**Shaikat Islam**
Tufts University
shaikat.islam@tufts.edu

# Homework 01 Part 02

COMP 135 : Fall 2019 : Professor Allen
Due Wednesday, September 25, 2019

## Contents

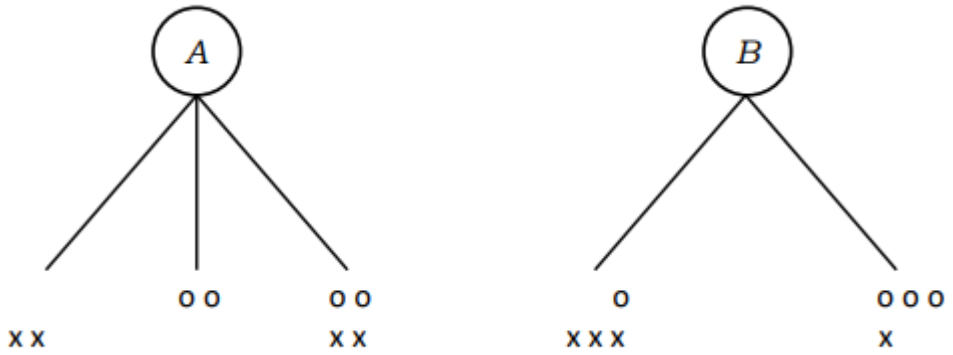
**Problem 1** **2**
(a) . . . . . 2
(b) . . . . . 3

**Problem 2** **4**

**Problem 3** **5**

# Problem 1

In the following diagram, we see a small data-set consisting of two output classes (marked by × and °, respectively), along with the results of splitting the data according to the two properties A (with possible values $a_1$, $a_2$, $a_3$) and B (with possible values $b_1$, $b_2$):




## (a)

For each of the two heuristics employed in the decision tree algorithm (counting and information-based), work out which of the features, A or B, would be chosen by that heuristic. In each case, show all work, including the calculations used to make the determination. You can assume that in the case of ties, each heuristic is used with a random tie-breaking procedure.

Answer

For the purpose of explanation, assume that the set of all positive labels p are represented by °, and the set of all negative labels n are represented by ×.
From an information-theoretic perspective, the entropy of a dataset can be calculated by the following formula: H(Examples) = -(P(Pos)$\log_2 P(Pos) + P(Neg) log_2 P(Neg)$)
Thus, the entropy of the entire dataset can be calculated for the output class A:
P(Pos) = 4/8 = .5
P(Neg) = 4/8 = .5
H(A) = -(P(Pos)$\log_2 P(Pos) + P(Neg) log_2 P(Neg)$)
H(A) = -(.5$\log_2$.5 + .5$log_2$.5)
H(A) = -(-1.0)
H(A) = 1.0
Now that we have the entropy of the output class A, we need to calculate Gain(A) using the following formulas:
Remainder(A) = $\sum_{k=1}^{d} (\frac{p_k+p_n}{p+n}) H(E_k)$
Gain(A) = H(A) - Remainder(A)
Gain(A) = 1.0 - (.25 $\times H(a_1)$ + .25 $\times H(a_2)$ + .5 $\times H(a_3)$)
H($a_1$) = $-(\frac{0}{2} log_2 \frac{0}{2} + \frac{2}{2} log_2 \frac{2}{2}) = 0$
H($a_2$) = $-(\frac{2}{2} log_2 \frac{2}{2} + \frac{0}{2} log_2 \frac{0}{2}) = 0$
H($a_3$) = $-(\frac{1}{2} log_2 .5 + \frac{1}{2} log_2 .5) = 1.0$
Gain(A) = 1.0 - (0 + 0 + .5) = .5
The entropy of output class B can be calculated in the same vein of thought, and since the total P(Pos) and P(Neg) are the same for both classes A and B, the entropy for class B is also 1.0.
Remainder(B) is different however, and must be recalculated in order to get Gain(B):
Gain(B) = H(B) - Remainder(B)

Problem 1 [(a)] continued on next page... 

Gain(B) = 1.0 - ($.5 \times H(b_1) + .5 \times H(a_2)$)
H($b_1$) = $-(\frac{1}{4}log_2\frac{1}{4} + \frac{3}{4}log_2\frac{3}{4}) \approx .8113$
H($b_2$) = $-(\frac{3}{4}log_2\frac{3}{4} + \frac{1}{4}log_2\frac{1}{4}) \approx .8113$
Gain(B) = 1.0 - ($.5 \times .8113 + .5 \times .8113) \approx .1887$
Using this information gain concept to rate the importance of an attribute, it can be said that the information-based heuristic would choose feature A due to it having a larger gain.

