Revision Number: 002
Date: January 28, 2023
From: Ryan Bugianesi
Subject: LSTM Neural Network Notes

1 Introduction

This document details the design of a Long-Short Term Memory (LSTM) neural network. The design and structure for this network was inspired from videos by Josh Starmer [1] and Ahlad Kunar [2] on YouTube. All diagrams shown in this document are credited to Josh Starmer and are included for visualization.

2 Implementation

The LSTM network is implemented in C++ using an object-oriented approach from a 'bottom up' perspective. The gates within the architecture were designed first and incorporated into more abstract classes.

2.1 Classes

2.1.1 ForgetGate

The ForgetGate class implements the first stage of a LSTM network neuron. This class uses the long-term memory, short-term memory, and input values to determine how much of the long-term memory is propagated to the rest of the architecture. The ForgetGate class accomplishes this by using a sigmoid activation function. After scaling the inputs by the saved biases and weights specific to this class only, the sum of the weighted input, the weighted value of the short-term memory, and the bias are used as the input to a sigmoid function. The result of this calculation represents the percentage of the long-term memory to keep. The long-term memory is then updated via multiplication with the output of the sigmoid function. A value of 1 would represent the propagation of the long term memory value, while 0 would cause the information to be lost, or "forgotten" to the network.

Figure 1, shown below, gives a visualization of the first stage. The green line represents the long term memory propagating through the entire neuron.

Similarly, the red line represents the short-term memory. The input to the neuron is shown in the blue box, and its path is shown by the gray line.

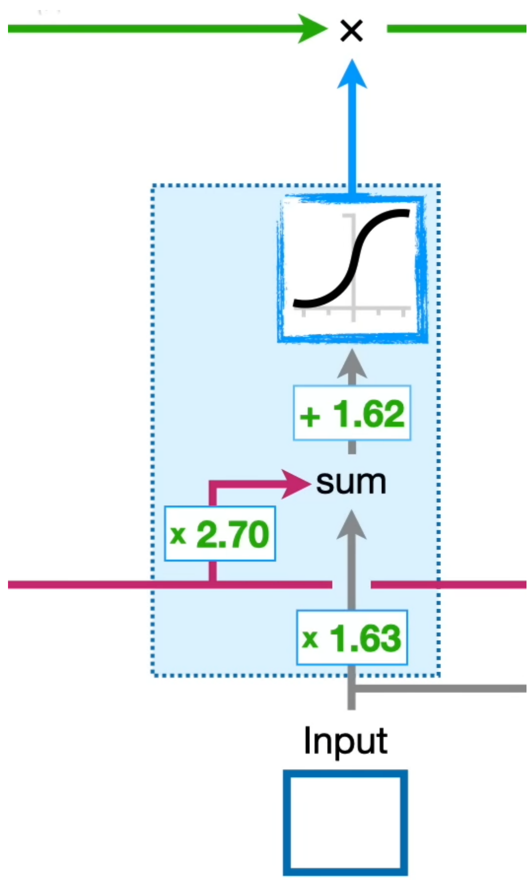

Figure 1 - LSTM Forget Gate

To understand how the data propagates through the network more clearly, it helps to express the operations through math. This also helps later when calculating error. To begin, the input gate's operation can be expressed as:

$$f_t = \sigma(x_t * U_f + h_{t-1} * W_f + b_f) \tag{1}$$

Where x_t is the input to the gate at time t, h_{t-1} is the value of the hidden state, or short-term memory, at time $t-1$, and U_f, W_f, and b_f are the weights and biases of the Forget Gate.

This equation will be used later when training the network for comparing error between expected value and predicted value (f_t).

2.1.2 InputGate

The Input Gate class implements the second function of a LSTM neuron. This class manages updating the long term memory with new information. Again, all three internal values are used to create the new memory. The green line (long term memory) gets an additional value added to it (representing the new information). This additional value is computed by the two blocks shown in Figure 2. The green block represents the calculation the network uses to determine how much of the new information is to be saved. Using the sum of the input value (gray line) and the short term memory value (red line), multiplied by their corresponding weights, the LSTM network calculates the value of the sigmoid function. This calculation returns a value between 0 and 1. This value is the percent of the new long term memory to be added to the existing long term memory.

The orange block uses the same values as the green block, with different weights, to calculate the actual value of the new long term memory. The block has a special name: the **candidate state**, and is discussed in The Architecture section. This calculation uses the tanh function. This function returns a value between -1 and 1, which represents the new information.

After completing the calculations of both blocks, the product of the computed values is added to the long term memory.

