CS-534 Machine Learning

Homework-3

Akash Agarwal
OSU ID: 933471097

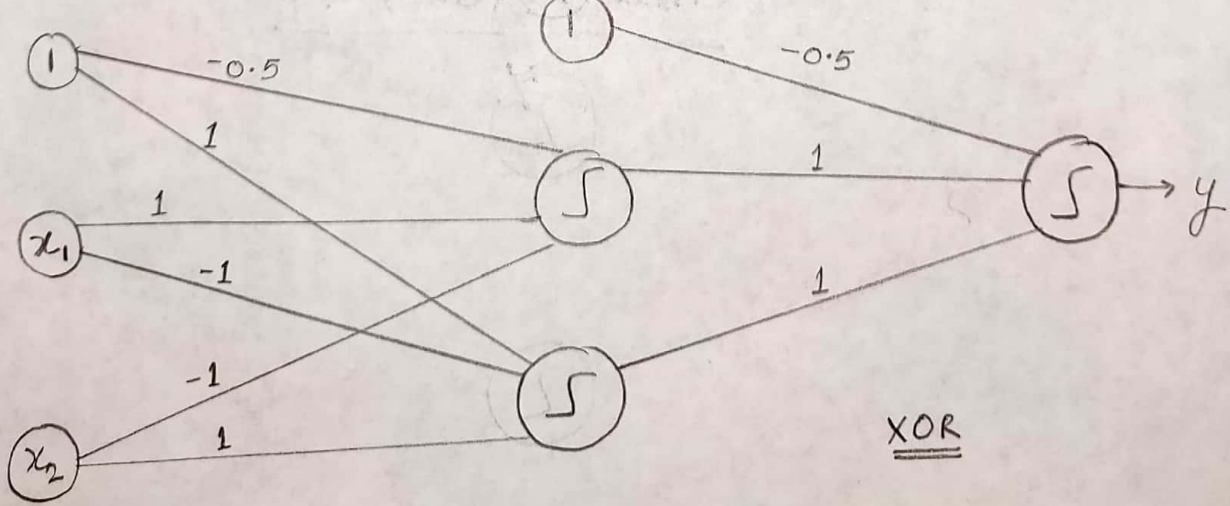
Ques 1. (A) It is impossible to implement XOR function $y = x_1 \oplus x_2$ using a single unit (neuron). However, you can do it with a neural net. Use the smallest network you can. Draw your network & show all the weights.

(B) Explain how can we construct a neural network to implement a Naive Bayes Classifier with Boolean features.

(C) Explain how can we construct a neural network to implement a decision tree classifier with boolean features.

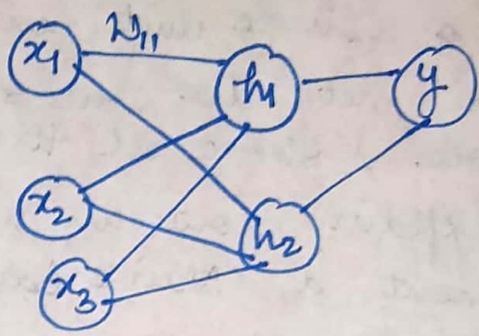
Ans 1. (A) $XOR = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$

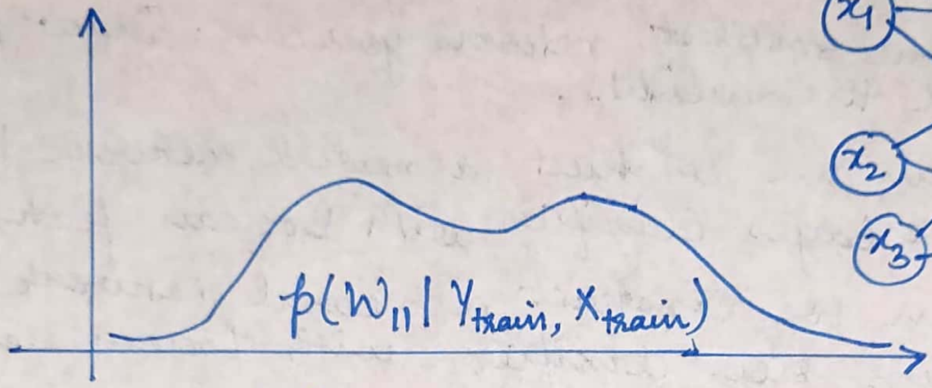
x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$	$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



XOR

Ans 1-(B): To construct a neural network that can implement Naive Bayes Classifier with boolean features, the weights of the neural network will contain the value of boolean distribution in the Naive Bayes classifier.





