UNIVERSITY OF SCIENCE
ADVANCED PROGRAM IN COMPUTER SCIENCE

THAI THIEN - 1351040
VAN DUY VINH - 1351050

SỬ DỤNG LATEX TRONG KHOÁ LUẬN TỐT NGHIỆP

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

HO CHI MINH CITY, 2017

UNIVERSITY OF SCIENCE
ADVANCED PROGRAM IN COMPUTER SCIENCE

THAI THIEN - 1351040
VAN DUY VINH - 1351050

SỬ DỤNG LATEX TRONG KHOÁ LUẬN TỐT NGHIỆP

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THESIS ADVISOR
NGHIÊM QUỐC MINH

HO CHI MINH CITY, 2017

ACKNOWLEDGMENTS

Tôi xin chân thành cảm ơn . . .

Xin cám ơn!

Tp. Hồ Chí Minh, ngày ... tháng ... năm
2016
Sinh viên thực hiện
Thai Thien Van Duy Vinh

Contents

List of Figures

List of Tables

ABSTRACT

Tóm tắt khóa luận: trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được. Bản tóm tắt dài từ 1 đến 2 trang.

Chapter 1

Introduction

Chapter 2

Technical Issues

2.1 Framework

2.1.1 Torch

Torch [1] is Lua scientific computing framework. Torch support high performing matrix calculation via multi-dimensional array call Tensor. Torch are built with C/C++, CUDA backend. Torch author choose Lua because Lua works well with C/C++ [1]. Thus, Torch is high performing and support GPU. Torch have neural network package (nn) package. Computation graph must be define before forward pass. A simple, single linear layer network can be easily defined with few line of code (see listing 2.1).

```
-- simple y = Ax + b linear layer
l = nn.Linear(2,3)
-- forward pass
x = torch.Tensor(2)
y = l:forward(x) -- vector dimension of 3
```

Listing 2.1: Simple linear layer in Torch

However, when a model need multiple module, such as multilayer perceptron (MLP), these module must be put into container. Figure A.1 illustrates on function of each nn container . In order to construct two-layer perception (eq 2.1), linear, tanh and softmax module must be packed into sequential module (see listing 2.2).

$$\begin{aligned} h &= tanh(W_1 * x + b_1) \\ y &= softmax(W_2 * h + b2) \end{aligned} \tag{2.1}$$

[1] http://torch.ch/

```
model = nn.Sequential()
model:add(nn.Linear(2,3))
model:add(nn.Tanh())
model:add(nn.Linear(3,5))
model:add(nn.SoftMax())
-- forward
x = torch.Tensor(2)
y = model:forward(x)
```

Listing 2.2: MLP in Torch

Torch provide nngraph package support build more complicate model. For example, define MLP in (eq 2.1) use nngraph (see listing 2.3)

```
model = nn.Sequential()
model:add(nn.Linear(2,3))
model:add(nn.Tanh())
model:add(nn.Linear(3,5))
model:add(nn.SoftMax())
-- forward
x = torch.Tensor(2)
y = model:forward(x)
```

Listing 2.3: MLP using nngraph

Sample code on training a model, see Appendix A.1

2.1.2 Theano

Theano [2] is a deep learning library on Python. It basic function is similar to Torch: matrix calculation, support GPU. Theano is define-and-run schema, which a computer graph must be built before it is executed.

```
x = T.dmatrix('x')
y = T.dmatrix('y')
z = x + y
f = function([x, y], z)
f([[1, 1], [2, 2]], [[3, 3], [4, 4]])
# result [[4, 4], [6, 6]]
```

Listing 2.4: Define function in Theano

[2] http://deeplearning.net/software/theano/

Comparing to Torch7, Theano are slower on most benchmark [1]. Theano does not provide nice template like linear layer. Thus, model must be defined from equation. It give researcher more control over mathematics aspect but cause more trouble for beginner. A sample code for MLP A.2. One more problem is that the 'define-and-run' scheme does not suitable for recursive neural network due to recompile the computation graph each training sample take time.

2.1.3 Pytorch

PyTorch uses same backend as Torch. However, PyTorch specially designed for Python. Pytorch have pre-define module (Linear layer, Convolution layer) like Torch. However, Pytorch does not require to pack model into container. In Pytorch a network are defined in forward-pass thanks to Dynamic Neural Networks feature. Therefore, user can use Python control flow to define a network. For example, one can use for loop to run recurrent neural network (see listing 2.5) .The features allows us to implement Recursive Neural Network for NLP, which the network change for every sample, much more easier.

```python
import torch
import torch.nn as nn
rnn = nn.RNNCell(10, 20)
seq_len = 10
input_dim = 100
hidden_dim = 150
input = Variable(torch.randn(seq_len, 1, input_dim))
hx = Variable(torch.zeros(1, hidden_dim))
output = []
for i in range(6):
    hx = rnn(input[i], hx)
    output.append(hx)
```

Listing 2.5: RNN

We also implement treelstm from original Torch7 [3] sentiment classification task in PyTorch and publish on Github [4].

