PROBCONS

Version 1.08

User Manual

Written by

Mahathi Mahabhashyam
mmahathi@cs.stanford.edu

and

Chuong Do
chuongdo@cs.stanford.edu

Overview

PROBCONS is a novel tool for generating multiple alignments of protein sequences. Using a combination of probabilistic modeling and consistency-based alignment techniques, PROBCONS has achieved the highest accuracies of all alignment methods to date.

The basis for the PROBCONS algorithm is the computation of *pairwise posterior probability matrices*, $\mathbf{P}(x_i \sim y_j \mid x, y)$, which give the probability that one should match letters x_i and y_j when aligning two sequences x and y. PROBCONS uses a simple probabilistic model that allows for efficient computation of these probabilities. Given these posterior matrices, PROBCONS applies the *probabilistic consistency transformation* to incorporate evidence from intermediate sequences. Finally, PROBCONS performs progressive alignment using a sum-of-pairs *maximum expected accuracy* objective function.

Algorithm Summary

Given a set of sequences to be aligned:

1. Compute *posterior probability matrices* for each pair of sequences.
2. Compute the *expected accuracy* of each alignment.
3. Apply the *probabilistic consistency transformation* to posterior matrices.
4. Compute a *guide tree* using the expected accuracies.
5. *Progressively align* the sequences using the guide tree.

References

PROBCONS is discussed in the following papers:

- Do, C.B., Brudno, M., and Batzoglou, S. 2004. PROBCONS: Probabilistic Consistency-based Multiple Alignment of Amino Acid Sequences. To appear in ISMB.
- Do, C.B., Brudno, M., and Batzoglou, S. 2004. ProbCons: Probabilistic Consistency-based Multiple Alignment of Amino Acid Sequences. To appear in AAAI.

PROBCONS is public domain software for details. See README for details.

Getting Started

To install and use PROBCONS,

1. Download the latest version of the PROBCONS source code from
   http://probcons.stanford.edu/download.html
2. Decompress the files.
   ```
   gunzip probcons_vX_XX.tar.gz
   tar xvf probcons_vX_XX.tar
   ```
3. This will create a subdirectory called `probcons/` inside of the current directory.
4. Change to the `probcons/` directory, and make the PROBCONS executable.
   ```
   cd probcons
   make
   ```
5. Align the sequences in the file `input` and send the result to the file `output`.
   ```
   ./probcons input > output
   ```

That's it!

Input/Output Format

Any file used as input for PROBCONS should be a **text file**. This means that the program will not work with .doc files from MS word or other formatted word processing files. Most word processors allow the user to save a text file, by selecting "Save as" in the file menu.

MFA format for input/output

PROBCONS accepts files in the MFA format and produces output in MFA format. The MFA format is specified below:

- The MFA format consists of multiple sequences.
- Each sequence in MFA format begins with a single-line description, followed by lines of sequence data.
- The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column.

ClustalW (ALN) format for output

If the `-clustalw` option is specified, then a ClustalW output file is produced instead of the regular MFA:

- The ClustalW format consists of a single header line followed by sequence data in blocks of 50 alignment positions.
- Each block consists of
    - one line of data for each of the sequences in the alignment; in particular,
        - the name of the sequence
        - 50 characters of the alignment
    - one annotation line indicating fully conserved (`*`), strongly-conserved (`:`), or weakly-conserved columns (`.`)

Example Usage

Running PROBCONS on the following input file:[1]

```
>plas_horvu
DVLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSGVDVSKISQEEYL
TAPGETFSVTLTVPGTYGFYCEPHAGAGMVGKVTV
>plas_chlre
VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSGVNADAISRDDYLN
APGETYSVKLTAAGEYGYYCEPHQGAGMVGKIIV
>plas_anava
VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHKQL
LMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV
>plas_proho
VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDKVPAGESAPALSNTKLRI
APGSFYSVTLGTPGTYSFYCTPHRGAGMVGTITV
>azup_achcy
VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDKGHNVETIKGMIPDGAEAFKSKINENY
KVTFTAPGVYGVKCTPHYGMGMVGVVEV
```

will generate the following output:

```
>plas_horvu
-DVLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPS--GVDVSKISQE
EYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVTV
>plas_chlre
--VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPS--GVNADAISRD
DYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKIIV
>plas_anava
--VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHK
QLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV
>plas_proho
VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPA--GESAPALSNT
KLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV
>azup_achcy
VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK-GHNVETIKGMIPD--GAEA------
-FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV
```

[1] This sequence is `1plc_ref1` from the BAliBASE collection: Thompson, J.D., Plewniak, F., and Poch, O. 1999a. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics* **15**(1): 87-88.

