PROJECT REPORT

NEURAL NETWORK IMPLEMENTATION USING BACKPROP ALGORITHM

CSCE-633: MACHINE LEARNING

Submitted By:

Amandeep Singh Bhal

UIN: 724003595

Table of Contents

OBJECTIVE OF THE PROJECT

The aim of this project is to implement a multi-layer neural network, and test it on at least 5 datasets from the UCI Machine Learning Repository. Compare the performance of our Neural Network to our Decision Tree program.

PROJECT IMPLEMENTATION

Main steps involved in implementation (explained in detail in later sections):

- Download dataset from UCI repository and make the control file for the data.
- Control file is preprocessed and data is read according to control file.
- Using Back Propagation algorithm to create neural network based on the passed parameters - number of hidden layers, learning rate.
- Testing is done using **Ten-fold cross validation** to calculate accuracies and confidence Intervals.
- Comparison of results is made with decision tree using confidence interval and T-test.

SCHOLASTIC UPDATES

I have used the gradient descent algorithm to search the hypothesis space of possible weight vectors which best fits the training examples. One of the main advantage of gradient descent algorithm is that it can work for training examples which are not linearly separable.

The weight vector is defined by: $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$

The update to weights for each example is obtained with the help of following rule:

$$\Delta w_i = \eta(t - o)\ x_i$$

This is defined by the gradient descent rule. Here t is the target value, o is the output unit and x_i is the i^{th} example.

MOMENTUM

To speed up the process of learning, the momentum is added to the weight update rule. Introduction of the momentum rate allows the attenuation of oscillations in the gradient descent. Multiplying momentum value to w in formula of w = w + delta(w).

HANDLING DISCRETE ATTRIBUTES

For handling discrete input, we do a preprocessing of the example dataset. If a discrete attribute has “n” values then we represent the attribute as n binary input nodes in the input layer. So if we have k attributes with n values then we have a total of (k)x(n) input nodes the represents each example data.

For example if we have an attribute weather having values sunny, overcast and rain then for an example like “sunny, hot, high, weak, no” the weather attribute is represented as 100, 1 for sunny and 0 for overcast and rain. On similar lines, we can represent other attributes of the example.

HANDLING CONTINUOUS ATTRIBUTES

For handling continuous attribute, we preprocess the example data where the continuous attribute values are normalized. For normalizing the attribute value, we find the minimum as well as the maximum value of that attribute from the example data and then normalize the value using the formula:

$$X_{i,\ 0\ \text{to}\ 1} = \frac{X_i - X_{Min}}{X_{Max} - X_{Min}}$$

THRESHOLD FUNCTION

For threshold function, we need a kind of unit whose output is also a differentiable function of its input. The function used here is the sigmoid unit, a unit very much similar to perceptron, but based on a smooth, differentiable threshold function.

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

STOPPING CRITERIA

We use the training data for training the neural network. After every iteration of the backpropagation algorithm the trained network is validated on the validation set. We find the Mean Square Error(MSE), and keep a copy of the best network created till now. We keep calling the BackProp algorithm as long as the MSE is reducing at every turn. Once we find a minimum after x iterations, we continue to search till 2*x iterations of backpropagation algorithm to see if we can find a set of weights with even lesser MSE on the validation set. There is also a chance that MSE might keep reducing even after a lot of iterations, this is handled by keeping an artificial bound on the number of iterations as well.

CHOICE OF DATASETS

1. Tic-Tac-Toe Data Set
2. Balance Scale Data Set
3. Iris Data Set
4. Japanese Credit Screening Data Set
5. Car Evaluation Data Set

Tic-Tac-Toe Data Set: This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). Interestingly, this raw database gives a stripped-down decision tree algorithm (e.g., ID3) fits. However, the rule-based CN2 algorithm, the simple IB1 instance-based learning algorithm, and the CITRE feature-constructing decision tree algorithm perform well on it.

1. Missing Values: No
2. Attributes: Discrete
3. Number of Instances: 958

Balance Scale Data Set: This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced.

a. Missing Values: No
b. Attributes: Discrete
c. Number of Instances: 625

Iris Data Set: This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

a. Missing Values: No
b. Attributes: Continuous
c. Number of Instances: 150

Japanese Credit Screening Data Set: Examples represent positive and negative instances of people who were and were not granted credit. The theory was generated by talking to the individuals at a Japanese company that grants credit.

a. Missing Values: Yes
b. Attributes: Continuous and Discrete
c. Number of Instances: 690

Car Evaluation Data Set: Car Evaluation Database was derived from a simple hierarchical decision model originally developed for the demonstration of DEX, M. Bohanec, V. Rajkovic: Expert system for decision making. Sistemica 1(1), pp. 145-157, 1990.). The Car Evaluation Database contains examples with the structural information removed, i.e., directly relates CAR to the six input attributes: buying, maint, doors, persons, lug_boot, safety.

a. Missing Values: No
b. Attributes: Discrete
c. Number of Instances: 1728

PREPROCESSING

Creating Control file:

A control file has been created for each data set, where we have following information

1. number_of_input_nodes: number of nodes in the input layer which is calculated as number of attributes x.

