Simran Rahi, 260604408
Veronica Rally, 260612206

18 Dec, 2017
COMP 561

COMP 561 FINAL PROJECT
Problem 2: Alignment algorithms for probabilistic sequences

Abstract

In the modern day, DNA sequencing is being done using high-throughput techniques. These new-age, error-prone techniques result in sequenced genomes that are better represented by probability matrices rather than deterministic strings. Additionally, advancements in bioinformatics have allowed the production of computationally generated ancestral genomes that are also best represented by probability matrices. To carry out basic local alignments between queries and probability matrices, this paper proposes an algorithm similar to BLAST that will allow for a higher likelihood of finding significant alignments; we postulate that an optimal alignment can be made with a probabilistic reference genome by incorporating the probabilities of each nucleotide's presence to the overall alignment score. It was found that this modified version of basic local alignment search tools was proficient at finding significant alignments.

Introduction

DNA sequencing, the process of determining the precise nucleotide sequence in a DNA molecule, is one of the most important biochemistry methods in modern science. Techniques to accomplish such a goal have been around since the 1970s beginning with the Maxam-Gilbert and Sanger methods (Griffiths, 2012). Since then, next-generation sequencing techniques (also known as high-throughput sequencing techniques) have been developed as a result of advancements in technology; some of these contemporary methods include Illumina (Solexa) sequencing, Roche 454 sequencing, SOLiD sequencing, and more (EMBL-EBI, 2016).

There are notable differences between the earlier and more recent sequencing techniques. Sanger sequencing, while highly accurate over long lengths, is quite expensive and takes about a week to sequence a few thousand nucleotides. Next-generation sequencing is fast, cost-effective, and has a high output; however, it is less accurate, producing errors at a rate of ~0.1-1x10^{-2} per base sequenced (Illumina, 2017). Since these new-age techniques are often used to sequence genomes that are billions of nucleotides long, the error rate has the potential to translate into millions of errors thereby making "deep sequencing" a necessity (i.e. sequencing that aims for a high number of replicate reads of each region of a sequence) (Lou et al., 2013).

Less accurate sequencing combined with the practice of deep sequencing results in probabilistic genomes rather than deterministic ones since it is impossible to determine with absolute certainty that each nucleotide sequenced is correct. That is, when the sequencing data is ambiguous due to possible errors, the uncertainty about the

identity of nucleotides at a given position can be better represented by the probabilities of each nucleotide's presence at that position (Blanchette, 2017) rather than a single, "possibly correct" guess. In this case, the genome would be depicted by a probability matrix rather than a string of nucleotides. The probability matrix would be of size 4xL where 'L' is the length of the genome and looks like:

		1	2	3	4	5	6	7	...
	A	0.01	0.5	0	1	0.2	0.1	0	
Genome =	C	0	0.5	0	0	0.3	0.1	0	
	G	0	0	1	0	0.4	0.8	1	
	T	0.99	0	0	0	0.1	0	0	

Genomes are sequenced for a multitude of reasons: to find gene location, determine gene function, make phylogenetic inferences, uncover disease-causing variations, and more (NCBI, n.d.). One of the most ubiquitous uses of sequenced genomes, however, is as a reference genome. A program such as the Basic Local Alignment Search Tool (BLAST) compares short query sequences to a reference genome, or database, in order to detect regions of alignment. These alignments allow insights to be gained on conserved sequences, gene families, and functional and evolutionary relationships between organisms (NCBI, n.d.), all of which are crucial for accomplishing the aforementioned goals of genome sequencing.

Apart from representing genomes that have been sequenced using error-prone, new-age techniques, probabilistic genomes can be used to depict genomes that are computationally inferred. For example, bioinformatics methods and orthologous sequences of many descendants can lead, via extrapolation, to the computation of likely ancestral sequences (Blanchette, 2008). This delineation of such a genome is probabilistic by nature, since it must incorporate the details arising from all its descendants while accounting for sequence conservation, polymorphisms, mutations, etc. However, one may still be interested in being able to compare deterministic queries to an ancestral genome as it can possibly give insight on the potential relationship between the organism from which the query originates from and the organism that the ancestral genome belongs to.

