
Report on Forensic Science Application

Assignment #4

Vikramaditya Jakkula [10917554]

Introduction

The goal of the assignment is to develop an ANN system which could classify a piece of glass from a beer bottle left behind at a crime scene. We need to train the ANN to classify the brewery the bottle came from using the chemical content of the glass. The developed ANN system would be trained and tested on the data given in the "***GlassData.csv***" file. In this we see a total of 214 samples of datasets of bottle glass given.

The given Attributes Information in "***GlassData.csv***" file is as follows:

1. Ordinal number of glass sample.
2. RI: refractive index
3. Na: Sodium
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of bottle: (class attribute)
 - 1] Anheuser-Busch, Inc.
 - 2] Miller Brewing Co.
 - 3] Blitz-Weinhard Brewing Co.
 - 4] Pete's Brewing Co.
 - 5] Samuel Adams Brew House.
 - 6] Plank Road Brewery.

The last attribute is the bottle class.

A] ANN Configuration Table

There were many runs performed on the system which including increasing and varying the hidden nodes and varying the epochs and learning rate. The detailed procedure is mentioned below in the appropriate section.

The best configuration is given as follows:

Number of Inputs: 9.

Number of Outputs: 6.

Number of Hidden Layers: 2.

Number of Nodes in Layer 1: 5.

Number of Nodes in Layer 2: 6.

Table 1: Displays the Best ANN Run.

# No.	Hidden Layers	Hidden Layer 1	Hidden Layer 2	Learning	Momentum	Epochs	Train Error	Test Error
1	2	5	6	0.1	0.9	1000	0.205	0.386364

Table 2: ANN Configuration table displaying the runs performed on the developed ANN system.

# No.	Hidden Layers	Hidden Layer 1	Hidden Layer 2	Learning	Momentum	Epochs	Training Error	Testing Error
1	2	5	6	0.1	0.9	1000	0.205	0.386364
2	2	5	6	0.1	0.9	750	0.185	0.545455
3	2	5	6	0.1	0.9	1200	0.22	0.5
4	2	5	6	0.01	0.9	1000	0.26	0.545455
5	2	5	6	0.2	0.9	1000	0.0235	0.477727
6	1	3	0	0.1	0.9	600	0.345	0.545455
7	2	3	1	0.1	0.9	600	0.715	0.727273
8	1	5	0	0.1	0.9	600	0.275	0.568182
9	2	5	1	0.1	0.9	600	0.65	0.727273
10	2	5	7	0.1	0.9	600	0.175	0.5
11	1	7	0	0.1	0.9	600	0.24	0.522727
12	2	9	8	0.1	0.9	600	0.13	0.5
13	2	10	10	0.1	0.9	600	0.175	0.431818

Note: As the question asks us for a table with the values from earlier requested table a complete table is given in Appendix A.

B] Graph: Training/Testing Error vs. Epochs.

The errors were calculated on the data computed out as the given output from the ANN simulator used to build the ANN system. The simulator was run for different epochs with the same configuration (which is the best configuration), which was found and fixed as the best configuration before running again, and then we calculate the error resulted in the testing and training from the outputted outtrain.csv and out test.csv. From these files we look at the predicted output and compare it with the desired output and if the predicted output of same label (class) with the highest value matches with the desired values pertaining to the same class then we consider it as a correct prediction and if it does not match then it is a incorrect prediction and in this was we compute the error to be the number of wrong predictions by the total number of predictions. We perform similar method to compute the error in the out train.csv (or the training error associated file). Thus both the training and testing error are obtained using the MS Excel tool for the best ANN. The Training/Testing Errors vs. Epochs for the best ANN is plotted below. We see that the RMSE and R-Square values cannot be used for this as this is a classification process and not a regression problem.

