Aashish Adhikari
CS 534 Machine Learning Written Assignment 3

New Link
Date
Page

(8) Solution:

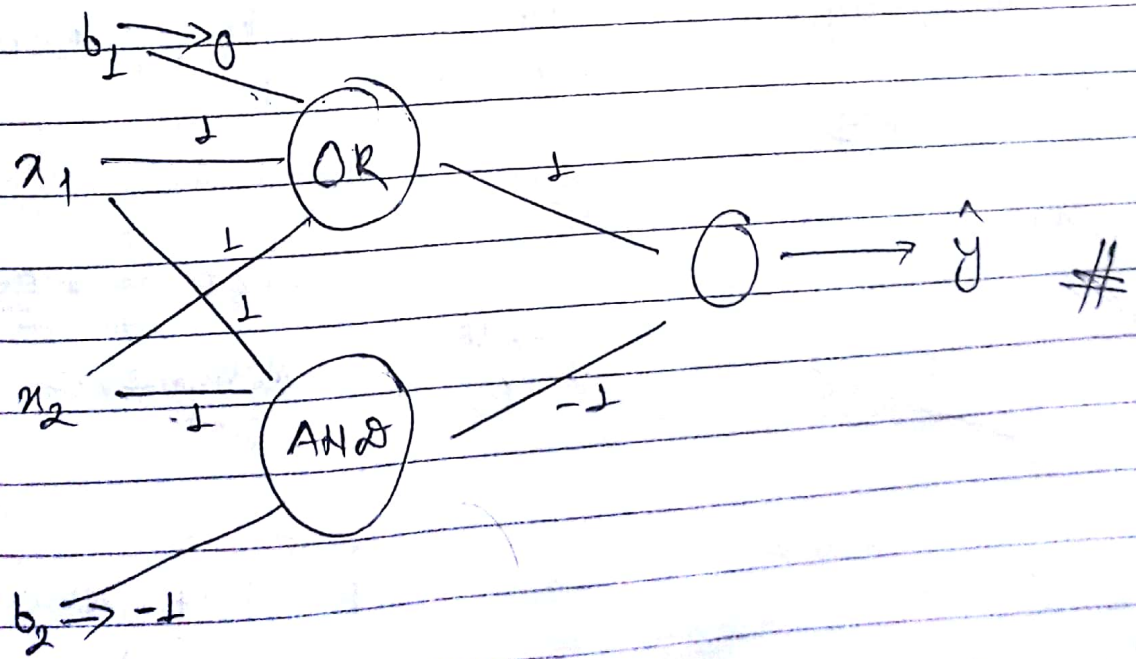
XOR can be implemented by making use of AND and OR.

x_1	x_2	OR	AND	XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

XOR is the same as OR except for the case when $x1 = x2 = 1$.

⇒ Subtracting AND from OR gives XOR.

So, our network becomes



#

b. Solution

Gaussian Naive Bayes classifier given as

$$P(Y=1|X) = \frac{1}{1+e^{w_0 + \sum_{i=1}^{n} w_i X_i}}$$

[from Tom Mitchell's Book Chapter 3 eq. 22]

where weights $w_1 \ldots w_n$ are given by

$$w_i = \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2}$$

$$w_0 = \ln \frac{1-\pi}{\pi} + \sum_{i}^{n} \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}$$

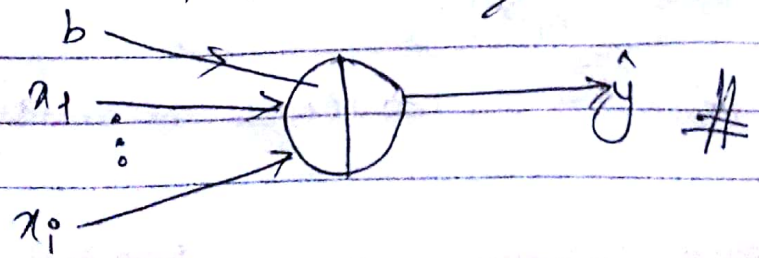
Also, we have the parametric probabilistic approach of Logistic Regression assumed is

$$P(Y=1|X) = \frac{1}{1+e^{w_0 + \sum_{i=1}^{n} w_i X_i}}$$

[from eq. 16]

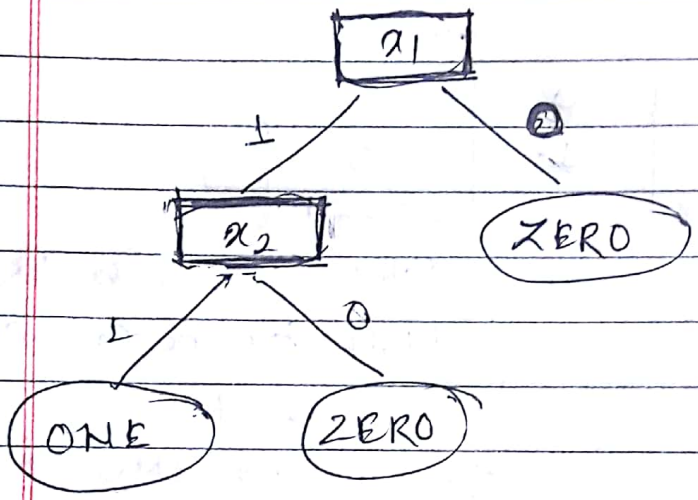
As observed, both the classifiers have an identical form.

Since a logistic regression unit is just a single layer Neural Network with 1 sigmoid unit, this implies that any Naive Bayes classifier ~~for~~ with Boolean features can be implement using a Neural Network.



