ECE 457 B: COMPUTATIONAL INTELLIGENCE

ASSIGNMENT #1
(Artificial Neural Networks Part 1 Module)

<u>Due Date: Please upload your solutions in the Learn's Dropbox folder "Assignmnet#1"</u>

<u>on or before 2pm on Friday February 12, 2021</u>

NOTE: To help with the adequate marking of your assignments, please follow these rules carefully:

1. All work should be carried out **<u>individually</u>**
2. The first page of the assignment must have the name and ID of the student
3. The items required must be numbered, labelled, and in numerical order.
4. The solutions should be typewritten. All graphs should be computer generated. You can use existing libraries in Matlab, Python or others (but state the software used and attach a copy of your program code).
5. All pages must be numbered sequentially
6. Show your steps and state any additional assumption you make.
7. Presentation of your results, the organization of your copy, and completion level count for **10% of the total mark.**
8. **The minimum score for the average of all assignments is required to be 50%. Invalid submission or no submission leads automatically to INC in the course.**
9. All copies will be checked by Turnitin similarity software

Problem 1 (25 marks)

For the standard perceptron, the training error due to training pattern (k) is given as the sum squared between the target sample and the output of the combiner (where we include implicitly the bias in the combiner output) as follows

$$E(\mathbf{w}) = (\tfrac{1}{2})\left(t^{(k)} - o^{(k)}\right)^2$$

Where $o^{(k)} = f(\sum_i w_i x_i^{(k)})$ and *f* is the activation function, given here as the step function (note than in the slides we have used the signum function as *f*.) The update rule is given by $\Delta w_i = \eta(t^{(k)} - o^{(k)})x_i^{(k)}$, where η is a positive learning parameter smaller than 1.

In the standard Adaline training we consider the Adaline training error due to training pattern (k) as follows (where we include implicitly the bias in the combiner output):

$$E(\mathbf{w}) = \tfrac{1}{2}(t^{(k)} - (\sum_i w_i x_i^{(k)}))^2$$

We use here notation as in the class notes. In the case, where we consider the error to be between the target sample $t^{(k)}$ and a filtered value of the combiner output going through the sigmoid function *s* , where *s* is given by .

$$s(\text{net}) = \text{sigmoid (net)} = \frac{1}{1+\mathrm{e}^{-(\text{net})}}$$

the new Adaline training error becomes given by:

$$E(\mathbf{w}) = \tfrac{1}{2}(t^{(k)} - s(\sum_i w_i x_i^{(k)}))^2$$

The LMS learning rule (Widrow-Hoff learning) for the Adaline and for one single pattern of training (k), states the weight update as:

$$\Delta \mathbf{w} = -\eta \nabla_{\mathbf{w}} E(\mathbf{w})$$

a) **(5 marks)** Using chain differentiation rule as seen in the lectures and derive the weight update formulae Δw_i for this special Adaline structure and show that it is given by :

$$\Delta w_i = \eta(t^{(k)} - s)\left(s^2.\mathrm{e}^{(-\sum_i w_i x_i^{(k)})}\right)x_i^{(k)}$$

b) **(5 marks)** Write two small programs implementing the weight update rule for the perceptron and the Adaline for an arbitrary number of input (dimension of input **x**) and arbitrary training patterns. Initial values for the weights could be selected randomly in the interval [-1 1].

c) **(10 marks)** We need to classify the following patterns using boundaries obtained from the perceptron and the Adaline. These data points are located in 3D space and the first

component x_0 of each vector is the artificial input associated with w_0, which plays the role of the bias term ($-q$). The remaining entries of the vector are the actual location of the data point in 3D space.
Class "C1=0" (x^1=[-1, 0.8, 0.7, 1.2] , x^2=[-1, -0.8,- 0.7, 0.2], x^3=[-1, -0.5,0.3,- 0.2], x^4=[-1, -2.8, -0.1, -2])
Class "C2=1" (y^1=[-1, 1.2,- 1.7, 2.2] , y^2=[-1, -0.8,-2, 0.5], y^3=[-1, -0.5,-2.7,- 1.2], y^4=[-1, 2.8, -1.4, 2.1])

If you are able to separate these classes, provide the equation of the boundaries in the case of the perceptron and the case of the Adaline. Draw, the data points of the two classes, and the corresponding boundaries in 3D Cartesian space (remember the first entry of each vector, is not part of the coordinate of the data point in 3D). Use the value of h as 0.6

d) **(2 marks)** We need to place a new data point belonging to "C1" in the location x^5=[-1.4,- 1.5, 2]. Is the classifier boundary still valid (for both perceptron and Adaline)

e) **(3 marks)** State, why an Adaline structure with LMS learning algorithm has better capabilities than the perceptron Hebbian learning rule.

