Sudha Anusha Sagi

1

1) SHORT ANSWERS

a) "All of the following answers are my own work"

S. Anusha

b) Regularization finds a model where it does not fit the training model so well so as to reduce overfitting. i.e., it makes the model less complex so as to decrease the variance in the unseen data

L1 Regularization → Lasso → ~~~~ $+ \lambda \sum_i |W_i|$

L2 Regularization → Ridge → ~~~~ $+ \lambda \sum_i (W_i)^2$

Both the models give some penalty to the model. In L1, it adds absolute value to the coefficient as penalty & in L2, it adds squared value to the coefficient as penalty.

To be more precise, lasso can only either totally remove or add feature coefficients depending on their correlation with the model. But ridge can have weights (high or low) for each of the feature's coefficients.

c) Decision trees are quite unstable & each model can give different results for the same dataset. i.e, they have high independence b/w the models & their errors.

The ensemble work well with unstable models as they combine each weak learners together & predict based on voting or giving weights to each of the model based on the accuracy of each model.

∵ Decision trees are unstable & can give different results everytime (i.e., this can lead to overfitting), ensemble methods like bagging & boosting can be used to decrease overfitting thereby decreasing variance & boosting can reduce bias without specifally increasing variance.

d) Usually the activation fn is added to the Neural Network to achieve some kind of non-linearity in the function.

In NN, during feedforward, we multiply (dot product) input vector with random weights vector. & so on, till the output node. Generally, this dot products is linear operation & gives similar o/ps. If we don't provide activation fns like RELU & sigmoid (which are basically based on confidence values) functions, our NN would just give some linear output & will not able to label or categoriz the outputs.
So, to have some non-linearity in the outputs, we use activation fns so that we can get desired output labels.

e) Yes, we can apply kernel transformation to NN. It helps in creating much more complex models that a required. For example, in image classification based on pixel values, having very high degree for kernel can make the computa-tion more complex & requires high computation time. It depends on the CPU availability & depends on the prob-lem that we are trying to solve.
In simple digit classification, with less no.of colors & their intensities, we need not require kernel transformation because from the rough distingutation b/w pixels, we will be able to predict the label. But for some complex images that are hard to read & identify, kernel transf, may be able to distinguish pixel by pixel (basically in the low confidence regions).

f) SVM would take longest to retrain. ②

① In Nearest Neighbours, we don't actually have any training. So, retraining would not be needed or happen.

② In Neural N/ws, if we encounter a new point, we can just train that single point by passing it to the neural network that we've already built & the previous points that we've trained can still be retained. So, the retraining is not all complex or time consuming here.

③ In SVM, if we encounter a new point, we need to change our seperator & margins depending on where the point lies. Suppose we have hard margin SVM, & the new point falls within the margin, we need to shift our boundary line & recalculate all the values like margin, support vectors etc. So, this is the longest to retrain.

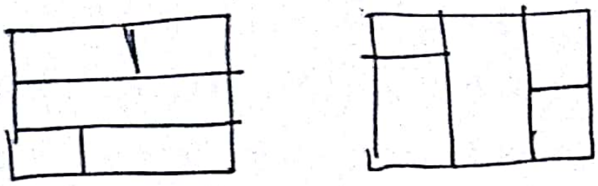
g) Bagging works on the concept of creating equal sized bags by drawing samples from the training data & creating different learners for each bootstrap that is created.

For eg, if we just have a single model on the whole training data, our model might overfit the data & sometimes we might even fit our model for the outliers that are present which will obviously lead to overfitting & high variance.

In bagging, all bags don't contain all sample points i.e., the model doesn't memorize all training points which makes each model have some amount of bias to the points that are not included. Also, outliers can easily be identified and the results can exclude them.

h) In decision trees, the boundaries are continuous & is rectangular form. & then each block can further be subdivided using continuous boundary.