We choose PyTorch because:

[3] https://github.com/stanfordnlp/treelstm

[4] https://github.com/ttpro1995/TreeLSTMSentiment

- Dynamic Neural Networks feature works well on data sequence with different length
- Intuitive framework
- Easy to install and run on CUDA

Chapter 3

Experiment

3.1 Dataset

3.1.1 Stanford Sentiment Treebank

In this thesis, we use Standford Sentiment Treebank (SST) dataset [2]. Standford Sentiment Treebank contains 11,855 sentences. Each data sentence consist of fined-grain sentiment labeled phrases in constituency parse tree structure (see **Figure 3.1**). There are total 215,154 phrases in whole dataset. The dataset was splitted into train/dev/test contain 8544/1101/2210 sentences each for training and evaluation models. After remove neutral sentiment sentences, there are 6920/872/1821 sentences remained in train/dev/test set.

SST dataset are publicly available online [1].

[1] https://nlp.stanford.edu/sentiment/index.html

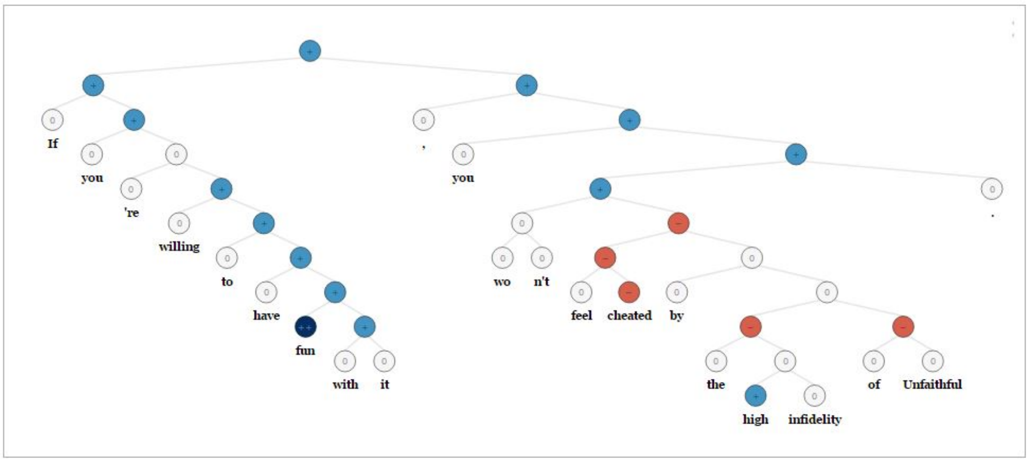

Figure 3.1: A parsed sentence in SST [a]

Preprocess

We use preprocess source code from [2] implementation [2] to preprocess SST.

3.1.2 Amazon Movies Review

We get Amazon Movies and TV reviews (4,607,047 reviews) and Amazon Book reviews (22,507,155 reviews) [**he2016ups**]. Listing **??** is sample of one book review.

```
{
  "reviewerID": "AH2L9G3DQHHAJ",
  "asin": "0000000116",
  "reviewerName": "chris",
  "helpful": [5, 5],
  "reviewText": "Interesting Grisham tale of a lawyer that takes millions of dollars from his firm after faking his own death. Grisham usually is able to hook his readers early and ,in this case, doesn't play his hand to soon. The usually reliable Frank Mueller makes this story even an even better bet on Audiobook.",
  "overall": 4.0,
  "summary": "Show me the money!",
  "unixReviewTime": 1019865600,
  "reviewTime": "04 27, 2002"
```

[a] Render by Pytreebank `https://github.com/JonathanRaiman/pytreebank`

[2] `https://github.com/stanfordnlp/treelstm`

Table 3.1: My caption

	Constituency Tree-LSTM	LSTM
Glove 42B		
Glove 840B		
Glove (Amazon)	88.45	
Glove (Amazon Sorted)	88.85	
Paragram-Phrase XXL		
SSWEu		

```
}
```

Listing 3.1: Amazon reviews sample

Preprocess

Step 1: We extract reviewText and overall from review dataset. We assume that overall valus are sentiment score for reviews. Reviews with overall 5 is very positive and 0 is very negative. We keep asin, reviewText, overall and ommit other data points.

Step 2: We group dataset by product (reviews with same asin). Then for each product, we sorted by overal.

Step 3: We dump all reviewText into plain text file. We preprocess Standford Tokenizer [3].

We also make a version of unsorted dataset. We preprocess as we do to our sorted dataset. However, in **Step 2:**, instead sort review, we suffle all reviews.

Train Glove on Amazon Review dataset

We train our word representation from Amazon dataset using Glove [**pennington2014glove**] on 15 iteration with windows size of 20.

3.2 Evaluation

Danh mục công trình của tác giả

1. Tạp chí ABC

2. Tạp chí XYZ

TABLE OF CONTENTS

English

[1] Collobert, Ronan, Kavukcuoglu, Koray, and Farabet, Clément. “Torch7: A matlab-like environment for machine learning”. In: *BigLearn, NIPS Workshop*. EPFL-CONF-192376. 2011.

