COMP/ELEC/STAT 502 HW 05

Kentaro Hoffman

March 20, 2016

Problem 1

Table 1: Parameters of Training BP Network to Fit Iris Data Set from the Previous HW

Network parameters	
Topology	$(1 + 1_{Bias})$ — $(2 + 1_{Bias})$ — 1
Transfer function	Hyperbolic Tangent with slope of 1
Learning parameters	
Initial weights	drawn from U[-0.1, 0.1]
Learning rate (α)	0.02
Momentum	0.5
Epoch size ($Epoch$)	200
Stopping criteria	error (Err_{train}) ≤ 0.04 OR learn count (t) $> 225,000$
Error measure (Err_{stop})	$Acc_X = \frac{\#correct\ \ hits}{\lvert X \rvert}$ % See formula (2) below
Input / output data, representation, scaling	
# training samples (N_{tr})	75
# test samples (N_{tst})	75
Scaling of inputs	None
Scaling of outputs	None
Parameters and error measures for performance evaluation	
Error of fit (Err_{fit})	$Acc_X = \frac{\#correct\ \ hits}{\lvert X \rvert}$ %
Final Error of fit for Testing Data	0.0267
Final Error of fit for Training Data	0.000
# learn steps performed	1,725 (threshold for Err_{stop} reached)
Monitoring frequency (m)	150 learning steps

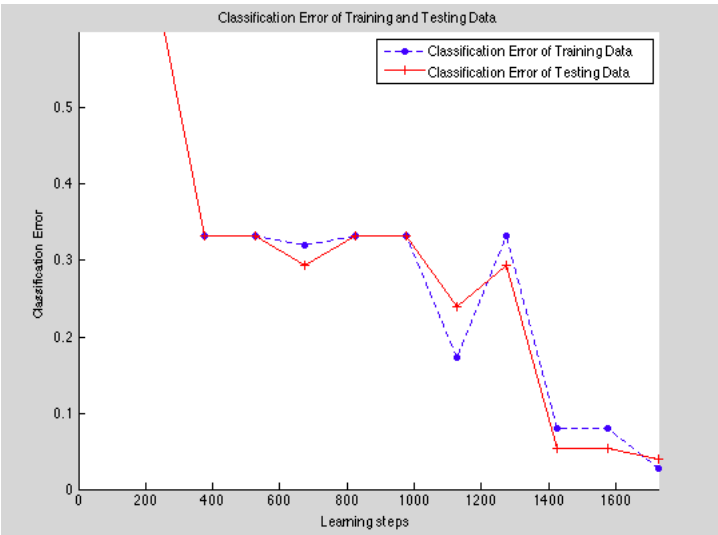
To transition to the Iris dataset, a few modifications were necessary. First was a change in Error measure from mean absolute error to classification accuracy. Specifically, the concept of a correct hit had to be established. Designating an output y^k to be giving the correct response only when $y^k = D^k$ did not lead to any form of convergence. Hence, the following function was applied to the outputs.

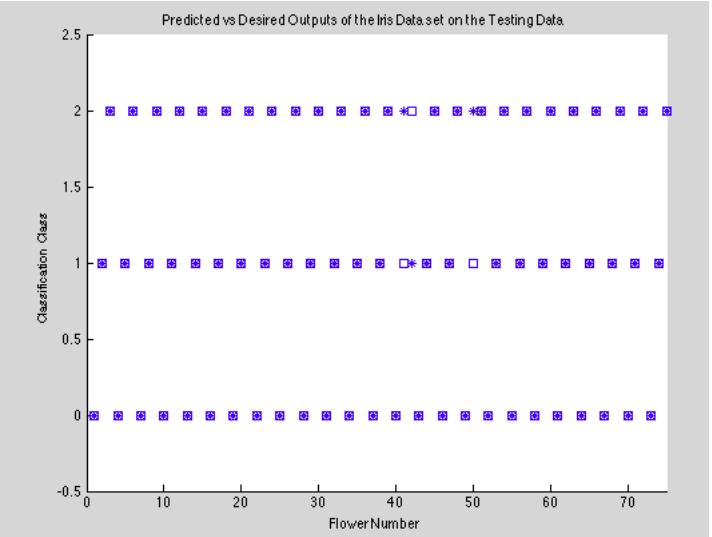
Let $D_y^k = \min(d_1^k, d_2^k, d_3^k)$. Where d_1^k is the distance from y^k to [1,0,0], d_2^k is from y^k to $[0, 1, 0]$, and d_3^k is the euclidean distance to $[0, 0, 1]$, the thresholded output, $\hat{y^k}$ is equal to $[1, 0, 0]$ if $D_y^k = d_1^k$, $[0, 1, 0]$ if $D_y^k = d_2^k$,

