SOFT COMPUTING-NEURAL NETWORK, ASSIGNMENT-03

I have combined the questions under these sections in the HomeWork ***"You should turn in the following"*** *and* ***"Naturally, you will need to accomplish the following tasks"*** *for completing my assignment.*

Question 1: Detail, supported by actual performance data, how well your best network(s) performed. Do you consider the level of performance adequate to the real world task envisioned? Why or why not?

Task: To create 2 neural network that will classify the signal data provided in the DJI Phantom dataset. The first neural network need to classify: LOADED DJI Phantom from UNLOADED DJI Phantom or Nothing.

What we did: There were two neural networks that were created for this classification task on DJI Phantom. First neural network classified considering label 1.0 and label 0.0 as one Class (DJI Phantom and Nothing) and label -1.0 (Unloaded DJI Phantom) as another class. Second neural network classified considering label 1.0 for Unloaded DJI Phantom and -1.0 for other two Classes.

Our network achieved an accuracy on 96.075% on first neural network and 97.820% on second neural network.

Debating on performance evaluation:

- Since the data was obtained as an assignment, I am not aware of the environmental settings where the data has been generated.
- I consider that an accuracy above 97% is adequate for the real work task. This is because an accuracy of above 97% equivalent to the saying that 600 rows were misclassified over 200000 rows. Such error is acceptable but we can approach other models to enhance our accuracy.
- In the first neural network, I haven't performed dimensionality reduction whereas I performed dimensionality reduction in the second neural network, just to measure what difference in the accuracy is seen.
- In the process of the calculating the accuracy I haven't perform scaling or normalization on the dataset. Since the dataset is generated from a machine (DJI Phantom Drone), it is that same measurement scale have been used in recording the data. Scaling and normalization tasks are needed when the dataset has been created using data from different devices which uses different measurement scales.
- The neural network created for the completion of the task is a two-layer neural network. The first layer is a hidden layer consisting of varying hidden neurons and the second layer is the output layer consisting of one output neuron.
- We have taken one output neuron for our task because it is two class classification perform as stated above. If we need to classify into three classes, we might need 2 output neurons.
- For dimensionality reduction I have performed Principal Component Analysis on the second neural network with varying components. I am varying the components till half the size of the original dataset dimensions.
- In the second neural network, I observed that after setting the number of components to 7, we achieved an accuracy of 84% but it starts to decline between the range of number of

components 8 to 19. After which I observe increase in the accuracy (>88%) between the range of number of components 21 to 25.

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
1	2	100	2	0.700	69.232
1	2	100	3	0.700	83.038
1	2	100	4	0.700	84.081
1	2	100	5	0.700	84.025
1	2	100	20	0.700	88.840
1	**2**	**100**	**24**	**0.700**	**88.999**
1	2	100	25	0.700	88.946

Above table is for 2nd Neural Network. *This is because I performed dimensionality reduction in the 2nd Task mentioned in the assignment.*

Question 2: Design of an adequate neural architecture. Do you need multiple layers? If so, how many? How many neurons should there be in each layer?

Input layer is not considered as one of the layer of the neural network.

No, I don't think that I will be needing multiple layers in the neural network. I think following parameters need to be taken care of:
1. Learning Rate
2. Learning Momentum
3. Number of Folds
4. Epochs
5. Number of Hidden Neurons

First Neural Network:

- In order to test the accuracy of the neural network, I kept the number of hidden neurons as a varying parameter. I achieved a good amount of accuracy when I set the number of hidden neuron to 3.
- First neural network created for the task is a two-layer neural network with varying number of hidden neurons in the hidden layer and one neuron in the output layer.
- The number of hidden neuron is considered to be one of the changing parameter in this task of classification.
- Epochs and Learning Rate may vary with architecture. In this architecture, I have kept the Epochs = 100 and Learning Rate = 0.100

hidden neuron	Number of Folds	Epochs	Learning Rate	Accuracy
1	2	100	0.100	94.085
2	2	100	0.100	95.091
3	**2**	**100**	**0.100**	**95.567**

- As it is evident from the table, we achieved an accuracy of 95.567% when we had 3 hidden

neurons, keeping other parameters constant.