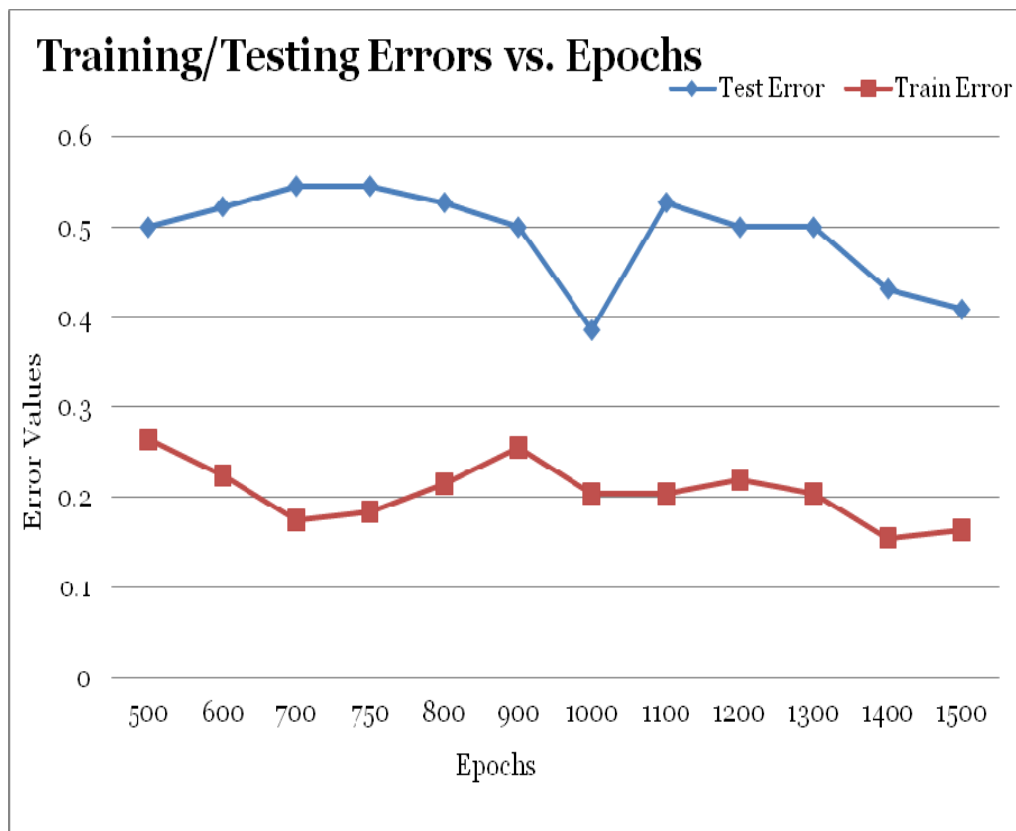


Figure 1: Training/Testing Errors vs. Epochs Graph.

Table 3: Computed Training/Testing Errors and Epochs for the best ANN

# No	Hidden Layer 1	Hidden Layer 2	Learning	Momentum	Epochs	Testing Error	Training Error
1	5	6	0.1	0.9	500	0.5	0.265
2	5	6	0.1	0.9	600	0.522727	0.225
3	5	6	0.1	0.9	700	0.545455	0.175
4	5	6	0.1	0.9	750	0.545455	0.185
5	5	6	0.1	0.9	800	0.527273	0.215
6	5	6	0.1	0.9	900	0.5	0.255
7	5	6	0.1	0.9	1000	0.386364	0.205
8	5	6	0.1	0.9	1100	0.527273	0.205
9	5	6	0.1	0.9	1200	0.5	0.22
10	5	6	0.1	0.9	1300	0.5	0.205
11	5	6	0.1	0.9	1400	0.431818	0.155
12	5	6	0.1	0.9	1500	0.409091	0.165

Note: Development Process [c] is explained below.

E] Data Preprocessing I: Normalization Explained!

The input and output data must be normalized for the ANN learning to prevent a specific factor from dominating the learning. In general, normalization gives a value between 0 and 1. However, we would “*normalize values between 0.05 and 0.95*” to prevent the generation of 0 and 1 for better input data. Normal Normalization is calculated by $X = (X_i - X_{min}) / (X_{max} - X_{min})$; where X is normalized value and X_{max} is the maximum value in the set of values and X_{min} is the minimum value in the set of values given.

But we use $X = ((X_i - X_{min}) / (X_{max} - X_{min}) * 0.90) + 0.05$; for generating the normalized values between 0.05 & 0.95. We apply this normalization to the input parameters.

We choose nine input attributes/parameters. Now in the next step we change the classes into 6 output parameters rather than having one output parameter.

Table 4: Sample Normalized datasets are displayed in the Table below.

RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Out put 1	Out put 2	Out put 3	Out put 4	Out put 5	Out put 6
0.24	0.33	0.78	0.42	0.62	0.15	0.28	0.05	0.49	0.95	0	0	0	0	0
0.31	0.35	0.80	0.34	0.62	0.13	0.31	0.05	0.36	0	0.95	0	0	0	0
0.50	0.56	0.86	0.22	0.31	0.07	0.37	0.05	0.05	0	0	0.95	0	0	0
0.38	0.50	0.83	0.31	0.44	0.05	0.35	0.05	0.05	0	0	0	0.95	0	0
0.31	0.34	0.80	0.40	0.54	0.13	0.32	0.05	0.40	0	0	0	0	0.95	0
0.31	0.34	0.80	0.36	0.59	0.14	0.32	0.05	0.05	0	0	0	0	0	0.95
0.30	0.42	0.79	0.30	0.56	0.12	0.31	0.05	0.05	0.95	0	0	0	0	0

D] Data Preprocessing II: Data Splitting Exposed!

In this we see that first we have normalized the given data file. Now the task of choosing the input and output parameters is looked into. Now we discard Ordinal number from the input datasets. We use 9 Inputs which include the refractive index (RI), Sodium (Na), Magnesium (Mg), Aluminum (Al), Silicon (Si), Potassium (K), Calcium (Ca), Barium (Ba), and Iron (Fe). We consider 6 outputs where we see that the 6 class were made into 6 outputs which would make classification better compared to having one output for classifying 6 classes. We are working towards a classification problem. We have seen that now the normalized data file would have 15 attributes computing of nine inputs and six outputs.

Then we first have to take the datasets required for testing from this normalized dataset. We see that we take 20% of the given data set is taken as testing set. Thus we follow the 80%-20% split. Thus we see that after getting the normalized dataset we should first take the test data out and see to it that we do not have any duplicates into the test data. So we handpick data for the test set and this data is seen to it such a way that each class contributes 20% of their datasets as test data thus making an overall contribution of 20% for test data and the remaining is used for the training data which is 80%. Now we see that the dataset for training is made of 170 datasets and we also observe that the classes have a less number of datasets and it is up to us to make a choice as should we go ahead and duplicate datasets which are considerably low and increase their class samples by duplicating the entries. Here we see that the class samples which are low include Blitz-Weinhard Brewing Co, Pete's Brewing Co. Samuel Adams Brew House, and Plank Road Brewery. Thus after removing the test data we see they have very less data sets to be used hence we go ahead and handpick few data sets for these classes and insert them again to increase the training set data for only these classes. We do not do down sampling as it is not good. Thus we raise all the low number datasets to 25 datasets and do not change the datasets which are more in number and belong to same class. Now when we add up we have good number of training data making it 217 samples and adequate samples for each respective class. These are clearly illustrated in the table and diagram below.

We see that we have handpicked some data only and have not removed data for classes which have more number of data sets. Now we save the training file as Train.csv and the testing file as Test.csv. We have used MS excel for the entire process. We use these two files as testing and training the data sets. The inputs are always 9 and the outputs are 6 for the entire experiment as this is a classification problem and not regression and we have fixed them so. We use the given ANN simulator to run the simulations and build an ANN system using the formulated training and test datasets.

Table 5: Training & testing data splits.

Class	Initial Total	Test Set	Total After removing Test Data	Duplicating Datasets	Training Set
Anheuser-Busch, Inc.	70	14	56	0	56
Miller Brewing Co.	76	15	61	0	61
Blitz-Weinhard Brewing Co.	17	4	13	12	25
Pete's Brewing Co.	13	3	10	15	25
Samuel Adams Brew House	9	2	7	18	25
Plank Road Brewery	29	6	23	2	25
	214	44	170		217
		20%	80%		

The above training and testing split table is represented as a 3-D illustration below. We see number of examples plotted against the different classes.