and $[0, 0, 1]$ if $D_y^k = d_3^k$. Ie it maps every output to the closest possible output. Then when, $\hat{y^k} = D^k$, we would designate this as a correct hit.

So in summary, an output is a correct hit if the point that is closest to the output is the desired output. When another possible point is closer, it is an incorrect hit.

Without Cross Validation



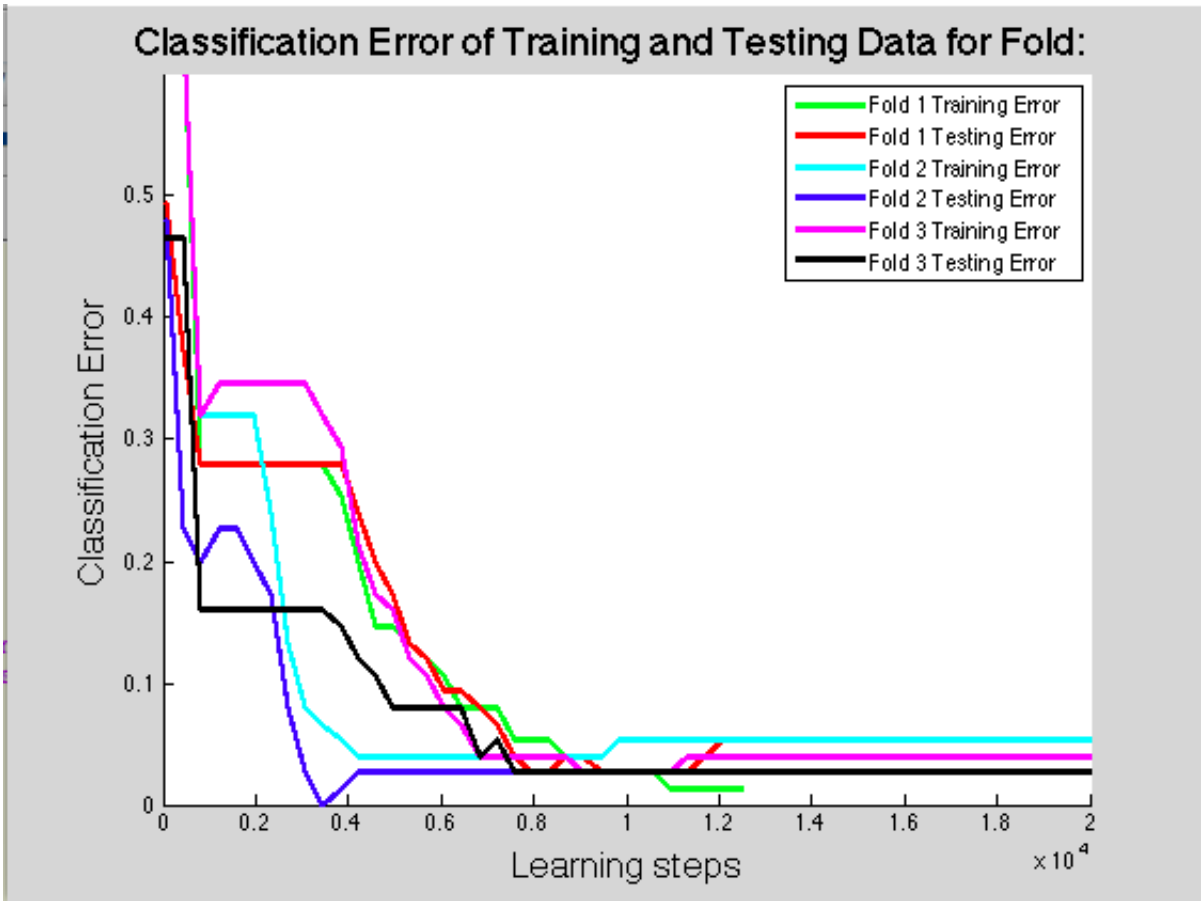


The plots above show the results from the previous homework where no cross validation was applied. The important things to note are how the simulation manages to converge very quickly (1725 steps) and it manages to achieve a respectable 4 percent (3/75) classification error. With k-fold cross validation, if we see that the time till convergence massively increases or if the classification error increases, then it seems likely that the model above is not as useful as it first seems as it has overfitted.

Table 2: Performance Evaluation for Fold 1

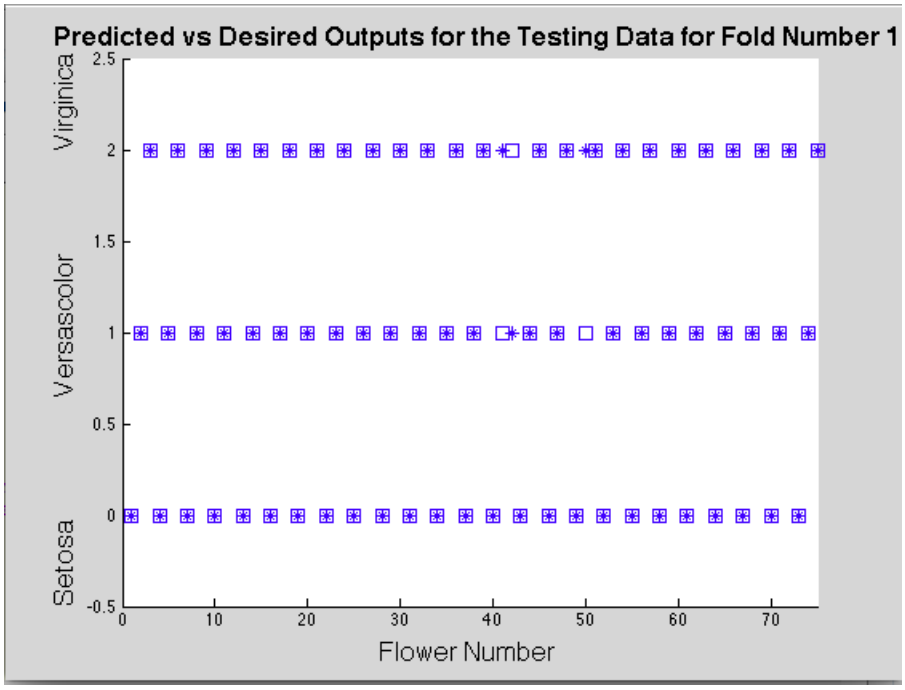
Parameters and error measures for performance evaluation	
Error of fit (Err_{fit})	$Acc_X = \frac{\#correct\ \ hits}{\|X\|}$ %
Final Error of fit for Testing Data	0.0267
Final Error of fit for Training Data	0.0400
# learn steps performed	1,725 (threshold for Err_{stop} reached)
Monitoring frequency (m)	150 learning steps

Learning History



Though it may seems convoluted, there are sevearl things that can be noted about the how the ANN is learning. The first is that unlike the previous homework, very few of the folds seem to converge very quickly. They are all downward trending, which means that they are learning the data better, but the fact that 4 out of the 6 (Fold 2 Train, Fold 2 Test, Fold 3 Train and Fold 3 Test) do not terminate, ie get a classification error on the training error of less than 0.04 seems to indicate that these parameters are having a hard time generalizing to the other folds. However, this does not mean that this was useless. On the contrary, the code takes much longer to run, but it does seem to provie outputs that produce good accuracy.

