Supplementary Material for

CONTRAfold: RNA Secondary Structure Prediction without Physics-Based Models

Chuong B. Do, Daniel A. Woods, and Serafim Batzoglou

Stanford University, Stanford, CA 94305, USA,
{chuongdo,danwoods,serafim}@cs.stanford.edu,
WWW home page: http://contra.stanford.edu/contrafold/

Abstract

In this supplementary material, we describe in full the structured conditional log-linear model (structured CLLM) used in the CONTRAfold program. We also provide detailed pseudocode explicitly showing the dynamic programming recurrences needed to reproduce the CONTRAfold algorithm, specifically CONTRAfold version 1.10.

1 Preliminaries

Let $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}, \texttt{N}\}$ be an alphabet, and consider some string $x \in \Sigma^L$ of length L. In the RNA secondary structure prediction problem, x represents an unfolded RNA string, and x_i refers to the ith character of x, for $i = 1, \ldots, L$. For ease of notation, we say that there are $L+1$ *positions* corresponding to x—one position at each of the two ends of x, and $L-1$ positions between consecutive nucleotides of x. We will assign indices ranging from 0 to L for each position. This is illustrated in Figure 1.

Let $\mathcal{Y}$ be the space of all possible structures of a sequence x. Structured conditional log-linear models (structured CLLMs) define the conditional probability of a structure $y \in \mathcal{Y}$ given an input RNA sequence x as

$$P(y \mid x; \boldsymbol{w}) = \frac{\exp(\boldsymbol{w}^T \mathbf{F}(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}^T \mathbf{F}(x,y'))} \tag{1}$$

$$= \frac{1}{Z(x)} \cdot \exp(\boldsymbol{w}^T \mathbf{F}(x,y)) \tag{2}$$

where $\mathbf{F}(x,y) \in \mathbb{R}^n$ is an n-dimensional vector of feature counts describing x and y, $\boldsymbol{w} \in \mathbb{R}^n$ is an n-dimensional vector of parameters, and $Z(x)$ (known as the *partition function* of a sequence x) is a normalization constant ensuring that $P(y \mid x; \boldsymbol{w})$ forms a legal probability distribution over the space of possible structures $\mathcal{Y}$. In this representation, the "weight" associated with a structure y for a sequence x is $\exp(\boldsymbol{w}^T \mathbf{F}(x,y))$. Because the *logarithm* of the weight is a *linear* function of the features $\mathbf{F}(x,y)$, this is typically known as the *log-linear* representation of a CRF.

Now, consider the following reparameterization of (2). For each entry w_i of $\boldsymbol{w}$, define $\phi_i = \exp(w_i)$. It follows that (2) may be rewritten as

$$P(y \mid x; \boldsymbol{w}) = \frac{1}{Z(x)} \cdot \prod_{i=1}^{n} \phi_i^{F_i(x,y)} \tag{3}$$

where $F_i(x,y)$ is the ith component of $\mathbf{F}(x,y)$. In this alternative representation, the weight associated with a structure y for a sequence x is a product, $\prod_{i=1}^n \phi_i^{F_i(x,y)}$. We refer to this as the *potential* representation of a CRF, where each parameter ϕ_i is called a *potential*.

In Figure 2, we list all of the potentials $\{\phi_i\}$ involved in scoring a structure y. Then, in Section 2, we define the feature counts $\{F_i(x,y)\}$ for a sequence x and its structure y. Finally, in the remaining sections, we describe the dynamic programming recurrences needed to perform inference using our probabilistic model.

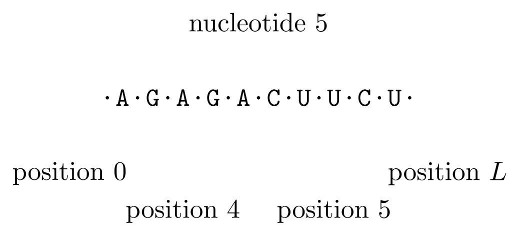

Figure 1: Positions in a sequence of length $L = 10$.

$\phi_{\text{hairpin base}}$	$\phi_{\text{hairpin length}}[\cdot]$	$\phi_{\text{helix base pair}}(\cdot,\cdot)$
$\phi_{\text{hairpin extend}}$	$\phi_{\text{helix change}}[\cdot]$	$\phi_{\text{helix closing}}(\cdot,\cdot)$
$\phi_{\text{helix extend}}$	$\phi_{\text{bulge length}}[\cdot]$	$\phi_{\text{single base pair stacking left}}((\cdot,\cdot),\cdot)$
$\phi_{\text{multi base}}$	$\phi_{\text{internal length}}[\cdot]$	$\phi_{\text{single base pair stacking right}}((\cdot,\cdot),\cdot)$
$\phi_{\text{multi unpaired}}$	$\phi_{\text{internal asymmetry}}[\cdot]$	$\phi_{\text{terminal mismatch}}((\cdot,\cdot),\cdot,\cdot)$
$\phi_{\text{multi paired}}$	$\phi_{\text{internal full}}[\cdot][\cdot]$	$\phi_{\text{helix stacking}}((\cdot,\cdot),(\cdot,\cdot))$

Figure 2: List of all potentials used in the CONTRAfold model.

2 Basic feature set

In this section, we define the feature counts $\{F_i(x,y)\}$ for a sequence x and a structure y. One way to do this is to give, for each potential ϕ_i shown in Figure 2, a formula *explicitly* specifying how to compute the corresponding feature $F_i(x,y)$.

Here, we will instead define feature counts *implicitly* by

1. decomposing a secondary structure y into four fundamental types of substructures: hairpins, single-branched loops, helices, and multi-branched loops;
2. defining a *factor*[1] for each type of substructure as a product of potentials from Figure 2;
3. defining the product $\prod_{i=1}^{n} \phi_i^{F_i(x,y)}$ as a product of factors for each substructure in y.

