# CSCE 478/878 (Fall 2016) Homework 2

Baofeng Zhou, Feiyu Zhu

October 17, 2016

## 1 Problem 1

### 1.1

The true error is

$$error_v(h) = \frac{10}{65} = \frac{2}{13} \tag{1}$$

With 90% probability, $error_v D(h)$ lies in

$$error_v(h) \pm 1.64\sqrt{\frac{error_v(h)(1 - error_v(h))}{N}} \tag{2}$$

$$= \frac{2}{13} \pm 1.64\sqrt{\frac{\frac{2}{13} * (1 - \frac{2}{13})}{65}} \tag{3}$$

Thus the two-sided confidence interval for the true error rate is $error_D(h) \in [0.080, 0.227]$.

### 1.2

For one-sided confidence interval, $z_c^{'} = z_{100 - \frac{100-c}{2}}$, thus $z_{95}^{'} = z_{90} = 1.64$. With 95% probability, $error_v D(h)$ lies in

$$error_v(h) \pm 1.64\sqrt{\frac{error_v(h)(1 - error_v(h))}{N}} \tag{4}$$

$$= \frac{2}{13} \pm 1.64\sqrt{\frac{\frac{2}{13} * (1 - \frac{2}{13})}{65}} \tag{5}$$

So the one-sided confidence interval for the true error rate is $error_D(h) \in (-\infty, 0.227]$ or $error_D(h) \in [0.080, +\infty)$.

### 1.3

For one-sided confidence interval of 90%, $z_c^{'} = z_{100-\frac{100-c}{2}}$, thus $z_{90}^{'} = z_{80} = 1.28$. With 90% probability, $error_v D(h)$ lies in

$$error_v(h) \pm 1.28\sqrt{\frac{error_v(h)(1-error_v(h))}{N}} \tag{6}$$

$$=\frac{2}{13} \pm 1.28\sqrt{\frac{\frac{2}{13} * (1-\frac{2}{13})}{65}} \tag{7}$$

So the one-sided confidence interval for the true error rate is $error_D(h) \in (-\infty, 0.211]$ or $error_D(h) \in [0.096, +\infty)$.

## 2 Problem 2

As can be known from backpropagation algorithm, the error terms of predictions in each layer can be computed as follows:

$$\delta_d^t = y_d^t(1-y_d^t)(r_d^t - y_d^t) \tag{8}$$

$$\delta_c^t = y_c^t(1-y_c^t)(\delta_d^t w_{dc}^t) \tag{9}$$

The weight updates are calculated as follows:

$$w_{dc}^{t+1} = w_{dc}^t + \eta \delta_d^t y_c^t \tag{10}$$

$$w_{ca}^{t+1} = w_{ca}^t + \eta \delta_c^t a \tag{11}$$

$$w_{cb}^{t+1} = w_{cb}^t + \eta \delta_c^t b \tag{12}$$

We executed backpropagation algorithm by hand. The final weights after two full passes of backpropagation can be found in Table 1:

| $\eta$ | 0.5 | | |
|---|---|---|---|
| | **Trial 1** | **Trial 2** | |
| $w_{ca}$ | 0.2 | 0.2024637 | 0.2024637 |
| $w_{cb}$ | 0.2 | 0.2 | 0.1960247 |
| $w_{c0}$ | 0.2 | 0.2024637 | 0.1984879 |
| $a$ | 1 | 0 | |
| $b$ | 0 | 1 | |
| $const$ | 1 | 1 | |
| $sum_c$ | 0.4 | 0.4024637 | |
| $y_c$ | 0.5986877 | 0.5992794 | |
| $w_{dc}$ | 0.2 | 0.2306952 | 0.1876884 |
| $w_{d0}$ | 0.2 | 0.2512709 | 0.1795067 |
| $sum_d$ | 0.3197375 | 0.3895218 | |
| $y_d$ | 0.5792603 | 0.5961675 | |
| $target : r_d^t$ | 1 | 0 | |
| $\delta_d$ | 0.1025418 | -0.1435284 | |
| $\delta_c$ | 0.0049274 | -0.0079515 | |

