*Recurrent Neural Network for*

*Time Series Prediction*

Raghuram Mandapalli   
Computer Science  
University of Texas at DallasTX,USA  
rxm180079@utdallas.edu  
  
  
Khaja Ghouse Mohiuddin Mohammed, CS  
University of Texas at DallasTX,USA  
kmm190000@utdallas.edu  
  
  
Rajat Kathal  
Computer Science*f organization* University of Texas at DallasTX,USA  
rxk190020@utdallas.edu

***Long short term memory networks(LSTMs) is one of the recuurent neural network(RNN) that was structured to model sequences and their dependencies with accuracy compared to conventional RNNs.Long short term memory networks(LSTMs) a variation of recurrent neural networks are an effective solution for sequence prediction problems like predicting stock sales, foreign exchange rates, future weather prediction, etc.In this paper we will explore Long short term memory (LSTMs) for predicting the national names. Here, we will introduce the distributed training of LSTM RNNs using asynchronous stochastic gradient descent optimization on the dataset. This architecture makes the best use of model parameters than the others considered and outperforms a deep feed forward neural network having an order of magnitude more parameters.***

Keywords— Long short term memory networks, recurrent neural networks, sequence prediction.

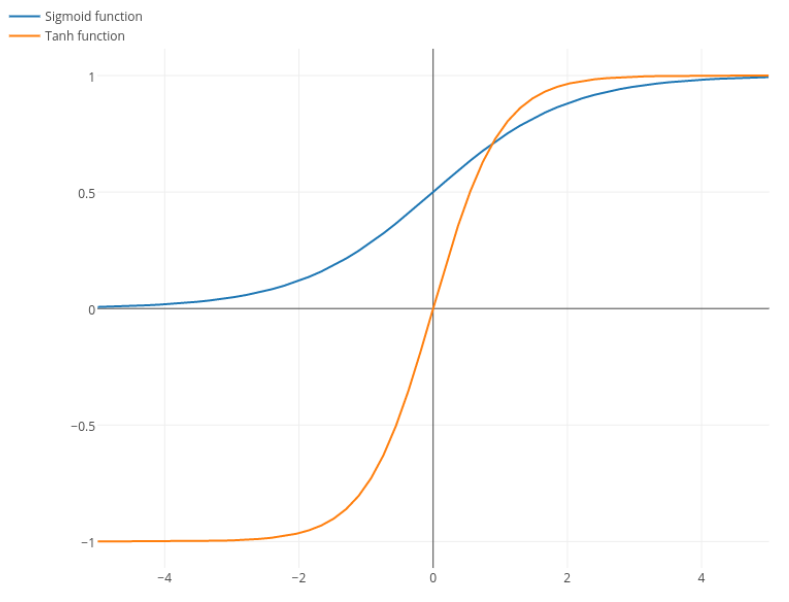
# Introduction

The traditional neural networks are not suitable for tasks that involve sequences of data like in language translation, time series, etc. In order to solve this issue, recurrent neural networks were built. Recurrent Neural Networks are feedforward neural networks along with internal memory. They not only take the current input into consideration but also the previous time step output while deciding the output for the current time step, this way it remembers and utilizes the past information. But the problem with recurrent neural networks is that they suffer from the problem of vanishing or exploding gradient and have a short memory. This problem can be overcome by using long short term memory networks(LSTM). The key difference between recurrent neural networks and LSTM is that the LSTM cell has the ability to decide whether it should store particular information or ignore it with the help of its gates, this way they can keep just the important information. In this report, we discuss the LSTM RNN architecture and its performance on our dataset.

# LSTM Neural network model description

We used recurrent neural network with long short term memory as our basic model to capture the structure of character sequences.In this model we took input of NationNames as a dataset and for each name transform the data to maximum length by adding dot(.) and map each character to id's. For training a dataset, we splitted the transform data into groups and converted each name into one hot encoding.

We will use sigmoid and tanh for activation functions.



