Separating Blended DNA Sequences

Vasim Mahamuda

Department of Computer Science
The University of Georgia
Athens, GA 30602, USA

Justin Martin

Department of Computer Science
The University of Georgia
Athens, GA 30602, USA

Ankur Oberai

Department of Computer Science
The University of Georgia
Athens, GA 30602, USA

Abstract

A novel technique is described for separating the DNA samples of two different species (*Peromyscus* and Oyster). The approach involves matching the mixed gene sequences of Oyster and *Peromyscus* to the already known DNA sequences of the species. A technique called BLAST [1] is used to measure similarities between the mixed sequences and the known sequences. The BLAST [1] results are taken, and then we perform a codon analysis. We calculate the dinucleotide frequency of each of the base pairs and then feed it to a Neural Network to build a model so the remaining unclassified sequences are also separated into known groups (Oyster or *Peromyscus*). Results from experiments are described which investigate the technique being presented.

1. Introduction

This paper outlines a mechanism for analyzing the DNA sequences and then separating them out into known sequences of either Oyster or *Peromyscus*.

> Mice of the genus *Peromyscus* are found in nearly every habitat from Alaska to Central America and from the Atlantic to the Pacific. They provide an evolutionary outgroup to the *Mus/Rattus* lineage. They also serve as an intermediary between the *Mus/Rattus* lineage and humans. Although *Peromyscus* has been studied extensively under field and laboratory conditions, research has been limited due to the lack of molecular resources [2].

The Oyster sequences originated from the Pacific Ocean and are of the species *Crassostrea gigas*; we will refer to this species as Oyster throughout the paper. The Department of Energy's Joint Genome Institute is responsible for mixing the *Peromyscus* and Oyster DNA sequences.

Section 2 describes the background involved in the project. Section 3 shows our ideas for classifying the mixed data, and Section 4 describes our implementation for processing this data with the approach incorporated. Section 5 shows the results we obtained, and in conclusion, Section 6 elaborates on future research that can be done.

2. Background

This paper tends to explore a heavily researched area called meta-genomics, using machine learning techniques.

> Metagenomics (also known as Environmental Genomics, Ecogenomics, or Community Genomics) is the study of

genetic material recovered directly from environmental samples. Traditional microbiology and microbial genome sequencing rely upon cultivated clonal cultures. This relatively new field of genetic research enables studies of organisms that are not easily cultured in a laboratory as well as studies of organisms in their natural environment [3].

Artificial Neural Networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued functions from examples. ANN learning has been successfully applied to problems such as interpreting visual scenes, speech recognition, and learning robot control strategies. The motivation behind using an ANN is there can be noise in the data samples involved in the project. Data samples that do not represent the actual facts are classified as noise. Noise can seriously misguide our classifier in a case where a noise-sensitive classifier is used. ANNs also give us a fast evaluation of the learned target function.

The data we have is comprised of DNA sequences.

Deoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms and some viruses. The main role of DNA molecules is the long-term storage of information. DNA is often compared to a set of blueprints, a recipe, or a code, since it contains the instructions needed to construct other components of cells, such as proteins and RNA molecules. The DNA segments that carry genetic information are called genes, but other DNA sequences have structural purposes or are involved in regulating the use of this genetic information. DNA is a long polymer made from repeating units called nucleotides. Although each individual repeating unit is very small, DNA polymers can be very large molecules containing millions of nucleotides. The DNA double helix is stabilized by hydrogen bonds between the bases attached to the two strands. The four bases found in DNA are adenine (abbreviated A), cytosine (C), guanine (G), and thymine (T). Each type of base on one strand forms a bond with just one type of base on the other strand. This is called complementary base pairing. A bonds only to T, and C bonds only to G. This arrangement of two nucleotides binding together across the double helix is called a base pair [4].

An illustration can be seen in Figure 1.