By specifying which potentials are included in the computation of the factor for each type of substructure, we thus define the feature counts $\{F_i(x,y)\}$ implicitly as the *number of times each potential ϕ_i is used in the product of factors for a structure y.*

2.1 Hairpins

A hairpin is a loop with only one adjacent base pair, known as its *closing base pair.* For $1 \le i \le j < L$, we say that a hairpin spans positions i to j if x_i and x_{j+1} form the closing base pair (see Figure 3). For hairpins, energy-based secondary structure folding algorithms such as Mfold assign free energy increments for each of the following:

- energies corresponding to the length of the loop (i.e., a hairpin spanning positions i to j has length $j - i$),
- terminal mismatch stacking energies as a function of the closing base pair (x_i, x_{j+1}) and the first unpaired nucleotides in the loop, x_{i+1} and x_j,
- bonus free energies for loops containing specific nucleotide sequences, and
- other special cases.

[1]To be clear, a *factor* is simply a collection of potentials that are associated with the presence of a particular secondary structure subunit in a structure y. For example, the factor associated with a hairpin loop is simply the product of the parameter potentials which are involved in "scoring" the hairpin loop.

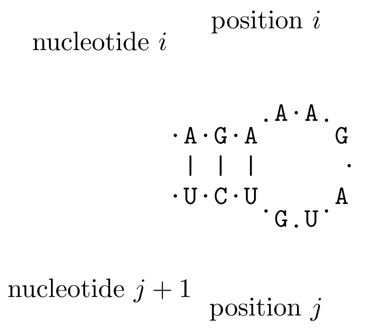

Figure 3: A hairpin loop of length 6 spanning positions i to j.

CONTRAfold uses a simplified scoring model for hairpins which ignores the latter two cases. In particular, the factor $\varphi_{\text{hairpin}}(i,j)$ of a hairpin spanning positions i to j is

$$\begin{aligned}&\varphi_{\text{hairpin}}(i,j) = \\ &\quad \phi_{\text{terminal mismatch}}((x_i,x_{j+1}),x_{i+1},x_j) \\ &\quad \cdot \begin{cases}\phi_{\text{hairpin length}}[j-i] & \text{if } 0 \le j-i \le 30 \\ \phi_{\text{hairpin base}} \cdot (\phi_{\text{hairpin extend}})^{\ln(j-i)} & \text{if } j-i > 30.\end{cases}\end{aligned} \tag{4}$$

In the above expression, the first term accounts for terminal mismatches arising from the fact that (x_i, x_{j+1}) are paired, but x_{i+1} and x_j are not.[2] The second term scores the hairpin based on its length. For loops under size 30, potentials are read directly from a table. For longer loops, the factor above directly imitates typical energy-based scoring schemes, which estimate the free energy increment of a loop of length $j-i$ as

$$a + b \cdot \ln(j-i), \tag{5}$$

for fixed constants a and b. By analogy, we have

$$\begin{aligned}&\phi_{\text{hairpin base}} \cdot (\phi_{\text{hairpin extend}})^{\ln(j-i)} \\ &\qquad = \exp(\ln(\phi_{\text{hairpin base}}) + \ln(\phi_{\text{hairpin extend}}) \cdot \ln(j-i)) && (6) \\ &\qquad = \exp(a' + b' \cdot \ln(j-i)) && (7)\end{aligned}$$

where

$$\begin{aligned}a' &= \ln(\phi_{\text{hairpin base}}) && (8) \\ b' &= \ln(\phi_{\text{hairpin extend}}). && (9)\end{aligned}$$

2.2 Single-branched loops

A single-branched loop is a loop which has two adjacent base pairs. The outermost base pair is called the *external closing base pair* whereas the innermost base pair is called the *internal closing base pair*. Suppose $1 \le i \le i'$ and $j' \le j < L$. We say that a single-branched loop spans positions i to i' and j' to j if x_i and x_{j+1} form the external closing base pair and $x_{i'+1}$ and $x_{j'}$ form the internal closing base pair. To ensure that the internal closing base pair is well-defined, we require that $i'+2 \le j'$ (see Figure 4).

A single-branched loop for which $i' = i$ and $j = j'$ is called a *stacking pair*. A single-branched loop for which either $i' = i$ or $j = j'$ (but not both) is called a *bulge*. Finally, a single-branched loop for which both

[2]Here, note that the order of the arguments is important so as to ensure that the parameters are invariant with respect to the orientation of the substructure. For example, we expect the parameter for AG stacking on top of CU to be the same as the parameter for UC stacking on top of GA.

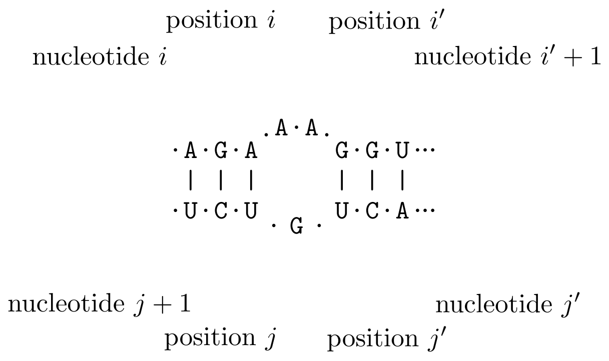

Figure 4: A single-branched (internal) loop of lengths 2 and 1 spanning positions i to i' and j' to j. Here, A-U is the external closing base pair and G-U is the internal closing base pair.

$i' > i$ and $j > j'$ is called an $\ell_1 \times \ell_2$ *internal loop*, where $\ell_1 = i' - i$ and $\ell_2 = j - j'$. For now, we will treat the problem of only scoring bulges and internal loops; we consider the scoring of stacking pairs separately in the next section.

