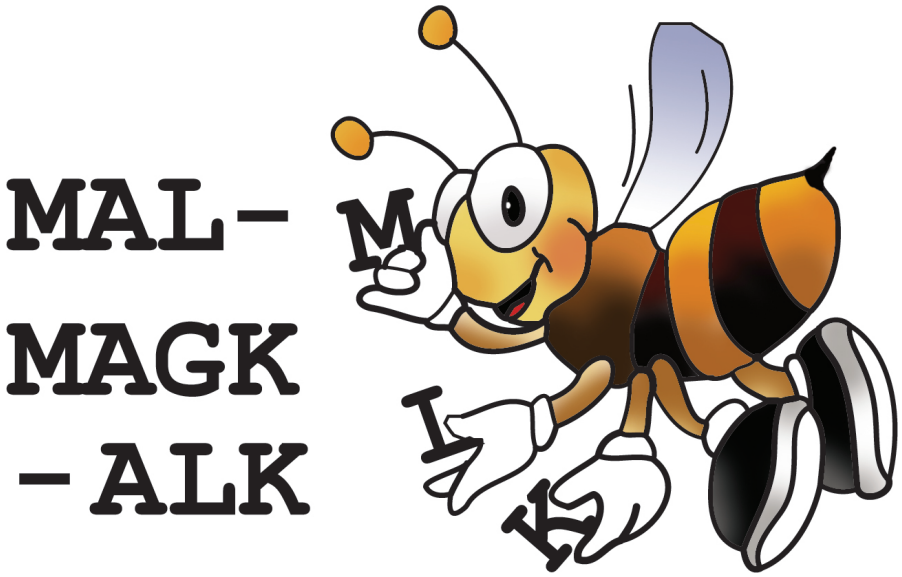

CONTRAlign 2.01

User Manual

(Last modified: August 14, 2008)

Contents

1 Description **2**

2 License (BSD) **3**

3 Installation **4**
3.1 *nix installation 4

4 Supported file formats **5**
4.1 Input file formats 5
4.1.1 MFA format 5
4.2 Output formats 6
4.2.1 MFA format 6
4.2.2 CLUSTALW format 6
4.2.3 Posteriors format 8

5 Usage **9**
5.1 Prediction mode 9
5.1.1 A single input file 9
5.1.2 Multiple input files 10
5.1.3 Optional arguments 11
5.2 Training mode 13

6 Citing CONTRAlign **15**

1 Description

CONTRAlign is a novel algorithm for multiple sequence alignment of protein and RNA sequences based on conditional log-linear models (CLLMs).

The CONTRAlign program was developed by Chuong Do at Stanford University in collaboration with Samuel Gross and Serafim Batzoglou. The source code for CONTRAlign is available for download from

http://contra.stanford.edu/contralign/

under the BSD license. The CONTRAlign logo was designed by Marina Sirota.

Any comments or suggestions regarding the program should be sent to Chuong Do (*chuongdo@cs.stanford.edu*).

2 License (BSD)

3 Installation

At the moment, CONTRAlign is only available for Unix-based systems (e.g., Linux). We will be porting CONTRAlign to other architectures and making the binaries available.

3.1 *nix installation

To compile CONTRAlign from the source code (for a *nix machine):

1. Download the latest version of the CONTRAlign source code from

   http://contra.stanford.edu/contralign/download.html

2. Decompress the archive:

   ```
   $ tar zxvf contralign_v#_##.tar.gz
   ```

   where the #'s are replaced with the appropriate version numbers for the tar.gz you want to install. This will create a subdirectory called `contralign` inside of the current directory.

3. Change to the `contralign/src` subdirectory.

4. Edit the file "Makefile" and select either protein or RNA alignment by supplying the appropriate definition of the `MODEL_TYPE` variable. For protein alignment, use

   ```
   MODEL_TYPE = -DRNA=0
   ```

   and for RNA alignment, use

   ```
   MODEL_TYPE = -DRNA=1
   ```

5. Compile the program.

   ```
   $ cd contralign/src
   $ make clean
   $ make
   ```

Now, your installation is complete!

