CS5567 Spring 2016 Assignment II. (70+30) pts.

NAME(s) (use Pathway format):

Raghunandan Rao Malangully: rm9vf Santhosh Kumar Gattu: sg6n6

Hirenbhai Harshadbhai Shah: hhstm4 Rakesh Reddy Bandi: rb8xf

Word processed electronic submission on Blackboard is due latest by 11 pm on Wednesday, Mar 2nd. Submissions received after the deadline will be graded only for effort for a maximum of 70% of the total grade (Refer to class syllabus for detailed grading policy). State any assumptions you make, justify your answers, show intermediate steps and explain your results for maximum credit. For machine learning questions, the answer should be written in the form of a concise scientific report with a focus on rationale and evaluation. The report should be a polished depiction of what you did, why you did it, how exactly you did it, and how well it worked.

All answers should be in your own words with any sources you refer to cited at the appropriate places. Any knowledge you acquire from the Internet should be written in your own words and be appropriately referenced. Copying and pasting from the Internet, each other or any other source will not count as your effort (Refer to class syllabus for detailed policy on plagiarism).

You may submit this assignment in groups of up to 4 each. Write your names on this sheet and include it as the cover page for your submission. The submission should consist of a standalone cogent word-processed report file and additional files (code, data, instructions to run your code). Code should be submitted as a text file (not copied and pasted into Word) annotated with comments (include your names within the code files).

Compress all files into an archive before uploading on Blackboard. Be sure to name your archive file using the names of group members, e.g., HW1\_RadhaKrishnaRomeoJuliet (and not just “HW1”). If you upload the wrong file, don’t panic. You may upload upto 3 times.

You may use any programming language or package of your choice for programming assignments in the homework. Possible choices are R, MATLAB, Python (e.g., sci-kit or code on textbook website) or JAVA (e.g., Weka). Be sure to cite your source if you use pre-existing code. Clearly distinguish between pre-existing code and your contributions.

Q1. (15) Write a program that does the following things:

1. Given n and p, generates n coin tosses, with each toss coming up heads with probability p.
2. Repeats ‘A’ a desired number of times.
3. Uses the Expectation Maximization algorithm to deduce p for two different coins that are used for B.

Include a discussion of the effect of the size of the training data on the accuracy of the EM estimates and the number of iterations taken to converge.

Q2. (15) The goal of this assignment is to use Neural Networks to predict if a tennis player won or lost a match based on his/her match statistics. You may reuse data from Assignment 1. Use an appropriate scheme for objective evaluation and reporting predictive accuracy.

1. Use a single layer perceptron.
2. Use a Feed Forward Backpropagation OR radial basis function network.

Include a discussion of the relative performance of “A” and “B” in your report.

**Home Work- 2 Report**

**1.A)**

The task here was to generate ‘n’ coin tosses with probability of the toss resulting in heads to be ‘p(H)’. In our case the value for ‘n’ was 10 and the value for p(H) was assumed to be 0.6. Our choice of language for this task was R. The “sample()” method in R was used for generating the data. The program provides the option to change the value of ‘n’ and ‘p(H)’ which can be provided as input by the user.

**1.B)**

The task here was to repeat (1A) task a desired number of times say ‘i’. This was achieved by using a for loop and calling the “sample()” method ‘i’ times to get random outcomes for each of the iteration. In our case the number of iterations for this task was 5. This value can be provided as an input by the user in order to provide the flexibility to the user. The assumption here was that there are 2 different coins but which coin generated which outcome was not known.

**1.C)**

The task here was to use EM algorithm to deduce ‘p(H)’ for two different coins, let’s say coin A and coin B. The initial values for likelihood of the toss resulting in head from coin A was assumed to be 0.6(thetaA) and similarly for coin B to be 0.5(thetaB). R language was used to perform this task. We implemented the EM algorithm by using the underlying mathematical formulae that predict the possible values for thetaA and thetaB after ‘k’ iterations at which point the algorithm converges.

