GENETIC SEQUENCE CLASSIFICATION AND PHYLOGENETIC CONSTRUCTION WITH N-GRAM METHODS

by

CHANDLER KINCAID

(Under the Direction of Khaled Rasheed)

ABSTRACT

This work presents two papers in which machine learning techniques are used to analyze genetic sequence information. In the first paper feature vectors were created from protein data of the influenza virus using N-gram methods common to text classification. A number of classifiers were trained on the feature vector and were successful in predicting influenza host organisms of corresponding viral strains. The best classifier achieved an accuracy of 97.2% on a set of over 700,000 sequences, the largest experiment of its kind to date. The second paper explores N-gram feature vectors in phylogenetic construction. Methods are presented which speed up feature vector creation by 26% over the state of the art. GPU optimized functions were examined in the distance matrix calculation task of phylogenetic construction and showed up to a 33x speed up in comparison with CPU based methods.

INDEX WORDS: Artificial Intelligence, Big Data, Bioinformatics, Classification, Computer Science, Data Science, Epidemiology, Influenza, Machine Learning, N-gram, Protein, Taxonomy, Phylogeny, Phylogenetic Construction

GENETIC SEQUENCE CLASSIFICATION AND PHYLOGENETIC CONSTRUCTION WITH N-GRAM METHODS

by

CHANDLER KINCAID

B. S., The University of California, Santa Cruz, 2013

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2108

GENETIC SEQUENCE CLASSIFICATION AND PHYLOGENETIC CONSTRUCTION WITH N-GRAM METHODS

by

CHANDLER KINCAID

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Quantifying Evolutionary Distance

At first glance the task of quantifying evolutionary distance might appear to be a simple one. For any organism a sequence of DNA exists which codes for the qualities of that organism. These sequences drift apart through the processes of natural selection and speciation. It follows that measuring the dissimilarity of two sequences should be an accurate measure of the evolutionary distance between two organisms. In reality measuring the evolutionary distance between organisms is complicated by a number of factors.

Evolution does not occur at a constant rate with respect to time. Quite a few variables can change the rate of evolutionary divergence. Some variables include innovations that expose a new adaptive niche, mass extinction events, geographic isolation, opening new living environments, ecological competition and many others. A significant example of this phenomena is the Cambrian Explosion. During the Cambrian Explosion the rate of creation and extinction of new species increased an entire order of magnitude from the previous period (Butterfield, 2007). During this relatively short period the majority of animal phyla we know today came into existence (Budd, 2013). From this evidence we can stipulate that even if an entirely accurate measure of evolutionary distance is known, we can not necessarily determine the time at which two species of families diverged. To determine that information scientists must rely on the fossil record and techniques such as radio carbon dating. While using a purely sequence based analysis, the chronology of a phylogenetic reconstruction cannot be strictly accurate.

Changes in genetic sequences are not a one-for-one match with phenotypical expression. It is estimated that only 10 – 15% of the human genome is functional (Ponting, 1970) although there is debate over what percentage of the genome might be chemically relevant to biological processes (Encode Consortium, 2012). The majority of mutations that sequences undergo do not make any changes in proteins that are created. These mutations are referred to as silent mutations. Silent mutations can occur when changes are made in non-coding parts of DNA as described but they can also occur in coding portions of DNA. This is possible because all amino acids, which code for proteins, can be created from multiple different sequences of DNA. For example in Figure 1.1 we can see that Alanine can have any codon in its third position and still code for the same protein. An important question when quantifying evolutionary distance arises, should silent mutations be considered when measuring the distance between two organisms? If only functional changes are considered then the measure of distance between two sequences more closely corresponds to the amount of evolutionary pressure that two organisms have experienced since their diverging point. If only silent mutations are considered then evolutionary pressure is not measured but instead the distance more closely represents the amount of time two organisms have undergone since their diverging point. This is still only an approximate measure given that the rate of mutation can be influenced by various factors and species have different corrective mechanisms for mutation. Some species have a greatly reduced amount of non-coding DNA making direct comparisons more challenging (Venkatesh, Gilligan, & Brenner, 2000).

The question of whether to consider non-coding DNA and silent mutations when quantifying evolutionary distance alludes to a larger choice when creating a distance metric, whether to use nucleotide or amino acid data. The use of nucleotide data has some strong arguments for it. With this kind of data no information is lost unlike with amino acid data as demonstrated in Figure 1.1. Estimating the probability of one nucleotide mutating into another is also simpler considering there are only four possibilities instead of 20. Some research has shown nucleotide based substitution models to be more robust than traditional amino acid substitution matrices in the context of phylogenetic inference (Miyazawa, 2013). Nucleotide models also have an advantage when quantifying the distance between two small, closely related sequences. Closely related sequences may have not undergone enough selective pressure to change a particular protein on a functional level. Most mutations are silent or synonymous which means that nucleotide data can capture this difference in some contexts where there is no difference in amino acid data. There are also advantages to using amino acid data. In many machine learning problems feature selection is used to filter out extraneous data that reduces the performance of an algorithm. From a machine learning perspective amino acid data can be thought of as feature extracted nucleotide data in which all features are guaranteed to have some change in a resultant protein. When sequences are more distantly related in general their functional differences are more pronounced. Tracking silent mutations in some cases can help to obscure the functional differences between sequences. In general it can be expected that in machine learning tasks nucleotide data will perform better with closely related sequences while amino acid data will perform better with distantly related sequences, although this may not always be the case.

In the context of N-gram feature selection, nucleotide and amino acid data have significant differences. Using a standard sliding window technique for creating features presents a serious problem for nucleotide data. Advancing the window one nucleotide at a time creates a feature vector in which many word counts do not code for functional information and even worse dramatically skew the word counts that do. This problem can be solved by advancing the feature window by three nucleotides at a time, however data must be complete and the starting position must be known. Using amino acid data in an N-gram method allows smaller sizes of N to convey more information than nucleotide data.

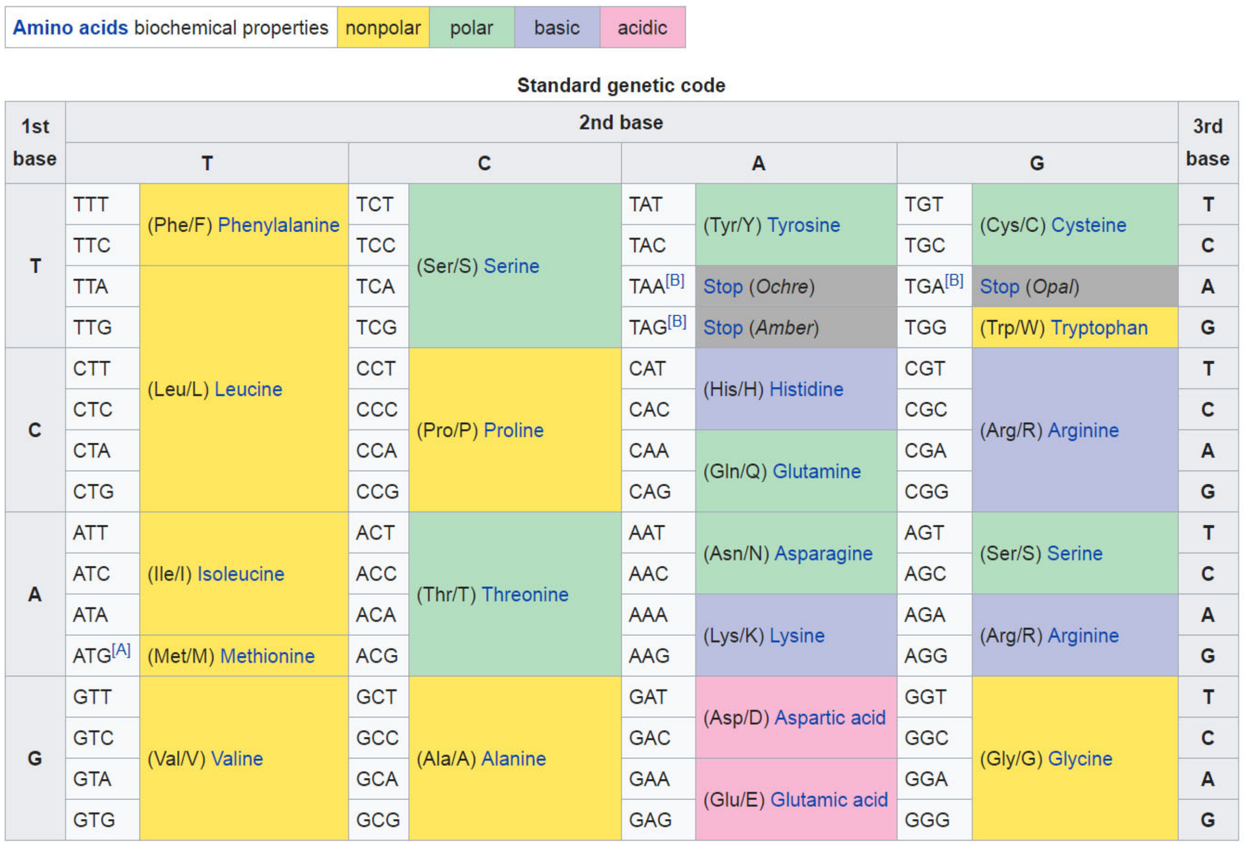
Substitution Matrices and Computational Complexity

