The techniques employed in this paper form the starting point in a metagenomic analysis pipeline. The clustering algorithms used in conjunction with the visualization tool allow a researcher to rapidly move to a more directed investigation of a metagenomic sample. The overarching goal of this pipeline is to provide a quick visual guide to an investigator as to what may be present in a habitat's sample and to identify which reference genomes may be selected for further analysis. Figure 3 illustrates how an investigator can use the visualization to see the amount of sequences that share similarities with the cluster labels.

The clustering performed here was limited to a small subset of a much larger metagenomic data file in order to validate the functionality of the method. The complete pipeline was executed using only a leading subset of the bases of each read as the input vectors to the SOM and K-means algorithms. Future work could use the pipeline to cluster larger metagenomic datasets and the complete read sequences. In addition, in this study the connectivity between the clustering, BLAST, and visualization was user driven, but in the future these steps could all be rolled into a single tool. Finally, the system described here could be enhanced to go in several different directions, including the classification of taxonomies present in a sample or the clustering of metagenomic reads for a reduction in computational requirements. In addition, it could also be adapted to work in a fashion similar to the HUMAnN system described in Abubucker et al. (2012), which seeks to determine the functionalities present in the metagenomic sample. This direction would involve finding similarities in the consensus (or weight) sequences within the map versus known pathways to elucidate possible functionalities.

ACKNOWLEDGEMENTS

This material is based upon work supported in part by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

DISCLOSURE STATEMENT

No potential conflict of interest was reported by the authors.

REFERENCES

Abubucker, S., Segata, N., Goll, J., Schubert, A. M., Izard, J., Cantarel, B. L., … Huttenhower, C. (2012). Metabolic reconstruction for metagenomic data and its application to the human microbiome. *PLoS Computational Biology*, *8*, e1002358. doi:10.1371/journal.pcbi.1002358

Aggarwal, C. C., & Reddy, C. K. (2013). *Data clustering: Algorithms and applications*. Hoboken: Chapman and Hall/CRC.

Altschup, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, *215*, 403–410. doi:10.1016/S0022-2836(05)80360-2

Bazinet, A.L, & Cummings, M.P (2012). A comparative evaluation of sequence classification programs. *BMC Bioinformatics*, *13*, 92. doi:10.1186/1471-2105-13-92

Chistoserdova, L. (2009). Functional metagenomics: Recent advances and future challenges. *Biotechnology and Genetic Engineering Reviews*, *26*, 335–352. doi:10.5661/bger-26-335

Culligan, E., Sleator, R., Marchesi, J., & Hill, C. (2014). Metagenomics and novel gene discovery: Promise and potential for novel therapeutics. *Virulence*, *5*, 339–412. doi:10.4161/viru.27208

Edgar, R.C (2010). Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, *26*, 2460–2461. doi:10.1093/bioinformatics/btq461

Ghosh, T. S., Mohammed, M. H., Rajasingh, H., Chadaram, S., & Mande, SS (2011). HabiSign: A novel approach for comparison of metagenomes and rapid identification of habitat-specific sequences. *BMC Bioinformatics*, *12*, S9. doi:10.1186/1471-2105-12-S13-S9

Handelsman, J., Rondon, MR, Brady, SF, Clardy, J., & Goodman, RM (1998). Molecular biological access to the chemistry of unknown soil microbes: A new frontier for natural products. *Chemistry & Biology*, *5*, R245–R249. doi: 10.1016/S1074-5521(98)90108-9

Hartigan, J. A. (1975). *Clustering algorithms*. New York: Wiley.

Jain, A.K (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*, 651–666. doi:10.1016/j.patrec.2009.09.011

Kelley, D.R, & Salzberg, S.L (2010). Clustering metagenomic sequences with interpolated Markov models. *BMC Bioinformatics*, *11*, 544. doi:10.1186/1471-2105-11-544

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*, 1–6. doi:10.1016/S0925-2312(98)00030-7

Kohonen, T. (2001). *The self-organizing map* (3rd ed.). Berlin: Springer.

