Nisa Ahsen Öztürk

22001858

EEE 443 Mini Project 2: Human Activity Recognition (HAR)

1. Introduction

The aim of this project is to build a human activity recognition (HAR) algorithm by using the recurrent neural network (RNN) structure. For the training algorithm the given dataset is used. Dataset contains samples from 6 different motion types. Dataset is created by using the signals of three different motion sensors. Moreover, dataset contains time series of train and test data with length of 150 units. The most significant advantage of RNN systems is to increase the accuracy rate for sequential data. The network contains one hidden layer with N number of neurons. Output layer has 6 neurons and error function is calculated by using cross entropy formula. Tanh function is used as the activation function of the hidden layer. Similarly, sigmoid function is used for the output function. The structure of the RNN circuit is given below.

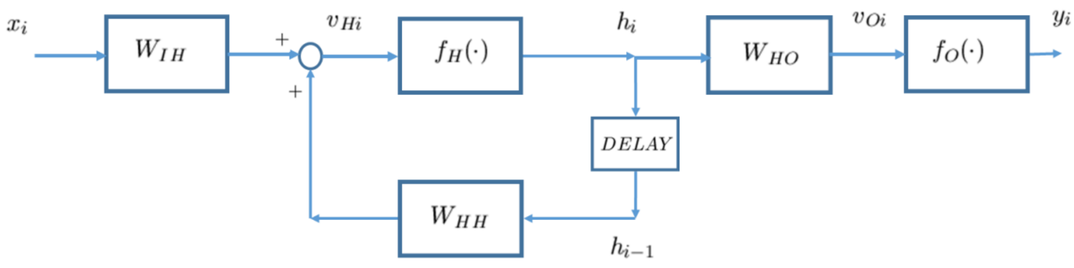

Figure 1: Structure of RNN network

2. Theory and Algorithm

2.1 Forward Propagation

Since the algorithm is recurrent output of the forward propagation is depended on the former values of the output of hidden layer denoted by h_{i-1}. One can write the forward propagation as follows.

$$v_{Hi} = x_i * W_{IH}{}^T + h_{i-1} * W_{hh}{}^T \qquad Eq.1$$

$$h_i = f_H(v_{Hi}) \qquad Eq.2$$

$$v_{Oi} = h_i * W_{HH} \qquad Eq.3$$

$$y_i = f_O(v_{Oi}) \qquad Eq.4$$

2.2 Back Propagation

The formulas for gradient descent and back propagation through time (BPTT) is given below. Firstly, start with gradient of W_{HO}

$$W_{HO} \leftarrow \ W_{HO} + \eta \frac{\partial E}{\partial W_{HO}} \qquad Eq.5$$

Where

$$\frac{\partial E}{\partial W_{HO}} = (y - d)^T * h \qquad Eq.6$$

Second part of the BPTT is to find the gradient for the W_{HH}.

$$\frac{\partial E}{\partial W_{HH}} = \sum_i \frac{\partial E_i}{\partial W_{HH}} \qquad Eq.7$$

$$\frac{\partial E_i}{\partial W_{HH}} = \sum \frac{\partial E}{\partial y_i} \ \frac{\partial y_i}{\partial h_i} \ \frac{\partial h_i}{\partial h_k} \ \frac{\partial h_k}{\partial W_{HH}}$$

$$= (y_n - d_n) W_{HO} (1 - h_n^2)[h_{n-1} + W_{HH}(1 - h_{n-1}^2) h_{n-2} \dots] \quad Eq.8$$

Note that cross entropy is used for error function.

Therefore,

$$E = -d_n \log y_n - (1 - d_n) \log(1 - y_n) \quad Eq.9$$

One can find gradient for W_{IH} similarly.

$$\frac{\partial E}{\partial W_{IH}} = \sum_i \frac{\partial E_i}{\partial W_{IH}} \qquad Eq.10$$

$$\frac{\partial E_i}{\partial W_{IH}} = \sum \frac{\partial E}{\partial y_i} \ \frac{\partial y_i}{\partial h_i} \ \frac{\partial h_i}{\partial h_k} \ \frac{\partial h_k}{\partial W_{IH}}$$

$$= (y_n - d_n) W_{HO} (1 - h_n^2)[h_{n-1} + W_{HH}(1 - h_{n-1}^2) h_{n-2} \dots] \quad Eq.11$$

3. Implementation and Results

To implement the project, dataset is read in the code. After that, dataset is shuffled since it is a uniform dataset. Shuffling can increase the reliability of the dataset. All parameters are initialized and forward function is written as theorical forward propagation.

