Deep Learning

Sentence-level Sentiment Classification with RNN

Author: Yantian Luo

Institute: Electronic Engineering

ID: 2018310742

Update: May 11, 2019

Contents

1 Introduction **1**

2 Algorithm Design **2**

2.1 RNNCell 2

2.2 GRUCell 2

2.3 LSTMCell 2

3 Results **4**

3.1 Plot the loss value and accuracy value of one-layer RNN with 3 rnncells against to every epoch during training 4

3.1.1 RNNCell 4

3.1.2 GRUCell 4

3.1.3 LSTMCell 5

Chapter 1 Introduction

In this homework, we focus on fine-grained sentence-level sentiment classification problem.

We use Stanford Sentiment Treebank (SST) dataset for experiments. This dataset contains 11,855 sentences, and has been splitted into the training / validation / test parts, containing 8,544 / 1,101 / 2,210 sentences respectively. Each sentence is categorized into one type of 5 sentiments. To preprocess the dataset, we use torchtext package. Before training the model, we need to tokenize words, building vocabulary, construct word embedding and intialize data iterator. torchtext has good wrappers for these operations and we can directly use dataloader in a similar way as before.

Chapter 2 Algorithm Design

In this homework, we need to complete the forward process of RNNCell, GRUCell and LSTMCell using PyTorch. Considering these three cells have a common forward API: def forward(self, input, state). We can design the following algorithms.

2.1 RNNCell

We can use Algorithm 1 to complete the forward process of RNNCell.

Algorithm 1 the forward algorithm of RNNCell

```
Input: input: $x$, hidden state: $s = [h, h]$(where $h$ is hidden state vector)
Output: new state: $s'$
1: $h' = \tanh(W_{ih}x + b_{ih} + W_{hh}h + b_{hh})$
2: return $s' = [h', h']$
```

2.2 GRUCell

We can use Algorithm 2 to complete the forward process of GRUCell.

Algorithm 2 the forward algorithm of GRUCell

```
Input: input: $x$, hidden state: $s = [h, h]$(where $h$ is hidden state vector)
Output: new state: $s'$
1: $r = \sigma(W_{ir}x + b_{ir} + W_{hr}h + b_{hr})$
2: $z = \sigma(W_{iz}x + b_{iz} + W_{hz}h + b_{hz})$
3: $n = \tanh(W_{in}x + b_{in} + r * (W_{hn}h + b_{hn}))$
4: $h' = (1 - z) * n + z * h$
5: return $s' = [h', h']$
```

2.3 LSTMCell

We can use Algorithm 3 to complete the forward process of LSTMCell.

Algorithm 3 the forward algorithm of LSTMCell

```
Input:   input: $x$, hidden state: $s = [h, h]$(where $h$ is hidden state vector)
Output:   new state: $s'$
1: $r = \sigma(W_{ir}x + b_{ir} + W_{hr}h + b_{hr})$
2: $z = \sigma(W_{iz}x + b_{iz} + W_{hz}h + b_{hz})$
3: $n = \tanh(W_{in}x + b_{in} + r * (W_{hn}h + b_{hn}))$
4: $h' = (1 - z) * n + z * h$
5: return $s' = [h', h']$
```

Chapter 3 Results

3.1 Plot the loss value and accuracy value of one-layer RNN with 3 rnncells against to every epoch during training

In this section, we use $lr = 0.01$ with 3 rnncells to complete the experiments, and the loss value and accuracy value curves are as follow.

