Neural Networks LAB#3 Report

Submitted By: Ahmed Muhammad Farhan,EMARO

Task 1: Find a prototype pair

Introduction:

By prototype pair we mean a pair of perceptron and adaline which achieves better convergence of the dataset. To guarantee better convergence of the Adaline, we used a decaying learning rate. For a learning rate that starts at iteration 1 with value eta0 and ends at iteration tmax with value etaf<eta0, you can use for instance

eta = (eta0-etaf)*(tmax-t)/(tmax-1)+etaf   (this is a linear decay and works for tmax>1)

The mean of each class was computed and the relating hyperplane was drawn perpendicular to that obtained by perceptron and Adaline. The results obtained are displayed .

Pseudo code:

1. Read given dataset.
2. Plot dataset with corresponding classes(targets).
3. Patterns with classes =-1 as "+".blue color
4. Patterns with classes as = 1 "*".red color
5. Add column of ones to take care about bias.
6. Separate class labels(victor)
7. Initial value of eta0(Learning rate as 0.001)
8. Initial value of etaf(Learning rate as 0.0001)
9. Plot data with corresponding classes for visualization.
10. Call the function my perceptron (dateset+bias,target vactor,eta0.etaf)
11. Plot the separating hyper plane (RED color) with weight victor obtained by the prceptron for visualization purposes.

[Weight vactor (w)]= perceptron (dataset+bias(X), class labels(Y), initial learning rate (eta0), final value of (etaf))

- This function implements the Rosenblatt's perceptron.
- Input = X -> Training Data (with columns as feature vectors)
- Y -> Corresponding Class Label of the training data
- N = Number of training examples
- M = Number of features + 1 (to take care of bias term)
- w = Create a initial weight vector randomly.
- Set a stopping criteria flag = -1; so that Process terminates when flag = 1.
- Niter = 0; initialize No. of iterations to find the weights.
- maxIter = 2000 , Maximum number of iteration before search for linearly separable hyper plane continues.
- Main while loop ,Run until correct weights are found or counter exceeds max iterations.
- numErrors = 0 , keep track of number of errors in each iteration.

- Assign intial value for eta for this iteration , using exponentially decaying eta.
    - eta = (eta0 - etaf)*((maxIter - nIter)/(maxIter-1)) + etaf;

- for i = 1:N ,For all training examples.
- Net input on neuron membrane.
- Compute Output using sign activation function .
- Compute error .
- Count number of errors made by hyper plane (corresponding to current set of weights)
- Update Rule w = w + eta*(y-a)*x
- Compute correction dw.
- Apply update
- End of for loop.
- Increment iterations counter.
- Check stopping criterion
- End of main while loop.
    - Delta (w) or correction
- Finalization
- Print unsuccessful attempt if max iterations reached.
- Print successful attempt if weight vactor is found within the iteration window.

12.Find solution using Adaline

[Weight vactor (w)]= Adaline(dataset+bias(X),class labels(Y), initial learning rate (eta0),final value of (etaf))

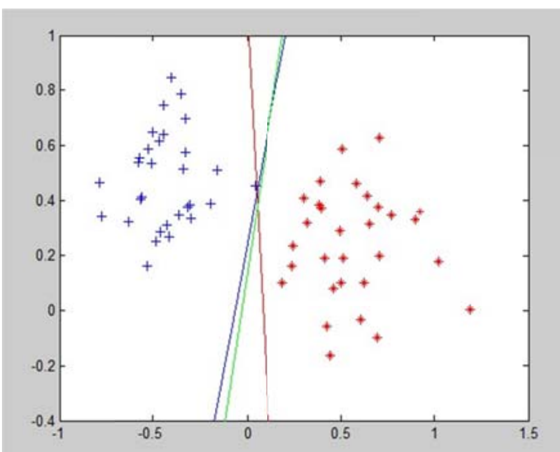
- Implementation of Adaline - Online Algorithm
- function w = Adaline1(X,Y,eta0,etaf)
- N =Number of training examples
- M = Number of feature vectors + 1
- initialize random initial weights
- for while loop termination. Loop terminates when flag = 0.
- maxIter = Emergency/Forced stop.
- maxMSE = Maximum value of mean squared error
- maxMSEPerc =Maximum mean squared error allowed (percentage change in MSE).
- nIter = count number
- mse = 0; Mean squared error
- delta update rule - online
- record previous weight. Used for stopping criteria
- Mean squared error.
- eta = (eta0 - etaf)*((maxIter - nIter)/(maxIter-1)) + etaf;   Update learning rate linear decay
- for all training data
- update step - online
- Calculate mean square error
- Take mean of the total error over all training examples.
- Check performance