Traditionally, evolutionary distance has been quantified by aligning sequences and comparing the results. Aligning multiple sequences has been shown to be an NP-Complete problem (Elias, Wang & Jiang, 2006). The worst case time complexity of multiple sequence alignment is the sequence length to the power of the number of sequences. A modest example of aligning 20 sequences of 10 amino acids long would have a time complexity of 10^{20}. This makes solving global optimal alignments of non trivial sequences completely intractable. For this reason heuristic methods were developed to search for multiple sequence alignments. There are a number of alignment algorithms which can be divided roughly into dynamic programming and word search families of methods. The most famous dynamic programming methods are the Needleman-Wunsch and Smith-Waterman. When aligning a pair of sequences these algorithms create a matrix with one sequence represented on each axis (Needleman & Wunsch, 1970). The matrix is then populated using a gap penalty, a match score, and a mismatch penalty. The match and mismatch scores are taken from a substitution matrix which is usually derived from empirical observation of sequences. When the matrix is populated the highest value path through the matrix is traced which informs where to insert gaps, if any, to align a pair of sequences. Given a perfect set of scores and penalties an optimal method will find the global best alignment for a pair of sequences. In practice no substitution matrix perfectly represents the probability of mutating from one nucleotide or amino acid to another but are sufficiently accurate for many purposes. Once aligned the distance between two sequences can be calculated by any arbitrary metric but it is most common to use the same scores and penalties used in the alignment process.

For greater than two sequences most algorithms aim to maximize the sum of pairwise alignments. Sum of pairwise score methods can still be computationally expensive and other methods exist which can overcome some of this complexity.

Introduction to Works

In this work two papers will be presented which explore evolutionary distance and alternatives to the standard substitution matrix and alignment methods. Both works rely on an N-Gram or K-mer approach in which a sequence is transformed into a feature vector by counting occurrences of patterns within the sequence. The first paper examines how an N-Gram approach can be used to create features for a classification task. In the largest experiment of its kind to date the Influenza virus is classified based on the host organism of a particular strain using a variety of classification models. The second paper explores how to accelerate the creation of these N-Gram feature vectors. It also examines how to speed up evolutionary distance calculations used in the construction of phylogenetic trees, or family trees.



Amino acids biochemical properties	nonpolar	polar	basic	acidic

Standard genetic code

1st base	2nd base								3rd base
	T		C		A		G		
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA[B]	Stop (*Ochre*)	TGA[B]	Stop (*Opal*)	A
	TTG		TCG		TAG[B]	Stop (*Amber*)	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

Figure 1.1: Amino Acid Nucleotide Encodings. Nucleotide to amino acid encodings. In *Wikipedia*. Retrieved August 21, 2018, from https://en.wikipedia.org/wiki/DNA_codon_table

CHAPTER 2

N-GRAM METHODS FOR INFLUENZA HOST CLASSIFICATION

Kincaid, J., Chandler Preliminary Version Published in Proceedings of the 2018 International

Conference on Bioinformatics & Computational Biology

Abstract

In this work feature vectors were created from protein data of the influenza virus using N-gram methods commonly used in text classification. Over 700,000 sequences were analyzed in total, creating one of the largest experiments of this type to date. A variety of classifiers were trained on these feature vectors. Their performance was compared in a task of predicting the influenza host organisms of corresponding viral strains. Random Forests, Linear Support Vector Machines, and Multilayer Perceptrons all achieved an accuracy of greater than 90% in a five fold cross-validation test. Multilayer Perceptrons performed the best with an accuracy of 97.2%.

Introduction

Influenza is one of the most successful viruses in nature. This success can be attributed to a number of factors including its highly contagious nature, large populations of host organisms, and ability to infect multiple species. Influenza is particularly notable for its fast antigenic drift and its ability to antigenic shift. Antigenic drift describes the evolution of virus antigens in response to the selective pressure of their host organisms immune systems. By changing the hemagglutinin and neuraminidase antigens, influenza can evade host immune systems which would otherwise recognize the virus from previous infections (Bouvier & Palese, 2008). This is also why influenza strains are named by their hemagglutinin and neuraminidase antigens (such as H1N1). Antigenic shift describes a process in which several strains of a virus can combine to create a new strain with features of each of its predecessors. Not all viruses are capable of antigenic shift and influenza is considered a canonical example of this phenomena. When a cell is infected with two different strains of influenza the antigens can be exchanged during the replication process potentially leading to a potent new strain with no existing immunity in the population (Zambon, 1999). Influenza is one of the most dangerous viruses that infect humans. The Spanish Flu of 1918 is thought to have killed 50 to 100 million people, close to 5% of the world population (National Archives and Records Administration). Influenza's potential for harm and ability to infect multiple species make it an important subject for studying mutation and disease vectors. Quickly determining the hosts that a particular strain can infect could help with estimating danger and inform quarantine procedures.

N-gram methods are a powerful set of tools for the analysis of sequences. In principal they involve taking a sequence and decomposing it into smaller pieces. This set of pieces is then used in the comparison to other sets of pieces. N-gram based techniques have been used in various computer science problems for many decades. Much of their initial success was focused around classifying written documents. N-Gram methods can be used to compare the syntactic similarity between words (Heer, 1974) or the content similarity of sentences, passages, or whole documents (Cavaner and Trenkle, 1994). In the former case the “gram unit” consists of groupings of letters or syllables that comprise a word. In the latter case the “gram unit” consists or words or groupings of words that comprise a document. The arity of an N-gram describes the number of pieces that comprise a feature. The frequently used Bag-of-words model is essentially a 1-gram model. This model takes no account of the order of terms but does count their frequency. The Bag-of-words model has been shown to be effective in some problems such as spam filtering where the ordering of terms is less important (Kolari et al., 2006). For other problems the ordering of terms can be an important carrier of information, thus the arity of N-grams becomes critical to preserving that information.

Several previous works have examined the use of machine learning techniques to classify the host organisms of influenza strains. Hidden Markov Models and Decision Trees were shown to have high accuracy in classify Influenza type A (ElHefnawi & Sheriff, 2014). Neural networks have also shown to be effective in classifying Influenza hosts (Attaluri, Chen, & Lu, 2010). In Attaluri, Chen, and Lu's work neural networks were trained on H1, H3, and H5 strain data using N-gram approaches on aligned and unaligned nucleotide data. Similar to the N-gram approach, Word Vectors have been used with a Support Vector Machine classifier to produce accurate results in a large dataset of over 150,000 sequences (Xu et al., 2017). The ElHefnawi, Attuluri, and Xu works have all shown reduced accuracy in predicting swine hosts in comparison to human and avian hosts across all methods. ELHefnawi & Sheriff suggest that their classifier performed worse in this category due to the swine hosts propensity for antigenic shift. It is also possible that swine host classification trends worse due to having fewer swine flue sequences available, creating a minority data class in comparison to human and avian strains. In this work a variety of classification techniques are used to classify the host organism of influenza strains. These classifiers include Random Forests, Naive Bayes, Linear Support Vector Machines, and Neural Networks.

