Global alignment algorithm with affine gap model

author: 张英豪

Algorithm

The principle of the algorithm is shown in the following two slides.

Summary of the dynamic programming solution

- Basis:
  - V(0, 0) = 0; V(i, 0) = -h-is; V(0, j) = -h-js
  - E(0, 0) = -∞; E(i, 0) = -∞; E(0, j) = -h-js
  - F(0, 0) = -∞; F(i, 0) = -h-is; F(0, j) = -∞
- Recurrence:
  - E(i, j) = max { E(i, j-1)–s, D(i, j-1)–h–s }
  - F(i, j) = max { F(i-1, j)–s, D(i-1, j)–h–s }
  - V(i, j) = max { V(i-1, j-1) + δ(S[i], T[j]), E(i, j), F(i, j) }

Global alignment algorithm with affine gap penalty

1. V(0, 0) = 0; E(0, 0) = -∞; F(0, 0) = -∞;
2. For i = 1 to n
   - V(i, 0) = -h-is; E(i, 0) = -∞; F(i, 0)=-h-is;
3. For j = 1 to m
   - V(0, j) = -h-js; E(0, j)=-h-js; F(0,j)= -∞;
4. For i = 1 to n
   - For j = 1 to m
     - E(i, j) = max { E(i, j-1)–s, V(i, j-1)–h–s }
     - F(i, j) = max { F(i-1, j)–s, V(i-1, j)–h–s }
     - V(i, j) = max { V(i-1, j-1) + δ(S[i], T[j]), E(i, j), F(i, j) }
5. Report V(n, m)

Base case

Recursive case

My codes

I wrote two programs to implement global alignment algorithm with affine gap model. Their filenames are `align_DP.py` and `align_DP_multi.py`.

In `align_DP.py`, I have fulfilled the requirements of the assignment. It can produce the same file as the reference output.

However, sequence alignment sometimes has multiple backtracking paths. In order to solve this problem, I wrote `align_DP_multi.py` program to achieve the output of multiple backtracking paths.

The core code for the model is as follows:

```
def affine_gap_model(seq1, seq2, alphabets, score_matrix, gap_open_penalty,
gap_extend_penalty):
    """
       Calculates the optimal global alignment between two sequences using
the affine gap penalty model.
       Args:
           seq1 (str): The first input sequence.
           seq2 (str): The second input sequence.
           alphabets (str): A string of characters representing the
alphabet used in the input sequences.
           score_matrix (ndarray): A ndarray of score matrix
           gap_open_penalty (int): The penalty for opening a gap in one of
the sequences.
           gap_extend_penalty (int): The penalty for extending an existing
gap in one of the sequences.
       Returns:
           aligned sequence 1, aligned sequence 2, alignment score
    """
    n = len(seq1)
    m = len(seq2)
    V = np.zeros((n + 1, m + 1))
    # E矩阵，用来存储gap在seq1中的值 F矩阵，用来存储gap在seq2中的值
    E = np.zeros((n + 1, m + 1))
    F = np.zeros((n + 1, m + 1))
    # 这样可以方便索引，可以直接根据字母判断是否匹配
    replace = {}
    for i, alphabet in enumerate(alphabets):
        replace[alphabet] = i
    # 初始化矩阵
    for i in range(n + 1):
        E[i][0] = -np.inf
        F[i][0] = gap_open_penalty + i * gap_extend_penalty
```

```
        V[i][0] = gap_open_penalty + i * gap_extend_penalty
    for j in range(m + 1):
        E[0][j] = gap_open_penalty + j * gap_extend_penalty
        F[0][j] = -np.inf
        V[0][j] = gap_open_penalty + j * gap_extend_penalty
    E[0][0] = -np.inf
    F[0][0] = -np.inf
    V[0][0] = 0
    # 填矩阵
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            E[i][j] = max(E[i][j - 1] + gap_extend_penalty, V[i][j - 1] +
gap_open_penalty + gap_extend_penalty)
            F[i][j] = max(F[i - 1][j] + gap_extend_penalty, V[i - 1][j] +
gap_open_penalty + gap_extend_penalty)
            match = V[i - 1][j - 1] + score_matrix[replace[seq1[i - 1]],
[replace[seq2[j - 1]]]]
            delete = E[i][j]
            insert = F[i][j]
            V[i][j] = max(match, delete, insert)
    score = V[n][m]
    # 回溯
    # alignments = []
    aligned_seq_1 = []
    aligned_seq_2 = []
    def traceback(i, j, align_seq_1, align_seq_2):
        if i == 0 and j == 0:
            aligned_seq_1.append(align_seq_1[::-1])
            aligned_seq_2.append(align_seq_2[::-1])
            return
        if i > 0 and j > 0 and V[i][j] == V[i - 1][j - 1] +
score_matrix[replace[seq1[i - 1]], [replace[seq2[j - 1]]]]:
            traceback(i - 1, j - 1, align_seq_1 + seq1[i - 1], align_seq_2
+ seq2[j - 1])
        if i > 0 and V[i][j] == F[i][j]:
            traceback(i - 1, j, align_seq_1 + seq1[i - 1], align_seq_2 + '-
')
        if j > 0 and V[i][j] == E[i][j]:
            traceback(i, j - 1, align_seq_1 + '-', align_seq_2 + seq2[j -
1])
    traceback(n, m, '', '')

    return aligned_seq_1, aligned_seq_2, score
```