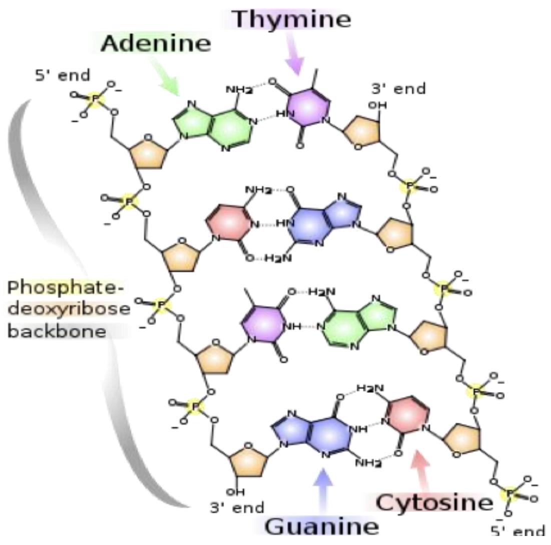


Figure 1

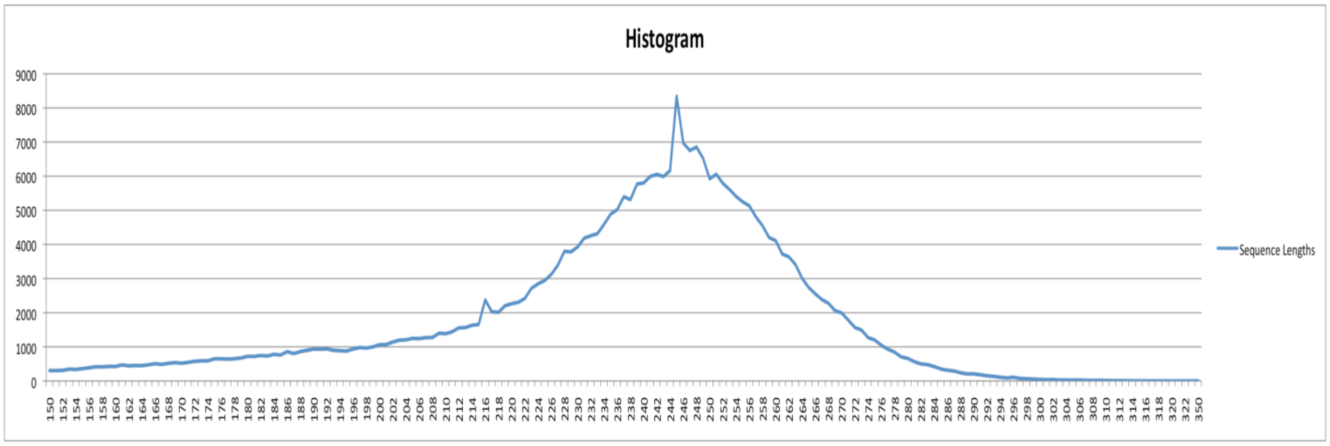


Figure 2

3. Approach

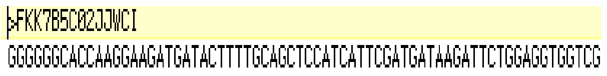
We first take the mixed sequences and try to match them to the known sequences. We have DNA sequences of length ranging between 150-350 bases in the mixed data set, and the pure data has sequences in the range of 600-800 bases. Our primary goal is to match the sequences from the mixed data set with those sequences from the known data set, allowing for insertions and deletions. A low e-value produced by the BLAST [1] algorithm indicates a high similarity score between the blasted sequence and the database it was blasted against. If we find a match with a high similarity score (or low e-value), then we directly classify the mixed data set sequences into either *Peromyscus* or Oyster. The process is completed using the BLAST [1] algorithm. It accounts for all the insertions and deletions between the sequences and returns the similarity score for the sequences.

We classify some of the sequences in the mixed data set by taking a threshold value for the similarity score. Using this information, we can say with high confidence the blasted sequences belong to the particular species they were blasted against. We take these classified sequences as the training set and now train an Artificial Neural Network (ANN) [5] on those sequences. The remaining unclassified sequences become the test set. We may also refer to an ANN simply as a neural network. To measure the validity of our training set, we use 10-fold cross-validation [5].

