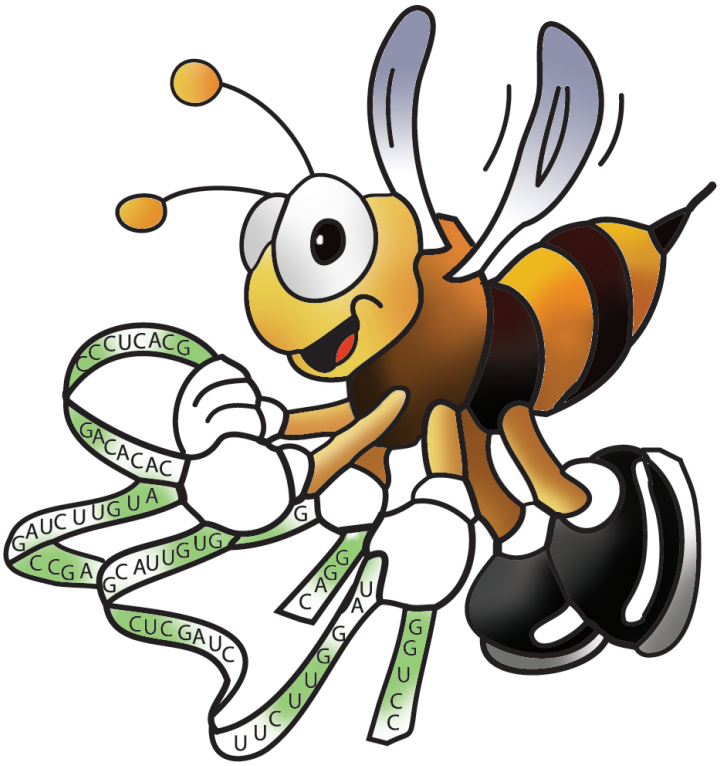

CONTRAfold 2.00

User Manual

Contents

1 Description

CONTRAfold is a novel algorithm for the prediction of RNA secondary structure based on conditional log-linear models (CLLMs). Unlike previous secondary structure prediction programs, CONTRAfold is the first fully probabilistic algorithm to achieve state-of-the-art accuracy in RNA secondary structure prediction.

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

http://contra.stanford.edu/contrafold/

under the BSD license. The CONTRAfold 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, CONTRAfold is only available for Unix-based systems (e.g., Linux). We will be porting CONTRAfold to other architectures and making the binaries available.

3.1 *nix installation

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

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

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

2. Decompress the archive:

   ```
   $ tar zxvf contrafold_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 `contrafold` inside of the current directory.

3. Change to the `contrafold/src` subdirectory and compile the program.

   ```
   $ cd contrafold/src
   $ make clean
   $ make
   ```

Now, your installation is complete!

4 Supported file formats

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

4.1 Input formats

An input file for the CONTRAfold program consists of a single RNA sequence to be folded. The input file must be in either FASTA, plain-text, or BPSEQ format. We describe each of these formats in detail.

4.1.1 FASTA format

A FASTA format file consists of:

1. A single header line containing the character '>' followed by a text description of the RNA sequence. Note that the description must fit on the same line as the '>' character.
2. One or more lines containing RNA sequence data. Each of these lines may contain the letters 'A', 'C', 'G', 'T', 'U' or 'N' in either upper or lower case (the output of the program will retain the case of the input). Any T's are automatically converted to U's. Any other letters are automatically converted to N's. All whitespace (space, tab, newline) is ignored. N's are treated as masked sequence positions which are ignored during all calculations. Other non-whitespace characters are not permitted.

For example, the following is a valid FASTA file:

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

But the following is not (starts with the wrong header character):

```
# sequence
ATGACGGT
```

4.1.2 Plain text format

A plain text format file consists of one or more lines containing RNA sequence data. Each of these lines may contain the letters 'A', 'C', 'G', 'T', 'U', or 'N' in either upper or lower case (the output of the program will retain the case of the input). Any T's are automatically converted to U's. Any other letters are automatically converted to N's. All whitespace (space, tab, newline) is ignored. N's are treated as masked sequence positions which are ignored during all calculations. Other non-whitespace characters are not permitted.

For example, the following is a valid plain text file:

```
NACGACAGUGUAUCACUAGUAcuuA
GUAUGUACUAUC

AGUAGUUGUUGUAGUUC
```

Note that the blank third line will be ignored, and the initial 'N' character will be treated as a placeholder character which appears in the output folded RNA but makes no contribution to the computations.

4.1.3 BPSEQ format

A BPSEQ format file contains exactly one line for each nucleotide in an RNA sequence. The ith line of the file contains three things separated by single spaces:

1. The integer i (with $i = 1$ representing the first nucleotide).
2. The ith character of the RNA sequence (which may be 'A', 'C', 'G', 'T', 'U', or 'N' in either upper or lower case; the output of the program will retain the case of the input; any T's are automatically converted to U's; any other letters are automatically converted to N's).
3. The index of the character to which the ith character base pairs, if known. If the character is known to be unpaired, then 0 appears here. If it is unknown whether this character base-pairs, then a -1 appears here.

Note that the BPSEQ format allows one to prespecify portions of an RNA secondary structure to CONTRAfold. If the `--constraints` flag is enabled, then these prespecified portions will be retained in the CONTRAfold output (see Section 5); otherwise, they are ignored.

For example, the following is a BPSEQ format file:

```
1 A 7
2 G -1
3 U -1
4 C 0
5 c -1
6 c -1
7 u 1
```

in which it is known that the first and last positions base pair, and the middle position does not base pair. However, the folding of the other positions is unknown.

However, the following is not a valid BPSEQ format file:

```
2 G -1
3 U -1
1 A 7
4 C 0
5 C -1
```

```
6 C -1
7 U 1
```

since all nucleotides in the file must appear in order.

4.2 Output formats

The results of a CONTRAfold secondary structure prediction are given in either BPSEQ, parenthesized, or posteriors format. We describe each of these in detail.

4.2.1 BPSEQ format

The BPSEQ output format is identical to the BPSEQ input format (see Section 4.1.3). Since CONTRAfold provides predictions for the pairing or non-pairing of every single nucleotide, no -1's will appear in the output.

4.2.2 Parenthesized format

A parenthesized output file consists of exactly three lines:

- The first line contains the character '>' followed by a short header string describing the folded sequence. If a FASTA input file is used, then the header from the FASTA file is used here. Otherwise, the string 'unnamed' will appear as the header.
- The second line contains the RNA sequence, with all capitalization from the original sequence preserved. If the original sequence contained T's, then they will appear as U's here. Letters other than A, C, G, U, or T will appear as N's here.
- The third line contains a sequence of '(', ')', and '.' characters, one for each nucleotide in the RNA sequence. A '.' character indicates that the corresponding nucleotide is unpaired. A nucleotide annotated with '(' pairs with the nucleotide annoted with the matching ')'. Since CONTRAfold generates only non-pseudoknotted structure predictions, the proper pairing will always be unambiguous.

For example, the following parenthesized structure is a completion of the valid BPSEQ file from Section 4.1.3.

```
>unnamed
AGUCccu
((...))
```

4.2.3 Posteriors format

The posteriors output format is distinct from the BPSEQ and parenthesized formats in that it does *not* provide a single prediction of RNA secondary structure. Instead, it provides a sparse representation of the base pairing posterior probabilities for pairs of letters in the RNA sequence. Specifically, the ith line contains

1. The integer i.
2. The ith character of the file.
3. A space-separated list of base-pairing probabilities of the form j:p_{ij}, where $j > i$ is the index of nucleotide to which the ith nucleotide might pair, and p_{ij} is the probability that this base pairing occurs.

For example, the following is a posteriors format output:

```
1 A 7:0.035 9:0.10
2 G 6:0.036 8:0.11
3 U
4 C
5 C
6 C
7 U
8 C
9 A
```

In the above, we see that nucleotide 2 has an 11% probability of pairing to nucleotide 8. Note that each pairing probability is reported only once (i.e., on the ith line, we show only the pairing probabilities to nucleotides $j > i$ which appear *after* the ith position in the RNA sequence).

5 Usage

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

- In "prediction" mode, CONTRAfold folds new RNA sequences using either the default parameters or a CONTRAfold-format parameter file.
- In "training" mode, CONTRAfold learns new parameters from training data consisting of pre-folded RNA sequences in BPSEQ format.

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

5.1 Prediction mode

In prediction mode, CONTRAfold predicts the secondary structure of one or more unfolded input RNA sequence, and prints the result to either the console or output files. The basic syntax for running CONTRAfold in prediction mode is

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

5.1.1 A single input file

For single sequence prediction, CONTRAfold generates parenthesized output (see Section 4.2.2) to the console (i.e., stdout) by default.

For example, suppose the file "seq1.fasta" contains a FASTA formatted sequence to be folded. Then the command