As the current implementation of BLAST only functions with deterministic reference genomes, a major problem reveals itself: significant alignments may be overlooked due to sequencing errors in the database or simplification of computationally generated ancestral genomes. To improve upon the current local alignment search tools, one must be able to compare query sequences not only to deterministic reference genomes, but also to probabilistic reference genomes. This paper presents a method in which an alignment of this manner can be accomplished.

Methods

The method delineated below for sequence alignment with a probabilistic genome was implemented using Python version 3.6.3 on Anaconda Spyder utilizing the Biopython, Pandas, and NumPy Python libraries. There are two major components to this method: preprocessing of the reference genome, or database, and the actual alignment between the query and database.

1 Pre-Processing

Preprocessing of the probabilistic reference genome is the first step of the proposed method. Its purpose is to provide a database from which likely matches in the genome can be identified quickly. Although this is a heuristic approach, it proves useful and effective in accelerating run time while still providing good matches; the brute-force alternative would be to consider the alignment of each position in the genome.

1.1 Implementation

Preprocessing requires the input of a FASTA file, which contains a sequence of nucleotides, and a text file, which corresponds to the probabilities for each listed nucleotide. Using the given genome, frequencies for each nucleotide are calculated by counting the number of occurrences of “A”, “C”, “T”, and “G” and dividing the values by the length of the genome. Following this count, a reference table is constructed. The nucleotide reported at a given position in the FASTA file has the probability reported at the corresponding position in the text file; however, the probabilities of the remaining nucleotides are not given. Instead, the probability that it is *not* the given nucleotide is divided among the remaining nucleotides proportionally to their respective frequencies. These frequencies are calculated as the conditional probability of finding one nucleotide given that it is not another:

$$Pr[n = N_1 | n != N_2] = Pr[n = N_1 , n != N_2]/Pr[n != N_2] = Pr[n = N_1]/Pr[n != N_2]$$

Using this, it is possible to dynamically calculate the probability of any nucleotide N_1 at any position i in the genome. Given that in the FASTA file, $n_i = N_2$, and the probability in the text file is p_i :

$$Pr [n_i = N_1] = (1 - p_i) * Pr[n_i = N_1 | n_i != N_2]$$

Once the conditional probabilities were calculated, it was possible to create data structures. These were used to reduce the time taken to query a sequence. To circumvent limited RAM, four lookup tables were constructed. Each lookup table contained the indices of all words in the reference genome that start with a different nucleotide. This enabled us to extend our word length *k* by one, because tables could be written, saved, and removed from RAM one at a time. The second through *k*-th nucleotides in the word were encoded to a number using a base-four encoding, such that the table consisted of 4^{k-1} indices. For each occurrence of the word in the genome, its location in the genome and the probability that it exists at that position were stored in a list at the corresponding index.

Because these are probabilistic genomes, many - nearly all, in some cases - of the possible word combinations may be possible at each position. The probability that a word exists at a given position is the product of the probabilities that each of those nucleotides is found in those positions. As such, the probabilities of these words can be exceedingly small. Although ideally, all words would be stored, a lower probability threshold was introduced to allow for the constrained RAM: only words with a probability greater than 1 in 1 billion were stored.

1.2 Run-Time

Encoding the words will take O(k) time. For a genome of length N, there are N-w positions with 4^k possible words per position. Overall, the asymptotic run time of preprocessing is $O(k*N*4^k)$. However, because the threshold is set, not all words will necessarily be considered. To improve running time, the probabilities of the words are built using a recursive function that breaks if the probability drops below the threshold. In practice, this should reduce the number of words checked. Further, although exponential in the word length, the word length should in practice not exceed around 11 nucleotides. In fact, a significant portion of the in-practice run time comes from writing the tables to CSV files.

