A Conjunctive Concept Learning Algorithm for Use with Categorical and Binary Datasets Using Patterns Found in Frequencies of Unique Row Occurrences

Kristopher Kurt Honetschlager
Winona State University-Rochester
101 Civic Center Dr NE Apt 207, Rochester, MN 55906
851 30th Ave SE, Rochester, MN 55904
1 (507) 261-9538
kris.honetschlager@protonmail.com

ABSTRACT

In this paper we describe an algorithm for extracting nearly every relationship between every one column and every other column or set of unique columns in a given dataset in a relatively easily understandable manner.

Technically, we describe an algorithm which searches iterating selections of one fixed column and another column or set of other columns, whose number is defined by the user, for predefined patterns when they're taken as a collective in the frequencies of unique row occurrences, and then output to the user, in relatively easily read and understandable terms, conditional statements or heuristic statements conveying those patterns observed like, "If X is True and Y is True (where X and Y are constituent elements or variables of the hypothesis), then Z is true (where Z is the conclusion).".

The motivation, behind the development of this algorithm, comes from a background in machine learning where it's seeming to become crucial that the reasoning behind the output of an estimator be relatively easily explainable to any person interacting with it.

The methods developed for testing this algorithm were rudimentary; we thought that if the algorithm was working properly, then giving it a fixed dataset with prearranged relationships between the features would cause it to pick up on the patterns and output those relationships.

Thorough testing has not yet been done, for x vector of size 1 however in each demonstration the algorithm has worked flawlessly.

In the future we intend to thoroughly test this algorithm and put out a proper publication if it seems worthy of it.

The algorithm has at least partially theoretically and empirically shown to be viable, further intensive testing will be required to make anything of it.

Categories and Subject Descriptors

D.3.3 [**Python**]: Language Constructs and Features – *control structures, packages*

General Terms

Your general terms must be any of the following 16 designated terms: Algorithms, Measurement, Documentation, Design, Reliability, Human Factors, Theory.

Keywords

hidden, data, conjunctive, concept learning, algorithm, algorithms, dataset, datasets, categorical, binary, value counts, unique row occurrences, heuristic, logic, heuristics, conditional statement, python, pandas, jupyter notebook

1. INTRODUCTION

The algorithm that we describe in this paper takes a dataset that's either categorical or binary, binarizing it in the case of it being categorical, and iteratively selects one column to be the label and a user defined x unique number of other columns, which are also iteratively selected and cannot include the label, to be the x-vector, effectively making a sub-dataset and takes the derivative frequency of unique row occurrences from this sub-dataset to make a new "value counts" dataset, perhaps also to be referred to as "row counts" dataset.

The row counts dataset includes all possible combinations of binary values of n size where n is the number of columns in the row counts dataset, not including the "count of unique row occurrences" column.

Because the original dataset has been binarized, all rows are now effectively binary; this is why we use all possible combinations of binary values of n size to populate the rows of the row counts dataset and add in the frequency of unique occurrences of the rows from the sub-dataset collected.

We get all possible outcomes of that size of binary dataset and can then see which occur most-frequently. Specifically, there are patterns to binary row counts datasets when a logical rule or conditional statement is occurring for the columns and label selected.

From the observation of those patterns, we can then give the user back the patterns observed in an easily understandable manner.

2. Motivation

For example, perhaps a domain expert is skeptical of the conclusion or conclusions an estimator came up with and is curious as to its reasoning in a way which wouldn't require them to have a graduate degree in machine learning. The implementation of logic as a form of conveying seems to be the proper solution for this problem.

In our attempting to talk about this estimator, we found ourselves speaking in terms of logic. It seems natural that this be the way that things are, and to tailor all other algorithms in this direction. The estimator is meant to be used by humans, why should it not have a read out of its conclusions that's comprehendible and holds to scrutiny through logic?