Table 1: Backpropagation process in Problem 2

# 3 Problem 3

## 3.1

From basic logic knowledge we can derive the truth table of $A \bigwedge \neg B$ as shown in Table 2:

| A | B | -B | $A \bigwedge \neg B$ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

Table 2: Truth table of $A \bigwedge \neg B$

Considering the decision surface, for this linearly separable problem we can simply set the linera threshold as $\frac{1}{2}$. For output larger than $\frac{1}{2}$ we make the prediction as 1; for output smaller than $\frac{1}{2}$ we predict as 0. Then we can get a

group of conditions from a single-layer, two-input perceptron:

$$w_0 < \frac{1}{2} \tag{13}$$

$$w_0 + x_B w_B < \frac{1}{2} \tag{14}$$

$$w_0 + x_A w_A > \frac{1}{2} \tag{15}$$

$$w_0 + x_A w_A + x_B w_B < \frac{1}{2} \tag{16}$$

Intuitively, we can calculate the value $w_0 = -1$, $w_A = \frac{5}{3}$, $w_B = -\frac{2}{3}$. Then we plot the ANN in Figure 1 with denoted weights.

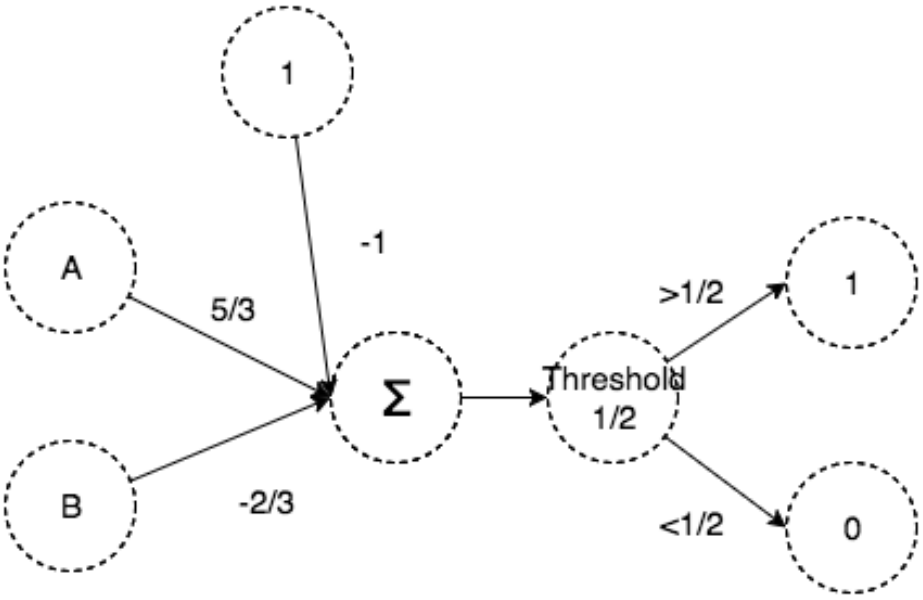


Figure 1: The perceptron that implements $A \bigwedge \neg B$

### 3.2

Table 3 represents the truth table of $[A \bigoplus B] \bigoplus C$.

| A | B | $A \bigoplus B$ | C | $[A \bigoplus B] \bigoplus C$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |

Table 3: Truth table of $[A \bigoplus B] \bigoplus C$

We attempt to design a two-layer perceptron with one hidden layer and one output layer to remap all the vectors made up by A, B and C such that the classes are linearly separable. So firstly we assume there are three units in the first hidden layer. The idea is to use the first and second units two solve the $(A\ XOR\ B)$ and then leverage the third unit to transfer C to the next layer. Thus we set the weights $w_{A3}, w_{B3}, w_{C1}, w_{C2}$ to be zero to eliminate their effects on corresponding units.

The activation function of this perceptron is a linear threshold. If the output of the output layer is larger than 0, then the output of the perceptron will be 1 and vice versa. The structure and each weight of the perceptron in this feedforword neural network is shown in Figure 2.