2 Alignment Search

Alignment search allows the user to locally align a short query sequence to the probabilistic reference genome. The alignment search received two values as input: the query sequence and a threshold value. The threshold value indicates the minimum probability of a word in the reference genome that will be considered for alignment. For sequences from more reliable sequencing techniques, or when comparing

sequences to a closely-related ancestral genome, this threshold could be set higher; lower thresholds allow a more flexible exploration of the genome. Each alignment is scored, and the alignment with the best score is returned.

2.1 Filtering

The algorithm iterates through each word in the query sequence and, using the appropriate table and the base-four encoding, accesses the list of indices and probabilities for that word in the reference genome. The indices to be considered are then filtered using the probability threshold. Additionally, these matches are filtered to ensure that the query could be aligned with that portion of the genome - namely, any index too close to the beginning or end of the genome for the query to be appropriately aligned there is removed.

2.2 Extension

In the extension phase, the query is aligned locally to the reference genome near the selected index. By extracting a short segment of the reference genome, global alignment could be conducted between the segment and the query. Across the extension phase, the alignment is scored. The initial score for each alignment is the sum of the probabilities of the nucleotides in the word, as indicated in the reference genome. Matches and mismatches are given a value to be added to the score - matches score positively with some value A, and mismatches negatively, with some value I.

2.2.1 Ungapped extension

In the ungapped extension phase, the words of the query and reference genome are aligned. Extending outwards from either side of this alignment, the pair of nucleotides in the query and reference genome are aligned. Because the reference genome is probabilistic, the scoring of these alignments is designed to take this into an account. When a match occurs, the value of A is scaled by the probability of that nucleotide in the reference genome; when a mismatch occurs, the value of I is equivalently scaled. This continues until either (i) the end of the query is reached or (ii) the score falls below a threshold. The threshold corresponds to one-eighth the length of the query, and is designed to provide a “point of no return” - a value beyond which subsequent ungapped alignment is unlikely to improve the score beyond its original value.

2.2.2 Gapped extension

If nucleotides in the query remain unaligned after the ungapped extension phase, the algorithm passes to a gapped extension phase. Here, a variant of the Needleman-Wunsch pairwise alignment algorithm (reference) is implemented. The values A for match and I for mismatch are used. The gap penalty follows an affine gap penalty. However, because the extracted segment of genome can be significantly longer than the query sequence, and because a very large number of gaps in the short query sequence are unlikely, the gap penalty is modified depending on its distance from the word. The closer the gap is to the word, the greater its penalty; the further away, the smaller.

2.3 Run-Time

Encoding the words will take O(k) time. For each of the O(L) words in the query of length L, each of the $O(N*4^k)$ matches in the genome of length N must be considered. The gapped extension is Needleman-Wunsch and would run in $O(L^2)$, and the ungapped extension would run in O(L). Thus, overall, any query would take $O(L^3*N*k*4^k)$ time. However, it is exceedingly unlikely that the subset of words in the query would match all words in the genome. Further, because the gapped extension is not likely to be run on the whole query at once, the extension should usually be less than $O(L^2)$. Finally, the option to set a threshold for word probability should reduce the words

considered. The worst-case scenario would have all words with equal probabilities (which would occur if the probability of each nucleotide at each position was 0.25), but such a genome would not be useful.

Results

To test the performance of the program developed, different queries, thresholds, and probabilistic genomes were used as input to the program. First, using the frequencies of each nucleotide determined from the reference genome (chromosome 22 of the predicted BoreoEutherian ancestor sequence), different queries were randomly generated to perform alignments with; these queries were of varying lengths and mutation rates to test alignment search running times and score distributions of each variation. Following this assessment, probabilistic reference genomes of different sizes were used to test to preprocessing running time.

1 The effect of mutation rate and threshold on score

We generated sets of 50 query sequences, with each query of length 50 nucleotides, using exact sequences drawn from the database and mutated at different rates: 0%, 1.25%, 2.5%, 5%, and 10% each for substitutions and indels. The set of 50 random query sequences was generated using the nucleotide frequency distributions found from the reference genome. The score value was recorded for the alignment of each using word-probability thresholds of 10^{-1}, 10^{-2}, 10^{-3}, and 10^{-4}. This test was used to evaluate the efficacy of the scoring system for sequences and the optimal threshold, and was conducted using a word length of 9 nucleotides.

