Homework Assignment 3

Kavya Kotian

March 21, 2019

1

2

For each of the following statements, either prove it is true or give a counterexample.

2.1 If P(a|b,c) = P(b|a,c), then P(a|c) = P(b|c)

$P[a \wedge (b \wedge c)] = P(b,c) * P(a|b,c)$ .....(1)
$P[(a \wedge c) \wedge b] = P(a,c) * P(b|a,c)$ .....(2)
$P(b,c) * P(a|b,c) = P(a,c) * P(b|a,c)$ .....from lines (1) and (2)
$P(b,c) = P(a,c)$ $.....asP(a|b,c) = P(b|a,c)$
$P(b \wedge c) = P(a \wedge c)$
$\frac{P(b \wedge c)}{P(c)} = \frac{P(a \wedge c)}{P(c)}$ .....dividing by P(c) on both sides
$P(b|c) = P(a|c)$

Hence Proved

2.2 If P(a|b,c) = P(a), then P(b|c) = P(b)

If P(a|b,c) = P(a), event a is independent of the event b c
We have information about relationship between a and (b,c), and nothing on the relationship between b and c. Hence the relationship P(b|c) = P(b) cannot be proved.

2.3 If P(a|b) = P(a), then P(a|b,c) = P(a|c)

If P(a|b) = P(a), event a is independent of the event b
We have information about relationship between a and b, and nothing on the relationship b given c and a . Hence the relationship P(b|c) = P(b) cannot be proved.

3

Deciding to put probability theory to good use, we encounter a slot machine with three independent wheels, each producing one of the four symbols BAR, BELL, LEMON, or CHERRY with equal probability. The slot machine has the following payout scheme for a bet of 1 coin (where "?" denotes that we don't care what comes up for that wheel):
BAR/BAR/BAR pays 20 coins
BELL/BELL/BELL pays 15 coins
LEMON/LEMON/LEMON pays 5 coins
CHERRY/CHERRY/CHERRY pays 3 coins
CHERRY/CHERRY/? pays 2 coins
CHERRY/?/? pays 1 coin

3.1 Compute the expected "payback" percentage of the machine. In other words, for each coin played, what is the expected coin return?

Probability of all the combinations where money can be earned:

BAR/BAR/BAR pays 20 coins. Payback is Probability that BAR/BAR/BAR occurs multiplied by the payback

$$P[BAR/BAR/BAR] = \frac{1}{64} \qquad ....a$$

$$payback[BAR/BAR/BAR] = 20 * \frac{1}{64} \qquad ....1$$

BELL/BELL/BELL pays 15 coins. Payback is Probability that BELL/BELL/BELL occurs multiplied by the payback

$$P[BELL/BELL/BELL] = \frac{1}{64} \qquad ....b$$

$$payback[BELL/BELL/BELL] = 15 * \frac{1}{64} \qquad ....2$$

LEMON/LEMON/LEMON pays 5 coins.Payback is Probability that LEMON/LEMON/LEMON occurs multiplied by the payback

$$P[LEMON/LEMON/LEMON] = \frac{1}{64} \qquad ....c$$

$$payback[LEMON/LEMON/LEMON] = 5 * \frac{1}{64} \qquad ....3$$

CHERRY/CHERRY/CHERRY pays 3 coins.Payback is Probability that CHERRY/CHERRY/CHERRY occurs multiplied by the payback

$$P[CHERRY/CHERRY/CHERRY] = \frac{1}{64} \qquad ....d$$

$$payback[CHERRY/CHERRY/CHERRY] = 3 * \frac{1}{64} \qquad ....4$$

CHERRY/CHERRY/? pays 2 coins.Payback is Probability that CHERRY/CHERRY/? occurs multiplied by the payback

$$P[CHERRY/CHERRY/?]$$
$$= P[CHERRY/CHERRY/BARorLEMONorBELL]$$
$$= \frac{1}{64} + \frac{1}{64} + \frac{1}{64} = \frac{3}{64} \qquad ....e$$

$payback[BAR/BAR/BAR] = 2 * \frac{3}{64}$ 5

CHERRY/?/? pays 1 coin. Payback is Probability that CHERRY/?/? occurs multiplied by the payback

$P[CHERRY/?/?]$

$= P[CHERRY/BARorLEMONorBELL/BARorLEMONorBELLorCHERRY]$

$= \frac{12}{64}$ ....f

$payback[BAR/BAR/BAR] = 1 * \frac{12}{64}$ 6

<u>Expected Payback</u> = $\frac{61}{64}$ ....adding 1,2,3,4,5

For each coin played, the expected coin return is $\frac{61}{64}$

3.2 Compute the probability that playing the slot machine once will result in a win.

Adding all probabilities where player wins money :

$= \frac{1}{64} + \frac{1}{64} + \frac{1}{64} + \frac{1}{64} + \frac{3}{64} + \frac{12}{64}$

$= \frac{19}{64}$

Hence the probability that playing the slot machine once will result in a win is = $\frac{19}{64}$

