Single Layer Perceptron and MLP Analysis for Gorman and Sejnowski Sonar Dataset.

By
Rishab Goel,
Electrical and Computer Engineering Department,
University of Florida

Abstract— **This document describes the analysis of training the multi-layer perceptron for Gorman and Sejnowski Sonar Dataset. A single hidden layer network with varied processing elements is compared with single layer perceptron for classification accuracy for train and test dataset using the confusion matrix.. This document also explores the nature of the dataset and how it should be arranged for better training of the network. It discusses how the parameters of the neural network like the learning rate, momentum, batch size and stopping criterion are varied and their impact on accuracy of the classification. We also present the initial and final weights and biases for the best trained network configuration for both single later perceptron as well as single hidden layer network.**

Index Terms—**Multilayer perceptron, Single Layer Perceptron, Sonar Data Set.**

I. Introduction

The Multilayer perceptron network is a feed forward artificial neural network that projects an input multi-dimensional space to a new subspace. The network could have multiple hidden layers with different number of processing elements and each processing element of the network uses a non-linear activation to project the input space .

$y = f(\sum wij*xi + b).$ (1)

The f function is the non-linear activation function. The network iteratively learns the weights to project the outputs closer to the desired output response. The weight update between each epoch for each processing element by:

$wij(n + l) = wij(n) + \eta * \Delta wij.$ (2)

;where η is the learning rate.

The Δwij gets updated by multiplication the gradient of the activation function, a sum of local errors δk at each network output PE, scaled by the weights connecting the output PEs to the ith PE and output of the activation (xj). The weight update is controlled by the free parameter learning rate. The stochastic gradient is the method used to train a multi-layer perceptron neural network.

The training of a neural network is not only a time consuming process, but also it suffers from a generalized stop criterion for training. There are some suggested tips and tricks to arrive at a stop criterion, but this is mainly governed by the data used for training. Choosing the number of hidden layer and the number of processing elements in each layer. For any learning algorithm, the control of the free parameter is most crucial, so controlling the learning rate (η) is the key of arrives at the desired minimum.

The dataset available for the analysis for the multilayer perceptron for the project is the Gorman and Sejnowski [1] sonar returns of the undersea metal cylinder and the rock of the similar size. We need to classify the dataset of sonar data into rock or a metal. The target rock or the metal both were 5ft in length and the sonar pulse was a wide-band FM chirp signal. The sonar returns collected from the distance of 10 meters with the aspect angles spanning 90 degrees for the cylinder and 180 degrees for the rock. The total samples of the data obtained are 208 with 97 of them being in the rock and 111 being for the metal. The data have an input dimension of 60 inputs and preprocessed and normalized between 0 and 1.0.

We compare the single layer perceptron and single hidden layer network in our analysis. The Methods section (II) enumerates the methods for data arrangement, neural network configuration, neural networks' parameter and stopping criterion. This section only introduces the method used in the experimentation and impact and reasons behind this are discussed in the Discussion section (IV). The results section (III) details the best results obtained online and batch training configuration on the basis of confusion matrix and accuracy of classification. The single layer perceptron and single hidden layer network are compared in this results section. The Discussion section (IV) also inferences the results listed in section (III) and presents the Hinton diagram for the weights and biases of the single layer perceptron and single hidden layer network. Finally the conclusion (V) concludes the analysis with lessons learnt and scope for further optimization.

II. Methods

A .Data arrangement methods

Gorman and Sejnowski [1] uses two data patterns for the analysis: aspect-angle independent and aspect-angle dependent data series. The aspect angle independent series data pattern divides the data samples into 13 disjoint sets each containing 16 data samples and remaining 192 data samples is the training sets. It takes iteratively each 13 dataset as test sets and trains the neural network with the same parameters. The aspect angle dependent series data pattern they divide the test and train dataset into 104 and 104 data samples each. They claim to attain better performance with aspect-angle dependent series data pattern. Our selection of ratio of test and train data sample more closely relates to the angle dependent series data pattern.