Problem 2 (15 marks)
Build a Madaline structure that is able to provide a compounded boundary (composed of two elementary boundaries) for the Exclusive Nor logic (XNOR) gate with two inputs. Draw the boundary in a 2D cartesian plot and show that the obtained compounded boundary (composed of two boundaries) is able to separate the two output classes (1 and -1).

Problem 3 (30 marks)
Using a feedforward back-propagation neural network that contains a single hidden layer (with a variable number of hidden nodes each having an activation function of the logistic form), investigate the outcome of the neural network for the following mappings:

- $f_1(x) = x*\sin(6\pi x)*\exp\left(-x^2\right)$ where x Î $\left[-1,\ 1\right]$
- $f_2(x) = \exp\left(-x^2\right)*\arctan(x)*\sin 4\pi x$ where x Î $\left[-2,\ 2\right]$

For each function, create three sets of *input/output* data, one for training and validation and one for testing. These will be random values within the interval of the variable x). You can choose the ratio as 80% of the data for training (including validation) and 20% of the data for testing. We will use 10-fold cross validation approach, which means that the training set will be divided into 90% training and 10% for validation.

a) **(15 marks)** We need to create a 4 by 4 grid of 16 models assessments for each function. Each model $M_{i,j}$

(*i, j* th entry of the grid) has *i* as number of data set (training and testing) and *j* as the number of hidden nodes in the neural network model. *i* takes the following values: $i = \{10, 40, 80, 200\}$ data points, while *j* takes the following values: $j = \{2, 10, 40, 100\}$ hidden nodes. For each $M_{i,j}$, apply 10-fold cross validation (and repeat the process 5 times by shuffling the data generated randomly) and get error average for training and validation. For each model and each 10-fold cross validation, use early stopping criteria to make sure the validation error and the training error are both small and to avoid overfitting. Please see lecture slides on model selection and K-fold cross validation. Generate a grid (table) with entries corresponding to each model $M_{i,j}$, where the entries have the best training and validation errors obtained for that model (using early stopping criteria).

b) **(10 marks)** Make qualitative and quantitative deductions in light of these simulations and find the best model that has acceptable bias (complex enough model) and an acceptable variance (no overfitting). Apply the full training data set to this model, and then apply the testing data (which the system never saw and draw the original functions and the best model obtained through neural network prediction.

c) **(5 marks)** Explain your deductions in light of the material taught in class and according to Appendix A.

Problem 4 (20 marks)

We need to develop a neural network based classifier for three various but related products. The collected data are the results of a chemical analysis of liquid products grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of products. The data file provided is in text format and has fourteen dimensions, the first of which determines the class of products (product '1', product '2', product '3'), which should serve as the output of the neural network classifier. The remaining ones determine the input of the classifier and has 13 constituents as:

1. Ethanol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted liquid
13. Proline

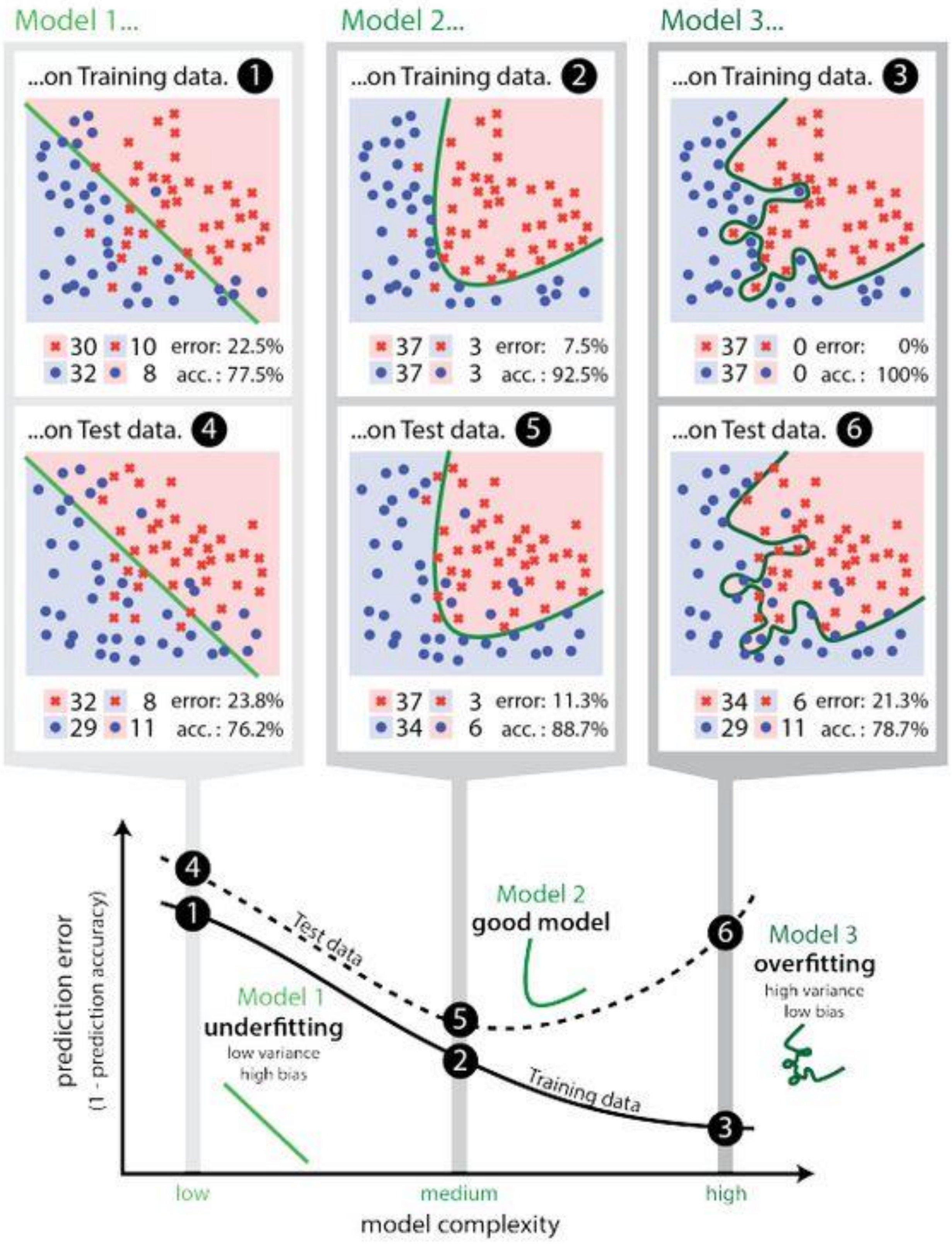
4.1 **(15 marks)** Build a classifier (multilayer neural network), vary its parameters (number of hidden layers and number of nodes in each layer) and try to find the best possible classification performance (a table illustrating various results as parameters are varied would be preferred). Please discuss.

4.2 **(5 marks)** Once this is done, classify (determine to which product they belong) the following entries each of which has 13 attributes:

a) 13.72; 1.43; 2.5; 16.7; 108; 3.4; 3.67; 0.19; 2.04; 6.8; 0.89; 2.87; 1285
b) 12.04; 4.3; 2.38; 22; 80; 2.1; 1.75; 0.42; 1.35; 2.6; 0.79; 2.57; 580
c) 14.13; 4.1; 2.74; 24.5; 96; 2.05; 0.76; 0.56; 1.35; 9.2; 0.61; 1.6; 560

Hint for implementation: You may wish to calibrate all input data to be all between 0 and one. Also from the set of data choose 75% of the data from product 1, product 2 and product 3 as training data and remaining 25% remaining for testing.

Apeendix A



Model 1...
...on Training data. 1
30 10 error: 22.5%
32 8 acc.: 77.5%
...on Test data. 4
32 8 error: 23.8%
29 11 acc.: 76.2%
Model 2...
...on Training data. 2
37 3 error: 7.5%
37 3 acc.: 92.5%
...on Test data. 5
37 3 error: 11.3%
34 6 acc.: 88.7%
Model 3...
...on Training data. 3
37 0 error: 0%
37 0 acc.: 100%
...on Test data. 6
34 6 error: 21.3%
29 11 acc.: 78.7%
prediction error
(1 - prediction accuracy)
Test data
Training data
Model 1
underfitting
low variance
high bias
Model 2
good model
Model 3
overfitting
high variance
low bias
low
medium
high
model complexity