```
$ ./contrafold predict seq.fasta
```

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

CONTRAfold can also write parenthesized, BPSEQ, or posteriors formatted output to an output file. To write parenthesized output to a file,

```
$ ./contrafold predict seq.fasta --parens seq.parens
```

To write BPSEQ output to a file,

```
$ ./contrafold predict seq.fasta --bpseq seq.bpseq
```

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

```
$ ./contrafold predict seq.fasta --posteriors \
       0.001 seq.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

```
$ ./contrafold predict seq.fasta --parens \
       seq.parens --bpseq seq.bpseq --posteriors \
       0.001 seq.posteriors
```

will generate three different output files simultaneously.

5.1.2 Multiple input files

For multiple input files, CONTRAfold generates parenthesized output (see Section 4.2.2) 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).

CONTRAfold can also write parenthesized, BPSEQ, or posteriors formatted output to several output files. In particular, CONTRAfold 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.fasta" and "seq2.fasta" each contain a FASTA formatted sequence to be folded. Then the command

```
$ ./contrafold predict seq1.fasta seq2.fasta \
       --parens output
```

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

Alternatively,

```
$ ./contrafold predict seq1.fasta seq2.fasta \
       --bpseq output
```

and

```
$ ./contrafold predict seq1.fasta seq2.fasta \
       --posteriors 0.001 output
```

generate BPSEQ and posteriors formatted outputs instead.

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

```
$ ./contrafold predict seq1.fasta seq2.fasta --parens \
       parens_output --bpseq bpseq_output --posteriors \
       0.001 posteriors_output
```

generates three different output subdirectories (parens_output, bpseq_output, and posteriors_output) each containing two files (seq1.fasta, seq2.fasta).

5.1.3 Optional arguments

CONTRAfold 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 CONTRAfold program on the command line. For example,

```
$ ./contrafold predict seq.fasta --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 which is correctly base paired gets a score of γ, and each nucleotide which is correctly not base paired gets a score of 1. Then, CONTRAfold finds the folding of the input sequence 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 CONTRAfold 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 `--parens`, `--bpseq`, or `--posteriors` in order to allow for writing to output files.

For example, the command

```
$ ./contrafold predict seq.fasta --gamma 100000
```

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

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

```
$ ./contrafold predict seq.fasta --gamma -1 \
       --bpseq output
```

will create the files

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

For multiple input files,

```
$ ./contrafold predict seq1.fasta seq2.fasta \
       --gamma -1 --bpseq output
```

will generate

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

Like before, multiple types of output (parens, BPSEQ, posteriors) may be requested simultaneously.

`--viterbi`

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

`--noncomplementary`

This option uses a folding model that allows non `AU/CG/GU` pairings in the CONTRAfold output. This option is slower and generally slightly less accurate than the default option of allowing only "canonical" base-pairings.

`--constraints`

This option requires the use of BPSEQ format input files. By default, any base pairings that are included in the BPSEQ file above are ignored. However, if the `--constraints` flag is used, then any base pairings in an input BPSEQ file are treated as constraints on the allowed structures. In particular,

1. A nucleotide mapping to a positive index i is constrained to base-pair with nucleotide i.

2. A nucleotide mapping to 0 is constrained to be unpaired.
3. A nucleotide mapping to -1 is unconstrained.

For example, given the following input BPSEQ file:

```
1 A -1
2 C -1
3 G -1
4 U 7
5 U 0
6 C 0
7 G 4
8 C -1
9 G -1
10 U -1
```

and the `--constraints` flag, then CONTRAfold will assume that positions 4 and 7 are constrained to be base-pairing, while positions 5 and 6 are constrained to be unpaired. The base-pairing of the remaining positions is decided by CONTRAfold.

`--params PARAMSFILE`

This option uses a trained CONTRAfold parameter file instead of the default program parameters. The format of the parameter file should be the same as the `default.params` file in the CONTRAfold 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. This option may be used in conjunction with the `--constraints` option in order to determine the CONTRAfold "energy" of a given RNA secondary structure specified in a BPSEQ file. For example, to compute the energy of a Viterbi parse generated via

```
$ ./contrafold predict seq.fasta --viterbi \
      --bpseq seq.bpseq
```

we can simply run

