Project info

Project title: Deep learning with MXNet

Project short title (30 characters): Implement high-level APIs for user-friendly RNN construction.

URL of project idea page: https://github.com/rstats-gsoc/gsoc2016/wiki/Deep-learning-with-mxnet

Bio of Student

- Yun is a graduate student majoring in Computer Science at New York University with emphasis on data science. It is not surprising that he is not only familiar with regular conventions (e.g. Git) and programming languages (e.g. C++) of software development, but also has solid knowledge of theories related to neural network and other machine learning applications. As a proof-of-concept to implement high-level API of LSTM which is one of MXNet's github issues, he had already built a many-to-many RNN model from scratch which managed to learn 8-bit binary addition calculation. Model was implemented in Rcpp which fits the language requirement of this project. Therefore the CS background supports Yun to read the source code of MXNet, communicate with mentors, identify important but absent features, and ultimately finish implementation.
- Yun is an experienced R user who has been participating in developing R package. For example, ChIPseeker, a Bioconductor package for bioinformatics analysis; honfleuR, an extension supporting object-oriented programming in R S4 methods for an existed R package to analyze single-cell sequencing data. Therefore his strong interest and 4-year experience in R language make him a self-motivated and competent participant to help make a powerful deep learning package dedicated to R language community.
- Yun has rich research experience in computational biology with a Master degree in Molecular Biology. He realized deep learning is becoming a promising strategy to analyze large scale of data produced by biomedical studies. Therefore Yun has solid expert-domain knowledge and strong interest to apply deep learning to computational biology field by writing application examples for MXNet package. Therefore Yun's participation can bring to new impacts -- deep learning has a soft-landing on life sciences researches thanks to MXNet's R package and likewise MXNet becomes more appealing to bioinformaticians and broaden user spectrum.

CONTACT INFORMATION

Student name: Yun Yan

Melange Link_id: fix-me!!!

Student postal address: fix-me!!!

Telephone(s): 917-756-3868

Email(s): yy1533@nyu.edu

Student affiliation

Institution: New York University

Program: Computer Science, Tandon School of Engineering

Stage of completion: 2015.09 - 2017.06

Contact to verify: fix-me!!!

Schedule Conflicts:

Off-keyboards on Sundays, otherwise there is no time schedule conflicts. I am dedicated to GSoC this summer.

MENTORS

Mentor names: Qiang Kou, Yuan Tang

Mentor link_ids: Fix-me!!!

Contact with Mentors:

Date	Event	Media	Days after establishment of idea
17-Feb-2016	MXNet initiated idea page on rstats-GSoC	NA	0
24-Feb-2016	Finished a proof-of-concept of RNN via Rcpp and asked how to join in project	Github	7
2-Mar-2016	Submitted "PR" as test solution for first time	Github	14
2-Mar-2016	Applied for participate in project and asked for potential tasks	Email	14
3-Mar-2016	PR merged	Github	15
3-Mar-2016	Mentor reminded student of F1 visa status	Email	15
8-Mar-			

2016	First draft of student application form	Email&Github	20

CODING PLAN & METHODS

In light of the open issues listed on MXNet repository and discussions with mentors, there are three problem domains in this project proposal.

- R language specific domain where issues arise in R language layer are discussed.
- Deep learning specific domain where useful yet absent high-level APIs for advanced models and their solutions are discussed.
- Case study specific domain where why and how to apply MXNet to computational biology research field are discussed.

R language specific domain

Make model structure compatible with Rdata

Current model structure is saved via `structure(model, class="MXFeedForwardModel")`. As suggested in MXNet's issue #362, it involved overhead of dumping and recovering state which was not recommended for low-level API. Hence enhancements in terms of R language layer and model structure designs are proposed.

There are two possible solutions suggested by discussion of MXNet's developers.

1. Make model structure object via R's S4 methods.
2. Since model structure is saved as json string stored Rdata, two helper function are needed.

