RNA Secondary Structure Prediction using Deep Neural Network

TODO
Department of Electrical and Computer Engineering
University of Toronto
`todo@toronto.edu`

Abstract

TODO

1 Introduction

- State-of-the-art methods based on dynamic programming. Basic building blocks are known local structure, with their associated free energy measured experimentally. Fixed energy parameters and hand-crafted rules.
- Emerging new dataset calls for flexible, extensible, end-to-end model that can be trained on new types of dataset (noisy, missing value).
- TODO review other papers using NN and discuss what's lacking.

2 Method

2.1 Problem Formulation

RNA secondary structure can be represented by a binary upper triangular matrix (excluding the diagonal).

As an example, a short RNA sequence GUUGUGAAAU of length 10 (ID `CRW_00083` TODO ref to database) takes a structure that consists of a stem and a loop, as seen in Fig 1(a). This structure can be represented by the upper diagonal 10×10 matrix with all 0's, except for positions $y_{1,10}, y_{2,9}$ and $y_{3,8}$, all having value 1. This contiguous stretch of 1's corresponds to the stem formed by the three base pairs: G-U, U-A and U-A. (TODO define x and y first) (TODO cite FORNA)

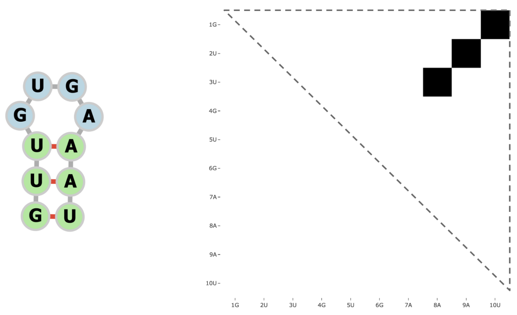

Figure 1: TODO

We formulate the predictive task as a conditional generative process. Specifically, given an input sequence with arbitrary length L: $\boldsymbol{x} = x_1, x_2, \ldots x_L$, we want to predict a distribution of structures conditioned on the sequence $P(\boldsymbol{Y} \mid \boldsymbol{x})$.

We factorize the conditional distribution as follows:

$$P(\boldsymbol{Y} \mid \boldsymbol{x}) = P(\{y_{i,i+1}\}_{i=1,2,\ldots L-1} \mid \boldsymbol{x})P(\{y_{i,i+2}\}_{i=1,2,\ldots L-2} \mid \boldsymbol{x}, \{y_{i,i+1}\}_{i=1,2,\ldots L-1}) \ldots P(y_{i,j} \mid \boldsymbol{x}, \{y_{todo}\})$$

The generative process implied by such formulation is illustrated in Fig 2. We generated one off diagonal slice at a time, conditioned on the input sequence, starting from the one adjacent to the diagonal line, as shown in yellow on the plot. When generating the second slice (in green), we condition on the input sequence and the generated values in the first slice (in yellow). When generating the third one (in blue), we condition on the input sequence and both the yellow and green slices. This process continues until we fill the upper triangular matrix, where the last entry (in red TODO color plot) is generated conditioned on the input and the entire upper triangular matrix except for itself.

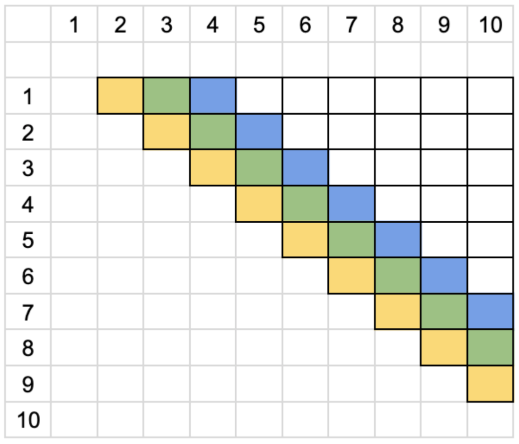

Figure 2: TODO

2.2 Model

We propose an architecture that encourages learning basic rules of base pairing and local structures, to construct the global structure, without having hard-coded parameters, such that the entire model can be learned end-to-end from sequence to structure. The architecture consists of the following components:

- two sets of 1D convolution layers on the 1-hot-encoded sequence
- Activations of each 1d conv layer (from both sets) are used to form a 2D feature map, where the (i, j)-th entry is the dot-product (can be replaced by a fully connected NN) between the activation of first set at position i, and the activation of second set at position j.
- Multiple 2D feature maps (formed via multiple layers of 1D conv) are concatenated, followed by a couple of 2D conv layers.
- Activation of the last 2D conv layer is concatenated with target from the previous "time-stamp" y^{t-1} (todo define notation), and the output is generated by an upper triangular convolution, which masks "future" timestamps and ensure the output is generated in auto-regressive fashion. (TODO more details on masked conv)
- Finally, there is a fully connected layer along the feature (3rd) dimension, with sigmoid activation to produce an output between 0 and 1, for each position in the upper triangular matrix.