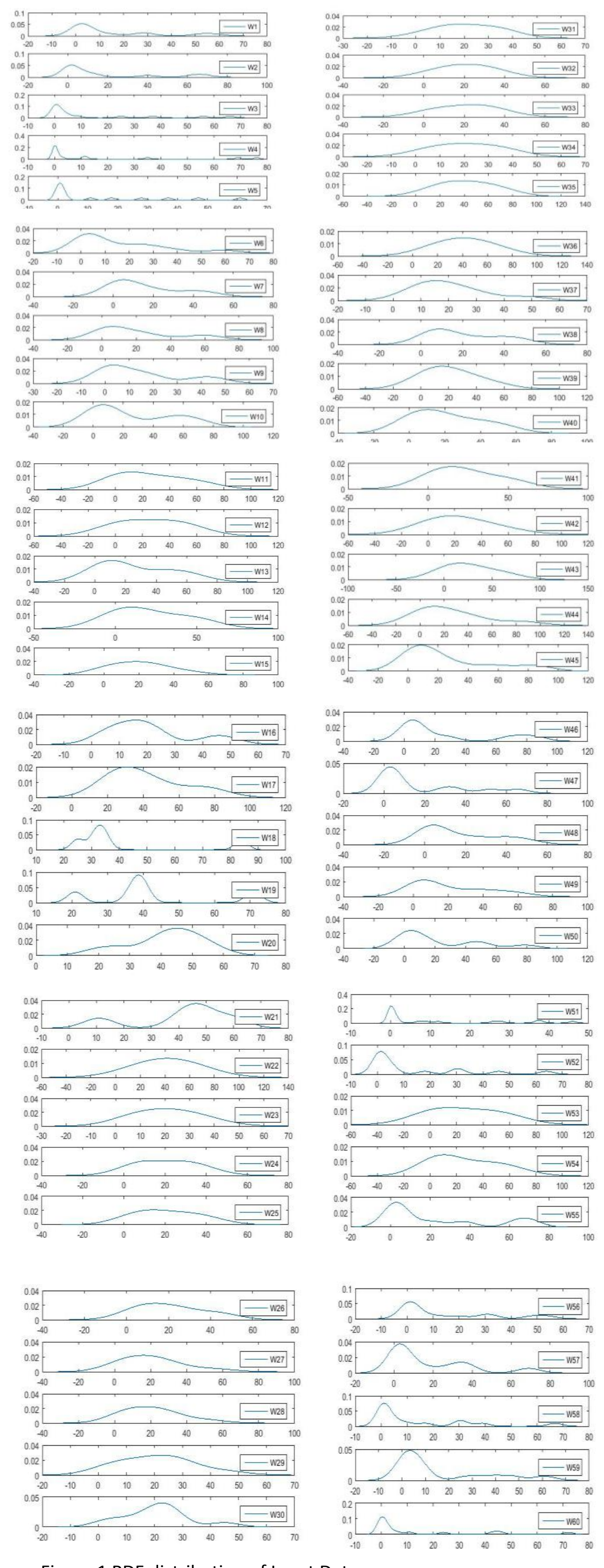

Figure 1 PDF distribution of Input Data

It becomes necessary to analyze and visualize the dataset before arriving to a desired methodology for the training. We individually take each dimension of the signal and analyze its histogram and fit to its pdf distribution. Figure 1. We observe that for all the 60 dimensions of the data the pdf distribution projected can be approximated by the Gaussian distribution. This plot gives intuition to use stochastic gradient descent with least square loss function algorithm for the weight updates and train the network.

It is to be noted that though the data is of 60 dimension the sample size is just 208 samples with 97 samples for the rock and 111 samples for the metal. The dataset could thus be attributed to be small dataset, so the test dataset and train dataset need to be chosen judiciously. For our current analysis of the dataset we divide 2/3 sample of the dataset of 208 samples for training and remaining 1/3 for classification. For sampling dataset we ensured that the samples in the test dataset are proportionally distributed among the two classes from the original dataset. This is ensured using partitioning the test and train samples exclusively from the data of each class.

We tried to sample the data in mainly three ways:

A) Initial 2/3 of Rock Class and Initial 2/3 of Metal class as the train data and rest 1/3 of each class as the test data.

B) For both Rock and Metal sample consecutively 2 samples were put in train data and next 1 sample was put in test data. This process was repeated until 208 samples are divided into test and train groups

C) We take random 2/3 sample from Rock class and random 2/3 sample from the Metal class and remaining 1/3 samples constitute the test class.

It is believed that method 3 random sample may give more wide data pattern samples which may or may not always give best results always. It also may trap the general probability density distribution of the data closer to probability density function of each dimension. The other two data arrangement are actually two predicted patterns of data that could be give better results. First one takes the initial samples while the other one takes samples with a stride across the whole dataset.

We also tested with cross validation data set using 10% of the training set as validation dataset. The 10% validation dataset was chosen randomly from the trained dataset and is expected to give better generalization. We also experimented with shuffling the train dataset at each epochs for generalization.