Energy-based scoring methods typically score internal loops and bulges by accounting for the following:

- energies based on the total loop length, $\ell_1 + \ell_2$,
- energies based on the asymmetry in sizes of each side of the loop, $|\ell_1 - \ell_2|$,
- special corrections for highly asymmetric $1 \times \ell$ (or $\ell \times 1$) loops
- terminal mismatch stacking energies for the external closing base pair (x_i, x_{j+1}) and its adjacent nucleotides in the loop, x_{i+1} and x_j,
- terminal mismatch stacking energies for the internal closing base pair $(x_{j'}, x_{i'+1})$ and its adjacent nucleotides in the loop, $x_{j'+1}$ and $x_{i'}$, and
- specific free energy increments for 1×1, 1×2, and 2×2 interior loops as a function of their closing base pairs and the nucleotides in the loop.

For computational tractability, many programs such as Mfold limit total loop lengths of single-branched loops to a small constant c (typically, $c = 30$).

In CONTRAfold, the total loop length, loop asymmetry, and terminal mismatch stacking interaction terms are retained. The special corrections for asymmetric interior loops are replaced with a more general two-dimensional table for scoring $\ell_1 \times \ell_2$ interior loops. Finally, the large lookup tables which exhaustively characterize the energies of all 1×1, 1×2, and 2×2 interior loops are omitted.

Specifically, for all $1 \leq i \leq i'$ and $j' \leq j \leq L - 1$ such that $i' + 2 \leq j'$ and $1 \leq i' - i + j - j' \leq c$, the factor $\varphi_{\text{single}}(i, j, i', j')$ for a bulge or internal loop is given by

$$\begin{aligned}
&\varphi_{\text{single}}(i, j, i', j') = \\
&\quad \begin{cases}
\phi_{\text{bulge length}}[i' - i + j - j'] & \text{if } i' - i = 0 \text{ or } j - j' = 0 \\
\phi_{\text{internal length}}[i' - i + j - j'] & \text{if } i' > i \text{ and } j > j' \\
\qquad \cdot \phi_{\text{internal asymmetry}}[|(i' - i) - (j - j')|] & \\
\qquad \cdot \phi_{\text{internal full}}[i' - i][j - j'] &
\end{cases} \\
&\quad \cdot \phi_{\text{terminal mismatch}}((x_i, x_{j+1}), x_{i+1}, x_j) \\
&\quad \cdot \phi_{\text{terminal mismatch}}((x_{j'}, x_{i'+1}), x_{j'+1}, x_{i'}).
\end{aligned} \tag{10}$$

Like most energy-based methods, we use $c = 30$ for computational tractability.

2.3 Helices

A single-branched loop for which $i' = i$ and $j = j'$ is known as a *stacking pair*. A sequence of one or more consecutive stacking pairs is called a *helix* (or stem); informally then, a helix consists of several consecutive

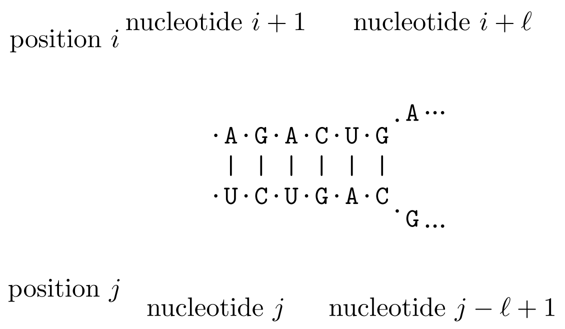

Figure 5: A helix of length $\ell = 6$ spanning positions i to j.

nucleotides of an RNA molecule directly base pairing to a set of consecutive nucleotides which appear later in the RNA sequence.

Now, consider a helix that matches nucleotides $x_{i+1}x_{i+2}\dots x_{i+\ell}$ in a sequence x to nucleotides $x_{j-\ell+1}x_{j-\ell+2}\dots x_j$ which appear later in the sequence. We say that this is a helix of length ℓ starting at positions i and j. Nucleotides x_{i+1} and x_j form the *external closing base pair* of the helix whereas nucleotides $x_{i+\ell}$ and $x_{j-\ell+1}$ form the *internal closing base pair* (see Figure 5).

Traditional energy-based methods such as Mfold score helices using

- a sum of interaction terms for each stacking pair, and
- penalties for each non-GC terminal closing base pair.

Since stacking pair interaction terms are based on the nearest neighbor model, only Watson-Crick and wobble `GU` base pairs are allowed; other pairings are necessarily treated as small symmetric interior loops.

CONTRAfold extends on traditional energy-based methods by including penalties for all possible closing base pairs (not just the "canonical" pairings). CONTRAfold also considers the interaction of every pair of bases in the stem rather than ignoring the non-canonical/non-`GU` base pairs which are not found in the regular nearest neighbor energy rules. Finally, CONTRAfold includes scores for helix lengths, allowing arbitrary scores for helix lengths of at most d (in practice, we set $d = 5$), and assigning affine scores for helices of length greater than d.