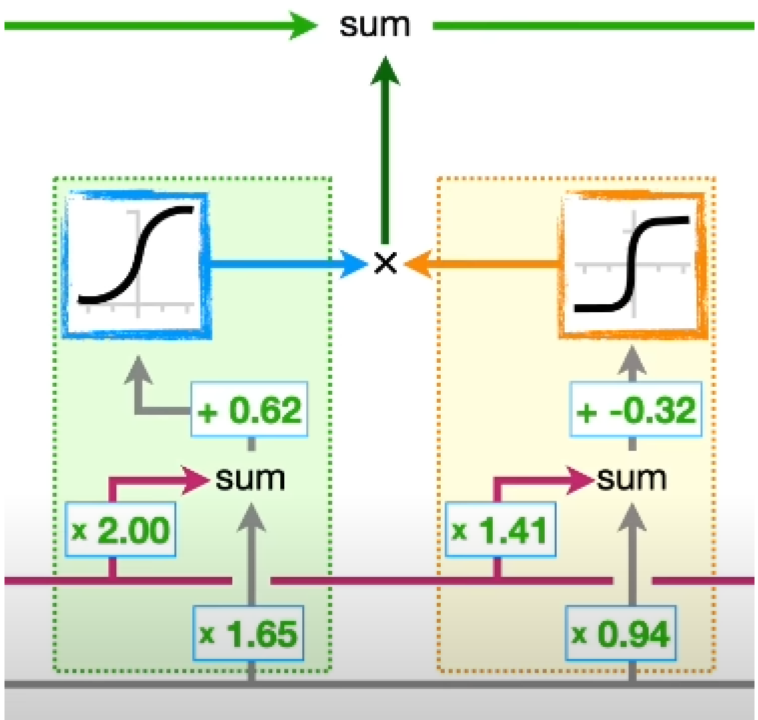

Figure 2 - LSTM Input Gate

The Input Gate's operation can be described by:

$$i_t = \sigma(x_t * U_i + h_{t-1} * W_i + b_i) \tag{2}$$

Where x_t is the input to the gate at time t, h_{t-1} is the value of the hidden state, or short-term memory, at time $t-1$, and U_i, W_i, and b_i are the weights and biases of the Input Gate.

2.1.3 Output Gate

The output gate (shown in *Figure 3* below) is in charge of generating the next short term memory to be passed on as output from an arbitrary neuron. This process is done in the same manner as the long term memory generation in the Input Gate. The inputs to this block of the LSTM neuron are the short term memory value, the input value, and the long term memory value.

The short term memory is used along with the input value and corresponding weights to generate a sum. This sum is added to a bias value and passed to a sigmoid function to normalize the value to the range of 0-1. The output value from the sigmoid function determines how much of the new short term memory is saved.

The long term memory is used to determine the value of the new short term memory. By passing the value of the long term memory generated in the previous stage to a *tanh*() function. The output value of this function is combined with the output of the sigmoid function in the adjacent block. This output value is the new short term memory, and it is an output of neuron.

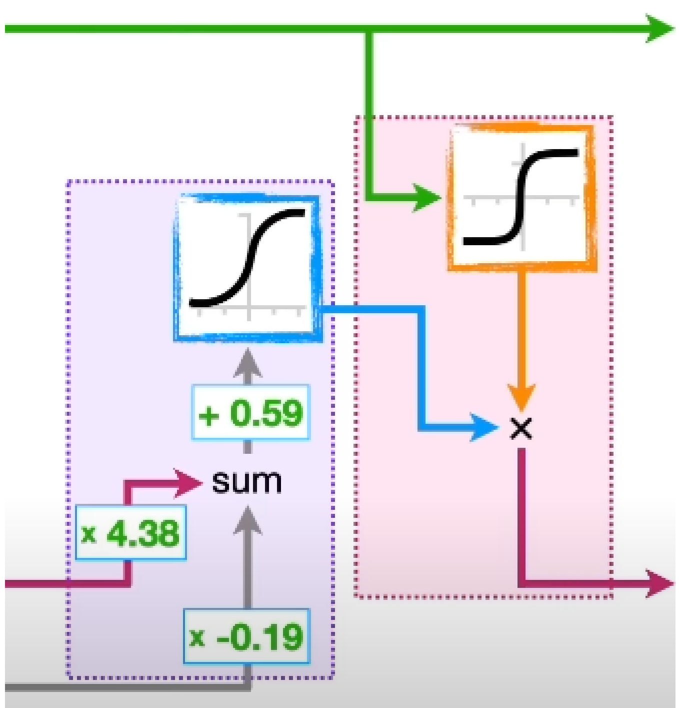

Figure 3 - LSTM Output

The Output Gate's operation can be described by:

$$o_t = \sigma(x_t * U_o + h_{t-1} * W_o + b_o) \tag{3}$$

Where x_t is the input to the gate at time t, h_{t-1} is the value of the hidden state, or short-term memory, at time $t-1$, and U_o, W_o, and b_o are the weights and biases of the Output Gate.

2.2 Architecture

The data is propagated through the network of gates by calculating the outputs of each gate and performing operations on the internal short and long term "memory" (a.k.a the hidden state and cell state respectively). The operations performed on these internal values can be represented by the following equations:

$$C_t = f_t * C_{t-1} + i_t * g_t \tag{4}$$

$$H_t = o_t * tanh(C_t) \tag{5}$$

Where C_t and H_t are the new values of the short and long term memory (Cell and Hidden states) and g_t is the value of the candidate state:

$$g_t = tanh(x_t * U_g + h_{t-1} * W_o + b_g) \tag{6}$$

The output of the network at a time t can be checked by performing the calculation for H_t and is often filtered through a softmax function to normalize the output to a probability distribution when the network is built using multidimensional data (i.e. $x_t \in \mathbb{R}^n$ where $n > 1$).

3 Conclusion

The LSTM neural network aims to fix the problem of vanishing gradients present in basic and recurrent neural networks by only allowing incremental updates to the recurrent state.