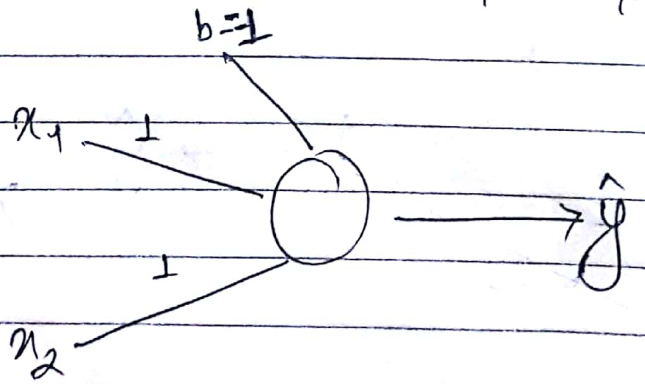
c). Solution:

We know that a decision tree can represent any boolean-valued function. For example, it can represent AND as x_1 AND x_2



We also know that a 2-layer NN can represent any Boolean function.

For example, x_1 AND x_2 can be represented using only 1-layer NN



where ~~$\hat{y}$ = sign~~

Activation = $x_1 w_1 + x_2 w_2 + b$

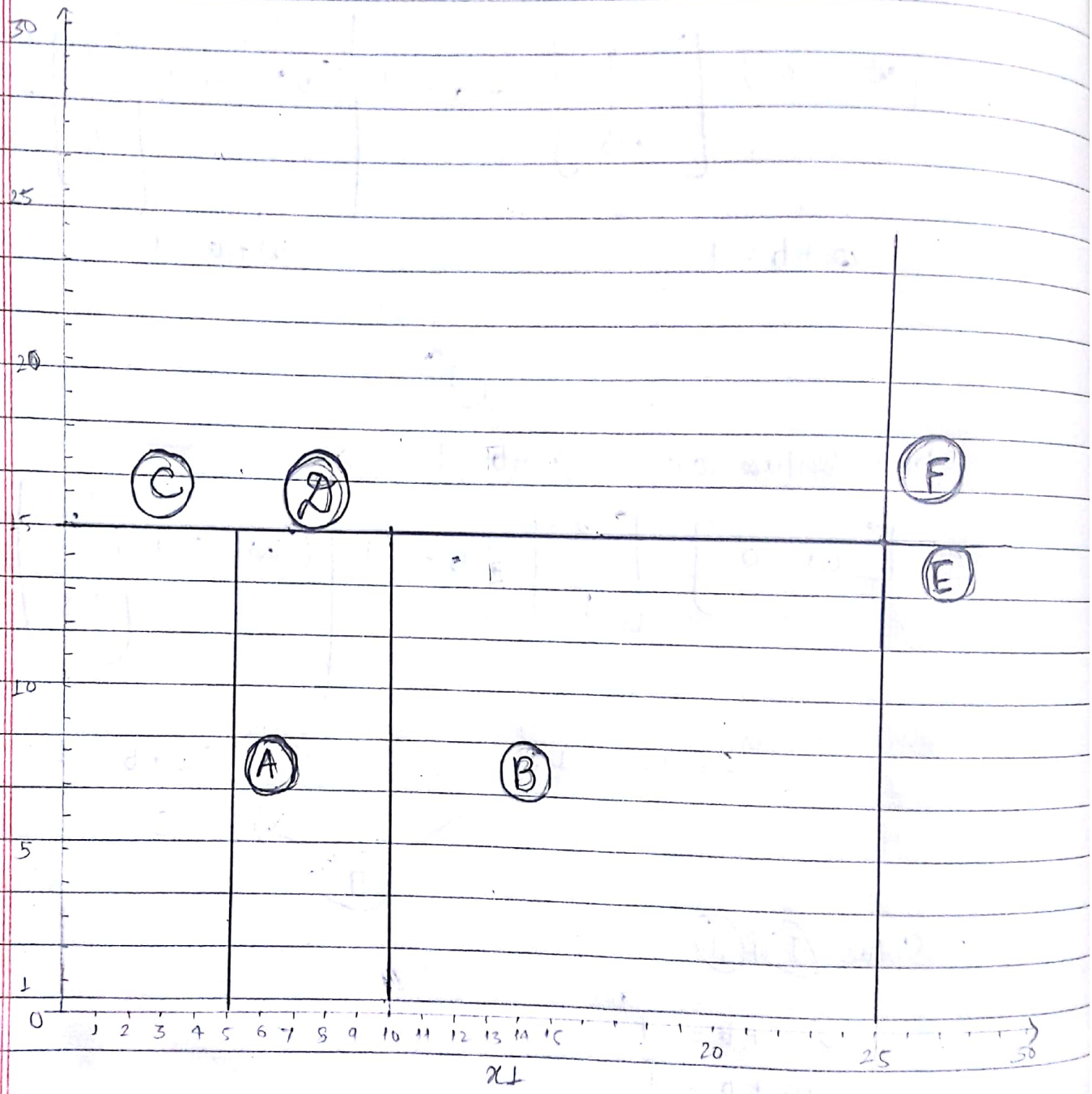
$= x_1 + x_2 - b$

$$\hat{y} = \begin{cases} 1 \text{ if Activation} > 0 \\ 0 \text{ if Activation} \leq 0 \end{cases}$$

Thus, a NN can be constructed to implement a decision tree classifier with boolean features.