The next goal was to determine and discuss the effect of training data size on the accuracy of EM estimates and the number of iterations taken to converge. The approach here was to generate varying size of training data by changing the values of ‘n’ and ‘i’. We compared the actual likelihood values with result of the EM algorithm and calculated the error. Since error is inversely proportional to the accuracy of the program. By comparing the error values in different scenarios we could comment on the accuracy for each scenario. In all cases the initial thetaA and thetaB was assumed to 0.6 and 0.5 respectively. In the experiments we conducted, we do not have any information regarding the outcomes as they are randomly generated.

**Following are the results with different sizes of training data.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | # n | # r | # c | # k | Actual  thetaA | Actual  thetaB | EM thetaA | EM thetaB | Error in P(h) for A | Error in P(h) for B |
| 1 | 20 | 2 | 10 | 8 | 0.9000 | 0.5000 | 0.897 | 0.511 | 0.003 | 0.011 |
| 2 | 50 | 5 | 10 | 10 | 0.800 | 0.450 | 0.796 | 0.519 | 0.004 | 0.069 |
| 3 | 100 | 5 | 20 | 42 | 0.677 | 0.642 | 0.580 | 0.775 | 0.097` | 0.133 |

where n- number of coin tosses.

r - number of rows in the matrix

c- number of columns in the matrix

EM thetaA - Predicted p(H) for coin A

EM thetaB - Predicted p(H) for coin B

Actual thetaA - Actual p(H) for coin A

Actual thetaB - Actual p(H) for coin B

k - number of iterations taken by EM algorithm to converge

Error - difference between Predicted likelihood and Actual likelihood.

**Effect of Training data size on the number of EM algorithm iterations to converge:**

From the experiment done we have predicted that as the size of training data increases number of iterations taken by the EM algorithm to converge also increases. So we assumed that there might be some relationship between the training data size and the iterations taken by EM algorithm to converge. But, on testing with the scenario in which training data was 50 coin tosses with the single outcome of heads by both the coins. It was contradicted. In the above case the algorithm converged in just 2 iterations. Comparing this result to the result of the above table’s second row one can infer that size of training data has nothing to do with the number of iterations taken by EM algorithm to converge.

Based on the experiment we conducted and the analysis we carried out we can deduce that the number of iterations taken by EM algorithm to converge depends on the outcomes of the coin tosses but not on the size of the training data.

**Effect of Training data size on the number of EM algorithm iterations to converge:**

Also from the experiment results we can say that as the size of the training data increases the error is also increases. So there seems to be a direct relationship between them. Since error is inversely proportional to the accuracy of any model, we can conclude that with the increase in training data size the accuracy reduces.

Q2) The task was to predict whether a tennis player would win or lose the match based on his match statistics. For the purpose of prediction neural networks was used. The data was collected manually by entering the corresponding values into a csv file. The following features of the match statistics were excluded from the data collection for the reasons mentioned against them.

* 1st serves in -This field does not signify if the player won the point or not even if the serve was in.
* Fastest serve – Even if the player has the fastest serve it is not granted the player won the point when he served the fastest.
* Average 1st serve speed – This feature does not undermine the player's performance on points he scored.
* Average 2nd serve speed – This number is not helpful in making a claim that the player with high average 2nd speed would have won most points to win the match.
* Total points won – This feature is a combination of the other parameters such as Net points won etc. Since we have considered the other parameters using this parameter would result in redundancy.
* Distance covered (M) – It provides the information on how much distance one player has covered in the course of the match. A player might cover a lot of court space through the match but it does not guarantee that he won most of the points. A player who can score more Aces would have less Distance covered.
* Distance Covered/Point (M) – This number is similar to the above feature but calculated per point won. This number does not weigh high in our algorithm.

The following features were considered for the prediction process for the reasons discussed against them.

