Alignment algorithms for probabilistic sequences

Yuxue Zhu
260737363

Introduction

In 1990, Basic Local Alignment Search Tool (BLAST) was designed as an efficient tool to align an interested sequence against given genomes. It is vital for bioinformatic scientist to identify the evolutionary relationship for the mystery species and detect the structural homology. Traditionally, the genome sequence using in BLAST is an DNA sequence where the nucleotide at each position is fixed. However, some genome sequences don't have a fixed nucleotide at each position when the sequencing data that resulted in the genome assembly is ambiguous at certain positions. Instead, each nucleotide has a probability to occur for a position of sequence. Therefore, the difference of alignment between probabilistic sequence and a normal sequence is that we can't know if there is a substitution. Since nucleotide of higher probability is more likely to appear, we can use the probability as match score and the probability of which nucleotide depend on the query. Inspired from BLAST, we designed an alignment algorithm for probabilistic sequence, which mimics the strategies used in BLAST with a different scoring system.

Methodology

1. **Project Objective**

   This project is designed and implemented an algorithm to find an optimal alignment between a short non-probabilistic sequence (query) and a long probabilistic sequence (genome) which is analogous to three main steps from BLAST. 1, Find perfect match region 1, Ungapped extension phase 3, gapped extension.

   Input:

   - A fasta file containing the sequences of the most possible nucleotides
   - A txt file giving the probability of the most possible nucleotide at each position. The probabilities of all the other three nucleotides are equal.
   - A short query sequence (50 ~100 bp)

   Output:

   - An optimal alignment between query and genome

2. **Implementation**

   2.1. Read in genome sequence as 'genome' and corresponding probability as an array 'prob'. The probability for other less probable nucleotide at genome I position is calculated by $\frac{1-prob[i]}{3}$.

   2.2. **Build word dictionary** ('findwords' function)

   To start off, we need to build a word dictionary based on genome with length w. we will store the possible words and all start positions in genome with their score as a dictionary where the structure is described as below:

Word dictionary	
key	value
'TTT'	position:score pairs

value of word dictionary	
key(position)	value(score)
23	0.2

$$score_{word} = sum\left(log(prob[i][n])\right)$$
$$i \in indexes\ of\ word\ in\ genome, n \in nucleotide\ of\ word$$

We generated all the possible words with length w to initialize the key of dictionary and loop though genome sequence to record the position and score.
Since the sequence is probabilistic, at each position there would be 4^w words. To focus on the most possible words, a threshold would be introduced to filter out the words with lower probability
Once the word dictionary is built up, we then find the word match positions in query by indexing the word dictionary with the subsequences of query of length w.

2.3. **Ungapped extension** (findHSP function)
To speed the time up, we would scan for high score pairs (HSP) by extending the perfect matches to both left and right side with certain length without gaps. the score of each base pair is defined as the log of the probability at the position of genome of the nucleotide in query.

$$score_{HSP} = sum\left(log(prob[i][n])\right)$$
$$i \in indexes\ of\ HSP\ in\ genome, n \in nucleotide\ of\ HSP$$

Given two extension length for left and right, the final score of the HSP will be the sum of score at each position of extending region and the score of responding word.
At this step, the starting and ending position of HSP of both query and genome will be reported and as well as the correlated score.

2.4. **Gapped extension** (getSequence function)
We applied a variant Needleman Wunsch algorithm to the rest upstream and downstream of query and genome. To allow a few gaps in the alignment, we defined the length of upstream and downstream of genome to align with from half to double of the length of upstream and downstream of query respectively, by which the insertion and deletion will be taken into consideration in a reasonable range. Then we selected the alignment with maximum score among different length of genome for a HSP.

```
$bestAlign = Null$
$for\ i = \frac{1}{2}\ length(upstream_{query})\ to\ 2 * length(upstream_{query})$:
```

$$upstream_{genome} = genome[HPSStarting - i{:}\, HPSStarting]$$
$$\text{bestAlign} = \max\left(best, NW\left(upstream_{query}, upstream_{genome}\right)\right)$$

The algorithm is the same for downstream alignment.