Second Neural Network

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
1	2	300	24	0.700	89.936
2	**2**	**300**	**24**	**0.700**	**97.802**
3	2	300	24	0.700	97.163

- In order to test the accuracy of the neural network, I kept the number of hidden neurons as a varying parameter. I achieved a good amount of accuracy when I set the number of hidden neuron to 3.
- First neural network created for the task is a two-layer neural network with varying number of hidden neurons in the hidden layer and one neuron in the output layer.
- The number of hidden neuron is considered to be one of the changing parameter in this task of classification.
- Epochs and Learning Rate may vary with architecture. In this architecture, I have kept the Epochs = 300 and Learning Rate = 0.700
- It is evident from the table, we have achieved a high accuracy of 97.8% when we keep 2 hidden neurons and keeping other parameters constant.

Question 3: How will I decide when my network(s) are sufficiently trained? How will I determine that the networks I have trained generalize well? What techniques, if any, will I use to encourage generalization?
Explain what training and validation preprocessing steps you took. Why did you take them? How to you presume your preprocessing would help your training?

For deciding whether my networks are sufficiently trained, I used to **epochs measure** to understand whether my network is trained or not. Below are the tables for changing epochs.

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
2	2	200	-	0.100	95.036
2	2	300	-	0.100	95.411
2	2	400	-	0.100	95.739
2	2	500	-	0.100	95.961
2	**2**	**1000**	**-**	**0.100**	**96.075**
2	**2**	**300**	**24**	**0.100**	**96.098**

Above table is for the first neural network. Initially, I didn't perform any dimensionality reduction in the first neural network. Since, I observe increase in accuracy by performing dimensionality reduction in the second neural network, I tried performing dimensionality reduction in first neural network which minutely increased my accuracy.

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
2	**2**	**200**	**-**	**0.700**	**96.763**
2	2	300	-	0.700	97.508
2	2	400	-	0.700	97.424

Above table is for second neural network. The reason for using dimensionality reduction in second neural network is that I achieve poor accuracy of neural network. There are cases when the network is unable to learn features since the feature size was 90. We tend to use dimensionality reduction when

- There is high variance across different random variables
- There is linear dependent relationship between different random variables.

These two factors didn't become visible while training the first neural network but were visible during second neural network training.

In order to make neural network well generalize we need to prevent overtraining or overfitting. Neural network is trained using Cross-Validation in order to make the network generalize. I have quantified the generalization of neural network using accuracy parameter.

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
2	3	100	-	0.100	93.849
2	**4**	**100**	**-**	**0.100**	**94.275**
2	5	100	-	0.100	93.380
2	6	100	-	0.100	94.180
2	7	100	-	0.100	94.066
2	8	100	-	0.100	94.090
2	9	100	-	0.100	94.173

It is apparent from the table that the neural network has learnt sufficiently when we set the number of folds to 4. *Based on the earlier analysis, I inferred that there is tradeoff between the number of epochs and number of folds.*

Question4: Explain why you chose the architecture you used. Note that this isn't any one universal answer to this question. There are, however, strategies you can use to narrow your choices. Some of these are mentioned in the book. Others can be gleaned from relevant literature available with a modest amount of research. How will I measure and report the quality of the trained network(s)? Should I care about false positives? False negatives? How should I report and interpret these situations and what relationships do they have to the real world problem?

First Neural Network with changing learning rate

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
2	**2**	**100**	**-**	**0.10**	**95.091**
2	2	100	-	0.09	94.619
2	2	100	-	0.08	94.078
2	2	100	-	0.07	94.059
2	2	100	-	0.06	94.021
2	2	100	-	0.05	93.933
2	2	100	-	0.04	93.753
2	2	100	-	0.03	93.489
2	2	100	-	0.02	92.940
2	2	100	-	0.01	92.239
2	2	100	-	0.009	92.172

Second neural network with changing learning rate

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
2	2	100	-	0.100	88.221
2	2	100	-	0.200	88.467
2	2	100	-	0.300	88.736
2	2	100	-	0.400	88.911
2	2	100	-	0.500	88.957
2	2	100	-	0.600	89.005
2	**2**	**100**	**-**	**0.700**	**89.015**
2	2	100	-	0.800	88.974
2	2	100	-	0.900	88.960
2	2	100	-	1.000	88.829
2	2	100	-	0.010	86.390

Regarding the methodology concerning accuracy, false positives and false negatives, I have employed as basic approach of "counting the hits". I am stating the accuracy based on this approach.