2.

a) solution



b) Solution

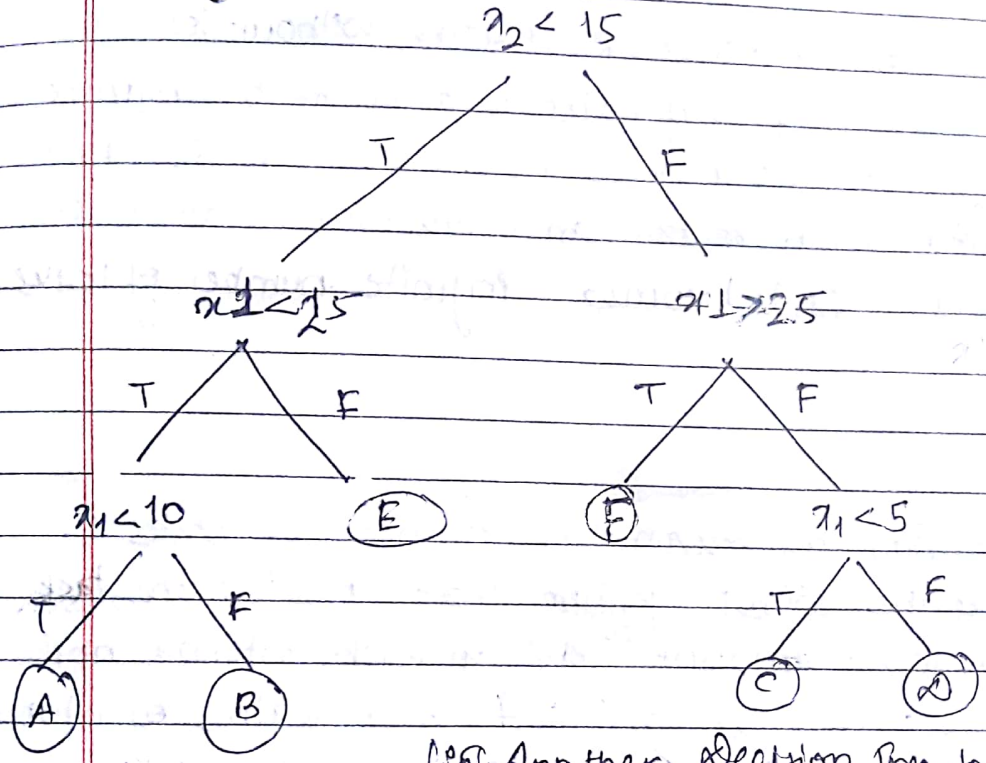


Fig: Another decision tree for the same question.

In my view, this doesnot pose any problem. The only difference is the length of the decision tree. This may make difference in computational time but both the trees reach the same final solution. So, the rate of learning may vary but the convergence is identical.

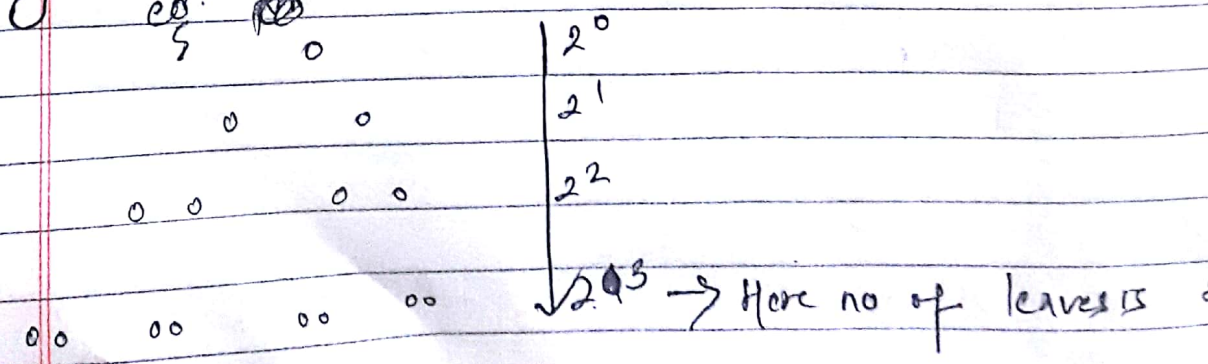
3 a. Solution

Since we are choosing the nodes at random without looking at the gain, I think the number of leaf nodes is going to be ... It is because we never stop choosing a node i.e., we remove the fact that we choose the node with maximum gain. So, we don't run out of the features and it just ...