2. no_of_attributes: no of attributes for input.

3. is_discrete: a list representing the information of type of each attribute. By type, we mean, discrete or continuous. true: discrete and false : continuous.

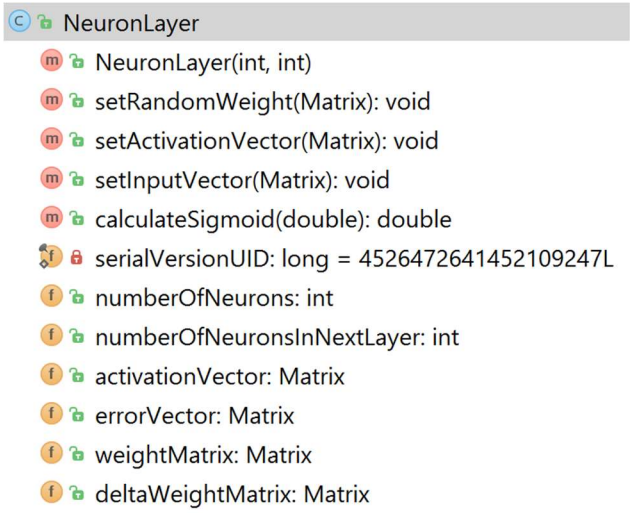
4. number_of_output_nodes: number of nodes in the output layer, which is calculated as number of possible values for output.

5. target_index: index of the target in the example dataset.

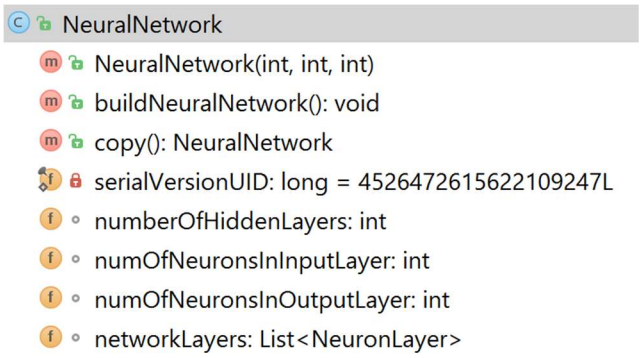
6. A list representing the possible values for each discrete attribute and output. For example : index0= sunny, overcast, rain represents that attribute 1 has three possible values.

CODE/DESIGN

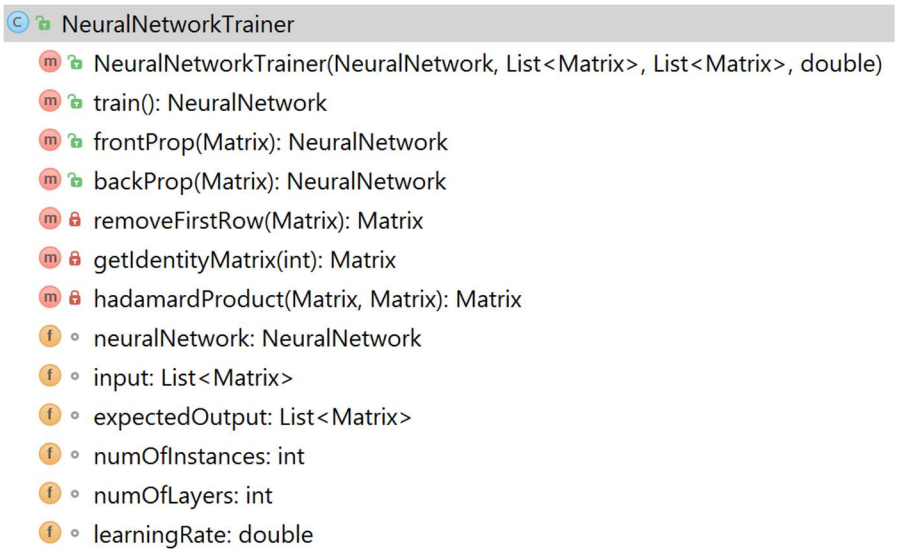
- **NeuronLayer:** Stores the architecture and values of a neural layer. This class have following properties and methods.



