ECE 544NA Fall 2016 Assignment 5

Jifu Zhao Date: 11/18/2016

TensorFlow

1. Methods

(1) In this assignment, we first implement the recurrent neural network in the format of basic RNN and LSTM. The input image has the dimension of 28×28, in this work, we can treat each pixel as one input, which has 784 steps, or treat each column as one input, which has 28 steps. Using basic RNN or LSTM, we finally get 4 different models. The whole structure of the RNN class is shown in Figure 1.

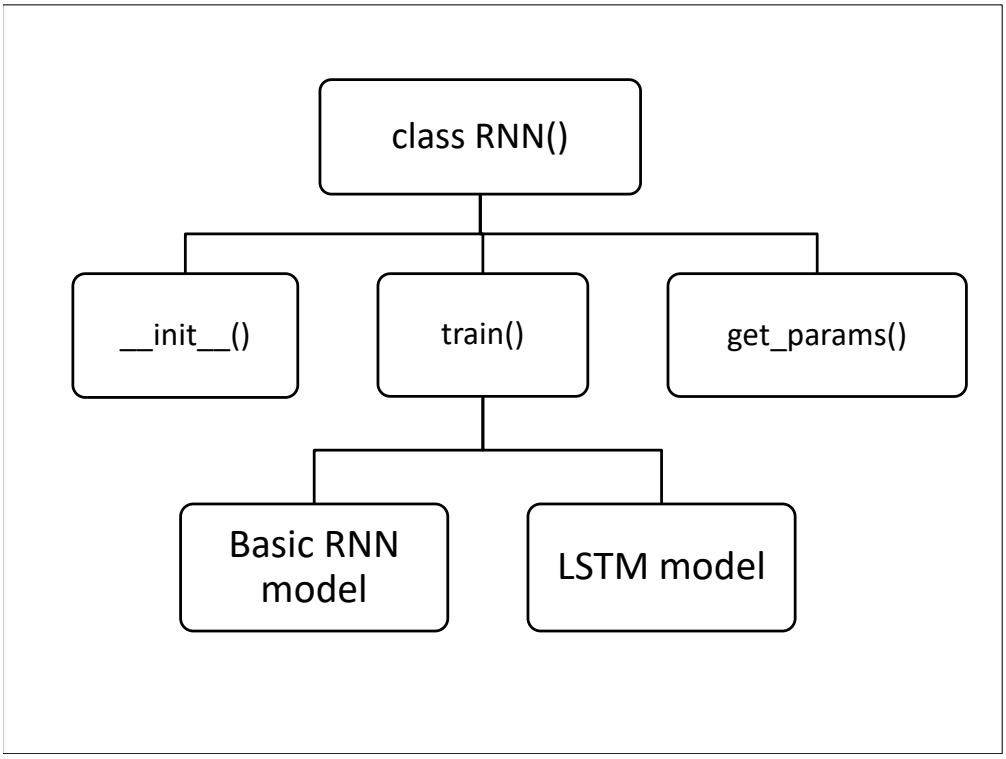

Figure 1: Algorithm Structure

In class RNN(), after initialization using the function $\_\_init\_\_()$, the RNN model will be trained according to the given parameters using the function $train()$. In this process, the we will first create the so-called RNN cell using the function $tf.nn.rnn\_cell.BasicRNNCell()$ or $tf.nn.rnn\_cell.BasicLSTMCell()$ for basic RNN model or LSTM model. Then the RNN will be created using the function $tf.nn.rnn(cell)$. The output from the RNN model has 100 dimensions, and we will use a fully-connected neural network for the final classification problem, where the final output has 10 dimensions. In this process, we find that when the

final output is the linear connection rather than the softmax connection, the model trained much faster and the accuracy is also good. So, we choose linear activation function and using the cross-entropy as the loss function. Through using the Adam optimizer, we can train the overall model directly. Due to the size of the large dataset, we choose mini-batch learning method and finally evaluating the model on all the training set and testing set.

Note: In this section, we followed some online tutorials for RNN implementions, including:

a. $https://www.tensorflow.org/versions/r0.11/tutorials/recurrent/index.html$

b. $https://tensorhub.com/aymericdamien/tensorflow-rnn$

c. $https://github.com/tflearn/tflearn/blob/master/examples/images/rnn\_pixels.py$

d. $https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/api_docs/python/functions_and_classes/shard0/tf.nn.rnn.md$

(2) Limited by the computation resources, we didn't train too many iterations, especially for the model with 784 steps. The detailed parameter settings are listed in Table 1.

