Final Project Report

FS19-ECE-802-602 - Selected Topics

Date: 12/12/2019

Name: Vishal Asnani

LSTM Equations:

1. Standard LSTM Model

The simple RNN model can be mathematically expressed as:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

where Whx, Whh, bh, Why and by are adaptive set of weights and σ is a nonlinear function. In the base LSTM model, the usual activation function has been equivalently morphed into a more complicated activation function, so that the hidden units enable the back propagated through time (BBTT) gradients technique to function properly [4]. The base LSTM uses memory cells in the base network with incorporated three gating mechanisms to properly process the sequence data. To do so, the base LSTM models introduce new sets of parameters in the gating signals and hence more computational cost and slow training speed.

The standard LSTM architecture described here is similar to as described in [5] but without peep-hole connections. The equations for the standard LSTM memory blocks are given as follows:

$$i_t = \sigma_{in}(U_i h_{t-1} + W_i x_t + b_i)$$
$$f_t = \sigma_{in}(U_f h_{t-1} + W_f x_t + b_f)$$
$$o_t = \sigma_{in}(U_o h_{t-1} + W_o x_t + b_o)$$
$$\tilde{c_t} = \sigma(U_c h_{t-1} + W_c x_t + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t}$$
$$h_t = o_t \odot \sigma(c_t)$$

where it, ft and ot are the input gate, forget gate and output gate at time t. The equations include non-linearity σ.

2. LSTM Variants

Two variants are used in the model described in this paper:

(a). LSTM Model 5a

A fixed real number with absolute value less than 1 has been set for the forget gate in order to preserve bounded-input-bounded-output (BIBO) stability. Meanwhile, the output gate is set to 1 (which in practice eliminates this gate altogether). Also, the bias term in the input gate equation is preserved.

$$
\begin{aligned}
&i_t = \sigma_{in}(u_i \odot h_{t-1} + b_i) \\
&f_t = \alpha, \ -1 < \alpha < 1 \ (\text{default}{=}0.96) \\
&o_t = 1.0 \\
&\tilde{c_t} = \sigma(U_c h_{t-1} + W_c x_t + b_c) \\
&c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \\
&h_t = o_t \odot \sigma(c_t)
\end{aligned}
$$

(b). LSTM Model 6

All gating equations replaced by appropriate constant. For BIBO stability, we found that the forget gate must be set ft = 0.59 or below. The other two gates are set to 1 each (which practically eliminate them to the purpose of computational efficiency).

$$
\begin{aligned}
&i_t = 1.0 \\
&f_t = \alpha, -1 < \alpha < 1 \ (\text{default}{=}0.59) \\
&o_t = 1.0 \\
&\tilde{c_t} = \sigma(W_c x_t + U_c h_{t-1} + W_c x_t + b_c) \\
&c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \\
&h_t = o_t \odot \sigma(c_t)
\end{aligned}
$$

C. Dataset

The dataset used for training both models described in the paper is UCI News aggregator dataset. This dataset contains headlines, URLs, and categories for 422,937 news stories collected by a web aggregator between March 10th, 2014 and August 10th, 2014.There are different categories included in the dataset. These are business; science and technology; entertainment; and health. Different news articles that refer to the same news item are also categorized together.

The dataset contains many columns for segregating the data. These are:

- ID: the numeric ID of the article
- TITLE: the headline of the article
- URL: the URL of the article
- PUBLISHER: the publisher of the article
- CATEGORY: the category of the news item; one of: -- b : business -- t : science and technology -- e : entertainment -- m : health
- STORY: alphanumeric ID of the news story that the article discusses
- HOSTNAME: hostname where the article was posted

- TIMESTAMP: approximate timestamp of the article's publication, given in Unix time (seconds since midnight on Jan 1, 1970)

To obtain the training and test dataset, sklearn.model_selection.train_test_split() function is used.

There are four categories of output: Business- ‘b’; Science – ‘t’; Entertainment- ‘t’ and health- ‘m’.

These classes contain unbalanced data. Therefore, number of categories is chosen to be 45000 and the dataset is reordered based on index and label value. A hot encoding is used for an efficient implementation of the model. The categorical outputs are labelled as [1. 0. 0. 0.] – e; [0. 1. 0. 0.] - b; [0. 0. 1. 0.] - t and [0. 0. 0. 1.] – m. Tokenizer API is used for creating unique tokens by eliminating any other character. This is done for setting up of the input data into integer.

Model Description

The model was built by using high level Keras API. The model uses different layers at every stage. Firstly, embedding layer is used to generate word embeddings. When one-hot encoding is applied to words, we end up with sparse vectors of high dimensionality and the semantics of the words is not considered. To address these two issues, embedding layer is used with dimension 128. Three convolution layers are then added with filter size as 64, kernel size as 3 and ‘Relu’ activation. Every convolution layer is followed by Max pooling layer. LSTM5a and LSTM6 layers are then added to the model with 64 hidden units, 20% dropout and ‘tanh’ activation. Dropout layer with value 20% is added after convolution block and every LSTM block. Finally, a dense layer with ‘sigmoid’ activation is added. Table I summarizes the network specification.

The model is made to run for 10 epochs. RMSprop optimizer with loss as binary cross entropy which uses accuracy as metrics is used. Exponential learning rate is used to update the learning rate after every epoch.

Table I: Network Specification

Input dimension	130x128
Number of hidden units	64
Non-linear function	tanh, sigmoid, softmax
Output dimension	4
Number of epochs	10
Batch size	64
Optimizer	RMSprop
Loss Function	Categorical cross-entropy

Experimental Evaluation

We now summarize the results obtained by training both the models i.e. the standard LSTM model and the model with two parameter-reduced variants of LSTM which are LSTM5a and LSTM6 on UCI News-aggregator dataset. Both the models were made to run for 10 epochs.