*Sigmoid = 1/(1+exp(-X))*

*Tanh = (exp(X) - exp(-X)) / (exp(X) + exp(X))*

*Softmax = exp(X)/(sum(exp(X),1))*

# https://github.com/navjindervirdee/neural-networks/blob/master/Recurrent%20Neural%20Network/LSTM.JPG?raw=true

*forgetActivation(fa) = sigmoid(forgetW \* [xt,at-1])*

*inputActivation(ia) = sigmoid(inputW \* [xt,at-1])*

*gateActivation(ga) = tanh(gateW \* [xt,at-1])*

*outputActivation(oa) = sigmoid(outputW \* [xt,at-1])*

*ct = (forgetActivation \* ct-1) + (inputActivation \* gateActivation)*

*OuputCell at = outputActivation \* tanh(ct)*

forgetActivation is a sigmoid layer that takes the output at-1 and tht current input at time t and combines them into

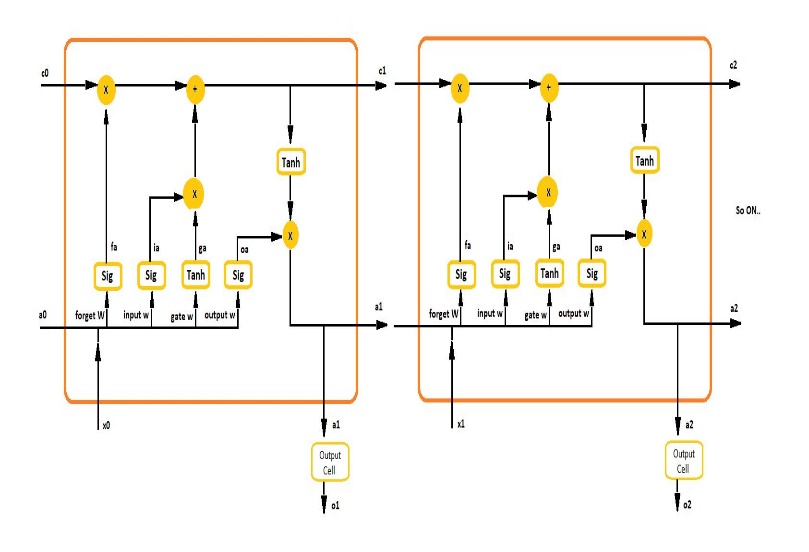
single unit and applies a linear tranformation followed by a sigmoid.

InputActivation gate takes the previous output and the new imput and passes them to another sigmoid layer. This gate returns a value between 0 and 1 and the previous state is multiplied by the forget gate and then added to new one by output gate.

ouputActivation gate controls how much of the internal state is passed to output and it works in a similar way to other ones.

Where inputW, outputW, forgetW are trainable parameters and \* denotes element-wise multiplication. Long term memory can be restored from ct where the information is written with inputActivation ia and previous cell ct-1 and updates with forgetActivation fa. While ouputcell is represented by multiplying outputActivation with tanh of ct cell.

After getting proper embeddings for the group dataset, we did forward propagation:



## Forward Pass

Let xt be the input vector at time t, n be the number of lstm cells and m be number of inputs then we get weights

* Input weights
* Forget weights
* Gate weights
* Output weights

Then the vector formula be like

ia = σ(*inputW \* [xt,at-1]*)

oa = σ(*outputW \* [xt,at-1]*)

fa = σ(*forgetW \* [xt,at-1])*

ot = g(oa)

where σ, g and h are point-wise non linear activation functions

. The logistic sigmoid is used as activation gate function

and g(x)=tanh(x) is called as input and output activation functions

In forward propagation, store the initial activations in cache, unroll the names, get embeddings and store the time t activations in cache and calculate loss, accuracy.

*Loss =* ***∑****(mean (log(pred) \* labels))*

*Accuracy(t) = (Y==predictions,axis=1) for all time steps*

*Accuracy = ((****∑****Acc(t))/batch\_size)/n for all time steps, n is number of chars in name*

After calculating loss and accuracy, we calculated ouputcell errorand single LSTM cell error.