```
$ ./contrafold predict seq.bpseq --constraints \
      --partition
```

Some quick notes regarding the partition function:

- When used in conjunction with partial constraints (i.e., only some of the mappings in the input BPSEQ file are -1's; see above), then this option computes the log of the summed unnormalized probabilities for all structures consistent with the partial constraints.
- In order to compute the log of the summed *probabilities* (which are normalized as opposed to the quantities mentioned above), you must also run

  ```
  $ ./contrafold predict seq.bpseq --partition
  ```

  and subtract this log partition value from the previous log partition value described above. Note that this quantity will always be greater than or equal to the log-partition above, implying that the log of the summed probabilities is necessarily non-positive (which makes sense as probabilities are at most 1).

5.2 Training mode

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

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

```
$ ./contrafold train input/*.bpseq
```

instructs CONTRAfold to learn parameters for predict all structures in

```
input/*.bpseq
```

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

1. Regularization may be *manually specified*. The current build of CONTRAfold uses 15 regularization hyperparameters, each of which is used for some subset of the parameters. To specify regularization manually, it is necessary to specify the regularization constant for each of the 15 hyperparameters. For example,

   ```
   $ ./contrafold train --regularize 1 1 1 1 1 \
          1 1 1 1 1 1 1 1 1 1 input/*.bpseq
   ```

   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 CONTRAfold's holdout cross-validation procedure to *automatically select* regularization constants. To reserve a fraction p of the training data as a holdout set, run CONTRAfold with the `--holdout` p flag.

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

   ```
   $ ./contrafold train --holdout 0.25 \
          input/*.bpseq
   ```

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

5.2.1 Optional arguments

In training mode, CONTRAfold accepts the following optional arguments:

`--params PARAMSFILE`

This option uses an existing CONTRAfold parameter file instead of the default initial parameters. The format of the parameter file should be the same as the `Default.params` file in the CONTRAfold source code; each line contains a single parameter name and a parameter value.

6 Visualization of folded RNAs

Besides the main program, the CONTRAfold package contains some additional tools for visualization of folded RNAs:

- `make_coords`: generates a set of coordinates for plotting a CONTRAfold BPSEQ file.
- `plot_rna`: converts a set of coordinates and a BPSEQ file into a viewable PNG.

In the following subsections, we describe the installation and use of these two tools for RNA visualization.

6.1 Installation

Currently, only UNIX installation is supported.

6.1.1 *nix installation

To compile CONTRAfold visualization tools from the source code (for a *nix machine):

1. Install the `libgd` graphics development library, available from

   http://www.boutell.com/gd/

2. Install the `libpng` PNG image library, available from

   http://www.libpng.org/pub/png/libpng.html

3. Compile the visualization tools:

```
$ make viz
```

6.2 Usage

Given an input FASTA file, generating an image of the predicted CONTRAfold structure involves three steps:

1. Generate a secondary structure prediction in BPSEQ format:

```
$ ./contrafold predict seq.fasta --bpseq \
      seq.bpseq
```

2. Run the `make_coords` program to generate an RNA layout:

```
$ ./make_coords output.bpseq output.coords
```

The resulting coordinates are placed in the `output.coords` file.

3. Run the `plot_rna` program to convert the layout into a PNG image:

```
$ ./plot_rna output.bpseq output.coords \
        --png output.png
```

The resulting PNG is placed in the `output.png` file and can be viewed with a web browser such as Mozilla Firefox. Alternatively, EPS format output is also available:

```
$ ./plot_rna output.bpseq output.coords \
        --eps output.eps
```

6.3 Additional options

The `plot_rna` has a couple of options which you can use to control the generated PNG files:

`--posteriors` *posteriorsfile*

If a CONTRAfold posteriors file is also available,then using the above option will generate a PNG file in which the letters of each RNA nucleotide is colored according to posterior probability confidence. Black letters indicate high confidence structure whereas lighter gray letters indicate lower confidence structure.

`--title "`*title*`"`

This option allows the user to annotate the generated RNA image with a title. Note that the title string should be surrounded with double quotation marks so as to ensure that it is interpreted as a single argument to the program.

In general, the CONTRAfold visualization tools generate RNA layouts which tend to be visually pleasing. The layout algorithm uses a simple deterministic layout rule, followed by a gradient-based optimization procedure. This type of procedure is not guaranteed to generate non-overlapping layouts for all RNA structures; in practice, however the visualization tools can provide reasonable visualizations for a large range of RNA structures.

7 Citing CONTRAfold

If you use CONTRAfold in your work, please cite:

> Do, C.B., Woods, D.A., and Batzoglou, S. (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14): e90-e98.