NAI project

Made by: Hrymailo Kyrylo (s17292)

Topic description: According to given dataset create and train neural network which successfully differentiate letter for being spam or not.

Dataset description: Dataset consist of 4601 records each one containing 57 attributes +1 desired output.

Feature selection: To remove less significant attributes I used **Weka environment** in which I applied ranking attribute evaluation called **Info Gain Attribute Evaluation** more information about it can be found on http://old.opentox.org/dev/documentation/components/infogainattribeval with using ranker method having default threshold=1.79769313486231570E308. I chose 30 top rated attributes out of output.

Testing model: I split randomly given set into two subset in proportion 4:1 for training and testing data correspondingly.

Experiment flow: before applying backpropagation algorithm I did data normalization the way it suits borders [0,1].

To achieve that I applied such formula for each attribute value:

new_value = (old_value – min(value))/(max(value) -min(value)

After that I specify some parameters for my algorithm, that are number of instances I would like to train, number of

epochs, learning rate and size of my hidden layer. After that I do forward propagation through my network, calculate error value of output layer and update weights values, than I move backward and update my hidden layer weights. As my activation function I selected unipolar sigmoid function. These operations I repeat epoch times and return weights set as a result value of my program execution

After I got my trained weight vector I evaluate it with my test set. I initialize 4 variable each of them indicating 4 possible states. They got filled during testing and are used later to create confusion matrix.

Programming software: I implemented this algorithm using **Python 3.6 programming language in Anaconda environment**.

Confusion matrix:

After applying set of experiments on input parameters to my train algorithm I obtained following results:

Case 1

Size of training set : 3681

Size of testing set : 920

Number of epochs: 100

Learning rate: 0.5

Size of hidden layer: 27

	0	1
0	312	91
1	38	479

Total Accuracy = (Value(0,0)+Value(1,1))/size*100% = 85.9782%

Accuracy of class 0 = Value(0,0)/(Value(0,0)+Value(1,0))*100% = 89.1428%

Accuracy of class 1 = Value(1,1)/(Value(1,1)+Value(0,1))*100% = 84.0351%

Class with highest accuracy: Class 0, class with lowest accuracy: Class 1.

Case 2

Size of training set : 3681

Size of testing set : 920

Number of epochs: 500

Learning rate: 0.5

Size of hidden layer: 27

	0	1
0	503	19
1	53	345

Total Accuracy = (Value(0,0)+Value(1,1))/size*100% = 92.1739%

Accuracy of class 1 = Value(0,0)/(Value(0,0)+Value(1,0))*100% = 94.7802%

Accuracy of class 0 = Value(1,1)/(Value(1,1)+Value(0,1))*100% = 90.4676%

Class with highest accuracy: Class 1, class with lowest accuracy: Class 0.

Case 3

Size of training set : 3681

Size of testing set : 920

Number of epochs: 200

Learning rate: 0.3

Size of hidden layer: 27

	0	1
0	473	87
1	75	285

Total Accuracy = (Value(0,0)+Value(1,1))/size*100% = 82.3913%

Accuracy of class 1 = Value(0,0)/(Value(0,0)+Value(1,0))*100% = 76.6129%

Accuracy of class 0 = Value(1,1)/(Value(1,1)+Value(0,1))*100% = 86.3139%

Class with highest accuracy: Class 0, class with lowest accuracy: Class .

Case 4

Size of training set : 3681

Size of testing set : 920

Number of epochs: 200

Learning rate: 0.5

Size of hidden layer: 27

	0	1
0	487	39
1	49	345

Total Accuracy = (Value(0,0)+Value(1,1))/size*100% = 90.4348%

Accuracy of class 0 =
Value(0,0)/(Value(0,0)+Value(1,0))*100% = 89.8437%

Accuracy of class 1 =
Value(1,1)/(Value(1,1)+Value(0,1))*100% = 90.8582%

Class with highest accuracy: Class 1, class with lowest accuracy: Class 0.

Experiment analysis

We can notice that with increasing of epoch actual accuracy got increased. I most of the cases class 0 has more instances that's because in my dataset majority of attributes are actually predicted to be of class 0. We can see clear correlation between learning step, number of epochs and accuracy of analysis.

Conclusion

During execution of my project I have created training model which classifies if email is spam. I did a number of experiment playing around with input parameters and splitting input dataset randomly into training and testing group. Experiments show remarkably good precision in classification although it takes quite huge amount of time to train network on such a large set of data.

Basically current learning algorithm can be applied for any similar group of dataset which has 2 decision classes.