Domain-aware Aligner Documentation

Group 5

December 15, 2022

1 Major task

This software developed an algorithm to align protein sequences considering their domain composition. Modular proteins can undergo domain loss or acquisition because of long genomic insertions, deletions, and duplications. Current alignment algorithms maximize an alignment score based on a match/mismatch and gap score. The domain content of a protein can be predicted with tools like Pfam or InterPro, and the predictions are already available in databases such as Uniprot. **Proteins with various domain architectures frequently produce poor results when aligned using current, reliable tools**. With the use of pairwise alignments and an algorithm that considers domain architectures, this software aims to produce all multiple sequence alignments.

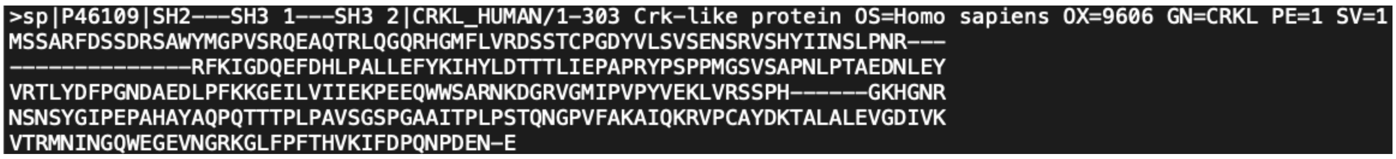
2 Input and Output

1. The **input section** includes the original sequences data (**fasta, txt**...) and annotated **tsv** data for target protein families from the Uniprot database (annotation data is an **optional input**). You can also use the built-in simulate data of this algorithm.

2. The **output section** includes the following four parts:
   - **Fasta** result from Domain-wise Greedy MSA
   - **Fasta** result from HMM-based Domain-level Alignment
   - Running logs of the algorithm (including unmatched information, and will be put on the screen)
   - Sum-of-pairs score of the alignment result (**optional**)

3 Key algorithms

Our core algorithm can be divided into two modules: domain-wise greedy MSA and HMM-based domain-level alignment. Domain-wise greedy MSA module categorizes protein sequences according to their domain composition (categorize_seqs_by_domain_info), and then use a progressive greedy algorithm to perform MSA on each of the domains and linkers in the sequences for each category. Specifically, hierarchical clustering was performed on all sequences based on hamming distance between their pairwise alignments (calculate_pairwise_distance_matrix). Then, MSA was generated gradually by merging sequence clusters based on the clustering result (merge_two_sequence_clusters). Pairwise alignments are implemented using a dynamic programming approach with three score and backtrack matrices



```
>sp|P46109|SH2---SH3 1---SH3 2|CRKL_HUMAN/1-303 Crk-like protein OS=Homo sapiens OX=9606 GN=CRKL PE=1 SV=1
MSSARFDSSDRSAWYMGPVSRQEAQTRLQGQRHGMFLVRDSSTCPGDYVLSVSENSRVSHYIINSLPNR---
--------------RFKIGDQEFDHLPALLEFYKIHYLDTTTLIEPAPRYPSPPMGSVSAPNLPTAEDNLEY
VRTLYDFPGNDAEDLPFKKGEILVIIEKPEEQWWSARNKDGRVGMIPVPYVEKLVRSSPH------GKHGNR
NSNSYGIPEPAHAYAQPQTTTPLPAVSGSPGAAITPLPSTQNGPVFAKAIQKRVPCAYDKTALALEVGDIVK
VTRMNINGQWEGEVNGRKGLFPFTHVKIFDPQNPDEN-E
```

Figure 1: CRK's category wise results.

to represent three affine gaps and match states (affine_gap_penalties_pair_wise_alignment). Then domain-level information was then used to build a customized Hidden Markov Model (HMM) to profile domain arrangement pattern (construct_HMM). A special state, 'SWAP' was added to this model, to represent the situation that two domains exchanging their position to map sequence (align_to_SWAP_HMM). We implement the Viterbi learning algorithm to iteratively train our model (HMM_Viterbi_Learning). After the model has been well trained, we perform HMM-based align for each input sequence (merge_hidden_paths) and merge then to a uniform pattern for further operation including benchmarking. For further implementation details including IO and helper functions, please refer to our source code.