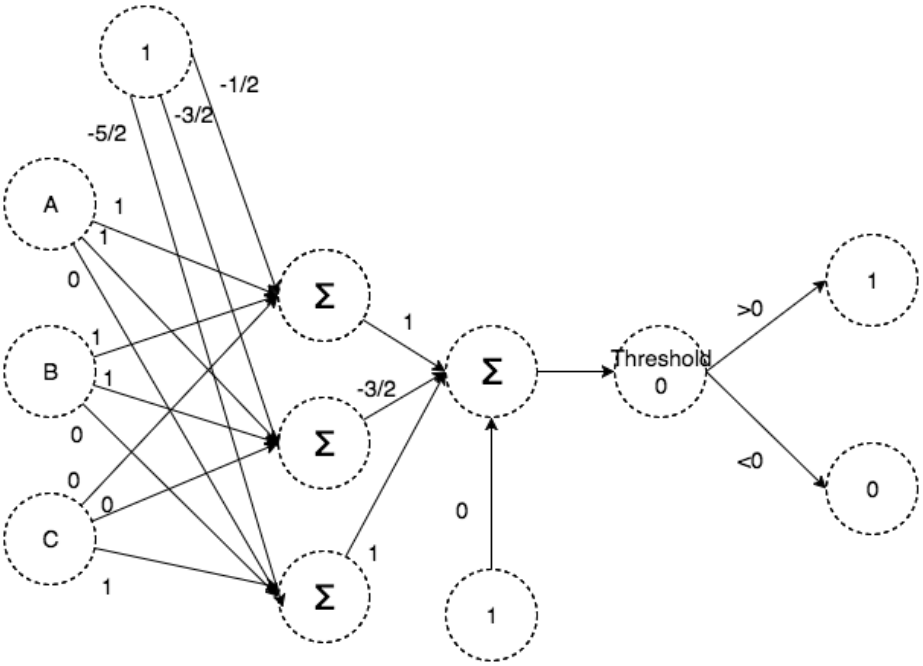


Figure 2: The multiple-layer perceptron that implements $[A \bigoplus B] \bigoplus C$

# 4 Problem 4

## 4.1 Introduction

In this experiment, we will implement the ANN algorithm. We will use back-propagation method to update the weights of the network. We will test the neural network under different parameters and compare the results. We will compare the performance of the designed network with our previous ID3 algorithm implementation by evaluating the corresponding error rates of these algorithms on the three datasets chosen for our experiment.

The report is structured as following sections: Section 4.2 discusses about how we preprocess the data to feed in the neural network; Detailed design of the structure of the neural network is claimed in section 4.3; Results from the experiments are shown in Section 4.4; In Section 4.5 the performance of our ANN algorithm is compared with our previous ID3 algorithm; Finally we give our conclusion in Section 4.6

## 4.2 Data preprocessing

The original data in the datasets are in different forms and have different numbers of classes, so we have to do preprocessing on the datasets to make sure the inputs to our designed neural network are numeric values.

### 4.2.1 Monks I

This dataset contains only numeric values, so no preprocessing is needed for this dataset.

### 4.2.2 Votes

This dataset contains attribute values, yes, no and $'?'$. We assigned 1 to yes, 0 to no and 0.5 to $'?'$. There are two classes, democrat and republican. We assigned 0 to democrat and 1 to republican.

### 4.2.3 Wine

This dataset contains all continuous numeric values. The values for every attribute were discretized into three intervals, $(-\infty, a], (a, b]$, and $(b, \infty)$, where a is the 33th percentile and b is the 67th percentile. We assigned 0 to values in interval $(-\infty, a]$, 1 to values in interval $(a, b]$, and 2 to values in interval $(b, \infty)$. There are three classes in this dataset, named 1, 2, and 3. We assigned 0 to class 1, 1 to class 2, and 2 to class 3.

## 4.3 Experimental setup

We have used C++ as our programming language. We will first test the characteristics of the neural network, then compare neural network with ID3 algorithm.