In particular, for $0 \le i \le i + 2\ell + 2 \le j \le L$, the factor $\varphi_{\text{helix}}(i, j, \ell)$ for a helix of length ℓ starting at i and j is:

$$\begin{aligned}
&\varphi_{\text{helix}}(i,j,\ell) = \\
&\quad \phi_{\text{helix closing}}(x_{i+1}, x_j) \\
&\quad \cdot \phi_{\text{helix closing}}(x_{j-\ell+1}, x_{i+\ell}) \\
&\quad \cdot \prod_{k=1}^{\ell} \phi_{\text{helix base pair}}(x_{i+k}, x_{j-k+1}) \\
&\quad \cdot \prod_{k=1}^{\ell-1} \phi_{\text{helix stacking}}((x_{i+k}, x_{j-k+1}), (x_{i+k+1}, x_{j-k})) \\
&\quad \cdot \varphi_{\text{helix length}}(\ell),
\end{aligned} \tag{11}$$

where

$$\varphi_{\text{helix length}}(\ell) = \left(\prod_{i=1}^{\min(d,\ell)} \phi_{\text{helix change}}[i] \right) \cdot (\phi_{\text{helix extend}})^{\max(\ell-d,0)}. \tag{12}$$

In this formulation, $\phi_{\text{helix closing}}(x_i, x_j)$ scores the use of a particular base pair for closing a helix. Similarly, $\phi_{\text{helix stacking}}((x_i, x_j), (x_{i+1}, x_{j-1}))$ scores the interaction for stacking (x_{i+1}, x_{j-1}) on top of (x_i, x_j). Finally,

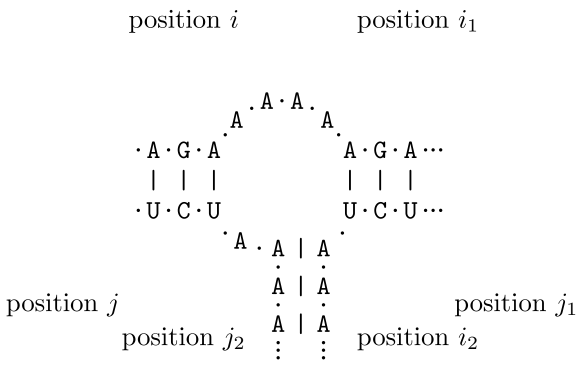

Figure 6: A multi-branched loop spanning positions i to i_1, j_1 to i_2, and j_2 to j.

the helix length score $\varphi_{\text{helix length}}(\ell)$ is designed so that the length component of the score for any helix of length $\ell \leq d$ is given explicitly as

$$(\phi_{\text{helix change}}[1]) \cdot (\phi_{\text{helix change}}[2]) \cdot \dots \cdot (\phi_{\text{helix change}}[\ell]), \tag{13}$$

and helices of length $\ell > d$ have a correction potential of $\phi_{\text{helix extend}}$ applied for each additional base pair.

2.4 Multi-branched loops

A multi-branched loop is a loop containing at least three adjacent base pairs. More formally, suppose $i \leq i_1 \leq j_1 \leq i_2 \leq j_2 \leq \ldots \leq i_m \leq j_m \leq j$ where $m \geq 2$ and $i_k + 2 \leq j_k$ for $k = 1, \ldots, m$. We say that a multibranch loop spans positions i to i_1, j_1 to i_2, …, and j_m to j if nucleotides (x_i, x_{j+1}) form the external closing base pair, and (x_{j_k}, x_{i_k+1}) form the internal closing base pairs for $k = 1, \ldots, m$ (see Figure 6).

Let the length ℓ of a multi-branched loop be the number of unpaired bases,

$$\ell = i_1 - i + j - j_m + \sum_{k=2}^{m} (i_k - j_{k-1}). \tag{14}$$

For computational tractability, most programs score multi-branched loops using

- energy terms dependent on the length of the loop.
- single base pair stacking energies describing the attachment of each helix to the multi-branched loop,
- coaxial stacking terms for helices on the multi-branched loop that are separated by at most one unpaired position

CONTRAfold uses a similar scoring scheme for multi-branched loops which ignores coaxial stacking. Specifically, if $1 \leq i \leq i_1 \leq i_1 + 2 \leq j_1 \leq i_2 \leq \ldots \leq j \leq L - 1$, then the factor associated with a multi-branched loop spanning positions i to i_1, j_1 to i_2, …, and j_m to j is

$$\begin{aligned} &\varphi_{\text{multi}}(i, j, i_1, j_1, \ldots, i_m, j_m) = \\ &\quad \phi_{\text{multi base}} \cdot (\phi_{\text{multi unpaired}})^{\ell} \cdot (\phi_{\text{multi paired}})^{m+1} \\ &\quad \cdot \varphi_{\text{multi mismatch}}((x_i, x_{j+1}), x_{i+1}, x_j) \\ &\quad \cdot \prod_{k=1}^{m} \varphi_{\text{multi mismatch}}((x_{j_k}, x_{i_k+1}), x_{j_k+1}, x_{i_k}). \end{aligned} \tag{15}$$

where

$$\begin{aligned} &\varphi_{\text{multi mismatch}}((x_i, x_{j+1}), x_{i+1}, x_j) = \\ &\quad \phi_{\text{single base pair stacking left}}((x_i, x_{j+1}), x_{i+1}) \cdot \phi_{\text{single base pair stacking right}}((x_i, x_{j+1}), x_j) \end{aligned} \tag{16}$$

This mirrors the affine energy models typically used for multi-branched loops in energy-based methods.

3 The Viterbi algorithm

We now specify the Viterbi algorithm for computing the most likely structure via dynamic programming recurrences. Let c be the maximum length of an internal loop or bulge.