3.1.1 RNNCell

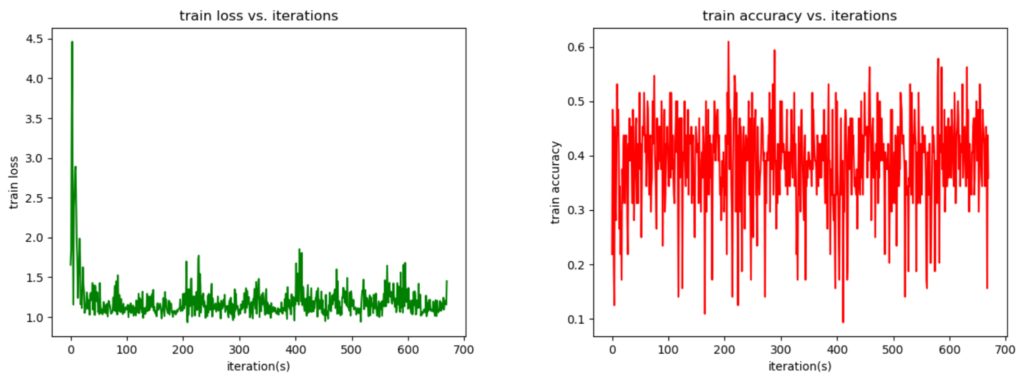

Figure 3.1: train loss curve and train accuracy curve using RNNCell against to every iteration

3.1.2 GRUCell

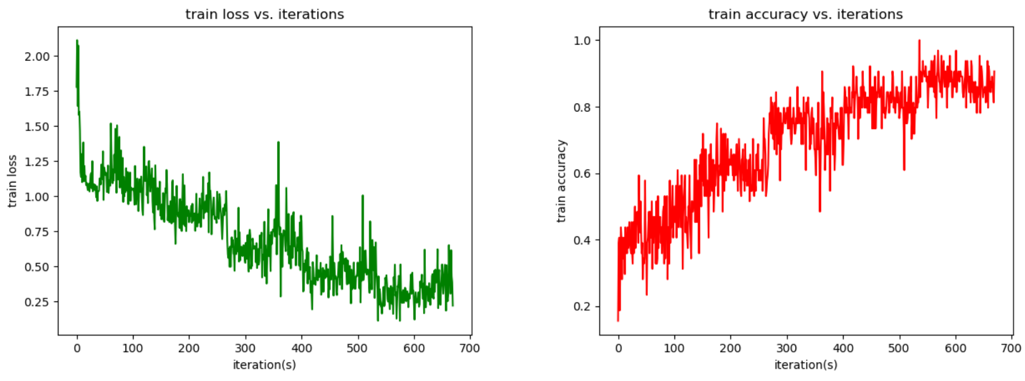

Figure 3.2: train loss curve and train accuracy curve using GRUCell against to every iteration

3.1.3 LSTMCell

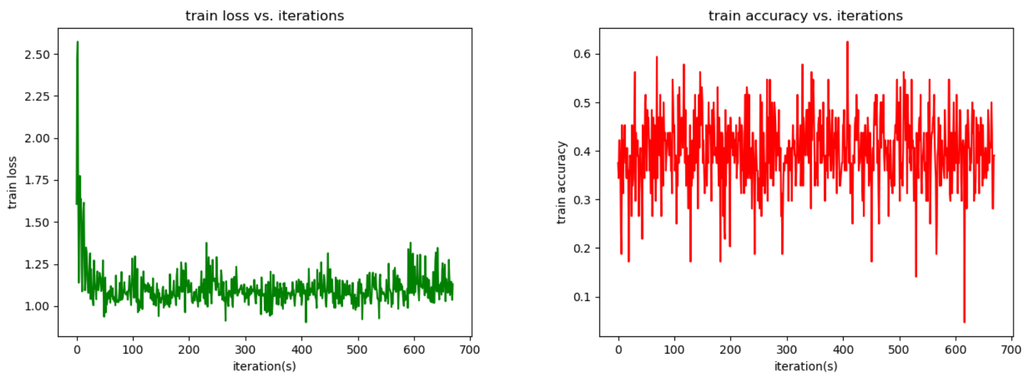

Figure 3.3: train loss curve and train accuracy curve using LSTMCell against to every iteration

3.2 Compare and analyze the performance of the 3 rnncells