The scores declined as the mutation rate increased (Figure 1). Further, whereas the standard deviations of the mutated sequences were relatively large, those of the unmutated sequences and the random sequences were small in comparison. The threshold of 10^{-3} provided the clearest differentiation between match quality and score, as delineated by the smaller standard deviations (error bars) and clear declines. A threshold of 10^{-3} would allow for approximately three of the nucleotides in a query word to differ from the highest-probability word in the genome.

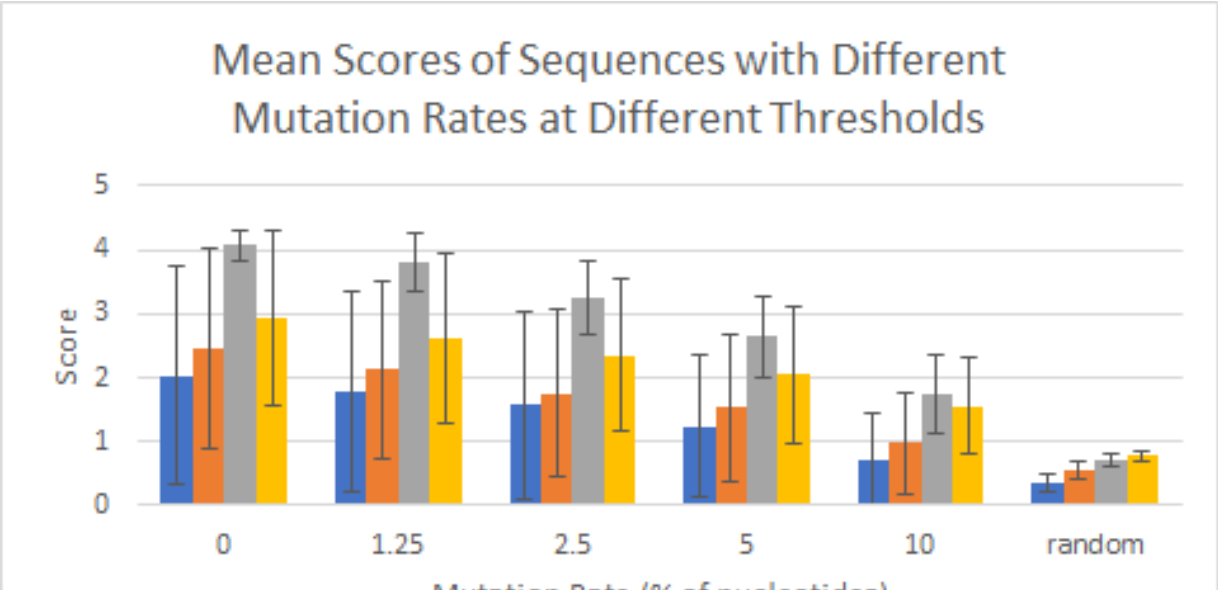

Figure 1. The mean scores (bars) and stand deviation (error bars) of sets of fifty sequences. Random sequences were generated using the reference genome's nucleotide distribution; others were generated using a sequence from the reference genome that was mutated. Each x0label other than random indicates the average percent of nucleotides that underwent substitutions and the average number of indels. Different colours indicate different word-probability thresholds.

2 The effect of query length on alignment search time

Using randomly generated queries of varying lengths ranging from 30 to 150 nucleotides long, the running time for an alignment search was measured. One query at each level of query length was inputted to the program and the running time of each was recorded (Table 1). This shows that running time of the alignment search increases as query length is continually increased. The lengths of the test queries were increased in each round until the program was aligning queries that were 150 nucleotides in length. The results of this manipulation can be found in Table 1. The results indicate that as query length increases, the running time for detecting an alignment with the probabilistic reference genome also increases in a nonlinear fashion.