Lai, C. -C. (2005). A novel clustering approach using hierarchical genetic algorithms. *Intelligent Automation & Soft Computing*, *11*, 143–153. doi:10.1080/10798587.2005.10642900

Li, W. (2014). Retrieved from http://weizhong-lab.ucsd.edu/cd-hit/ CD-HIT [Clustering program]. Sanford-Burnham Medical Research Institute.

Li, W., Fu, L., Niu, B., Wu, S., & Wooley, J. (2012). Ultrafast clustering algorithms for metagenomic sequence analysis. *Briefings in Bioinformatics*, *13*, 656–668. doi:10.1093/bib/bbs035

Li, W., & Godzik, A. (2006). CD-HIT: A fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, *22*, 1658–1659. doi:10.1093/bioinformatics/btl158

Liao, Q., Huang, Y., He, S., Xie, R., Lv, Q., Shilai, Y., … Qian, C. (2013). Cluster analysis of citrus genotypes using near-infrared spectroscopy. *Intelligent Automation & Soft Computing*, *19*, 347–359. doi:10.1080/10798587.2013.824719

Marsland, S. (2011). *Machine learning: An algorithmic perspective*. Boca Raton: CRC Press.

Murphy, K. (2012). *Machine learning: A probabilistic perspective*. Cambridge: MIT Press.

Weber, M., Teeling, H., Huang, S., Waldmann, J., Kassabgy, M., Fuchs, B. M., … Glöckner, F. O. (2011). Practical application of self-organizing maps to interrelate biodiversity and functional data in NGS-based metagenomics. *The ISME Journal*, *5*, 918–928. doi:10.1038/ismej.2010.180

NOTES ON CONTRIBUTORS

Damien Ennis is Chair of the Department of Mathematics at Truckee Meadows Community College, USA. He received a B.S. in Mathematics from the University of Nevada, Reno, an M.S. in Bioinformatics from the Johns Hopkins University, and M.S. and Ph.D. degrees in Computer Science and Engineering from the University of Nevada, Reno. His research interests are in machine learning, bioinformatics, and game development.

Sergiu Dascalu is a professor in the Department of Computer Science and Engineering and the Director of the Software Engineering Lab (SOELA) at the University of Nevada, Reno, USA. He received an M.S. in Automatic Control and Computers from the Polytechnic Institute of Bucharest, Romania in 1982 and a Ph. D. in Computer Science from Dalhousie University, Halifax, Canada in 2001. His research interests are in software engineering, human-computer interaction, simulation environments, and data-intensive systems.

Frederick C. Harris, Jr. is a professor in the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab at the University of Nevada, Reno, USA. He received B.S. and M.S. degrees in Mathematics and Educational Administration from Bob Jones University in 1986 and 1988, respectively, and M.S. and Ph.D. degrees in Computer Science from Clemson University in 1991 and 1994, respectively. He is a Senior Member of both the ACM and ISCA. His research interests are in parallel computation, computational neuroscience, computer graphics and virtual reality.

Appendix A. XML File Formats

A.1 MappedSequences.xml

```
< ?xml version = “1.0” encoding = “UTF-8”? >
- < MappedData >
< Data  Squence = “CATTCTTATATCTCTTGTTCCGCCTTCATAATCTTACACCTGGCCTTCCTCCTATTA
CGAACCTTCCACCGCGTAATTTACCTCTTACAGAACTCAAACA”  > GCXNWHH02GP572,18,22,TTTTGC
TTTATATTTTTCTTTATCTTCAAATTTAAGTTTTTCATTTTTTTCCACTAAATCTATTACCACTTCTAAGTA-
AATAACTTTAAGAAGTTCGACT < /Data >
    < Data  Squence = “GAACGTCATTAGGATACCGACTACGCAGAAGAGATGACAACT  TCACCAACTA
CAATAAGATGCAGTCCACTATAGCTATGTTGGCGCAGGCGAACGCATT”  > GCXNWHH02HND40,25,9,
AAAGAGAAAAGATTAAAAAGAAAATAGAAAAAAAAAAAAAACGATGAAGATAAAATGAAAATGGTAA-
ACAATAAAAATATTGAAAAAGATAAAAAAAAAAA < /Data >
      < Data  Squence = “TTTACAGATATGCGCTCCTGCTCTATTCCCAAAGCAAGGAAGATTCTGTATTA
TCTTGTTTTGATAGGTGCAATCTTTGGCTCGTCTGCAATTAAAGTGC”  > GCXNWHH02HFAV5,27,29,
TTTTAGAACCTTACTTTTTTCTTTGTTTTTACTAATGTAAAACATTTTTGAGCTATTTTTTTTCATCCTTTT-
ATTAATTCTTGTCTTATCCATACTATCA < /Data >
    < Data   Squence = “CTCATTATTCAGACTTCTGGATCCTTGATATATGGAGCTATGTGTCCATTCT
CTCTTATAGGTGTTGCAATCCAGTATTTCCGTTAGTTGGCACCGAGGC”  > GCXNWHH02FVWJK,7,11,
TTTTTTATAAAGTCTTAAAAATTGTCTATGGTTTTCTTGGATTTTCACTATATTACCAAGAAAAGGAGTT-
ACTAAAGTTTATCTTTGTAAGAATTAATGA < /Data > .
    . . .
```

