Car Evaluation

02/March/2017

1 Group members

Name	Roll number	email-id
Shivam Porwal	1611CS14	shivam.mtcs16@iitp.ac.in
Debanjan Sarkar	1611CS17	debanjan.mtcs16@iitp.ac.in

2 Abstract of the project

Car Evaluation Dataset evaluate the target concept(CAR) with 3 other intermediate concept. PRICE(overall price),TECH(technical characteristic) and COMFORT(comfort). Now totally we have 6 attribute, each attribute is a part of one of the intermediate concept as described above. These attributes are :
1. Buying(buying price)
2. Maint(price of maintenance)
3. Doors(number of doors)
4. Persons(capacity in terms of persons to carry)
5. *Lug_Boot*(the size of luggage bot)
6. Safety(estimated safety of the car)

The number of the instances in the training data are 1728 , and there are 6 number of attributes as mentioned above. This is basically a multi-class classification problem. we will classify the instance into 4 classes:

Here are the attribute values:

Table 1: **Attribute values**

buying	{v-high, high, med, low}
maint	{v-high, high, med, low}
doors	{2, 3, 4, 5- more}
persons	{2, 4, more}
lug_boot	{small, med, big}
safety	{low, med, high}

Table 2: **Class Distribution (number of instance per class)**

unacc	1210
acc	384
good	69
v-good	65

3 Introduction

To implement such type of model we will use the concept of Convolution Neural Network.To implement it we will use the python library "keras" in which we are using backend as Theano. To find the highest

accuracy we will consider various scenarios or cases which are described in the upcoming topics.

3.1 Data sources

https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

3.2 Literature survey

In this survey we have studied various techniques on car evaluation dataset. Each technique has different output with different accuracies. The accuracy achieved under different experiment conditions or setting by Decision Tree, Naive Bayesian, and Artificial Neural Network (ANN) are presented.

Classification Accuracy of Decision Tree :

Decision tree builds classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Percentage Split		Time in Seconds		Decision Tree	
Training %	Testing %	Build	Test	Correct %	Incorrect %
90	10	0.07	0.01	93.06	6.93
66	44	0.01	0.01	90.81	9.18
50	50	0.01	0.02	92.7	7.29
10 Folds		0.01	0.01	93.22	6.77

fig : Accuracy Result of Decision Tree

Classification Accuracy of Naive Bayesian : The Naive Bayesian algorithm, named after Thomas Bayes is a learning algorithm as well as a statistical method for classification. It captures uncertainty in a principled way by using probabilistic approach. Naive Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined.

Percentage Split		Time in Seconds		Naive Bayesian	
Training %	Testing %	Build	Test	Correct %	Incorrect %
0	10	0.02	0.05	93.06	6.93
66	44	0	0.03	92.51	7.48
50	50	0	0.04	92.7	7.29
10 Folds		0	0.25	93.51	6.48

fig : Accuracy Result of Naive Bayes

Classification Accuracy of ANN :

The Artificial Neural Network (ANN) algorithm takes data as input then process and generalizes output using biological brain patterns of humans or animals. It is designed to learn in a non linear mapping between input and output data.

Percentage Split		Time in Seconds		ANN	
Training %	Testing %	Build	Test	Correct %	Incorrect %
90	10	7.1	0	93.06	6.93
66	44	7.19	0.01	90.81	9.18
50	50	6.98	0.02	92.7	7.29
10 Folds		7	0.03	93.51	6.48

fig : Accuracy Result of ANN

The following tests were performed by Tijana Jovanovic, Faculty of Organisation Sciences, University of Belgrade.
In this example they will be using 80% of data for training the network and 20% of data for testing it.
Training attempt 1:

- Network Type: Multi Layer Perceptron
- Training Algorithm: Backpropagation with Momentum
- Number of inputs: 21
- Number of outputs: 4 (unacc,acc,good,vgood)
- Hidden neurons: 14

Training Parameters:

- Learning Rate: 0.2
- Momentum: 0.7
- Max. Error: 0.01

Training Results:
For this training, they used Softmax transfer function.