3.1 Definitions

We define the following factors:

- $\varphi_{\text{do outer}}(i)$, $0 \leq i \leq L$: the best possible score for folding the substring $x_{i+1}x_{i+2}\cdots x_L$, assuming that the ends of this substring belong to the exterior loop of the RNA.
- $\varphi_{\text{do helix}}(i,j,n)$, $0 \leq i \leq j \leq L$
  - $0 \leq n < d$: the best possible score for folding the substring $x_{i+1}x_{i+2}\cdots x_j$, assuming that exactly n letters on each side of the substring are paired in a helix – i.e., $(x_i, x_{j+1}), (x_{i-1}, x_{j+2}), \ldots, (x_{i-n+1}, x_{j+n})$ all form base pairs, but x_{i-n} and x_{j+n+1} do not base pair.
  - $n = d$: the best possible score for folding the substring $x_{i+1}x_{i+2}\cdots x_j$, assuming that at least d letters on each side of the substring are paired in a helix – i.e., $(x_i, x_{j+1}), (x_{i-1}, x_{j+2}), \ldots, (x_{i-d+1}, x_{j+d})$ all form base pairs, and possibly more.
- $\varphi_{\text{do multi}}(i,j,n)$, $0 \leq i \leq j \leq L$
  - $0 \leq n < 2$: the best possible score for folding the substring $x_{i+1}x_{i+2}\cdots x_j$, assuming that the substring is part of a multibranch loop that contains exactly contains n adjacent helices besides the exterior helix.
  - $n = 2$: the best possible score for folding the substring $x_{i+1}x_{i+2}\cdots x_j$, assuming that the substring is part of a multibranch loop that contains exactly at least 2 adjacent helices besides the exterior helix.

3.2 Recurrences

For each of the factors described in the previous subsection, we now give the appropriate recurrence along with a description of the cases handled by the recurrence.

3.2.1 Exterior loop

When generating a substring belonging to the exterior loop, there are three cases:

1. the substring is of zero length,
2. the first base of the substring belongs to the exterior loop,
3. the first base belongs to a helix that is adjacent to the exterior loop.

This gives:

$$\varphi_{\text{do outer}}(i) = \max \begin{cases} 1 & \text{if } i = L \\ \phi_{\text{outer unpaired}} \cdot \varphi_{\text{do outer}}(i+1) & \text{if } 0 \leq i < L \\ \max\limits_{\substack{i' \\ i+2 \leq i' \leq L}} \left(\phi_{\text{outer branch}} \cdot \varphi_{\text{do helix}}(i,i',0) \cdot \varphi_{\text{do outer}}(i')\right) & \text{if } 0 \leq i \leq L. \end{cases}$$

Note that in the last case, we require that $i + 2 \leq i'$ so as to ensure that the definition of $\varphi_{\text{do outer}}(i)$ is not circular (actually, it would suffice to require that $i < i'$; however, the requirement we make here works as well since a helix must contain at least two base pairs).

3.2.2 Helix

To generate a helix for the substring $x_{i+1}x_{i+2}\cdots x_j$, there are several cases:

1. no surrounding positions belong to the helix yet and (x_{i+1}, x_j) base pair,
2. n surrounding positions belong to the helix (where $0 < n < d$) and (x_{i+1}, x_j) base pair,
3. at least d surrounding positions belong to the helix and (x_{i+1}, x_j) base pair,
4. at least one surrounding position belongs to the helix and $x_{i+1}x_{i+2}\cdots x_j$ form a hairpin loop,
5. at least one surrounding position belongs to the helix and $x_{i+1}x_{i+2}\cdots x_j$ form the beginning of a single-branched loop,
6. at least one surrounding position belongs to the helix and $x_{i+1}x_{i+2}\cdots x_j$ form the beginning of a multi-branched loop.

This gives:

$$\varphi_{\text{do helix}}(i,j,n) =$$
$$\max\begin{cases} \phi_{\text{helix change}}[1]\cdot\phi_{\text{helix closing}}(x_{i+1},x_j) & \text{if } 0\le i<i+2\le j\le L \text{ and } n=0 \\ \qquad\cdot\,\phi_{\text{helix base pair}}(x_{i+1},x_j)\cdot\varphi_{\text{do helix}}(i+1,j-1,1) & \\ \phi_{\text{helix change}}[n+1]\cdot\phi_{\text{helix stacking}}((x_i,x_{j+1}),(x_{i+1},x_j)) & \text{if } 0<i<i+2\le j<L \text{ and } 0<n<d \\ \qquad\cdot\,\phi_{\text{helix base pair}}(x_{i+1},x_j)\cdot\varphi_{\text{do helix}}(i+1,j-1,n+1) & \\ \phi_{\text{helix extend}}\cdot\phi_{\text{helix stacking}}((x_i,x_{j+1}),(x_{i+1},x_j)) & \text{if } 0<i<i+2\le j<L \text{ and } n=d \\ \qquad\cdot\,\phi_{\text{helix base pair}}(x_{i+1},x_j)\cdot\varphi_{\text{do helix}}(i+1,j-1,d) & \\ \phi_{\text{helix closing}}(x_{j+1},x_i)\cdot\varphi_{\text{do loop}}(i,j) & \text{if } 0<i\le j<L \text{ and } n>0 \end{cases}$$

Here, note that whenever a case depends on (x_i, x_{j+1}), we ensure that $0 < i$ and $j < L$. Also, if a case depends on x_{i+1} and x_j, we ensure that $i + 2 \le j$.

3.2.3 Loop

To generate a loop for the substring $x_{i+1}x_{i+2}\cdots x_j$, there are several cases:

1. $x_{i+1}x_{i+2}\cdots x_j$ form a hairpin loop,
2. $x_{i+1}x_{i+2}\cdots x_j$ form the beginning of a single-branched loop,
3. $x_{i+1}x_{i+2}\cdots x_j$ form the beginning of a multi-branched loop.

This gives:

$$\varphi_{\text{do loop}}(i,j) =$$
$$\max\begin{cases} \varphi_{\text{hairpin}}(i,j) & \text{if } 0<i\le j<L \text{ and } n>0 \\ \max\limits_{\substack{i',j' \\ i\le i'<i'+2\le j'\le j \\ 1\le i'-i+j-j'\le c}} \left(\varphi_{\text{single}}(i,j,i',j')\cdot\varphi_{\text{do helix}}(i',j',0)\right) & \text{if } 0<i\le j<L \text{ and } n>0 \\ \phi_{\text{multi base}}\cdot\phi_{\text{multi paired}} & \text{if } 0<i\le i+2\le j<L \text{ and } n>0. \\ \qquad\cdot\,\varphi_{\text{multi mismatch}}((x_i,x_{j+1}),x_{i+1},x_j)\cdot\varphi_{\text{do multi}}(i,j,0) & \end{cases}$$

Note that in the case of single-branched loops, $i' + 2 \le j'$ since the inner helix must have at least one base pairing, and $1 \le i' - i + j - j' \le c$ to ensure that the loop has length at least 1, but no more than c (for efficiency).