4 Supported file formats

In this section, we describe the input and output file formats supported by the CONTRAlign program.

4.1 Input file formats

CONTRAlign accepts input files in MFA format. The sequences in the MFA format may either be unaligned sequences (containing no gaps) or pre-aligned sequences. When performing predictions, any gaps in the input files is ignored; however, pre-aligned sequences are required when performing training.

4.1.1 MFA format

An MFA (Multi-FASTA) format file consists of a collection of one or more FASTA-formatted sequences. Each sequence consists of:

1. A **single header line** beginning with the character '>' followed by a text description of the sequence. Note that the description must fit on the same line as the '>' character.

2. One or more lines containing **protein or RNA sequence data**.

   - For protein sequences, each of the lines may contain the letters 'A', 'R', 'N', 'D', 'C', 'Q', 'E', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', 'B', 'Z', or 'X' in either upper or lower case, and all other letters are automatically converted to X's.
   - For RNA sequences, each of these lines may contain the letters 'A', 'C', 'G', 'T', 'U' or 'N' in either upper or lower case, any T's are automatically converted to U's, and all other letters are automatically converted to N's.

   The output of the program will retain the case of the input. All whitespace (space, tab, newline) is ignored. X's and N's are treated as masked sequence positions which are ignored during all calculations (i.e., any scoring terms involving an X or N will be skipped). With the exception of gaps (see below), other non-whitespace characters are not permitted.

3. (Optional) In order to specify a pre-existing alignment, **gap characters** may be included in the sequence data above. In particular, valid gap characters include '.' and '-'. An MFA file specifying a gapped multiple sequence alignment should have the same total number of characters of sequence data (i.e., letters plus gaps) for each sequence.

For example, the following is a valid MFA file containing a single RNA sequence:

```
>sequence
acggagaGUGUUGAU
CUGUGUGUUACUACU
caucuguaguucuag
uugua
```

Similarly, the following is a valid MFA file with two RNA sequencess:

```
>seq1
acguuggcu
>seq2
gCGUCu
```

Also, the following is a valid MFA file with three-prealigned protein sequences:

```
>seq1
RNDCQEg
>seq2
RN--Qeg
>seq3
rnD-QEg
```

But the following is not a valid MFA file (starts with the wrong header character):

```
# sequence
ATGACGGT
```

4.2 Output formats

The results of a CONTRAlign alignment prediction are given in either MFA, CLUSTALW, or posteriors format. We describe each of these in detail.

4.2.1 MFA format

The MFA output format is identical to the MFA input format (see Section 4.1.1). The sequences in the output MFA file will always be given in the same order as in the input.

4.2.2 CLUSTALW format

The CLUSTALW output format for CONTRAlign is meant to be largely compatible with the output typically given by the CLUSTALW series of alignment programs. The output file consists of

1. A single line with the **header**, "CLUSTALW (CONTRAlign) multiple sequence alignment" followed by a blank line.

2. A list of **multiple alignment blocks** and annotation strings. Each block, except the last, is followed by a blank line. Each block consists of
   (a) One line for each sequence in the alignment, containing
      - the **name** of the sequence
      - a section of **aligned sequence data** (containing '-' characters for gaps)
   (b) An **annotation string** beneath the aligned sequence data, using the following annotation characters:
      - For protein alignments
        - '*' denotes a perfectly conserved amino acid residue
        - ':' denotes columns where all characters belong to one of the following sets:
          * {S, T, A}
          * {N, E, Q, K}
          * {N, H, Q, K}
          * {N, D, E, Q}
          * {Q, H, R, K}
          * {M, I, L, V}
          * {M, I, L, F}
          * {H, Y}
          * {F, Y, W}
        - '.' denote columns where all characters belong to one of the following sets:
          * {C, S, A}
          * {A, T, V}
          * {S, A, G}
          * {S, T, N, K}
          * {S, T, P, A}
          * {S, G, N, D}
          * {S, N, D, E, Q, K}
          * {N, D, E, Q, H, K}
          * {N, E, H, Q, R, K}
          * {F, V, L, I, M}
          * {H, F, Y}
      - For RNA alignments
        - '*' denotes a perfectly conserved amino acid residue

