NLP UNIT 2 EVALUATION :
BIDIRECTIONAL RNN BASED SORTING MODEL

Problem Statement:

Given a list of numbers, the model should output the sorted form of the input.

Procedure:

1) **Dataset Creation:**

An input text file with 10,000 sequences of lists with arbitrary lengths ranging from 2 to 10 and numbers in the range from 0 to 31 was generated randomly. Our training data consisted 8000 sequences and test data consisted of 2000 sequences.

2) **Model :**

For modelling the bidirectional RNN, we used the vanilla RNN code by the author Andrej Karpathy, and modified it to work like Bi-directional RNN.

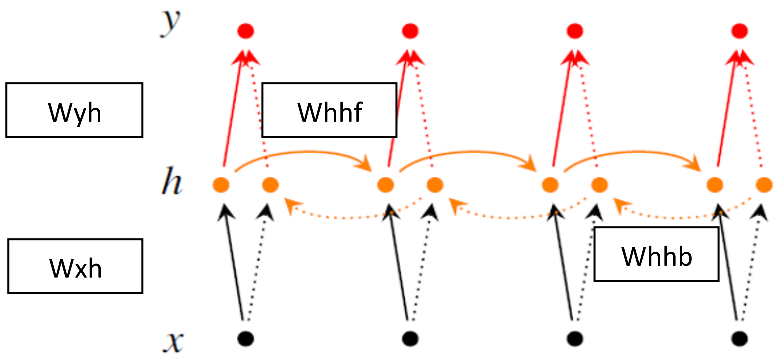

Fig1. Bidirectional RNN model

In Fig1:
1. x represents the input vector of 4 dimensions
2. h represents the hidden layer with 4 hidden units
3. y represents the output layer of 4 dimensions

Hyperparameters:

alpha : The rate at which the model is learnt, we tried it with is 0.001,0.005,1e-1.
Number of hidden units : Tried with 8,16,24
Size of each input unit vector: 32

Parameters:

There are six parameters:
- *Wxh :* Weights between input layer and hidden layer
- *Whhf :* Weights between hidden layer and hidden layer for forward pass
- *Whhb :* Weights between hidden layer and hidden layer for reverse forward pass
- *Why :* Weights between hidden layer and output layer.
- *bh :* n bias units for n hidden units.
- *by :* n bias units for n output units

Functions:

Input : One- hot vector whose size is number of sequence of numbers
Hidden (Forward): tanh function with (Wxh.x + Whh.h(prev) + bh) as its parameter
Hidden (Backward): tanh function with (Wxh.x + Whhb.h(next) + bh) as its parameter
Output: Softmax function with (Wyh.[h(forward);h(backward)]+by) as its parameter.
Contains probability distribution among 32 sequences.

3) **Procedure :**

1. For each list of numbers, we passed 0 to n numbers from the list at nth time step, and also, the target ,ie, sorted numbers till that time step is passed as target numbers.
   That is, if the sequence is [6,1,3,5,2]:
   At 1st time step, we passed [6] as input and [6] as target,
   At 2nd time step, we passed [6,1] as input and [1,6] as target,
   At time step 3, we passed [6,1,3] as input and [1,3,6] as target.
   This goes on till the time reaches the size of the list.
   So, the number of input units is not constant. It varies with respect to the length of the sequence.
2. For each such sequence, one-hot vectors were created for each number in each input unit, forward and backward propagations were applied and weights were updated till they converged.
3. After the weights were updated, this model was used to test data for new sequence of lists of numbers.

4) **Building the model:**

In the forward propagation,
1. h(forward) is evaluated from left to right
2. h(backward) is evaluated from right to left
3. Output, o is evaluated (we evaluated it from left to right).

Error propagation in backward propagation:
1. Errors were propagated from output to hidden using first order differential function of softmax.
2. Errors were propagated from hidden to input from right to left, using first order differential function of tanh.
3. Errors were propagated from hidden to input from left to right, using first order differential function of tanh.

Weights were updated.

Process continues till loss gets minimized beyond a threshold.

Results and discussions:

NO. OF HIDDEN UNITS	**Accuracy (%) with Learning rate = 0.001**	**Accuracy (%) with Learning rate = 0.005**	**Accuracy (%) with Learning rate = 1e-1 (1.73)**
8	18.75	18.80	19.04
16	19.11	19.38	19.06
24	19.14	18.84	18.84
32	18.89	18.98	18.78

The bidirectional RNN with 16 hidden units gave better accuracy than 8/16/32 hidden units.
It did not give better results with alpha being 1e-1.The accuracy reduces from 24 hidden units to 32 hidden units, because the number of weights and bias to be found must be less than the number of the training points, else, the neural network will be over trained and will have low generalization capacity.

Challenges faced:

1. Initially, the model had so many errors with respect to dimensions. Dimensions were analyzed at every layer and resolved.
2. Converting the code from RNN to Bidirectional RNN was challenging. The concepts of RNN had to be applied in both the directions from input to hidden layer and the results from forward and backward directions had to be combined and sent to output layers. This took time, but we could implement it.
3. Initially, the probability distributions were uniform over the entire 32 unit vector. We took time to realize that it gradually assigns weights as it learns the model.
4. Initially, the loss remained constant over the number of epochs. It was because the weight matrices were not getting updated, so this problem was resolved.
5. Even after updating the matrices, the change in loss after every epoch is very negligible.