Fold 1



Fold 1 Evaluation	
Final Error of fit for Training Data	0.000
Final Error of fit for Testing Data	0.0400
# learn steps performed	12,750

Table 3: Confusion matrices for training and test data for for Fold 1

(a) Training data, Fold 1

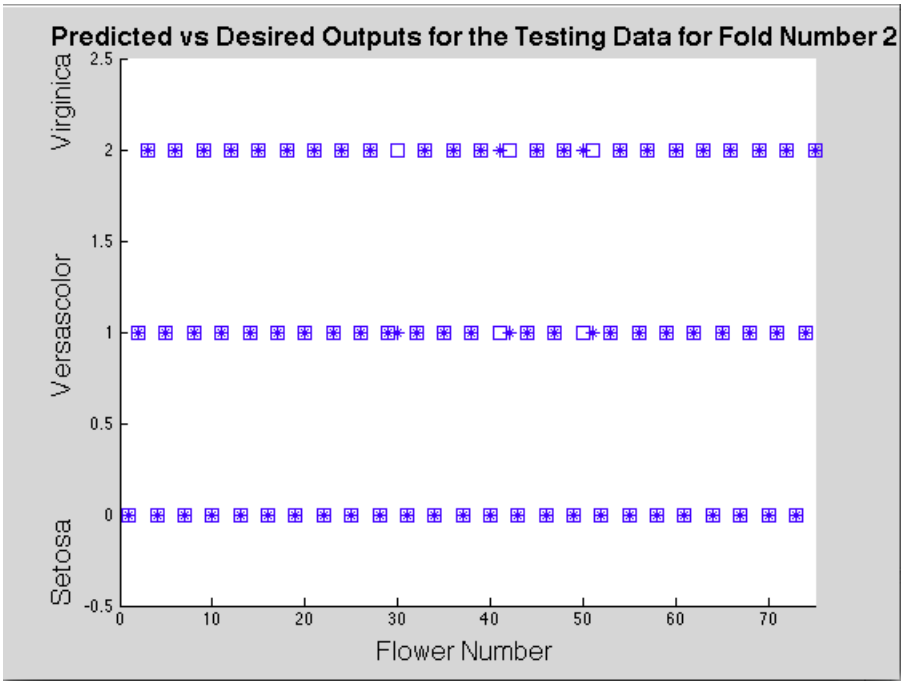
True Class		Predicted Class: Set	Ver	Vir
	Set	25	0	0
	Ver	0	25	0
	Vir	0	0	25

(b) Test data, Fold 1

True Class		Predicted Class: Set	Ver	Vir
	Set	25	0	0
	Ver	0	23	2
	Vir	0	2	23

From this table, we can see that the although the ANN took much longer to converge, (7600 vs 1725), it was still able to make good predictions, an misclassification of 4 percent (4/75). This is a good sign as it makes it likely that the parameters from the previous homework have not overfitted. Moreover, we can see from the tables that the ANN only has trouble distinguishing between Veriginca and Versacolor, perhaps hinting that Setosa is more different from the other two species.

Fold 2



Fold 2 Evaluation	
Final Error of fit for Testing Data	0.0267
Final Error of fit for Training Data	0.0533
# learn steps performed	225000 (threshold for Err_{stop} reached)

Table 4: Confusion matrices for training and test data for for Fold 2

(a) Training data, Fold 2

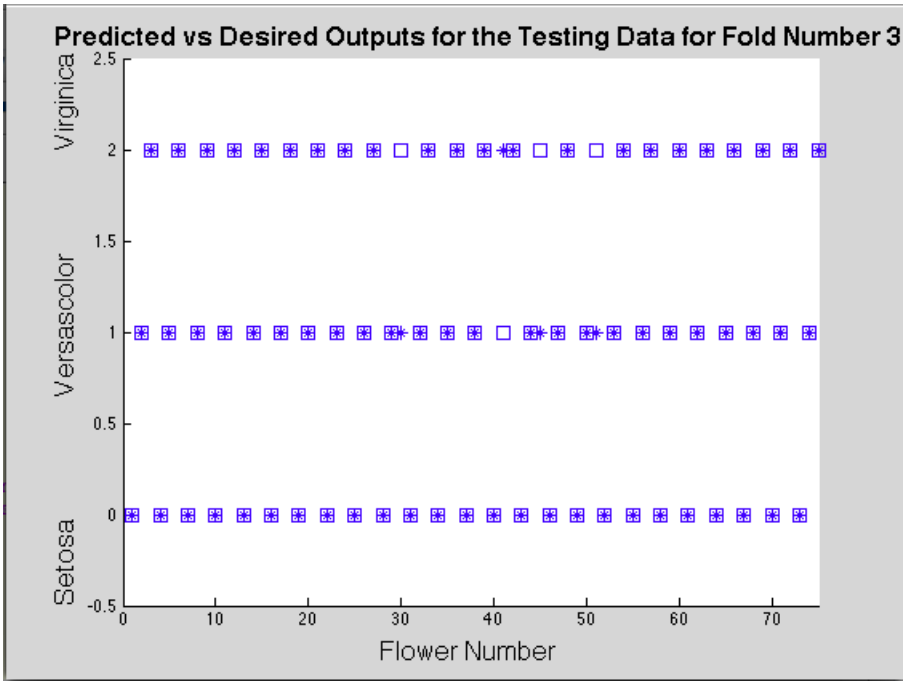
True Class		Predicted Class: Set	Ver	Vir
	Set	25	0	0
	Ver	0	25	0
	Vir	0	0	25