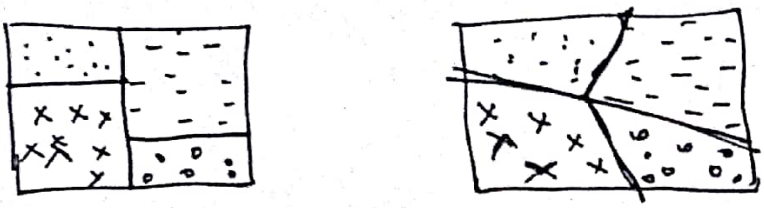
For eg:- boundaries created by decision trees are. i.e,



} rectilinear decision boundaries.

So, I would look how the decision boundaries are.

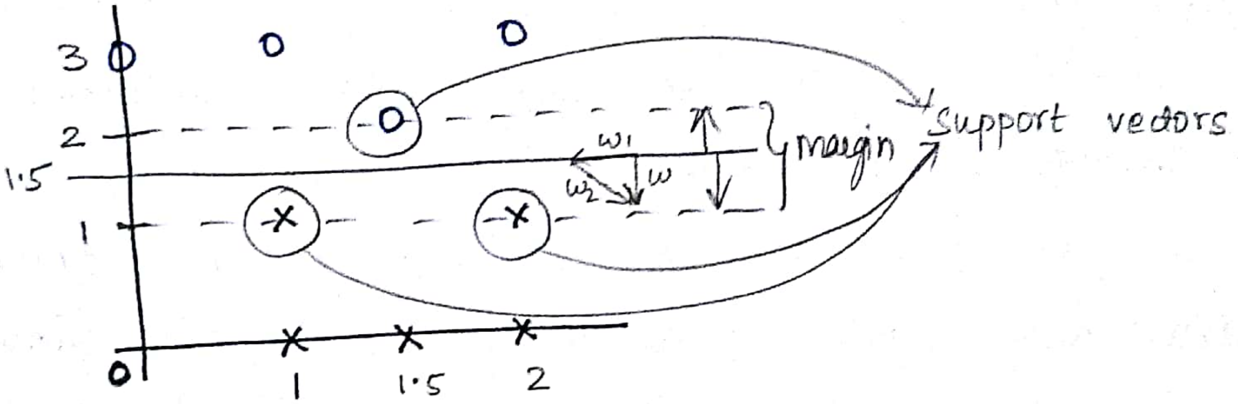
Eg:-



Decision Trees   Neural Networks.

2) SUPPORT VECTOR MACHINES.

a)



b) $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = y$.

$\boxed{w_1 = 0}$

$\Rightarrow$ when $x_2 = 1.5 \Rightarrow w_0 + w_2(1.5) = 0 \Rightarrow \boxed{w_0 = -1.5 w_2}$

$\Rightarrow$ when $x_2 = 1 \Rightarrow w_0 + w_2(1) = 1 \Rightarrow -1.5 w_2 + w_2 = 1$

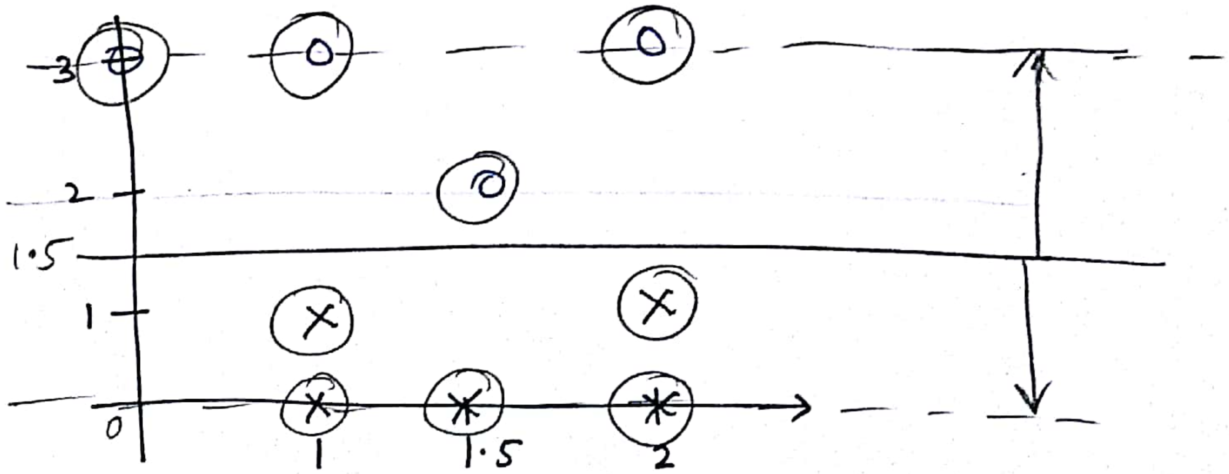
$\Rightarrow -0.5 w_2 = 1 \Rightarrow \boxed{w_2 = -2}$