At training time, different timestamps can be trained in parallel. At test time, we need to initialize the output at time $t = -1$, typically with a matrix of all zeros, then sample one slice at a time, until the full upper triangular matrix is filled. For a sequence of length $L - 1$, we need to run $L - 1$ steps sequentially. Note that multiple outputs can be sampled in parallel at test time.

TODO plot for NN architecture

2.3 Training

We trained the model using a synthetic dataset, constructed by sampling 50000 random sequences with length between 10 and 100. For each sequence, we ran RNAfold (TODO ref) with the default parameters and record the minimum free energy structure.

For each minibatch, we zero-pad the sequence array and structure matrix to the maximum length in the minibatch. When computing the loss and gradient, entries in structure matrix that were padded are being masks, in addition to the lower triangular entries (since we're only predicting the upper triangular matrix).

Note that although we present a single output structure for each input sequence at training time, the model is capable of generating a distribution of structures at test time.

TODO hyperparameters

2.4 Sampling structures

At test time, we can sample structures conditioned on the input sequence. As described in Section ?, we initialize the output structure with a matrix of all zeros, then sample one slice at a time until the upper triangular matrix is filled with sampled values. At each step, we sample a binary label for each position in the current slice based on the bernoulli probability predicted by the model. To ensure the sampled structure is valid, when sampling the label for location (i, j), if i-th or j-th position is already paired with another position (from samples in the previous timestamps), then we set $y_{i,j}$ to 0 without sampling from the model output.

3 Analysis

3.1 Generate distribution of structures

As an example, for a 323-base long sequence (TODO link to DB), we sampled 20 structure from the model output, as shown in Fig 3. On the left we show the binary upper triangular matrix sampled by the model (TODO it's too tiny to see anything), on the right we show the corresponding secondary structure rendered as a graph (only showing 4 due to space limit).

3.2 Run time comparison

3.3 Performance comparison

3.4 Interesting cases

3.5 Differentiable model

4 Conclusion

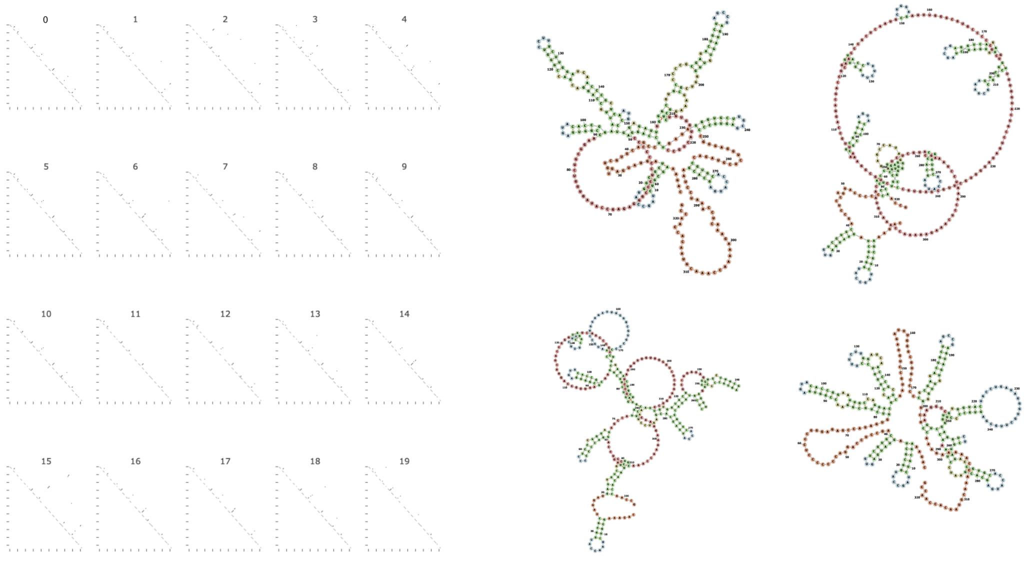

Figure 3: TODO