CSE 4065

Computational Genomics

CSE 4094

Project 1 Report

Ahmet Bozbay 150119861

Anıl Batuhan Aslan 150119656

Çağan Kurt 150119677

What is the Definition of Dynamic Pairwise Sequence Alignment?

Dynamic pairwise sequence alignment is a method used to align two sequences of DNA, RNA, or protein in order to compare them and identify regions of similarity or conservation. The goal of dynamic pairwise sequence alignment is to identify the optimal alignment between two sequences, taking into account the possibility of insertions, deletions, and substitutions of nucleotides or amino acids.

Dynamic pairwise sequence alignment algorithms use a scoring system to evaluate the quality of different alignments and determine the best alignment based on the highest score. The scoring system typically assigns a positive or negative value to each possible alignment, depending on whether it is a match or a mismatch between the two sequences. Insertions and deletions are typically penalized, while matches are rewarded.

Dynamic pairwise sequence alignment algorithms can be used to identify conserved regions of a gene or protein, to study evolutionary relationships between different species, or to design probes or primers for use in molecular biology techniques such as PCR or DNA sequencing.

What is the Needleman-Wunsch algorithm?

In this project we used the Needleman-Wunsch algorithm. The Needleman-Wunsch algorithm is a dynamic programming algorithm used to align two sequences of nucleotides or amino acids. It is named after its inventors, Saul B. Needleman and Christian D. Wunsch, who published it in their 1970 paper "A general method applicable to the search for similarities in the amino acid sequence of two proteins".

The algorithm is used to find the optimal global alignment between two sequences, which means aligning the two sequences in a way that maximizes the number of matches between them. To do this, it compares each position in the two sequences and assigns a score based on how well the characters at those positions match. It also allows for the insertion of gaps in the alignment to allow for the

alignment of subsequences that are not contiguous in either of the original sequences.

The algorithm works by constructing a matrix of scores for all possible alignments of the two sequences. It then uses this matrix to determine the optimal alignment by following a path through the matrix that maximizes the score. The path through the matrix is chosen based on a set of rules that specify the cost of inserting a gap, the cost of a character mismatch, and the cost of a character match.

The Needleman-Wunsch algorithm is widely used in bioinformatics and has many applications, including the alignment of DNA sequences for gene prediction and the alignment of protein sequences for protein structure prediction and evolutionary analysis.

How does the code work?

The code defines a function called dynamic_sequence_alignment that aligns two DNA sequences using the dynamic programming approach. The function takes four parameters: seq1 and seq2, which are the two DNA sequences to be aligned, and match_score and mismatch_score, which are the scores for aligning matching and mismatching characters, respectively. The function also has two additional parameters, open_gap_score and extend_gap_score, which are the scores for opening and extending a gap, respectively.

The function first initializes a matrix of all zeros with dimensions m+1 by n+1, where m and n are the lengths of seq1 and seq2, respectively. It then fills in the matrix using the dynamic programming approach, which involves comparing the scores for aligning the characters in the two sequences in different ways and choosing the maximum score at each step.

After the matrix has been filled in, the function traces back through the matrix to retrieve the optimal alignment of the two sequences. It does this by starting at the bottom right corner of the matrix and working backwards, following the path that corresponds to the maximum score at each step.

Finally, the function returns the two aligned sequences and the alignment score.

The code then tests the dynamic_sequence_alignment function by reading in six DNA sequences from files named test1.seq, test2.seq, etc., aligning the first and last lines of each file (which are assumed to be the two DNA sequences), and printing the aligned sequences and the alignment score.

Description of the code piece by piece;

```
for i in range(1,6):
    with open('test'+str(i)+'.seq', 'r') as f:
        data = f.readlines()
        firstdnapiece=data[0]
        seconddnapiece = data[-1]
```

This code uses a for loop to iterate over the values 1, 2, 3, 4, and 5. For each value of i, the code opens a file named test<i>.seq, where <i> is the value of i (e.g., test1.seq for i=1, test2.seq for i=2, etc.).