Following results were generated in the First Attempt

	Network Inputs						Real Outputs			
Instance number	Buying	Maint	Doors	Persons	Lug boot	Safety	Unacc	Acc	Good	VGood
1.	0,0,0,1(vhigh)	0,0,0,1(vhigh)	1,0,0,0(2)	1,0,0(2)	0,1,0,(med)	0,1,0(med)	1	0	0	0
2.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,1,0 (4)	0,0,1 (big)	0,0,1 (high)	0	0	0	1
3.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,1,0 (4)	1,0,0 (small)	1,0,0 (low)	1	0	0	0
4.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,0,1 (more)	1,0,0 (small)	0,1,0 (med)	0	1	0	0
5.	1,0,0,0 (low)	0,1,0,0 (med)	0,0,0,1 (5more)	0,1,0 (4)	0,0,1 (big)	0,1,0 (med)	0	0	1	0

The **output neural network produced** for this input is, respectively:

	Network Inputs						Outputs neural network produced			
Instance number	Buying	Maint	Doors	Persons	Lug boot	Safety	Unacc	Acc	Good	VGood
1.	0,0,0,1(vhigh)	0,0,0,1(vhigh)	1,0,0,0(2)	1,0,0(2)	0,1,0,(med)	0,1,0(med)	1	0	0	0
2.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,1,0 (4)	0,0,1 (big)	0,0,1 (high)	0,0009	0,0002	0,0053	0,9931
3.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,1,0 (4)	1,0,0 (small)	1,0,0 (low)	1	0	0,0001	0
4.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,0,1 (more)	1,0,0 (small)	0,1,0 (med)	0,0033	0,9965	0,0025	0
5.	1,0,0,0 (low)	0,1,0,0 (med)	0,0,0,1 (5more)	0,1,0 (4)	0,0,1 (big)	0,1,0 (med)	0,0002	0,0006	0,9973	0,0016

Training attempt 2:

- Network Type: Multi Layer Perceptron
- Training Algorithm: Backpropagation with Momentum
- Number of inputs: 21
- Number of outputs: 4 (unacc,acc,good,vgood)
- Hidden neurons: 14

Training Parameters:

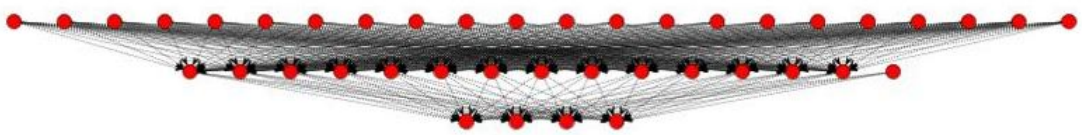
- Learning Rate: 0.3
- Momentum: 0.6
- Max. Error: 0.01

Training Results:
For this training, they used Softmax transfer function.

Following results were generated in the First Attempt

	Network Inputs						Outputs neural network produced			
Instance number	Buying	Maint	Doors	Persons	Lug boot	Safety	Unacc	Acc	Good	VGood
1.	0,0,0,1(vhigh)	0,0,0,1(vhigh)	1,0,0,0(2)	1,0,0(2)	0,1,0,(med)	0,1,0(med)	1	0	0	0
2.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,1,0 (4)	0,0,1 (big)	0,0,1 (high)	0	0	0	0,9996
3.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,1,0 (4)	1,0,0 (small)	1,0,0 (low)	1	0	0	0
4.	1,0,0,0 (low)	1,0,0,0 (low)	0,0,0,1 (5more)	0,0,1 (more)	1,0,0 (small)	0,1,0 (med)	0	1	0	0
5.	1,0,0,0 (low)	0,1,0,0 (med)	0,0,0,1 (5more)	0,1,0 (4)	0,0,1 (big)	0,1,0 (med)	0	0	1	0

3.3 Network Architecture :



fig(1) : Convolution Neural Network

3.4 Our Results:

Case 1:

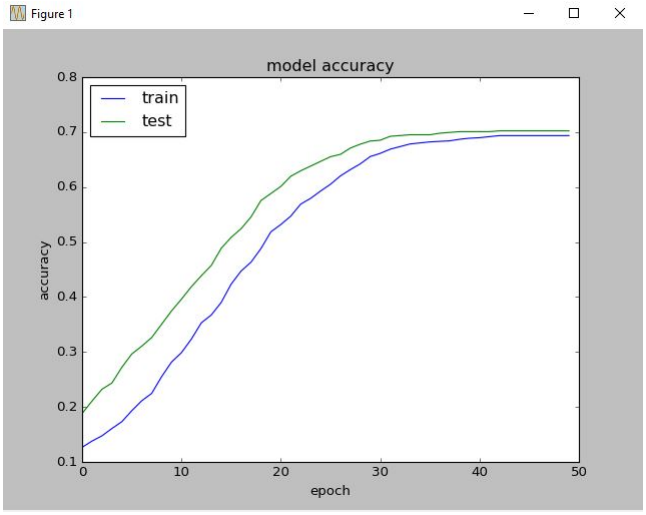
- Input Neuron: 21
- Number of Hidden Layers: 2
- Hidden Neurons in first layer: 100
- Hidden Neurons in first layer: 80
- Output Neurons : 4
- Activation function : Softmax function (both in hidden layer and output layer)
- Learning rate : 0.01
- Momentum : 0.7
- Number of Iterations : 50

Result of Case 1:

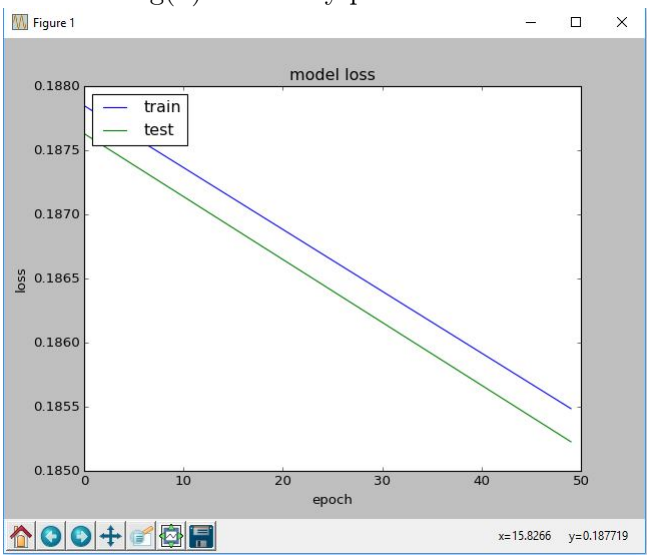
```
Epoch 48/50
1051/1051 [==============================] - 0s - loss: 0.1856 - acc: 0.6936 - val_loss: 0.1853 - val_acc: 0.7023
Epoch 49/50
1051/1051 [==============================] - 0s - loss: 0.1855 - acc: 0.6936 - val_loss: 0.1853 - val_acc: 0.7023
Epoch 50/50
1051/1051 [==============================] - 0s - loss: 0.1855 - acc: 0.6936 - val_loss: 0.1852 - val_acc: 0.7023
('mean squared error :', 0.18522654995959029)
('PREDICTED', array([[ 0.25511843,  0.25099114,  0.24640666,  0.24748382],
       [ 0.25787479,  0.25146687,  0.24438058,  0.24627775],
       [ 0.25605804,  0.25050306,  0.24628173,  0.24715714],
       ...,
       [ 0.25801155,  0.25079462,  0.24267635,  0.24851747],
       [ 0.25723404,  0.25106379,  0.2437406 ,  0.24796154],
       [ 0.25563523,  0.25113648,  0.24443632,  0.24879205]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0]]))
```

fig(a): Output of Case1

Graphical Representation:



fig(b): Accuracy plot of Case1



fig(c): Error plot of Case1

Case 2:

- Input Neuron: 21
- Number of Hidden Layers: 3
- Hidden Neurons at Layer 1: 100
- Hidden Neurons at Layer2 : 80