Methods

Data

Data was taken from the NCBI FTP site. The data included a list of 730,106 influenza sequences in the FASTA format and a list of influenza sequence ID's with corresponding metadata. Influenza sequences were keyed and joined with their corresponding metadata to create a main dataset. The target variable of host organism had three categories: human, avian, and swine. These categories have a balance of 53%, 27%, and 20% respectively. This balance means that there were 386,956 human strains, 197,129 avian strains, and 146,021 swine strains.

N-gram Creation

Traditionally an n-gram counter takes a sentence and counts the occurrence of unique words or pattern of words. In the context of this experiment a word is a single amino acid residue and a sentence is a sequence of amino acids corresponding to an influenza virus. Patterns are created using a sliding window of fixed length of the whole of the sequence, thus patterns that overlap in the sequence are still counted. The n-gram vectors for this experiment were created using the Spark ML library. Six feature vectors were created corresponding to n-gram lengths of 1, 2, 3, 5, 10, and a final vector which contains all features from the other five vectors combined. A vocabulary limit of the one thousand most common features was used for all vectors except for the combined vector. For smaller pattern lengths the feature vector is not constrained by the vocabulary limit since the limit exceeds to total number of combinatorial possibilities for the given pattern length. For example, a pattern length of two can only produce 400 possible features given that there are only 20 kinds of amino acids observed. In every case the N-Grams comprise the input vectors for the classifiers and the target is the host organism.

Classifiers

Four classifiers were used in the two experiments. Parameters for each classifier were tuned to produce optimal results prior to their comparison. At least 15 runs were conducted with each classifier to tune these settings. The four classifiers used were Random Forests, Linear Support Vector Machines, Naive Bayes, and Multilayer Perceptron. Random Forests are an ensemble learner with decision trees as their base. A decision tree looks at features and splits data based on the maximum information gained by performing a split. It does this repeatedly to create additional splits and the overall decision tree. Random Forests use subsamples of the data for different decision trees and has these tree vote for a classification. Random Forests has the advantage that in practice it tends to perform better than decision trees in accuracy while also avoiding overfitting. A Linear Support Vector Machine works by examining data and looking for a series of vectors which linearly separate each feature according to the target feature. These vectors create an N-Dimensional hyperplane. The goal of the classifier is to maximize the distance of the hyperplane from the nearest instance of any given class. Naive Bayes treats all features as conditionally independent given a class feature. Naive Bayes looks at the probability of observing a given feature given a target feature and uses those probabilities to make a prediction given data. Multilayer Perceptron Classifiers are built by combining many perceptrons together to form a network. A perceptron is an extremely simplified model of a biological neuron which takes input and emits output based on an activation function. For this experiment a Random Forest classifier was created with 50 trees and a maximum depth of nine. A Support Vector Machine classifier was created with 500 iterations and a regularization parameter of 0.1. A Naive Bayes classifier was created with smoothing. All classifiers were created using the Spark ML library. A Multilayer Perceptron classifier was created using one hidden layer. Best results were obtained with a single hidden layer of identical proportion to the input layer and trained for 500 epochs. Accuracy was accessed after each training epoch and averaged across the five cross-validation folds. In each feature set accuracy peaked before reaching 500 epochs.

Hardware

Main experiments were conducted on a small cluster of five nodes, one master and four executors. Each node consisted of 12 AMD Opteron cores. Each node was allocated 20 gigabytes of memory but memory usage was not a significant factor for N-Gram creation and Classification.

Results

Random Forests had the best performance with 2-Gram features. Naive Bayes had the best performance with the all-feature combination but in general performed poorly in the classification task. Linear Support Vector machines had the best Performance with the all-feature combination however the 3-Gram feature results were very close. The Multilayer Perceptron classifier had the best results with the 2-Gram features. In general the all-feature combination performed very closely to best single feature category in each classifier. This is to be expected considering that each feature set is a subset of the all-feature combination. The best accuracy for Naive Bayes was 54.8%. The best Accuracy for Random Forests was 91.3%. The best accuracy for Linear Support Vector Machines was 92.8%. The best Accuracy for Multilayer Perceptron was 97.2%. The best Multilayer Perceptron model achieved a recall of 96.8% and a precision of 97.5%. Results for all classifiers were created using a five fold cross-validation.

Discussion

Accurate classification of influenza host organisms has been demonstrated for a number of machine learning classifiers. Xu et al. explored Support Vector Machines achieving accuracy in the mid 90%. ElHefnawi and Sherif found accuracy between 91.2% and 100% using Decision Trees. Attalari, Chen and Lu found accuracy between 91% and 98% using Neural Networks. In this experiment machine learning classifiers similar to the ones in all three works were tested and compared. Accuracy in this experiment was slightly lower than the accuracy reported in the respective works, however a direct comparison cannot be made given this dataset was considerably larger and less stringently curated than in previous works. With the exception of Naive Bayes, all classifiers performed well in the classification task with Multilayer Perceptrons achieving the highest accuracy. The recall and precision of the best Multilayer Perceptron model show that it is performing well on all three target classes considering the smallest minority class is 20% of the data. Multilayer Perceptrons likely performed the best out of the classifiers due to the increased complexity of this model compared to the other models. This complexity may allow it to pick up on features that the other models may have missed. The 2-Gram and 3-Gram feature sets performed consistently well on all the models. A question remains in why the 2-Gram Multilayer Perceptron model performed better than the all-feature combined model, given that the 2-Gram features are a subset. A possible explanation is that the all-feature Multilayer Perceptron network was not sufficiently large enough to learn all the relevant features. While the all-feature network was proportionately larger than the 2-Gram network in respect to the inputs this may not have been enough to capture all information in the feature space. These results suggest that a number of machine learning methods perform well in viral classification tasks and are capable of doing so at large scale using modest resources. The methods examined in this work could be important in many application areas in the future, in particular using the efficient creation of features through N-gram methods. Gene sequencing technology is becoming more affordable at a precipitous rate and the amount of genetic sequence data available is similarly expanding rapidly. Efficient and effective means for creating insights will be important in the future given this wealth of data. These methods could possibly be adapted for applications such as diagnosing and screening for genetic disorders, cancer research, personalized medicine, and many others.

References

Attaluri, Pavan K., Zhengxin Chen, and Guoqing Lu. "Applying neural networks to classify influenza virus antigenic types and hosts." *Computational Intelligence in Bioinformatics and Computational Biology* (CIBCB), IEEE Symposium, 2010.

Bouvier, Nicole M., and Peter Palese. "The biology of influenza viruses." *Vaccine* 26, 2008, D49-D53.

ElHefnawi, Mahmoud, and Fayroz F. Sherif. "Accurate classification and hemagglutinin amino acid signatures for influenza A virus host-origin association and subtyping." *Virology* 449, 2014, 328-338.

National Archives and Records Administration, “The Influenza Epidemic of 1918.”, National Archives and Records Administration, www.archives.gov/exhibits/influenza-epidemic/

index.html. Accessed 7/24/17.

Wang, Lusheng, and Tao Jiang. "On the complexity of multiple sequence alignment." *Journal of computational biology* 1.4, 1994, 337-348.

Xu, Beibei, et al. "Predicting the host of influenza viruses based on the word vector." *PeerJ* 5, 2017, e3579.

Zambon, Maria C. "Epidemiology and pathogenesis of influenza." *Journal of Antimicrobial Chemotherapy* 44. suppl 2, 1999, 3-9.