3 a Solution

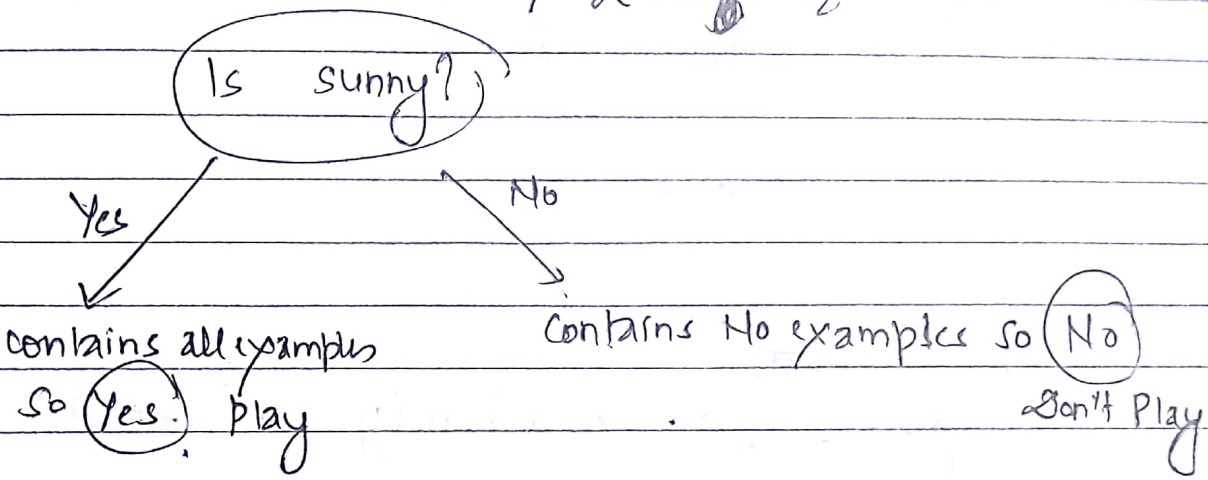
In such a scenario, the maximum number of nodes is possible if only we take the worst feature that reduces the task of classification by the least amount i.e., at each internal node, we choose a feature that gives us the least value of information gain. This happens when the dataset is divided such that the no of examples in each label is equal and also, the number of examples for each possible value of a feature x_i is same for every possible value for every feature. In such a case, the classification decreases logarithmically and the [no of leaf nodes maximum possible] given that the splits are binary is 2^m.

eg:



b) Solution

If we use the original mutual information version, we get the best possible solution. So, it is always going to be at least as small as the answer in the first case, but never bigger. In the best case, it could contain only 2 leaves eg.



c) Solution

The accuracy of the decision tree produced doesnot change because a problem can have multiple optimal decision trees. The ordering of the features that we choose doesn't affect the fact that a node will always classify a label of "Play Tennis" as "Play Tennis" no matter at what point of time it is classified. The fact that we consider all the features no matter the order in the decision tree makes sure we get an accurate tree. The only problem with the random choice is that we get a longer complex decision tree.

4. Solution

We have 3 features.

For all 3, we find out Entropy H.

~~$H_{A=}$~~

At the root $H(y=1) = -\frac{3}{6} \times \log_2 \frac{3}{6} - \frac{3}{6} \times \log_2 \frac{3}{6}$

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

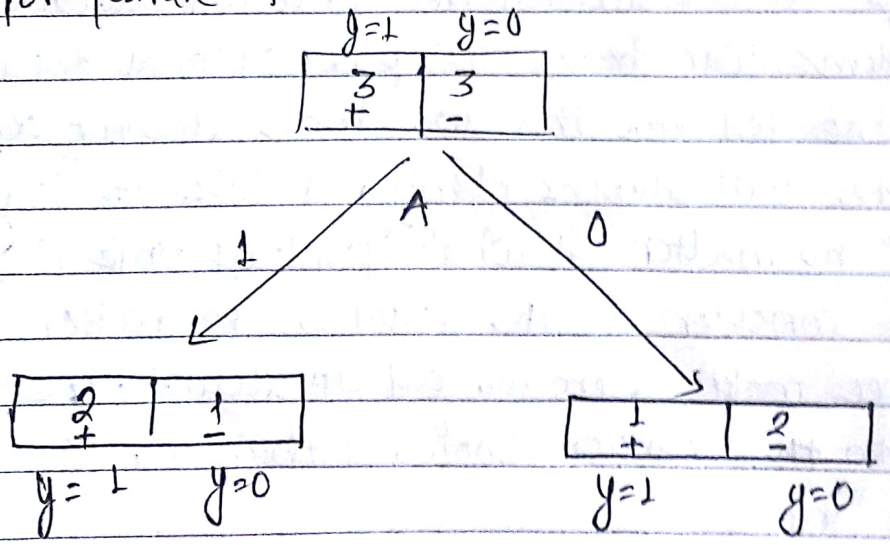
$= 1.$

For feature A,

$H(y|A) = P(A=0)\, H(y|A=0) + P(A=1)\, H(y|A=1)$

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

For feature A



Left Branch $H(y|x=\text{true}) = -\frac{2}{3} \times \log_2 (2/3) - \frac{1}{3} \times \log_2 (1/3)$

$= \frac{0.584 \times 2}{3} + \frac{1.584}{3}$

$= 0.9133$

New Link
Date
Page

Right Branch $H(y|x=\text{false}) = -\frac{1}{3}\times\log_2(1/3) - \frac{2}{3}\times\log_2(2/3)$