B. Training Methods

Two methods were used for training the dataset: single layer perceptron and single hidden layer perceptron. The single layer perceptron method could be applied to a linear separable dataset successfully. Though from the literature survey analysis of the methods applied from the Sonar Dataset of Rock and Metal it seems the data is not linearly separable. We arrive at this conclusion as the training and testing dataset accuracy attained on average over 10 iterations was 79.3% and 73.1% respectively [1]. But since the data arrangement and distribution is different for us compared to Gorman and Seljowski, so we want to observe the performance for our data pattern for single layer perceptron as well. The other method chosen for analysis is a single hidden layer neural network and the number of processing elements were chosen were varied from 2, 4, 8, 12 , 24 , 30, 36. According to literature survey of Gorman and Seljnowski, 12 and 24 processing elements neural network gave good test data accuracy with 12 PEs being slightly more stable. We tested few instances for 2 hidden layers considerable improvements so we limited our analysis to single hidden layer only. Though keeping in mind both paper [1] and [2] we focus more on 24 PEs hidden layer network.

C. Parameters for training

We use sigmoid function only for the activation function and stochastic gradient method for backpropagation. The batch size was varied as 1, 3, 5 for a single layer perceptron and single hidden layer perceptron.

The learning rate parameter was varied as 1.0 , 1.5 and 2.0 with adaptive variation with respect to epochs using the expression:

$$\eta (n)=\eta_0/(1 +n/n_0) \qquad (3)$$

;where n_0 is a parameter determined by experimentation and n is the value of the iterations of each epochs. We also added momentum term to our training algorithm varied it for 0.0 and 0.01 for smoother convergence of the weights.

The initial weights & biases were randomly chosen for a uniform distribution ranging from -0.3 to 0.3. The range for the weights was inducted from the research paper [1] of Gorman & Slejowski which turns out to be equivalent to:

$$[-\sqrt{\frac{6}{fan_{in}+fan_{out}}}, \sqrt{\frac{6}{fan_{in}+fan_{out}}}] \qquad (4)$$

This formula for random weights initialization basis your weights & biases initialization on the dimensions of the inputs and outputs of the dataset which seems to be good initial condition for the weights. We also ran the tests with and shuffling of data pattern for each epochs.

D. Stopping criterion

We employed a crude stopping criterion to reach up to maximum 300, 600 or 700 epochs in order. We kept mean square error threshold of 0.01(online) as well as 0.001(batch) but didn't go further below it to avoid overfitting of the data to training dataset.

To reduce the chance of overfitting we tried to do an early stop on the basis of minimization of the mean square error for validation dataset. We stopped as soon as the mean square error for the validation dataset starts increasing. The weights used for test the trained and test data set was the weights with the minimum square error for the trained data. .We implemented an additional variant for dataset analysis using the validation where we save the weights for the least mean square error obtained for validation dataset, but don't stop on its basis. We keep tabulating the least weights for the validation data set until minimum square error for trained data goes below the threshold or maximum epochs expires. These least weights are later used for testing and generation of the confusion matrix.

III. Results

The Initial results shown are for online training for single layer and single hidden layer perceptron. The confusion matrix are used to present the classification prediction of the training methods x, y indexes of the matrices representing actual and predicted class's labels. Rock class label is given 0 so correct prediction goes (0, 0), while for Metal class label is 1 so it is correct prediction are listed in (1, 1). The (1, 0) and (0, 1) depicts wrong prediction for metal class and rock class respectively. It is to be noted that since the dataset is small 1 or 2 correct and wrong prediction are reflected drastically in the accuracy.

The results detailed in Figure,2, 3 & 4 are the online training accuracy for 0 12 and 24 processing elements with learning rate being 2.00 (adaptive n_0 = 400) momentum= 0.01 and mse threshold = 0.01.

19	3
13	34

36	0
29	74

18	1
14	36

39	1
20	50

76.81% 79.13% 78.26% 81.73%

(a) (b) (c) (d)

Figure 2

Fig 2 (a),(b) Maximum Test and Train Confusion Matrix for single layer perceptron with no cross validation respectively with Accuracy% mentioned below. (c),(d) Maximum Test and Train Confusion Matrix for single layer perceptron with 10% cross validation respectively with Accuracy% mentioned below. Cross Validation method has slightly better accuracy compared to no validation.