It should be noted that I modified the `write_output()` in order to output multiple backtracking paths.

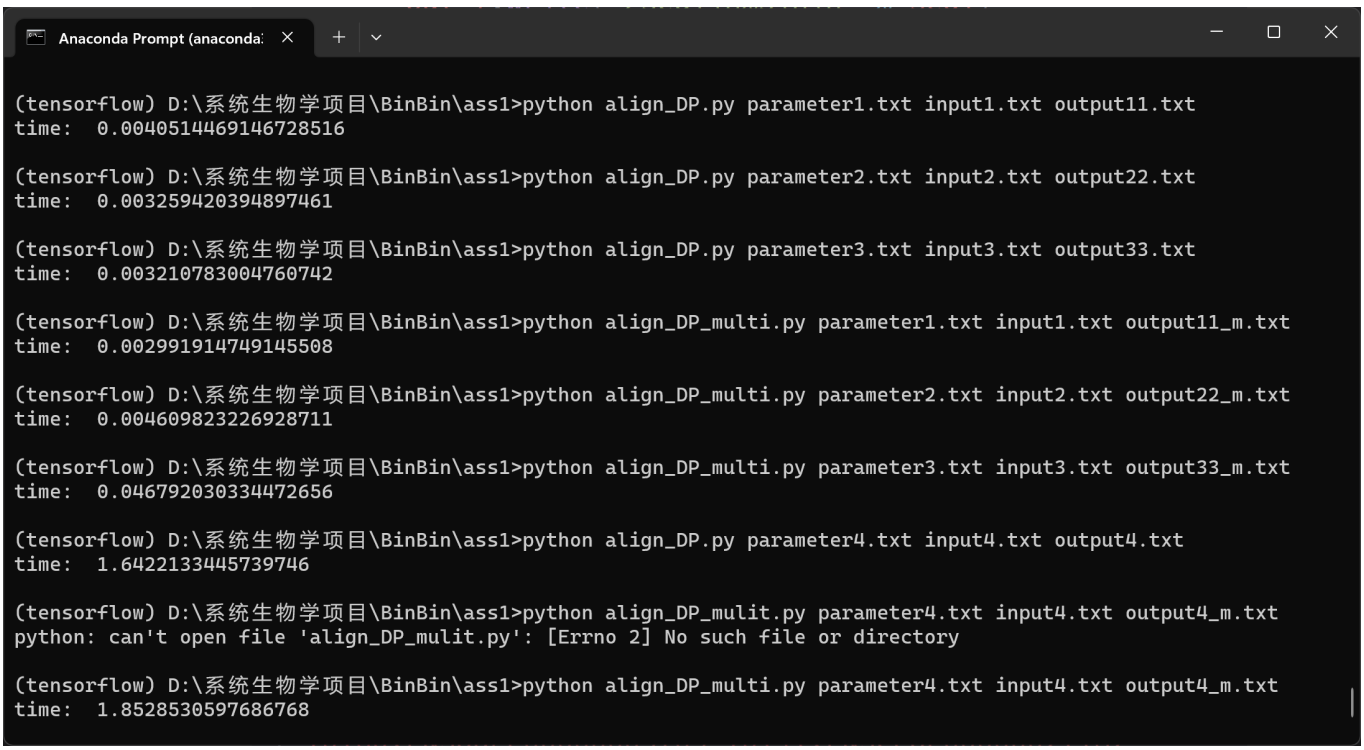
```python
# function to write output.txt
def write_output(outfile, max_score, seq1, seq2):
    with open(outfile, 'w') as out_f:
        out_f.write("score = %d\n" % max_score)
        out_f.write(">seq1\n%s\n\n" % seq1)
        out_f.write(">seq2\n%s" % seq2)
```

How to run the codes?

You can run it with the following command.

```
python align_DP.py parameter.txt input.txt output.txt
python align_DP_multi.py parameter.txt input.txt output.txt
```

The running process is below, and the result can be seen in the files.