3.2.4 Multi-branched loops

To generate a multi-branched loop for the substring $x_{i+1}x_{i+2}\cdots x_j$, there are several cases:

1. the substring is of length zero and has at least 2 adjacent helices (other than the exterior helix),
2. the first letter of the substring is unpaired,
3. the first letter of the substring belongs to a helix that is adjacent to the multi-branch loop and fewer than 2 adjacent helices (other than the exterior helix) have been generated already.
4. the first letter of the substring belongs to a helix that is adjacent to the multi-branch loop and at least 2 adjacent helices (other than the exterior helix) have been generated already.

From this, we obtain

$$\varphi_{\text{do multi}}(i,j,n) =$$
$$\max\begin{cases} 1 & \text{if } 0 \le i = j \le L \text{ and } n = 2 \\ \phi_{\text{multi unpaired}} \cdot \varphi_{\text{do multi}}(i+1,j,n) & \text{if } 0 \le i < j \le L \text{ and } 0 \le n \le 2 \\ \max\limits_{\substack{j' \\ i+2 \le j' \le j}} \begin{pmatrix} \phi_{\text{multi paired}} \cdot \varphi_{\text{multi mismatch}}((x_{j'}, x_{i+1}), x_{j'+1}, x_i) \\ \cdot\, \varphi_{\text{do helix}}(i,j',0) \cdot \varphi_{\text{do multi}}(j',j,\min(2,n+1)) \end{pmatrix} & \text{if } 0 < i \le j < L \text{ and } 0 \le n \le 2 \end{cases}$$

As before, in the last case, the condition $i+2 \le j'$ ensures that $x_{j'}$ and x_{i+1} are valid, and the conditions $0 < i$ and $j < L$ ensure that $x_{j'+1}$ and x_i are valid.

4 The inside algorithm

The inside algorithm looks just like Viterbi, with max's replaced by $\sum$'s. We repeat these recurrences here, for convenience:

For $0 \leq i \leq L$,

$$\alpha_{\text{do outer}}(i) = \sum \begin{cases} 1 & \text{if } i = L \\ \phi_{\text{outer unpaired}} \cdot \alpha_{\text{do outer}}(i+1) & \text{if } 0 \leq i < L \\ \sum\limits_{\substack{i' \\ i+2 \leq i' \leq L}} (\phi_{\text{outer branch}} \cdot \alpha_{\text{do helix}}(i, i', 0) \cdot \alpha_{\text{do outer}}(i')) & \text{if } 0 \leq i \leq L \end{cases}$$

For $0 \leq n \leq d$ and $0 \leq i \leq j \leq L$,

$$\alpha_{\text{do helix}}(i,j,n) =$$

$$\sum \begin{cases} \phi_{\text{helix change}}[1] \cdot \phi_{\text{helix closing}}(x_{i+1}, x_j) & \text{if } 0 \leq i < i+2 \leq j \leq L \text{ and } n = 0 \\ \qquad \cdot \phi_{\text{helix base pair}}(x_{i+1}, x_j) \cdot \alpha_{\text{do helix}}(i+1, j-1, 1) & \\ \phi_{\text{helix change}}[n+1] \cdot \phi_{\text{helix stacking}}((x_i, x_{j+1}), (x_{i+1}, x_j)) & \text{if } 0 < i < i+2 \leq j < L \text{ and } 0 < n < d \\ \qquad \cdot \phi_{\text{helix base pair}}(x_{i+1}, x_j) \cdot \alpha_{\text{do helix}}(i+1, j-1, n+1) & \\ \phi_{\text{helix extend}} \cdot \phi_{\text{helix stacking}}((x_i, x_{j+1}), (x_{i+1}, x_j)) & \text{if } 0 < i < i+2 \leq j < L \text{ and } n = d \\ \qquad \cdot \phi_{\text{helix base pair}}(x_{i+1}, x_j) \cdot \alpha_{\text{do helix}}(i+1, j-1, d) & \\ \phi_{\text{helix closing}}(x_{j+1}, x_i) \cdot \alpha_{\text{do loop}}(i,j) & \text{if } 0 < i \leq j < L \text{ and } n > 0 \end{cases}$$

For $0 \leq i \leq j \leq L$,

$$\alpha_{\text{do loop}}(i,j) =$$

$$\sum \begin{cases} \varphi_{\text{hairpin}}(i,j) & \text{if } 0 < i \leq j < L \text{ and } n > 0 \\ \sum\limits_{\substack{i',j' \\ i \leq i' < i'+2 \leq j' \leq j \\ 1 \leq i'-i+j-j' \leq c}} (\varphi_{\text{single}}(i,j,i',j') \cdot \alpha_{\text{do helix}}(i',j',0)) & \text{if } 0 < i \leq j < L \text{ and } n > 0 \\ \phi_{\text{multi base}} \cdot \phi_{\text{multi paired}} & \text{if } 0 < i \leq i+2 \leq j < L \text{ and } n > 0. \\ \qquad \cdot \varphi_{\text{multi mismatch}}((x_i, x_{j+1}), x_{i+1}, x_j) \cdot \alpha_{\text{do multi}}(i,j,0) & \end{cases}$$