We have to find :-

$$p(y \mid x ; Y_{train}, X_{train})$$

$$= \sum \underbrace{p(y \mid x ; w)}_{\text{This is the output of Neural Net}} \underbrace{p(w \mid Y_{train}, X_{train})}_{\text{Weight.}}$$

here, $p(w \mid Y_{train}, X_{train})$ can represent any Boolean Distributio

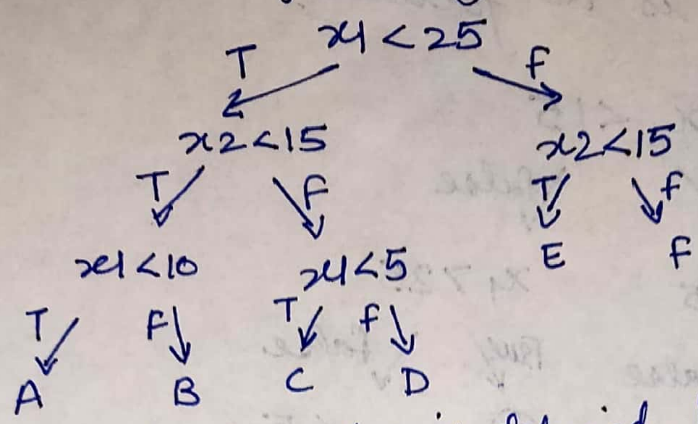
$$= \frac{p(Y_{train} \mid X_{train}, w)\, p(w)}{Z}$$

Ans

Ans 1(C): for a decision tree with boolean features, we can use any differentiable function as a splitting function. eg. We can use sigmoid funcⁿ. $\phi(x) = \frac{1}{1+e^{-x}}$. The sigmoid function can split the data into left & right with probability. If we take maximum probability at the output node, we can classify that output. This decision tree can easily be represented by a neural network with sigmoid function as an activation function.

Ans.

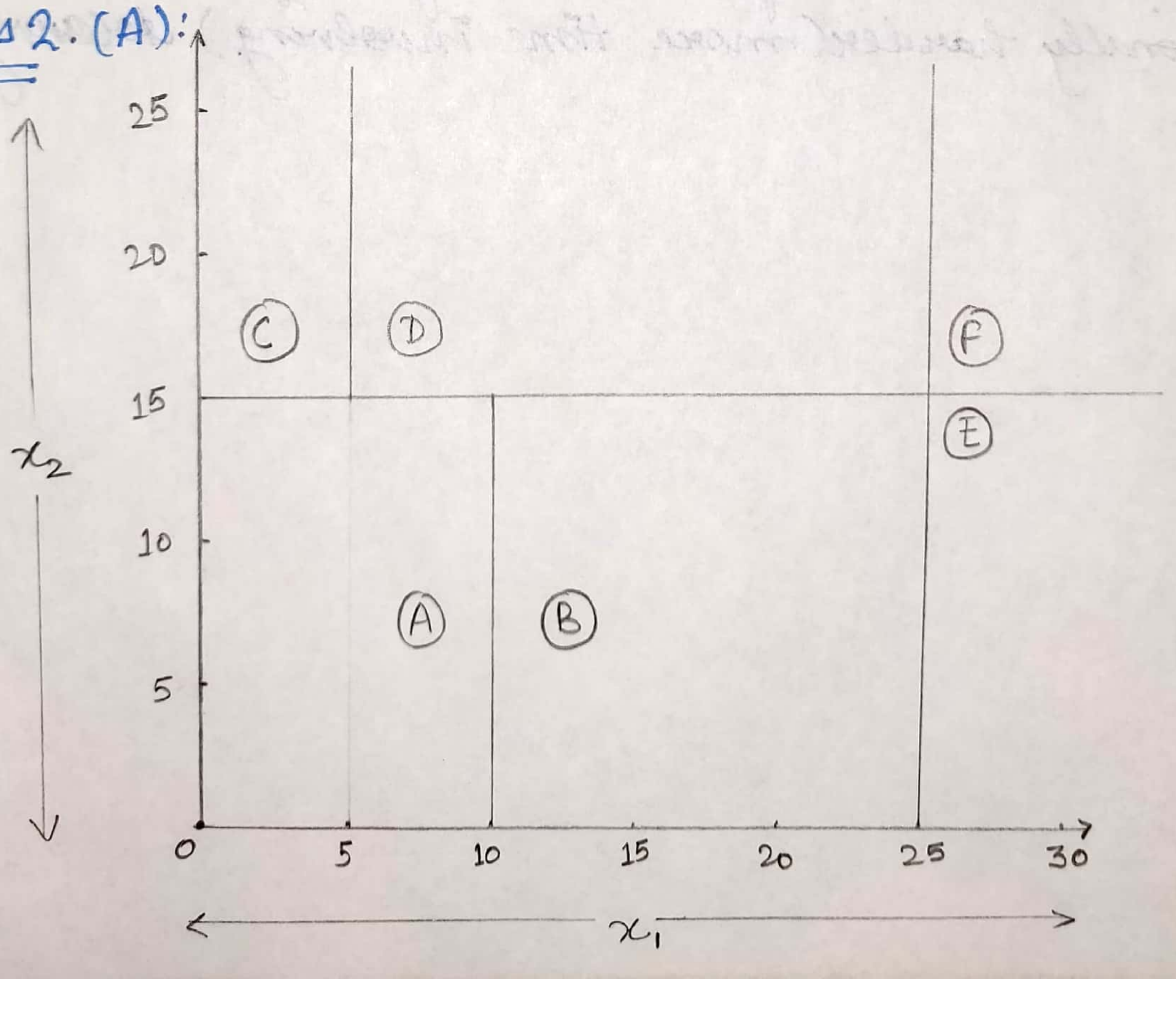
Ques 2. Consider the following decision tree :