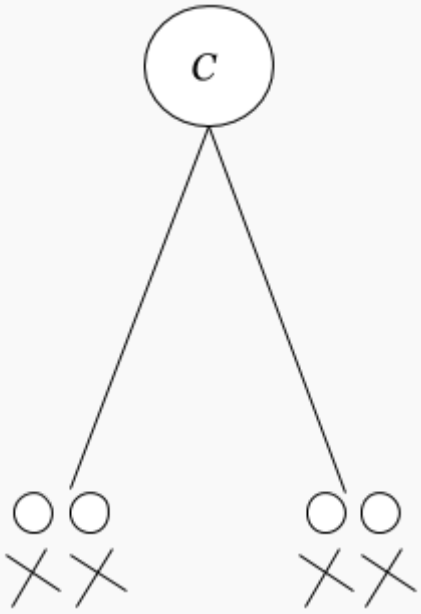
According to Daume, a counting-based heuristic would choose a feature based on a higher ratio of P(Pos) to P(Neg), which is represented by a histogram in Chapter 1 of A Course in Machine Learning. The larger the absolute value of the difference between P(Pos) and P(Neg), the more information that is useful for that value:
For $a_1$ : $P(Pos) = \frac{2}{2}, P(Neg) = \frac{0}{2}, Difference = |P(Pos) - P(Neg)| = 1$
For $a_2$ : $P(Pos) = \frac{0}{2}, P(Neg) = \frac{2}{2}, Difference = |P(Pos) - P(Neg)| = 1$
For $a_2$ : $P(Pos) = \frac{2}{4}, P(Neg) = \frac{2}{4}, Difference = |P(Pos) - P(Neg)| = 0$
Now do the same for B:
For $b_1$ : $P(Pos) = \frac{3}{4}, P(Neg) = \frac{1}{4}, Difference = |P(Pos) - P(Neg)| = .5$
For $b_2$ : $P(Pos) = \frac{1}{4}, P(Neg) = \frac{3}{4}, Difference = |P(Pos) - P(Neg)| = .5$
As a result, it seems that a counting based heuristic would also choose output class A due to it having more values with greater importance than those in class B, which have lower values of importance, as calculated.

## (b)

What conclusions can you draw from these results? That is, what does this mean about the heuristics, and about the tree-building algorithm itself? (Your answer should consist of a few sentences, minimum, of reflection and analysis.)

Answer

As a result, one can assume that both the counting-based and information-theoretic heuristics prefer datasets with values that lean more towards a certain label (either p or n) – there is not much that can be said about the importance of a feature if all of its values lie in the middle. For example, create an output class C like so:



The Gain(C) would equal 0 in an information theoretic heuristic, as would the difference between P(Pos) and P(Neg) be 0 when calculated using a counting-based approach. This output class is not useful, and in such a case of a tie, the choice of heuristic is better off determined by chance.

Thus, in the case of both heuristics, a decision tree algorithm is as useful/important as its dataset.

# Problem 2

In the sample performance shown for the decision-tree algorithm (slides of 11 September, slide 28), we can see that the classification accuracy of the algorithm is not a monotonic function of training set size – that is, sometimes the performance goes down, even as more training data is added. You may or may not have seen similar performance in runs of your own program on the mushroom data-set.
Why does this happen? That is, what could cause the algorithm to actually do worse when supplied with more information. Your answer should be as clear as possible, explaining what features of the algorithm, and what scenarios, exactly, can cause this to occur (for any given heuristic). If you consult other sources in your thinking about this question, remember to cite those sources in your response.

