Analysis of artificial Neural Network:

Pranav Pande (pmpande) | Nishant Shah (nishshah)

It was seen that initial weights, number of neurons in hidden layer and values of target vectors affected the learning time significantly. We started with the problem assigning initial weight as a random number between $(-1/\sqrt{i},1/\sqrt{i})$ where i is the number of input neurons. But rigorous training of this structure achieved us the maximum accuracy of 30% in test set.

So we changed the initial weight to random number between (-0.1,0.1). With less training with this structure, we are able to achieve accuracy of average 70% in test set.

We conducted eight different experiments with different initial weight, number of neurons in hidden layer, values of target function and number of training for giving training set.
Results and configuration of the network are detailed below:

Observation 1:

Algorithm:	ANN	**No of hidden neurons:**	**40**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**0.1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**1**	
Confusion Matrix:	**0**	**90**	**180**	**270**
0	165	16	20	38
90	10	170	6	38
180	25	28	142	41
270	17	23	5	199
Total Correct:	676	Total	943	
Accuracy:	**71.68610817**			

Observation 2:

Algorithm:	ANN	**No of hidden neurons:**	**40**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**1**	
Confusion Matrix:	0	90	180	270
0	161	23	23	32
90	11	175	2	36
180	28	40	138	30
270	15	39	4	186
Total Correct:	660	Total	943	
Accuracy:	**69.98939555**			

Observation 3:

Algorithm:	ANN	**No of hidden neurons:**	**100**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**1**	
Confusion Matrix:	0	90	180	270
0	148	11	18	62
90	7	143	6	68
180	18	13	137	68
270	7	16	5	216
Total Correct:	644	Total	943	
Accuracy:	**68.29268293**			

Observation 4:

Algorithm:	ANN	**No of hidden neurons:**	**100**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**10**	
Confusion Matrix:	0	90	180	270
0	144	30	14	51
90	7	173	1	43
180	21	38	132	45
270	7	27	2	208
Total Correct:	657	Total	943	
Accuracy:	**69.67126193**			

Observation 5:

Algorithm:	ANN	**No of hidden neurons:**	**60**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-1,1)*1/SQRT(No of input neurons)**	**Learning Rate:**	**0.1**	
Target vector:	**1 for True, -1 for False**	**No. of training**	**20**	
Confusion Matrix:	0	90	180	270
0	0	26	94	119
90	0	49	68	107
180	0	76	42	118
270	0	23	79	142
Total Correct:	233	Total	943	
Accuracy:	**24.70837752**			

Observation 6:

Algorithm:	ANN	**No of hidden neurons:**	**100**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-1,1)*1/SQRT(No of input neurons)**	**Learning Rate:**	**0.1**	
Target vector:	**1 for True, -1 for False**	**No. of training**	**1**	
Confusion Matrix:	0	90	180	270
0	183	1	42	13
90	146	6	53	19
180	124	5	74	33
270	168	0	65	11
Total Correct:	274	Total	943	
Accuracy:	**29.05620361**			

Observation 7:

Algorithm:	ANN	**No of hidden neurons:**	**100**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**0.1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**10**	
Confusion Matrix:	0	90	180	270
0	174	21	22	22
90	10	170	8	36
180	35	29	147	25
270	12	30	12	190
Total Correct:	681	Total	943	
Accuracy:	**72.21633086**			

Observation 8:

Algorithm:	ANN	**No of hidden neurons:**	**40**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**0.1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**5**	
Confusion Matrix:	0	90	180	270
0	172	17	26	24
90	13	172	9	30
180	35	23	157	21
270	16	28	10	190
Total Correct:	691	Total	943	
Accuracy:	**73.2768**			

So one can see that we were able to achieve maximum accuracy of 73.2768 %.
It was also seen that number of training set and number of trainings affect the accuracy in exponential way. One experiment with training set from 8000th row to 10000th row and training them for just 2 times gave us accuracy of 68%.
Result and configuration of it is as below.

Very small training dataset

Algorithm:	ANN	**No of hidden neurons:**	**40**	
No. of input neurons:	192	No. of output Neurons:	4	
Initial weights:	**random(-0.1,0.1)**	**Learning Rate:**	**0.1**	
Target vector:	**1 for True, 0 for False**	**No. of training**	**2**	
No. of training examples:	**2000**			
Confusion Matrix:	0	90	180	270
0	179	21	34	5
90	34	157	20	13
180	35	37	160	4
270	42	34	22	146
Total Correct:	642	Total:	943	
Accuracy:	**68.08059385**			

The results from K-nearest neighbor approach also provides us best success rate of around 70.23 %.

K	Accuracy
5	68.23 %
10	69.56 %
75	70.23 %

Considering the amount of execution time required to run the knn approach we have tested very less combinations, out of which we have provided most useful ones.

From the above results we can infer that the accuracy with both the approaches are nearly same but considering the execution time and the risk of overfitting the data we conclude that neural network classifier approach is best among the two approaches with following factors.

Weight initialization: Random(-0.1,0.1)
Learning rate: 0.1
Number of hidden units/nodes: 40
Number of iterations: 5