2018 Scheme

1	N	T							

NITTE EDUCATION TRUST

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution | Accredited with 'A+' Grade by NAAC | NBA | Approved By UGC | AICTE | Govt. of Karnataka | Affiliated to VTU, Belagavi)
Yelahanka, Bangalore - 560 064

KNOWLEDGE • CHARACTER • UNITY

Department of Computer Science and Engineering

Course Title with code	Introduction to Machine Learning 18CSE751	**Maximum Marks**	**30 Marks**
Date and Time	20-Nov-2021 , 2:30 – 3:30 pm	**No. of Hours**	1.0
Course Instructor(s)	Dr.Vani V		

Instructions to Students

1. Answer any **two full questions.**
2. Any missing data may assume suitably.

Q. No	Question	MAX MARKS	CO	BL	PO and PSO
1	Consider the 4 x 4 Confusion Matrix given in **Table 1** and answer the questions a through f.	15	1	3	1,2/1,2

		Computed Decision			
		Class 1	Class 2	Class 3	Class 4
Actual	Class 1	10	3	3	2
	Class 2	3	20	0	1
	Class 3	2	2	15	2
	Class 4	3	3	1	20

Table 1. 4x 4 Confusion Matrix

Use the above four-class confusion matrix below to answer the questions a through f.

Solution:

a. What percent of the instances were correctly classified? **(1 Mark)**

65/90% = 72.22%

b. According to Confusion matrix, how many *Class 1/Class 2/Class 3 /Class 4* instances are there in the dataset? **(1 Mark)**

Class 1 : 18

Class 2: 24

Class 3: 21

Class 4: 27

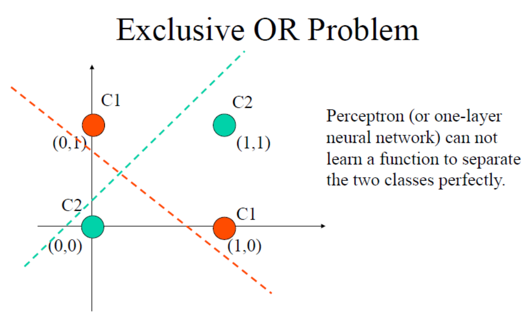
c. How many instances were incorrectly classified with *Class 1/Class2/Class 3/Class 4*? **(1 Mark)**

Class 1: 8

Class 2: 8

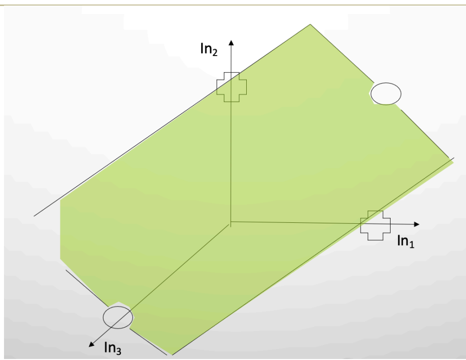
<table>
<tr>
<td></td>
<td>Class 3: 4
Class 4: 5
d. Calculate Sensitivity, Specificity of Class 1/Class2/Class 3/Class 4.
Sensitivity (Recall): TP/(TP+FN)
Class 1: 10/18=0.56
Class 2: 20/24 = 0.83
Class 3: 15/21 = 0.71
Class 4: 20/27 = 0.74
Specificity: TN/(TN+FP)
Class 1: 55/63=0.87
Class 2: 45/53 = 0.85
Class 3: 50/54=0.93
Class 4: 45/50 = 0.9
e. Calculate FPR and FNR of Class 1/Class2/Class 3/Class 4.
FPR = 1-Specificity
Class 1: 0.13
Class 2: 0.15
Class 3: 0.07
Class 4: 0.1
FNR = 1-Sensitivity
Class 1: 0.44
Class 2: 0.17
Class 3: 0.29
Class 4:0.26
f. Calculate F1 score Class 1/Class2/Class 3/Class 4.
F1 Score = 2*(P*R)/(P+Recall) (P)recision = TP/(TP+FP)
Class 1: 0.56 P = 10/18 = 0.56 ; 2*(0.56* 0.56)/(0.56+0.56)
Class 2: 0.77 P =20/28 =0.71 ; 2*(0.71*0.83)/(0.71+0.83)
Class 3: 0.75 P=15/19 = 0.79; 2*(0.79*0.71)/(0.79+0.71)
Class 4: 0.77 P=20/25 =0.8 ; 2*(0.8*0.74)/(0.8+0.74)</td>
<td>(4 Marks)

(4 Marks)

(4 Marks)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. a</td>
<td>“An additional dimension would resolve the linear separability problem.” Justify / contradict the statement with necessary illustrations
Solution:

A decision boundary solving the XOR problem in 3D with the crosses below the surface and the circles above it.</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>1,2/1,2</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

A	B	C	Out
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	0



If we rewrite the problem in three dimensions instead of two, then it is perfectly possible to find a plane (the 2D analogue of a straight line) that can separate the two classes. Writing the problem in 3D means including a third input dimension that does not change the data when it is looked at in the (x, y) plane, but moves the point at (0, 0) along a third dimension. So, the truth table for the function is the one shown on the left side
(where ‘C’ has been added, and only affects the point at (0, 0)).

Scheme: Linear Separability problem with perceptron : 2 marks
Addition of a dimension with justification & illustration: 5 marks
</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. b</td>
<td>

Discuss any 3 variants of gradient descent algorithm. Also, suggest a solution to avoid local minima

Solution:

Batch algorithm.