### 4.3.1 Dataset partitioning

We will randomly choose half of the instances in the dataset as training set, and the remaining instances will be used as test set. In order to generate the sets randomly, we will first shuffle the dataset by exchanging the position of each instance with another instance at a random position, then we will take the first half of the shuffled dataset as training set and the rest as test set. We will use the training set to train the neural network and using the testing set to get the error rate.

### 4.3.2 Design of neural network

**Activation function:** We will use sigmoid function as our activation function. Sigmoid function:

$$\delta(x) = 1/(1 + exp(-x))$$

Derivative of sigmoid function:

$$\delta'(x) = \delta'(x)(1 - \delta'(x))$$

**Number of hidden nodes:** The number of nodes in the hidden layer will obviously affects the test results. Increasing the number of hidden nodes may make the error rate become 0 or make it larger. We will start by just using one hidden node, then increase the number of hidden nodes until it reaches 5 or the error rate becomes 0. We can find out if a certain number of hidden nodes can lead to better results.

**Number of output nodes:** The number of output nodes in the neural network is dependent on the number of classes in the dataset. For example, in datasets Monks I and Votes, the number of classes are 2, so there should be 2 output nodes. In dataset Wine, as there are three classes, there will be 3 output nodes.

**Stop criterion:** We will stop the program after 100000 times of iteration. In every iteration, every instance in the training set will be feed into the neural network, then the weights will be updated on-line using back-propagation method.

**Confidence interval:** We will evaluate the 95% confidence interval for the error rates obtained by changing the number of hidden nodes.

### 4.3.3 Comparison with ID3

We will compare the performance of neural network with that of ID3. We will use sigmoid function as the activation function. We will keep the learning rate at 0.5 and the number of hidden nodes will be 3.

We will use 10-fold cross validation and t-test to evaluate the test results. In 10-fold cross validation, we will randomly shuffle the original dataset and separate the shuffled dataset into 10 parts. We will do 10 tests and in every test, one part of these 10 parts is used as test set while the rest 9 parts were used as training set. After the error rates of these 10 tests have been acquired, the t- test will be used to compare the performance of ANN and ID3.

## 4.4 Test Results

### 4.4.1 Test of neural network

In the tests, we will use 0.5 as the learning rate, and repeat the experiments with different number of hidden nodes from 1 to 5.

**Dataset Monks I** The following Figure 3 and Table. 4 show the error rate of the prediction when we implement different number of nodes in the hidden layer on dataset Monks I.

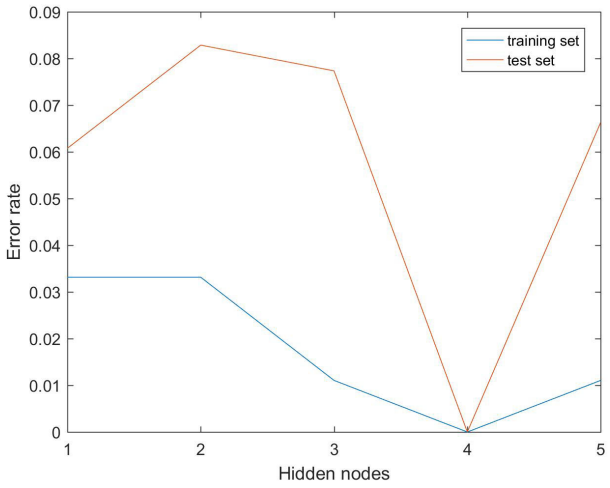


Figure 3: Test results of dataset Monks I

| Number of hidden nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Training error | 0.0331492 | 0.0331492 | 0.0110497 | 0 | 0.0110497 |
| Test error | 0.0607735 | 0.0828729 | 0.0773481 | 0 | 0.0662983 |

Table 4: Test results of dataset Monks I

**Dataset Votes** The following Figure 4 and Table. 5 show the error rate of the prediction when we implement different number of nodes in the hidden layer on dataset Votes.