My plan for S4 methods to save model structure is to use the following sample function:

```
setClass("MXFeedForwardModel",
         slot = list(symbol = Rcpp_MXSymbol, arg.params = <mx_param_type>, aux.params = <mx
_param_type>))
```

My plan for json/Rdata methods is to add parameter to existed functions `mx.model.load` and `mx.model.save`, which has following syntax:

```
mx.model.save <- function(model, prefix, iteration, asRdata = TRUE) {}
mx.model.load <- function(model, prefix, iteration, useRdata = TRUE) {}
```

Alternatively *write two helper functions* suggested by MXNet's developer.

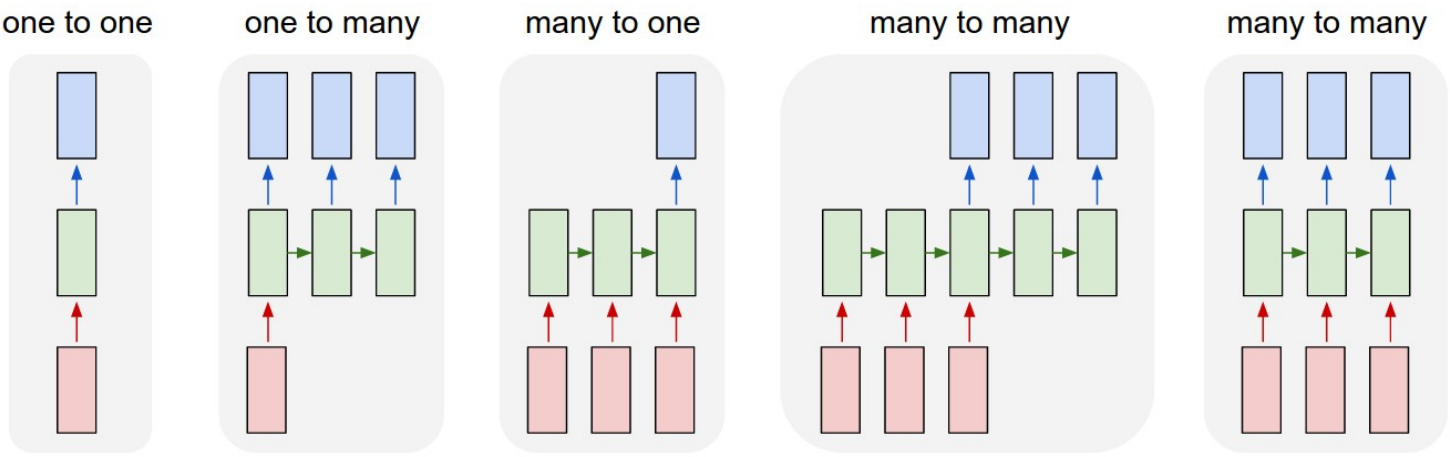
Deep learning specific domain

Regular R users would find it not straightforward to use currently available basic functions, e.g. `mx.symbol.FullyConnected`, `mx.symbol.Activation` to build advanced neural networks, e.g. recurrent neural network (RNN). Hence, high-level functions for building advanced networks are highly suggested be supported by MXNet R package.

There exists `mx.symbol.Convolution` for building convolution layer, but APIs for RNN are absent, as highlighted in MXNet's issue #1420.