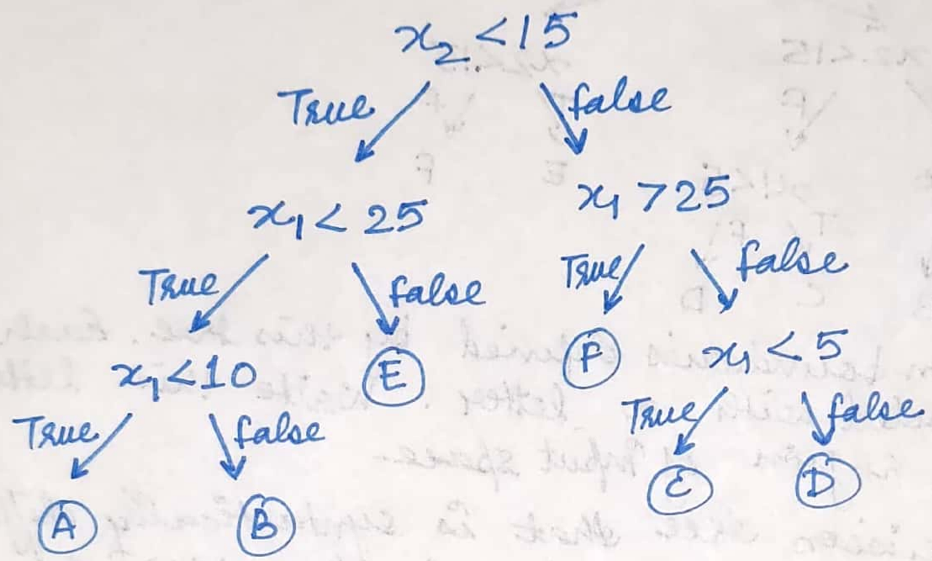
(A) Draw the decision boundaries defined by this tree. Each leaf of the tree is labeled with a letter. Write this letter in the corresponding region of input space.

(B) Give another decision tree that is syntactically different but defines the same decision boundaries. This demonstrates that the space of decision trees is syntactically redundant. How does this redundancy influences learning (does it make it easier or harder to find an accurate tree) ?

Ans 2. (A):



Ans 2 (B): A different decision tree that defines the same decision boundary, is as follows:-



The above tree doesnot have any difference in the accuracy as compared to the previous decision tree. There can be a difference in terms of computation time. Thus, the redundant information might make it computationally harder (more time involving) to converge.