A.2 Labeled ConsensusSequences.xml

```
 < ?xml version = “1.0” encoding = “UTF-8”? >
- < ConsensusSequences >
< Sequence >  < Location > 0,10 < /Location >  < Count > 28 < /Count >  < Weights > TTTTACTTCATTT-
 Weights > TTTTACTTCATTTTCCAATTACTTAAAGCTTTATTATTCTTTTTTCTCTATTCTTTCCATATTAT-
TTTTTATATTCTCTACTTTTTTTTTGTATTTTTTT < /Weights >
```

< Consesus > TTTTGAATTCTTTTCCATATTGACTAAGAGCTCATTGTGTTTCGTCTTCATAGTTCCCA-
TTATCTAATCTATAACCTTTACTTTTTTTTCCTATTTTTTT < /Consensus > </Sequence >

< Sequence > < Location > 24,7 < /Location > < Count > 29 < /Count > < Weights > TTTTTGAGA-
Weights > TTTTTGAGAAAAAGAAAAAGAAGATAAAAAAAAAAAAAAAGAATGAAGATAGAAGAAGA-
ATTGAGAGGAGTAAGGGAAAGGGTTAAGAAAGTTAATAAAA < /Weights > < Consesus > GATTTTGG-
Consesus > GATTTTGGAAAAAGATTGCGAAGAAAAGAAAGAAAAAGAAGAGTAAATGTATATGATAA-
AAAGGTAAAGGTAGAGGAAAGGGTTAATAATGTACCTATTA < /Consensus > < TopBlastResult >
NC_018664.1 Clostridium acidurici < /TopBlastResult > </Sequence >

< Sequence > < Location > 16,11 < /Location > < Count > 29 < /Count > < Weights > TTTCTTTTT-
Weights > TTTCTTTTTTGAACTTTACCTTTTGCTACTTTGGAACTTTTTAAAACACCTAGAGCATTAACA-
ACAAACCCTTCTACTGTTTCTTTTAAATCTTCTACCT < /Weights > < Consesus > TTTCAGTTTTGACCA-
Consesus > TTTCAGTTTTGACCATCATCCTTTACTATTTTCGTACTTTTCTCAACTCCAACAGCAATGAA-
AGCAAACATTTAAGCAGTTTCTTTTAAATCACCTTCTT < /Consensus > < TopBlastResult > BAFA
01000036.1 Staphylococcus aureus < /TopBlastResult >

< /Sequence > - < Sequence > < Location > 29,0 < /Location > < Count > 30 < /Count > < Weights >
TTTGGAAGAGATTTGAAAGAAGAAAGATAGAAAAAACTTAGGAACTAAAGAAATAAGAGCGCATAAGA-
GAAACAAGAAAAAAAAAAGAGATTTTTTGGAA < /Weights > < Consesus > TTTGGAAGGTATGTTAAA-
Consesus > TTTGGAAGGTATGTTAAAAAAGAAAAAAAGAAAATACGTAGGAATAACTATAATAAGACCT-
CATAGTATAAACCAGGGAAAAATAGGAATTTTTATGGAT < /Consensus > < TopBlastResult > AKXG
02000035.1 Leptospira interrogans serovar Grippotyphosa < /TopBlastResult > </Sequence > .

. . .