CS534 - HW 1

Keith Chester

Due date: June 12 2022

Problem 1

In problem 1, we are tasked with creating a recursive and linear time agent for following propositonal logic statements. The work associated with this problem can be found in *problem1.py*.

Problem 2

In this problem we are exploring a first-order logical knowledge base and writing out logical expressions utilizing it. The knowledge base is represented as:

- CopyOf(d, a) - Predicate - Disk d is a copy of album a
- Owns(p, d) - Predicate - Person p owns disk d
- Sings(p, s, a) - Predicate - Album a inludes a recording of song s sung by person p
- Wrote(p, s) - Person p wrote song s

We are also injecting the following constants:

- $McCartney$ - a person
- $Gershwin$ - a person
- $BHoliday$ - a person
- Joe - a person
- $EleanorRigby$ - a song
- $TheManILove$ - a song
- $Revolver$ - an album

Within this, we express the following statements using first-order logic:

(a) $\text{Wrote}(Gershwin, TheManILove)$

(b) $\neg\,\text{Wrote}(Gershwin, EleanorRigby)$

(c) $\text{Wrote}(Gershwin, TheManILove) \vee \text{Wrote}(McCartney, TheManILove)$

(d) $\exists s\,\text{Wrote}(Joe, s)$

(e) $\exists d\,\text{Owns}(Joe, CopyOf(d, Revolver))$

(f) $\forall s\,\text{Sings}(McCartney, s, Revolver) \Rightarrow \text{Wrote}(McCartney, s)$

(g) $\forall s\,\text{Sings}(p, s, Revolver) \neg\,\text{Wrote}(Gershwin, s)$

(h) $\forall s\,\text{Wrote}(Gershwin, s) \Rightarrow \text{Sings}(p, s, a)$

(i) $\exists a \forall s \, \text{Sings}(p, s, a) \, \text{Wrote}(Joe, s)$

(j) $\exists d, a \, \text{Owns}(Joe, \text{CopyOf}(d, a)) \wedge \text{Sings}(BHoliday, s, a)$

(k) $\exists d_i \, \text{CopyOf}(d_i, a) \forall a \, \text{Sings}(McCartney, a, s) \wedge \forall \, \text{Owns}(Joe, d)$

(l) $\exists d \, \text{CopyOf}(d, a) \forall a \, \text{Sings}(BHoliday, s, a) \wedge \text{Owns}(Joe, d)$

Problem 3

In this queston, we are looking at the following table of three binary input atributes, and a singular binary output:

Example	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1

a.

Using the Gini Index, we aim to create a decision tree for this data. For Gini Index, we are looking at the probability distribution of a particular variable being wrong when randomly chosen. From this we can begin to branch our tree into a set of decisions that lead us to a particular preditctive result.

$$Gini = 1 - \sum_{i=1}^{n} p_i^2 \tag{1}$$

...where p_i is the probability of a given object being classified into a particular class.

So, calculating the Gini index for our given data:

For A_1, we are look at the indexes of positive and negative outputs:

$$A_1 = 1, y = 1 \rightarrow \frac{2}{4} = 0.5 \tag{2}$$

$$A_1 = 1, y = 0 \rightarrow \frac{2}{4} = 0.5 \tag{3}$$

...for the Gini calculation of:

$$1 - (0.5^2 + 0.5^2) = 0.5 \tag{4}$$

$$A_1 = 0, y = 1 \rightarrow \frac{0}{1} = 0 \tag{5}$$

$$A_1 = 0, y = 0 \rightarrow \frac{1}{1} = 1 \tag{6}$$

...for the Gini calculation of:

$$1 - (0^2 + 1^2) = 0 \tag{7}$$

and the weighted sum of these Gini indexes to create the Gini index for A_1:

$$\sum p_i * g_i = (\frac{4}{5} * 0.5 + \frac{1}{5} * 0) = 0.4 \tag{8}$$

Now we continue this for A_2:

$$A_2 = 1, y = 1 \rightarrow \frac{2}{3} = 0.67 \tag{9}$$

$$A_2 = 1, y = 0 \rightarrow \frac{1}{3} = 0.33 \tag{10}$$

...for the Gini calculation of:

$$1 - (0.67^2 + 0.33^2) = 0.44 \tag{11}$$

$$A_2 = 0, y = 1 \rightarrow \frac{0}{2} = 0 \tag{12}$$

$$A_2 = 0, y = 0 \rightarrow \frac{2}{2} = 1 \tag{13}$$

...for the Gini calculation of:

$$1 - (0^2 + 1^2) = 0 \tag{14}$$

and the weighted sum of these Gini indexes to create the Gini index for A_2:

$$\sum p_i * g_i = (\frac{3}{5} * 0.44 + \frac{2}{5} * 0) = 0.27 \tag{15}$$

...and finally, following the process for A_3:

$$A_3 = 1, y = 1 \rightarrow \frac{1}{2} = 0.5 \tag{16}$$

$$A_3 = 1, y = 0 \rightarrow \frac{1}{2} = 0.5 \tag{17}$$

...for the Gini calculation of:

$$1 - (0.5^2 + 0.5^2) = 0.5 \tag{18}$$

$$A_3 = 0, y = 1 \rightarrow \frac{1}{3} = 0.33 \tag{19}$$

$$A_3 = 0, y = 0 \rightarrow \frac{2}{3} = 0.67 \tag{20}$$

...for the Gini calculation of:

$$1 - (0.33^2 + 0.67^2) = 0.44 \tag{21}$$

and the weighted sum of these Gini indexes to create the Gini index for A_3:

$$\sum p_i * g_i = (\frac{2}{5} * 0.5 + \frac{3}{5} * 0.44) = 0.27 \tag{22}$$

So the Gini indexes for each input is

Input	Gini index
A_1	0.4
A_2	0.27
A_3	0.27

Both A_3 and A_2 have the same Gini index, thus either can be utilized as the root node of our decision tree. For our purposes, we will choose A_3. From this we repeat the computations above for the positive and negative branches of our A_3 input.

Positive A_3:

Example	A_1	A_2	A_3	Output y
x_2	1	0	1	0
x_4	1	1	1	1

Gini Index of $A_1 = 1$ when $A_3 = 1$:

$$A_1 = 1, y = 1 \rightarrow \frac{1}{2} = 0.5 \tag{23}$$

$$A_1 = 1, y = 0 \rightarrow \frac{1}{2} = 0.5 \tag{24}$$

$$1 - (0.5^2 + 0.5^2) = 0.5 \tag{25}$$

Gini Index of $A_1 = 0$ when $A_3 = 1$ is not applicable, as there is no examples anymore of $A_1 = 0$. Thus our weighted Gini Index is still 0.5.

Gini Index of $A_2 = 1$ when $A_3 = 1$:

$$A_2 = 1, y = 1 \rightarrow \frac{1}{1} = 1.0 \tag{26}$$

$$A_2 = 1, y = 0 \rightarrow \frac{0}{1} = 0.0 \tag{27}$$

$$1 - (1.0^2 + 0.0^2) = 0.0 \tag{28}$$

Ginin Index of $A_2 = 0$ when $A_3 = 1$:

$$A_2 = 1, y = 1 \rightarrow \frac{0}{1} = 0.0 \tag{29}$$

$$A_2 = 1, y = 0 \rightarrow \frac{1}{1} = 1.0 \tag{30}$$

$$1 - (0^2 + 1.0^2) = 0.0 \tag{31}$$

...for a Gini index, when weighted, to still be 0.0 for A_2.

Input	Gini index
A_1	0.5
A_2	0.0

...so we see that we want to use A_2 for the next split as it has the lowest Gini index, and completes this branch of the tree as we have a perfect split.

Negative A_3:

Example	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_3	0	1	0	0
x_5	1	1	0	1

Gini Index of $A_1 = 1$ when $A_3 = 0$:

$$A_1 = 1, y = 1 \rightarrow \frac{1}{2} = 0.5 \tag{32}$$

$$A_1 = 1, y = 0 \rightarrow \frac{1}{2} = 0.5 \tag{33}$$

$$1 - (0.5^2 + 0.5^2) = 0.5 \tag{34}$$

Gini Index of $A_1 = 0$ when $A_3 = 0$:

$$A_1 = 1, y = 1 \rightarrow \frac{0}{1} = 0 \tag{35}$$

$$A_1 = 1, y = 0 \rightarrow \frac{1}{1} = 1.0 \tag{36}$$

$$1 - (1.0^2 + 0.0^2) = 0.0 \tag{37}$$

...for the weighted sum of a Gini index of A_1 when $A_3 = 0$:

$$\sum p_i * g_i = (\frac{2}{3} * 0.5 + \frac{1}{3} * 0) = 0.33 \tag{38}$$

Moving onto the Gini Index of A_2 when $A_3 = 0$, we first explore when $A_2 = 1$:

$$A_2 = 1, y = 1 \rightarrow \frac{1}{2} = 0.5 \tag{39}$$

$$A_2 = 1, y = 0 \rightarrow \frac{1}{2} = 0.5 \tag{40}$$

$$1 - (0.5^2 + 0.5^2) = 0.5 \tag{41}$$

Looking at $A_2 = 0$ when $A_3 = 0$:

$$A_2 = 1, y = 1 \rightarrow \frac{0}{1} = 0.0 \tag{42}$$

$$A_2 = 1, y = 0 \rightarrow \frac{1}{1} = 1.0 \tag{43}$$

$$1 - (1.0^2 + 0.0^2) = 0.0 \tag{44}$$

...and our weight index for A_2 when $A_3 = 0$ is:

$$\sum p_i * g_i = (\frac{2}{3} * 0.5 + \frac{1}{3} * 0) = 0.33 \tag{45}$$

for our collected Gini indexs being:

Input	Gini index
A_1	0.33
A_2	0.33

...thus making it inconclusive which of A_1 or A_2 we could branch off of. We shall choose A_1 for our purposes. One more branch to go!

This next branch doesn't need much in the way of calculations; to save time, we can just observe that when $A_3 = 0$ and $A_1 = 1$, $A_2 = y$. When $A_3 = 0$ and $A_1 = 0$, $y = 0$.

Thus we exhaustively formed the following tree:

b.

Now we utilize Information Gain to create a decision tree for this data. Here we will be calculating the amount of entropy for each branch.

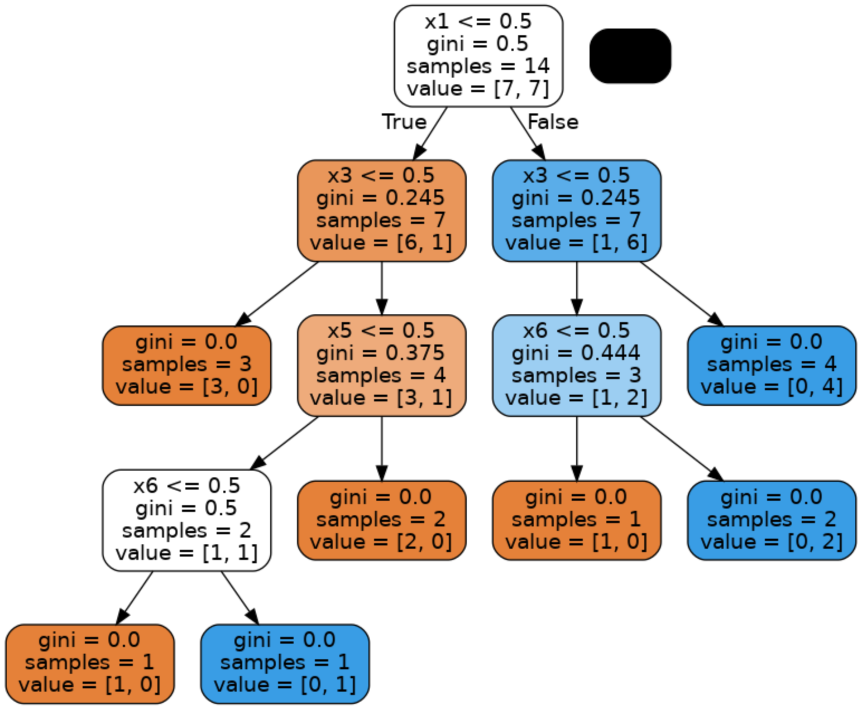
Problem 4

In this section, we consider the following data input with six inputs and a singular target output:

Example	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	$A_1 0$	$A_1 1$	$A_1 2$	$A_1 3$	$A_1 4$
x_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
x_2	0	0	0	1	1	0	0	1	1	0	1	0	1	1
x_3	1	1	1	0	1	0	0	1	1	0	0	0	1	1
x_4	0	1	0	0	1	0	0	1	0	1	1	1	0	1
x_5	0	0	1	1	0	1	1	0	1	1	0	0	1	0
x_6	0	0	0	1	0	1	0	1	1	0	1	1	1	0
T	1	1	1	1	1	1	0	1	0	0	0	0	0	0

When we run our code found in *problem4.py*, we find that we can train both a perceptron and decision tree on this data. If given unseen data of $A_{15} :< 1, 1, 0, 0, 1, 1 >$ with result $T_{15} = 1$, we find that both accurately predict this result.

Whne we generate the decision tree, we can output its resulting branches, seen as follows:



Our code that utilizes the perceptron makes a perceptron makes a perceptron enacting the following linear equation:

$$T = 6x_1 + 0x_2 + 3x_3 - 2x_4 - 4x_5 + 4x_6 - 4 \tag{46}$$