Ques 3. In the basic decision tree algorithm, we choose the features /value pair with the maximum information gain as the test to use at each internal node of the decision tree. Suppose we modified the algorithm to choose at random from among those features/value combinations that had non-zero mutual information, & we kept all other parts of the algorithm unchanged.

(A) What is the maximum no. of leaf nodes that such a decision tree could contain if it were trained on m training examples?

(B) What is the maximum no. of leaf nodes that a decision tree could contain if it were trained on m training examples using the original maximum mutual information version of the algorithm? Is it bigger, smaller, or the same as your answer to (b)?

(C) How do you think this change (using random splits v/s maximum information mutal info. splits) would affect the accuracy of the decision trees produced on average? Why?

Solution: Ans3(A) If the decision tree uses random split method to split a dataset of m training examples then in the worst possible case, our algorithm will generate case where the total number of leaves is equal to the total no. of training examples. Thus, in the worst possible case, the maximum no. of leaf nodes that the tree could contain if it were trained on m training examples is 'm'. Ans.

Ans 3(B): If we use original maximum mutual informa-tion gain criterion to train our decision tree model for m training examples, then in the worst case it can act like the random algorithm in previous part such that it splits in such a way that the no. of

leaf nodes is equal to the no. of training examples only
∴ In this case also, max no of leaf nodes = m. The size of the tree should remain the same.

Ans 3 (C): Since this decision tree will also converge to produce the same decision boundary, the accuracy of the decision tree doesnot change.
A problem can have any no. of optimal decision trees. The only problem with random splitting is that we can have a longer more complex decision tree that takes a lot of computation time to converge.

Ques 4. Consider the following training set :

A	B	C	Y
0	1	1	0
1	1	1	0
0	0	0	0
1	1	0	1
0	1	0	1
1	0	1	1

Learn a decision tree from the training set shown above using the information gain criterion.

Solution: A, B and C act as features for our output Y.
for all 3 features, we find out Entropy H.

At the root, $P(y=1) = \frac{3}{6} = \frac{1}{2}$ $\quad P(y=0) = \frac{3}{6} = \frac{1}{2}$

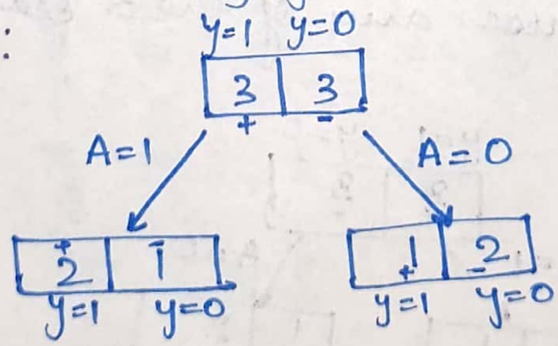
$$\therefore H(y) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$$

$$= \frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2$$

$$= \frac{1}{2} + \frac{1}{2} = 1$$

lets calculate the entropy for each feature.

For feature A :



left Branch: $H(y \mid A=\text{true}) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}$
$= 0.9133$

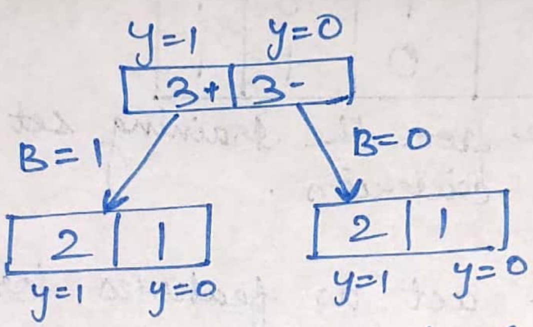
Right branch: $H(y|A=\text{false}) = -\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3} = 0.9133$

$\therefore$ combined uncertainity $= \frac{3}{6} \times 0.9133 + \frac{3}{6} \times 0.9133$