Now $w_0 = -1.5(w_2) \Rightarrow w_0 = -1.5(-2) \Rightarrow w_0 = -\frac{3}{2} \times -2$

$\Rightarrow \boxed{w_0 = +3}$

$\therefore \beta_0 = +3, \ \beta_1 = 0, \ \beta_2 = -2$

c)



Scanned by CamScanner

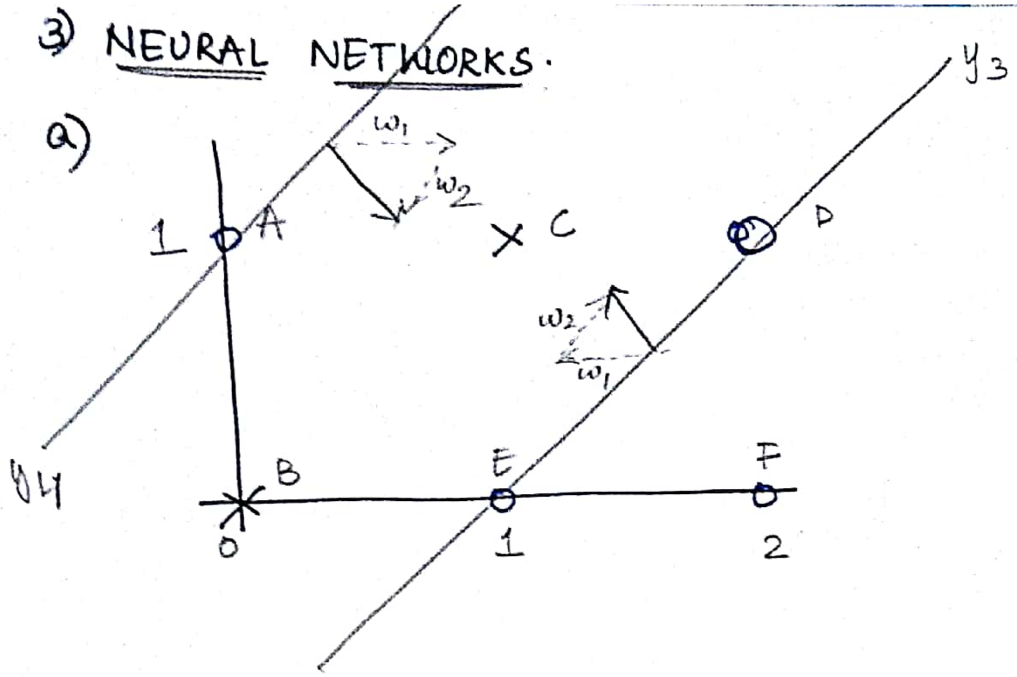
d) The model is almost same where the sepelator is still at 1·5

But the soft margin increases, which includes high no·of misclassifications.

Because for high values of c, we generally tend towards hard margin & as 'c' value becomes lower, the margin increases allowing some misclassifications & as 'c' keeps decreasing the no·of misclassifications increase.

3) NEURAL NETWORKS.

a)



b) $y_3 = x_1 + x_2 + \text{bias}$

$$y_3 = \underset{w_1}{\underline{-1}} + \underset{w_2}{\underline{(+1)}} + (+1)$$