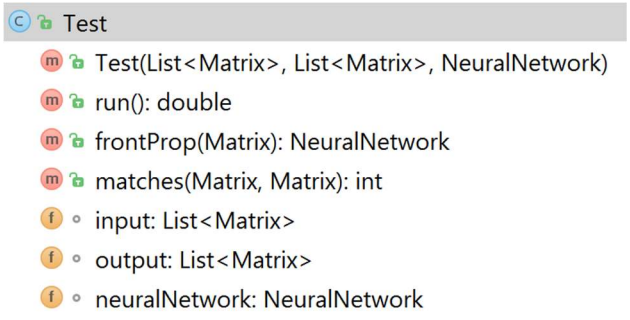
- **NeuralNetwork:** Stores the architecture and properties of neural network. This class have following properties and methods.



- **NeuralNetworkTrainer:** This is class which makes architecture of neural network and uses backward propagation algorithm to get best neural network using training data, validation data and MSE values.



- **Test:** This class is used to test the neural network for accuracy using Test Data.



- **BackProp Algorithm:**

```
public NeuralNetwork backProp(Matrix expectedOutput){
    Matrix output = new Matrix(expectedOutput.getRowDimension()+1,1);
    output.set(0,0,1);
    for(int i=0; i<expectedOutput.getRowDimension(); i++){
        output.set(i+1,0,expectedOutput.get(i,0));
    }
    Matrix Ok = this.neuralNetwork.networkLayers.get(this.neuralNetwork.networkLayers.size()-1).activationVector;
    Matrix I = getIdentityMatrix(Ok.getRowDimension());

    NeuronLayer outputLayer = this.neuralNetwork.networkLayers.get(this.neuralNetwork.networkLayers.size()-1);
    outputLayer.errorVector = hadamardProduct(output.minus(Ok),hadamardProduct(Ok,(I.minus(Ok))));

    //back prop the error
    for(int i=this.neuralNetwork.networkLayers.size()-2; i>=0; i--){
        NeuronLayer currLayer = this.neuralNetwork.networkLayers.get(i);
        NeuronLayer nextLayer = this.neuralNetwork.networkLayers.get(i+1);
        Matrix Oj = currLayer.activationVector;
        Matrix Wjk = currLayer.weightMatrix.transpose();
        Matrix dk = removeFirstRow(nextLayer.errorVector);
        I = getIdentityMatrix(Oj.getRowDimension());

        Matrix a = Wjk.times(dk);
        Matrix c = I.minus(Oj);

        currLayer.errorVector = hadamardProduct(a,hadamardProduct(Oj,c));
    }

    //updating weight matrix for each layer
    for(int i=0; i<this.neuralNetwork.networkLayers.size()-1; i++){
        NeuronLayer currLayer = this.neuralNetwork.networkLayers.get(i);
        NeuronLayer nextLayer = this.neuralNetwork.networkLayers.get(i+1);
        currLayer.deltaWeightMatrix = nextLayer.errorVector.times(currLayer.activationVector.transpose()).times(this.learningRate);
        currLayer.deltaWeightMatrix = currLayer.deltaWeightMatrix.getMatrix(1,currLayer.deltaWeightMatrix.getRowDimension()-1,0,
                currLayer.deltaWeightMatrix.getColumnDimension()-1);
        currLayer.weightMatrix = currLayer.weightMatrix.plus(currLayer.deltaWeightMatrix);
    }

    return this.neuralNetwork;
}
```

- **Ten-Fold Cross Validation:**

  For testing, we have implemented Ten-fold cross-validation. The initial dataset is first randomized and then it is divided into 3 groups: 10% for test data, 60% for training data, 30% validation data. The network is trained using the training data, then it is validated against the validation set to check if more iterations of the backpropagation algorithm is required. Finally, it is tested against the Test data to calculate various parameters such as accuracy, mean-squared error and confidence interval. This process is repeated for 10 iterations to cover every 10% of the example data set.