–

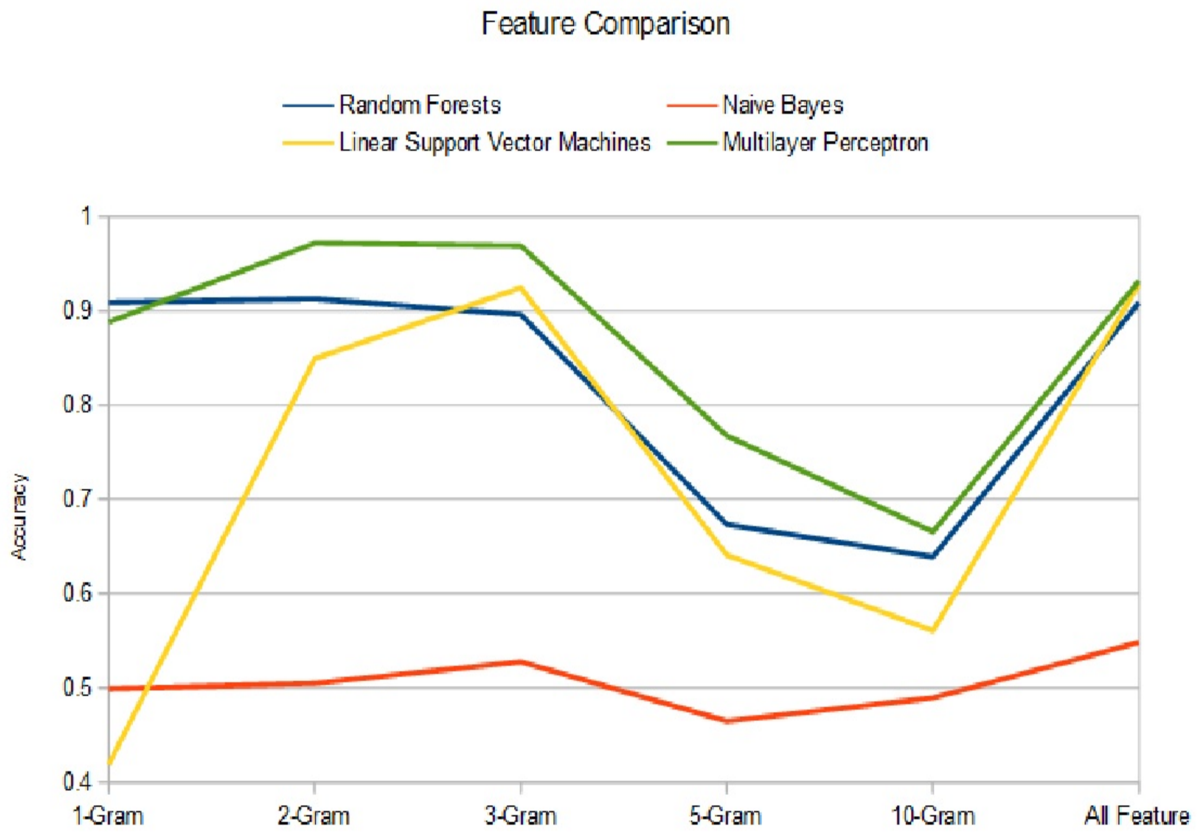


Figure 2.1: Classifier Accuracy by Feature Vector. Accuracy for all classifiers for each N-Gram length as well as the combined feature set.

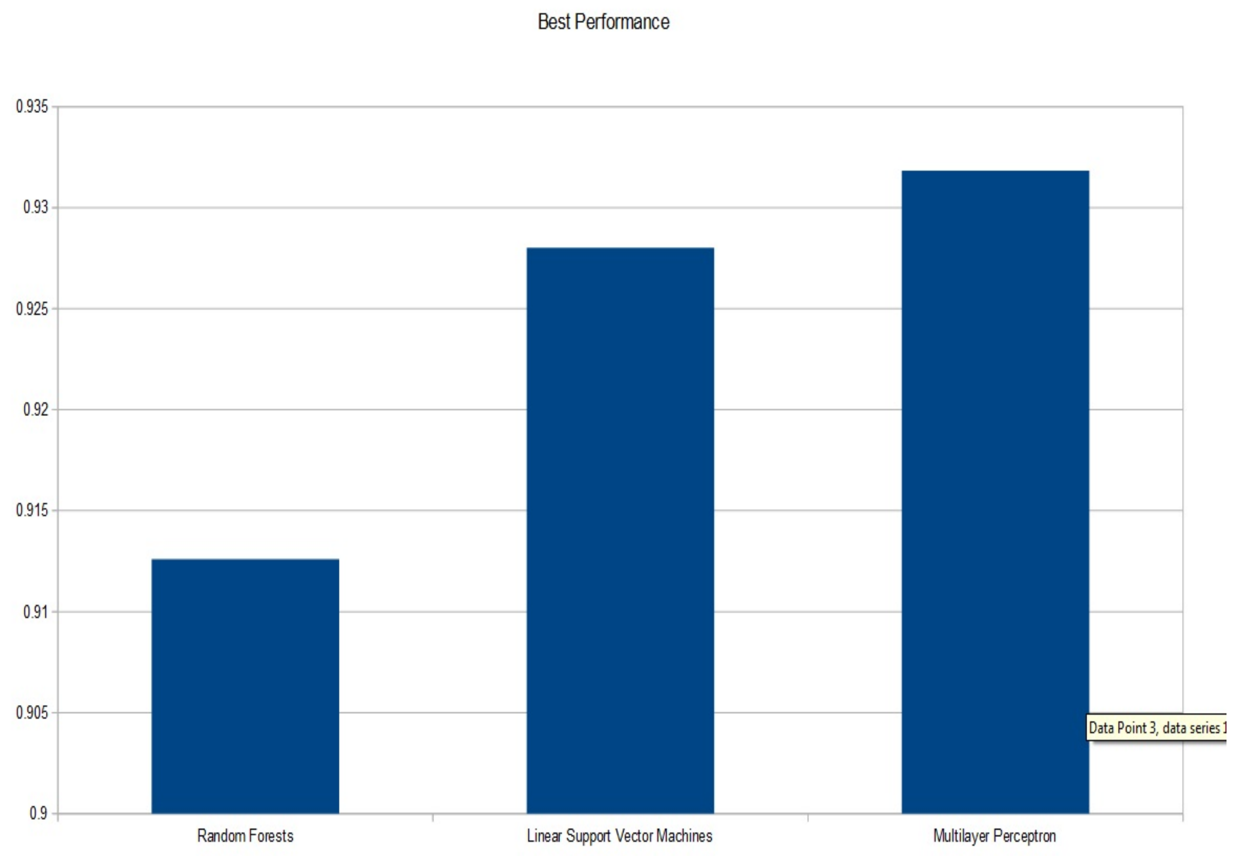


Figure 2.2: Top 3 Model Accuracy. The best performance for Random Forests, Linear Support Vector Machines, and Multilayer Perceptron.

	RF	NB	LSVM	MLP
1-Gram	90.9%	49.9%	41.9%	88.8%
2-Gram	91.3%	50.5%	84.9%	97.2%
3-Gram	89.6%	52.8%	92.4%	96.9%
5-Gram	67.3%	46.5%	64.1%	76.8%
10-Gram	63.9%	48.9%	56.1%	66.6%
All-Feature	90.9%	54.8%	92.8%	93.2%

Table 2.1: Percentage Accuracy of Classifiers by Feature. Random Forests, Naive Bayes, Linear Support Vector Machines, Multilayer Perceptron accuracy for each feature vector set. Average across five fold cross-validation.

CHAPTER 3

ACCELERATING ALIGNMENT FREE PHYLOGENETIC CONSTRUCTION

Kincaid, J. Chandler To Be Submitted To IEEE International Conference on Machine Learning and Applications 2018

Abstract

The process of constructing a phylogenetic tree from a set of genetic sequence data is a combination of two distinct problems, feature creation and phylogenetic construction. This work discusses the two problems and how they vary in complexity in terms of time and space. A data set of 5220 bacteria nucleotide sequences is used to benchmark performance. In the feature creation task a method is presented which completes the feature creation task in 74% of the time it takes the fastest standard Python library. When multi-processed to run on four cores this method is on average five times faster than the fastest standard Python library and up to twice as fast as a multi-processed standard library. In the phylogenetic construction task a method of GPU optimization is presented which is up to 33 times faster than CPU methods for distance matrix calculations.