26	1
6	36

54	0
11	74

26	2
6	35

54	4
4	63

89.85% 92.08% 88.40% 93.6%

(a) (b) (c) (d)

Figure 3

Figure 3 (a) ,(b) Maximum Test and Train Confusion Matrix for 12 PEs in 1 hidden layer network with no cross validation respectively with Accuracy% mentioned below. (c),(d) Maximum Test and Train Confusion Matrix for 12 PEs in 1 hidden layer network with 10% cross validation respectively with Accuracy% mentioned below. Cross Validation method has slightly better accuracy compared to no validation.

26	2
6	35

61	1
4	73

25	1
7	36

48	0
10	67

88.40% 92.08% 88.40% 92.00%

(a) (b) (c) (d)

Figure 4

Table 4 (a) ,(b) Maximum Test and Train Confusion Matrix for 24 PEs in 1 hidden layer network with no cross validation respectively with Accuracy% mentioned below. (c),(d) Test and Train Confusion Matrix for 24 PEs in 1 hidden layer network with 10% cross validation respectively with Accuracy% mentioned below. Cross Validation method didn't impact the maximum accuracy obtained.

The

It is observed that maximum accuracy for the hidden layer network is good but average per sample or average over random samples for the accuracy didn't give any enthusiastic results for hidden results for the online.

TABLE I. ONLINE TRAINING VARIED PEs ACCURACY ANALYSIS

# of PEs	**Avg Train Accuracy %**	**Train Standard Deviation %**	**Avg Test Accuracy %**	**Train Standard Deviation %**	**Max Accuracy %**
0	77.92	5.1	70.63	4.33	79.13
2	85.06	4.2	77.19	3.45	85.19
4	86.43	5.1	74.60	4.35	83.18
8	87.10	6.1	77.67	2.43	84.0
12	85.10	10.34	78.50	7.4	89.85
24	86.71	9.3	75.50	8.9	88.40
30	85.1	9.1	74.7	5.2	83.50
36	86.10	7.1	77.69	5.7	91.3

Table I displays average accuracy over training and testing dataset along with standard deviation and maximum accuracy obtained for the online training over 30 random weights and for 20 instances of random data samples from the train dataset so for 600 instances of training.

It is to be noted that stopping criterion used for the above values was mean square error threshold of 0.01 and maximum epochs were set to be 100. The learning rate used was η_0 2.0 with adaptive learning rate with experimentally chosen to be 400. The momentum was set to 0.01.

Using the weights with lowest mean square error on the cross validation dataset varied the maximum accuracy from -2.0 % to 2% from the maximum accuracy attained in without validation. Though the average of each configuration got pushed maximum by 3%. The highest average accuracy attained for testing dataset using the method mentioned **79.5%** for the hidden layer network of 24 processing elements.

46	4
19	70

83.45%
(a)

65	0
0	74

100.00%
(b)

27	5
5	32

85.50%
(c)

28	1
4	36

92.75%
(d)

Figure 5

Figure 5 (a) ,(b) shows Maximum Train Data Confusion Matrix for single layer perceptron and 24 PEs in 1 hidden layer network with no cross validation respectively with Accuracy% mentioned below. (c),(d) Test Confusion Matrix for single layer perceptron and 24 PEs in 1 hidden layer network respectively with Accuracy% mentioned below. Results shown are for batch of size 5 with adaptive η starting at 0.7 and momentum 0.01 with max epochs 800 and mean square error threshold 0.001.

TABLE II. SINGLE LAYER PERCEPTRON VS SINGLE HIDDEN LAYER (24 PEs)

# of PEs	**0**	**24**
Max Test Accuracy Batch =1	79.3%	88.40 %
Max Test Accuracy Batch =5	85.50%	92.75 %
Best Avg. Test Accuracy Batch=1 for Single Train data of 50 Random Weights	77.19%	83.40 %
Best Avg. Test Accuracy Batch=5 for Single Train data of 50 Random Weights	85.41%	89.56%
Avg. Test Accuracy Batch=1 for Random Data sample	70.63%	79.13 %
Avg. Test Accuracy Batch=5 for Random Data samples & weights	79.79%	85.60%

Table II compares the test set accuracy for single layer perceptron and 24 PEs single hidden layer network. Note that batch size 1 has learning rate as 2.0 (adaptive n_0 = 400), momentum 0.01 and mean square error threshold 0.01. Results shown for batch size 5 with adaptive η starting at 0.7 (adaptive n_0 = 400), momentum 0.01 and mean square error threshold 0.001.

We achieved the best accuracy for 24 PEs data samples of 92.75% and maximum per data sample average accuracy being 89.56% and maximum overall average being 85.60%.

