# FIS homework assignment 3

Ketan Kokane

March 21, 2019

# 0.1

Depth First Search algorithm can be used to fill in words in the grid. A decent heuristic to cut down the search would be only look for words between minimum and maximum word length permitted by the given puzzle.

*word or character*

It is better to fill in a word at a time compared to filling in character because it would reduce the number of steps required to fill in the puzzle. Also, if we character was used to fill in the word, extra search would be required to find all the words which contain that character in the required order.

Formulation as CSP

**Variables** = Empty boxes of the given puzzle, can be used as the variables for formulating this problem as a Constraint Satisfaction problem

**Domain** = {a-z} 26 alphabets.

**Constraints** = grid which in the path of two words should have the same character.

The word formed by filling up the grid with character should be a correct english word.

*word or character*

If words are chosen as variables for this problem it would simply blow up the domain space, as every variable wont have the same words in it due to length constraint. Even if the word are chosen, still we need to check for the overlap of the two words, which again would involve having constraints with characters. so on atomic level the crossword puzzle seems to be a character based puzzle, thus modelling CSP using characters as variable makes more sense and also be tractable.

# 0.2

## 0.2.1 a

$P(a|b,c) = P(b|a,c)$

$P(a \cap b \cap c)/P(b \cap c) = P(a \cap b \cap c)/P(a \cap c)$

$P(b \cap c) = P(a \cap c)$

$P(c)P(b/c) = P(c)P(a/c)$

$P(b/c) = P(a/c)$

### 0.2.2 b

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

It cannot be proved.

Means that given 'a' independent of 'c' and 'b', thus

$$P(a \cap b \cap c) = P(a)P(b)P(c)$$

Prove that b is independent of c, which is not possible to prove.

Counter Example

let a , b and c be any three events with probability

$P(a) = 0.25$, $P(b) = 0.25$, $P(c) = 0.25$

we know that $\frac{P(a \cap b \cap c)}{P(b \cap c)} = P(a)$

Now lets assume $\frac{P(b \cap c)}{P(c)} = P(b)$

which would mean $P(a \cap b \cap c) = P(a) * P(b) * P(c)$ as $P(b \cap c) = P(b) * P(c)$

but as we can see that $0.25 \neq 0.25 * 0.25 * 0.25$

**the assumption contradicts, hence by contradiction** $\frac{P(b \cap c)}{P(c)} \neq P(b)$

### 0.2.3 c

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

$P(a|b) = P(a)$ therefore $P(a \cap b) = P(b) * P(a)$

$P(a \cap b/c) = P(a/c) * P(b/c)$

$P(a \cap b/c) = P(a/c) * P(b/c)$

$P(a \cap b \cap c)/p(c) = P(a/c) * P(b \cap c)/P(c)$

$P(a \cap b \cap c) = P(a/c) * P(b \cap c)$

$P(b \cap c) * P(a/b,c) = P(a/c) * P(b \cap c)$

$P(a/b,c) = P(a/c)$

Hence proved.

## 0.3

### 0.3.1 pay back percentage

- probability of BAR BAR BAR $= \frac{1}{4} * \frac{1}{4} * \frac{1}{4} = \frac{1}{64}$
- Similarly probability for 3 BELLS CHERRY and LEMON $= \frac{1}{64}$
- probability of CHERRY CHERRY ? $= \frac{1}{4} * \frac{1}{4} * (\frac{1}{4} + \frac{1}{4} + \frac{1}{4}) = \frac{3}{64}$

- probability of CHEERY ?? $= \frac{12}{64}$

  $Expectedpayback = \frac{20}{64} * \frac{15}{64} * \frac{5}{64} * \frac{3}{64} * \frac{6}{64} * \frac{12}{64}$

  $= \frac{61}{64} = 95\%$

### 0.3.2 probability of win

$ProbabilityofWin = \frac{1}{64} * \frac{1}{64} * \frac{1}{64} * \frac{1}{64} * \frac{3}{64} * \frac{12}{64} = \frac{19}{64}$

### 0.3.3 Simulation

Attached the code and read me file which runs the required simulation.