(b) Test data, Fold 2

True Class		Predicted Class: Set	Ver	Vir
	Set	25	0	0
	Ver	0	23	2
	Vir	0	3	22

In terms of accuracy, Fold 2 is a little worse than Fold 1 but not much worse. It misclassifies 5 points insteadof 3 giving it an error of 6.6 percent. In addition the algorithmn never managed to achieve the required error rate so it ran the whoel 225,000 learning steps. In addition, once again, Setosa was not hard to classify, but Versacolor and Virginca gave the algorithmn issues.

Fold 3



Fold 3 Evaluation	
Final Error of fit for Testing Data	0.0267
Final Error of fit for Training Data	0.0267
# learn steps performed	225000 (threshold for Err_{stop} reached)

(a) Training data, Fold 3

		Predicted Class		
		Set	Ver	Vir
True Class	Set	25	0	0
	Ver	0	25	0
	Vir	0	0	25

(b) Test data, Fold 3

		Predicted Class		
		Set	Ver	Vir
True Class	Set	25	0	0
	Ver	0	24	1
	Vir	0	3	22

In Fold 3, things are similar to how they were in Fold 2. The ANN was never able to converge, but it managed to give solid accuracies with 2.6 percent (2/75 misclassifcations) on the training and testing data, and 4/75 or a 5 percent error for the original testing data. Once again, the Versacolor and the Virginca were the ones that the algorithmn had a hard time classifying.

Table

	% Correct Training	% Correct Testing	Average	Standard Deviation
Fold 1	1	0.9333	0.96665	0.047164022
Fold 2	0.9467	0.9733	0.96	0.01880904
Fold 3	0.9733	0.9733	0.9733	0
Average	0.973333333	0.959966667		
Standard Deviation	0.026650016	0.023094011		

From this table, we get another view at the variations among the Folds and in the folds. The first and perhaps most important thing to note is that all of the Averages were above 95 percent meaning that in all cases, this set of parameters is a yeilds good accurate perdictions. However, the standard deviation illustrates some disturbances. The accuracy between the training and the testing samples can be quite large as Fold 1 has a training accuracy of 1 but a testing accuracy of 0.93. On the other hand Fold 3 has both of the accuricies be 0.97. So perhaps some more folds are needed to give a better image of the accuracy of the dataset.

Summary In summary, although the runtime of the algorithmn has increased, and the folds have varying levels of accuracy, the high accuracy of the results makes me inclinded to say that this set of parameters has produced generalizable results.

Question 2.1

2

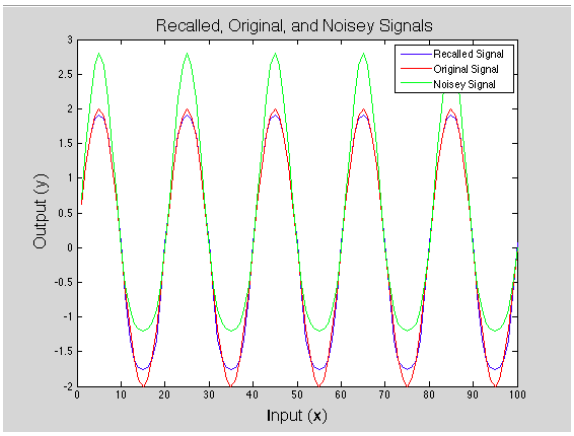
Below are the parameters for the Communication Problem. When multiple parameter values were tried, the optimal is designated with an asterisk. **Stopping Criteria**