Introduction

Problem Description

Phylogenetic construction is the task of creating a family tree from a set of organisms of unknown relationship to one another. In theory phylogenetic construction can use any feature of an organism for comparison such a physical trait but most commonly genetic information is used. From this genetic sequence data an evolutionary distance is estimated between organisms, and this distance is used to construct a phylogenetic tree. Estimating evolutionary distance happens to be a complicated, subjective, and often computationally difficult problem. Traditionally sequences are aligned to account for gaps, insertions, and changed nucleotides or amino acids. This alignment relies on a substitution matrix, which is a probability distribution for substitution, but these matrices do not generalize well to all alignment problems. Additionally, aligning many sequences at once is computationally intractable for relatively small numbers of sequences. To overcome this heuristic methods usually rely on maximizing the sum of pairwise alignment scores. Once alignment is achieved a distance score between sequences can be calculated which also involves some subjective choices in the scoring method. N-Gram methods provide an opportunity to reduce some of this computational complexity while providing a method that is generalizable to more problems than a method relying on a particular substitution matrix. N-Gram methods involve counting occurrences of subsequences in a sequence for a given subsequence length. The counts can be compiled into a numerical feature vector. Once a set of feature vectors is created from genetic sequences a number of methods can be used to estimate evolutionary distance. The most common approach is to calculate the Euclidean distance between two feature vectors. This approximation of evolutionary distance can be calculated between each organism in a set to form a distance matrix. This distance matrix is the core of many phylogenetic construction algorithms. Neighbor Joining, and many similar algorithms work by finding a minimum distance, often with some weighting factors, and assuming these two organisms share a common ancestor. From this point distance is calculated to this theoretical ancestor and the the process can be repeated iteratively until an entire phylogenetic tree is created.

Feature Creation

Five feature creation methods were explored which fall into 3 broader categories of methods including dictionary lookup, feature hashing, and binary decoders.

Dictionary lookup is by far the simplest of the feature creation methods explored and carries with it a number of advantages. In a dictionary lookup method a list of all observed vocabulary terms is created with each term corresponding to a specific feature index. This can either be done on the fly or with a predetermined dictionary if assumptions about the data are known. A document or sequence can be scanned and converted into an n-gram feature vector by matching n-grams in the document to vocabulary in the dictionary. For n-grams of length greater than one a new unique index can be created from the multiple indexes in the dictionary. In the context of text classification this creates a number of problems to be overcome, many of which become worse in large distributed systems. With text there may be situations in which a complete dictionary is not known beforehand and must be constructed on the fly. In a distributed system this means many nodes accessing a shared global variable which can create a bottleneck. For n-grams larger than one a unique index must be made from existing unary indexes, but this requires a dynamically resizable list of features with sparse representation, otherwise all previously computed feature vectors would have to be updated to reflect new terms. In a distributed system this also raises the problem of each node agreeing on what index a new feature should inhabit. This requires either an expensive broadcast operation to the other nodes or a reliable method of producing a new index such as hashing. This is why in many distributed systems the preferred method of n-gram feature creation is to use the hashing trick to start with and avoid expensive broadcast operations and dynamic list sizes. Genetic sequence data however has some unique properties that circumvent these problems. In nucleotide data there are only four unary terms: “A”, C”, “G”, “T” and for amino acid data there are only 20. For any practically usable n-gram length a complete dictionary of all combinatorial possibilities can be precomputed. For example a 2-gram nucleotide dictionary would consist of “AA”, “AC”, “AG”, “AT”, “CA”, “CC”... etc. In the case of a distributed system this dictionary can be broadcasted once to all nodes in the system and the feature space will be a known fixed length. If the dictionary data structure is properly optimized this lookup operation is essentially O(1). The only remaining computation is to find the appropriate index in the sequence's feature vector and increment it by one. This is a key advantage over other feature creation methods which require more computation per feature such feature hashing or binary decoding. The primary disadvantage is that the dictionary must fit into memory to be efficiently useful.

Hashing feature creation, often referred to as the Hashing Trick, aims to create a unique ID for any given feature without the need for a dictionary. Using various methods an n-gram is hashed and the resulting output should be a unique identifying integer which can be translated to an index in the feature vector. This method is particularly advantageous in a distributed system context. Since a dictionary is not needed, feature hashing avoids expensive broadcast operations between nodes in the system. Since each node is presumably using the same hashing function the unique identifier created for a given n-gram should be consistent across all nodes, and thus the feature vectors should align correctly. The primary disadvantages of feature hashing are computational complexity, increased complexity of implementation, hash collisions, and size of the resulting feature vector. Hashing feature creation relies on a hashing function which theoretically can be as simple as an O(1) operation in the case of an identity hash. In practice a necessary function requires a dramatic increase in computational steps from the simple O(1) operation of a identity hash or a dictionary lookup method. Implementing a hashing function with appropriate characteristics for this application is also far more complicated than implementing a simple dictionary method, fortunately well optimized libraries for appropriate hashing functions are available for many popular languages. Hashing functions also have the issue of hashing collisions. Hashing collisions occur when two different and distinct inputs produce identical outputs from the hashing function. Interestingly, hashing collisions have been used for estimating evolutionary distances by intentionally designing the hash function to collide when presented with similar inputs. In the context of feature creation, hash collisions are generally an undesirable aspect. Minimizing hash collisions often require a trade off between other desirable features and this presents a unique design challenge. The number of possible outputs of the hashing function can be increased to avoid hash collisions but this will generally increase the computational complexity. Distance calculations from resulting features typically rely on a dense representation of the feature vectors. This means that if the hash function output possibilities are increased, the overall size of features used in the phylogenetic construction algorithm may dramatically increase. Fortunately in the context of comparing genetic sequence data the size of the total number of n-gram combinations can usually be precalculated which can be used to reduce or eliminate hash collisions.

Binary decoders are a special type of demultiplexer whose function can be made from logic circuits or emulated through linear algebra operations. A multiplexer is a circuit or mathematical function that takes many inputs and combines them to be sent over fewer channels than the original input. A demultiplexer in contrast is a circuit or mathematical function that takes one or a few inputs and breaks a complicated signal into many simpler outputs. This behavior can be utilized to create feature vectors from binary data. In this application a nucleotide sequence can be considered a base four data scheme: “A”, “C”, “G”, “T”. From this scheme it is simple to convert nucleotide data into a binary format to be used in the binary decoder to create a feature vector. Consider the sequence “ACGT”: A reasonable conversion of this sequence would look like “00, 01, 10, 11” or “00011011” with each of the four possible nucleotides corresponding to a unique two digit binary number. From this binary scheme a one-hot style encoding can be derived using a binary decoder and these vectors added to each other to create a count of each n-gram seen in a given sequence. Two ways of constructing a binary decoder were considered including a hardware solution and a linear algebra solution. A hardware binary decoder, in this application sometimes called a 1-of-n binary decoder, is most simply an electronic circuit comprised of AND gates and logical inverters. By design the number of outputs for such a circuit will always be two to the power of the number of inputs. Binary decoders are cheaply available in standard integrated circuit packages but using IC's would greatly reduce the configurability and have to be purpose built for a particular n-gram schema. The more flexible approach is to mirror this kind of logic in a Field Programmable Gate Array. An FPGA allows configuration before processing, and a larger number of inputs and outputs to accommodate larger n-gram values. With a sufficiently advanced FPGA many n-gram features could be processed in parallel simultaneously, potentially providing a significant speedup from a dictionary lookup method with virtually no memory overhead. Binary decoders will likely not provide a speed up for very large order n-grams because of the complexity of the output scales exponentially with the complexity of the input. FPGA's also have the disadvantage of requiring special expertise and hardware to implement and a thorough investigation of their performance is beyond the scope of this paper. Fortunately their function can be emulated through a linear algebra operation which can be optimized for operation on a Graphical Processing Unit. The input vector can be multiplied by several masking vectors and summed to achieve the final desired output. This method of binary decoding also allows for many n-gram features to be processed in parallel simultaneously and carries the same exponential penalties for large order n-grams. Preliminary testing was done with such a function but proved to be less efficient than other methods in practice. While actual computation of the feature vector was exceedingly fast, the time introduced from transferring data from system RAM to VRAM on the GPU proved to be so slow as to negate any performance advantage for this method. Using other hardware combinations or using a more well optimized library for GPU calculations could potentially eliminate this drawback and warrants further investigation.

