VERTICALLY INTEGRATED

PROJECT (VIP) REPORT

NEUROMORPHIC SOLUTIONS TO HIGH DIMENSIONAL DNA SUBSEQUENCE MATCHING VIA K-MER WISE ALIGNMENT

*Ethan Jacob Moyer*

ENGR 370: Applications, Algorithms, & Architecture for

Neuromorphic Computing

*Dr. Anup Kumar Das*

Drexel University

4 May 2020

**INTRODUCTION**

BACKROUND:

Sequence alignment of biological molecules, such as deoxyribonucleic acid (DNA), focuses on identifying regions that are of high similarity due to common factors. Typically, these factors are derived from functional, structural, or evolutionary relationships shared between the sequences being aligned (Mount). More specifically, the field of bioinformatics is concerned with sequence alignment algorithms as longer sequences become increasingly difficult to align by hand. One such algorithm is Pairwise Sequence Alignment (PSA). In this algorithm, each element in a query sequence (the sequence being aligned) is compared each element of a reference sequence (the sequence to which the query is being aligned). Although it has mainly two different dynamic programming implementations, such as through the Needle-Wunsch (NW) and Smith-Waterman (SW) techniques, the general theme of alignment remains constant (Needleman) (Smith). A positive score is assigned to matching elements and a negative score is assigned to nonmatching elements. In this way, the optimal alignment between a query and a reference is identified by the highest scoring segment (Salamat). Different scoring matrices can be used to alter these scores. For instance, the NUC44 nucleotide scoring matrix is commonly used in MATLAB as it takes into account all ambiguous and unambiguous base pairs (MathWorks, “NUC44”).

The main focus of this research only involves the nucleotide similarity between sequences. As a result, the project is structured around identifying sequence matches rather than sequence alignments, which as previously mentioned requires evolutionary data. The scope of this research operationally defines subsequence matching as the matching of k-mer sequences based entirely on their percent similarity with respect to PSA scores. Since the SW technique focuses primarily on local alignment while the NW technique focuses on global alignment, the former will be used as the basis for alignment scores. Through the use of SW technique, k-mer wise subsequences can be matched on the basis of a 95 percent similarity threshold; more simply, if when aligning with respect to a reference k-mer, a query k-mer scores equal to or higher than 95 percent of the score when the reference is aligned to itself, then it is considered to be matched.

RATIONAL:

Originally, the premise behind this research was to determine where a match exists in a sequence of k-mers. This was thought to correspond to the position of a match, which would serve as an intrinsic measurement for the matching of two k-mers. Since each match is based on a 95 percent similarity threshold, these k-mers can be classified into classes. Analogous to equivalence classes, which in mathematics are similar sets when an equivalence relation is imposed on them, these classes serve as groups that house k-mers that are at least 95% similar.

It was hypothesized that ‘dictionary’ files could contain disjoint information about the appearance of subsequences that are at least 95% similar, which would indicate where matches would occur in a given reference sequence. This concept was mainly tackled with careful curation of the data and the use of Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs).

ANALOGY:

An analogous problem is asking to classify similar words across repeats of the same sentence. This might be based on high dimensional quantifiers, such as properties of being a noun, verb, conjunction, etc., context indicators that the English language takes for granted, such as verb-noun agreement and implied antecedents, or words that simply describe similar things. Using context cues, a computer could generate disjoint data for different words. For instance, one sentence copy would be classified solely based on a specific class of closely related nouns, such as the noun ‘CAR.’ An algorithm might place different subtypes of cars into that word class, like ‘TRUCK’ or ‘VAN.’ A separate sentence copy would be classified based on another English quantifier or context indicator. In this way, sentence data would be organized into discrete classes, each containing disjoint information about its respective word group. An output file would then contain TRUE cases whenever a query word exists in the same class as a reference word, corresponding to where in each sentence these words appear. As one could imagine, the task of classifying words is trivial for short sentences and when there are relatively few classes. The task becomes more intensive once both of these attributes increase.

Such an increase is observed when applying this problem structure to DNA subsequence matching. It is similar to the word classification problem for the English language in that both problems are based on text classification. One difference between the two is the convention by which text is interpreted. In English, sentences are composed of words, each of which is comprised of an arrangement of 26 unique letters. In the genetic language, these sequences are composed of a composition of four unique bases. This distinction between *an arrangement* in the English language and *a composition* in the genetic language is imperative of the increased complexity when applying word classification to DNA subsequence classification.