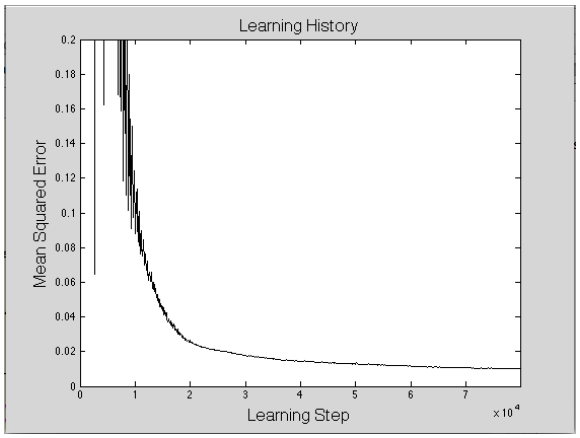
Table 5: Parameters of Training BP Network to Fit Question 2.1

Network parameters	
Topology	$(1 + 1_{Bias})$ — $(3^*/10/20 + 1_{Bias})$ — 1
Transfer function	Hyperbolic Tangent with slope of 1
Learning parameters	
Initial weights	drawn from U[-0.1, 0.1]
Learning rate (α)	0.1/0.01*/0.001/0.0001
Momentum	0/0.8/1
Epoch size ($Epoch$)	100
Stopping criteria	error (Err_{stop}) ≤ 0.01 OR learn count (t) $> 300,000$
Error measure (Err_{stop})	MSE of the unscaled training data see formula (1) below
Input / output data, representation, scaling	
# training samples (N_{tr})	100
Scaling of inputs	Inputs were divided by 2.1 so that they can span from [-0.95, 0.95]
Scaling of outputs	maps raw inputs to [-0.5,0.5]
Parameters and error measures for performance evaluation	
Error of fit (Err_{fit})	Same as Error Measure
Final Error of fit for Training Data	0.0100
# learn steps performed	79,800 (threshold for Err_{stop} reached)

(1) $Err_{stop} = \frac{1}{N_{tr}} \sum_{k=1}^{N_{tr}} (D^k - y^k)^2$, where D^k is the unscaled output for the kth pattern, y^k is the unscaled output of the simulation for the kth pattern, N_{tr} is the number of training samples.