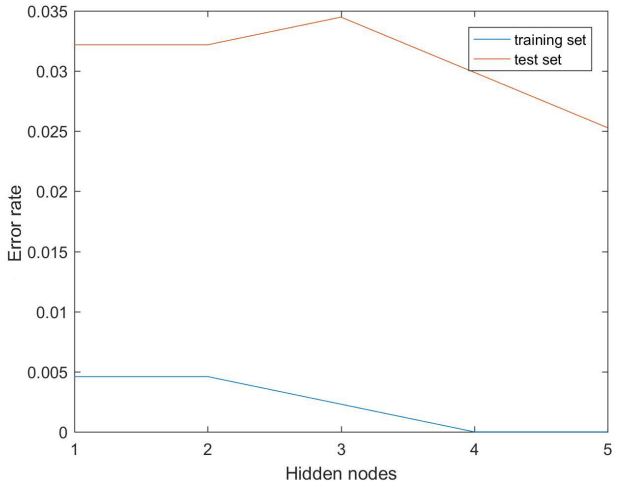


Figure 4: Test results of dataset Votes

| Number of hidden nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Training error | 0.0045977 | 0.0045977 | 0.00229885 | 0 | 0 |
| Test error | 0.0321839 | 0.0321839 | 0.0344828 | 0.0298851 | 0.0252874 |

Table 5: Test results of dataset Votes

**Dataset Wine** The following Figure 5 and Table. 6 show the error rate of the prediction when we implement different number of nodes in the hidden layer on dataset Wine.

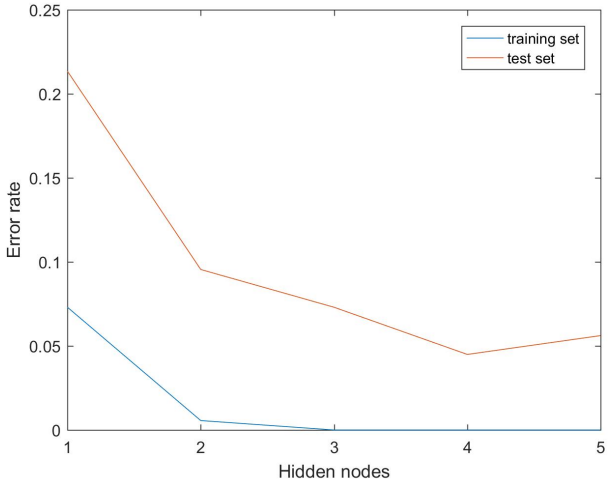


Figure 5: Test results of dataset Wine

| Number of hidden nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Training error | 0.0730337 | 0.00561798 | 0 | 0 | 0 |
| Test error | 0.213483 | 0.0955056 | 0.0730337 | 0.0449438 | 0.0561798 |

Table 6: Test results of dataset Wine

**Summary of error analysis** From our results, we can see that the error rate can be 0 for the training sets of the three datasets. However, the error rate of test sets can not be 0, except the error rate of the test set of dataset Monks I when the number of hidden nodes is 4. We can see that increasing the number of hidden nodes seems to reduce the error rates of test sets, but the error rates of training sets remain the same after the number of hidden nodes has reached a certain values (3 or 4). More hiddens nodes don't necessarily lead to better results, the error rates increased when the number of hidden nodes increased from 4 to 5 for dataset Monks I. There will be overfitting if the number of hidden nodes keeps increasing when a smaller number of hidden nodes can already make the error rates very small. Overfitting will exist if the number of hidden nodes is larger than 4 for dataset Monks I, 4 for dataset Votes, and 3 for dataset Wine.

### 4.4.2 Confidence interval analysis

Confidence interval is another statistic to estimate the distribution or the error. Two-sided confidence interval of each dataset are computed and listed in the following tables.

**Dataset Monks I** The number of instances in this dataset is N=181. We obtain the following Figure 6 and Table. 7 to illustrate the confidence interval.