* Aces - An ace is a service that is in and is not returned by the opponent. So number of aces is directly proportional to the number of points a player has won. The more this number the better are the chances of player winning the match.
* First serve points - This feature gives the number of points won by a player on his first serve. It has a positive impact on the player’s chance of winning the match.
* Second serve points - This feature gives the number of points won by a player on his second serve. It also has a positive impact on the player’s chance of winning the match.
* Net points won - It gives the count of points that a player has won when playing close to the net. It is directly proportional to the probability of player winning the match.
* Breakpoints won - Points won by a player by breaking the opponent’s serve can be derived from this feature. It weighs in the favor of the player winning the match.
* Receiving points won - Points won by a player while the opponent was serving and yet did not cause a break in the opponent’s serve. It favors the chances of a player winning the match.
* Winners - Shots that result in points for the striking player. A high value of winners would signify better chances of player winning the match.
* Double faults - This number is the faults in serve which results in the opponent earning points for faults. The feature has inverse relation to the percentage of a player winning the match.
* Unforced errors - This feature specifies the number of errors made by a player without the opponent forcing him to do so. High amount of unforced errors would lower the chances of a player winning the match.

Data preprocessing:

1. Initial step is to read the data from the csv file and store it as a data frame.
2. **Data Normalization**:

We can normalize the each column of the data using the below formula:

zi=xi−min(x)/(max(x)−min(x))

where x=(x1,...,xn) and zi is ith normalized data.

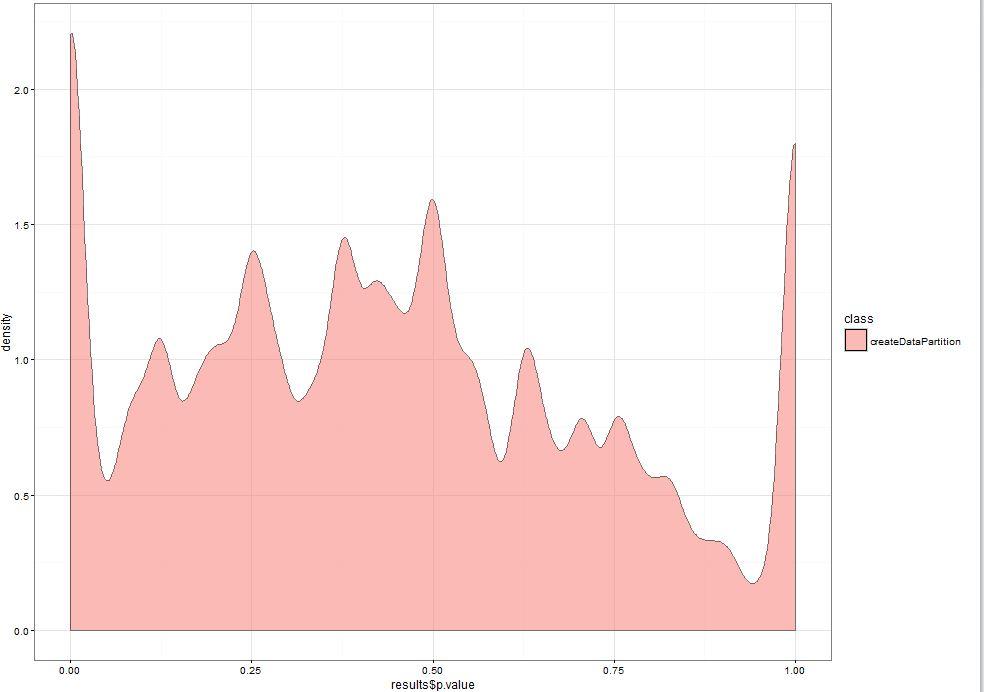
1. **Correlation Matrix** among input (or independent) continuous variables are calculated and we can observe if there is any correlation between the input variables.

It can be observed that No two variables are highly correlated

1. Partitioning Data into Train and Test datasets in 60:40.

A series of test/training partitions are created using createDataPartition from caret library in R. We did 4000 repetitions using random sampling by setting up the seed value 4000 and running it through the loop.

Below graph shows partition density for the p.value.



2 A)

**Fitting the Neural Network using Single Layer Perceptron:**

Training the network is the process of finding the best values of weights to maximise the accuracy of classification. A training iteration is when the network has been shown every sample of training data one time. Training continues over multiple iterations, until the weights reach a steady state value, or a maximum number of iterations is reached.

For our model we need to classify the Win or Loss for statistics of the Player. The Result column is either Win or Loss. The nnet package in R has a slightly strange requirement that the target variable of the classification (ie Result) be in a particular format i.e. It must be in the numerical values. So we change the Win or Loss values to respective 1 and 0 values respectively.