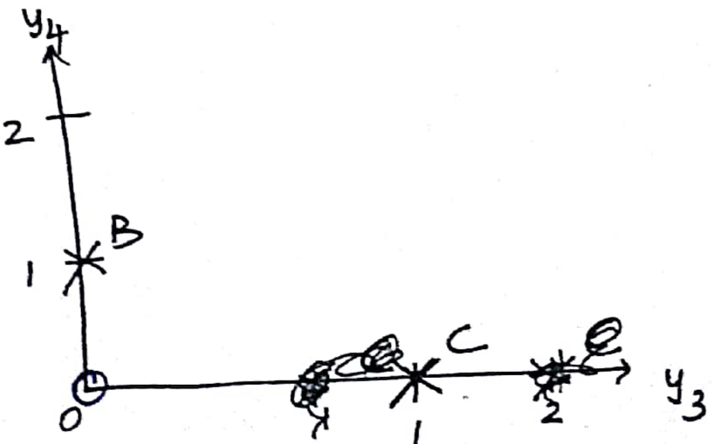
$\therefore$ Weights are $-1, +1, +1$

$y_4 = x_1 + x_2 + \text{bias}$

$$y_4 = \underset{w_1}{\underline{+1}} + \underset{w_2}{\underline{(-1)}} + (+1)$$

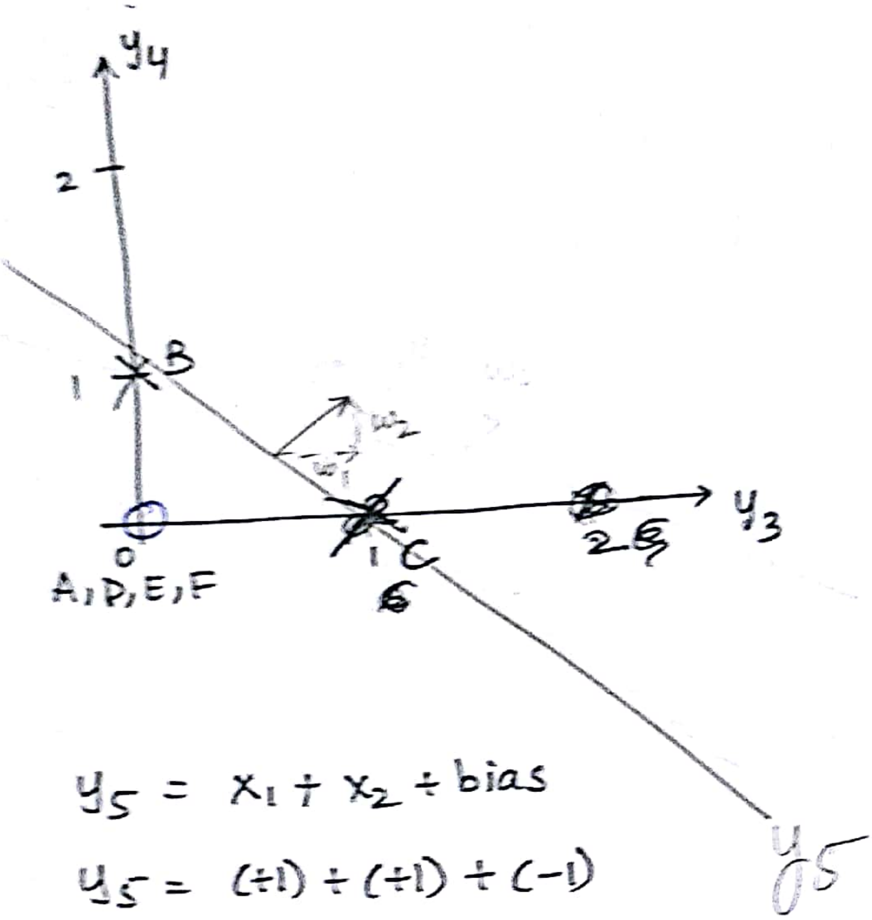
$\therefore$ Weights are $-1, +1, +1$

c)



Transformed y_3 & y_4 space.

d)