$= 0.9133$

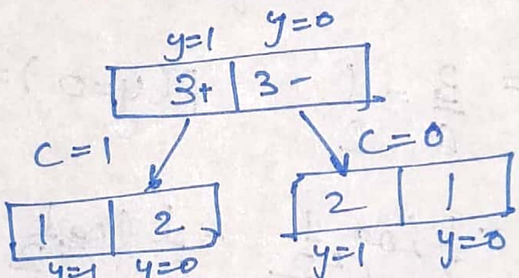
$\therefore \boxed{H(y|A) = 0.9133}$

for feature B



Since it is identical to A, $\boxed{H(y|B) = 0.9133}$

for feature C



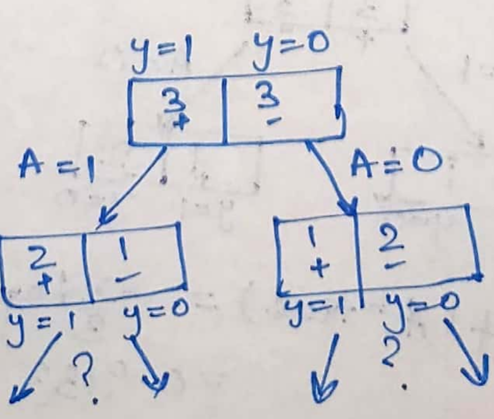
This is also identical to A&B $\therefore \boxed{H(y|C) = 0.9133}$

We know that, Information Gain, $I(x,y) = H(y) - H(y|X)$

$\therefore I(A,y) = I(B,y) = I(C,y) = 1 - 0.9133$
$= 0.0867$

Thus, we can use any of the 3 features as root & proceed.

Let us take A.



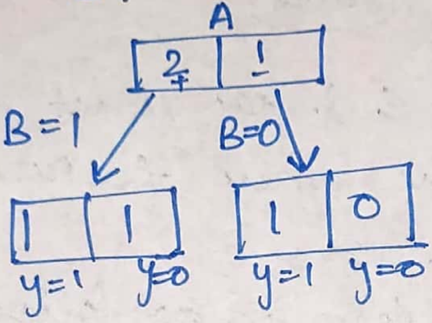
We need to decide which feature should do the next split.

We'll repeat the same process as above.

for left branch, $p(y=0) = \frac{1}{3}$ $\quad p(y=1) = \frac{2}{3}$

$$\therefore H(y) = -\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right)$$
$$= 0.9133$$

If we put feature B.



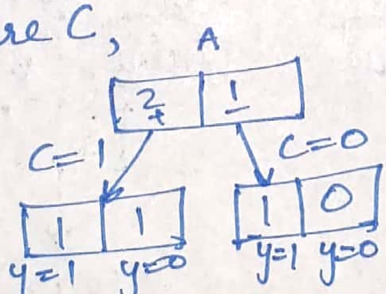
Left Branch $\quad H(y|B=\text{true}) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$

Right Branch $\quad H(y|B=\text{false}) = -\frac{1}{1}\log 1 = 0$

$\therefore$ combined uncertainity $= \frac{2}{3}\times 1 + 0 = 0.666$

$$\therefore \boxed{H(y|B) = 0.\bar{6}}$$

If we put feature C,



This tree is identical to the one with Branch B.

$$\therefore \boxed{H(y|C) = 0.\bar{6}}$$

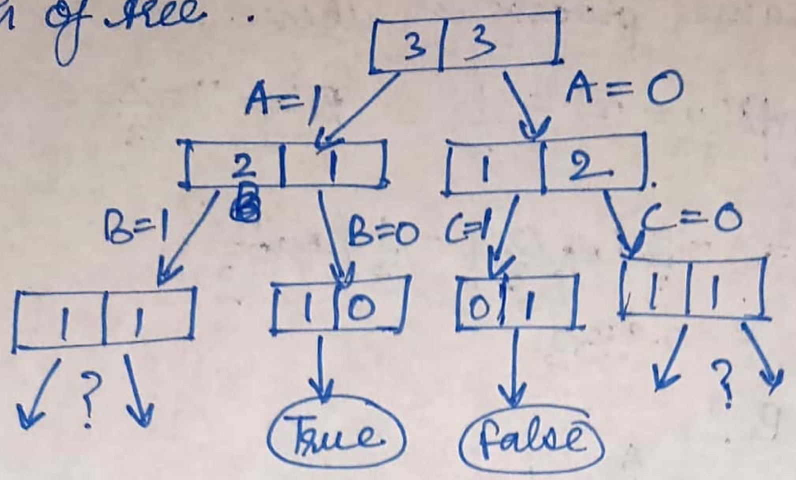
$\therefore$ We can choose either B or C

$\Rightarrow$ Let us choose B.