IV. DISCUSSION

A. Experiments and Results analysis.

The three data arrangement as mentioned in the section II were used and iteratively tested for varied number of processing elements, learning rate and batch sizes. The data arrangement (A) and (B) were not at all crossing 75% accuracy for the test data set for any random combination of weights with modulation in the above mentioned parameters. The average accuracy just hovered just around 70%. This clearly indicated these were not desired data arrangement to achieve

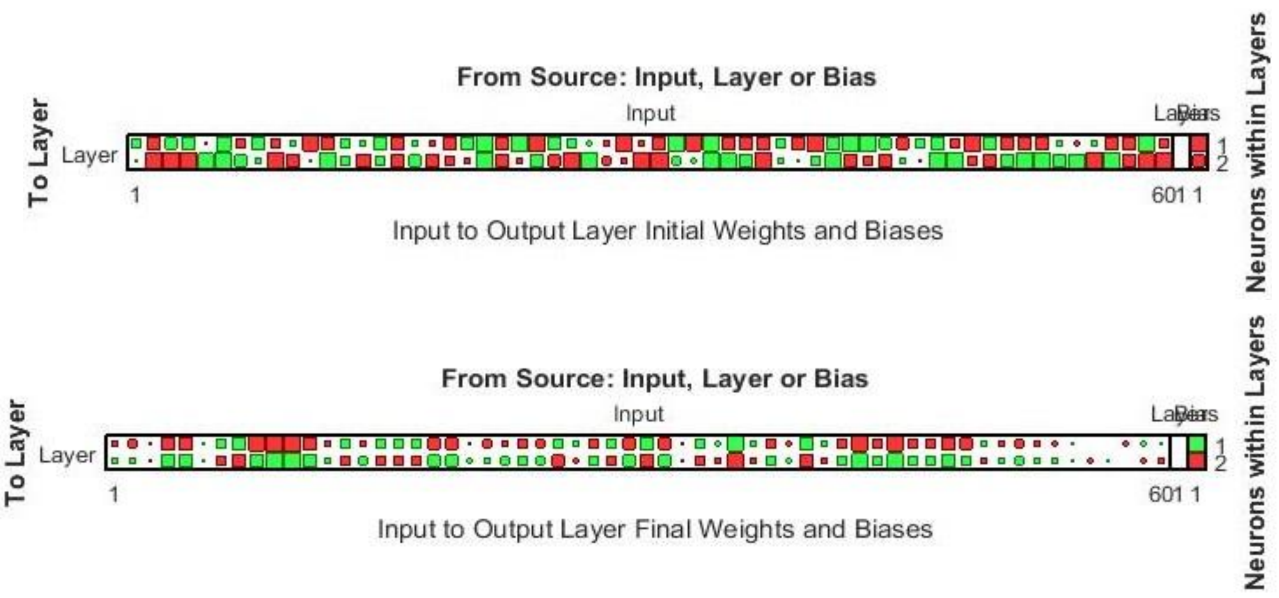

Fig 6 Hinton plots for initial and trained weights & biases for Single layer Perceptron.

accuracy close to 90% [1] or above 90% [2].

Our analysis started with the literature survey [1] and initial network configuration were based on [1]. Methods used in [2] were attractive, but used aspect-angle independent data sampling while we were using aspect-angle dependent data sampling. Moreover, they initially attempt to reduce the data dimensionality using principal component analysis followed by classification. So we initiated our experiments with the network configuration suggested by the analysis of Gorman and Seljwoski [1] for batch size as 1. As mentioned in Table I we achieve an accuracy of close to 90% using 24 and 36 processing elements in a single hidden layer for random data samples for a particular initial weights. The single layer perceptron ((# of PEs = 0), achieved good average and peak accuracy compared to a single hidden layer.

The mean square error threshold was set to 0.01. When we decreased it further for online training sets the test data accuracy reduced, while train data accuracy increased due to overfitting the model to the trained data. The learning rate parameter was increased from 1.0, 1.5 to 2.0, which resulted in the convergence of the network increase in accuracy. The learning adaptation on the basis of formula (3) was added along with momentum for smooth convergence of the network. They both reduce repeated to and fro bouncing around the actual minima of the network by modulating the learning parameter impact on weights. The n_0 value was arrived at 400 purely by iterative testing on random data and weight samples.