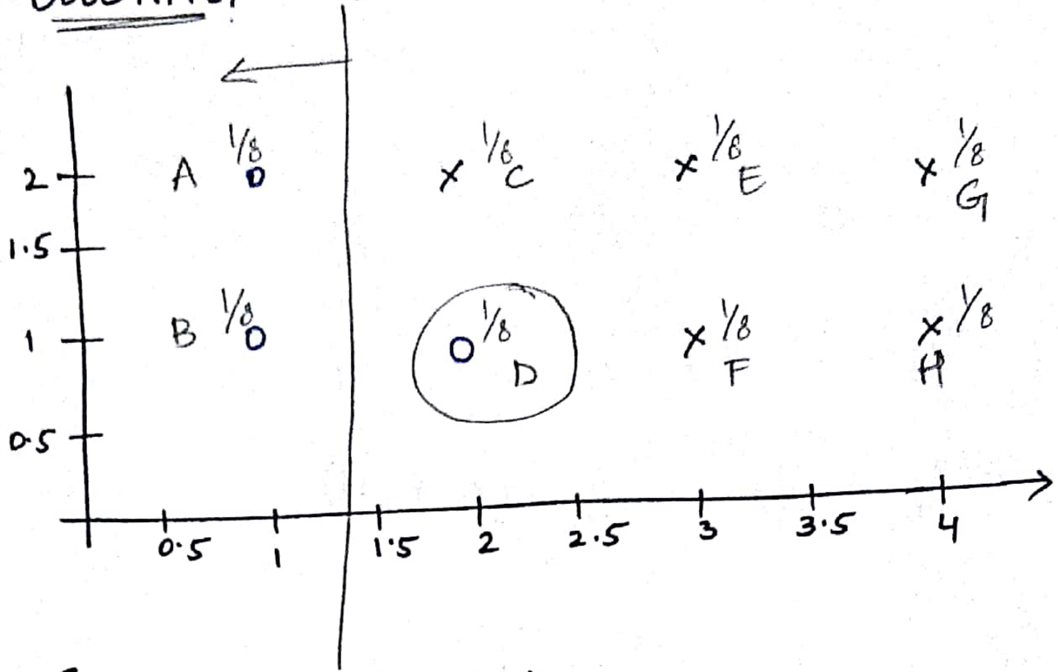
e) $y_5 = x_1 + x_2 + \text{bias}$

$y_5 = (+1) + (+1) + (-1)$

$\therefore$ Weights are +1, +1, -1

4) **BOOSTING.**

a)

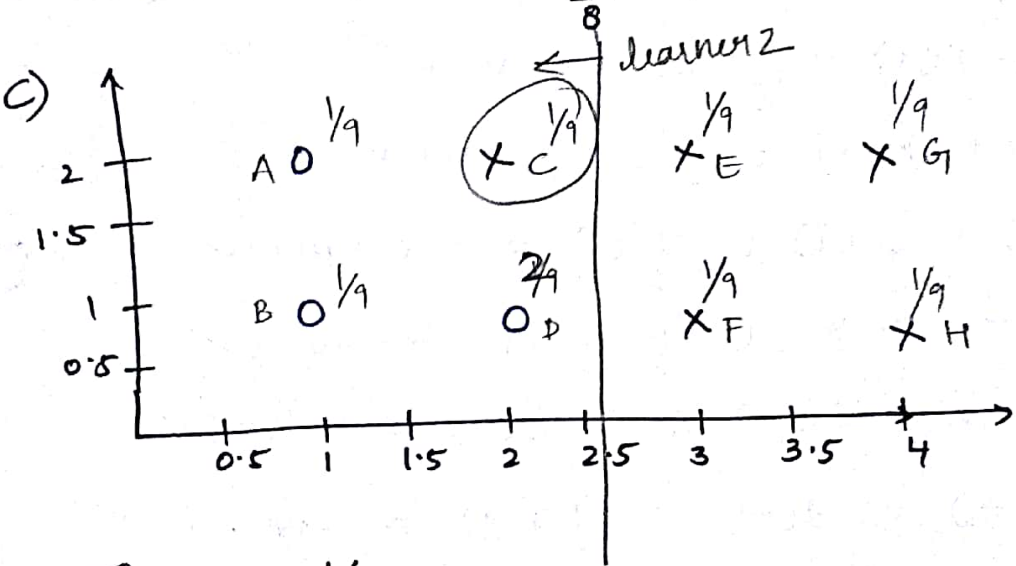


Error = $\frac{1}{8}$ (point D)

b) $\alpha = \frac{1-\epsilon}{\epsilon} = \frac{1-\frac{1}{8}}{\frac{1}{8}} = \frac{7}{1} = 7.$ $\boxed{\alpha = 7}$