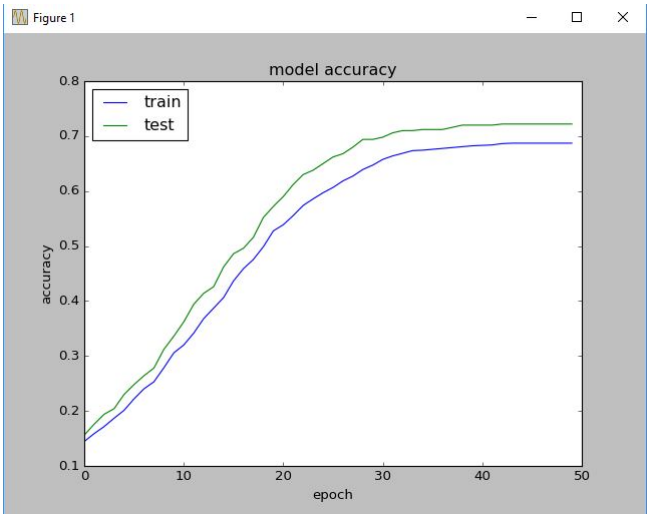
- Hidden Neurons at Layer2 : 70
- Output Neurons : 4
- Activation function : Softmax function (both in hidden layer and output layer)
- Learning rate : 0.1
- Momentum : 0.7
- Number of Iterations : 50

Result of Case 2:

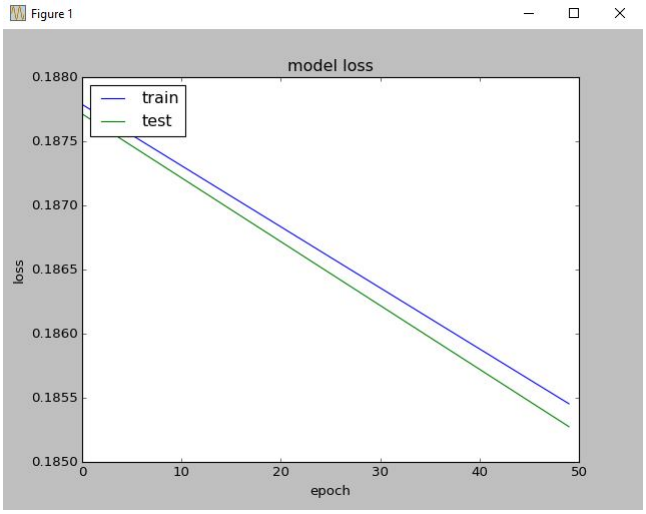
```
Epoch 48/50
1253/1253 [==============================] - 0s - loss: 0.1855 - acc: 0.6872 - val_loss: 0.1854 - val_acc: 0.7220
Epoch 49/50
1253/1253 [==============================] - 0s - loss: 0.1855 - acc: 0.6872 - val_loss: 0.1853 - val_acc: 0.7220
Epoch 50/50
1253/1253 [==============================] - 0s - loss: 0.1855 - acc: 0.6872 - val_loss: 0.1853 - val_acc: 0.7220
('mean squared error :', 0.18527450728416442)
('PREDICTED', array([[ 0.25498781,  0.25123912,  0.24630475,  0.24746837],
       [ 0.25774479,  0.25171679,  0.24427842,  0.24626009],
       [ 0.25592873,  0.25075242,  0.24617855,  0.24714026],
       ...,
       [ 0.25769141,  0.2488703 ,  0.24372566,  0.24971269],
       [ 0.2569291 ,  0.24913627,  0.24478082,  0.24915382],
       [ 0.2553277 ,  0.2492083 ,  0.24547216,  0.24999177]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [0, 0, 1, 0],
       [1, 0, 0, 0],
       [0, 0, 1, 0]]))
```

fig(d): Output of Case2

Graphical Representation:



fig(e): Accuracy plot of Case2



fig(f): Error plot of Case2

Case 3:

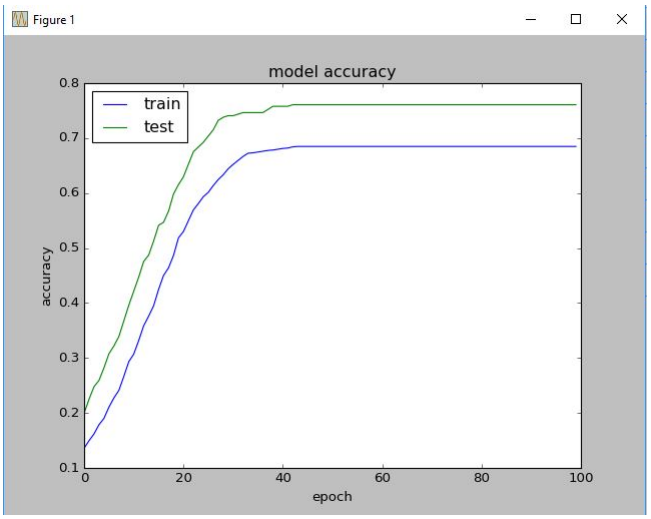
- Input Neuron: 21
- Number of Hidden Layers: 4
- Hidden Neurons at Layer 1: 100
- Hidden Neurons at Layer 2 : 80
- Hidden Neurons at Layer 3 : 70
- Hidden Neurons at Layer 4 : 60
- Output Neurons : 4
- Activation function : Softmax function (both in hidden layer and output layer)
- Learning rate : 0.1
- Momentum : 0.7
- Number of Iterations : 50

