Project 2 – Neural Network

Submitted by – Shantanu Bansal

**Algorithm**

In this project we implement a multi-layer neural network using the Backpropagation algorithm. Backpropagation is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function. It is a generalization of the delta rule to multi-layered feedforward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Backpropagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.

**Activation Function**

Although there exist several functions such as Unit Step, Gaussian, Piecewise Linear etc that can be used as an activation function in the neural network, we have used the Sigmoidal function whose output value varies between 0 and 1 and is given by:

Sigmoid(y) =

**Stochastic Updates to Network Weights**

In this implementation we stochastically update the network by feedforwarding a single input and then backpropagating the error gradient before sending the next input to the network. In a single epoch the entire training dataset is sent exactly once through the network with the weights being updated after each instance of input data.

for (IOArray data : trainingData) {

ANN.feedForward(network, data.getInput());

ANN.backPropagate(network, data.getOutput(), learningRate);

}

**Handling of Discrete Inputs**

All the discrete values in the dataset have been mapped to binary values. For any attribute having k distinct values, the implementation generates k attributes and assigns them values 0 or 1 depending on the value in the original input dataset.

For example:

In the Voting Dataset, the attribute ‘religious-groups-in-schools’ has 2 possible values y & n. This attribute in all instances will be replaced by 2 attributes A & B each having 2 possible choices 1 & 0 and this value is decided by the actual value in the dataset. religious-groups-in-schools = y translates to A = 1 & B = 0 and religious-groups-in-schools = n translates to A = 0 & B = 1.

**Normalization of Continuous Values**

Continuous values are normalized using feature scaling that reduces them to a range of 0.0 – 1.0 using the formula:

Normalize(x) =

Here min(attribute(x)) and max(attribute(x)) return the minimum and maximum values respectively, associated with the attribute of input x.

**Stopping Criterion**

90% of the entire dataset is set aside to build the neural network out of which 70% is used to train the network and the remaining 30% is used to validate it and calculate its mean square error (MSE). After every iteration of training the network with the training set, the validation set is used to calculate the MSE and a copy of the network weights is created if the mean square error of the network in current iteration is lower than that of the previous iteration. If the MSE begins to rise then, to ensure that we did not achieve a local minima, we continue to train the network for twice the number of iterations while monitoring the MSE and updating the best network weights, if a lower MSE is achieved during the process. An upper bound on the number of iterations has been established as 10000 since for these datasets if the MSE does not converge within these 10000 iterations, we can adjust the learning rate to achieve a faster convergence. Also, for some of the datasets the MSE begins to increase initially in an untrained network. To resolve this issue a lower bound of 100 iterations has been fixed which ensures that the network is atleast partially trained before we declare as having achieved a minima.

**Datasets Used**

All datasets were downloaded from the UCI Machine Learning Repository

1. **Congressional Voting Records (**[**https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records**](https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records)**)**:

This dataset includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes. The decision value classifies the record as either belonging to a democrat or a republican. All the attributes in this dataset are discrete.

**Number of Attributes -> 16**

**Attribute Characteristics -> Categorical**

**Number of Instances -> 435**

**Missing Values -> Yes**

1. **Iris (**[**https://archive.ics.uci.edu/ml/datasets/Iris**](https://archive.ics.uci.edu/ml/datasets/Iris)**)**:

This dataset includes details of 3 discrete classes of iris plant described by 4 continuous parameters.

**Number of Attributes -> 4**

**Attribute Characteristics -> Real**

**Number of Instances -> 150**

**Missing Values -> No**

1. **Tic-Tac-Toe Endgame (**[**https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame**](https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame)**)**:

This dataset 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". All the attributes in this dataset are discrete.

**Number of Attributes -> 9**

**Attribute Characteristics -> Categorical**

**Number of Instances -> 958**

**Missing Values -> No**

1. **Banknote Authentication (**[**https://archive.ics.uci.edu/ml/datasets/banknote+authentication**](https://archive.ics.uci.edu/ml/datasets/banknote+authentication)**):** The dataset was generated from images that were taken from genuine and forged banknote-like specimens. Based on 4 continuous values the dataset classifies each record as genuine or forged.

**Number of Attributes -> 5**

**Attribute Characteristics -> Real**

**Number of Instances -> 1372**

**Missing Values -> No**

1. **Credit Approval (**[**https://archive.ics.uci.edu/ml/datasets/Credit+Approval**](https://archive.ics.uci.edu/ml/datasets/Credit+Approval)**):**

This dataset concerns credit card applications and classifies each record as approved (+) or rejected (-) on basis of several continuous and discrete values.