If we have a non-uniform distribution of sequence lengths, then we need to have different neural networks for different distributions of lengths. Suppose we have three distributions of the data. We modify each distribution by padding the margins of the smaller length sequences with "don't care" symbols. Three distributions can then be used to train three different neural networks. As we can see from the histogram (Figure 2), the majority of sequences are 200-280 bases in length. Another aspect of Figure 2 is the lengths of sequences are uniformly distributed. In order to achieve maximum results when using ANNs, uniformly distributed data should be used. There is a slight issue to be addressed regarding our data sets. The inputs for an ANN should be of fixed length, and our data is of variable length. One idea is to find the longest sequence and set the input length for the ANN to that length. If any sequence is smaller in length, the first and last inputs can be padded with "don't care" symbols. Although this solves the variable length problem, we now have another issue to solve. Neural networks must take numbers as their inputs, and our data is represented as letters, namely A, G, T, C, and n; the n occurs when previous methods used by the biologists could not make a decision to use A, G, T, or C within a sequence. Consider representing each of the letters as a number, namely 1, 2, 3, and 4. Using numbers creates a very undesirable problem because the neural network will associate the numbered labels as input weights. This leads us to an even better solution – use an encoding procedure. We have discovered an encoding procedure described below for training the neural network. We took each sequence from the mixed data set and calculated the dinucleotide compositions; we calculated the percentages of AA, AT, AG, AC, An, TA, TT, TG, TC, Tn, GA, GT, GG, GC, Gn, CA, CT, CG, CC, Cn, nA, nT, nG, nC, and nn. These will be the attributes for the input to the neural network.

4. Implementation

Once we receive the similarity scores (e-values) between the mixed sequences and the databases, we are able to perform Machine Learning techniques on our data. First, we must parse the mixed sequences to separate each sequence from its key. In Figure 3, the key occurs after the ">", and the sequence occurs on the next line. The keys and sequences are added to a hash map since each key is unique. Next, the BLAST output is parsed. We must now decide which sequences will belong to our training set. If we take a collection of mixed sequences and perform the BLAST algorithm on them using a particular database, say Oyster, we will label the sequences that have an e-value below a certain threshold as Oyster DNA. The sequences with e-values above the threshold become the testing set, or the sequences that need to be classified by the neural network, and



```
>FKK7B5C02JJWCI
GGGGGCACCAAGGAAGATGATACTTTTGCAGCTCCATCATTCGATGATAAGATTCTGGAGGTGGTCG
```

Figure 3

we label them as unknown.

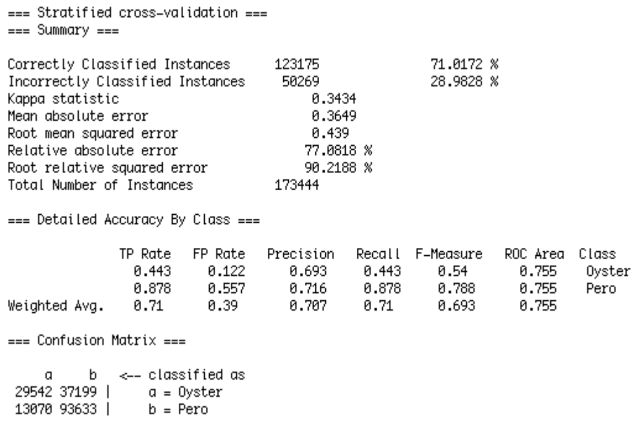
Since the BLAST output contains sequence keys and not the sequences themselves, we must match each key from the BLAST output to its corresponding sequence from the hash map we built earlier for the file containing mixed sequences. When we retrieve a sequence based on its key, we can give it the label it deserves; it is either labeled as a species name or unknown. We found the files containing mixed sequences have some sequences appearing more than once. Obviously, we do not want sequences to appear more than once in either the training or testing set. This issue is taken care of through the use of hash maps. When we assign sequences to a label, the sequence itself is represented as the key in a hash map with its corresponding value as the label. Using hash maps make this redundancy is easy to eliminate.