4 Setting up running environment

- This program requires python 3.7 or above. Please first ensure the python environment of the device
- The input sequences should be biological data of queriable modular proteins with **at least 2 domains** with **Uniprot Entry ID** (for compatibility with HMM model) downloadable from UniProt database. We also support testing on an internally simulated data set.
- Make sure that all python dependency packages have been correctly installed (shown in Installation)

5 Installation

There are three prerequisites that users need to satisfy:

1. Downloaded the compressed file of the software (**zip file**)
2. Input: annotated information provided by the Uniprot database (**optional**, not necessary in networking conditions)
   the **tsv** file should contain at least two columns of **Entry** and **Domain** [**FT**]
3. Installed the **python modules** required by the software algorithm:
   - **numpy 1.23.5**
   - **blosum 1.2.2**
   - **argparse 1.4.0**
   - **pandas 1.5.2**
   - **bs4 0.0.1**

Use **pip** to install the required modules:
```
$ pip install numpy blosum argparse pandas bs4
```

6 Parameters and Explanation

- **-h (Optional)** it will return a explanation of the parameters usage
- **-i (Optional)** the path to read in raw sequences file, or **ignore this parameter to use the build-in simulated data**
- **-o (Required)** the path to save the result fasta file, default path is current system path
- **-ref_mode (Required)** ref_mode must be in one of [**'tsv', 'uniprot', 'simulate'**], **database reference selection**, 'simulate' means to use the build-in simulate data
- **-r** If ref_mode is in **'tsv'**, r must be the path to read in uniprot reference tsv file
- **-sigma (Optional)** set the gap open penalty, enter an **integer**, default value is 11
- **-epsilon (Optional)** set the gap extension penalty, enter an **integer**, default value is 1

7 Test running instruction

You can follow the following reference steps to run the program:

- A zip file of the program is available from our GitHub
- Download the zip file of the program to a local directory, and use the **$ unzip** command on the terminal to extract the program
- Use the **$ cd** command to change the current path to the program folder
- Run the code using the following usage template:
  1. $ python main.py **-i** SRC.fasta **-o** **-ref_mode** uniprot **-sigma** 10
  2. $ python main.py **-o** **-ref_mode** simulate
  3. $ python main.py **-i** CRK.txt **-o** **-ref_mode** tsv **-r** /crk_uniprot.tsv
  4. $ python eval.py **-i** result_HMM.fasta

8 Readme file

This software is based on **python3.7 or later**. For data alignment, this algorithm will **occupy a large memory**. Therefore, please use it properly based on your operating device. Before using this software, ensure that you have installed python packages such as **numpy 1.23.5, blosum 1.2.2, argparse 1.4.0, pandas 1.5.2, and bs4 0.0.1** in your python environment. This software zip file contains five python script files, as well as three test cases.

The input sequences should be biological data of queriable modular proteins with **at least 2 domains** with **Uniprot Entry ID** (for compatibility with HMM model) downloaded from UniProt database. We also support testing on an internally simulated data set.

What you could to operate are the **main.py and eval.py**.
main.py will produce the domain-wise greedy MSA alignment, the alignment sequence result of HMM-based Domain-level alignment and the visualization figure.
eval.py will calculate the sum-of-pairs score of the alignment.

Usage of this software:
$ python main.py **-i** [input file path] [**-o** output file path] **-ref_mode** ['tsv', 'uniprot', 'simulate'] [**-r** uniprot tsv file path] [**-sigma** int] [**-epsilon** int]
$ python eval.py **-i** [input file path]

Example1:
$ python main.py **-i** SRC.fasta **-o** **-ref_mode** uniprot **-sigma** 10
Example2:
$ python main.py **-o** **-ref_mode** simulate
Example3:
$ python main.py **-i** CRK.txt **-o** **-ref_mode** tsv **-r** /crk_uniprot.tsv
Example3:
$ python eval.py **-i** result_HMM.fasta

Developers:
Biomedical Informatics 3 mini-project Group5