Updating the Forget Gate of the LSTM to Improve Pattern Learning

Preston Robertson

Department of Industrial and Systems Engineering, Mississippi State University, Mississippi State, MS, 39762 USA

Abstract

The Long Short Term Memory(LSTM) model is a neural network that specializes in time-series based data... The goal of this project is to improve the forget gate of the LSTM model to improve the pattern learning of the model.

Keywords: Example, example, example, example.

1. Introduction

The problem of analyzing time-series based data has been a goal of researchers, since normal artificial neural network learning can not be directly applied. The architecture of a standard neural network or a convolutional neural network(CNN) can not change or update as time goes on after training the model. For example, the artificial neural network or CNN will handle image data but can not have a video as the input data. This is due to the inability to vectorize video data compared to image data. That is where the recurrent neural network(RNN) is implemented. The architecture of the recurrent neural network allows for a feed-forward network that learns from results from previous nodes. The RNN models can also not have a video format as input data; however, the model processes each image in the video data in such a way that the model still learns as if the images were still in video formats. This change to architecture allows for new types of data to be analyzed, such as: video, speech, vibrations, robot control, sign language translation, etc.

The RNN model has a common issue of vanishing gradient when learning. This vanishing gradient refers to the learning slowdown of the model. This vanishing gradient occurs due to the models infinite node nature. For example, cell 1 will have a larger impact on the results of cell 2 than cell 43's results impact on cell 44. This is due to the amount of information being transferred and eventually the old information will overshadow the new inputs. The Long Short Term Memory(LSTM) model is an attempt to fix this issue through adding a variable named "cell state". The LSTM was initially made to revolutionize speech detection and is even used in popular services such as Apple's Siri (REFERENCE). The

Email address: pgr41@msstate.edu (Preston Robertson)

LSTM helps fix the vanishing gradient problem by giving each cell the ability to forget old information. Before the discussion of how the LSTM model forgets, it is important on how the information is translated through out each cell. Each cell has its own output (which is the objective of the artificial neural network) and a cell state that is a recorded value transferred through each cell. This how each cell forgets data. Each cell takes the previous layer's output and the current cell state and compares them. The more these values do not match, the more the cell forgets its cell state. This process is called the forget gate.

The forget gate has had significant impact in the analysis of speech data, which is partially the reason why the accuracy of speech recognition in products such as Amazon's Alexa has increased (REFERENCE). However, the LSTM model is not yet perfect. The forget gate has issues itself. The cell state is a single value propagating through each cell. When the cell state drops to a significantly low value in the forget gate, then essentially all information before is lost. This is an issue due to the pattern nature of data sets. For example, let's say the LSTM model is analyzing stock prices. If the stock price has a significant decline then model will forget the old information and adjust to the new information. However, a researcher may notice the pattern be a seasonal issue (such as the stocks of a swimming wear company). The LSTM model will never find this pattern and will have significant error when the pattern repeats. The second issue with the cell state being a single value, keeps the vanishing gradient problem. Since the third cell will have a larger impact on the cell state value than the impact of the 45th cell in the LSTM model. The objective of the paper is to reduce the vanishing gradient problem through adding more values similar to the cell state.

2. Literature Review

As early as the 1980s, evolutionary algorithms have been applied to optimize artificial neural network (ANN) architectures [1].

2.1. Original LSTM

The first LSTM model was proposed in 1997 as a solution to the RNN models difficulty to handle long term dependencies in data (REFERENCE). The model has found great success in speech recognition data, DBLSTM-HMM by Alex Graves at the University of Toronto (REFERENCE). These resuls are due to ...

2.2. LSTM with the Forget Gate

The original LSTM model initially only added the cell state to the RNN model. This model architecture was very popular in the early 2000s before the invention of the forget gate. then later the forget gate was added by Gers, Schmidhuber, and Cummins in the year 2000 (REFERENCE). The motivation for this addition to the RNN model was due to the over saturation that forms in the model. Due to the inability to forget data, the cell state would not properly update to represent the new data. Let's say that there is an original

LSTM model that predicts where Jim eats lunch. Jim went to his favorite restaurant for the last 5 years, but then it closed down for the winter season. Without the ability to forget, the LSTM model will continue predict Jim will eat at the restaurant despite that not being the correct output value. This is due to the over saturation of the previous data. This over saturation happens in several data sets such as predicting stocks, since there are outside factors that the model can not account for, the model will be wrong about trends. However, with the ability of the model to forget the old data, then the model can adapt to the new trends despite not getting all the information.