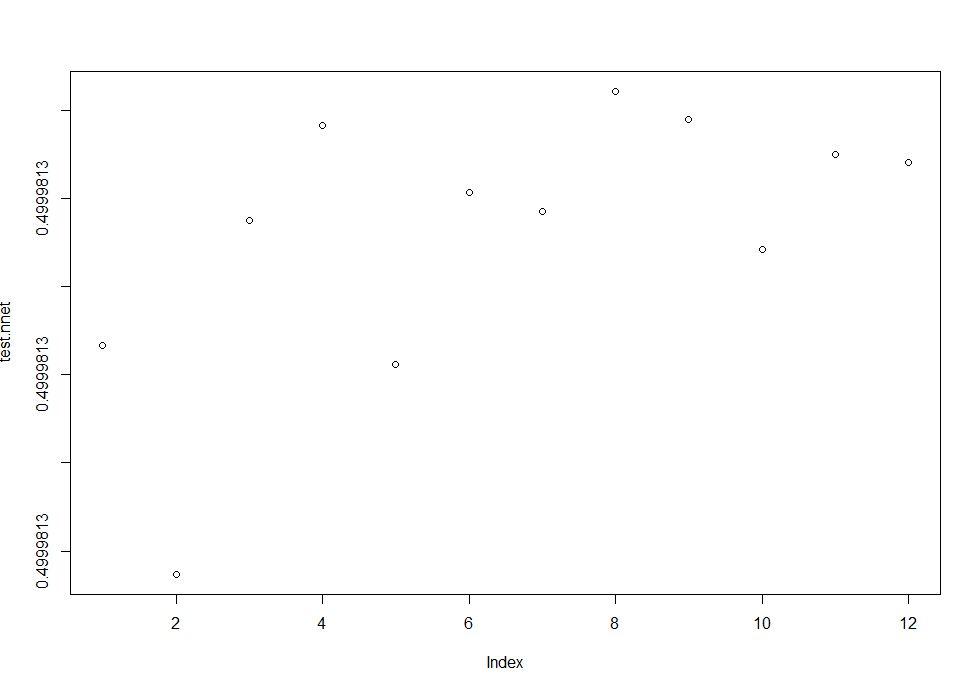
The package we used is nnet, which lets you construct standard Artificial Neural Network with one hidden layer of sigmoid function neurons.

First, we construct the formula for classification. In this case we want to classify the Win or Loss using all features of statistics as inputs. We add a small decay to the weights, so they decrease over time unless reinforced by new data. The second is to increase the maximum number of iterations before training halts, from 100 to 200. These parameters makes the neural network to obtain a good state.

The next parameter is hidden neurons to use h. We have taken the hidden neurons=8 because the hidden neurons must never be twice the number of inputs i.e. 9 input parameters. Generally if we have less hidden neurons we get high training error and high generalization error due to underfitting and high statistical bias. If we take too many hidden units, we are getting low training error but still have high generalization error due to overfitting and high variance.