The most important part about our implementation is generating the training and testing set files. We use the WEKA [6] package where the input must be files of the extension .arff. The most important aspects of these files are, of course, the attributes and examples. We calculate the dinucleotide frequency of each of the base pairs for each sequence, which is a timely process. Having five possible characters in a sequence, namely A, T, G, C, and n, there are 25 possible combinations of pairs for which to calculate the frequency. Once each dinucleotide frequency is obtained, a floating-point number is calculated as one attribute for a sequence. This floating-point number represents a weight for a dinucleotide frequency. Thus, each example contains 25 attributes plus the class it belongs to; the class will be either Oyster or Pero in the training set and “?” in the testing set.

Since the time required to process these large files was a big challenge, we chose to run the parsing code on a server as a background process. The first time we parsed the files, it took around ten hours to finish. That particular server had one 1.7 GHz processor. Having access to a second server, which had three Dual-Core AMD processors with a speed of over 2.0 GHz each, we were able to run the same process in about 1.5 to 2 hours. This was a big difference in time for our experiments.

5. Results

The results are taken by using the neural network classifier from the Machine Learning algorithm collection in WEKA [6]. The model is built using the default parameters, with the learning rate at 0.3 and a momentum of 0.2.



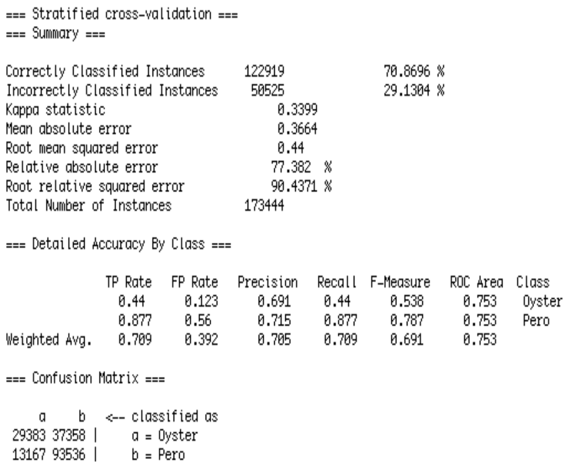
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      123175               71.0172 %
Incorrectly Classified Instances     50269               28.9828 %
Kappa statistic                          0.3434
Mean absolute error                      0.3649
Root mean squared error                  0.439
Relative absolute error                 77.0818 %
Root relative squared error             90.2188 %
Total Number of Instances           173444

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.443     0.122      0.693     0.443     0.54        0.755    Oyster
                 0.878     0.557      0.716     0.878     0.788       0.755    Pero
Weighted Avg.    0.71      0.39       0.707     0.71      0.693       0.755

