

### **CIBB 2019**



# On Secondary Structure Analysis by Using Formal Grammars and Artificial Neural Networks

Semyon Grigorev, Polina Lunina

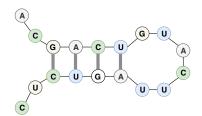
JetBrains Research, Programming Languages and Tools Lab Saint Petersburg University

September 6, 2019

## Genomic sequences analysis

- Problems
  - Genomic sequences classification
  - Subsequences detection
  - Secondary structure prediction
- Secondary structure handling
  - Covariance models
  - Hidden Markov Models
  - Probabilistic grammars
- Probability estimation for noisy data processing





#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

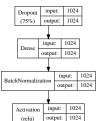
Result of classification

### Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

#### Neural Network

Dense neural network with more than 10 dense lavers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i, j] = 1 \iff$ s1  $\xrightarrow{*} \omega[i, j]$ , and 0 otherwise.

### Vectors

$$\left(\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

#### Result of classification

Neural Network

Parser

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Parser extracts features of the

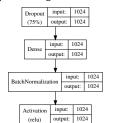
ture. Implementation of parsing

tiplications (Valiant, Okhotin)

and utilizes GPGPU.

given sequence secondary struc-

algorithm is based on matrix mul-



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i,j]=1 \iff$   $s1 \stackrel{*}{\longrightarrow} \omega[i,j]$ , and 0 otherwise.

#### Vectors

$$\left(\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

#### Result of classification

Neural Network

Parser

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Parser extracts features of the

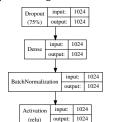
given sequence secondary struc-

ture. Implementation of parsing

tiplications (Valiant, Okhotin)

and utilizes GPGPU.

algorithm is based on matrix mul-



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i,j]=1 \iff$   $s1 \stackrel{*}{\longrightarrow} \omega[i,j]$ , and 0 otherwise.

#### Vectors

$$\left(\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

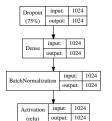
Result of classification

#### Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

#### Neural Network

Dense neural network with more than 10 dense lavers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i, j] = 1 \iff$ s1  $\xrightarrow{*} \omega[i, j]$ , and 0 otherwise.

#### Vectors

$$\left(\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

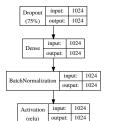
Result of classification

#### Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

#### Neural Network

Dense neural network with more than 10 dense lavers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i, j] = 1 \iff$ s1  $\xrightarrow{*} \omega[i, j]$ , and 0 otherwise.

#### Vectors

$$\left(\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

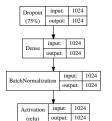
Result of classification

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

#### Neural Network

Parser

Dense neural network with more than 10 dense lavers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i, j] = 1 \iff$ s1  $\xrightarrow{*} \omega[i, j]$ , and 0 otherwise.

#### Vectors

$$\left(\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

#### Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure.

#### Sequences

Each sequence is treated as a text in  $\{A, C, G, U\}$  alphabet.

### Result of classification

Neural Network

and utilizes GPGPU.

Parser

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

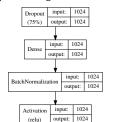
Parser extracts features of the

given sequence secondary struc-

ture. Implementation of parsing

tiplications (Valiant, Okhotin)

algorithm is based on matrix mul-



#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i,j]=1 \iff$   $s1 \stackrel{*}{\longrightarrow} \omega[i,j]$ , and 0 otherwise.

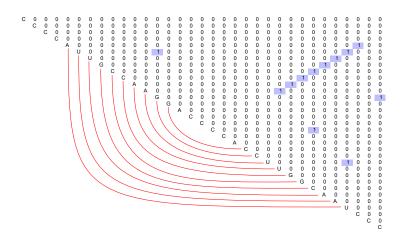
### Vectors

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so, it can be ignored.