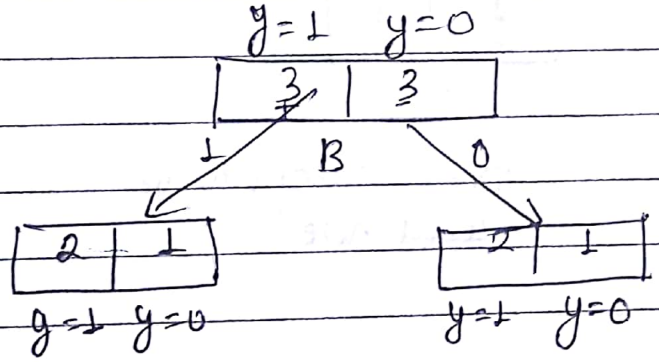
$= 0.9133$

∴ Combined uncertainty is the weighted entropy of all the branches

$= \frac{3}{6} \times 0.9133 + 3/6 \times 0.9133$

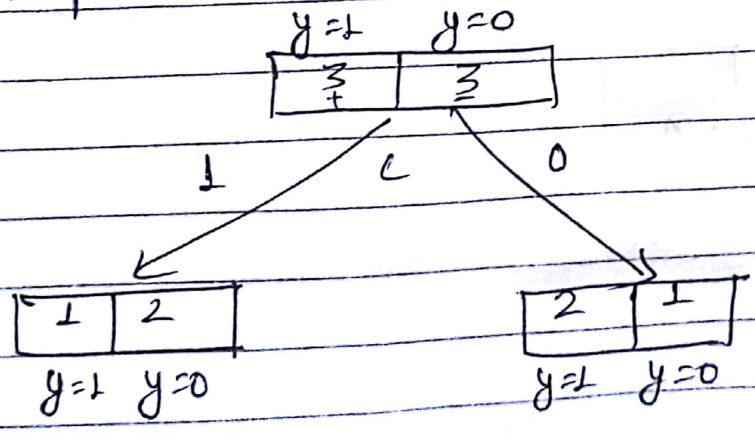
$= 0.9133. \Rightarrow H(y|A) = 0.9133$

for feature B



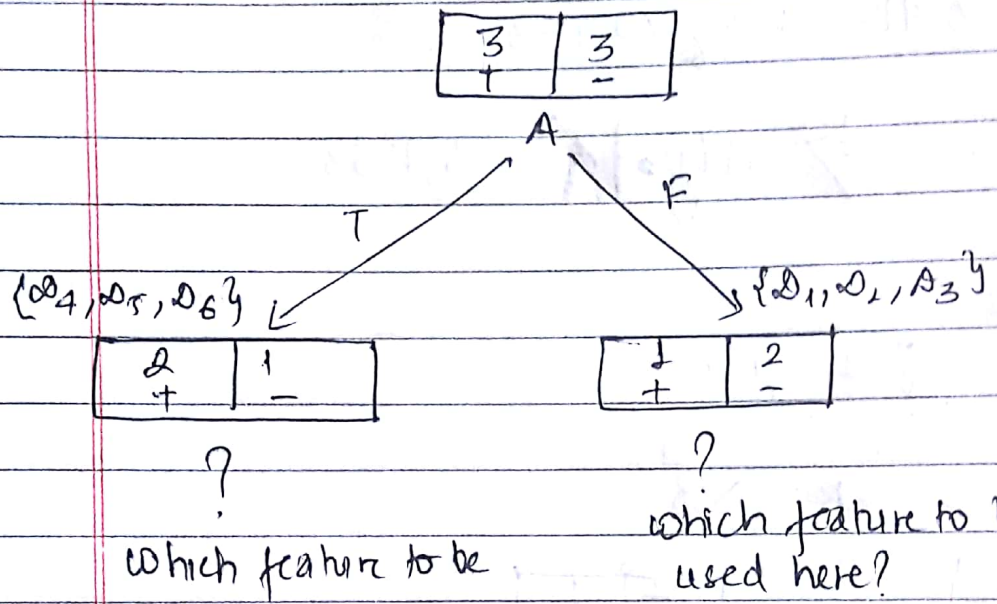
Since it is identical to feature A, $H(y|B) = 0.9133$

for feature C



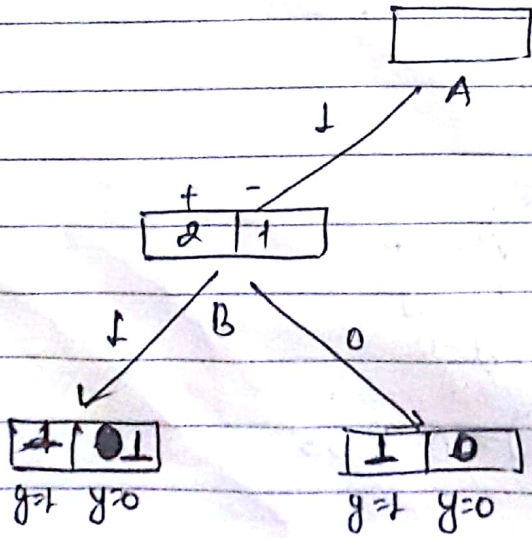
Identical $\Rightarrow H(y|C) = 0.9133$

So, Information Gain, $I(x,y) = H(y) - H(y|x)$ is going to be same for all 3 features.
Thus, we can use any of the 3 features as the root node and proceed.



$H(y)$ at left branch $= -\frac{2}{3} \times \log_2\left(\frac{2}{3}\right) - \frac{1}{3} \times \log_2\left(\frac{1}{3}\right)$
$= 0.9133.$