The code then reads the lines of the file into a list called data and assigns the first and last elements of this list to the variables ‘firstdnapiece’ and ‘seconddnapiece’, respectively. These variables contains the two DNA sequences that will be aligned.

```
# Fill in the matrix using the dynamic programming approach
for i in range(1, m+1):
    for j in range(1, n+1):
        # Calculate the scores for aligning the two characters
        match = dp[i-1][j-1] + match_score \
            if seq1[i-1] == seq2[j-1] \
            else dp[i-1][j-1] + mismatch_score
        delete = dp[i-1][j] + open_gap_score + extend_gap_score
        insert = dp[i][j-1] + open_gap_score + extend_gap_score
        # Take the maximum of the three scores
        dp[i][j] = max(match, delete, insert)
```

This code is a nested for loop that iterates over the indices of the two DNA sequences, seq1 and seq2. The outer loop iterates over the indices of seq1, and the inner loop iterates over the indices of seq2.

For each pair of indices (i, j), the code calculates three different scores:

The match score, which is the score for aligning the characters at indices i-1 and j-1 in seq1 and seq2, respectively. If the characters are the same, the match score is equal to dp[i-1][j-1] + match_score, otherwise it is equal to dp[i-1][j-1] + mismatch_score.

The delete score, which is the score for aligning the character at index i-1 in seq1 with a gap in seq2. The delete score is equal to dp[i-1][j] + open_gap_score + extend_gap_score.

The insert score, which is the score for aligning the character at index j-1 in seq2 with a gap in seq1. The insert score is equal to dp[i][j-1] + open_gap_score + extend_gap_score.

The code then takes the maximum of the three scores and assigns it to dp[i][j]. This process is repeated for every pair of indices (i, j), filling in the matrix dp with the scores for aligning the characters at each pair of indices.

```
i = m
j = n
while i > 0 or j > 0:
    if i > 0 and j > 0 and dp[i][j] == dp[i-1][j-1] + (match_score if seq1[i-1] == seq2[j-1] else mismatch_score):
        align1 = seq1[i-1] + align1
        align2 = seq2[j-1] + align2
        i -= 1
        j -= 1
    elif i > 0 and dp[i][j] == dp[i-1][j] + open_gap_score + extend_gap_score:
        align1 = seq1[i-1] + align1
        align2 = "-" + align2
        i -= 1
    else:
        align1 = "-" + align1
        align2 = seq2[j-1] + align2
        j -= 1
# Return the alignments and the score
return align1, align2, dp[m][n]
```

This code is a while loop that traces back through the matrix dp to retrieve the optimal alignment of the two DNA sequences, seq1 and seq2. The loop continues until either i becomes 0 or j becomes 0, which corresponds to reaching the beginning of one of the sequences.

At each step of the loop, the code checks the value of dp[i][j] and compares it to the values of dp[i-1][j-1], dp[i-1][j], and dp[i][j-1]. Based on the value of dp[i][j], the code determines which alignment operation was used to reach the current position in the matrix, and updates the variables align1 and align2 accordingly.

If dp[i][j] is equal to dp[i-1][j-1] + (match_score if seq1[i-1] == seq2[j-1] else mismatch_score), then the characters at indices i-1 and j-1 in seq1 and seq2 were

aligned. The code appends these characters to the beginning of align1 and align2, respectively, and decreases the values of i and j by 1.

If dp[i][j] is equal to dp[i-1][j] + open_gap_score + extend_gap_score, then the character at index i-1 in seq1 was aligned with a gap in seq2. The code appends this character to the beginning of align1 and a gap to the beginning of align2, and decreases the value of i by 1.

If none of the above conditions are met, then the character at index j-1 in seq2 was aligned with a gap in seq1. The code appends a gap to the beginning of align1 and this character to the beginning of align2, and decreases the value of j by 1.

After the loop has finished, the code returns the aligned sequences, align1 and align2, and the alignment score, dp[m][n].