- prediction made by current solution
- Number of classification errors
- stop when mse is greater then maxMSE.
- increment iteration counter.
- print some final messages regarding result.
- end.

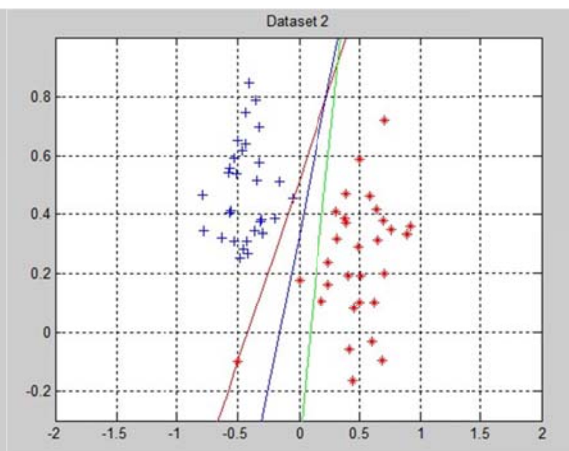
13. BLUE color of Adaline Hyper plane
14. Compute mean of the classes as m1 and m2.
15. Find slope of the line s1
16. Find average of the slope.
17. S2 slope of line perpendicular to S1 = -S1
18. Plot the perpendicular line passing through mid_point of means
19. Y = mx+c
20. Line color == green

RESULTS Task -1

Dateset 3.1



Dataset 3.2




```
***** Perceptron *****

- Sucessful attempt.
 Found weight vector.
 Number of errors = 0
Number of Iterations = 85

***** Adaline *****

 Success.
Process converged found weight vactor.
Iterations = 66
 MSE = 0.13449
```

```
***** Perceptron *****

- Sucessful attempt.
 Found weight vector.
 Number of errors = 0
Number of Iterations = 930

***** Adaline *****

- Success.
Process converged found weight vactor.
Iterations = 288
 MSE = 0.15156
```

TASK -2 **Nearest neighbor classification**

Scheme 1 Sequential Training set

Pseudo code:

1. Load Data in the format specified in readdigits.m file
    - Load Data
    - Feature vectors - pixels
    - Number of training examples
    - Class Labels - (0, 1, 2 ... 9)
    - Putting label for 0 in the last (1,2,3 ... 9 0)
2. rr = no input patterns cc = feature vectors
3. specify the classes from 0 to 9
4. initiate sens best case as zeros(5,1) , for five experiments
5. initiate spec best case as zeros(5,1) , for five experiments
6. initiate sens1 worst case as zeros(5,1) , for five experiments
7. initiate spec1 worst case as zeros(5,1) , for five experiments
8. initialize sequential sample size vactor N.

9. call sequential.m function, for sample sizes of 10,50,100,250,500 which returns sens,spec,sens1,spec1,best digit, worst digit.

[spec, sens,spec1,sens1,i,j] = sequential (N,target_dataset,X)

- training set, n=number of patterns in the training set
- initialize error count =0
- select target for test set
- initialize new targets 0.
- Use nearest neighbor function to find tergets for new training set
- If new targets differ form test set then increment error counter.
- **Compute no of correct = N-error**
- **Compute accuracy and frequency.**
- **Compute confusion matrix.**

		Output	
		0	1
Target	0	NR CORRECT NEG	NR FALSE POS
	1	NR FALSE NEG	NR CORRECT POS

-
- **Print some results on matlab terminal.**
- **Select max and min values form correctly classified targets.**
- **Compute sensitivity for best and worst case.**

$$\text{sensitivity} = \frac{\text{prob. true pos}}{\text{prob. true pos} + \text{prob. false neg}} = \frac{p_{11}}{p_{11} + p_{10}} \approx \frac{c_{11}}{c_{11} + c_{10}}$$