Phylogenetic Construction

Once a representative feature vector is created from a genetic sequence an algorithm is necessary to construct a resulting phylogenetic tree. The computational complexity of the process is both different from the feature creation and hugely dependent on the size of the feature space. As seen in Figure 3.3 and Table 3.3 increasing the order of n-grams results in an exponential increase in the size of the dense feature vector data. Increases in the number of sequences naturally results in a linear increase in the size of the feature vector data. Most phylogenetic construction methods rely on constructing an initial distance matrix between each sequence in a data set. Some heuristic versions of popular algorithms compute some subset of this distance matrix and consequentially trade certain algorithmic guarantees for reduced complexity. Unlike the feature vectors, the distance matrix size increases exponentially with both the number of sequences and the order of of n-gram size. Two popular families of algorithms for phylogenetic construction are Neighbor Joining, and Pair Group Method Arithmetic Mean, or PGMA. Both families are very similar in their process but vary based on some theoretical guarantees. Neighbor Joining has the important guarantee that the phylogenetic output will be correct given one condition, that the input distance matrix is nearly additive. Nearly additive means that each distance in the matrix must differ from the true evolutionary distance by less than half of the shortest branch length in the phylogenetic tree. In practice the input distance matrix rarely reflects the actual evolutionary distance but Neighbor Joining tends to perform quite well despite this. The PGMA family including its weighted and unweighted variants work similarly to Neighbor Joining but have a few key distinctions. The PGMA algorithms work better under a constant evolutionary clock model, meaning that they perform well when evolutionary distance between sequences is reflective of real evolutionary distance and that the change rate in distance is constant over time. PGMA also produces a rooted tree where Neighbor Joining does not. The weighted version creates distances to a newly joined node by taking a straight 50-50 average between the distances of its two child nodes. The unweighted version averages distances in a way that takes into account the distances to the children of the child nodes being joined in an equal fashion. Both families of algorithms run in $O(n^3)$ time in their unadulterated form. $O(n^2)$ of that time is taken up by computing the initial n by n distance matrix which means that any speedup in this process will translate into a significant overall speedup of either family of algorithm.

Related Works

The N-Gram approach to phylogenetic construction is also commonly referred to as an Alignment Free approach or a Composition Vector Approach. The term Alignment Free refers to forgoing the need to use the common multiple sequence alignment processess and the Composition Vector term refers to the feature vectors that are created by the N-Gram process. While this method has been replicated on a variety of datasets its introduction is relatively new compared to traditional methods. One of the first examples of this approach was called k-string composition and was used to create a phylogenetic tree from 109 prokaryote genomes that agreed with biological domain knowledge (Wang & Hao, 2004). An early and popular software package for this method is CVTree (Composition Vector Tree) developed and hosted by the Bejing Institute for Genomics and Fudan University, Shanghai. The term CVTree is sometimes used interchangeably for the specific software package or generalized methods based on Composition Vectors. This software or similar methods have been used in a number of experiments on different data. In a set of 432 prokaryote organisms CVTrees found general agreement with manual biological models (Gao et al., 2007). Disagreement with the biological was described as known points of interest or contention amongst biologists implying that CVTrees may be able to resolve errors in manual classification or provide new insights. This results was later extended to a set of 892 prokaryotes with experiments in fungi and virus genomes (Qiang, Zhao, & Bailin, 2009). Disagreement with biological models was further explored in the context of Shigella, a type of bacteria closely related to E. coli. Early molecular and phenotype testing described Shigella as a subspecies of E. coli, but CVTree results showed several forms of Shigella to all be members of a distinct species within the same genus as E. Coli (Guanghong, Zhao, & Bailin, 2012). As of this writing Shigella is considered even more distant to E. coli and has been placed within it's own genus. These results show at least an ability to call attention to aspects of the biological model which need further refinement.

As of the writing of this work some research has been conducted in accelerating distance matrix calculation using parallel computing and little research has been done in accelerating N-Gram feature creation for this application. GPU acceleration using the CUDA library has been shown to be effective in speeding up multiple sequence alignment and Neighbor Joining based on such an alignment (Liu, Smidcht, & Maskell, 2009). In the Neighbor Joining task a GPU was used to perform calculations to populate the distance matrix in parallel. In the combined operation of multiple sequence alignment and resulting Neighbor Joining GPU acceleration showed a 26x speed up over serial computation. The traditional multiple sequence alignment and phylogenetic construction process has also shown to benefit from parallelization in the form of distributed computing. A near linear speed improvement was shown when using multiple GPU's for the Unweighted Pair Group Method with Arithmetic Mean (Hua et al., 2017). UPGMA is a phylogentic construction algorithm that is very similar to Neighbor Joining but differs slightly in how it weights its distance matrix calculation and subsequent distances to newly created nodes. To date only one work has experimented with accelerating N-Gram based phylogenetic construction with distributed computing (Xu, Ji, & Zhang, 2017). In a single core comparison their work was slower than the CVTree package taking 24508 seconds to execute versus CVtree's 18093 seconds. They showed a near linear speed increase when adding additional CPU cores and executed the construction task in 2799 seconds when using 15 cores. While accelerating N-Gram based phylogenetic construction is a recent undertaking, previous work suggests that both the feature creation and distance matrix calculation processes may benefit greatly form parallel processing.

Methods

Data set

A list of genetic sequence accession numbers were gathered from the supplemental resources Xingjian et al used in their performance benchmarks for phylogenetic construction. The nucleotide sequences were downloaded with NCBI's Entrez utility using the list of accession number. In total 5220 nucleotide sequences were used from a diverse sampling of bacteria.

Feature Creation Methods

Three feature creation methods were benchmarked including dictionary lookup, Collections Counter, Scikit-Learn CountVectorizer with and additional two multi-processed versions of dictionary lookup and Collections Counter. Scikit-Learn CountVectorizer is a word counting function with an optimized back end that can process a variety of data. In this experiment the function was set to analyze character permutations and given a precomputed vocabulary to work with. To take full advantage of CountVectorizer it was fed batches of sequences at a time to reduce data transfer time and overhead. Batch size was varied in each test to properly saturate memory and produce the fastest possible computation time. Collections Counter is a function from the Python Collections library which has also been heavily optimized. In this experiment a list of all n-grams in a given sequence is computed then fed to the Collections Counter function to produce a feature vector. The dictionary lookup method relies on precomputing an exhaustive vocabulary which takes the from of a Python dictionary data type. The keys of the dictionary are the expected n-gram strings and the values are the indexes of the corresponding feature vector which are incremented when an n-gram is observed.

Distance Matrix Calculation

MXNet was used to created a distance calculation algorithm that runs on both CPU and GPU.

Both the CPU and GPU version of the distance matrix use the same exact algorithm and code save one detail, a context variable which changes the data location and engine of execution. To minimize latency and eliminate disk read times as a confounding factor the entire set of feature vectors is held in memory. From this set a matrix is constructed with distances being calculated on a per row basis using a tiling feature to compare a given row vector to a set of column vectors. In the case of CPU processing the distances in a row are calculated sequentially. Due to the GPU's parallel processing abilities it can process and entire row of distances roughly simultaneously. In all experiments distance was calculated as the euclidean distance between two feature vectors.

Hardware

All experiments were benchmarked on a Sandy Bridge core i5 processor with 8gb of RAM. The GPU used was and Nvida GTX 760.

Results

In the feature creation task the dictionary lookup method and its multi-processed counterpart performed substantially better than the fastest standard Python libraries. The Collection Counter library shows a pattern that its time complexity grows with higher n-grams faster than any other method. In the 2-gram task the Collection Counter library is actually faster than Scikit-learn CountVectorizer but it is quickly outpaced when higher n-grams are considered. Multiprocessing a method shows a close to linear speedup in both the dictionary lookup method and the Collection Counter library with the multi-processed methods being almost four times faster corresponding to four cores utilized instead of just one. In the 2-gram, 4-gram, and 6-gram tasks the dictionary lookup method completes in 74.2%, 73.1%, and 76.2% of the time of the next fastest single process method respectively. The same comparison between multi-processed dictionary lookup and multi-processed Collection Counter yields 72.7%, 51.7%, and 42.7% respectively.

In the distance matrix construction task the GPU variant shows increased acceleration with higher order n-grams. In the 2-gram task the completion times are almost the same. This is expected because the computational difficulty of the 2-gram task is relatively small given that feature vectors are only 16 features long and the GPU variant of the algorithm requires data transmission time for system RAM to GPU VRAM. In comparison the 4-gram features are 256 features long and the 6-gram feature vectors are 4096 features long. This exponential increase in computational difficulty favors the higher parallelization possible through GPU optimization and has diminishing penalties for moving larger amounts of data from RAM locations.