- All the training examples are presented to the neural network, **the average sum-of-squares error** is then computed, and this is used to update the weights. Thus, there is **only one set of weight updates for each epoch** (pass through all the training examples).
- Only update the weights once for each iteration of the algorithm
- The weights are moved in the direction that most of the inputs want them to move, rather than being pulled around by each input
- The batch method performs a more accurate estimate of the error gradient and will thus converge to the local minimum more quickly.
- **Batch algorithm** makes a better estimate of the steepest descent direction, so that the direction it chooses to go is a good one, but this just **leads to a local minimum.**
-

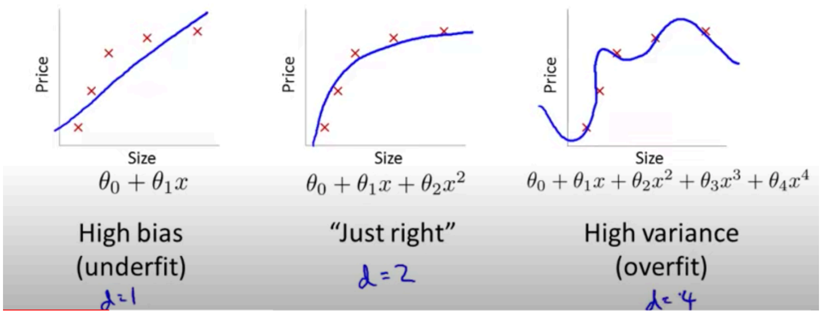
Sequential algorithm

- where the errors are computed, and the weights updated after each input.
- This is not guaranteed to be as efficient in learning, but it is simpler to program when using loops,
- Therefore, **much more common**. Since it does not converge as well, it can also sometimes avoid local minima, thus potentially reaching better solutions.
- A **minibatch method** is to find some happy middle ground between the two.
- Split the training set into random batches
  - Estimating the gradient based on one of the subsets of the training set
  - Performing a weight update,
  - And then using the next subset to estimate a new gradient and using that for the weight update,
- Until all the training set have been used.
- The training set are then randomly shuffled into new batches and the next iteration takes place.
</td>
<td>8

3 variants x 2 = 6 marks

Adding Momentum – 2 marks</td>
<td>2</td>
<td>2</td>
<td>1,2/1</td>
</tr>
</table>

	• **If the batches are small, then there is often a reasonable degree of error in the gradient estimate, and so the optimisation has the chance to escape from local minima, though at the cost of heading in the wrong direction.** **Adding momentum** • can help to avoid local minima, • makes the dynamics of the optimisation more stable, improving convergence $w^t_{\zeta\kappa} \leftarrow w^{t-1}_{\zeta\kappa} + \eta\delta_o(\kappa)a^{\text{hidden}}_{\zeta} + \alpha\Delta w^{t-1}_{\zeta\kappa},$ • Where t is used to indicate the current update and t − 1 is the previous one. • $\Delta w^{t-1}_{\zeta\kappa}$ is the previous update that we made to the weights • So , $\Delta w^t_{\zeta\kappa} = \eta\delta_o(\kappa)a^{\text{hidden}}_{\zeta} + \alpha\Delta w^{t-1}_{\zeta\kappa}$ • $0 < \alpha < 1$ is the momentum constant. Typically, a value of $\alpha = 0.9$ is used. This is a very easy addition to the code and can **improve the speed of learning a lot.**				
3. a	Consider the following scenarios and determine the type of machine learning approach that is best suited with necessary justification. **Solution:** i. Determine the characteristics of a successful medical sales representative. **Supervised learning .** Because it mentions about **successful /unsuccessful** medical sales representative . ii. Do meaningful attribute relationships exist in a dataset containing information about mentally-ill employees? **Unsupervised learning.** This is because we need to group(cluster) similar attribute values together inorder to discover pattern about mentally-ill employees iii. Do the tourism industries sustain after this Covid19 pandamic? **Unsupervised learning.**Here inorder to determine the sustainability of the tourism industry, we need to look at the torurism industries dataset. There are different attributes and its relationship will help in determining the sustainability . Once we could cluster the data then, we can determine whether the industries can sustain or not (supervised) iv. Determine the next key press of a gamer immersed in a 3D game. **Supervised learning.** In any gaming,there will be fixed set of keys (categories). Based on the probability of each key press and its conditional probabilities we can do the classification to determine the next key press Y if the current key pressed is X.	6 4 x 1.5 = 6 marks	1	2	1,2/1
3. b	Assume an unbalanced data set D, with N tuples and 4 classes with the distribution of 10% class 1, 30% class 2, 20% class 3	3	1	2	1,2/1

	and remaining class 3 samples. In this scenario, how will you calculate balanced accuracy of the machine learning model? **Solution:** We can compute the balanced accuracy as the sum of sensitivity and specificity divided by 2. However, a more correct measure is Matthew's Correlation Coefficient (MCC) , which is computed as: $MCC = \frac{\#TP \times \#TN - \#FP \times \#FN}{\sqrt{(\#TP + \#FP)(\#TP + \#FN)(\#TN + \#FP)(\#TN + \#FN)}}$				
3. c	Discuss with necessary illustrations to justify the statement "more complex models do not necessarily result in better results" **Solution:** It is called **bias-variance trade-off.** **A model can be bad for two different reasons**. Either it is **not accurate** and doesn't match the data well, or it is **not very precise** and there is a lot of variation in the results. The first of these is known as the bias, while the second is the statistical variance. More complex classifiers will tend to improve the bias, but the cost of this is higher variance, while making the model more specific by reducing the variance will increase the bias. **Bias/variance** 	6 3 + 3	2	2	1,2/1

Faculty Signature	Course Co-ordinator/Mentor Signature	HoD Signature
		Dr. Thippeswamy M N