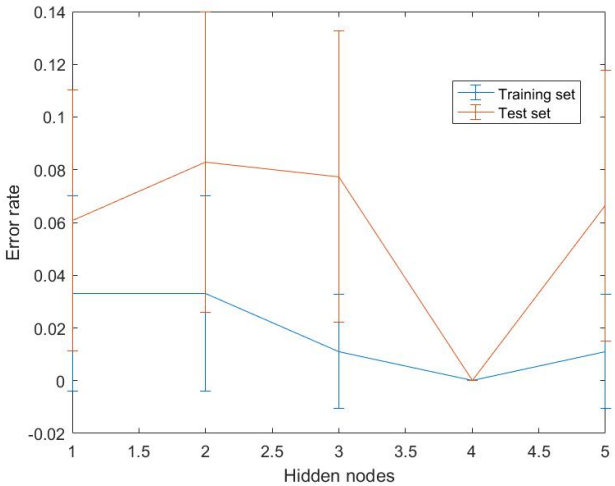


Figure 6: Confidence Intervals of dataset Monks I

| Number of hidden nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Training confidence interval | 0.0141 | 0.0141 | 0.0099 | 0 | 0 |
| Test confidence interval | 0.0367 | 0.0367 | 0.0379 | 0.0354 | 0.0326 |

Table 7: Confidence Intervals of dataset Monks I

**Dataset Votes** The number of instances in this dataset is N=435. We obtain the following Figure 7 and Table. 8 to illustrate the confidence interval.

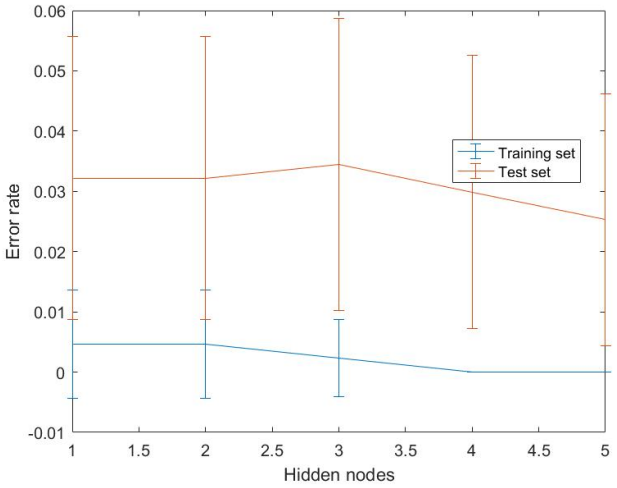


Figure 7: Confidence Intervals of dataset Votes

| Number of hidden nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Training confidence interval | 0.0141 | 0.0141 | 0.0099 | 0 | 0 |
| Test confidence interval | 0.0367 | 0.0367 | 0.0379 | 0.0354 | 0.0326 |

Table 8: Confidence Intervals of dataset Votes

**Dataset Wine** The number of instances in this dataset is N=179. We obtain the following Figure 8 and Table. 9 to illustrate the confidence interval.

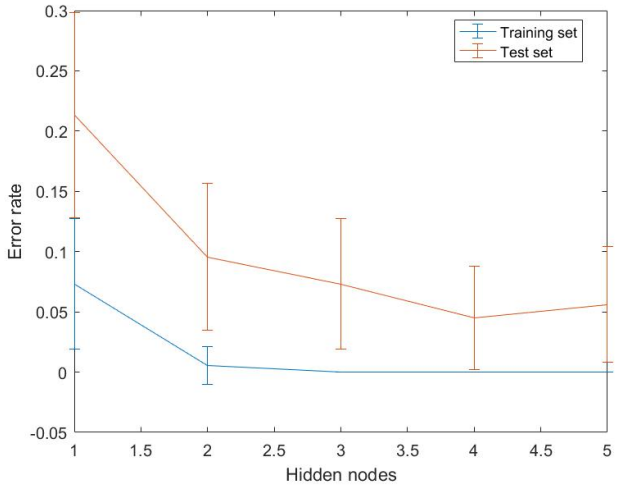


Figure 8: Confidence Intervals of dataset Wine

| Number of hidden nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Training confidence interval | 0.0541 | 0.0155 | 0 | 0 | 0 |
| Test confidence interval | 0.0851 | 0.0611 | 0.0541 | 0.043 | 0.0478 |

Table 9: Confidence Intervals of dataset Wine

#### 4.4.3 ROC curve

As there are 2 or more output nodes, and each node has an output. We will just use the output of node 0 as the scores of the output. The #H in the legend of Figures in this section represents the number of hidden nodes.