If we put feature B,



~~If we put B there~~

∴ left branch $H(y|B=true)$

$= -\frac{1}{2} \times \log_2(1/2) - \frac{1}{2} \times \log_2(1/2)$

$= 1/2 + 1/2$

$= 1.$

right branch $H(y|B=false)$

$= -\frac{1}{2} \times \log_2(1) - 0$

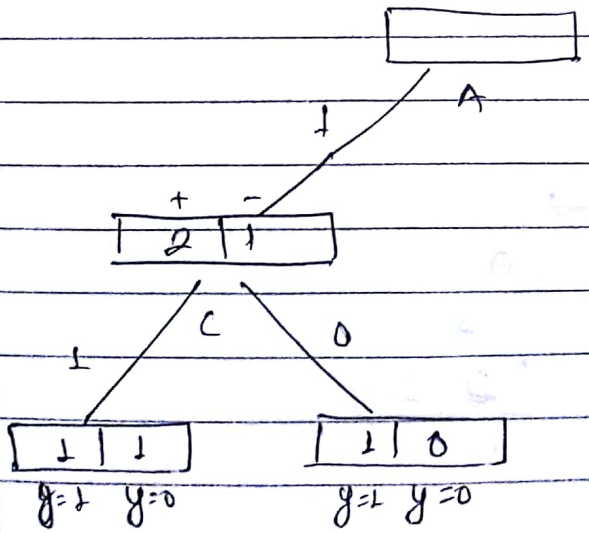
$= 0$

⇒ combined uncertainity

$= \frac{2}{3} \times 1 + 0$

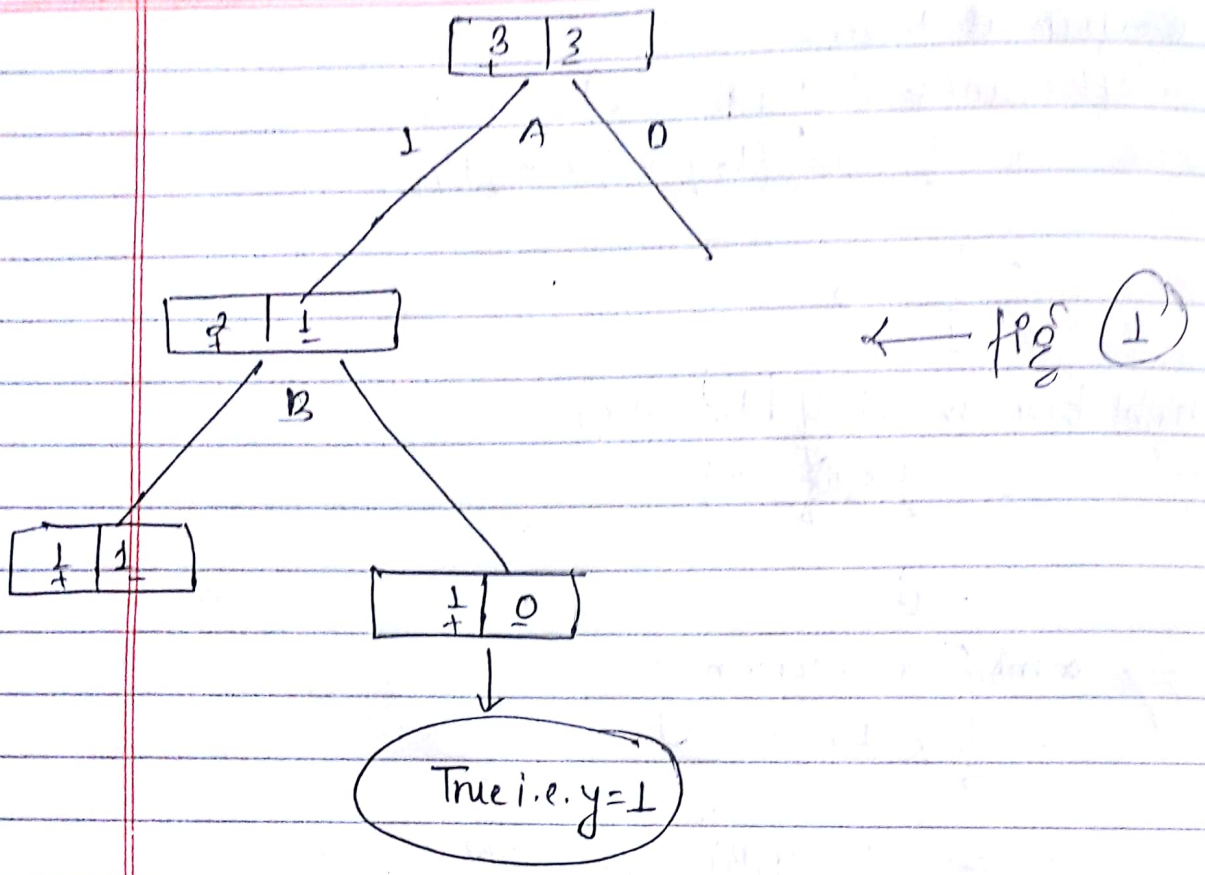
$= \frac{2}{3}$ $H(y|B) = 2/3 = 0.66..$