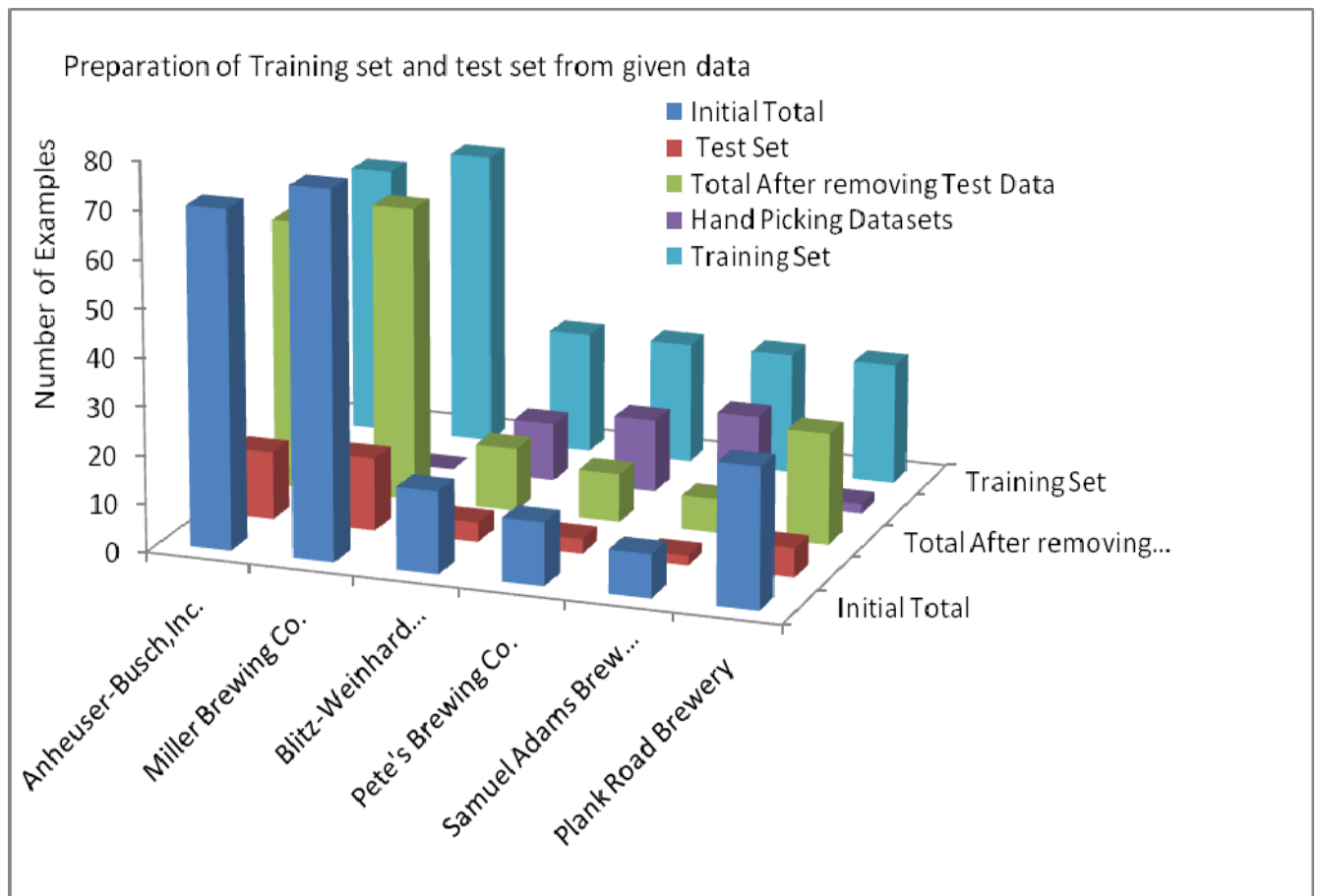


Figure 2: 3-D illustration of the datasets splits with respect to the six different classes. [Note handpicking datasets in the legend is same as duplicating datasets, please kindly note this].

C] Development Process: A walk through!

Now from above we have seen that the given data file is normalized and the later data splitting is performed to identify the training and test sets and we use them with the provided ANN simulator. Now coming to building the system the main task is choosing the right parameters of learning rate, momentum, the number of hidden layers, and the epochs required for the experiment to run and later we record these runs to find the best possible configuration which gives us the lowest possible error. After getting the environment set and ready, we first take the hidden nodes to be one layer with 3 nodes and run the experiment and later we increase it to two layers and we use 3 hidden nodes in one layer and 1 hidden node in layer two and we see the error is high and now we increase the hidden nodes in layer one to 5 and make it a single layer and run the experiment and later run 5 nodes in layer two in this way we run the experiment several times with varying the hidden nodes till we reach 10 and 10 with 600 epochs and find that the error is moderate around the having 5 hidden nodes and have two hidden layers. Later we change the epochs to 1000 and changing the number of hidden layers and find that having two hidden layers with 5 nodes in layer one and 6 nodes in layer two had the least error of 0.3863. We now change the epochs and run it for fewer and more epochs and see that it does not perform better. But we record all these runs for finding the errors as we need to later use them for plotting the graphs required. We later change the learning rate and momentum and see that the error is more than the minimum which was obtained earlier and thus we have the best ANN configuration with a learning rate of 0.1, momentum of 0.9, two hidden layers with 5 hidden nodes in first and 6 hidden nodes in the second and run of 1000 epochs resulted in error rate of 0.38 on the testing set.