- **Compute specificity for best and worst case.**

$$\text{specificity} = \frac{\text{prob. true neg}}{\text{prob. true neg} + \text{prob. false pos}} = \frac{p_{00}}{p_{00} + p_{01}} \approx \frac{c_{00}}{c_{00} + c_{01}}.$$

-
- **Return these performance indexes along with best and worst number to the parent function (calling function).**

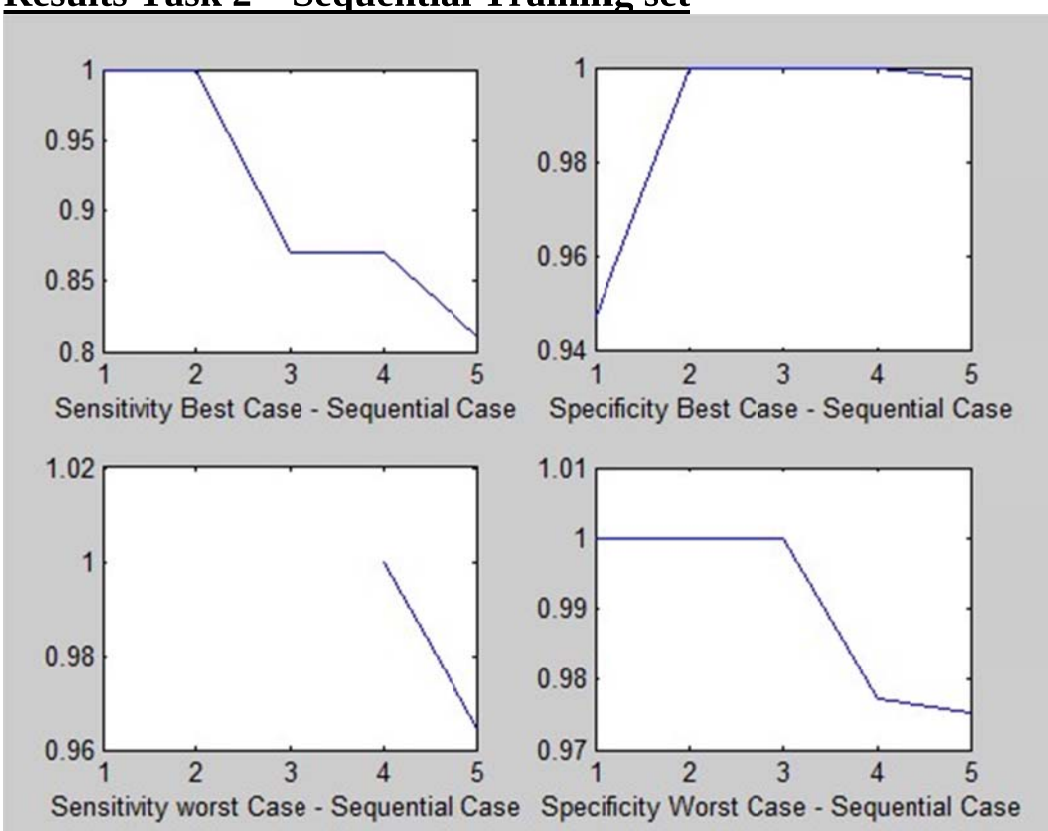
10. Plot the graphs for sensitivity and specificity for best case and worst case.

Scheme 2 Random Training set

1. Repeat the same procedure as indicated above the difference comes from selecting training set and we have to modify or sequential. Function to ramdom.m function as follows
2. [spec,sens,spec1,sens1,i,j] = random(N,target_dataset,X)

- training set, n=number of patterns in the training set
- initialize error count =0
- select target for test set which is basically all the data set because selection is ramdom
- initialize new targets 0.
- Select targets for training set form random indexes.
- Continue with the same procedure as above.

Results Task 2 Sequential Training set



Results Task 2   Random Training set