An example of a CLUSTALW format output alingment is shown below

```
CLUSTALW (CONTRAlign) multiple sequence alignment

seq1    RNDCRDCAQ
seq2    ---RNDCAQ
            .****
```

4.2.3 Posteriors format

The posteriors output format is applicable only when the requested output of CONTRAlign consists of pairwise sequence comparisons. The posteriors output format is distinct from the MFA and CLUSTALW formats in that in addition to an alignment, a posteriors output file also provides a sparse representation of the alignment match posterior probabilities for letters from the given input sequence pair.

For a pair of sequences x and y where the length of the first sequence x is L_x, the posteriors output format consists of:

1. A section giving an **MFA-formatted alignment prediction** (see Section 4.1.1). This prediction is computed using the maximum expected accuracy (MEA) decoding algorithm.
2. A single line containing the character '#' which serves as a **separator character**.
3. A section containing L_x lines, where the ith line contains
   (a) The integer i.
   (b) A space-separated list of base-pairing probabilities of the form j:p_{ij}, where p_{ij} is the alignment match probability for x_i and y_j.

For example, the following is a posteriors format output:

```
>seq1
RNDCRDCAQ
>seq2
---RNDCAQ
#
1 1:0.19849
2 2:0.188896
3 3:0.177315
4 1:0.69297 4:0.161425
5 2:0.710275 5:0.138814
6 3:0.744155 6:0.132637
7 4:0.78179
8 5:0.79337
9 6:0.796858
```

In the above output, the first C in sequence 1 is predicted to have a 69.297% chance of aligning with the R in sequence 2, and a 16.1425% chance of aligning with the C in sequence 2.

5 Usage

CONTRAlign has two modes of operation: prediction mode and training mode.

- In "prediction" mode, CONTRAlign aligns a set of sequences using either the default parameters or a CONTRAlign-format parameter file.
- In "training" mode, CONTRAlign learns new parameters from training data consisting of pre-aligned sequences.

Most users of this software will likely only ever need to use CONTRAlign's prediction functionality. The optimization procedures used in the training algorithm are fairly computationally expensive; for this purpose, the CONTRAlign program is designed to support automatic training in a parallel computing environment via MPI (Message Passing Interface).

5.1 Prediction mode

In prediction mode, CONTRAlign computes multiple alignments for given input sequence sets, and prints the result to either the console or output files. The basic syntax for running CONTRAlign in prediction mode is

```
$ ./contralign predict [OPTIONS] INFILE(s)
```

5.1.1 A single input file

For aligning a set of sequences, CONTRAlign generates MFA output (see Section 4.1.1) to the console (i.e., stdout) by default.

For example, suppose the file "seq.mfa" contains a set of sequence to be folded. Then the command

```
$ ./contralign predict seq.mfa
```

will fold the sequence and display the results to the console in MFA format.

CONTRAlign can also write MFA, CLUSTALW, or posteriors formatted output to an output file. To write MFA output to a file,

```
$ ./contralign predict seq.mfa --mfa out.mfa
```

To write CLUSTALW output to a file,

```
$ ./contralign predict seq.mfa --clustalw out.clustalw
```

To write all posterior pairing probabilities greater than 0.001 to a file,

```
$ ./contralign predict seq.mfa --posteriors \
       0.001 out.posteriors
```

Note that here, the backslash character is used to denote that a command-line is broken over several lines; it is not necessary if you type everything on a single line.

Finally, it is also possible to obtain multiple different types of output simultaneously. For example, the command

```
$ ./contralign predict seq.mfa --mfa \
       seq.mfa --clustalw seq.clustalw --posteriors \
       0.001 out.posteriors
```

will generate three different output files simultaneously.