In backward propagation, we applied chain rule and calculated the errors for each step and stores the derivatives in dictionary.We updated parameters with Adam Optimizer which is classical stochastic gradient descent procedure which updates network weights iteratively based on the training data.

*Using Exponentially Weighted Averages*

*Vdw = beta1 x Vdw + (1-beta1) x (dw)*

*Sdw = beta2 x Sdw + (1-beta2) x dw^2*

*W = W - learning\_rate x ( Vdw/ (sqrt(Sdw)+1e-7)*

# DATASET

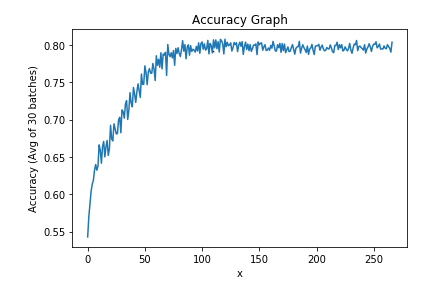
We used NationalNames.csv dataset from data.gov. The objective of the task is to predict how babies born in the US are named. For cleaned dataset, we preprocessed the raw data as follows: we lowercased all the names, removed unnecessary paranthesis and added dot (.) to the names to equal length.

# Results and analysis

We used Adam optimizer for LSTM model and learning rate was set to 0.01 for every 100 iteraions. Additionally batch of size 20 was used.We’ve tabulated epochs, loss and accuracy for 1000 epochs and observed that loss is decreasing after few 100 epochs and accuracy goes increasing after increasing epochs.

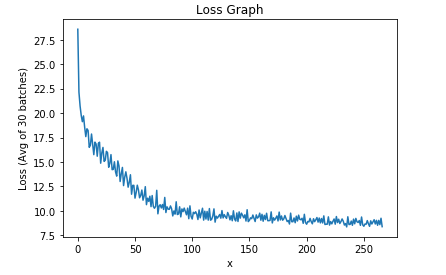
|  |  |  |
| --- | --- | --- |
| Epochs | Loss | Accuracy |
| 100 | 0.395 | 71.232 |
| 200 | 0.323 | 71.818 |
| 300 | 0.327 | 72.727 |
| 400 | 0.407 | 69.545 |
| 500 | 0.358 | 71.364 |
| 600 | 0.298 | 76.818 |
| 700 | 0.349 | 72.722 |
| 800 | 0.293 | 75 |
| 900 | 0.228 | 80 |
| 1000 | 0.315 | 75.909 |

We calculated accuracy and loss over 250 batches and plotted a graph.



By observing that, as number of epochs increases

Accuracy is increasing.



By observing the graph, loss is decreasing as epochs increasing.

# References

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[2] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” Neural Networks, IEEE Transactions on, vol. 5, no. 2, pp. 157–166, 1994.

[3] F. A. Gers and J. Schmidhuber, “LSTM recurrent networks learn simple context free and context sensitive languages,” IEEE Transactions on Neural Networks, vol. 12, no. 6, pp. 1333–1340, 2001.

[4]. Pearlmutter, B. A. (1995). Gradient calculations for dynamic

recurrent neural networks: A survey. IEEE Transactions on

Neural Networks, 6(5):1212{1228. Pineda, F. J. (1987).

Generalization of back-propagation to recurrent neural networks.

[5]. Mozer, M. C. (1989). A focused back-propagation algorithm

for temporal sequence recognition. Complex Systems

[6]. Ring, M. B. (1993). Learning sequential tasks by incrementally

adding higher orders. In S. J. Hanson, J. D. C. and Giles, C. L.,

editors, Advances in Neural Information Processing Systems 5,

pages 115{122. Morgan Kaufmann..

[7]. Williams, R. J. (1989). Complexity of exact gradient

computation algorithms for recurrent neural networks.

[8] M. Hermans and B. Schrauwen, “Training and analysing

deep recurrent neural networks,” in Advances in Neural Information

Processing Systems, 2013.