Sequence alignment for probabilistic genome based on BLAST

Qianyu Huang 260669624

Course: COMP 561 Computational Biology Methods and Research (4 credits)

Professor: Mathieu Blanchette

McGill University, Montreal, Quebec, Canada H3G 1A4

Abstract

Sequence alignment has always been an important procedure in bioinformatics. It can identify similarity between genomes, provide phylogenetic trees, as well as develop homology models for protein structure. Although many different algorithms were developed in the past few decades for various purposes of genomic sequence alignment, few are able to align sequences that are probabilistic, for instance, sequences obtained from ancestral sequence reconstruction. Here we develop an algorithm inspired by the basic local sequence alignment tool (BLAST) that allows aligning a definite query with an uncertain genomic database.

Implementation available on: https://github.com/sylhuang/COMP561-Final-Project

Introduction

When studying genomes, sequence alignment is an extremely useful method that provides us information across different species, within the same species and along its evolutionary ancestry. Many algorithms today can fulfill most needs in sequence alignment, such as Needleman-Wunsch algorithm for optimal global alignment (Needleman & Wunsch, 1970), Smith-Waterman algorithm for optimal local alignment, and basic local alignment search tool (BLAST) for heuristic local alignment (Altschul et al., 1990). However, these algorithms can only align sequences that are certain, but in some situation, there can be uncertainty about the genome of a species. For example, while reconstructing genome of an ancestral DNA sequence from species alive today through a probabilistic approach, the result cannot be verified and is often expressed in their posterior probability given the data (Ashkenazy et al. 2012). Therefore, a new alignment algorithm is needed for such tasks.

For most data predicted, nucleotides that share equal probability of being correct are rare; in other words, majority of nucleotides predicted still have much higher confidence level than the other three types of nucleotides. Hence, we can borrow the idea from BLAST with consideration of matched nucleotide can be incorrect, and mismatched nucleotide can be correct.

Materials and Methods

Data

The probabilistic database used is a portion of portion of chr22 of the predicted Boreoeutherian ancestor sequence of length 604122 base pairs, provided by Mathieu Blanchette's Computational genomics Lab. Each nucleotide predicted has its corresponding probability of being correct; if incorrect, then the other three nucleotide types share an equal possibility of being the right one.

The query sequence used for testing is generated by picking a random starting location in the database, generating each nucleotide based on their probability. Then, an error rate of t% is introduced to the sequence, meaning t% of the location in the sequence is chosen at random, and a single substitution, insertion or deletion (also chosen at random) occurs there.

Algorithm

Comparing to sequence alignment between two definite sequences, alignment between a probabilistic database and a definite query only differs when there is a match, or a mismatch; a match may gain smaller score, and a mismatch may get less penalty if the confidence of the nucleotide is low. On the other hand, the probability will not be a factor if a nucleotide is inserted or deleted. Therefore, I adopt this algorithm from BLAST, with major modifications on the scoring scheme during gapped extension.

While indexing the database, the score of each w-mer is also recorded, which is obtained by summing up the probability of each nucleotide within the w-mer. For a w-mer of length 11 base pairs, if the score is lower than 8.8, meaning the w-mer has already lower than 80% correct nucleotides, it is discarded from the indices.

Then, the query is scanned for hits in database. All hits are considered for ungapped extension. Again, the score is computed by summing up the probability of each nucleotide within range. High-scoring gapless alignments (HSPs) with score higher than 10% of query length are used for gapped extension.

The optimal alignment during gapped extension is defined by the following scoring scheme:

$$M_{i,j} = \begin{cases} M_{i-1,j-1} + match(q_i, d_j) \\ M_{i-1,j} - 1 \\ M_{i,j-1} - 1 \end{cases}$$

$$\text{match}(q_i, d_j) = \begin{cases} prob(d_j), & if\ q_i = d_j \\ prob(d_j) - \dfrac{1 - prob(d_j)}{3}, & if\ q_i \neq d_j \end{cases}$$

where $prob(d_j)$ is the corresponding probability of nucleotide in the probabilistic genome. In this case, for a nucleotide with 100% confidence, a match will still gain it +1 and a mismatch will penalize it with -1.

Among all alignments after the gapped extension, the one with the highest score will be reported.

Results

Queries with various size and error rate (t%) are aligned with the original database. Each set contains 100 trials of randomly generated queries. The alignment result is considered correct if the reported start of alignment in the query and the position in database that generates the query add up to the reported start of alignment in the database (Figure 1).