Genome Length (nucleotides)	Preprocessing Time (seconds)
604,466	1502.7606
60,446	123.3218
6,044	14.5412
604	3.4509
60	2.2014

Table 1. Queries of different lengths were inputted to the alignment search algorithm to determine the length's effect on running time. There is a positive, nonlinear correlation between the two variables.

Figure 2. The preprocessing running time increases as the probabilistic genome length also increases in a nonlinear fashion. The log was taken of all data points (Table 2) to manage the range of data and was subsequently graphed to reveal an exponential trend.

Discussion

This paper employs an algorithm similar to BLAST with the added benefit of allowing alignments between query sequences and probabilistic, rather than deterministic, reference genomes. The implementation of the methods described result in a program that takes a probabilistic genome and query sequence as input and outputs the optimal alignment between the two sequences. Throughout the manipulation of this program conclusions reached about the time and space complexities and inferences were made regarding the future application of this development.

The first test conducted explored the relationship between not only user-dictated threshold values and alignment scores, but also between mutation rates with alignment scores. It was discovered that as the mutation rate increased (i.e. the query taken exactly from the reference genome was changed more and more), the score for its alignment decreased as well; this trend demonstrates the proposed algorithm's

ability to (i) penalize alignments that are not exact and (ii) recognize that certain queries are better/worse than their less mutated counterparts. This functionality of the program is important because it allows alignments to be detected between sequences that have diverged evolutionarily through random mutations and polymorphisms. It was also seen that in all cases, except the random case, the threshold value of 10^{-3} led to alignments with the highest score indicating that this "sweet spot" filtered out unlikely matches and was not too narrow as to exclude some significant ones.

The second test conducted intended to describe the relationship between query length and the time required to detect the optimal alignment in the reference genome. It was found that as the queries got longer, the running time of the alignment search also got longer. This correlation is not linear as an alignment search with a 30 nucleotide sequence takes ~4 seconds and an alignment search with a 150 nucleotide sequence took ~40 minutes. Once again, this drastic increase in running time can be justified by the asymptotic analysis of the algorithm and indicates a range of query sequence lengths that are most likely to deliver efficient and fruitful matches.

The last test conducted aimed to delineate the correlation between probabilistic reference genome length and the running time to preprocess that genome. It was discovered that there was an exponential relationship between these two variables; this relationship is further explained by the running time complexity of preprocessing being $O(k*N*4^k)$, where k is the word length and N is the length of the genome. This information plays an important role when one considers the utility of this program; as the genome grows, it becomes less and less plausible to preprocess it in an efficient, quick, and usable manner. As a result, one may infer that this program is best suited for probabilistic sequence alignment of organisms with short genomes; for example, preprocessing the genome of *Carsonella rudi* (160,000 base pairs) (Than, 2006) would be far faster and more easily subject to updates than the genome of a human (3 billion base pairs). It should also be noted that while the genome is small, the action of writing the lookup tables to CSV files is what takes up much of the time; as the genome lengthens, the time spent writing to the CSV files will take up smaller and smaller proportions of the overall running time. With this observation, one can posit that as the genome keeps increasing, running time will eventually become linear.

The main discovery of the tests performed is that the algorithm developed runs in various different times depending various parameters: word length, query length, genome length, and threshold. Preprocessing of the portion of chromosome 22 of the predicted BoreoEutherian ancestor sequence alone takes about ~25 minutes whereas queries being aligned with this probabilistic sequence can run in many different times; both of these results can be best explained by the asymptotic analyses reported in the methods.

A limitation of this algorithm includes a lack of flexibility between the word length used for pre-processing and the word length used for the query alignment search. Since the probabilistic genome is preprocessed in a manner that requires a decided word length, all future query alignment searches carried out using this genome are required to use the same word length. This unalterable variable impacts the specificity of potential alignments; smaller word lengths may allow for less specific alignments whereas longer word lengths may filter out potentially significant matches. An additional limitation to this methodology is the amount of RAM this process utilizes. On the average

computer, large genomes, queries, and word lengths may pose space efficiency problems; however, more powerful computers (such as those found at the NIH) are less likely to run into this issue and therefore will be able to use this program most efficiently. A last limitation of this study applies to the tests we performed on the program. In the second test that investigated the relationship between query length and running time, only one randomly generated query sequence was used to generate data. However, a more sound approach to this investigation would be to generate many random sequences at each length level and then to average their running times as it is possible to see notable variation depending on the specific query sequence.