Table II summarizes the results obtained for both the models. As we can clearly see that by adding two parameter-reduced variants of LSTM together in one model with convolution layers have increased the efficiency by 1%. Fig. 1 shows the model accuracy and loss of both the models.

The proposed model was also made to run for different learning rates of the RMSprop optimizer used. Three learning rates 0.001, 0.005 and 0.01 were used to train the dataset. The results of the modified model with different learning rates are summarized in Table III.

Fig.2 shows the model accuracy for different learning rates. Corresponding loss graph of two different learning rates is shown in Fig. 3.

TABLE II

COMPARISON OF TWO MODELS

Model	Accuracy (%)	Loss
Standard LSTM	95.1	0.14
Modified LSTMs	96.1	0.132

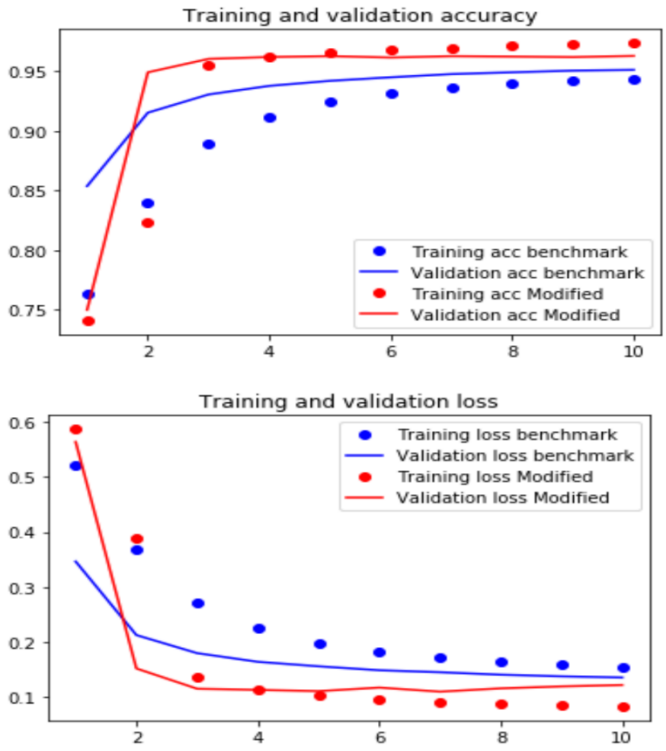

Fig. 1 Accuracy and loss of model with standard and modified LSTM

TABLE III

COMPARISON OF MODIFIED MODEL WITH TWO DIFFERENT LEARNING RATES

Learning rate	Accuracy (%)	Loss
0.001	96	0.133
0.01	62.6	5.997

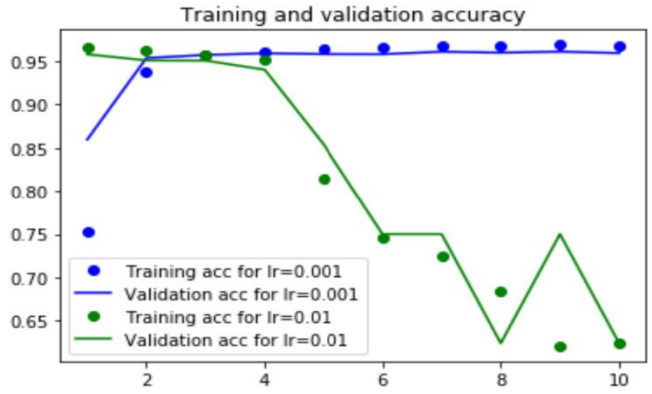

Fig. 2 Accuracy of model with learning rate as 0.001 and 0.01.

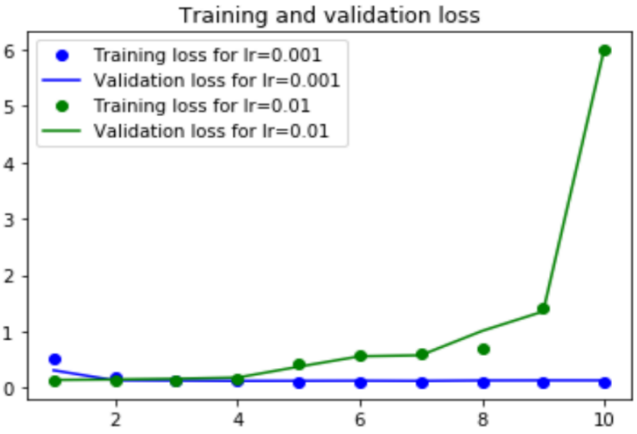

Fig. 3 Loss of model with learning rate as 0.001 and 0.01.

Conclusion

In this paper, a model with two parameter-reduced variants of standard LSTMs and convolutional layers was compared with the model comprising of standard LSTM layer. The two LSTM used were LSTM 5a and LSTM 6. Both the variants were defined using different values of gates.

The experimental results show that the model with LSTM variants is giving much more accuracy as compared to the one with standard LSTM. There has been a 1% increase in the accuracy of the modified model.

Due to resource constraints, the models in this paper could only be tested for low number of epochs. For future line of research, one could run these models for more number of epochs to see if performance is improved or not. UCI dataset which is used to train both the models had only four

categories. A dataset with a greater number of categories can be tested on the model to determine the performance.

References

[1] Hochreiter, S., Schmidhuber, J. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

[2] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555, 2014.

[3] Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

[4] Salem, F., “A Basic Recurrent Neural Network Model,” arXiv. Preprint arXiv: 1612.09022, Dec. 2016.

[5] Akandeh, A., Salem, F. M., Simplified Long Short-term Memory Recurrent Neural Networks: part II, arXiv. Preprint arXiv:1707.04623v1