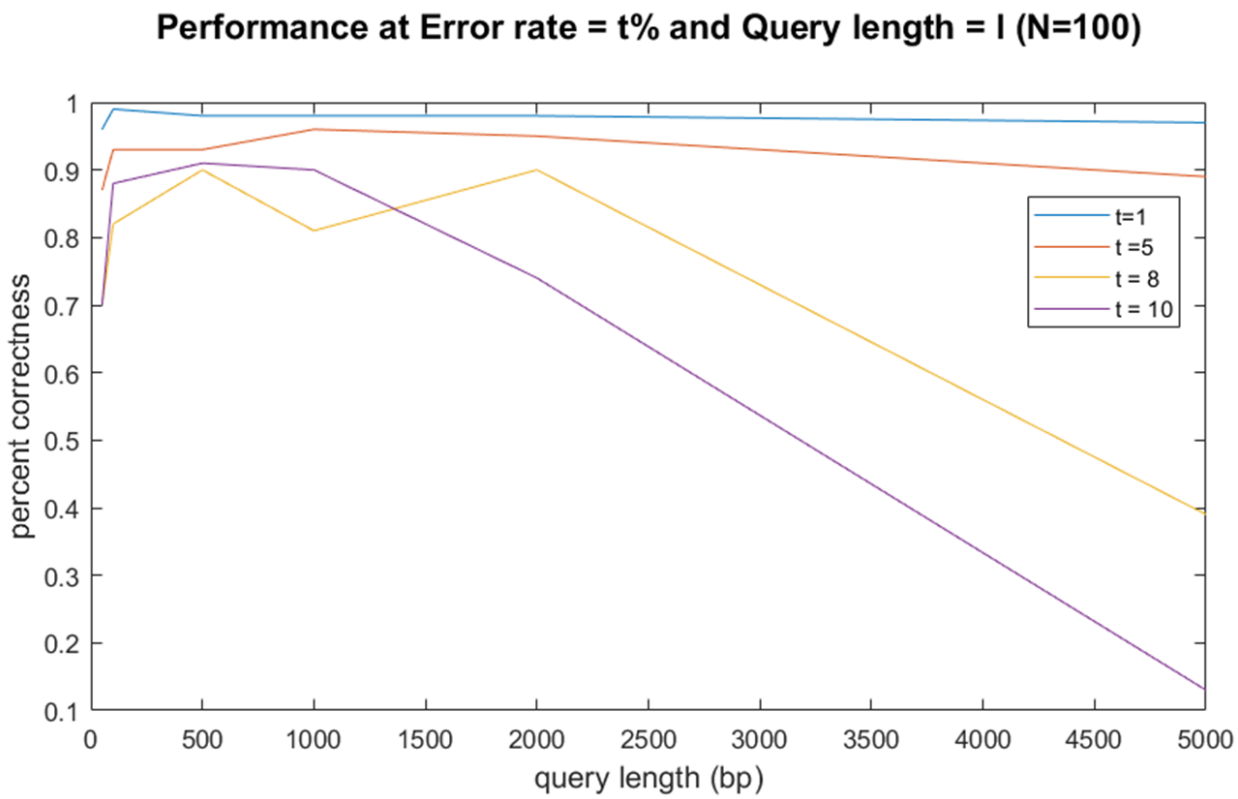

Figure 1. *Percent correctness of probabilistic sequence alignment.* Query length tested are 50 bp, 100 bp, 500 bp, 1000 bp, 2000 bp and 5000 bp. Each set contains 100 trials of different queries.

From the figure we can see that, as the error rate grows, the performance decreases dramatically, which can be expected; since if the is one error (substitution, insertion or deletion) every 10^{th} nucleotide, it would be much less possible to find a seed of length 11 to start with. Also, as the query length grows longer, it's harder for the algorithm to find the right location, probably because there are more errors outside of the HSP regions. Yet, with a reasonable error rate, the performance is quite consistent, regardless of query length.

Discussion

Since this algorithm is a modified version of BLAST, there is no big difference in the running time and the space needed. The preprocessing takes O(d) running time and storage space, where d is the size of the probabilistic database. Scanning for seed takes O(q) running time and space, where q is the size of query. The running time for ungapped extension is also linear in size of extension. The most expensive part is the gapped extension. Since the Needleman-Wunsch algorithm can be memory consuming, especially when one of the sequence can be billions of nucleotides long, I cut out the database region that is farther than 4 times length of the query remaining, which is impossible to reach or there would be too many gaps in the alignment.

Three parameters that I did not look at for enough time are the two thresholds and the ratio between indel penalty and match score. The first threshold is the threshold for determining the useless seeds in the database. I arbitrarily chose 80% but a higher number should give fewer false positives and a lower number should give more sensitive scanning. This might have a effect when the error rate is higher. Similarly, I also chose arbitrarily 10%*length of query for threshold of valuable HSPs. In case there are many errors that interrupt HSPs, a higher threshold will miss the right target; on the other hand, a lower HSP threshold will take algorithm longer to run and will give many false positives. The scoring ratio will affect the alignment depending on distribution of error types. Here, all errors occur equally, while insertion and deletion gets a lot higher penalty than a substitution.

References

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990). Basic local alignment search tool. *J Mol Biol*, 215(3):403-10.

Ashkenazy H, Penn O, Doron-Faigenboim A, Cohen O, Cannarozzi G, Zomer O, Pupko T (2012). FastML: a web server for probabilistic reconstruction of ancestral sequences. *Nucleic Acids Res*, 40(Web Server issue): W580–W584.

Needleman SB, Wunsch CD (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443-53.