Table 1: Parameter settings

Model	Basic RNN (t=784)	LSTM (t=784)	Basic RNN (t=28)	LSTM (t=28)
steps size	784	784	28	28
input size	1	1	28	28
learning rate	0.0005	0.0001	0.0001	0.0001
iterations	2000	1000	5000	5000
batch size	100	100	100	100

2. Results

(1) During the training process, in each iteration, we also record the mini-batch training accuracy. The convergence curves for mini-batch training-corpus accuracy of 4 different models are shown in Figure 2.

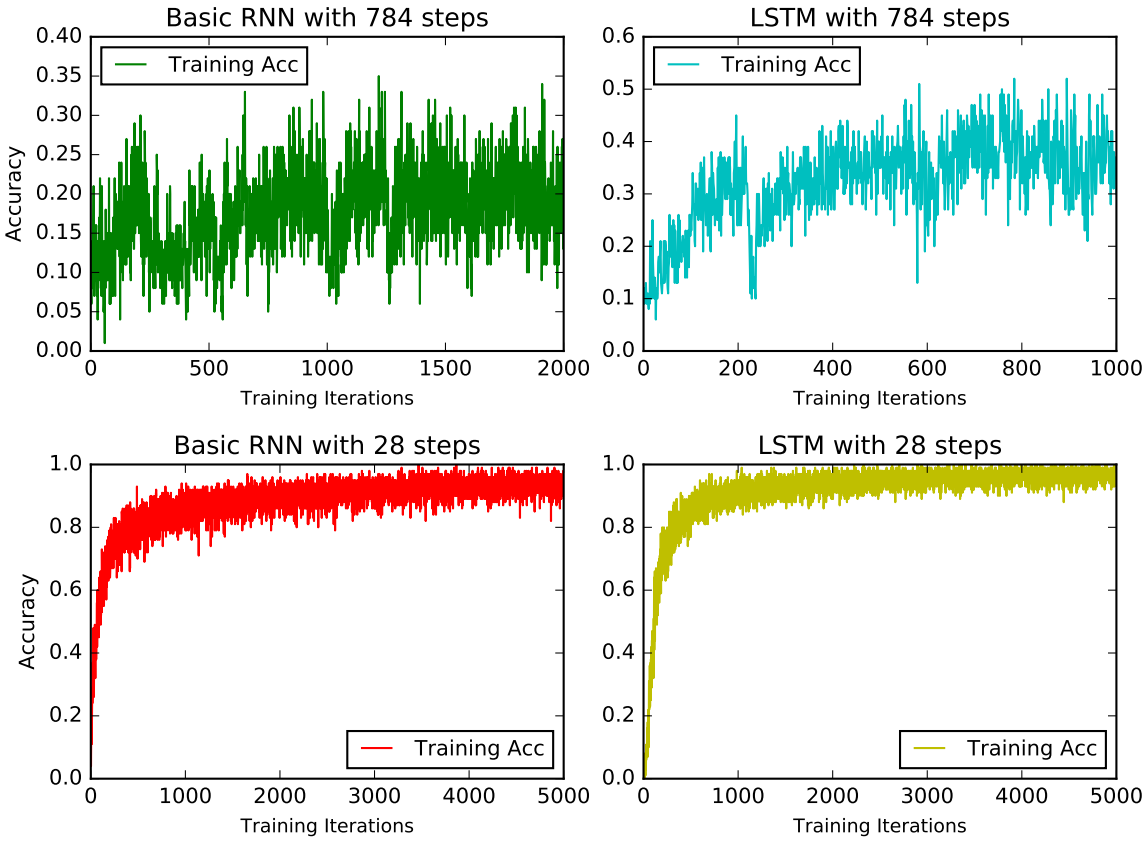

Figure 2: Convergence Curve

Since the mini-batch is not very big (100 in this case), the learning curves for Basic RNN and LSTM with 784 steps are not very smooth. But for 28 steps case, both basic RNN model and LSTM model converge much better.

(2) After finishing the iterations, we evaluate the model on the all training set and testing set. The final training and testing accuracy are shown in Table 2

Table 2: Training and testing accuracies

Model	Training Set	Testing Set
Basic RNN (t=784)	20.40%	20.95%
LSTM (t=784)	40.82%	40.76%
Basic RNN (t=28)	93.61%	94.20%
LSTM (t=28)	96.68%	96.51%