RNN

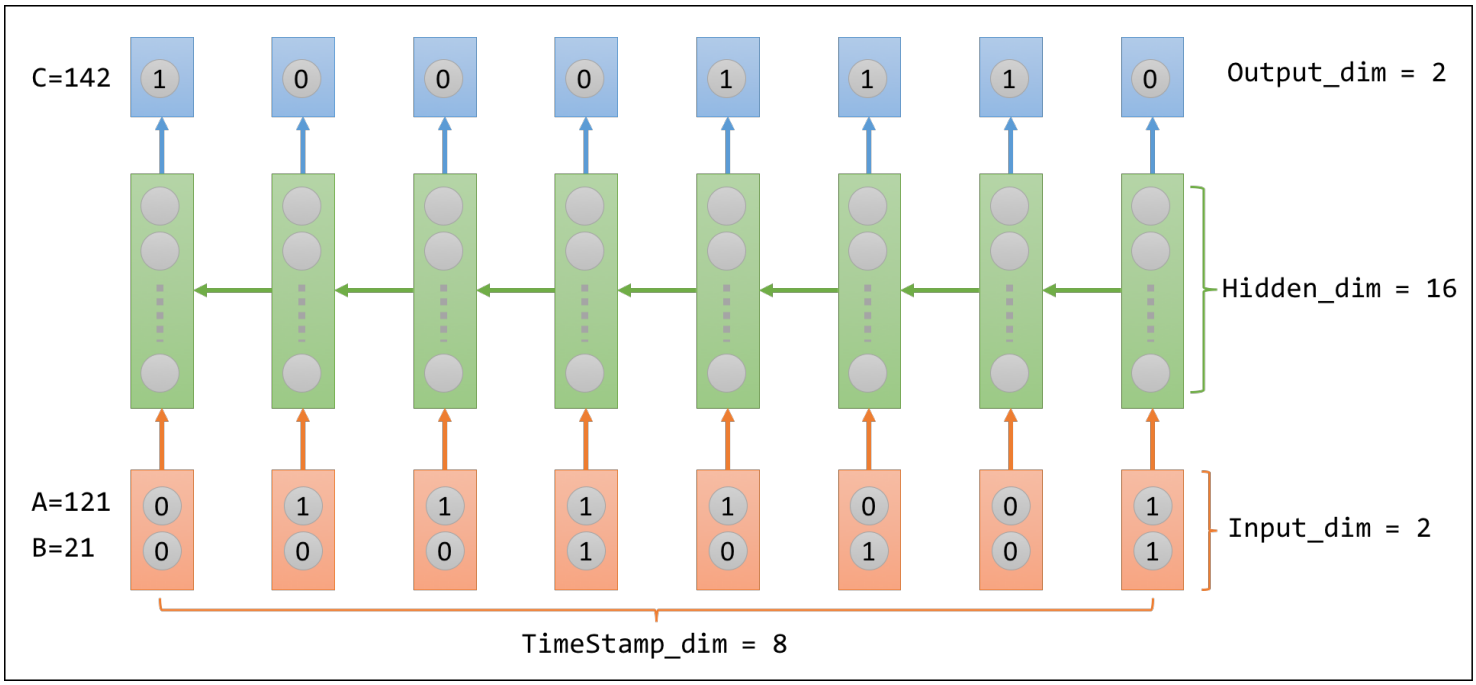
My plan is to implement high-level function supporting one-to-many, many-to-one, many-to-many, synced many-to-many RNN models.The product of this domain is expected to be a function with following snippet.



(Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

```
mx.symbol.RNN(symbol, name, hidden_dim, type = c('1toN', 'Nto1', 'NtoN', 'syncNtoN'), ...)
```

On one hand, as a proof-of-concept I had already implemented RNN to learn 8-digit binary calculus using Rcpp starting from scratch (See Gist), e.g. learning $01111001 + 00010101 = 10001110$ for $121 + 21 = 142$. Because the each digit is equivalent to timestamp, shown as following figure, my proof-of-concept implementation was indeed equivalent to and did manage to fulfill a synced many-to-many RNN construction.



(The green arrows indicating the direction of forward propagation within hidden layer are reversed for better illustration of binary addition.)