The stopping criterion for the network was tried to be generalized using an early stop using 10% of training data for cross-validation. The training of the model would terminate as soon if the mean square error for the validation dataset increased instead of decreasing. The weights that had least mean square error for the validation data set were used for testing the accuracy. The early stop methodology dropped the performance very badly, the maximum accuracy for the test data falling in the range of 60-65% while average reaching closer to 50%. The small size of the dataset was the reason that minimized the chances of early stop on the basis of the validation dataset. We eliminated the early stop using validation dataset instead we compute least mean square each time.

The other technique using the weights with minimum mean square error for the checking the accuracy for training and test data set as mentioned give some improvements. This was because of the reason that evaluating the trained weights on separate data from the train dataset gives the generalization to the weights which may lead to slight decrease in train data accuracy, but would lead to increase the test data accuracy. We observe that in fig0 and fig 12 for 0 and 12 number of PEs that accuracy was slightly better than no validation accuracy. Though in fig 24 we don’t observe this impact, but this accuracy improvement depends upon the choice of the validation dataset. We randomly select the validation data set so we could say that for 24 PEs the validation dataset sampled

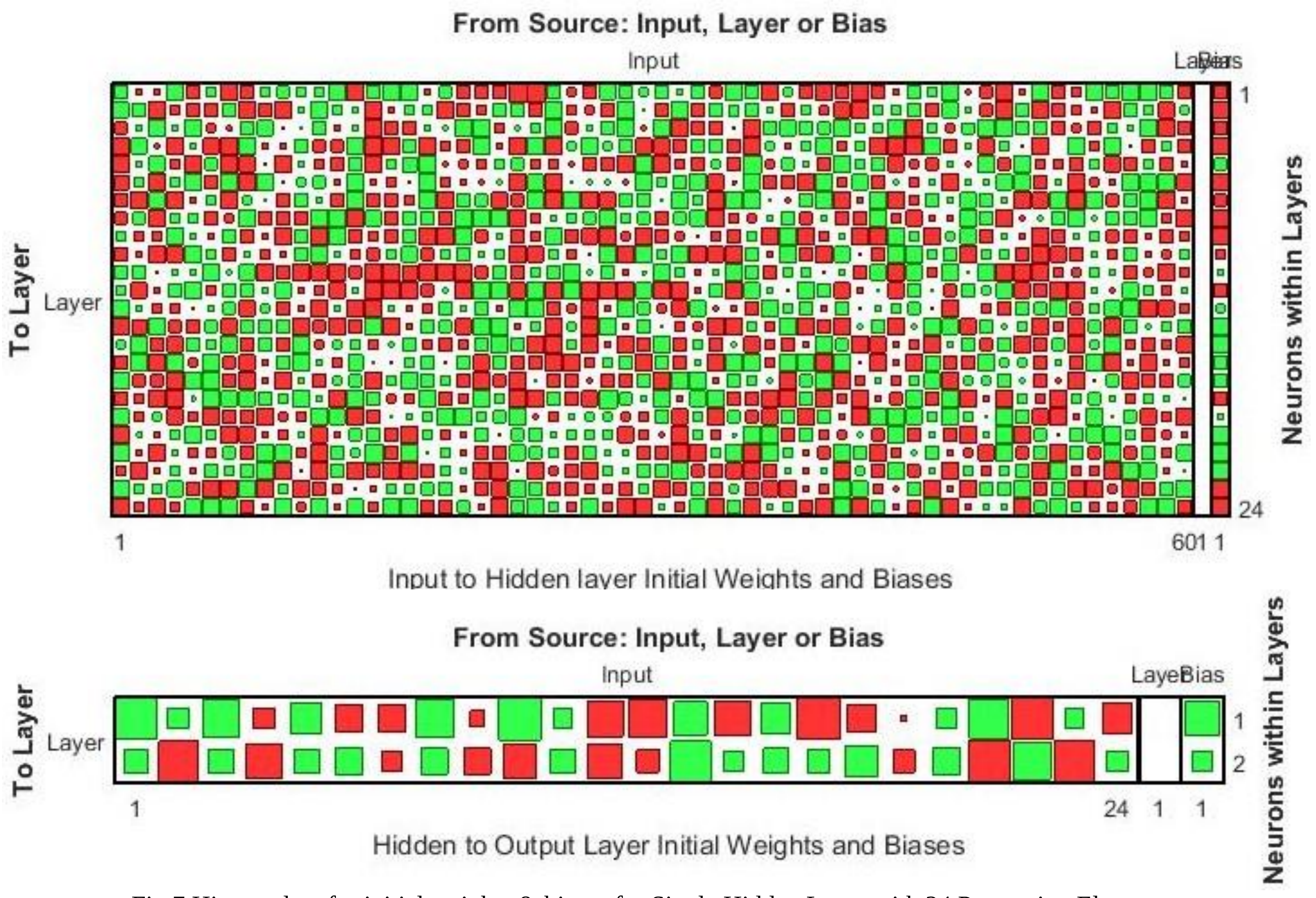

Fig 7 Hinton plots for initial weights & biases for Single Hidden Layer with 24 Processing Elements.