A potential future application of this algorithm is for optimal protein alignments between a deterministic amino acid sequence and a probabilistic one. Protein sequencing, most often carried out by mass spectroscopy (MS) and Edman degradation, is the process of determining the amino acid sequence of a given peptide. Much like the high-throughput technologies used for DNA sequencing, the methods used for protein sequencing are subject to error due to a lack of robust statistical and computational methods to deal with the results of MS; as a result, proteomic datasets contain a large number of false positives (Nesvizhskii, 2010). With this source of uncertainty in sequenced peptides, using a probability matrix to depict the amino acid sequence would be a more accurate representation than simply a sequence of amino acids. Using this probabilistic peptide sequence in the same manner as the probabilistic DNA genome, one can posit that protein alignment can be solved similarly to DNA alignment. Protein alignment in this fashion will allow comparisons of protein sequences between organisms (i) to detect common evolutionary ancestors and (ii) to find proteins with similar functions (Pearson, 2013).

References (MLA)

Griffiths, Anthony J.F. “DNA sequencing.” Encyclopædia Britannica, Encyclopædia Britannica, inc., 5 July 2012, www.britannica.com/science/DNA-sequencing.

“What is Next-Generation DNA Sequencing.” EMBL-EBI Train online, 8 June 2016, www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-you-will-learn/what-next-generation-dna-.

“Deep Sequencing.” Deep Sequencing, Illumina, 2017, www.illumina.com/science/technology/next-generation-sequencing/deep-sequencing.html.

Lou, Dianne I., et al. “High-Throughput DNA sequencing errors are reduced by orders of magnitude using circle sequencing.” Proceedings of the National Academy of Sciences of the United States of America, National Academy of Sciences, 3 Dec. 2013, www.ncbi.nlm.nih.gov/pmc/articles/PMC3856802/.

Blanchette, Mathieu. “COMP 561 Final Project - Fall 2017.” McGill University, Fall 2017, http://cs.mcgill.ca/~blanchem/561/

“Why are genomes sequenced?” National Center for Biotechnology Information, U.S. National Library of Medicine, https://.support.ncbi.nlm.nih.gov/link/portal/28045/28049/Article/751/Why-are-genomes-sequenced.

“BLAST: Basic Local Alignment Search Tool.” National Center for Biotechnology Information, U.S. National Library of Medicine, https://blast.ncbi.nlm.nih.gov/Blast.cgi.

Blanchette, Mathieu, et al. “Computational Reconstruction of Ancestral DNA Sequences.” Methods in Molecular Biology: Phylogenomics, Humana Press Inc., 2008, www.bx.psu.edu/miller_lab/dist/11_Blanchette.pdf.

Than, Ker. “Smallest Genome of Living Creature Discovered.” LiveScience, Purch, 12 Oct. 2006, www.livescience.com/1091-smallest-genome-living-creature-discovered.html.

Nesvizhskii, Alexey I. “A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics.” Journal of proteomics, U.S. National Library of Medicine, 10 Oct. 2010, www.ncbi.nlm.nih.gov/pmc/articles/PMC2956504/.

Pearson, William R. “An Introduction to Sequence Similarity (“Homology”) Searching.” Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis ... [Et al.], U.S. National Library of Medicine, June 2013, www.ncbi.nlm.nih.gov/pmc/articles/PMC3820096/.

Baldwin, Michael A. “Protein Identification by Mass Spectrometry.” Molecular & Cellular Proteomics, Molecular & Cellular Proteomics, 1 Jan. 2004, www.mcponline.org/content/3/1/1.full.