2.3. Peephole LSTM

3. Methodology

3.1. Mathematical Model

In the proposed model...

$$\begin{aligned} \max \quad & \hat{R}_l(b_{\cdot l-c}, \dots, b_{\cdot l-1}) && (1) \\ & \sum_{i=1}^{C} b_{il} \leq n, \forall l && (2) \\ & b_{il} \neq b_{kl}, i \in \Theta, k \in \Gamma && (3) \\ & l < B && (4) \\ & b_{il} \in \{0, 1\}, \forall i, l && (5) \end{aligned}$$

where $\hat{R}_l = (1 - \tau)Q + \tau T$. Model accuracy ($Q$) and the measure of convergence trend (T) are used in the objective function in Equation (1), where $b_{\cdot l}$ is the neural architecture vector for the l-th layer, which is a binary vector, c indicates the number of previous layers used to predict the performance of the l-th layer.

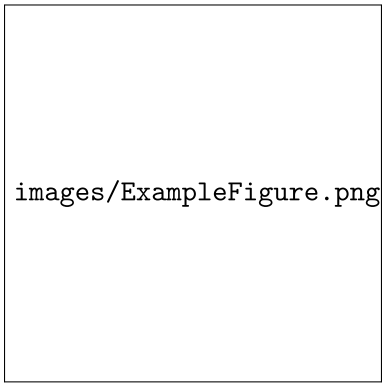


Figure 1: The caption

Cite the Figure 1. Cite Table 1. Cite Algorithm 1.

Table 1: Table caption

Block (i)	1	2	3	4	5	6	7	8
Type	C1	C2	C3	C4	R1	R2	R3	R4
b_{i1}	0	1	1	0	0	0	0	0
b_{i2}	1	0	0	1	0	0	0	0
b_{i3}	0	0	0	0	1	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
b_{in}	0	0	0	0	1	0	0	0

Algorithm 1 Example

```
1:  $B$: Maximum number of layers added; $S$: Neural block space
2:  procedure 1: RNN INITIALIZATION($S, D_e, R(x), c$)
3:      Initialize current model set: $\Omega = \emptyset$
4:      Evaluate CNNs: $\eta_1 \leftarrow M_1(D^e_{Te})$
5:      Train predictor: $R'(x) \leftarrow R(M_1, c; \eta_1)$
6:      Update model set: $\Omega \leftarrow M_1$
7:  procedure 2: ADAPTIVE OPTIMIZER($\Omega, B, S, D_e, R'(x), c$)
8:      for $l = 2, \ldots, B$ do
9:          Generate a model set with $l$ block: $M_l \leftarrow$ ModelGenerator ($\Omega$, $l$, $S$)
10:         Predict performance of CNNs: $\eta^p_l \leftarrow R'(M_l, c)$
11:         Select top $k$ highest performance CNNs: $M^k_l \leftarrow \eta^p_l(M_l)$
12:     return $M^1_B$
```

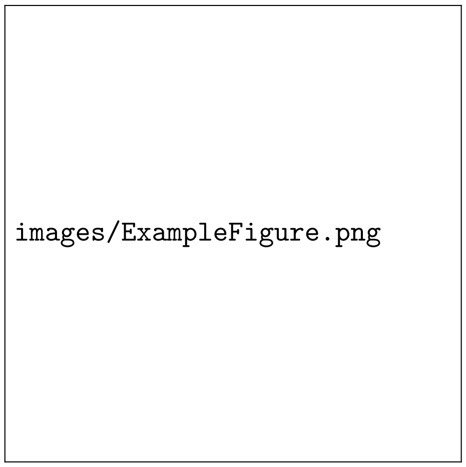
4. Experimental Results

4.1. Data Description

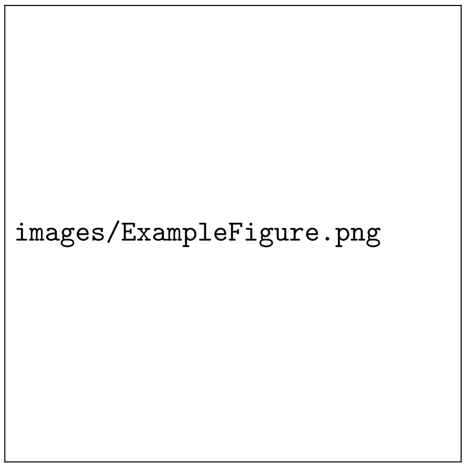
After the data preprocessing, Algorithm **??** is used to extract 3D retinal samples. In this study, α and β are set as 100 and 10, respectively. Each sample has a 50% overlap in each sliding step. Table 2 provides an overview of the statistical properties of the collected retinal samples.

Table 2: An overview of data

Class	Samples	Intensity Distribution			
		Mean	SD	Skewness	Kurtosis
Normal	1,820	78.15	51.65	0.78	0.17
AMD	1,494	64.30	42.08	1.04	1.35
DME	1,764	66.63	45.43	1.01	0.82



(a) Subfig 1



(b) Subfig 2

Figure 2: Example of the figure with sub-figures

5. Conclusions and Future Work

References

[1] G. F. Miller, P. M. Todd, and S. U. Hegde, “Designing neural networks using genetic algorithms.” in *ICGA*, vol. 89, 1989, pp. 379–384.