Outputs

--*-*-*-*-*-* Test1.seq *-*-*-*-*-*-*-*

First given Dna align is : -C-GAGACCGA-C-GAAGAGGTT--TGGC--CCCA-A--CCAGGTTCCCT---GA---TCACGTA--ACT-TACCGGCCAAAAGGACTGGCCTTACTAAG-GCCTTTGTCTACTGC-G-G--GT-C--CGG-G-GGC-CGTT--GGT-T-TC-GGCAGAACACTTC

Second given Dna align is : CCTG-GACCGAGCTTAA-A--TTGCTAGCAATACAGATGCC-GCTTCCTTGGGGAGGGT-GTGTAGGA-TGTA--GG--TTAACGAAT-G-----C-AAGTTCCGGGGTAT-C-GCAGAGTCGTGCTACGGCGTGGCAC-TTAGGGTCTCTCGGGAAAAAGAGTAG

Calculated dynamic sequence alignment socre is : **210.0**

--*-*-*-*-*-* Test2.seq *-*-*-*-*-*-*-*

First given Dna align is : GTGT-GGTTGCTTGCATCACTCCGTGTACATGTGACAACCGAACGAGTTGATCCAGCTTTGTTAAGTCAGCTTCGAATG--C---G-G-T--AGCTCTCA--AA--TATGATATGACTCTTGGGGTAGATGCTGGGGACCTATTGCGC-C-CAAAGCGATAT--TCGGGGCA-CCGGTTTAGGGTACCCTATCAAGGCGAT-AC-TT--CGATGCAGTGTGATGCCGG-A-G-GTGTCG-GTCAGCATGTG--A-GAGCT

Second given Dna align is : G-GTAGGTT-CGTGAAGCACTCC-TGGAC-TCTGACCA-C-AAC-A--TAATCAAGC---G--CAG--AG-TT-GAATGACCAAAGCGCTCAAGCTTTCACGAACGTCTCATAT--C-C---GCG--G-T-C---GTA-CAAACGCGCTCTTAAAAC-A-ATCGTCTATCCATCCGG----GGATA-CC-A--ATGG-G-TGACATTAAC--TG--G-G-CAGGCCGGCACGCG-GACGCGACAG-AT-TGAAATG-GAT

Calculated dynamic sequence alignment socre is : **369.0**

--*-*-*-*-*-* Test3.seq *-*-*-*-*-*-*-*

First given Dna align is : C----------G---G----------G-GA-AAGA-CGG--A-ATGCATCG-ACCATCGGACAAT-GC-CTCACTGGAAGCGCTG--CTGATTTTTGCGCA-ACGAGCCTCG--GACCTCCCG-C-TCAAA-CT-TACGAA--A-ATGACTCCACC-----AG-CACTGAA-CCAACTGGCCTCCAG-TGA-AGAT---AGTAT-CT---A-CA--A-A-TC-G--TT---TGC-C-GGGAGAAA-----C--ATT-TG-T---A-----TGGT--A-G-TCAGCGT-T--CT--G-CACGTCACGT-AATCGTTCACTAGT-TGTGGGGTATACCAGG-TCGTAAT--GAGATGTT-A-G-T-A--TG--A---ATCGTTTATAA-CGCT-TGTCATAGGAGT-TCCGAATAATCGTC--ACT-A-GGTCAA-TGGC--C--C---C-CTACTTGTAATAACT-ACGTC-TGATTGAGAAACAGAT-CGTTAGTCATCGGTTAATAGTCC-CGGAAGATAACGGGTT---CTTGC-GT---T-T-----T---TG-CGAATACTACTATCTCATGGCGAT-GGAGCGT-TTGGTCCCATCCAGCCG-CGCGAGTATC-ACTTGTTCGCC-TGCACCT-GTC-C--GACCTTTCGATGGGGAGCTCCTTTCATTGGTTGTAGGTACTAAGGGTGAGCAATGTCACGTGACCCAA-GGA-GCCGTGTGTAATTTCCACTTGCTCAGAAAAG--CCTC-GAC---A-AT--CTG-GG-A-CCGACACCTGAGTG-AT-GGCTTACA-GACCAGGGGAG-GG-GGGCAGGTTCCCTGGCCACAGAATGGCACG-CC-CTGAGGAGGCACCGGCCCAA-CGTCG-CTGG