Result of Case 3:

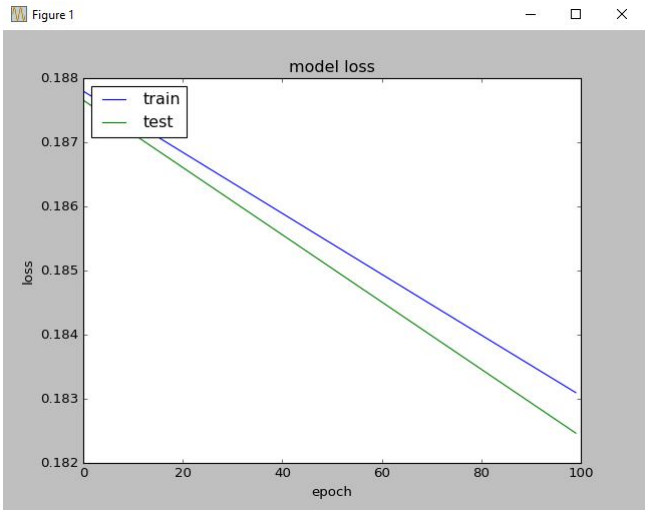
```
Epoch 98/100
1377/1377 [==============================] - 0s - loss: 0.1832 - acc: 0.6848 - val_loss: 0.1826 - val_acc: 0.7607
Epoch 99/100
1377/1377 [==============================] - 0s - loss: 0.1831 - acc: 0.6848 - val_loss: 0.1825 - val_acc: 0.7607
Epoch 100/100
1377/1377 [==============================] - 0s - loss: 0.1831 - acc: 0.6848 - val_loss: 0.1825 - val_acc: 0.7607
('mean squared error :', 0.18246412226277539)
('PREDICTED', array([[ 0.26247782,  0.25109023,  0.24260128,  0.2438307 ],
       [ 0.26528937,  0.25153652,  0.24056438,  0.24260977],
       [ 0.26358503,  0.25057003,  0.24240461,  0.24344037],
       ...,
       [ 0.26388296,  0.24856946,  0.24161132,  0.2459363 ],
       [ 0.26217005,  0.24867274,  0.24234924,  0.24680793],
       [ 0.26497576,  0.24912071,  0.24032374,  0.24557976]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]]))
```

fig(g): Output of Case3

Graphical Representation:



fig(h): Accuracy plot of Case3



fig(i): Error plot of Case3

Case 4:

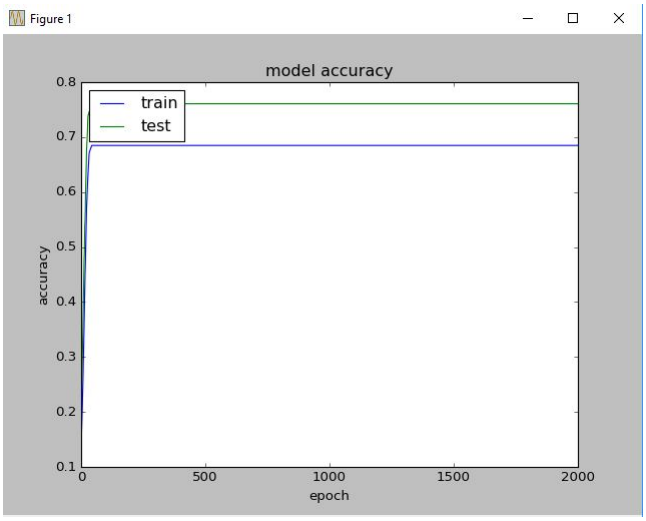
- Input Neuron: 21
- Number of Hidden Layers: 4
- Hidden Neurons at Layer 1: 100
- Hidden Neurons at Layer2 : 80
- Hidden Neurons at Layer 3 : 70
- Hidden Neurons at Layer 4 : 60
- Output Neurons : 4
- Activation function : Softmax function (both in hidden layer and output layer)
- Learning rate : 0.1
- Momentum : 0.7
- Number of Iterations : 2000