If we put feature C,



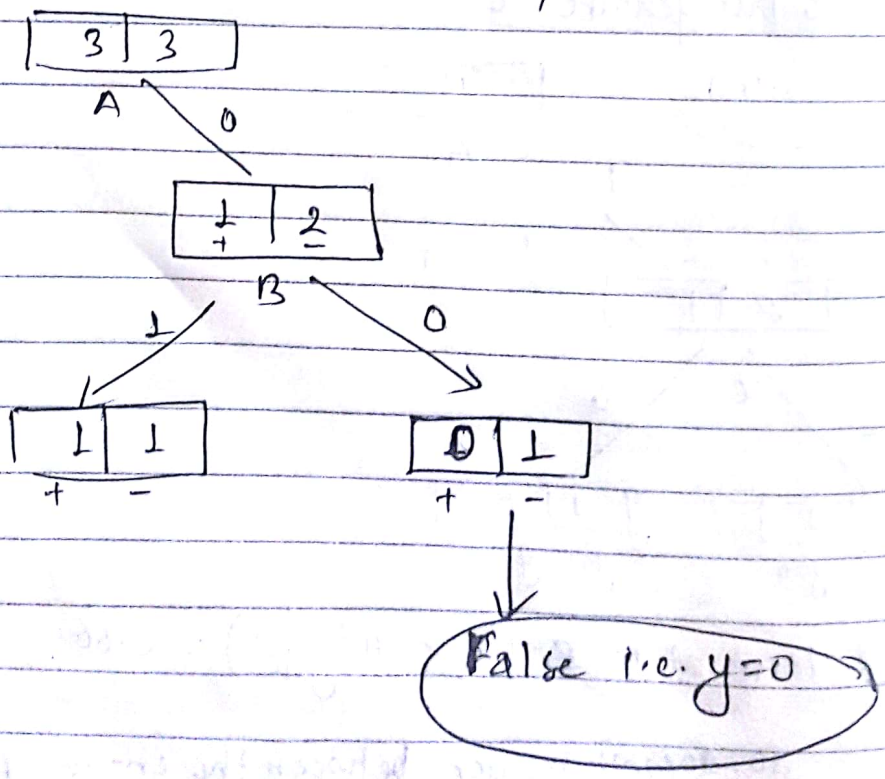
Identical to B ⇒ $H(y|C) = 0.66.$

So, doesn't matter between the choice of B & C here.



← पहले (1)

For the right branch of A, let's say we use B.



~~If we used~~

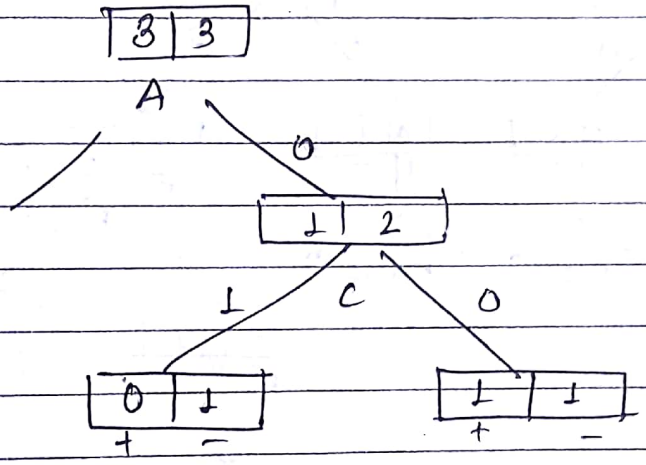
$H(y|B=true) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2})$

$= 1$

$H(y|B=false) = 0$

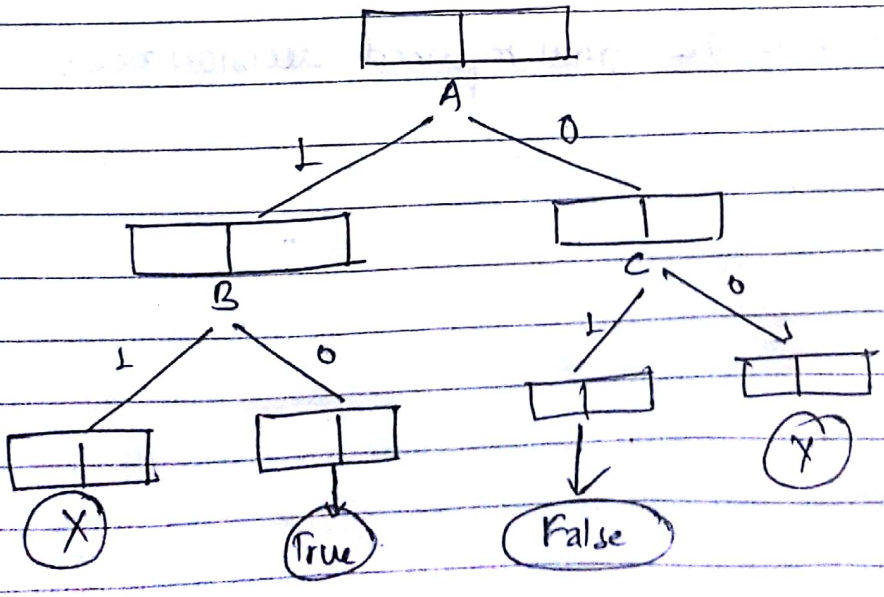
combined $H(y|B) = 0.66$.