**Number of Attributes -> 15**

**Attribute Characteristics -> Categorical, Integer, Real**

**Number of Instances -> 690**

**Missing Values -> Yes**

**Results**

**Table 1:** Best Results for the datasets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameters | Voting Records Dataset | Iris Dataset | Tictactoe Dataset | Banknotes Dataset | Credit Dataset |
| # of Layers | 0 | 1 | 2 | 2 | 1 |
| # of nodes | - | 4 | 4 | 16 | 2 |
| Learning Rate | 0.1 | 0.05 | 0.1 | 0.5 | 0.1 |
| Epochs to Convergence | 106 | 928 | 493 | 187 | 72 |
| Accuracy | 96.74% | 97.33% | 98.53% | 99.71% | 87.10% |
| Standard Deviation | 2.37% | 4.42% | 0.97% | 0.67% | 4.37% |
| Standard Error | 0.75% | 1.40% | 0.31% | 0.21% | 1.38% |
| Confidence Interval | 96.74 ± 1.67 | 97.33 ± 3.12 | 98.53 ± 0.69 | 99.71 ± 0.47 | 87.1 ± 3.08 |

The table above shows the best version of the algorithm and the mean accuracy and confidence interval achieved using it. For some of the datasets the same accuracy was achieved for a different configuration. There are two criteria that can be employed to pick the best network:

1. Pick the network that gives the best accuracy in minimum epochs
2. Pick the network that gives the best accuracy and uses the minimum parameters (weights)

**Table 2:** Comparison of performance of networks with 0, 1 and 2 hidden layers and 2, 4, 8 and 16 hidden nodes on two different datasets

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # of hidden layers | | 0 | 1 | | | | 2 | | | |
| # of nodes | | - | 2 | 4 | 8 | 16 | 2 | 4 | 8 | 16 |
| Iris DataSet | Learning Rate | 0.2 | 0.01 | 0.05 | 0.05 | 0.2 | 0.2 | 0.05 | 0.2 | 0.2 |
| # of Parameters | 15 | 19 | 35 | 67 | 131 | 25 | 55 | 139 | 403 |
| Epochs to Convergence | 9369 | 5668 | 928 | 1577 | 473 | 1423 | 1372 | 928 | 583 |
| Accuracy | 96.67% | 97.33% | 97.33% | 97.33% | 96.00% | 96.00% | 97.33% | 96.67% | 97.33% |
| Standard Deviation | 3.34% | 4.42% | 4.42% | 5.33% | 4.42% | 4.42% | 3.27% | 4.47% | 3.27% |
| Standard Error | 1.06% | 1.40% | 1.40% | 1.69% | 1.40% | 1.40% | 1.03% | 1.41% | 1.03% |
| Confidence Interval | 96.67 ± 2.36 | 97.33 ± 3.12 | 97.33 ± 3.12 | 97.33 ± 3.77 | 96.00 ± 3.12 | 96.00 ± 3.12 | 97.33 ± 2.30 | 96.67 ± 3.14 | 97.33 ± 2.30 |
| Voting Records DataSet | Learning Rate | 0.1 | 0.05 | 0.2 | 0.05 | 0.2 | 0.2 | 0.1 | 0.1 | 0.5 |
| # of Parameters | 33 | 69 | 137 | 273 | 545 | 75 | 157 | 345 | 817 |
| Epochs to Convergence | 106 | 1436 | 217 | 453 | 148 | 6082 | 4410 | 882 | 1326 |
| Accuracy | 96.74% | 93.02% | 95.81% | 95.81% | 96.51% | 81.63% | 77.67% | 70.23% | 68.60% |
| Standard Deviation | 2.37% | 10.96% | 2.90% | 3.25% | 3.64% | 16.91% | 17.53% | 14.85% | 13.33% |
| Standard Error | 0.75% | 3.47% | 0.92% | 1.03% | 1.15% | 5.35% | 5.54% | 4.70% | 4.22% |
| Confidence Interval | 96.74 ± 1.67 | 93.02 ± 7.74 | 95.81 ± 2.05 | 95.81 ± 2.3 | 96.51 ± 2.56 | 81.63 ± 11.93 | 77.67 ± 12.35 | 70.23 ± 10.48 | 68.6 ± 9.41 |

**Comparison results**

In the Iris dataset, the network takes a long time to converge (9369 epochs) if it is configured without using any hidden layers, even though the learning rate has been kept quiet high (0.2). As we add 1 hidden layer, we start getting a better performance from the network. As can be seen from the table above, the best performance is achieved when the network has 1 hidden layer and 4 nodes in that layer. In this configuration we achieve the best accuracy with a fairly low learning rate and small number of iterations compared to other configurations. Another advantage of this configuration is the low count of parameters which signifies the overall low time consumption of the network. As we add another hidden layer to this network, we can see that we achieve a similar accuracy but at a cost of increased learning rate with no reduction in iterations to achieve convergence. This dataset is not linearly seperable enough to be classified properly by a network with no hidden layers without much training. However, this data is also not extremely complex and as a result, adding a second hidden layer does not give any significant advantage.