=== Confusion Matrix ===

     a     b   <-- classified as
 29542 37199 |     a = Oyster
 13070 93633 |     b = Pero
```

Figure 4

Figure 4 shows the results of training the network using 10-fold cross-validation with 15 hidden nodes and a training time of 10 epochs. The percentage of correctly classified instances is 71.0172%. A confusion matrix (Figure 4) contains information about the actual and predicted classifications done by a classification system. The entries in the confusion matrix have the following meaning in the context of our study:

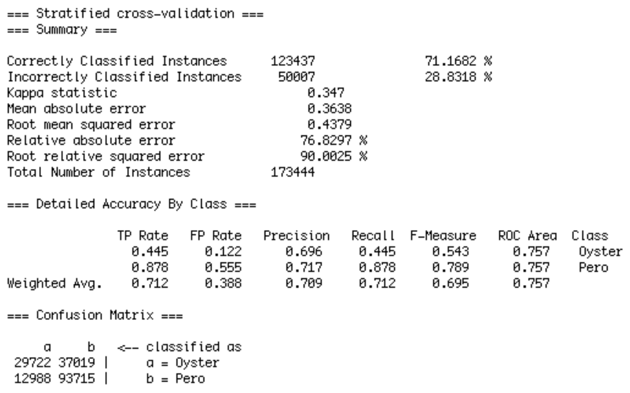
- 29,542 is the number of Oysters classified as Oysters
- 37,199 is the number of Oysters classified as *Peromyscus*
- 13,070 is the number of *Peromyscus* classified as Oyster
- 93,633 is the number of *Peromyscus* classified as *Peromyscus*

Some other results include:



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      122919               70.8696 %
Incorrectly Classified Instances     50525               29.1304 %
Kappa statistic                          0.3399
Mean absolute error                      0.3664
Root mean squared error                  0.44
Relative absolute error                 77.382  %
Root relative squared error             90.4371 %
Total Number of Instances           173444

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.44      0.123      0.691     0.44      0.538       0.753    Oyster
                 0.877     0.56       0.715     0.877     0.787       0.753    Pero
Weighted Avg.    0.709     0.392      0.705     0.709     0.691       0.753

=== Confusion Matrix ===

     a     b   <-- classified as
 29383 37358 |     a = Oyster
 13167 93536 |     b = Pero
```

Figure 5



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      123437               71.1682 %
Incorrectly Classified Instances     50007               28.8318 %
Kappa statistic                          0.347
Mean absolute error                      0.3638
Root mean squared error                  0.4379
Relative absolute error                 76.8297 %
Root relative squared error             90.0025 %
Total Number of Instances           173444

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.445     0.122       0.696     0.445     0.543      0.757    Oyster
                 0.878     0.555       0.717     0.878     0.789      0.757    Pero
Weighted Avg.    0.712     0.388       0.709     0.712     0.695      0.757

=== Confusion Matrix ===

     a     b   <-- classified as
 29722 37019 |     a = Oyster
 12988 93715 |     b = Pero
```

Figure 6

Figures 5 and 6 have the same configurations as Figure 4 except that Figure 5 has a training time of 7 epochs, and Figure 6 has a training time of 13 epochs. Figure 5 displays 70.8696% correctly classified instances, whereas Figure 6 has 71.1682% of correctly classified instances. As you can see, the percent of correctly classified instances increased as the training time increased.

The cross-validation approach is used to determine the number of iterations that yield the best performance on the validation set. The mean of these estimates for the optimal number of iterations is then calculated, and a final run of the Backpropagation algorithm is performed, training on all examples for iterations with no validation set.

6. Conclusion and Future Work

In our project, we classify the mixed data at a high level of abstraction. We can obtain better results if the work is extended to comparing the sequences against the clone libraries and taking into consideration the fact that we are not aware of the purity of the data sequences from Oyster. There can be more similar species as the one we used, such as *Caenorhabditis elegans*, *Ciona intestinalis*, *Ciona savignyi*, and *Drosophila melanogaster*.

Given the enormous size of the files, we could only perform limited experimentation with the neural network factors, such as the number of epochs and hidden nodes. The time it takes to perform experiments is a large factor in this project. The different network configurations, such as the learning rate and momentum, can also give more results. Experimenting with such details can result in a better classifying network.

Combinations of models can result in extremely beneficial classifiers. Combining models can be achieved through common methods like bagging, boosting, and stacking.

References

[1] http://en.wikipedia.org/wiki/BLAST

[2] Julie L. Weston Glenn, Chin-Fu Chen, Adrienne Lewandowski, Chun-Huai Cheng, Clifton M. Ramsdell, Rebecca Bullard-Dillard, Jianguo Chen, Michael J. Dewey, and Travis C. Glenn, *Expressed sequence tags from Peromyscus testis and placenta tissue: analysis, annotation, and utility for mapping,* BMC Genomics 2008.

[3] http://en.wikipedia.org/wiki/Metagenomics

[4] http://en.wikipedia.org/wiki/DNA

[5] Tom M. Mitchell, *Machine Learning*, McGraw-Hill 1997

[6] http://www.cs.waikato.ac.nz/ml/weka/