Note: More development information can be obtained. Please feel free to email me if any questions.

F] Confusion Matrix for Best ANN

Table 3: Confusion matrix for the Best ANN.

	Predicted Value						
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	
Class 1	78.57%	14.29%	7.14%	0%	0%	0%	
Class 2	26.67%	53.33%	0%	13.33%	6.67%	0%	
Class 3	25%	75%	0%	0%	0%	0%	
Class 4	33.33%	33.33%	0%	0%	0%	33.33%	
Class 5	0%	0%	0%	0%	100%	0%	
Class 6	0%	0%	0%	0%	0%	100%	

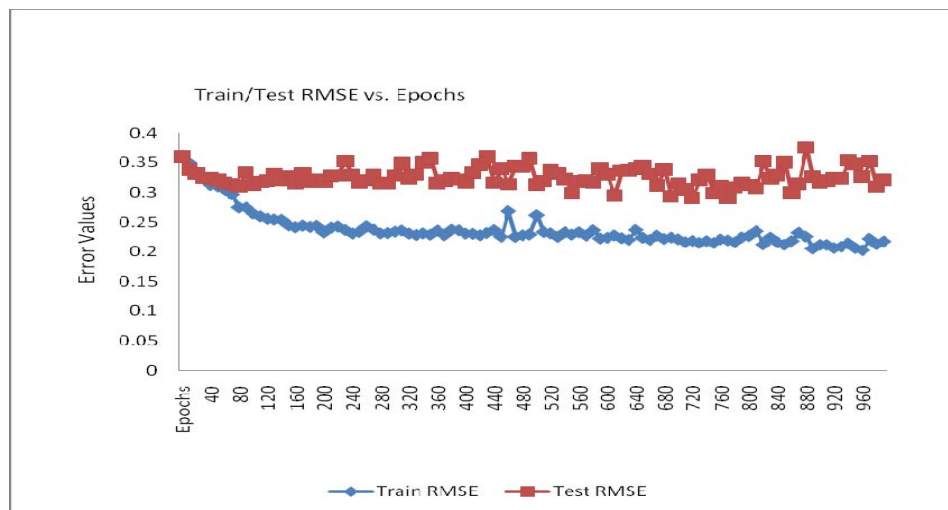
A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. We calculate confusion matrix as described in the class.

Appendix A: Complete Table for Runs performed.

#	Hidden Layers	Hidd. Layer 1	Hidd. Layer 2	Learn Rate	Momen. Rate	epochs	Train Error	Test Error	Train RMSE	Test RMSE
1	2	5	6	0.1	0.9	1000	0.105	0.3863	0.21867	0.32105
2	2	5	6	0.1	0.9	750	0.185	0.5454	0.20230	0.35772
3	2	5	6	0.1	0.9	1200	0.22	0.5	0.20348	0.33823
4	2	5	6	0.01	0.9	1000	0.26	0.5454	0.21984	0.34992
5	2	5	6	0.2	0.9	1000	0.0235	0.4777	0.22127	0.34690
6	1	3	0	0.1	0.9	600	0.345	0.5454	0.25031	0.35970
7	2	3	1	0.1	0.9	600	0.715	0.7272	0.31401	0.33242
8	1	5	0	0.1	0.9	600	0.275	0.5681	0.23035	0.35113
9	2	5	1	0.1	0.9	600	0.65	0.7272	0.30961	0.32967
10	2	5	7	0.1	0.9	600	0.175	0.5	0.20377	0.34783
11	1	7	0	0.1	0.9	600	0.24	0.5227	0.20632	0.37570
12	2	9	8	0.1	0.9	600	0.13	0.5	0.17717	0.36544
13	2	10	10	0.1	0.9	600	0.175	0.4318	0.19409	0.31319

Note: R-Square is not required as this is not regression problem.

Appendix B: Training/Testing RMSE vs Epochs



References

- 1] Class Notes
- 2] Discussion in class and email discussions and hints provided by lecturer.
- 3] General information on confusion matrix on the World Wide Web.
- 4] Additional Information provided in class and class prescribed textbook.