(Double weight of D)

c)

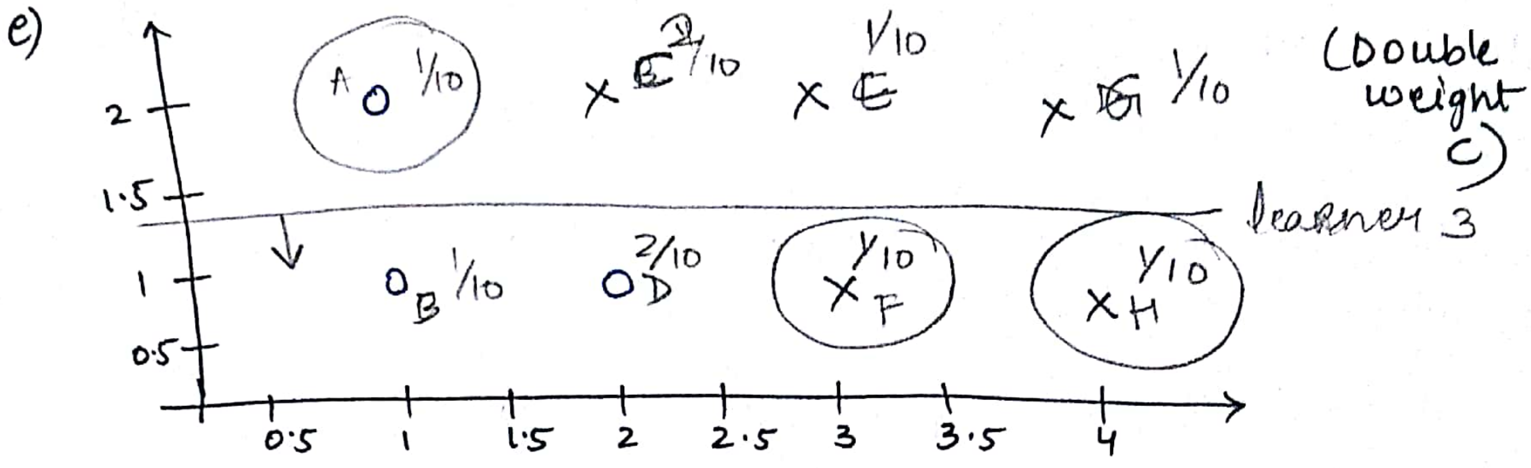


Error = 1/9

d) $\alpha = \frac{1-\frac{1}{9}}{\frac{1}{9}} = \frac{8}{1} = 8.$

$\boxed{\alpha = 8}$

e)



$$\text{Error} = \frac{1}{10} + \frac{1}{10} + \frac{1}{10} = \frac{3}{10} \quad (A, F, H)$$

f) $$\alpha = \frac{1 - \frac{3}{10}}{\frac{3}{10}} = \frac{7}{3} \Rightarrow \boxed{\alpha = \frac{7}{3}}$$

g) Classification for $\overset{(+ve)}{A} \Rightarrow 7(1) + 8(1) + \frac{7}{3}(-1) = +ve$ ✓

(+ve) B $\Rightarrow 7(1) + 8(1) + \frac{7}{3}(1) = +ve$ ✓

(−ve) C $\Rightarrow 7(-1) + 8(1) + \frac{7}{3}(-1) = -ve$ ✓

(+ve) D $\Rightarrow 7(-1) + 8(1) + \frac{7}{3}(1) = +ve$ value ✓

(−ve) E $\Rightarrow 7(-1) + 8(1) + \frac{7}{3}(1) = -ve$ ✓

(−ve) F $\Rightarrow 7(-1) + 8(-1) + \frac{7}{3}(+1) = -ve$ ✓

(−ve) G $\Rightarrow 7(-1) + 8(-1) + \frac{7}{3}(-1) = -ve.$ ✓

(−ve) H $\Rightarrow 7(-1) + 8(-1) + \frac{7}{3}(+1) = -ve$ ✓

∴ All the points are classified correctly after 3 iterations.

∴ Training Error = 0 //.