Implementation of Recurrent Neural Networks

Lab Assignment 5

Sumedha Janani Siriyapuraju
Dept. of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur, India
sumedhasjs@gmail.com

***Abstract*—A recurrent neural network (RNN) is a neural network that has weighted connections inside each layer. RNNs may store data while processing new input since they have loops. They are excellent for processing tasks requiring consideration of previous inputs because of their memory (such as time-series data). Because of this, modern deep-learning learning networks are built using RNNs.**

***Index Terms*—Recurrent Neural Networks, Forward Pass, Backward Pass, Loss Function, Hidden State**

I. Introduction

An artificial neural network that employs sequential data or time series data is known as a recurrent neural network (RNN). These deep learning algorithms are included into well-known programs like Siri, voice search, and Google Translate. They are frequently employed for ordinal or temporal issues, such as language translation, natural language processing (nlp), speech recognition, and image captioning. Recurrent neural networks (RNNs) use training data to learn, just like feedforward and convolutional neural networks (CNNs) do. They stand out due to their "memory," which allows them to affect the current input and output by using data from previous inputs. Recurrent neural networks' outputs are dependent on the previous parts in the sequence, unlike typical deep neural networks, which presume that inputs and outputs are independent of one another.

II. Method

- The function for a single forward pass cell which calculates output for a single timestamp was implemented. This returns the output for the next timestamp, output for the current timestamp and cache consisting of the timestamp input and output parameters
- The RNN forward pass is implemented by cascading the single cell several times and storing each parameter for each cell for backpropagation steps later
- Backpropagation function is used to compute the derivatives with respect to the cost in order to update the parameters.

Formula for calculating current state:

$$h_t \ = \ f(h_{t-1}, x_t)$$

where
h_t is the current state,
h_{t-1} is the previous state and,
x_t is the input state

Loss Function: In the case of a recurrent neural network, the loss function **L** of all time steps is defined based on the loss at every time step as follows:

$$L(\hat{y}, y) \ = \ \sum_{t=1}^{T_y} L(\hat{y}^{<t>}, y^{<t>})$$

Back propagation through time: Back propagation is done at each point in time. At time step **T**, the derivative of the loss **L** with respect to weight matrix **W** is expressed as follows:

$$\frac{\delta L^{(T)}}{\delta W} \ = \ \sum_{t=1}^{T} \frac{\delta L^{(T)}}{\delta W} \ for \ all \ t$$

III. Discussion

- The basic functions for RNN forward and backward passes were written and implemented.
- RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.
- Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.
- These functions can be later used to train a model along with other functions like gradient clipping function or activation function to solve exploding and vanishing gradient problems and other optimization functions

IV. Conclusion

A function for performing the operations of forward and backward propagation of a Recurrent Neural Network has been successfully implemented.

Appendix (Code)

the code is available here