If we put C instead,

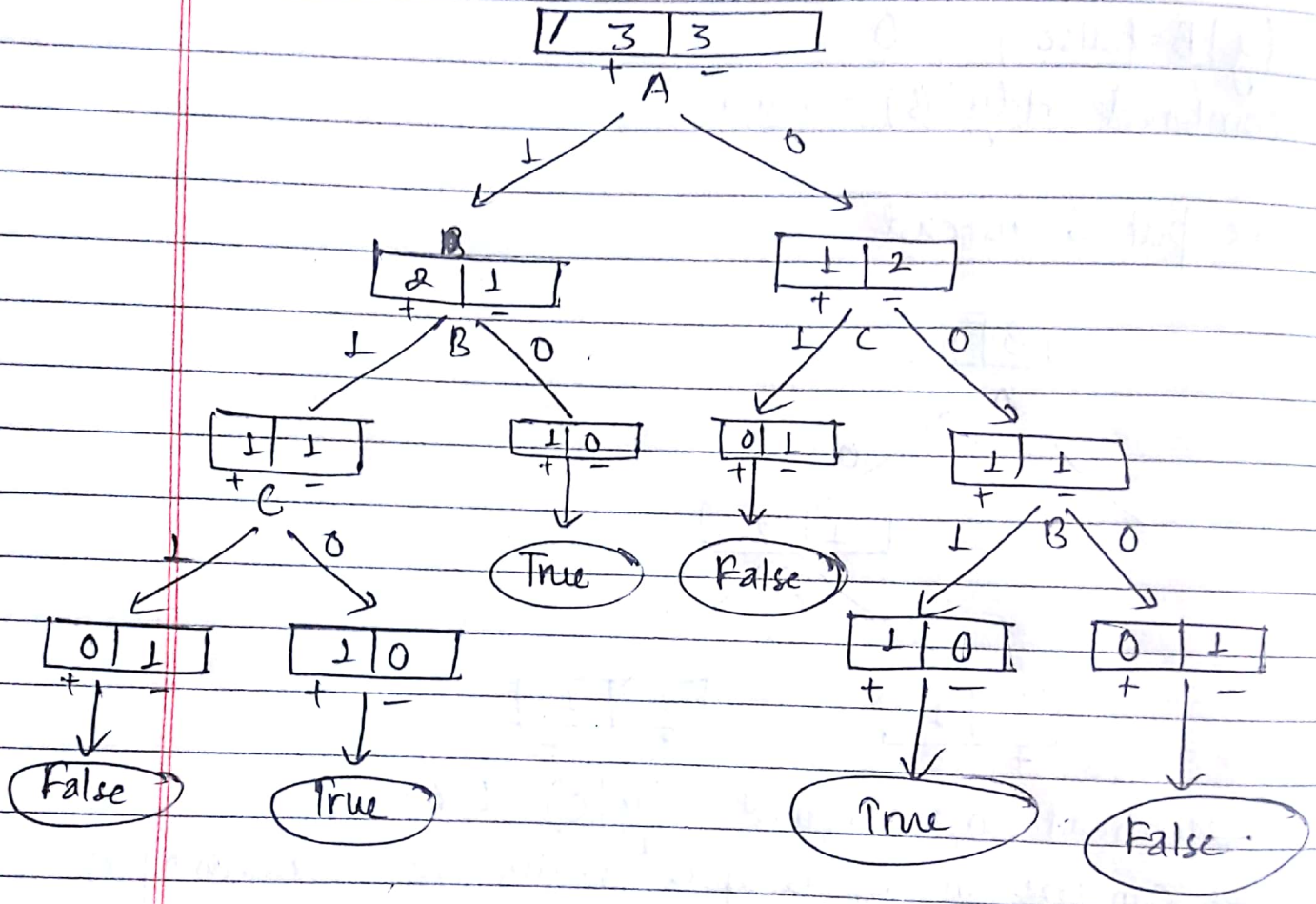


Identical so, combined $H(y|C) = 0.66$.

We can use any of B of C as the right branch of A.



Now at branches X & Y, we can use features C & ~~and~~ B respectively because those are the only features left to use there.



which is the final required decision tree.

Q. 5. Solution

$$\sum_{j=1}^{N} \mathcal{D}_{i+1}(j)\, I(h_i(x_j) \neq y_j) = \sum_{h_i(x_j)\neq y_j}^{N} \mathcal{D}_i(j)\, e^{\alpha} \quad \text{---(1)}$$

Normalizing the sum of $\mathcal{D}_i(j)\, e^{\alpha_i}$ to make sure it is a distribution that adds to a probability of 1.

Note: $\mathcal{D}_i(j)$ gives a probability of seeing this example in the next iteration.

$$\Rightarrow \frac{1}{Z_i} \sum_{h_i(x_j)\neq y_j}^{N} \mathcal{D}_i(j) e^{\alpha_i} = \underset{\mathcal{D}_{i+1}}{P}\left(h_i(x_j) \neq y_j\right)$$

Also, $\frac{1}{Z_i} \sum_{h_i(x_j)\neq y_j}^{N} \mathcal{D}_i(j) e^{\alpha_i} = e^{\alpha_i} \cdot \epsilon_i$ where $\epsilon_i = \text{error}(h_i, S, \mathcal{D}_i)$ & $\alpha_i = \frac{1}{2} \ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$

Now, eqn (1) becomes

$$\sum_{j=1}^{N} \mathcal{D}_{i+1}(j)\, I(h_i(x_j) \neq y_j) = \frac{\epsilon_i}{Z_i} \sqrt{\frac{1-\epsilon_i}{\epsilon_i}} \quad \& \quad \text{---(2)}$$

$$\sum_{j=1}^{N} \mathcal{D}_{i+1}(j)\, I(h_i(x_j) = y_j) = \frac{\sqrt{(1-\epsilon_i)\epsilon_i}}{Z_i} \quad \text{---(3)}$$

However we know that the sum of probabilities is 1

So, $$\frac{1}{Z_i}\left[\sum_{h_i(x_j)\neq y_j}^{N} \mathcal{D}_i(j) + \sum_{h_i(x_j)=y_j}^{N} \mathcal{D}_i(j)\right] = 1$$

But from eqns (2) & (3) the two terms are equal.

Hence,

New Link
Date
Page

$$\sum_{j=1}^{N} D_{i+1}(j)\, I(h_i(x_j) \neq y_j) = 50\%$$

Hence, proved. #

Q.6. Solution

From AdaBoost Algorithm, the probability of seeing an example in the next iteration is given by

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

Combining in a single form gives

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

However, the recursive formulation for D_t can be made as

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

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

$$= D_1(i) e^{-y_i \alpha_1 h_1(x_j)} \times e^{-y_i \alpha_2 h_2(x_j)} \times \cdots$$

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

From question, $w_t^i = e^{\left(-y_i \sum_{t=1}^{t-1} \alpha_t h_t(x_i)\right)}$ ——— (2)

By comparision of (1) & (2), $\boxed{D_t(i) \propto w_t^i}$, proved. #