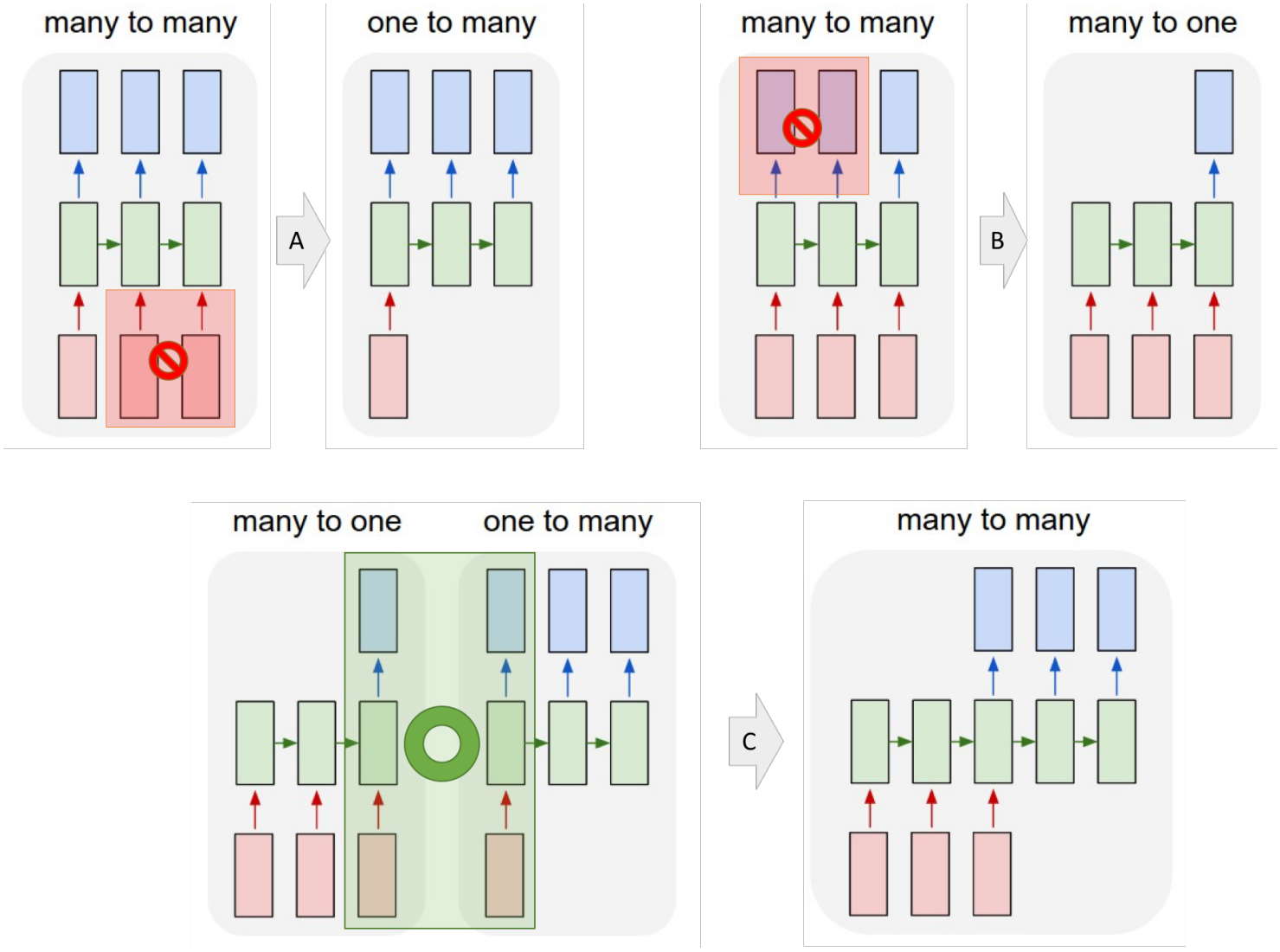
On the other hand, thanks to the fact that the approach of MXNet to build model is declarative, composing computation graph is more friendly for both R users and developers. Given the fact that I am clear about theory and implementation of RNN, translating imperative codes to declarative operations is what I am next expected

to do.