An arrangement is operationally defined as grouping of characters where only specific permutations carry meaning. For example, the word ‘ACR’ has little meaning compared to ‘CAR’ and ‘ARC.’ A composition is operationally defined as a grouping of characters where generally all permutations may carry meaning. In the case of DNA nucleotides, ‘ATCAGTC’ caries as much meaning as ‘TTCCAA.’ As displayed by the provided examples, the concept of an arrangement pairs nicely with English words and the concept of a composition pairs nicely with the genetic language. In most cases, the length of subsequences is much greater than the length of words. As previously mentioned, it becomes increasingly difficult to classify subsequences as the length of each subsequence and the number of possible classes increases. For these reasons, DNA subsequence matching is much more complex than language word classification; nonetheless, high dimensional word classification serves as a perfect analogy from which DNA subsequence matching can be built.

Moreover, the need for the neural network in the analogous word classifying problem is strikingly similar to the problem in DNA subsequence matching. If the number of quantifiers is low, there really is not much need for a network to do prediction. However, as these quantifiers increase, the complexity of classifying words—or in the case of this research, subsequences—increases exponentially.

**DATA**

STRUCTURE:

In order to use machine learning to predict protein alignment, it was postulated that there needs to be two main aspects of the data and model. The first hypothesized aspect was that the data needs to somehow resemble alignment of sequences. It is far too difficult to generate alignment scores or positions based off a singular sequence. Therefore, the main focus for the output of the model was the positions of the most optimal alignment instead of specific alignment scores. In order to resemble optimal alignment positions in the data, the sequences were split up into sequentially appearing k-mer segments, or subsequences of k long molecules, each one starting one more nucleotide to the right than the preceding one. The position of each subsequent k-mer in a list represents the position at which it appears in the sequence. Structured in a table, the first column is this list of k-mer sequences. The second column is a logical list identifying where a selected sequence appears in the list of k-mer sequences. In this way, the position of logical TRUE values in the list identify where an alignment occurs in the sequence because of the one-to-one correspondence between the logical value list and the list of k-mer subsequences.

The following figure illustrates the basic set up of the data structure that was fed through the LSTM model

**Figure 1:** Entry Data Structure

Query (Sq)

|  |  |
| --- | --- |
| Reference (Sr) | Logical (L) |
| k0 | m0 |
| k1 | m1 |
| . | . |
| . | . |
| . | . |
| ki | mi |
| . | . |
| . | . |
| . | . |
| kn-1 | mn-1 |
| kn | mn |

where ki represents an individual k-mer segment of the reference sequence Sr and mi represents the logical value of a match of the query sequence Sq against ki.

COLLECTION:

The DNA sequence that was generated to create the training, test, and validation data sets was obtained using the *randseq* function from the MATLAB Bioinformatics Toolbox. In a custom MATLAB function, *generateDNA*, *randseq* generates a sequence of length n and saves it to a specified file, ‘dna.txt.’ In another custom function, *gatherData*, a cell character array of short k-mers are generated for a certain small length. These k-mers are then loop through in sequence and fed into another custom function, *generateTable*, with the name of the sequence stored in ‘dna.txt,’ the cell character array of k-mers, the length of the generated k-mers, and an alpha value that corresponds to the 95% percent similarity in the local alignment matches between sequence. In the *generateTable* function, a table of sequence matches for each k-mer given to the function is generated by aligning the given k-mer with the sequence of k-mers at 95% similarity. Specifically, through the use of the MATLAB Bioinformatics Toolbox, a modified (SW) alignment technique is used through the function *localalign* in order to perform the alignments of each k-mer in *generateTable* (Barton). The result of each iteration is a structure, which is represented in Figure 1, of k-mers and their respective Boolean matching values when compared to k-mers already existing in the sequence. In this way, only data that is applicable to the sequence is generated and included in the model data sets. Duplicate data is removed automatically.

Barton, G. (1993). An efficient algorithm to locate all locally optimal alignments between two sequences allowing for gaps. CABIOS 9, 729–734.

PIPELINE:

The following diagram displays the data pipeline for the paradigm of how the model should function

**Figure 2:** Data Pipeline

S0, S1, … , Sj, … , Sn-1, Sn

Model trained on specific set at 95% similarity

L0, L1, … , Lj, … , Ln-1, Ln

where Sj represents a list of k-mer fragments of a unique gene sequence and Lj represents a list of logical values corresponding to the occurrences of those k-mer fragments within the specific set.

CURATION:

This data curation along with other data analysis were carried out with several MATLAB functions built to efficiently separate and analyze the data. These functions along with their documentation are included on the project’s GitHub repository.

**MODEL**

PROPOSED MODEL:

The structure of the data leads into the second hypothesized aspect: the use of a LSTM RNN) based model. Since the list of k-mer subsequences is structured like time series, LSTM is arguably the best model for this format of data (Münch).