Discussion

In this work is presented a number of methods to increase the speed of phylogenetic construction including feature creation and distance matrix calculation. In the feature creation task the dictionary lookup method proved to be significantly faster than the fastest standard Python libraries across all tested n-gram sets. In the distance matrix calculation task a GPU optimized algorithm was shown to be up to 33 times faster in higher order n-gram feature vector sets. All new methods provided in this work were specifically designed to be scaleable using either multiple processing cores, or using libraries which support distributed multi-node computation. Future work could use these functions to replicate other results which show a near linear speedup when adding nodes to a distributed system designed for phylogenetic construction (Xu, Ji, & Zhang, 2017). Two new methods of feature creation were theorized in the paper including hardware binary decoders and linear algebra binary decoders. Theses two methods could be tested in future works and have the potential to speedup feature creation time from the best results described in this work.

References

Atteson K. "The performance of neighbor-joining algorithms of phylogeny reconstruction". *In: Jiang T., Lee D.T. (eds) Computing and Combinatorics. COCOON,* 1997, *Lecture Notes in Computer Science, vol 1276. Springer, Berlin, Heidelberg*

Gao, L., Qi, J., Sun, J. et al. "Prokaryote phylogeny meets taxonomy: An exhaustive comparison of composition vector trees with systematic bacteriology:. *Science in China Series C,* 2007, 50: 587. https://doi.org/10.1007/s11427-007-0084-3

Guanghong, Zuo., Zhao, Xu., Bailin, Hao. Shigella. "Strains Are Not Clones of Escherichia coli but Sister Species in the Genus Escherichia." *Egyptian Journal of Medical Human Genetics*, Elsevier, 29 Dec. 2012, www.sciencedirect.com/science/article/pii/S167202291200112X.

Hua, G.J., Hung, C.L., Lin, C.Y., Wu, F.C., Chan, Y.W., & Tang, C. Y. "MGUPGMA: A Fast UPGMA Algorithm With Multiple Graphics Processing Units Using NCCL". *Evolutionary Bioinformatics Online,* 2017, 13, 1176934317734220. http://doi.org/10.1177/1176934317734220

Qi, J., Wang, B. & Hao, BI. "Whole Proteome Prokaryote Phylogeny Without Sequence Alignment: A K-String Composition Approach". *J Mol Evol,* 2004, 58: 1. https://doi.org/10.1007/s00239-003-2493-7

Qiang, Li., Zhao, Xua., Bailin, Hao.“Composition Vector Approach to Whole-Genome-Based Prokaryotic Phylogeny: Success and Foundations.” *Egyptian Journal of Medical Human Genetics*, Elsevier, 29 Dec. 2009, www.sciencedirect.com/science/article/pii/S0168165609005896.

Xingjian Xu, Zhaohua Ji, Zhang Zhang. "CloudPhylo: a fast and scalable tool for phylogeny reconstruction". *Bioinformatics*, 2017, Volume 33, Issue 3, Pages 438–440, https://doi.org/10.1093/bioinformatics/btw645

Y. Liu, B. Schmidt and D. L. Maskell, "Parallel reconstruction of neighbor-joining trees for large multiple sequence alignments using CUDA," *2009 IEEE International Symposium on Parallel & Distributed Processing, Rome*, 2009, pp. 1-8.
doi: 10.1109/IPDPS.2009.5160923

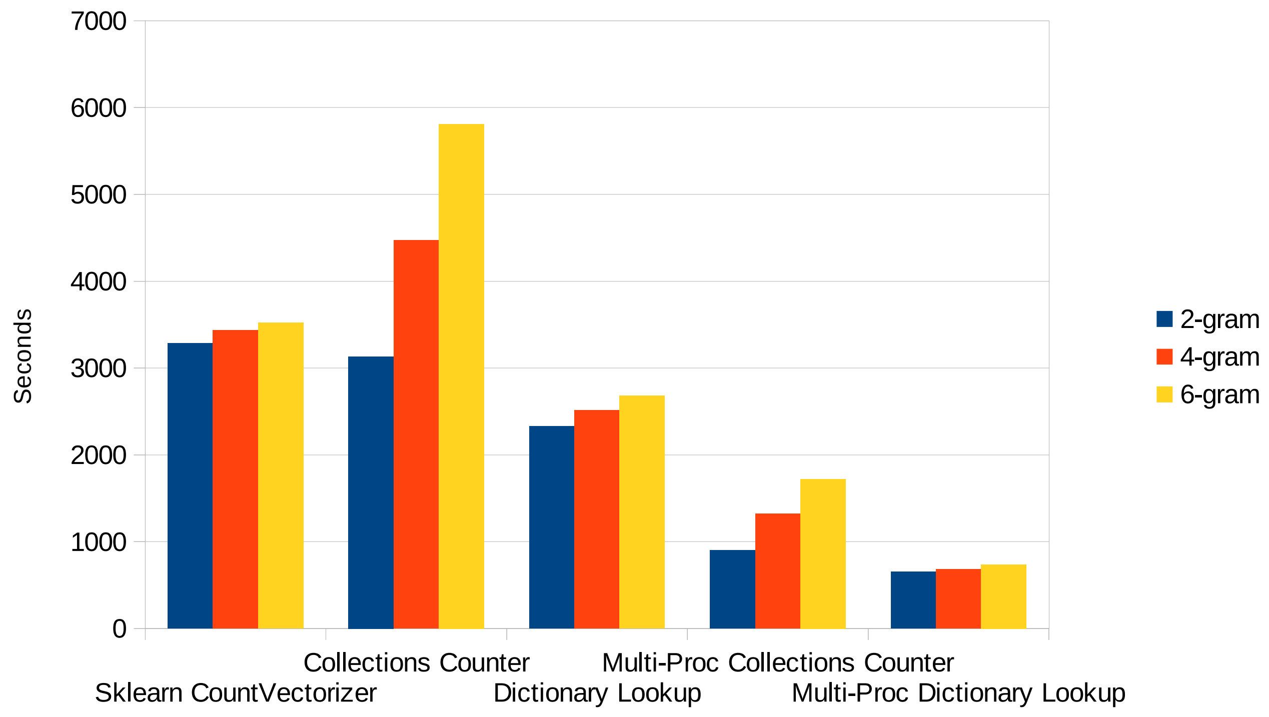


Figure 3.1: Feature Vector Creation Time. Five methods of creation a feature vector and the time they took to process 5220 sequences into 2, 4, and 6-gram vectors. Time in seconds.

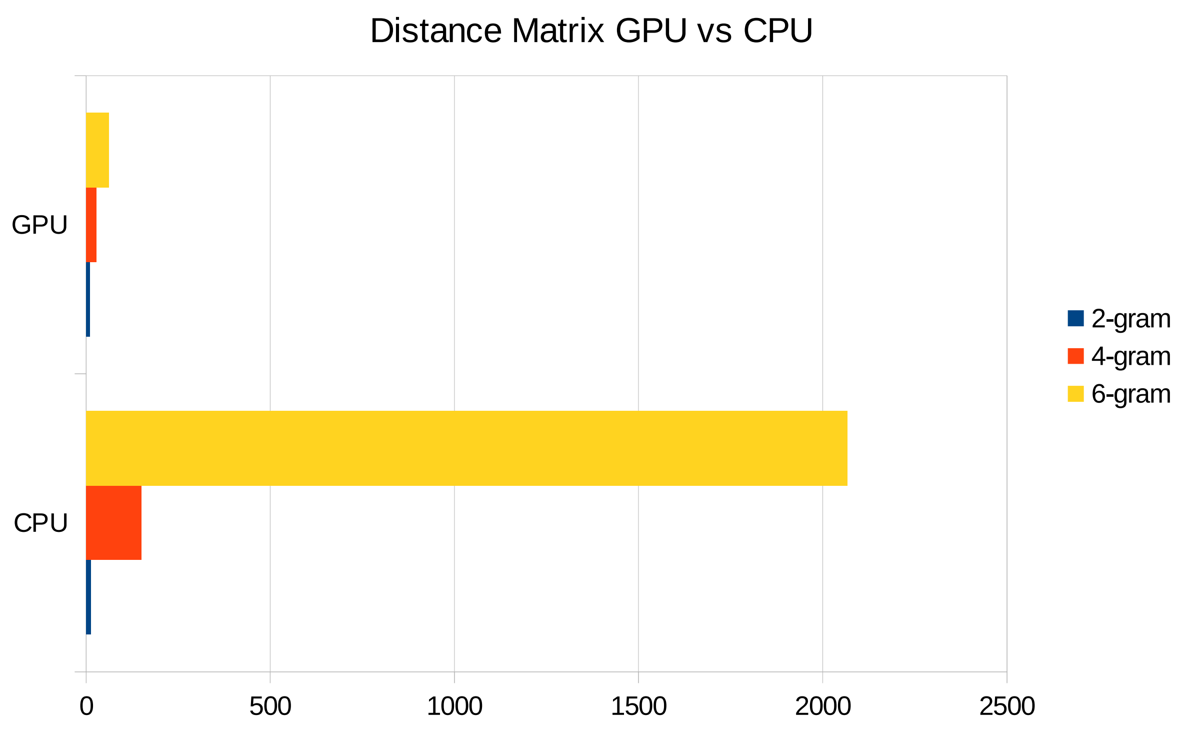


Figure 3.2: Distance Matrix GPU vs CPU. Time to calculate the distance matrix between 5220 sequences for 2, 4, and 6-gram vectors comparing CPU to GPU. Time in seconds.