If the `-clustalw` option is specified, then the following output is produced instead:

```
PROBCONS version 1.08 multiple sequence alignment

plas_horvu    -DVLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPS--GVDVSKISQE
plas_chlre    --VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPS--GVNADAISRD
plas_anava    --VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHK
plas_proho    VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPA--GESAPALSNT
azup_achcy    VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK-GHNVETIKGMIPD--GAEA------
              *.:    .     : *  :.: .*:*: *  .    **:       *   .  .  :*.

plas_horvu    EYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVTV
plas_chlre    DYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKIIV
plas_anava    QLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV
plas_proho    KLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV
azup_achcy    -FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV
              .     ..  ...: .   .* *   * ** * **** : *
```

Changing the number of insertion state pairs

For efficiency reasons, the number of insertion state pairs used in PROBCONS is fixed at compile time. To change the number of insertion state pairs used, edit the following line in the `Makefile`:

```
OTHERFLAGS = -DNumInsertStates=1 -DVERSION="1.08"
```

and then recompile the program with `make`. While any positive integer may be specified in the `Makefile`, default parameter values exist only when the number of insert state pairs is 1 or 2.

Command-line Options

PROBCONS offers several command-line options, which are detailed below.

General usage

```
./probcons [OPTION]... MFAFILE [MFAFILE]...
```

-clustalw

Use CLUSTALW output format instead of MFA.

Description:

Generates alignments in the ClustalW output format.

Example usage:

```
./probcons -clustalw input.mfa > output.aln
```

-c, --consistency REPS

Use 0 ≤ REPS ≤ 5 (default: 2) passes of consistency transformation.

Description:

Each pass applies one round of the consistency transformation on the set of sequences. The consistency transformation is described in detail in the mentioned papers. In each round, the aligner computes the consistency transformation for each pair of sequences using all other sequences. The aligner then updates the posterior probability matrices of the pairwise alignments.

Example usage:

```
./probcons -c 1 input.mfa > output.mfa
./probcons --consistency 1 input.mfa > output.mfa
```

-ir, --iterative-refinement REPS

Use 0 ≤ REPS ≤ 1000 (default: 100) passes of iterative refinement.

Description:

This specifies the number of iterations of iterative refinement to be performed. In each stage of iterative refinement, the set of sequences in the alignment is randomly partitioned into two groups. After projecting the alignments to these groups, the two groups are realigned, resulting in an alignment whose objective score is guaranteed to be at least that of the original alignment.

Example usage:

```
./probcons -ir 1000 input.mfa > output.mfa
./probcons --iterative-refinement 1000 input.mfa > output.mfa
```

-pre, --pre-training REPS

Use 0 ≤ REPS ≤ 20 (default: 0) rounds of pre-training before aligning the sequences.

Description:

This specifies the number of rounds of EM to be applied on the set of sequences being aligned. This option is used in case the default parameters are not appropriate for the particular sequences being aligned; in general, this option is not recommended as it may lead to unstable alignment parameters.

Example usage:

```
./probcons -pre 1 input.mfa > output.mfa
./probcons --pre-training 1 input.mfa > output.mfa
```

-pairs

Generate all pairwise alignments of all possible pairs of sequences.

Description:

When this option is selected, PROBCONS generates all pairs pairwise maximum expected accuracy alignments using the posterior matrices without generating a full multiple alignment. The names of the files are based on the header comments for each of the sequences in the original input file with `.fasta` appended. When the –clustalw option is selected, then `.aln` is used as a suffix instead.

Example usage:

```
./probcons -pairs input.mfa > output.mfa
```

where `input.mfa` consists of

```
>seq1
ATGC
>seq2
ATGC
>seq3
ATGC
```

generates the files

```
seq1-seq2.fasta
seq1-seq3.fasta
seq2-seq3.fasta
```

-viterbi

Use Viterbi decoding rather than maximum expected accuracy alignment.