While implementing the back propagation part and training network, I encountered with several overflow warnings and division by zero error. Those overflows and error, is led to "nan" outcomes in the output of the network. In order to eliminate this error, clipping is used in BPTT algorithm. However, this implementation could not increase the accuracy of the training network.

To train and test the network 8 different cases are observed. Accuracy is observed by using top1,2 and 3 accuracy. Top 2 accuracy is the percentage that the correct output is among the top 2 maximum outputs.

Note that the accuracy of the network without any training is 16.666%

Training	Top 1 Accuracy(%)	Top 2 Accuracy(%)	Top 3 Accuracy(%)	Misclassify

N=50
$\eta = 0.05$
Batch= 30
	37.12	79.3	86.60	653

N=50
$\eta = 0.05$
Batch= 10
	28.66	82.35	84.74	843

N=50
$\eta = 0.1$
Batch= 30
	32.21	77.83	84.5	755

N=50
$\eta = 0.1$
Batch= 10
	24.2	64.90	81.36	998

N=100
$\eta = 0.05$
Batch= 30
	37.80	69.10	74.33	639

N=100
$\eta = 0.05$
Batch= 10
	22.66	46.16	66.16	1066

N=100
$\eta = 0.1$
Batch= 30
	14.52	36.83	71.39	1663

N=100
$\eta = 0.1$
Batch= 10
	12.22	36.41	69.43	1877

Table 1: Training algorithm outcomes

Test	Top 1 Accuracy	Top 3 Accuracy	Top 3 Accuracy	Misclassify

N=50
$\eta = 0.05$
Batch= 30
	28.8	62.36	66.42	166

N=50
$\eta = 0.05$
Batch= 10
	20.0	58.6	68.92	240

N=50
$\eta = 0.1$
Batch= 30
	16.66	50.33	55.50	300

N=50
$\eta = 0.1$
Batch= 10
	24.2	64.90	72.33	198

N=100
$\eta = 0.05$
Batch= 30
	21.58	54.66	64.76	222

N=100
$\eta = 0.05$
Batch= 10
	12.66	34.50	50.21	379

N=100
$\eta = 0.1$
Batch= 30
	12.66	38.3	50.66	379

N=100
$\eta = 0.1$
Batch= 10
	18.73	32.0	62.85	256

Table 2: Test set outcomes

From the Table 1 and 2, one can conclude that the best case occurs when N=50, η=0.05 and batch size is 30. In terms of learning rate, data response as expected since the lower learning rate gives more accurate results. On the other hand, as the neuron number increases, accuracy of the networks decreases. This result is an unexpected behavior. This behavior can be derived

from the overfitting or rapid increase (explosion) in gradients. The clipping did not fix the accuracy problem in this case. Clipping is useful to eliminate the divergence problem, however, accuracy could not be increased by this method. Therefore, the results and experiment support that overfitting may be occurred during the implementation.
For all cases, top 1,2 and 3 accuracy results were parallel with the theorical expectations. Since the top 3 accuracy choses among 3 maximum outputs, the correct prediction rate was higher in each case.

4. Conclusion

The aim of this project was to implement a RNN to identify 6 different motion types. Three different sensors' signals generate the dataset as a sequence. The project is beneficial since it helps to understand the BPTT algorithm. Gradients are derived by using BPTT and training is realized by using Python. The best case occurred when N=50, η=0.05 and batch size is 30. This result can indicate that there is an overfitting in the training process. Because in theory, more accurate results should be observed when neuron number is larger. Therefore, it can be said that implementation process was not entirely accurate. However, it is tested that no training behavior gives 16.66% accuracy. Therefore, it can be said that the training algorithm still trains the network. Therefore, the implementation was successful. The accuracy of the algorithm can be improved.