Since the right branch for which A=0 is also identical there also we can choose either B or C.

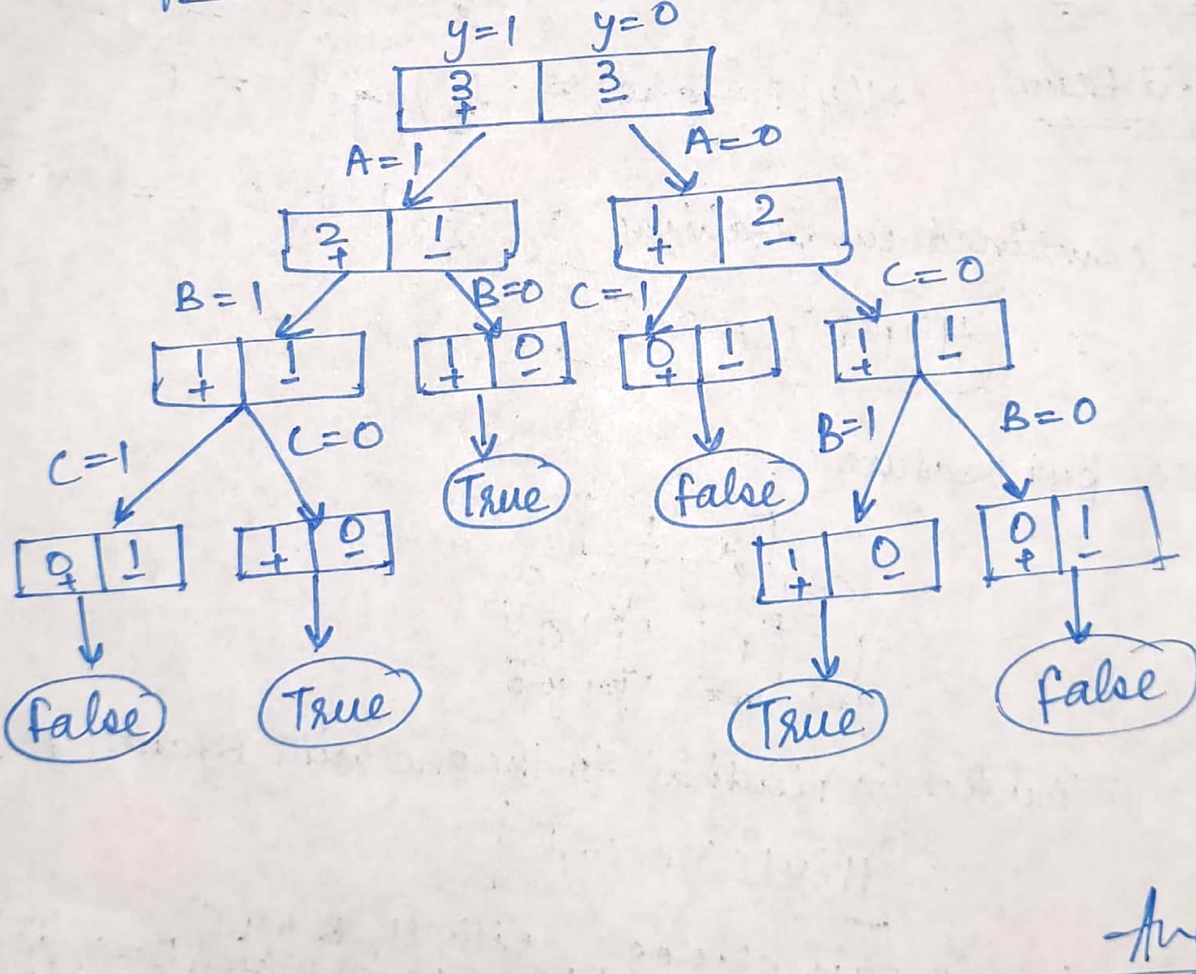
Therefore, ~~out de~~ we need to decide on the

next depth of tree.
~~Since this~~



for left branch we use C & for right branch we use B, because these are the only features left in the corresponding branches.