Best Results



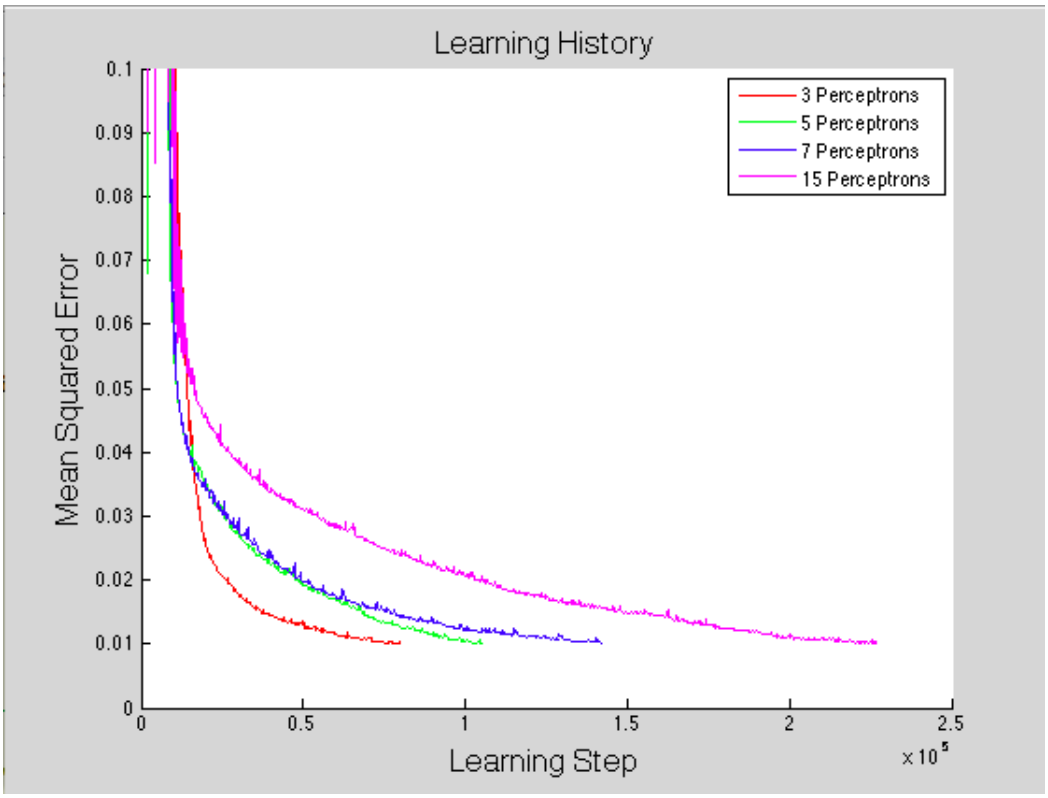


From the plots above, we can see that the ANN has done an fairly well job of learning the pattern. The plot on the left shows that while the ANN was able to reduce a substantial amount of the noise that was added to the system. However some issues remain at the peaks of the sine waves. Moreover, the learning History of the ANN was very smooth and shows that the learning process also had no major difficulties.

Number of Perceptrons

Number of PE s (not including Bias)	3	5	7	15
MSE	0.01	0.01	0.01	0.01
Learning Steps	79800	104900	141900	226900

In each of the following experiments, the number of PE were varied between 3 and 20.

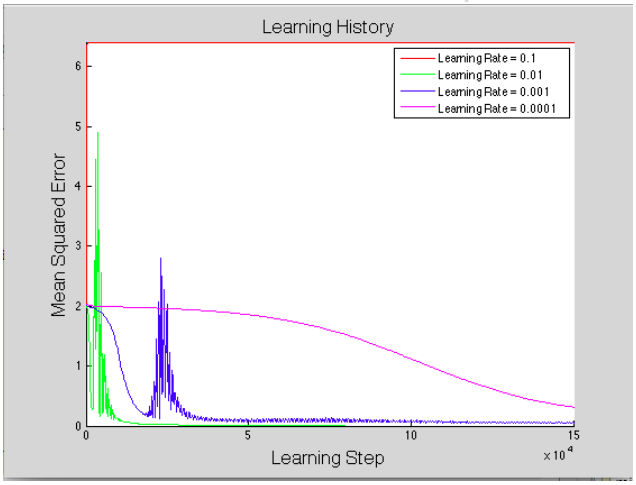


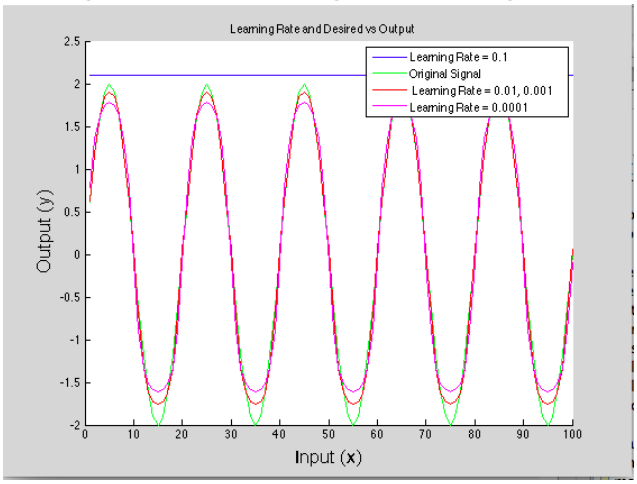
From the learning history, we can deduce a very clear pattern. Varying the number of perceptrons seems to make the ANN take longer to converge. This effect seems to be working in an ordered fashion as well because 5 took longer than 3, 7 took longer than 5 and 15 took longer than 7. This accords with my study of Perceptrons in the previous HW where it seemed that as long as there are enough perceptrons so that the Network can converge (usually more than 1 or 2), then the fewest number of perceptrons is desireable. This is most likely due to the fact that a network with more perceptrons is more complex and would thus require more time for it to learn the same data. So 3 will the be number of perferred perceptrons for this question.