Description:

Generates all-pairs pairwise alignments using the Viterbi algorithm. Note that this option has the effect of automatically turning on `-pairs`. This option is not recommended but is available for comparison to the maximum expected accuracy alignments.

Example usage:

```
./probcons -viterbi input.mfa > output.mfa
```

-v, --verbose

Report progress while aligning (default: off).

Description:

Turning on this option instructs the aligner to report the progress on all pairwise alignments during the initial alignment step, all consistency transformation calculations, and all iterative refinement steps.

Example usage:

```
./probcons -v input.mfa > output.mfa
./probcons --verbose input.mfa > output.mfa
```

-annot FILENAME

Write annotation for multiple alignment to file FILENAME.

Description:

Turning on this option causes the program to write quality scores for columns in the produced alignment to FILENAME. The quality score for each column of the alignment is given on a separate line and is an integer between 0 and 100 inclusive, representing the expected percentage of correct pairwise matches in the column. Columns containing only one non-gap character automatically have quality score 0.

Example usage:

```
./probcons –annot output.mfa.annotations input.mfa > output.mfa
```

generates the file `output.mfa.annotations`, containing

```
96
94
84
...
```

-t, --train FILENAME

Compute EM transition probabilities, store in FILENAME (default: no training).

Description:

This option is used to train the aligner using a set of sequences. The test sequences are read from the files specified in MFAFILE(s). Each file in MFAFILE(s) is taken as a separate training instance. This performs exactly one round of EM training on the sequences; multiple calls to PROBCONS are needed in order to obtain convergence. The training parameters are written to the file FILENAME in as three lines:

initMatchProb initInsertXProb initInsertYProb
startInsertXProb startInsertYProb
extendInsertXProb extendInsertYProb

Example usage:

```
./probcons -t trained.params input.mfa input2.mfa input3.mfa
./probcons --train trained.params input.mfa input2.mfa input3.mfa
```

generates the file `trained.params` with contents

```
0.9999138713 0.0000430496 0.0000430496
0.0144627076 0.0144627076
0.6306074262 0.6306074262
```

-p, --paramfile FILENAME

Read initial/final and transition probabilities from FILENAME (default: parameters read from Defaults.h).

Description:

This file specifies the initial/final probabilities and transition probabilities for the HMM model used by the aligner. The HMM model consists a Match state, and Insert X state, and an Insert Y state, and is described in more detail in the mentioned papers. The file format consists of three lines, containing:

initMatchProb initInsertXProb initInsertYProb
startInsertXProb startInsertYProb
extendInsertXProb extendInsertYProb

Example usage:

```
./probcons -p trained.params input.mfa > output.mfa
./probcons --paramfile trained.params input.mfa > output.mfa
```

where the file `trained.params` has contents

```
0.9999138713 0.0000430496 0.0000430496
0.0144627076 0.0144627076
0.6306074262 0.6306074262
```

-m, --matrixfile FILENAME

Read scoring matrix parameters from FILENAME (default: matrix read from the Defaults.h).

Description:

This file specifies the emission probabilities that are to be used for scoring alignments. By default, an emission probabilities based on the BLOSUM62 matrix are used. The file format consists of:

- one line with twenty letters specifying the order of the amino acid alphabet to be used in describing the pair emission probabilities
- twenty lines describing pair emission probabilities where the *n*th line:
  - contains *n* entries
  - the *m*th entry of the nth line gives the joint probability for emitting amino acid *m* with amino acid *n* (the matrix is assumed to be symmetric)
- one line with twenty letters specifying the order of the amino acid alphabet to be used in describing the single emission probabilities
- one line describing single emission probabilities
  - the *m*th entry gives the single emission probability for emitting amino acid *m* in an insertion state (assumed to be the same for insert X and insert Y states)

Example usage:

```
./probcons -m blosum62.matrix input.mfa > output.mfa
./probcons --matrixfile blosum62.matrix input.mfa > output.mfa
```

where the file `blosum62.matrix` contains:

```
A R N D C Q E G H I L K M F P S T W Y V
0.02373072
0.00244502 0.01775118
0.00210228 0.00207782 0.01281864
0.00223549 0.00161657 0.00353540 0.01911178
0.00145515 0.00044701 0.00042479 0.00036798 0.01013470
...
A R N D C Q E G H I L K M F P S T W Y V
0.07831005 0.05246024 0.04433257 0.05130349 0.02189704 ...
```