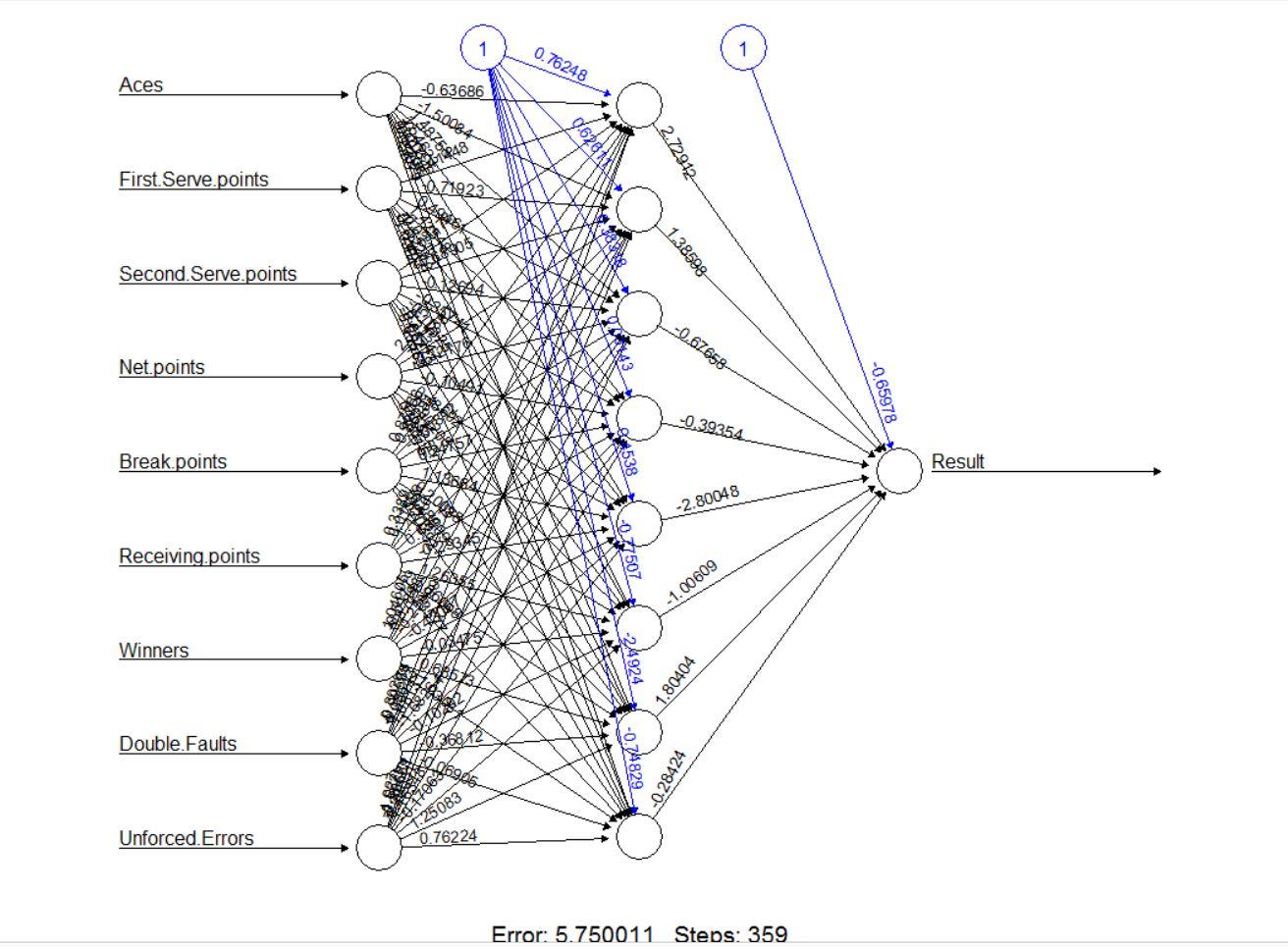
Feature Vector scaling:

Using the training data to train, the artificial neural network has to cover a very wide possibility space of weights to find appropriate values, making tiny changes to match the variations of input values. The Rang parameter in R helps us to solve this by setting the range of weights between +rang and –rang. We have decided on the rang value to 0.5 which is default value. Since our training input is not large the rang parameter value 0.5 could best fit the model. Normalized data which scales the data values between -1 to 1. By normalizing the data all features are comparable in same range and scale. Finally we used the model to predict the results for the test data and we showed the results in confusion matrix.



2.B)

For the purpose of prediction using a complex neural network we decided to use the Feed Forward Backpropagation. This technique was implemented using the neuralnet algorithm in R. The training data collected was provided to the algorithm along with the default threshold value that is decided by the implementation of neuralnet. The activation function used for this purpose was a logistic function. The reason being that we can map the values to real number space. In our case the values were real numbers so we decided it would be better to use a real number based activation function. Assuming the correlation between the features to be low, the features were provided as input to the neuralnet method.



We tested both the approaches in by providing the same test data two both of them and comparing the prediction accuracy for each of them. This step was performed by permuting the test data in various ways and comparing the prediction results for each case. Based on our comparisons we concluded that the Feed Forward Backpropagation has better prediction accuracy. Although the number of iterations that are required by Feed Forward Backpropagation network is relatively high.