Time analysis

We need to fill in 3 tables, each is of size n×m.

- Space complexity = O(nm)

Each entry can be computed in O(1) time.

- Time complexity = O(nm)

Multiple backtracking paths take longer than one backtracking path.

Homology sequences

I download these two sequences of PD-L1. These sequence are stored in `input4.txt` and the parameters in `parameter4.txt`.

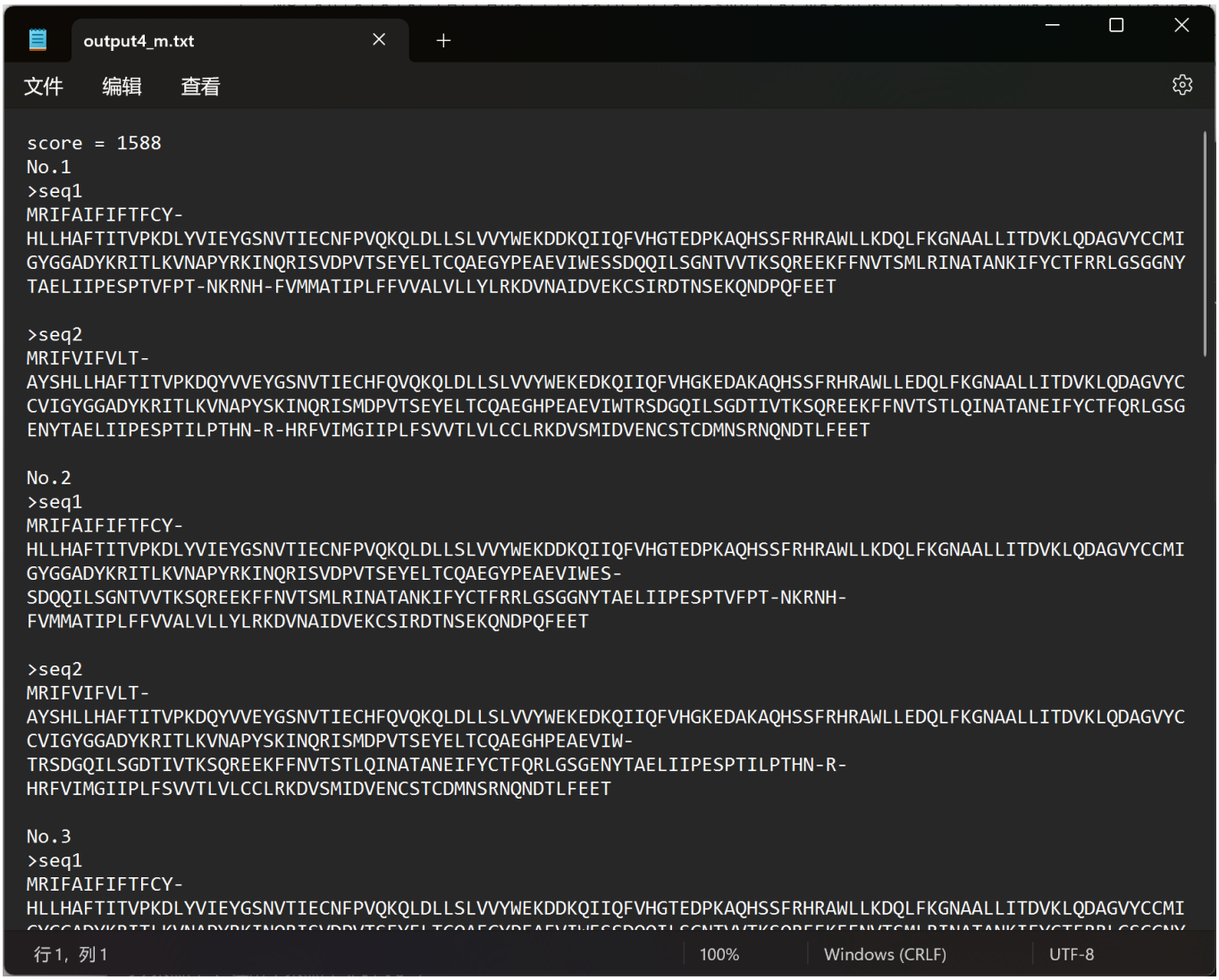
eggnogapi6.embl.de/get_sequence/10160.ENSODEP00000005705