out was good to obtain generalization. Though as mentioned before this generalization boosted the average accuracy for all PEs mentioned in Table 1 with 24 PEs configuration getting the maximum boost. The shuffling of the dataset at each epoch seems to be ineffective in fact in tends to increase the epochs to converge.

The batch training with the same learning rate didn't give attractive results, so on suggestion from the [3] we minimized the learning rate by the increase in the batch size. The learning rate got reduced to 0.4 initially and then increased to 0.07 to get better results. We also reduced the mean square error threshold to 0.001 for the batch as it get us to a better accuracy for both test and train dataset. The learning rate 0.7 with n_0 = 400 along with momentum 0.01 attained the sweet spot.

The confusion matrix of fig 2, 4 & 5 proves the fact that 24 PEs has better accuracy than the single layer perceptron. For batch training the best accuracy of the single layer perceptron closer to 80% (table 2) gives an idea that the data is linearly separable to an extent for particular train and test data samples. There was considerable improvement in the average of 24 processing elements case to 85.6% and best prediction of 92.60%. This clearly indicates that particular data sample when used as input and are transformed a new subspace could give even better accuracy. This could be seen in paper [2] which achieved 97.3% accuracy using angle independent data series with principal component analysis for feature extraction and 24 PEs single layer MLP for classification.

B. Hinton diagrams of weights and biases

We published the Hinton plots of the initial and final weights & biases of single layer perceptron with best accuracy in figure 6 .Figure 7 and 8 have the Hinton diagram of the initial and final weights & biases for the best accuracy (92.50%) 24PEs configuration. Only input weights and biases to a particular layer are represented in these plot and layer (middle) weights don't carry any information for these plots. The green color indicates the positive value while red negative values of the weights. The area of each red and green block is directly proportional to the absolute value of the each weights. The number of such blocks are number of weights and biases connected to layer with number of rows indicating the number of processing elements in the layer and columns indicates the input dimension attached to the layer. In Figure 6, Input weights and final weights have dimension 2x60 and biases 2x1 which indicates 2 outputs PEs and 60 is the dimension of the input. In Figure 7 and 8, first plot shows initial and final weights and biases for input to hidden layer respectively, weights being 24x60 and biases 24x1 which indicates 24 hidden PEs and 60 is the dimension of the input. In Figure 7