In the voting dataset, the best performance is achieved by using a network that has no hidden layers. This shows that the data is almost linearly seperable and does not require a complex network to achieve good accuracy of prediction. As we add a single hidden layer, we achieve comparable accuracy but at the cost of increased network complexity in terms of parameters. Also, it can be seen in the configuration of 1 hidden layer with 16 hidden nodes that we have to increase the learning rate by 100% in order to achieve a comparable accuracy within a reasonable convergence time. This further reduces the necessity of a complex network such as this. Finally, as we add another hidden layer to the network we can see the overall performance of the network deteriorating.

**Table 3:** Comparison of performance of Neural Network (Backpropagation) with Decision Tree (ID3 with Pessimistic Pruning)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Parameters | Iris Dataset | | Voting Records Dataset | | Banknotes Dataset | | Tictactoe Dataset | | Credit Dataset | |
|  | Neural Net | Decision Tree | Neural Net | Decision Tree | Neural Net | Decision Tree | Neural Net | Decision Tree | Neural Net | Decision Tree |
| Accuracy | 97.33% | 95.33% | 96.74% | 82.09% | 99.71% | 98.61% | 98.53% | 69.05% | 87.10% | 75.80% |
| Standard Deviation | 3.27% | 4.27% | 2.37% | 11.77% | 0.67% | 1.28% | 0.97% | 8.71% | 4.37% | 6.12% |
| Standard Error | 1.03% | 1.35% | 0.75% | 3.72% | 0.21% | 0.40% | 0.31% | 2.75% | 1.38% | 1.94% |
| Confidence Interval | 97.33 ± 2.30 | 95.33 ± 3.01 | 96.74 ± 1.67 | 82.09 ± 8.30 | 99.71 ± 0.47 | 98.61 ± 0.89 | 98.53 ± 0.69 | 69.05 ± 6.13 | 87.1 ± 3.08 | 75.80 ± 4.33 |

The table above compares the performance of the implemented neural network with that of the decision tree implemented in the previous project. A glance at the table shows that neural network performs better that the decision tree for all the datasets but to show if this improvement is statistically significant we conduct the pooled T-Test on the two results.

**Pooled T-Test**

Accuracy\_Difference, d = accuracy(method 1) – accuracy(method 2)

Standard Error, SE =

Here, s1 & s2 are standard deviations and n1 and n2 are sample sizes of method 1 and method 2 respectively

t = d/SE

If t < 2.23 then we can accept hypothesis H0 with 95% probability, that method 1 is **not** significantly better than method 2.

Else, we can reject hypothesis H0 and accept alternate hypothesis with 95% probability, that method 1 is significantly better than method 2.

**Iris Dataset**

d = 97.33 – 95.33 = 2

SE = 1.70

t = 1.18 < 2.23

Hence, with 95% confidence we can say that the neural network is not statistically significantly better than the decision tree for this dataset, even though it gives a better accuracy.

**Voting Records Dataset**

d = 96.74 – 82.09 = 14.65

SE = 3.80

t = 3.86 > 2.23

Hence, with 95% confidence we can say that the neural network is statistically significantly better than the decision tree for this dataset.

**Banknotes Dataset**

d = 99.71 – 98.61 = 1.10

SE = 0.46

t = 2.39 > 2.23

Hence, with 95% confidence we can say that the neural network is statistically significantly better than the decision tree for this dataset.

**Tictactoe Dataset**

d = 98.53 – 69.05 = 29.48

SE = 2.77

t = 10.64 > 2.23

Hence, with 95% confidence we can say that the neural network is statistically significantly better than the decision tree for this dataset.

**Credit Dataset**

d = 87.10 – 75.80 = 11.30

SE = 2.38

t = 4.75 > 2.23

Hence, with 95% confidence we can say that the neural network is statistically significantly better than the decision tree for this dataset.

**Conclusion**

* The neural network produces better accuracy than the decision tree in all the cases. However, only for the iris dataset the accuracy of the neural network is not statistically significantly better than that of the decision tree. Neural networks are better at learning a problem since the use of a validation set during backpropagation trains the network better to avoid overfitting and perform better in unforeseen conditions.
* In all the datasets chosen there was no significant improvement in accuracy when adding more hidden layers. In the Iris dataset it takes longer for the weights to converge without any hidden layers, but once a single hidden layer is added the convergence time is reduced significantly. In some of the datasets, particularly the voting records dataset, the convergence time increased by adding more layers to the network. There were a few cases where in order to achieve quicker convergence, the learning rate had to be increased significantly, even close to 1.
* In some of the datasets the mean square error went up initially which is why it was essential to make sure that the network is trained for some minimum number of iterations.
* In most of the cases, the most optimal configuration had 1 hidden layer and varying number of hidden nodes. It is quite likely that the chosen datasets were not complex enough to reap the benefits of a complex network.