Answer

As a continuation of **Problem 1b**, assume that the entropy of a training set H(X) = 1, the highest possible value. Assume also that X has $x_n$ possible value sets, each with an equal number of positive and negative labels totalling to **n** total labels for the entire training set.
The information gain on this dataset can be calculated as such, as $\mathbf{n} \to \infty$ :

Remainder(X) = $\sum_{k=1}^{d} (\frac{p_k+p_n}{p+n}) H(E_k)$
Gain(X) = H(X) - Remainder(X)
Gain(X) = 1.0 - ((1/n) $\times H(x_n) + (1/n) \times H(x_{n-1}) + ... + (1/n) \times H(x_0)$)
H($x_n$) = $-(\frac{1}{2} log_2 \frac{1}{2} + \frac{1}{2} log_2 \frac{1}{2}) = 1$
H($x_{n-1}$) = $-(\frac{1}{2} log_2 \frac{1}{2} + \frac{1}{2} log_2 \frac{1}{2}) = 1$
...
H($x_0$) = $-(\frac{1}{2} log_2 \frac{1}{2} + \frac{1}{2} log_2 \frac{1}{2}) = 1$
Gain(X) = 1.0 - ((1/n) $\times 1 + (1/n) \times 1 + ... + (1/n) \times 1) = 1.0 - 1.0 = 0$

From the preceding calculation, it is important to note that even though $\mathbf{n} \to \infty$ – that is, the total amount of information available in the training set increases towards infinity, the total information gain for the entire training set X is 0. The training set is not useful, **even though the information available for the training set increases.** This is one such case for an information theoretic approach, in which the total information gain actually decreases as more information is added to the training set; if the information is not **important**, in that as the information **I** in a set increases, the function **Gain(I)** decreases, the algorithm will continue to do worse.

This is also the case for a counting-based heuristic; the difference between P(Pos) and P(Neg) for each value set $x_n$ will always be 0 as P(Pos) equals P(Neg) for each value set, in the case described above. Even if one added an infinite amount of information, the algorithm will continue to do poorly because the information is not very **important**.

One can then say that the decision-tree algorithm is sensitive to the nature and details of its training set; if the training set does not have any useful information, the algorithm will continue to do poorly.

# Problem 3

Given the behavior noted in the previous question, we can say that the decision-tree algorithm is sensitive to the details of its training set. As a result, many modern applications of decision trees use variations on the base algorithm, often generating multiple distinct trees. Explain at least one of these approaches, and how it works exactly. Be sure to highlight ways in which it is meant to improve over the base algorithm. If you consult other sources in your thinking about this question, remember to cite those sources in your response.

Answer

An example of a modern application of a decision tree that uses variations on the base algorithm is the **Random Forest**[a] tree algorithm. It is an algorithm developed by Leo Breiman, and to explain it very concisely, it is an approach that utilizes uncorrelated tree models that operate as a crowd, so as to outperform any individual model.

The Random Forest model has a larger number of single, distinct, uncorrelated decision trees that act as a larger part of one whole. Each single tree relays a prediction for a label, and the label with the most relays becomes the Random Forest's model prediction.

The algorithm, as described by Matthew Bernstein[b], a Graduate Student of Bioinformatics at the University of Wisconsin, is as such: for every singular tree in the ensemble, select a bootstrap sample (a smaller sample taken from a larger sample) $\mathbf{s}$, such that $\mathbf{s}^i$ is the $\mathbf{i}$th bootstrap sample. Then, using that sample, train a decision tree using a modified algorithm, where, instead of determining the importance of all possible feature-splits using a information-theoretic or counting-based heuristic, one chooses a random subset $\mathbf{f}$ of $\mathbf{F}$, where $\mathbf{F}$ is the set of all features. Split the tree on the best feature in $\mathbf{f}$, the random subset. Return the tree, and doing this on every bootstrap sample creates a random forest.

The benefits of this approach lie in in the reduction of variance using the combination of the bootstrapping and random feature selection. Because one chooses to split features based on a random subset, the entire model becomes a lot more effective when tested because the singular trees that make up the random forest of trees are uncorrelated. Using an information-theoretic or counting-based heuristic increases correlation among the individual trees, which makes a forest under-perform in comparison. By having low correlation, the individual trees have a protective feature among all other tree, as individual errors are also no longer correlated, as much. Furthermore, this approach can recognize outliers in data. One disadvantage is that some classifications can be difficult to interpret.

[a] Gupta, Bhumika, Aditya Rawat, Akshay Jain, Arpit Arora, and Naresh Dhami. Analysis of Various Decision Tree Algorithms for Classification in Data Mining. International Journal of Computer Applications 163, no. 8 (2017): 1519. https://doi.org/10.5120/ijca2017913660.

[b] Bernstein, Matthew. "Random Forests" http://pages.cs.wisc.edu/ matthewb/pages/notes/pdf/ensembles/RandomForests.pdf