eggnogapi6.embl.de/get_sequence/10141.ENSCPOP00000027182

```
; 10160.ENSODEP00000005705 10141.ENSCPOP00000027182
>seq1
MRIFAIFIFTFCYHLLHAFTITVPKDLYVIEYGSNVTIECNFPVQKQLDLLSLVVYWEKDDKQIIQFVHGTEDPK
AQHSSFRHRAWLLKDQLFKGNAALLITDVKLQDAGVYCCMIGYGGADYKRITLKVNAPYRKINQRISVDPVTSEY
ELTCQAEGYPEAEVIWESSDQQILSGNTVVTKSQREEKFFNVTSMLRINATANKIFYCTFRRLGSGGNYTAELII
PESPTVFPTNKRNHFVMMATIPLFFVVALVLLYLRKDVNAIDVEKCSIRDTNSEKQNDPQFEET

>seq2
MRIFVIFVLTAYSHLLHAFTITVPKDQYVVEYGSNVTIECHFQVQKQLDLLSLVVYWEKEDKQIIQFVHGKEDAK
AQHSSFRHRAWLLEDQLFKGNAALLITDVKLQDAGVYCCVIGYGGADYKRITLKVNAPYSKINQRISMDPVTSEY
ELTCQAEGHPEAEVIWTRSDGQILSGDTIVTKSQREEKFFNVTSTLQINATANEIFYCTFQRLGSGENYTAELII
PESPTILPTHNRHRFVIMGIIPLFSVVTLVLCCLRKDVSMIDVENCSTCDMNSRNQNDTLFEET
```

The result is below:



output4_m.txt

文件 编辑 查看

```
score = 1588
No.1
>seq1
MRIFAIFIFTFCY-
HLLHAFTITVPKDLYVIEYGSNVTIECNFPVQKQLDLLSLVVYWEKDDKQIIQFVHGTEDPKAQHSSFRHRAWLLKDQLFKGNAALLITDVKLQDAGVYCCMI
GYGGADYKRITLKVNAPYRKINQRISVDPVTSEYELTCQAEGYPEAEVIWESSDQQILSGNTVVTKSQREEKFFNVTSMLRINATANKIFYCTFRRLGSGGNY
TAELIIPESPTVFPT-NKRNH-FVMMATIPLFFVVALVLLYLRKDVNAIDVEKCSIRDTNSEKQNDPQFEET

>seq2
MRIFVIFVLT-
AYSHLLHAFTITVPKDQYVVEYGSNVTIECHFQVQKQLDLLSLVVYWEKEDKQIIQFVHGKEDAKAQHSSFRHRAWLLEDQLFKGNAALLITDVKLQDAGVYC
CVIGYGGADYKRITLKVNAPYSKINQRISMDPVTSEYELTCQAEGHPEAEVIWTRSDGQILSGDTIVTKSQREEKFFNVTSTLQINATANEIFYCTFQRLGSG
ENYTAELIIPESPTILPTHN-R-HRFVIMGIIPLFSVVTLVLCCLRKDVSMIDVENCSTCDMNSRNQNDTLFEET

No.2
>seq1
MRIFAIFIFTFCY-
HLLHAFTITVPKDLYVIEYGSNVTIECNFPVQKQLDLLSLVVYWEKDDKQIIQFVHGTEDPKAQHSSFRHRAWLLKDQLFKGNAALLITDVKLQDAGVYCCMI
GYGGADYKRITLKVNAPYRKINQRISVDPVTSEYELTCQAEGYPEAEVIWES-
SDQQILSGNTVVTKSQREEKFFNVTSMLRINATANKIFYCTFRRLGSGGNYTAELIIPESPTVFPT-NKRNH-
FVMMATIPLFFVVALVLLYLRKDVNAIDVEKCSIRDTNSEKQNDPQFEET

>seq2
MRIFVIFVLT-
AYSHLLHAFTITVPKDQYVVEYGSNVTIECHFQVQKQLDLLSLVVYWEKEDKQIIQFVHGKEDAKAQHSSFRHRAWLLEDQLFKGNAALLITDVKLQDAGVYC
CVIGYGGADYKRITLKVNAPYSKINQRISMDPVTSEYELTCQAEGHPEAEVIW-
TRSDGQILSGDTIVTKSQREEKFFNVTSTLQINATANEIFYCTFQRLGSGENYTAELIIPESPTILPTHN-R-
HRFVIMGIIPLFSVVTLVLCCLRKDVSMIDVENCSTCDMNSRNQNDTLFEET

No.3
>seq1
MRIFAIFIFTFCY-
HLLHAFTITVPKDLYVIEYGSNVTIECNFPVQKQLDLLSLVVYWEKDDKQIIQFVHGTEDPKAQHSSFRHRAWLLKDQLFKGNAALLITDVKLQDAGVYCCMI
```

行 1，列 1 100% Windows (CRLF) UTF-8

Code avalibility

The code of `Global alignment algorithm with affine gap model` and the scripts to generate the results shown in this paper are available at https://github.com/grassdream/Affine-gap-model.