**Dataset Monks I** Figure 9 shows the ROC curve of the ANN on dataset Monks I:

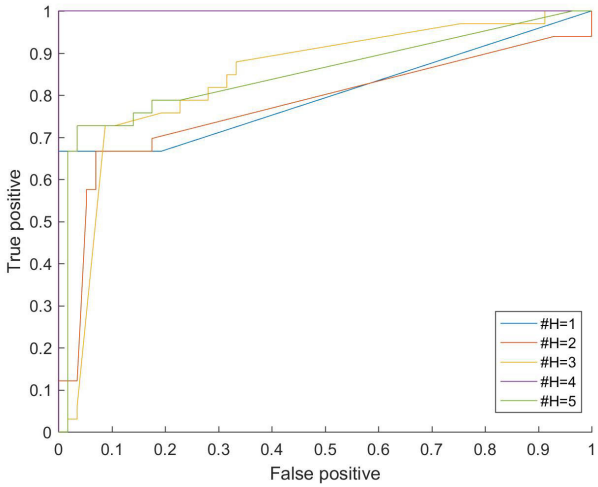


Figure 9: ROC curves of dataset Monk I

**Dataset Votes** Figure 10 shows the ROC curve of the ANN on dataset Votes:

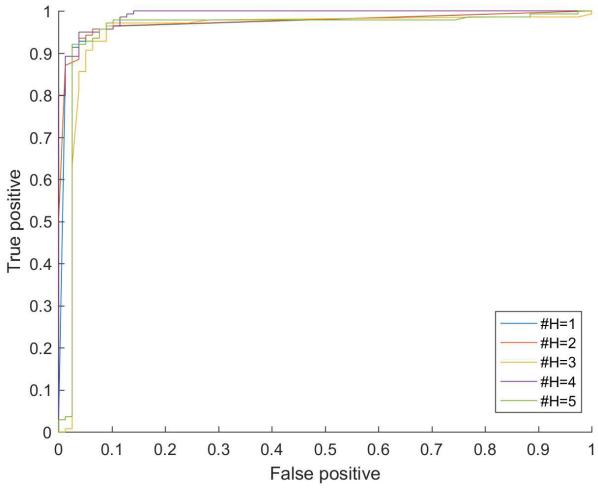


Figure 10: ROC curves of dataset Votes

**Dataset Wine** Figure 11 shows the ROC curve of the ANN on dataset Wine:

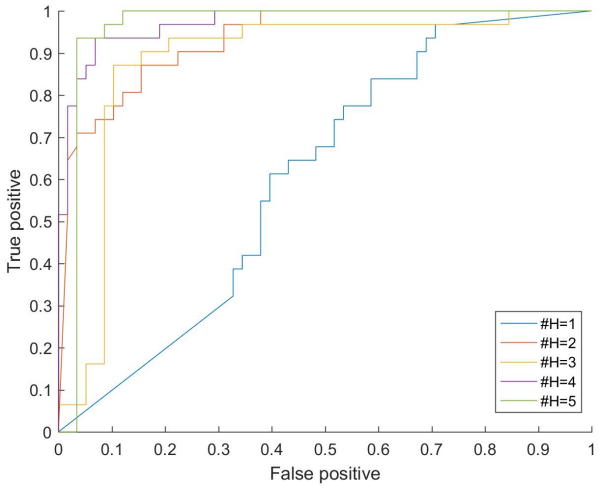


Figure 11: ROC curves of dataset Wine

In the aboved figures, we can see the classification performance our ANN algorithm with different hidden nodes.

## 4.5 Comparison with ID3 algorithm

In order to compare the results of our experiments on backpropagation algorithm with our previous results on ID3 algorithm, we adopted K-fold cross-validation to statistically compare the prediction accuracy.