(The plot of Desired vs Output is omitted as all the perceptrons converge to the same MSE so they have the same Output plot as the one seen above).

Learning Rate

Learning Rate	0.1	0.01	0.001	0.0001
MSE	6.41	0.01	0.01	0.0394
Learning Steps	3000000	79800	1714100	3000000





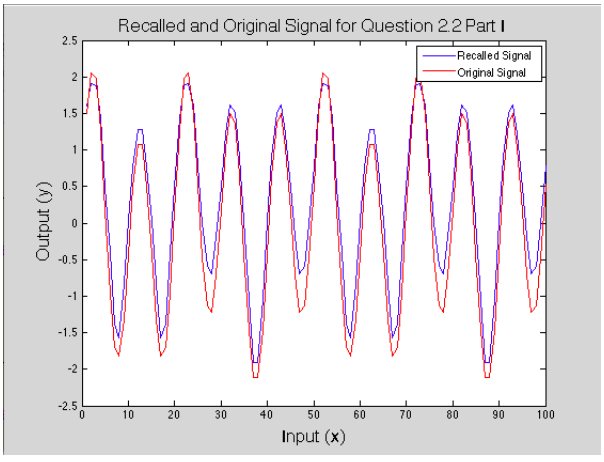
From the plot, we can see that like the number of perceptrons, the learning rate is no good when it is too large (0.1) or when it is too small (0.0001). When the learning rate is too large, as you can see on the Desired vs Output graph, the outputs will start to converge to a completely different value than the desired values. However, if the values are too small, as seen in the learning history plot above, it takes longer and longer for the ANN to converge. So the optimal ANN in this case would be 0.01.

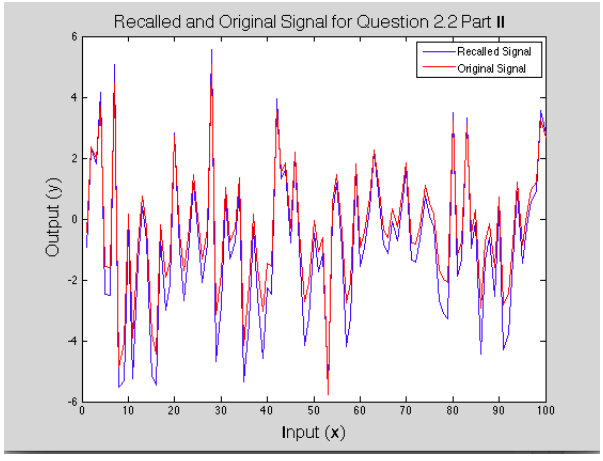
Question 2.2

For the following part of the HW, the following parameters were used:

Table 6: Parameters of Training BP Network to Fit Question 2.1

Network parameters	
Topology	$(1 + 1_{Bias})$ — $(79 + 1_{Bias})$ — 1
Transfer function	Hyperbolic Tangent with slope of 1
Learning parameters	
Initial weights	drawn from U[-0.1, 0.1]
Learning rate (α)	0.01
Momentum	0.8
Epoch size ($Epoch$)	100
Stopping criteria	error (Err_{stop}) ≤ 0.01 OR learn count (t) $> 300,000$
Error measure (Err_{stop})	MSE of the unscaled training data see formula (1) below
Input / output data, representation, scaling	
# training samples (N_{tr})	100
Scaling of inputs	The Inputs of the first signal were divided by 1.9 so that they can span from [-0.95, 0.95] The second set of Inputs were divided by 4.8 for the same reasons
Scaling of outputs	Both of the signals were scaled so that they would maps raw inputs to [-0.5,0.5]





From the plots above, we can see that although the two signals have different ranges, and thus require different factors of scaling, the ANN is doing a good job recovering the original signal. Naturally, it is having difficulties with the more complicated Part II inputs and especially those that have a negative output, but overall, the filtering does give the user a good idea of what the underlying singal looked like.