**Figure 3:** Machine Learning Model

Input Embedding Layer

Output Dense Layer

Time Distributed LSTM Layer

Dropout Layer

Dropout Layer

Time Distributed LSTM Layer

**MEHTODS**

LSTM RNN

1. Background

LSTM networks are a category of RNN that predict on time series data, or data that is formatted as time-steps. More generally, an RNN is a class of artificial neural networks that can form temporal connections and exhibit dynamic behavior (Dupond). Each LSTM network has two main layers indicative of the architecture of the network: a sequence input layer and an LSTM layer. The sequence input layer is responsible for accepting the data for the model as a time series, meaning that each step in the time series is just some small movement along a sequence. In the LSTM layer, the model learns “long-term dependencies between time steps of data” (MathWorks, “LSTM”).

Specifically, the LSTM layer is characterized by an input gate, an output gate, and a forget gate, which all work in combination to house relative features in each memory cell. Each memory cell is responsible for managing the error time-step to time-step as the model progresses. Due to these three different gates, the memory cell works to manage the flow of information based on short- and long-term patterns in sequences (Verleysen).

1. Results

Despite the fact that the DNA data was set up as time steps, the LSTM failed to correctly identify new true cases. That is, out of the true cases, there was not a single one that was correctly predicted following the training of the model.

1. Alternatives

Due to time constraints with a shortened Spring term at Drexel, no alternatives were attempted. One particularly interesting alternative, however, is included in the discussion.

**DISSCUSION**

CONCLUSION

With the use of locally generated sequences, the analysis of different subsequence matches was explored using a popular technique in machine learning and neural network architecture: LSTM RNN. Although this method is known for recognizing and classifying on long-term dependencies, this model proved to be ineffective in classifying independent time-steps of DNA k-mer data. One major simplification that was made in order to generate enough data in a timely manner is that each table structure fed into the network was disjoint from every other table structure. In other words, each table structure contained two unique classifications. Therefore, the model may have had difficulty in determining patterns and decencies from table to table considering the data lacked overlap of classes between different table structures. Instead of employing classification through LSTM, it may be interesting to explore prediction—as explained in the next section.

FUTURE RESEARCH

Instead of simply storing a logical TRUE or FALSE as the value corresponding to each k-mer, store the physical percent difference of the two alignment scores. In this way, the data is formatted for prediction instead of classification, which may prove to be more reliable. One main caveat of the data is that it each file was independent of each other. Although each file contained the same TRUE and FALSE classes, the classes represented matchings against different query sequences each time. In other words, one file contains classifications different match from that of another file, and so forth. The idea was to use overlapping independent files in order to classify similarly scoring sequences into the same class, which would represent sequences with 95% similarity or higher. By replacing the TRUE and FALSE classes in each file with simply the percent difference in scores, it is possible that LSTM can internally recognize similarly scoring segments. We are not concerned with absolute scores here, because larger k-mers that are an exact match score higher than smaller k-mers that are an exact match.

With respect to this alternative, data collection and curation would not be too difficult to manage, considering a pipeline was previously used to do these two steps. In fact, at the time of writing this paper, the data for this alternative has already been collected. The curation step would require sorting the data as its fed into the model because of the large number of

**REFERENCES**

Dupond, Samuel (2019). "A thorough review on the current advance of neural network structures". Annual Reviews in Control. 14: 200–230.

MathWorks, LSTM. "Long Short-Term Memory Networks - MATLAB & Simulink". Mathworks.Com, 2020, mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html. Accessed 2 June 2020.

MathWorks, NUC44. "Return NUC44 Scoring Matrix For Nucleotide Sequences - MATLAB Nuc44". MathWorks.com, 2020, mathworks.com/help/bioinfo/ref/nuc44.html. Accessed 3 June 2020.

Mount D. M., (2004). Bioinformatics: Sequence and Genome Analysis (2nd ed.). Cold Spring Harbor Laboratory Press: Cold Spring Harbor, NY.

Münch, Philipp. LSTM for binary DNA sequence classification. Github, 2020, github.com/philippmuench/dna\_lstm. Accessed 3 June 2020.

Needleman, S. B., Wunsch, C. D.,“A general method applicable to the search for similarities in the amino acid sequence of two proteins,” Journal of molecular biology, vol. 48, no. 3, pp. 443–453, 1970.

Salamat, S. and Rosing, T., 2020. FPGA Acceleration Of Sequence Alignment: A Survey. UC San Diego: Department of Computer Science and Engineering, pp.2,3.

Smith, T. F., Waterman, M. S., et al., “Identification of common molecular subsequences,” Journal of molecular biology, vol. 147, no. 1, pp. 195–197, 1981.

Verleysen, Michel. Proceedings / 23Rd European Symposium On Artificial Neural Networks, Computational Intelligence And Machine Learning, ESANN 2015, Bruges, Belgium, April 22-23-24, 2015. Ciaco, 2015.