### 4.5.1 Dataset Monks I

**Neural Network** The 10-fold cross-validation results of our ANN algorithm is shown as follows:

| Fold | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 0.0555556 | 0.111111 | 0.0555556 | 0.111111 | 0.0555556 | 0.0555556 | 0.166667 | 0.0555556 | 0.222222 | 0.111111 |

Table 10: 10-fold error of dataset Monks1

The average error rate is:

$$E_1 = 0.1000 \tag{17}$$

**ID3** The 10-fold validation result of ID3 algorithm is shown as follows:

| Fold | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1111 | 0 |

Table 11: 10-fold error of dataset Monks2

The average error rate from ID3 is:

$$E_2 = 0.0110 \tag{18}$$

Comparing ID3 and ANN algorithms, the error difference estimate between ID3 and ANN is:

$$P = -0.0889 \tag{19}$$

The parameters used in t-test are listed below:

$$S_p = 0.0123 \tag{20}$$

$$T_c = 1.833 \tag{21}$$

$$P + T_c S_p < 0 \tag{22}$$

There is 95% confidence that ID3 is better than ANN.

#### 4.5.2 Dataset Votes

**Neural network** The 10-fold cross-validation results of ANN algorithm on dataset Votes is shown as follows:

| Fold | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 0.116279 | 0.0697674 | 0.0465116 | 0.0232558 | 0.0697674 | 0.0697674 | 0.139535 | 0.0930233 | 0.0697674 | 0.0465116 |

Table 12: 10-fold error of dataset Votes

The average error rate is

$$E_1 = 0.0740 \tag{23}$$

**ID3** The 10-fold validation result of ID3 algorithm on dataset Votes is shown as follows:

| Fold | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 0.0455 | 0.1364 | 0.0227 | 0.0227 | 0.0909 | 0.0455 | 0.0227 | 0.0227 | 0.0455 | 0.1136 |

Table 13: 10-fold error of dataset Votes

The average error rate from ID3 is

$$E_2 = 0.0568 \tag{24}$$

Comparing ID3 and ANN algorithms, the error difference estimate between ID3 and ANN is:

$$P = -0.0176 \tag{25}$$

The parameters used in t-test are listed below:

$$S_p = 0.0188 \tag{26}$$

$$T_c = 0.883 \tag{27}$$

$$P + T_c S_p < 0$$

There is 80% confidence that ANN is better than ID3

#### 4.5.3 Dataset Wine

**Neural network** The 10-fold cross-validation results of ANN algorithm on dataset Wine is shown as follows:

| Fold | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 0.176471 | 0.0588235 | 0.224719 | 0.224719 | 0.11236 | 0.0561798 | 0.11236 | 0.11236 | 0.11236 | 0.0561798 |

Table 14: 10-fold error of dataset Votes

The average error rate is

$$E_1 = 0.1247 \tag{28}$$

**ID3** The 10-fold validation result of ID3 algorithm on dataset Wine are shown as follows:

| Fold | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 0.1667 | 0.1667 | 0.2778 | 0.1667 | 0.1111 | 0.0556 | 0.0556 | 0 | 0 | 0.0556 |

Table 15: 10-fold error of dataset Votes

The average error rate from ID3 is

$$E_2 = 0.1056 \tag{29}$$

Comparing ID3 and ANN algorithms, the error difference estimate between ID3 and ANN is:

$$P = -0.0191 \tag{30}$$

The parameters used in t-test is listed below:

$$S_p = 0.0218 \tag{31}$$

$$T_c = 0.543 \tag{32}$$

$$P + T_c S_p < 0$$

There is 70% confidence that ID3 is better than ANN.

#### 4.5.4 Summary

We can see that ID3 algorithm is not necessarily better than ANN. For some dataset (Monks I), ID3 is better, but for other datasets (Votes), ID3 is not better than ANN.

### 4.6 Conclusion

In this experiment, we have realized ANN algorithm for three data sets. We have examined the effects of changing some of the parameters of the neural network and evaluated the corresponding results. We also compared the performance of our neural network with that of ID3 algorithm using 10-fold cross-validation and t-test. We have found out that neural network is not always better than ID3. Each one of them is better at certain datasets. The ANN algorithm takes

a long time to train while ID3 algorithm is much faster. We should consider the characteristics of an algorithm before we choose it as the classifier for a given dataset.