As a result of this proposal, the pipeline of building the same RNN is expected to be shown as following snippet which is declarative and standardized, rather than 200+ lines of customized and imperative C++ codes.

```
data <- mx.symbol.Variable("data") ## (N, input_dim=2, timestamp_dim=8)
net  <- mx.symbol.RNN(symbol = data, name = 'rnn', hidden_dim = 16, type = 'syncNtoN')
net <- mx.symbol.Activation(symbol = net, act.type = 'sigmoid')
net <- mx.symbol.LogisticRegressionOutput(net)
```

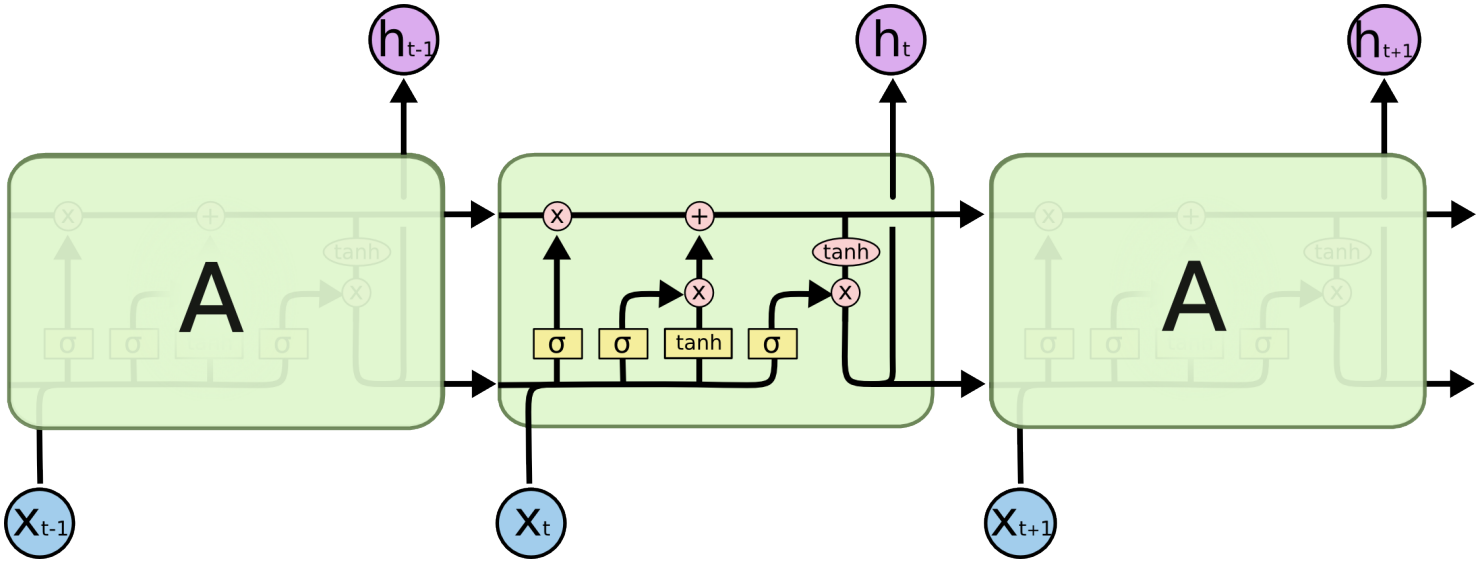
Once imperative-to-declarative translation were finished, solutions for composing graph of the rest three categories of RNN are expected to be straightforward, in light of the fact that these three are derivations of synced many-to-many RNN, shown as following diagram.



(Operation A: suppress part of input; Operation B: suppress part of output; Operation C: combine/group the two layers highlighted in green rectangular. Figures are modified from here)

LSTM

LSTM is specific type of RNN while it is independently listed given its popularity.