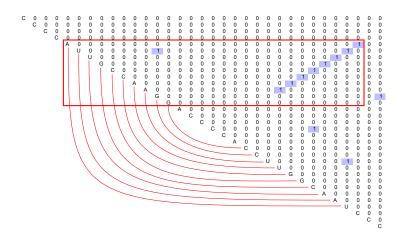
## Example

#### CCCCAUUGCCAAGGACCCCACCUUGGCAAUCCC



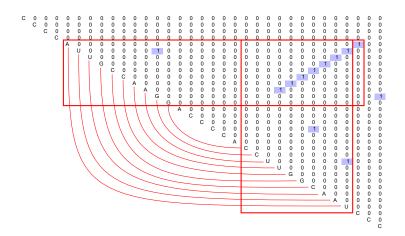
## Example

#### CCCCAUUGCCAAGGACCCCACCUUGGCAAUCCC



## Example

#### CCCCAUUGCCAAGGACCCCACCUUGGCAAUCCC



### Data locality preservation

- Represent parsing result as an image
- Use convolutional layers for these images processing
- Compare image- and vector-based networks on the same data

## Parsing results representation

#### Matrices

$$\left(\begin{array}{ccccc} 0 & 1 & 0 & \dots & 1 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots \end{array}\right)$$

Parsing result is a boolean matrix M which represents secondary structure features for sequence  $\omega$ :  $M[i,j]=1 \iff \mathfrak{s1} \stackrel{*}{\longrightarrow} \omega[i,j]$ , and 0 otherwise.

#### **Images**



The false bits of the matrix are represented as white pixels and the true bits as black ones. This approach makes it possible to process sequences of different lengths since the images can be transformed to a specified size. Data locality is preserved.

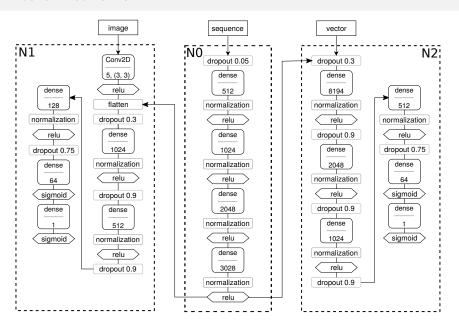
#### Vectors

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored. Requires the equal length of the input sequences and breaks the data locality.

## Parsing elimination

- Create a network which handles initial sequences
- Use two-staged learning
  - Train network on images or vectors for a given problem
  - ► Extend it by several input layers that take the initial nucleotide sequence as an input and convert it to the parsing result

### Neural networks



### **Evaluation**

- tRNA sequences analysis tasks
  - Classification into two classes: eukaryotes and prokaryotes
  - ► Classification into four classes: archaea, bacteria, plants and fungi
- Databases
  - tRNADB-CE
  - Genomic tRNA database

### Results

# EP — eukaryotes/prokaryotes task ABFP — archaea/bacteria/plants/fungi task

Classifier	EP		ABFP	
Approach	Vector-based	Image-based	Vector-based	Image-based
Base model	94.1%	96.2%	86.7%	93.3%
accuracy	94.170			
Extended model	97.5%	97.8%	96.2%	95.7%
accuracy	91.570			
Samples for	20000:5000:10000		8000:1000:3000	
train:valid:test	(57%:14%:29%)		(67%:8%:25%)	

### Results

EP — eukaryotes/prokaryotes task ABFP — archaea/bacteria/plants/fungi task

Classifier	Class	Vector-based approach		Image-based approach	
		precision	recall	precision	recall
EP	prokaryotic	95.8%	99.4%	96.2%	99.4%
	eukaryotic	99.4%	95.6%	99.4%	99.5%
ABFP	archaeal	91.1%	99.2%	91.6%	98.5%
	bacterial	96.6%	95.1%	95.2%	95.5%
	fungi	98.5%	94.9%	97.5%	94.3%
	plant	99.4%	95.7%	99.2%	94.7%

### Conclusion

- The modifications of our approach for biological sequences analysis were implemented
  - ▶ Parsing result in a form of image can be handled by convolutional layers
  - ► The parsing step can be removed from the trained model use which allows to run models on the original RNA sequences
- These modification improve the quality of the solution
- The improved version is applicable for real-world problems

### Future work

- Other RNA sequences analysis tasks
  - 16s rRNA classification
  - ► Chimeric sequences filtration
- Secondary structure prediction by using generative networks
- The use of deep convolutional networks for secondary structure analysis

### Contact Information

- Semyon Grigorev:
  - s.v.grigoriev@spbu.ru
  - Semen.Grigorev@jetbrains.com
- Polina Lunina: lunina polina@mail.ru
- Secondary structure analyzer project: https://research.jetbrains.org/groups/plt\_lab/projects?project\_id=43

## Thanks!