Since the genome sequence is probabilistic, a base pair would always to have a chance to mismatch with each other. Instead of rewarding the matching pair and penalizing the mismatch pair, we will only consider the mismatching probability by multiplying penalty with the probability of all nucleotide that don't match with in query. The gap penalty is the same as used in original NM.
Moreover, it is not allowed to have gaps in alignment before the upstream of query and after the downstream of the query. Since the score would be equivalent to the alignment with shorter genome sequence but have lower score.

Score=matrix of size (length(S_query)+1)*(length(S_genome)+1)
Score[0][:]=-inf
Score[q][g]=

$$max\begin{cases}\text{Score}[\text{q}-1][\text{g}-1] + \text{mismatch} * (1 - \text{prob}[\text{g}][\text{nucleodtideAtQuery}]) \\ \text{Score}[\text{q}-1][\text{g}] + \text{gapPenalty} \\ \text{Score}[\text{q}][\text{g}-1] + \text{gapPenalty}\end{cases}$$

2.5. Assemble gapped alignment with HSP

To find the final alignment, we will take the alignments with maximum score of the sum of score for upstream, downstream and HSP from every hits.

$$Score_{alignment} = \max\left(Score_{upstream} + Score_{HSP} + Score_{downstream}\right) HSP \;\in hits$$

Furthermore, we trace back of the alignment of ungapped extension and retrieve the corresponding HSP by the position index to get the alignment then concatenate them together.

Results

we select a sub sequence of length 50 from genome as query1 and made some substitutions of query1 as query2. We tested the algorithm using these two queries where query1 is supposed to completely match genome from position 111681 to 111729 while query is evaluated by percentage of correct matched bases.
query1 : CCCACCTCTGTGTCAAACAGTGGGGTACATGCTCTTGCTTAATCCAGCTG
query2 : CCCATTGTGTGTCATACAGTCAGGGTACATGTTGCCTCTTAATCCAGCTG

initial values:

word threshold	**length(word)*log(0.5)**
HSP threshold	length(HSP)*log(0.7)
gap penalty	-1
mismatch weight	-1

For simplicity, we set the same ungapped extension length for downstream and upstream.

1. **Query1 is perfectly match every nucleotide with genome.**
   we first aligned query1 with genome to examine if the algorithm can make a perfect match and investigate the running time by tunning the word length from 3 to 7 value with ungapped extension length from 3 to 8.
   The alignment from each combination of word length and ungapped length exactly matched the genome subsequence without gaps which indicates the algorithm is indeed to find the alignment of highest score.
   By looking into the running time with different word length and ungapped length. We found that there is significant decrease at beginning with word length 3 by increasing ungapped length or with ungapped length 3 by raising word length whereas the running time won't vary too much at a lower value within 100s with longer word length or ungapped length. (Figure1)
   Since Needleman Wunsch alignment would take the longest time, the longer HSP, the shorter length will be processed for Needleman Wunsch which will shorten the running time. The ungapped length and word length both will determine the length of HSP, then short word and ungapped portion will give longer running time.
   Moreover, the running time won't change in a wide range for all word length if ungapped length is longer than 5 which we used these values for query2 to evaluate the accuracy and running time of the other parameters.
2. **Query2 is aligned with genome with an acceptable accuracy**
   Given ungapped length 5, we aligned the mutant sequence query2 with genome with different parameters of word length, HSP threshold and gap penalty and found the following results.

   2.1. Running time
   The HSP threshold was initialized at length(HSP)*log(0.7) and by changing value inside log, the running time dops as increasing the HSP value as well as word length which has the same trend of running time of query1 (Figure2). It is explained that the high threshold results in less HSP for the next gapped extension.