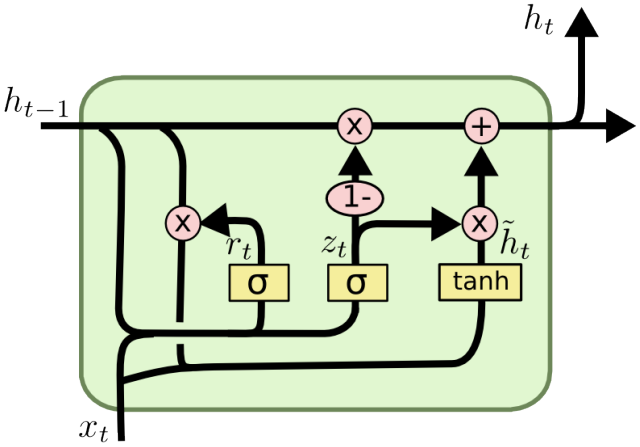


(Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

The developer of MXNet's Julia package once posted a step-by-step tutorial for constructing LSTM. Therefore I am expected to follow the logics to implement symbol operation for LSTM.

GRU

GRU is derivation of LSTM with simplified gates.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

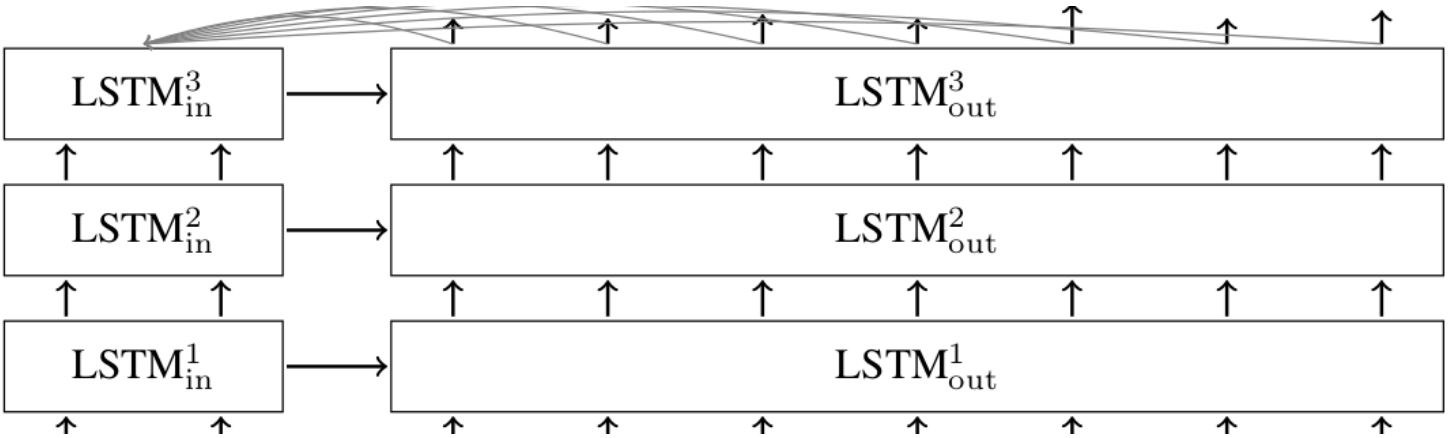
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

(Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Bidirectional RNN

Ref: http://www.cs.toronto.edu/~graves/asru_2013.pdf

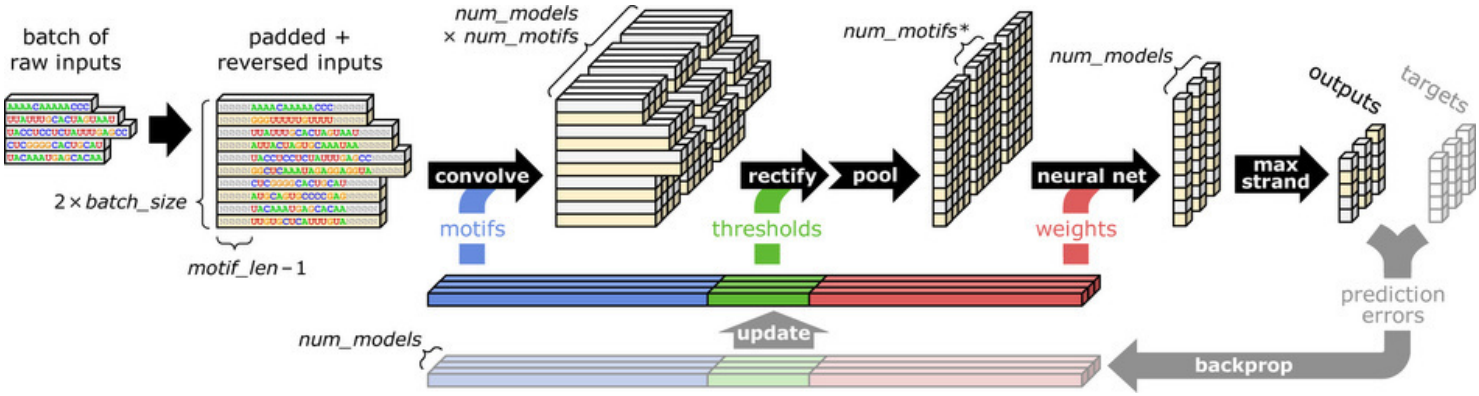
Multi-layer RNN



(Source: https://www.tensorflow.org/versions/r0.7/tutorials/seq2seq/index.html)

Application specific domain

Reproduce the CNN model published on Nature paper to draw attention of R users in bioinformatics, computational biology field.



(Source: http://www.nature.com/nbt/journal/v33/n8/full/nbt.3300.html)

TIMELINE

Each working period is composed of 4-days coding, 1-day for general testing, 1-day for writing documents and case studies.

Week	Mon	Tue	Wed	Thu	Fri	Sat	Project
5/2/16							MXNet Demo
5/9/16							Paper Reading
5/16/16							Contact Mentor
5/23/16	Code (Starts)	Code	Code	Code	Test	Doc	RNN Many-to-Many
5/30/16	Code	Code	Test	Code	Code	Doc (Ends)	ditto
	Code					Doc	

6/6/16	(Starts)	Code	Code	Code	Test	(Ends)	RNN 1-to-N
6/13/16	Code (Starts)	Code	Code	Code	Test	Doc (Ends)	RNN N-to-1
6/20/16	Eval	Code (Starts)	Code	Code	Test	Doc (Ends)	RNN N-to-N
6/27/16	Doc	Doc (Starts)	Code	Code	Code	Code	Bi-RNN
7/4/16	Test	Doc (Ends)	Code (Starts)	Code	Code	Code	Stack-RNN
7/11/16	Test	Doc (Ends)	Code (Starts)	Code	Code	Code	LSTM
7/18/16	Test	Doc (Ends)	Code (Starts)	Code	Code	Code	GRU
7/25/16	Test	Doc (Ends)	Contact	Code (Starts)	Code	Code	Application
8/1/16	Code	Test (Ends)	Contact	Doc (Starts)	Code	Code	R-language specific domain
8/8/16	Code	Code	Test	Doc	Code	Code	ditto
8/15/16	Code	Code	Test	Doc (Ends)	Test (Starts)	Test	Wrap
8/22/16	Doc (Ends)	END	Eval	Eval	Eval	Eval	Final Eval
8/29/16	SUBMIT						

MANAGEMENT OF CODING PROJECT

My codes are fork of MXNet

Fork of MXNet's repository: https://github.com/Puriney/mxnet.

Travis tests

My codes can directly use the Travis tests currently used by MXNet's main repository thus the only thing I am expected to do is to linking my repository to Travis.

Expected Commits Frequency

Commits will be pushed in every 2 days. Commits within a working period (6 days) are squashed as one

commit to make history clean and friendly to be ready to be merged into MXNet's main repository.

Being absent for 10 days suggests I must come across with problems.

TEST

I submitted a "pull request" which afterwards merged (See: here) into MXNet main repository.