RESULTS

- Accuracy Table

Data Set	Mean Accuracy	Num of Layers	Num of Hidden Nodes	Learning Rate	Iterations (for convergence)	MSE	Confidence Interval on Accuracy
Car Evaluation	98.67	4	21	0.4	175	0.005	[98.3442, 99.0073]

Japanese Credit	98.57	2	47	0.1	60	0.0017	[95.5492, 99.5937]
Iris	96.36	3	4	0.5	150	0.0185	[93.7490, 98.9903]
Tic Tac Toe	98.36	4	27	0.25	350	0.0127	[97.7722, 98.9613]
Balance Scale	96.67	4	20	0.25	300	0.019	[95.9126, 97.4391]

Weight Matrix for Iris dataset during one iteration run is as follows:
-0.0342414451 0.0429871062 0.0379387636 -0.0004123741
-0.0073848444 -0.0014382822 0.0415744438 -0.0119272290
0.0114378247 0.0285414602 0.0157190535 0.0217535806

- Iris Dataset

	0 Hidden Layer	1 Hidden Layer	2 Hidden Layer
Accuracy	92.78	96.21	91.50
Confidence Interval	[86.8371, 98.7402]	[93.4318, 98.9907]	[88.0467, 94.9632]

- Weight Balance Dataset

	0 Hidden Layer	1 Hidden Layer	2 Hidden Layer
Accuracy	92.33	94.88	96.67
Confidence Interval	[90.4733, 94.1944]	[93.4188, 96.3588]	[95.9126, 97.4391]

Effect of Changing the number of nodes in Hidden Layer in Weight Balance Dataset:

	5 Nodes	10 Nodes	15 Nodes
Accuracy	98.3704	97.5353	97.4213
Confidence Interval	[97.8925, 99.5678]	[96.7052, 99.1678]	[95.6579, 98.6278]

T-TEST

A paired T-test is implemented to determine whether any of the decision tree and neural network algorithms perform statistically different. We compare the performance of Decision Tree algorithm and Neural network algorithm below.

Dataset	Decision Tree	Neural Network
Car Evaluation	[90.65, 90.94]	[98.3442, 99.0073]
Japanese Credit	[84.80,86.06]	[95.5492, 99.5937]
Iris	[92.33, 93.66]	[93.7490, 98.9903]
Balance Scale	[46.54, 48.96]	[95.9126, 97.4391]
Tic Tac Toe	[79.79, 82.75]	[97.7722, 98.9613]

CONCLUSION

I have used BackProp algorithm to train neural network along with gradient descent algorithm to find the best weight vectors from hypothesis space that best approximates the training examples. I have trained network on 5 datasets; Tic-Tac-Toe, Balance Scale, Iris, Japanese Credit and Car Evaluation. I have tested the network using Ten-Fold cross validation and also compared the results with the decision tree accuracy on the same datasets, using T-test. It is observed that Neural network performs better than ID3 algorithm however the time taken in algorithm execution is more. Neural Network also performs better in case of datasets having noisier data.

Neural network is therefore more proficient to give the better classification by using nonlinear boundaries. In addition, decision tree is not good for online learning since any data includes some exceptional situation will force the tree to be fall apart and need to be constructed again. On the other side, Neural Network is capable of reflecting the information of new instance on a model very efficiently by just changing the weight values.

The major factor that impacted the performance of the neural network was the number of hidden layers, momentum value and learning rate constant. For two of the datasets, Iris and weight balance, we have changed the hidden layer to see the performance. As we see, with the increase in number of hidden layers, there is a slight improvement in the performance of the network but using the more hidden layers led to slow convergence of the network.

With the increase in learning rate, the convergence was faster but as we increase it further, it will lead to overfitting of the data. So we maintain a constant learning rate of 0.5 to obtain reasonable results.

With the increase in the number of hidden layers there is a slight improvement in the performance but the time to convergence increased. The best result was achieved with 2 hidden layers with a comparatively faster convergence rate. However, in Iris dataset accuracy for 2 hidden layers decreases slightly as compared to 1 hidden layer, this can be due to overfitting.

The best configuration of the neural network depends on the best data type. For example, in weight balance dataset, the best configuration was a network having 2 hidden layer, 20 hidden nodes, and 0.25 learning constant.

EXAMPLE OUTPUT

10 CROSS VALIDATION

Accuracy is 95.51612903225814

Iteration ends !!

Accuracy is 91.96774193548384

Iteration ends !!

Accuracy is 94.54838709677414

Iteration ends !!

Accuracy is 96.13121185701807

Iteration ends !!

Accuracy is 96.61290322580643

Iteration ends !!

Accuracy is 97.86017221436803

Iteration ends !!

Accuracy is 97.53887675106

Iteration ends !!

Accuracy is 91.68487340958698

Iteration ends !!

Accuracy is 93.52268345970934

Iteration ends !!

Accuracy is 93.50502948317745

Iteration ends !!

Accuracy is 99.36254980079681

Iteration ends !!

----------SUMMARY----------

Mean Accuracy: 94.88880084652423

MSE 0.03790496799822286

Confidence Interval: [93.4188, 96.3588]