[2] Socher, Richard et al. “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631. 2013, p. 1642.

[3] Tokenizer, Stanford. “Part-of-Speech Tagger, and Named Entity Recognizer. Downloaded from: h ttp”. In: *nlp. stanford. edu/software* ().

Appendix A

APPENDICES 1

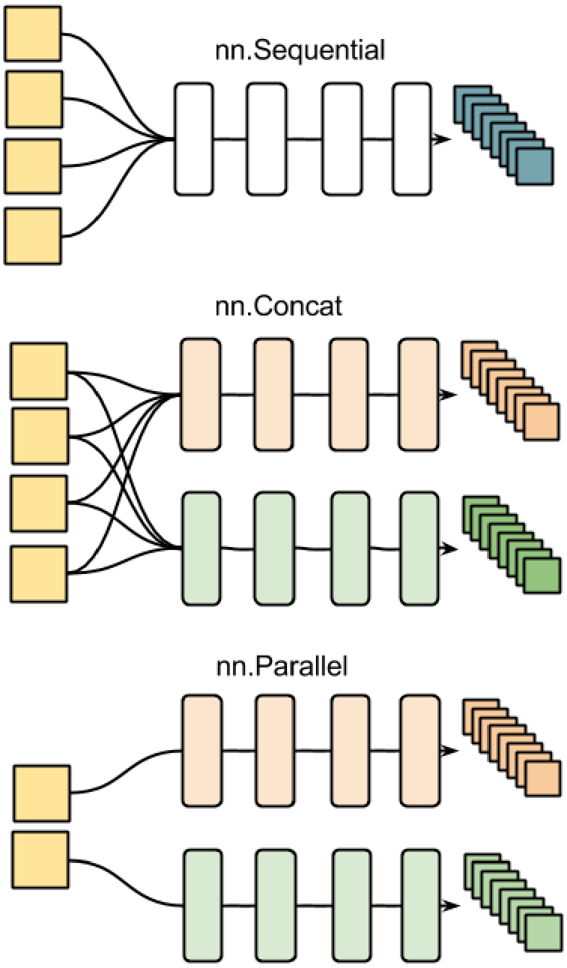

Figure A.1: Torch nn container [a]

[a] https://github.com/soumith/cvpr2015/blob/master/Deep%20Learning%20with%20Torch.ipynb

```lua
require 'nn'
require 'optim'

model = nn.Sequential()
model:add(nn.Linear(1,1))

criterion = nn.MSECriterion()

x = torch.Tensor{{1,2,3,4,5,6,7,8,9,10}}
x = x:t()
y = torch.Tensor{{3,5,7,9,11,13,15,17,19,21}}
y = y:t()

params, gradParams = model:getParameters()

function feval(params)
gradParams:zero()
local outputs = model:forward(x)
local loss = criterion:forward(outputs,y)
local dloss_doutput = criterion:backward(outputs,y)
model:backward(x, dloss_doutput)
return loss, gradParams
end

local optimState = {
learningRate = 0.01
}

for epoch = 1, 100 do
optim.sgd(feval,params, optimState)
end

test = torch.Tensor{{1,3,5,7,9,11,100}}
test = test:t()

print (model:forward(test))
```

Listing A.1: MLP using nngraph

```python
import numpy as np
import theano
import theano.tensor as T
import theano.tensor.nnet as nnet
```

```

class MLP:
def __init__(self):
x = T.dvector()
y = T.dscalar()
t1 = np.array(np.random.rand(3, 3), dtype=theano.config.floatX)
theta1 = theano.shared(t1)  # 3x3 weight matrix
t2 = np.array(np.random.rand(4, 1), dtype=theano.config.floatX)
hid1 = MLP.sigmoid_layer(x, theta1)  # hidden layer
theta2 = theano.shared(t2)  # 4x1 weight matrix
output_layer = T.sum(MLP.sigmoid_layer(hid1, theta2))
fc = (output_layer − y)∗∗2
self.cost = theano.function(inputs=[x,y], outputs = fc,updates=[
(theta1, Xor.grad_desc(fc, theta1)),
(theta2, Xor.grad_desc(fc, theta2))
])
self.run_forward = theano.function(inputs=[x],outputs=output_layer)

@staticmethod
def sigmoid_layer(x, w):
b = np.array([1], dtype=theano.config.floatX)
new_x = T.concatenate([x,b])
m = T.dot(w.T, new_x)
h = nnet.sigmoid(m)
return h


@staticmethod
def grad_desc(cost, theta):
alpha = 0.1
return theta − (alpha∗ T.grad(cost, wrt=theta))

def forward(self, x):
output = self.run_forward(x)
return output


def train(self, train_x, train_y, n_epoch):
cur_cost = 0
for epoch in range(n_epoch):
for i in range(len(train_x)):
cur_cost = self.cost(train_x[i],train_y[i])
```

```python
if epoch % 1000 == 0:
print(cur_cost)
print ('train complete')
```

Listing A.2: Theano MLP

Appendix B

APPENDICES 2

Đây là phụ lục 2.