## 0.4

**Solution**:

$$Entropy = \sum A_k log_2(A_k) \tag{1}$$

$E(A_1, positives) = -2/4 * log_2(2/4) - 2/4 * log_2(2/4)$
$E(A_1, negatives) = 0/1 * log_2(0/1) - 1 * log_2(1)$
Information Gain by $A_1 = 1 - 4/5 * E(A_1, positives) - 1/5 * E(A_1, negatives)$

$$= 0.19$$

$E(A_2, positives) = -2/3 * log_2(2/3) - 1/3 * log_2(1/3)$
$E(A_2, negatives) = 0 - 2/2 * log_2(2/2)$
Information Gain by $A_2 = 1 - 3/5 * E(A_2, positives) - 2/5 * E(A_2, negatives)$

$$= 0.44$$

$E(A_3, positives) = -1/2 * log_2(1/2) - 1/2 * log_2(1/2)$
$E(A_3, negatives) = -1/3 * log(1/3) - 2/3 * log_2(2/3)$
Information Gain by $A_3 = 1 - 2/5 * E(A_3, positives) - 3/5 * E(A_3, negatives)$

$$= 0.049$$

**After making A2 as the root node**
$E(A_1, positives) = -2/2 * log_2(2/2) - 0$
$E(A_1, negatives) = 0 - 1 * log_2(1)$

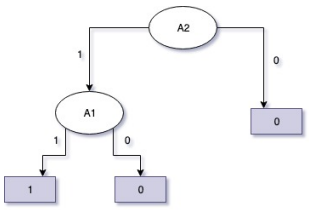


Figure 1: Final Decision tree representation

Information Gain by $A_1 = 0.91 - 2/3 * E(A_1, positives) - 1/3 * E(A_1, negatives)$

$$= 0.91$$

$E(A_3, positives) = -1 * log_2(1) - 0$
$E(A_3, negatives) = -1/2 * log(1/2) - 1/2 * log_2(1/2)$
Information Gain by $A_3 = 0.91 - 1/3 * E(A_3, positives) - 2/5 * E(A_3, negatives)$

$$= 0.25$$

## 0.5 Problem 5

### 0.5.1 1

```
// initialization
ExamplesSeenAt = []
TotalExamplesSeenSoFar = 0
function CLASSIFY(currentNode, examples) returns a label
if currentNode is a leafNode then return Label(currentNode) ;
// if the currentNode is not a leaf node then do the condition test on
 the
ExamplesSeenAt[node]+ = 1
TotalExamplesSeenSoFar+ = 1
requiredAttribute = getAttributeRequiredForTestAt(currentNode)
if examples has requiredAttribute then
nextNode = testExampleOnCurrentNode(exmaple,currentNode)
return CLASSIFY(nextNode,example) ;
else
    // if the required attribute is missing then explore each path of the
     // currentNode and weight the output (labels) generated by them
    nodes = getAllSubsequentNodesOf(currentNode)
    // gets all the child nodes of the given node
    Labels = []
    // initialise a storage to store all the labels generated by exploring
     all the paths and carry forward only the label with maximum
     weight
    foreach node in nodes do
        // calculate labels generated by each path and weight them
        weight = ExamplesSeenAt[node]/TotalExamplesSeenSoFar
          labels.append[weight * CLASSIFY(nextNode,example)]
    end
    return maximum(Labels)
```

### 0.5.2 modifications to information gain

Assumptions: assuming that given 12 examples in the dataset 11 of them have all the required attributes, but the 12th example has the missing at-

tribute, and the solution to the missing attribute is to consider it to be one of the value which is highly repeated in the previous seen exmaple, and use that value in the calculation

$totalexamples = t$

$pos$ be the positive examples

$neg$ be the negative examples

original IG

$$= totalEntropyOfDataset - \frac{pos}{t} * Entropy(pos) - \frac{neg}{t} * Entropy(neg)$$

In the the original information gain we count the example with missing value in the total examples but don't count it in the pos or neg

$examplesWIthMissingAttribute$ be the number of examples which have the required attribute missing

$$maximum(pos, neg) + examplesWIthMissingAttribute$$

by doing this we have used the 'as-if' value of the attribute with higher frequency. and updated the $pos$ or $neg$ based on their frequency in the dataset

the same formula can be used again to calculate the required Information gain but with consideration of the missing attribute

**IG with incremented** $pos$ or $negs$

$$= totalEntropyOfDataset - \frac{pos}{t} * Entropy(pos) - \frac{neg}{t} * Entropy(neg)$$