∴ The final decision tree is as follows :-



Ans

Ques 5. Please show that in each iteration of Adaboost, the weighted error of h_i on the updated weights D_{i+1} is exactly 50%. In other words, $\sum_{j=1}^{N} D_{i+1}(j) I(h_i(X_j) \neq y_j) = 50\%$

Solution: We know that,

$$D_{i+1}(j) = D_i(j) e^{\alpha_i} \quad ; h_i(x_j) \neq y_j$$

$$\& \quad D_{i+1}(j) = D_i(j) e^{-\alpha_i} \quad ; h_i(x_j) = y_j$$

i.e. we increase the weights for next iteration if we classified our example incorrectly & we decrease the weight if we classified correctly.

$$\therefore \sum_{j=1}^{N} D_{i+1}(j) I(h_i(X_j) \neq y_j) = \sum_{h_i(X_j) \neq y_j} D_i(j) e_i^{\alpha} \quad -(1)$$

Now, $\frac{1}{Z_i} \sum_{h_i(x_j) \neq y_j}^{N} D_i(j) e^{\alpha_i} = P_{D_{i+1}} [h_i(x_j) \neq y_j]$

$$\therefore \frac{1}{Z_i} \sum_{h_i(x_j) \neq y_j}^{N} D_i(j) e^{\alpha_i} = e^{\alpha_i} . \epsilon_i \quad \forall \epsilon \text{ is the error rate}$$

Also, $\alpha_i = \frac{1}{2} \log\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$ Putting each value in eq^n (1)

$$\therefore \sum_{j=1}^{N} D_{i+1}(j) I(h_i(x_j) \neq y_j) = \frac{\epsilon_i}{Z_i} \sqrt{\frac{1-\epsilon_i}{\epsilon_i}} = \frac{\sqrt{\epsilon_i(1-\epsilon_i)}}{Z_i}$$

Similarly, $\sum_{j=1}^{N} D_{i+1}(j) I(h_i(x_j) = y_j) = \frac{\sqrt{(1-\epsilon_i)\epsilon_i}}{Z_i}$

Also, $\frac{1}{Z_i}\left[\sum_{h_i(x_j) \neq y_j}^{N} D_i(j) + \sum_{h_i(x_j) = y_j}^{N} D_i(j)\right] = 1 \quad \therefore \sum_{j=1}^{N} D_{i+1}(j) I(h_i(X_j) \neq y_j) = 50\%$

Ans.

Ques 6. In class we showed that Adaboost can be viewed as learning an additive model via functional gradient descent to optimize the following exponential loss function:

$$\sum_{i=1}^{N} \exp\left(-y_i \sum_{\ell=1}^{L} \alpha_\ell h_\ell(x_i)\right)$$

Our derivation showed that in each iteration ℓ, to minimize this objective we should seek an h_ℓ that minimizes the weighted training error, where the weight of each example $w_i^\ell = \exp(-y_i \sum_{t=1}^{\ell-1} \alpha_t h_t(x_i))$ prior to normalization. Show how this definition of w_i^ℓ is proportional to the $D_\ell(i)$ defined in Adaboost.

Solution: We know that:

$$D_{t+1}(i) = D_t(i) \times \begin{cases} e^{\alpha_t} & \text{if } h_t(x_j) \neq y_j \\ e^{-\alpha_t} & \text{if } h_t(x_j) = y_j \end{cases}$$

$$\Rightarrow D_{t+1}(i) = D_t(i) e^{-y_i \alpha_t h_t(x_i)}$$

$$\Rightarrow D_t(i) = D_{t-1}(i) e^{-y_i \alpha_{t-1} h_{t-1}(x_j)}$$

$$= D_{t-2}(i) e^{-y_i \alpha_{t-2} h_{t-2}(x_j)} \times e^{-y_i \alpha_{t-1} h_{t-1}(x_j)}$$

$$= D_1(i) e^{-y_i \alpha_1 h_1(x_j)} \times e^{-y_i \alpha_2 h_2(x_j)} \times \ldots \times e^{-y_i \alpha_{t-1} h_{t-1}(x_j)}$$

$$= D_1(i) e^{-y_i \sum_{t=1}^{t-1} \alpha_t h_t(x_j)} \quad \text{—(I)}$$

A.T.Q, weight of each training example $w_i^\ell = \exp(-y_i \sum_{t=1}^{\ell-1} \alpha_t h_t(x_j))$ —(II)

On comparing eq^n (I) & (II)

$$\boxed{D_\ell(i) \propto w_\ell^i}$$

Hence proved.