Second given Dna align is : -ACGTCACATGCCAGGGGGCGGTGTGCAGTTCTCGACCGGAAACGCCCACCCA-CGA-CGTACCGCCAAAGCGTCGCCTGG

AGTGAGAAGACCGGATATATGCAACGAACAATCGCAAAATAGCTC-CA-TGTACGC-CTGTCCCG-TAATACCGCATGCGAGCCTCGAAG-CC-CCCGACGTCAAACCTCAAC-AAGTATAGGGTTCCACCGTATGAGCCACAGAATTC-AGT-TCCTCC-GCTGATAG-TCCCGGT-TCCTCCAATCAGCAGATTCTGACTTAGGT-CACTGGAAGAAACCTGGCGTATTGTGATGAAATTTCGTGGTGGACGCT-ACCGTCTAGGTAGAGCCAC-T-A-GTGCATTGTACACT-GTCT-T----T-TTCC-GGCT-ATAATCGGAG-T-TTCAGGCTGACTTGCCACCCA--GGTCA-AATAG-TATGTCA-CGGTGTAT--G-AT-CTC-TCGGGCTGATAG-CAAGGGGCGGCGTCGGGCACTCCTTG-AAT---TGAC--CAT--TTG-G---C-G-TGCCTT-GT--T-GCCTCCT--TCCTC--AAG--CA-GGGTTAGGCTT-CAGTAAGTGTGGGGGTGGCTGCCG-A-A--A-GA---C--GG-GCTCGG-G-GTGTT-G-CCCA---A--CGTC-TG-G--TCGAC---TTC-CCAT-AATCTGGGCACAAGA-C-GT--AT--GGA---CC---C----G-TG-CGCT--TAATGGT--GC-ATTTCACGCGA-GAAACGGAGGCAG-GTGCCA--TCCGC--G-GC-GAAAAGATCATCAGACATGACATAAC-GAGGAACCCGA-TCCGGAGTGAATACG-TCACATG-CCAGGGG-GCGGTGTGCA-GTT--CTCGAC-CGGAA----ACGCCCACCCACGACGTACC-GCCAAAGCGTCGCCTGG

Calculated dynamic sequence alignment socre is : **1209.5**

--*-*-*-*-*-* Test4.seq *-*-*-*-*-*-*-*

First given Dna align is : A------G---G----------C--CGAAAACGTCGCGAAT-T-GACCCTGGCGACGCCGCCGAACGGGACCTCCGTTAGT-G---T-GGGAGGTCATCAATCTCGTTCGCTAGCGGCTGACAC--C-AATCACTATAAG-TCTGTCATGAC

Second given Dna align is : -TAAACA-AGA-AGATAATCG

CTTC---AA-GT--CAAATATAGATCCTGGC--CG-CTCC--ACGGG--CT---TAAGTCGTTCTCCGAAGGT-A-CGATCTGGTT-G---GATGCT-TC-CGTCTAA--ACAAGAAGAT-AATC--G--

Calculated dynamic sequence alignment socre is : **179.0**

--*-*-*-*-*-* Test5.seq *-*-*-*-*-*-*-*

First given Dna align is : -CGG--GTAGTTAACCCT-ACA-GCATAGAGTCGCGAGATAAAGTGCAGGA-GTCTTTCGCGGCAGATTCGTACCTCA---ACCACGTGCTACTT---TCTGGCATCACGAATCTGCCGCATAGGTCCGTGAGT-CCATATGA

Second given Dna align is : A-GGAAGTAGTT-AGCCTAACAGGCATAGAGTCGCGACAT-ATGTG-AAGATGTCATT--CGG--TATTCAAACCTCATGCATCA-TTGC--CTTGAGTC--GC-T--C-----CTGGAGCATA-GTCCCTGAGTGCCATATGA

Calculated dynamic sequence alignment socre is : **261.5**