4

Consider the following data set comprised of three binary input attributes (A1, A2, and A3) and one binary output:

Example	A1	A2	A3	Output y
$\mathbf{x}_1$	1	0	0	0
$\mathbf{x}_2$	1	0	1	0
$\mathbf{x}_3$	0	1	0	0
$\mathbf{x}_4$	1	1	1	1
$\mathbf{x}_5$	1	1	0	1

Use the algorithm to learn a decision tree for these data. Show the computations made to determine the attribute to split at each node.:

1. **Depth 0 :**

   $$InformationGain(A) = B(\frac{positive}{positive+negative}) - Remainder(A)$$

   <u>Information Gain for A1 :</u>

   $= B(\frac{2}{5}) - [\frac{4}{5}B(\frac{2}{4}) + \frac{1}{5}B(\frac{0}{1})]$

   $= 0.97 - [(\frac{4}{5}) * 1 + (\frac{1}{5}) * 0]$

   $= 0.97 - 0.8$

   $= 0.17$

Information Gain for A2 :
$= B(\frac{2}{5}) - [\frac{2}{5}B(\frac{0}{2}) + \frac{3}{5}B(\frac{2}{3})]$
$= 0.97 - [(\frac{2}{5}) * 0 + (\frac{3}{5}) * 0.910]$
$= 0.97 - 0.55$
$= 0.42$

Information Gain for A3 :
$= B(\frac{2}{5}) - [\frac{3}{5}B(\frac{1}{3}) + \frac{2}{5}B(\frac{1}{2})]$
$= 0.97 - [(\frac{3}{5}) * 0.910 + (\frac{2}{5}) * 1]$
$= 0.97 - 0.946$
$= 0.024$

A2 has the most information gain. Hence we choose A2 as the root node.

2. **Depth 1 :**

Information Gain for A1 :
$= B(\frac{2}{3}) - [\frac{1}{3}B(\frac{0}{1}) + \frac{2}{3}B(\frac{2}{2})]$
$= 0.910 - [(\frac{1}{3}) * 0 + (\frac{2}{3}) * 0]$
$= 0.910 - 0$
$= 0.910$

Information Gain for A3 :
$= B(\frac{2}{3}) - [\frac{2}{3}B(\frac{1}{2}) + \frac{1}{3}B(\frac{1}{1})]$
$= 0.910 - [(\frac{2}{3}) * 1 + (\frac{1}{3}) * 0]$
$= 0.910 - 0.67$
$= 0.24$

A1 has the most information gain. Hence we choose A1 as the node at Depth 1.

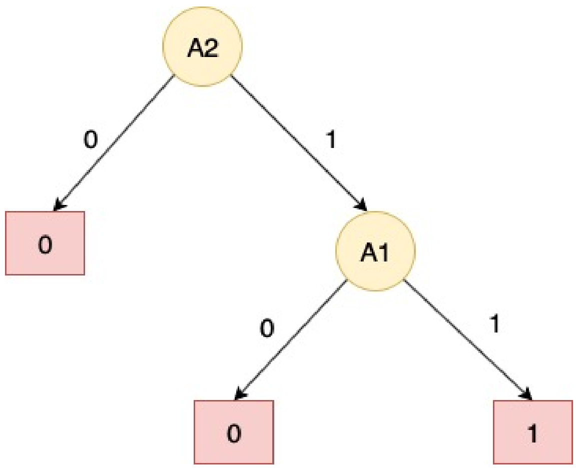

Figure 4.1: Decision-tree

The standard DECISION-TREE-LEARNING algorithm described in the chapter does not handle cases in which some examples have missing attribute values.

5.1 First,we need to find a way to classify such examples,given a decision tree that includes tests on the attributes for which values can be missing. Suppose that an example x has a missing value for attribute A and that the decision tree tests for A at a node that x reaches. One way to handle this case is to pretend that the example has all possible values for the attribute, but to weight each value according to its frequency among all of the examples that reach that node in the decision tree. The classification algorithm should follow all branches at any node for which a value is missing and should multiply the weights along each path. Write a modified classification algorithm for decision trees that has this behavior.

Data: A decision tree with weight(frequency of the attribute value) stored with every branch.(IMPORTANCE FUNCTION in the decision tree example updates the frequency while calculating information gain)
Result: Test examples classified
function
`CLASSIFICATION-IN-MISSING-ATTRIBUTES`(*decisionTree,example,depth0*) **returns** *a list of string*

if *textnormal$attributes$ is empty* **then**
  return path
end
else if $attributeA$ exists **then**
  Check which branch does the value belongs to;
  Go to the next node; `CLASSIFICATION-IN-MISSING-ATTRIBUTES`(*decisionTree, examples – attributeA, depth+1*) **Append the value to the path list;**
  return path
end
else
  choose the branch with argmax(branch weight) Go to the next node;
  `CLASSIFICATION-IN-MISSING-ATTRIBUTES`(*decisionTree, examples – attributeA, depth+1*) **Append the value to the path list;**
  return path
end