For $0 \leq n \leq 2$ and $0 \leq i \leq j \leq L$,

$$\alpha_{\text{do multi}}(i,j,n) =$$

$$\sum \begin{cases} 1 & \text{if } 0 \leq i = j \leq L \text{ and } n = 2 \\ \phi_{\text{multi unpaired}} \cdot \alpha_{\text{do multi}}(i+1, j, n) & \text{if } 0 \leq i < j \leq L \text{ and } 0 \leq n \leq 2 \\ \sum\limits_{\substack{j' \\ i+2 \leq j' \leq j}} \begin{pmatrix} \phi_{\text{multi paired}} \cdot \varphi_{\text{multi mismatch}}((x_{j'}, x_{i+1}), x_{j'+1}, x_i) \\ \cdot \alpha_{\text{do helix}}(i, j', 0) \cdot \alpha_{\text{do multi}}(j', j, \min(2, n+1)) \end{pmatrix} & \text{if } 0 < i \leq j < L \text{ and } 0 \leq n \leq 2 \end{cases}$$

5 The outside algorithm

The outside algorithm corresponding to the inside algorithm given in the previous section is shown below:

For $0 \leq i \leq L$,

$$\beta_{\text{do outer}}(i) = \sum \begin{cases} 1 & \text{if } i = 0 \\ \phi_{\text{outer unpaired}} \cdot \beta_{\text{do outer}}(i-1) & \text{if } i > 0 \\ \displaystyle\sum_{\substack{i' \\ 0 \leq i' \leq i'+2 \leq i}} (\phi_{\text{outer branch}} \cdot \alpha_{\text{do helix}}(i', i, 0) \cdot \beta_{\text{do outer}}(i')) & \end{cases}$$

For $0 \leq n \leq d$ and $0 \leq i \leq j \leq L$,

$$\beta_{\text{do helix}}(i,j,n) =$$

$$\sum \begin{cases} \phi_{\text{outer branch}} \cdot \beta_{\text{do outer}}(i) \cdot \alpha_{\text{do outer}}(j) & \text{if } 0 \leq i < i+2 \leq j \leq L \text{ and } n = 0 \\ \displaystyle\sum_{\substack{i',j' \\ 0 < i' \leq i < j \leq j' < L \\ 1' \leq i - i' + j' - j \leq c}} \Big(\varphi_{\text{single}}(i', j', i, j) \cdot \beta_{\text{do loop}}(i', j')\Big) & \text{if } 0 < i < i+2 \leq j < L \text{ and } n = 0 \\ \displaystyle\sum_{n'=0}^{1} \sum_{\substack{j' \\ j \leq j' < L}} \begin{pmatrix} \phi_{\text{multi paired}} \cdot \beta_{\text{do multi}}(i, j', n') \\ \cdot \varphi_{\text{multi mismatch}}((x_j, x_{i+1}), x_{j+1}, x_i) \\ \cdot \alpha_{\text{do multi}}(j, j', n'+1) \end{pmatrix} & \text{if } 0 < i \leq j < L \text{ and } n = 0 \\ \displaystyle\sum_{\substack{j' \\ j \leq j' < L}} \begin{pmatrix} \phi_{\text{multi paired}} \cdot \beta_{\text{do multi}}(i, j', 2) \\ \cdot \varphi_{\text{multi mismatch}}((x_j, x_{i+1}), x_{j+1}, x_i) \\ \cdot \alpha_{\text{do multi}}(j, j', 2) \end{pmatrix} & \text{if } 0 < i \leq j < L \text{ and } n = 0 \\ \phi_{\text{helix change}}[1] \cdot \phi_{\text{helix closing}}(x_i, x_{j+1}) & \text{if } 0 < i \leq j < L, \text{ and } n = 1 \\ \quad \cdot \phi_{\text{helix base pair}}(x_i, x_{j+1}) \cdot \beta_{\text{do helix}}(i-1, j+1, 0) & \\ \phi_{\text{helix change}}[n] \cdot \phi_{\text{helix stacking}}((x_{i-1}, x_{j+2}), (x_i, x_{j+1})) & \text{if } 1 < i \leq j < L-1, \text{ and } 1 < n \leq d \\ \quad \cdot \phi_{\text{helix base pair}}(x_i, x_{j+1}) \cdot \beta_{\text{do helix}}(i-1, j+1, n-1) & \\ \phi_{\text{helix extend}} \cdot \phi_{\text{helix stacking}}((x_{i-1}, x_{j+2}), (x_i, x_{j+1})) & \text{if } 1 < i \leq j < L-1 \text{ and } n = d \\ \quad \cdot \phi_{\text{helix base pair}}(x_i, x_{j+1}) \cdot \beta_{\text{do helix}}(i-1, j+1, d) & \end{cases}$$

For $0 \leq i \leq j \leq L$,

$$\beta_{\text{do loop}}(i,j) = \sum_{n'=1}^{d} \phi_{\text{helix closing}}(x_{j+1}, x_i) \cdot \beta_{\text{do helix}}(i, j, n') \qquad \text{if } 0 < i \leq j < L \text{ and } n > 0$$