Result of Case 4:

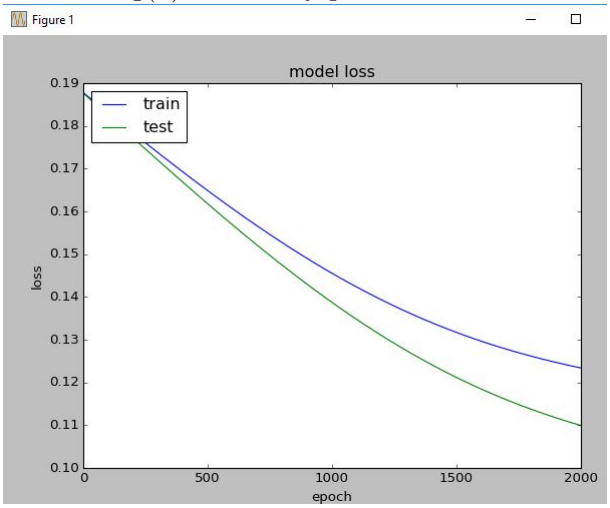
```
Epoch 1996/2000
1377/1377 [==============================] - 0s - loss: 0.1234 - acc: 0.6848 - val_loss: 0.1100 - val_acc: 0.7607
Epoch 1997/2000
1377/1377 [==============================] - 0s - loss: 0.1234 - acc: 0.6848 - val_loss: 0.1100 - val_acc: 0.7607
Epoch 1998/2000
1377/1377 [==============================] - 0s - loss: 0.1234 - acc: 0.6848 - val_loss: 0.1100 - val_acc: 0.7607
Epoch 1999/2000
1377/1377 [==============================] - 0s - loss: 0.1234 - acc: 0.6848 - val_loss: 0.1100 - val_acc: 0.7607
Epoch 2000/2000
1377/1377 [==============================] - 0s - loss: 0.1234 - acc: 0.6848 - val_loss: 0.1099 - val_acc: 0.7607
('mean squared error :', 0.10994575832813894)
('PREDICTED', array([[ 0.57206655,  0.18295768,  0.1211511 ,  0.12382458],
       [ 0.57741946,  0.1815847 ,  0.1189913 ,  0.12200451],
       [ 0.59014457,  0.17648116,  0.11531767,  0.11805657],
       ...,
       [ 0.56895602,  0.18234132,  0.12239027,  0.12631236],
       [ 0.54921758,  0.18871984,  0.12910526,  0.13295737],
       [ 0.55484444,  0.1873481 ,  0.12680177,  0.13100566]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]]))
```

fig(j): output of Case4

Graphical Representation:



fig(k): Accuracy plot of Case4



fig(l): Error plot of Case4

3.5 Implementation :

The implementation of following models can be found on following link :
https://github.com/shiv8989/carevaluation

4 Future work :

In this program we are training the model based on some hidden layer. if the size of hidden layer is increased then model can be trained more accurately but complexity of the network may increase. Due to which it is also possible for reduction in accuracy. To solve this problem we can add more data, due to increase in data-size. Model can be trained more accurately and accuracy will increase. Due to the limited amount of data the accuracy can not cross the certain threshold. If we train the network by large number of data items performance will increase.

References

[1] Knowledge acquisition and explanation for multi-attribute decision making by M Bohanec, V Rajkovic

[2] Machine Learning by Function Decomposition Blaz Zupan, Marko Bohanec, Ivan Bratko, Janez Demsar

[3] https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

[4] http://neuroph.sourceforge.net/tutorials/car$_e$*valuation*1/*car*$_e$*valuation*1.*html*