5.1.2 Multiple input files

For multiple input files, CONTRAlign generates MFA output (see Section 4.1.1) to the console by default. The output is presented in the order of the input files on the command-line. Using console output is not allowed when MPI is enabled, or when certain other options are selected; in general, we recommend the usage of explicitly specified output files or subdirectories when dealing with multiple input files (see below).

CONTRAlign can also write MFA, CLUSTALW, or posteriors formatted output to several output files. In particular, CONTRAlign creates a subdirectory (whose name is specified by the user) in which to store the results, and writes each prediction to a file in that subdirectory of the same name as the original file being processed.

For example, suppose that the files "seq1.mfa" and "seq2.mfa" each contain sets of sequences to be independently aligned. Then the command

```
$ ./contralign predict seq1.mfa seq2.mfa \
       --mfa output
```

will create a subdirectory called `output` and will place the results in the files `output/seq1.mfa` and `output/seq2.mfa`.

Alternatively,

```
$ ./contralign predict seq1.mfa seq2.mfa \
       --clustalw output
```

and

```
$ ./contralign predict seq1.mfa seq2.mfa \
       --posteriors 0.001 output
```

generate CLUSTALW and posteriors formatted outputs instead.

Observe that if multiple input files have the same base name, then overwriting of output may occur. For example, if the input files list contains two different files called `seq/input` and `input`, the output subdirectory will contain only a single file called `input`.

You may also generate multiple types of output simultaneously, as before. Remember, however, to use different output subdirectory names for each. The command

```
$ ./contralign predict seq1.mfa seq2.mfa --mfa \
       mfa_output --clustalw clustalw_output \
       --posteriors 0.001 posteriors_output
```

generates three different output subdirectories (mfa_output, clustalw_output, and posteriors_output) each containing two files (seq1.mfa, seq2.mfa).

Finally, in some cases, you may wish to align a group of sequences from multiple MFA files. In this case, you can supply the `--conflate` option which tells CONTRAlign to treat the input files collectively as a single file (i.e., the concatenation of all input files) with the same name as the first specified file. For example, the command

```
$ ./contralign predict seq1.mfa seq2.mfa --conflate
```

will return a single MFA alignment consisting of all the sequences in both of the supplied input files.

5.1.3 Optional arguments

CONTRAlign accepts a number of optional arguments, which alter the default behavior of the program. To use any of these options, simply pass the option to the CONTRAlign program on the command line. For example,

```
$ ./contralign predict seq.mfa --viterbi \
       --noncomplementary
```

The optional arguments include:

`--gamma` γ

This option sets the sensitivity/specificity tradeoff parameter for the maximum expected accuracy decoding algorithm. In particular, consider a scoring system in which each nucleotide belonging to a correct match gets a score of γ, and each nucleotide correctly gapped gets a score of 1. Then, CONTRAlign finds the alignment of the input sequences with maximum *expected accuracy* with respect to this scoring system.

Intuitively,

- If $\gamma > 1$, the parsing algorithm emphasizes sensitivity.
- If $0 \leq \gamma \leq 1$, the parsing algorithm emphasizes specificity.

In addition, if the user specifies any value of $\gamma < 0$, then CONTRAlign tries trade-off parameters of 2^k for $k \in \{-5, -4, \ldots, 10\}$, and generates one output file for each trade-off parameter. Note that this must be used in conjunction with either `--mfa`, `--clustalw`, or `--posteriors` in order to allow for writing to output files.

For example, the command

```
$ ./contralign predict seq.mfa --gamma 100000
```

runs the maximum expected accuracy placing almost all emphasis on sensitivity (predict correct matches).

The naming convention used by CONTRAlign when $\gamma < 0$ follows somewhat different conventions from normal. Running

```
$ ./contralign predict seq.mfa --gamma -1 \
       --clustalw output
```

will create the files

```
output/output.gamma=0.031250
output/output.gamma=0.062500
...
output/output.gamma=1024.000000
```

For multiple input files,

```
$ ./contralign predict seq1.mfa seq2.mfa \
       --gamma -1 --clustalw output
```

will generate

```
output/output.gamma=0.031250/seq1.mfa
output/output.gamma=0.031250/seq2.mfa
...
output/output.gamma=1024.000000/seq1.mfa
output/output.gamma=1024.000000/seq2.mfa.
```

Like before, multiple types of output (MFA, CLUSTALW, posteriors) may be requested simultaneously.