For $0 \leq n \leq 2$ and $0 \leq i \leq j \leq L$,

$$\beta_{\text{do multi}}(i,j,n) =$$

$$\sum \begin{cases} \phi_{\text{multi base}} \cdot \phi_{\text{multi paired}} & \text{if } 0 < i < i+2 \leq j < L \text{ and } n = 0 \\ \quad \cdot \varphi_{\text{multi mismatch}}((x_i, x_{j+1}), x_{i+1}, x_j) \cdot \beta_{\text{do loop}}(i,j) & \\ \phi_{\text{multi unpaired}} \cdot \beta_{\text{do multi}}(i-1, j, n) & \text{if } 0 < i \leq j \leq L \text{ and } 0 \leq n \leq 2 \\ \displaystyle\sum_{\substack{i' \\ 1 \leq i' < i'+2 \leq i}} \begin{pmatrix} \phi_{\text{multi paired}} \cdot \varphi_{\text{multi mismatch}}((x_i, x_{i'+1}), x_{i+1}, x_{i'}) \\ \cdot \alpha_{\text{do helix}}(i', i, 0) \cdot \beta_{\text{do multi}}(i', j, n-1) \end{pmatrix} & \text{if } 2 < i \leq j < L \text{ and } 1 \leq n \leq 2 \\ \displaystyle\sum_{\substack{i' \\ 1 \leq i' < i'+2 \leq i}} \begin{pmatrix} \phi_{\text{multi paired}} \cdot \varphi_{\text{multi mismatch}}((x_i, x_{i'+1}), x_{i+1}, x_{i'}) \\ \cdot \alpha_{\text{do helix}}(i', i, 0) \cdot \beta_{\text{do multi}}(i', j, 2) \end{pmatrix} & \text{if } 2 < i \leq j < L \text{ and } n = 2. \end{cases}$$

6 Posterior decoding

Given the inside and outside matrices computed in the previous sections, we can now compute the posterior probabilities for paired and unpaired residues. Specifically, the posterior probability p_{ij} that nucleotide i pairs with nucleotide j (where $1 \leq i < j \leq L$) is given by

$$p_{ij} = \frac{1}{Z(x)} \cdot \sum \begin{cases} \phi_{\text{helix change}}[1] \cdot \phi_{\text{helix closing}}(x_i, x_j) & \text{if } 1 \leq i < j \leq L \text{ and } n = 0 \\ \qquad \cdot\, \phi_{\text{helix base pair}}(x_i, x_j) \cdot \alpha_{\text{do helix}}(i, j-1, 1) & \\ \qquad \cdot\, \beta_{\text{do helix}}(i-1, j, 0) & \\ \displaystyle\sum_{n=2}^{d} \begin{pmatrix} \phi_{\text{helix change}}[n] \cdot \phi_{\text{helix stacking}}((x_{i-1}, x_{j+1}), (x_i, x_j)) \\ \cdot\, \phi_{\text{helix base pair}}(x_i, x_j) \cdot \alpha_{\text{do helix}}(i, j-1, n) \\ \cdot\, \beta_{\text{do helix}}(i-1, j, n-1) \end{pmatrix} & \text{if } 1 < i < j < L \\ \phi_{\text{helix extend}} \cdot \phi_{\text{helix stacking}}((x_{i-1}, x_{j+1}), (x_i, x_j)) & \text{if } 1 < i < j < L \\ \qquad \cdot\, \phi_{\text{helix base pair}}(x_i, x_j) \cdot \alpha_{\text{do helix}}(i, j-1, d) & \\ \qquad \cdot\, \beta_{\text{do helix}}(i-1, j, d) & \end{cases} \tag{17}$$

where

$$Z(x) = \alpha_{\text{do outer}}(0) = \beta_{\text{do outer}}(L). \tag{18}$$

Using these posterior probabilities, the posterior decoding algorithm described in the full paper can be used to find the maximum expected accuracy parse.

7 Gradient

The gradient of the CONTRAfold conditional log-likelihood objective with respect to the parameters $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w} : \mathcal{D}) = \sum_{i=1}^{m} \left(\mathbf{F}(x^{(i)}, y^{(i)}) - \mathbb{E}_{y' \sim P(y|x^{(i)};\boldsymbol{w})}[\mathbf{F}(x^{(i)}, y')] \right),$$

where the expectation is taken with respect to the conditional distribution over structures y' for the sequence $x^{(i)}$ given by the current parameters $\boldsymbol{w}$. We now describe the construction of a dynamic programming algorithm for computing the expectation $\mathbb{E}_{y' \sim P(y|x^{(i)};\boldsymbol{w})}[\mathbf{F}(x^{(i)}, y')]$ based on modifying an implementation of the inside recurrences from Section 4.

First, initialize a vector $\boldsymbol{z} \in \mathbb{R}^n$ to the zero vector. In a typical implementation of the inside algorithm, computing entries of inside table involves repetitions of statements of the form

$$\alpha_a(i,j) \leftarrow \alpha_a(i,j) + (\text{product of some } \phi\text{'s}) \cdot (\text{product of some } \alpha_{a'}(i',j')\text{'s}).$$

We will replace each such statement with several statements—one for each ϕ_k appearing in the product above. Specifically, for each ϕ_k in the product, we will create a statement of the form

$$z_k \leftarrow z_k + \frac{\beta_a(i,j) \cdot (\text{product of some } \phi\text{'s}) \cdot (\text{product of some } \alpha_{a'}(i',j')\text{'s})}{Z(x)}$$

where $Z(x) = \alpha_{\text{do outer}}(0)$. At the end of this modified inside algorithm, then, the vector $\boldsymbol{z}$ will contain the desired feature expectations.

For example, applying the transformation to the rules for the $\alpha_{\text{do outer}}$ recurrence gives the following:

```
1 $\boldsymbol{z} \leftarrow \mathbf{0}$
2 for $i \leftarrow 0$ to $L$ do
3     if $i < L$ then
4         $z_{\text{outer unpaired}} \leftarrow z_{\text{outer unpaired}} + \beta_{\text{do outer}}(i) \cdot \phi_{\text{outer unpaired}} \cdot \alpha_{\text{do outer}}(i+1)$
5     end
6     for $i' \leftarrow i+2$ to $L$ do
7         $z_{\text{outer branch}} \leftarrow z_{\text{outer branch}} + \beta_{\text{do outer}}(i) \cdot \phi_{\text{outer branch}} \cdot \alpha_{\text{do helix}}(i,i',0) \cdot \alpha_{\text{do outer}}(i')$
8     end
9 end
```