   2.2. Accuracy
   In contrast to the running time, the accuracy of alignment didn't have any changes with neither small or high threshold and word length (Firgure3), which indicates that the sub sequences of query2 with highest probability would always be selected.
   Since HSP threshold and word length are used in the first steps of BLAST, then we investigated in the effects of penalty in the third step, gapped extension.
   Given mismatch weight as -1, the accuracy decreased a bit for the small and large gap penalties where the large gap penalties lower the accuracy less than small penalties. Within range of -0.6 to -1, the accuracy was staying at the maxima value of more than

0.8 (Figure4). Overall, accuracy was fluctuated within relative high values for different gap penalty.

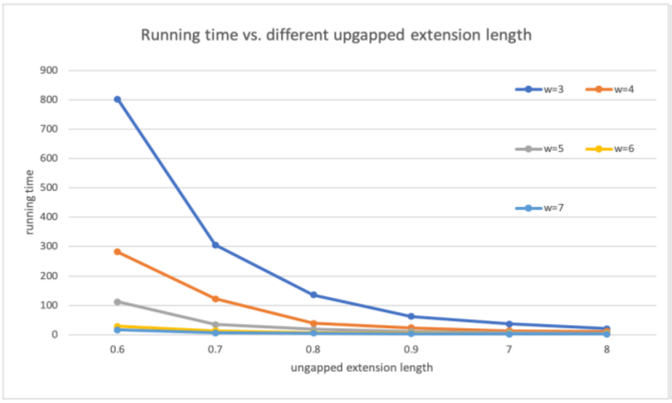


Figure1: running time vs. ungapped extension length with different word length for query1

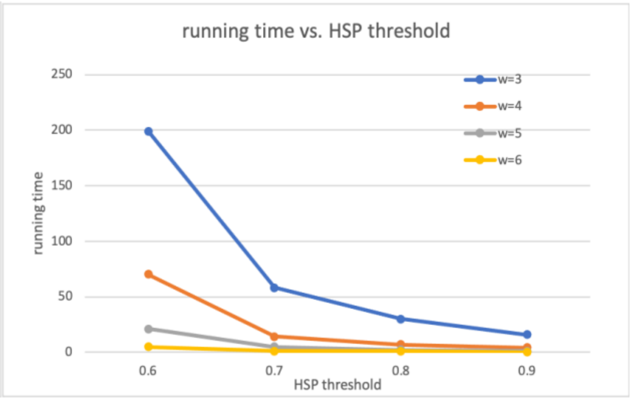


Figure2: running time vs. HSP threshold length with different word length for query2

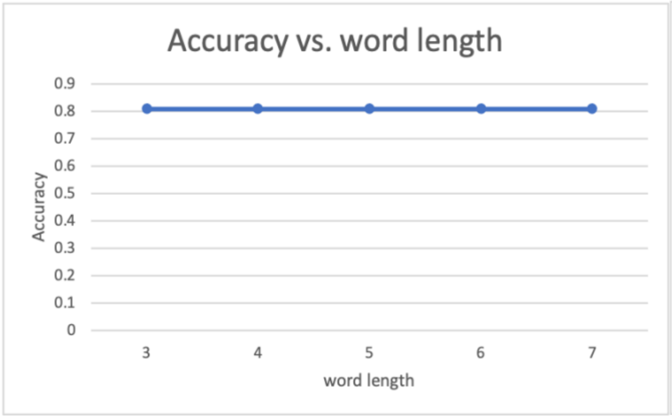
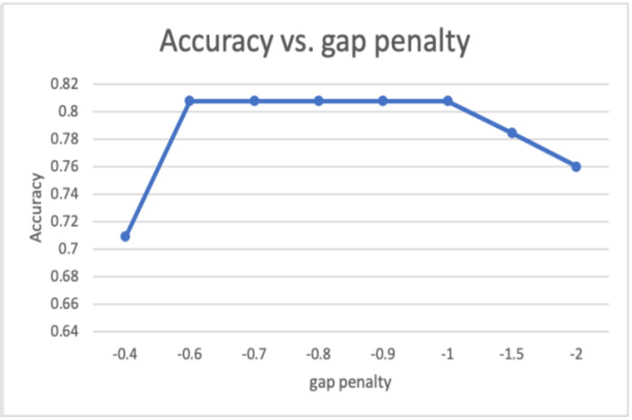