I am indifferent towards false positives as I am not a domain expert, who can validate the the data. There can be chances of machine error. Moreover, the data is generated from some external source whose confidence value is not known. Yes, I am concerned about false negatives as this defines the accuracy of my trained network. False Negatives are the cases in the testing results of the classifier where the classifier predicts that a feature set belongs to class A but in actual data it belongs to class B.

In the absence of gold standard data (human curated), false negative value defines the confidence

of trained network.
False negatives are considered as misclassification and are calculated as; 100 – Accuracy.
After achieves accuracy above 96% states that the network misclassified 4% of the samples. This is acceptable in those real time environments which does not involve human factors.

The architecture which I have used is a two-layer backpropagation neural network. First layer is a the hidden layer. The hidden neurons are taken as varying parameters in this experiment. Second layer is the output layer consisting of only 1 neuron. This is because we model this task as two class classification problem. Our architecture is simple in its formation but we have worked a lot on capturing the intricacies of the network such as epochs, #neurons, learning rate, split cross validation and dimensionality reduction.
There are other powerful neural network architectures such as n-layer neural network, extreme machine learning neural network, Deep neural network, Convolutional neural network etc. These heavy architectures used simple neural network at atomic level. We use these architectures for pattern recognition, object detection, high dimensional and complex data analysis.
Another class of neural network are feed forward neural network and perceptron. Using both the neural network architecture has given 69% and 69.232% accuracy. These architectures have a tendency to fall into convergence issues.

Question 5: What training termination conditions did you use? Why did you use them? Explain how you chose to measure performance of your final products, including how you determined that the final products do indeed generalize to an acceptable level.

Two convergence conditions such as:

1. Error Difference
2. Epochs
3. Decrease in testing accuracy.

I have used epochs as termination conditions. Epochs don't affirm that the neural network has attained a convergence rate which can be done using Error Difference thresholding. *In order to test the neural network under different conditions, I used epochs and "Decreasing Testing accuracy" as a measure for telling the performance of the network.*

Varying Hidden Neurons, Epochs, and Number of Folds with constant # PCA Components based measuring the performance of my final product (2nd Neural Network)

hidden neuron	Number of Folds	Epochs	PCA Components	Learning Rate	Accuracy
1	2	200	24	0.700	90.164
1	2	300	24	0.700	89.936
2	2	300	24	0.700	97.802
2	**3**	**300**	**24**	**0.700**	**97.602**
3	**2**	**300**	**24**	**0.700**	**97.163**
2	**2**	**400**	**24**	**0.700**	**97.819**
2	**2**	**500**	**24**	**0.700**	**97.660**

Black: Baseline **Purple: changed epochs** **Red: changing #hidden neuron**
Orange: Changing Number of Folds

We have used epochs and "Decrease in Testing accuracy" as the two measure to test whether the network has *underfit* or *overfit*. **Underfit** is the condition when the network do not learn more than what it has learnt but that learning is not complete. **Overfit** is the condition when the network tries to learn from noise rather than data. These two measure we have used are termed as *Early Stopping Measure* which prevent from regularization of the network. It is this regularization job that Dropout Algorithm and Weighted Decay Algorithm focusses on.

As stated above, I have used hits based accuracy measure. After modifying the labels in the dataset with respect to the task, we just equated the prediction of the classifier with the actual value for calculating the accuracy.

We determined that the network has attained a sufficient level of generalization based on a series of experiments. These experiments are: (while performing these experiments we kept other hyper-parameters as constant)

1. Changing the number of hidden neurons
2. Changing the Number of Folds
3. Changing the Epochs
4. Changing number of PCA components
5. Learning Rate.

Varying hidden neurons, number of folds, epochs and learning rate in First Neural Network

hidden neuron	Number of Folds	Epochs	Learning Rate	Accuracy
2	2	200	0.100	95.036
3	2	200	0.100	95.567
2	3	200	0.100	93.849
2	2	400	0.100	95.739
2	2	200	0.080	94.078

Black: Baseline **Purple: changed epochs** **Red: changing #hidden neuron**
Orange: Changing Number of Folds **Blue: Changing learning rate**