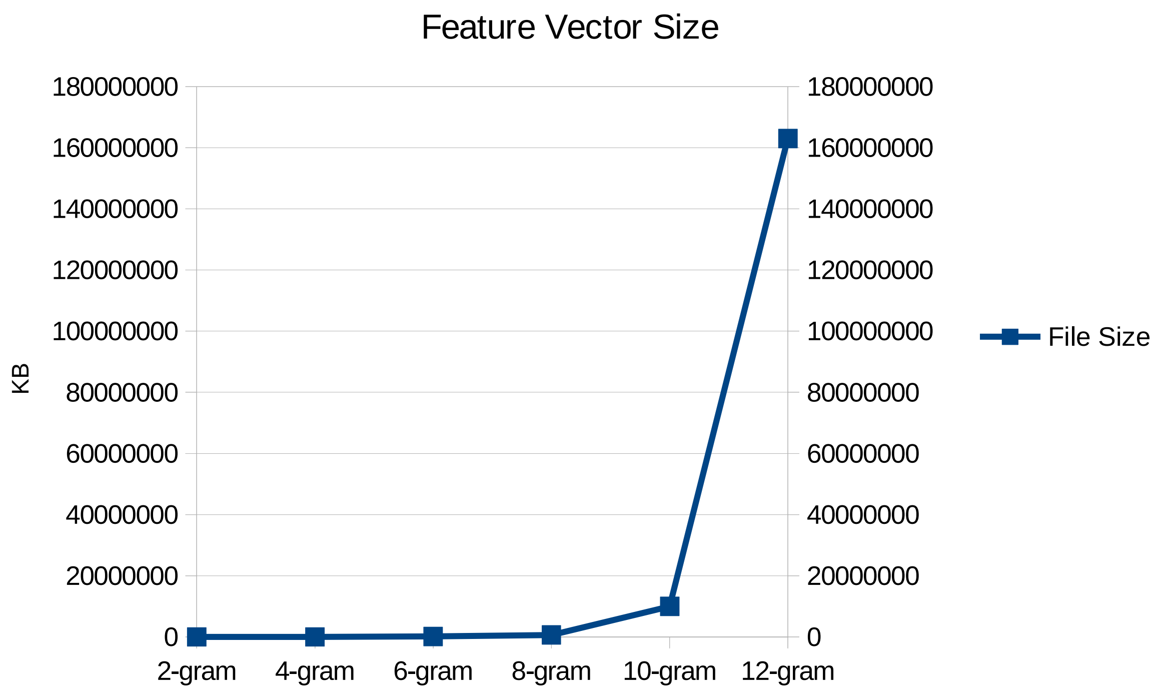


Figure 3.3: Feature Vector File Size. Dataset when converted into a feature vector comparing 2, 4, 6, 8, 10, and 12-gram sizes in Kilobytes.

	2-gram	4-gram	6-gram
SKLearn CountVec	3287	3438	3519
Collections Counter	3132	4473	5810
Dictionary Lookup	2328	2514	2680
MP Collections Counter	899	1321	1723
MP Dictionary Lookup	654	683	736

Table 3.1: Feature Vector Creation Time. Five methods of creation a feature vector and the time they took to process 5220 sequences into 2,4, and 6-gram vectors. Time in seconds

	2-gram	4-gram	6-gram
CPU	12	150	2067
GPU	10	28	61

Table 3.2: Distance Matrix GPU vs CPU. Time to calculate the distance matrix between 5220 sequences for 2, 4, and 6-gram vectors comparing CPU to GPU. Time in seconds.

	2-gram	4-gram	6-gram	8-gram	10-gram	12-gram
File Size	362	3510	163000	656000	10000000	163000000

Table 3.3 Feature Vector File Size. Dataset when converted into a feature vector comparing 2, 4, 6, 8, 10, and-12 gram sizes in Kilobytes.

CHAPTER 4

CONCLUSION

Many problems in biology and particularly in genetics are extremely complex and often computationally difficult. Advances in bioinformatics have shown that a number of these problems are good candidates for machine learning methods.This work presented two papers in which machine learning techniques were used to analyze protein information. In the first paper feature vectors were created from protein data of the influenza virus using N-gram methods common to text classification. A number of classifiers were trained on the feature vector and were successful in predicting influenza host organisms of corresponding viral strains. A Multilayer Perceptron Classifier acheived the highest accuracy with 97.2%. The second paper explored accelerating feature creation and distance matrix calculation. A Dictionary Lookup method was detailed which outperfroms two standard python libraries in both single core and multi-processed tasks. In single core tasks the Dictionary Lookup method averaged a 25% reduction in computation time over the leading standard Python library. Similar improvements were also seen in the Distance Matrix calclulation tasks. The Distance Matrix calculation is an essential portion of the Neighbor Joinging family of algorithms and many other phylogenetic construction algorithms. Showing a performance improvement in this calculation has great significance. GPU optimized distance matrix calculation was shown to be on par with CPU calculation at trivial N-Gram lenghs but acheived up to a 33x speed increase in the 6-gram task.

REFERENCES

Budd, Graham E. “At the Origin of Animals: The Revolutionary Cambrian Fossil Record.” *Current Genomics* 14.6, 2013, 344–354. *PMC*. Web. 22 Oct. 2017.

Butterfield, Nicholas J. “Macroevolution And Macroecology Through Deep Time.”*Palaeontology*, vol.50, no. 1, 2007, pp. 41–55., doi:10.1111/j.14754983.2006.00613.x.

Cavanar, W. B. & Trenkle, J. M. “N-Gram Based Text Categorization.” P*roceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994, pp161-175. Las Vegas, US.

Elias, Isaac. "Settling the intractability of multiple alignment." *Journal of Computational Biology,* 2006, 1323-1339.

The Encode Consortium, “An integrated encyclopedia of DNA elements in the human genome” *Nature*, vol. 489, no. 7414, May 2012, pp. 57–74., doi:10.1038/nature11247.

Heer, T. De. “Experiments with syntactic traces in information retrieval.”*Information Storage and Retrieval*, vol. 10, no. 3-4, 1974, pp. 133–144., doi:10.1016/0020-0271(74)90015-1.

Kolari, P., Java, A., Finin, T., Oates, T., Joshi, A. *AAAI'06 proceedings of the 21st national conference on Artificial intelligence*, 2006, Vol. 2, pp.1351-1356.

Miyazawa, Sanzo. “Superiority of a mechanistic codon substitution model even for protein sequences in Phylogenetic analysis” *BMC Evolutionary Biology Vol 13*-257, 2013, doi:org/10.1186/1471-2148-13-257

Needleman, Saul B., and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of molecular biology* 48.3, 1970, 443-453.

Ponting, Chris P. “What fraction of the human genome is functional?” *Genome Research*, Cold Spring Harbor Lab, 1 Jan. 1970.

Venkatesh, Byrappa., Gilligan, Patrick ., Brenner, Sydney. “Fugu: a compact vertebrate reference genome” *FEBS Letters*, Vol. 476, no. 1–2, 2000, pp. 3-7, ISSN 0014-5793, doi:10.1016/S00145793(00)01659-8.