Figure3: accuracy vs. word length for query2 with word length 5, ungapped length 5 and HSP threshold 0.7



Firgure4: accuracy vs. gap penalty for query2 with word length 5, ungapped length 5 and HSP threshold 0.7

Discussion

In this project, we developed an algorithm based on BLAST for probabilistic sequence. Firs build the word dictionary with the words passing threshold. Then search for the query sequence and extend the targets in order of ungapped and gapped phase to identify the alignment of highest score. The variation from traditional BLAST is that scoring system. Instead of substitution matrix, we take the log of probability of the nucleotide of query in the corresponding position of genome and set the thresholds for the word dictionary and ungapped extension to narrow down the really unlikely HSP.
By tuning the parameters, we found running time is decreasing as elevating HSP threshold and word length which is a character of traditional BLAST and the accuracy reaches the maxima value within a range of appropriate gap penalty values. The higher penalty value gave an

alignment with less gaps which is opposite to lower penalty value. Interestingly, both high and low penalty produced the best alignment with shorter genome sub sequences. Overall, the accuracy is at a fairly high level (Appendix).
In addition, HSP threshold and word length didn't change the accuracy for our query. It could result from the word dictionary where we chose the word with every nucleotide that has at least 0.5 chance to occur. This could filter out the most impossible word so that HSP threshold and word length will have a less effect on accuracy.

This model can give an alignment with relative high accuracy for short query. in reality, the query could be much longer than 50 bp. Furthermore, this experiment assumes the upstream and downstream ungapped extension length is the same which could not be the best case for longer sequences to reach the unpgap maxima appearing at different length. We might investigate in upstream and downstream maxima and test on longer query sequences for the future work.

Appendix

```
w: 5, r: 5, l: 5, penalty: -2.000000, hspthre: 0.700000
Q: CCCATTGTGTGTCATACAGTCAGGGTACATGTTGCCTCTTAATCCAGCTG
G: CCACCTCTGTGTCAAACAGT_GGGGTACATGCTCTTGCTTAATCCAGCTG
111681 to 111729
Score: -14.129827
acc: 0.760000
Time: 5
w: 5, r: 5, l: 5, penalty: -0.900000, hspthre: 0.700000
Q: CCCAT_TGTGTGTCATACAGTCAGGGTACATGTTGCCT_CTTAATCCAGCTG
G: CCCACCTCTGTGTCAAACAGTG_GGGTACATGCT_CTTGCTTAATCCAGCTG
111680 to 111729
Score: -10.789827
acc: 0.807692
Time: 5
w: 5, r: 5, l: 5, penalty: -0.400000, hspthre: 0.700000
Q: CCCATTGTGTGTCATA_CAGTCAGGG_TACATGTTGC_CT__CTTAATCCAGCTG
G: CC_TCTGTGT__CA_AACAGT__GGGGTACATG___CTCTTGCTTAATCCAGCTG
111684 to 111729
Score: -7.583160
acc: 0.709091
Time: 5
```

Bibliography

[1] Altschul S.F., Gish W., Miller W., Myers E.W., Lipman D.J. Basic local alignment search tool. J. Mol. Biol. 1990;215:403–410.
[2] Altschul SF, Madden TL, Sch¨affer AA, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25(17):3389–3402. doi:10.1093/nar/25.17.3389
[3] Eddy SR. A probabilistic model of local se- quence alignment that simplifies statistical significance estimation. PLoS Comput Biol. 2008;4(5):e1000069. Published 2008 May 30. doi:10.1371/journal.pcbi.1000069
[4] Needleman, Saul B. & Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". Journal of Molecular Biology. 48 (3): 443–53. doi:10.1016/0022-2836(70)90057- 4. PMID 5420325.