`--viterbi`

This option uses the Viterbi algorithm to compute alignments rather than the maximum expected accuracy (posterior decoding) algorithm. The alignments generated by the Viterbi option tend to be of slightly lower accuracy than posterior decoding, so this option is not enabled by default. Also, this option is only available when performing pairwise alignments.

`--pairwise`

This option instructs CONTRAlign to perform all pairwise alignments rather than multiple alignments.

`--annealing`
This option replaces the standard progressive alignment dynamic programming algorithm for alignment with an implementation of the sequence annealing algorithm described in:

> Schwartz, A.S., and Pachter, L. (2007) Multiple alignment by sequence annealing. *Bioinformatics*, 23(2): e24-e29.

`--pc_iters NUM`
This option sets the number of iterations of probabilistic consistency to use before running the multiple alignment algorithm.

`--sc_iters NUM`
This option sets the number of iterations of spectral consistency to use before running the multiple alignment algorithm.

`--params PARAMSFILE`

This option uses a trained CONTRAlign parameter file instead of the default program parameters. The format of the parameter file should be the same as the `contralign.params.protein` file in the CONTRAlign source code; each line contains a single parameter name and a parameter value.

`--version`

Display the program version number.

`--verbose`

Show detailed console output.

`--partition`

Compute the log partition function for the input sequence.

5.2 Training mode

In training mode, CONTRAlign infers a parameter set using RNA sequences with known (or partially known) secondary structures in CLUSTALW format. By default, CONTRAlign uses the L-BFGS algorithm for optimization.

For example, suppose `input/*.clustalw` refers to a collection of 100 files which represent sequences with known structures. Calling

```
$ ./contralign train input/*.clustalw
```

instructs CONTRAlign to learn parameters for predict all structures in

```
input/*.clustalw
```

without using any regularization. The learned parameters after each iteration of the optimization algorithm are stored in

```
optimize.params.iter1
optimize.params.iter2
...
```

in the current directory. The final parameters are stored in

```
optimize.params.final
```

and a log file describing the optimization is stored in

```
optimize.log
```

In general, running CONTRAlign without regularization is almost always a bad idea because of overfitting. There are currently two ways to use regularization that are supported in the CONTRAlign program:

1. Regularization may be *manually specified*. The current build of CONTRAlign uses 15 regularization hyperparameters, each of which is used for some subset of the parameters. To specify a single value shared between all of the regularization hyperparameters manually, one can use the `--regularize` flag. For example,

   ```
   $ ./contralign train --regularize 1 \
         input/*.clustalw
   ```

   uses a regularization constant of 1 for each hyperparameter. In general, we recommend that you do not perform training yourself unless you know what you are doing; also do not hesitate to ask us.

2. The recommended usage is to use CONTRAlign's holdout cross-validation procedure to *automatically select* regularization constants. To reserve a fraction p of the training data as a holdout set, run CONTRAlign with the `--holdout` p flag.

   For example, to reserve $1/4^{\text{th}}$ of the training set for holdout cross-validation, use

   ```
   $ ./contralign train --holdout 0.25 \
         input/*.clustalw
   ```

   Note that the `--holdout` and `--regularize` flags should not be used simultaneously.

6 Citing CONTRAlign

If you use CONTRAlign in your work, please cite:

> Do, C.B., Gross, S.S., and Batzoglou, S. (2006) CONTRAlign: Discriminative training for protein sequence alignment. In *Proceedings of the 10th Annual International Conference on Computational Molecular Biology (RECOMB)*, 160-174.

Other relevant references include:

> Do, C.B., Foo, C.-S., Ng, A.Y. (2007) Efficient multiple hyperparameter learning for log-linear models. In *Advances in Neural Information Processing Systems* 20.