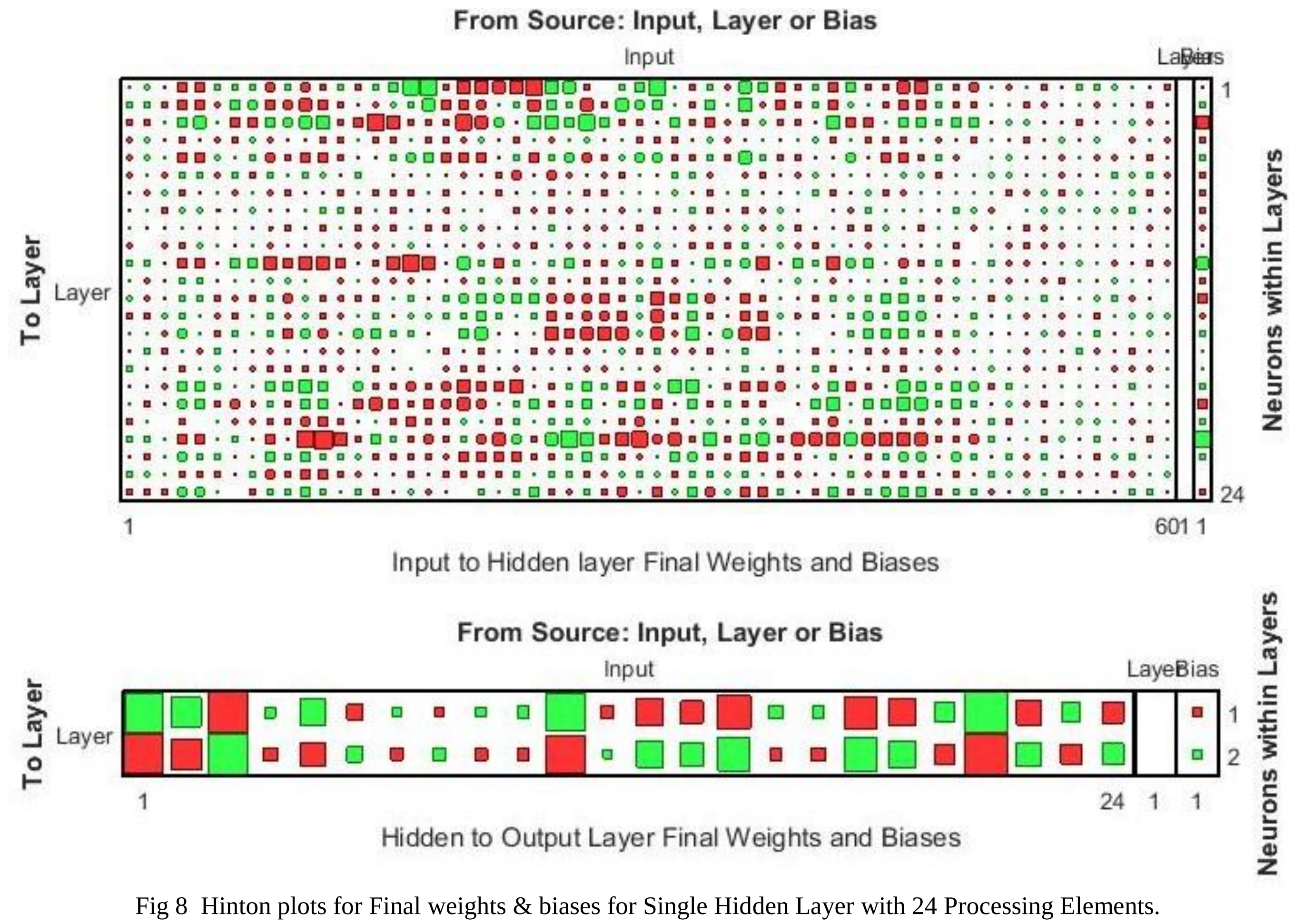

Fig 8 Hinton plots for Final weights & biases for Single Hidden Layer with 24 Processing Elements.

and 8, first plot shows initial and final weights and biases for hidden layer to output layer respectively, weights being 2x24 and biases 2x1 which indicates 2 outputs PEs and 24 outputs of the hidden layer. The initial weights are uniformly distributed form -0.3 to 0.3 as mentioned in section II c. Figure 8 indicates the weights which got us the best accuracy seems to be concentrated more around zero value, while the biases have higher positive or negative values for the input to hidden layer subspace projection, while for hidden to output layer projection the weights are more evenly distributed and low biases.

V. Conclusion

It could be concluded that for the sonar dataset the MLP network with 24 processing elements and a single hidden layer achieves best accuracy of 92.50%. This indicates single hidden layer can separate this sonar data set very successfully and for achieving higher accuracy increasing the depth of the network would be computationally intensive option. The single layer perceptron the peak accuracy shoots up to 85.50% while on average for large random data samples and weights was just under 80%. This accuracy of the single layer perceptron for the aspect angle dependent data pattern of the sonar dataset being partially linearly separable.

The batch training gives better results than online training but with a lower learning rate which removes the ambiguity the batch training is very less computationally intensive than online training. It also suggests that batch training tends to trap the global data pattern better, while online may overshoot that global pattern and not converge to the best result. The adaptive learning rate with momentum smoothens out the weights updates and in turn giving a better trained weights. While, the validation dataset may increase generalization depending upon the chosen validation dataset sample.

Though the lack of achieving a concrete stop criterion seems one reason to not achieve higher peak accuracy and crossing the 90% mark for the average accuracy. The data preprocessing using principal component analysis suggested in [2] seems a good addition to our best network parameter configuration to achieve better peak accuracy.

References

[1] Gorman and Sejnowski: “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets,” Neural Networks. Vol 1, Pergamon Journals, 1988

[2] Experiments on Gorman and Sejnowski’s sonar data Markus Mannevaara Johan Jilderin December 3, 2001.

[3] D. Randall Wilsona, Tony R. Martinezb “The general inefficiency of batch training for gradient descent learning” Computer Science Department, Brigham Young University, Provo, accepted 8 April 2003

[4] http://www.is.umk.pl/projects/datasets.html

[5] https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)