3. Innerworkings of The Algorithm

3.1 Binary Datasets

Fundamentally the method we generated works with binary datasets. So far, the latest implementation also works with categorical datasets by dynamically breaking them down into binary datasets. The reason binary datasets are used is because binary datasets are more generalizable than using the original categorical dataset, in the case that a categorical dataset is used. If a categorical dataset were used with this method, we're not sure that it would be as easily generalizable as binary datasets because premade methods exist to encode categorical datasets as binary datasets.

3.2 ITERATING THROUGH LABELS

One aspect of my implementation is that we iterate through each possible label in the dataset. The purpose for doing so is to try and find each of the best possible rules/heuristics for predicting the label of any given instance in the dataset.[1]

3.2.1 Pairing with All Combinations of Columns Not Including the Label

In addition to iterating through each possible label, we test every possible n combination of columns, where n is some integer less than the number of columns in the dataset minus one. The purpose for doing this is to sort the rules with the highest accuracy in prediction for each label and to print those rules back to the user, extending what we stated in point 3.4.

3.2.2 Accuracy Score

Accuracy score here is defined as the number of instances that the rule correctly identifies for the label over the total number of instances in the dataset.

3.2.3 Sorting Rules

We use a prebuilt method for sorting all rules in descending order by their accuracy score.

3.3 UPSAMPLING THE LABEL

We thought it would be necessary that when selecting a label for rule generation that the label be up sampled for the purpose of accuracy. We're not quite sure how best to explain the reasoning behind my thinking this should be the case at the moment.

Proceedings of the 18th Winona Computer Science Undergraduate Research Seminar, April 23 & 25, 2018, Winona, MN, US.

3.4 VALUE COUNTS OF BINARY DATASETS

We had intuited at some point during this research that something like value counts would be very useful, though we didn't know that value counts existed at the time. They turn out to be extremely useful for diagnosing the type of rule that is being observed. Binary columns work very well for making traditional logic statements like "If p is true, then q is true". Truth or falsehood being denoted by 1 or 0 respectively.

3.4.1 Characteristics of Value Counts of Binary Datasets

Table 1. 2 Column Value Counts Dataset

X	**Y**	**Number of Occurrences**
1	1	0
1	0	0
0	1	0
0	0	0

A value count dataset is like the one above for 2 columns of the original dataset where y is the label and X is one of the other columns that isn't the label. There are other ways to construct a value counts dataset, but this is the form we'll be referring to.

One characteristic of a value count dataset is that the number of instances is equal to two to the number of columns in the original dataset.

When indexing the instances of the value counts dataset from zero, the label for each instance whose indexes modulus with 2 is equal to 0, the label's classification of that instance is positive or 1. For instances whose label classification is negative or 0, all the indexes are odd numbers, or numbers whose modulus with 2 is equal to 1.

The logical inverse of any statement with an index less than half the number of instances in the value counts dataset is two to the number of instances in the value counts dataset minus one minus the index of the statement.

Example: If there are 2 columns in the value counts dataset and we're looking for the inverse conditional statement of the instance located at index 0 (1, 1), which is less than 22/2, in the value counts dataset; the inverse instance would be located at index (22 - 1) – 0 = 3 (0, 0).

We've begun to refer to the instances as statements because you can think of them in that way as we explained before.

3.4.2 Positive Heuristic Generation

Positive instances are instances from the value counts dataset where the label is equal to 1.

If you were to take any given positive instance in the value counts dataset and divide that number by the sum of all of the number of occurrences of all of the positive instances in the value counts dataset, the number you would retrieve greater than 0.5; then that instance could be the heuristic for predicting if a given instance in the original dataset is positive.

Additionally, if you were to divide the number you retrieved by the sum all of the number of occurrences of all of the instances in the value counts dataset and retrieve a number greater than 0.5; then the given instance you initially chose is at least a likely heuristic for testing if any given instance in the original dataset is positive.

Table 2. 2 Column Value Counts Dataset Positive Instance

X	Y	Number of Occurrences
1	1	100
1	0	0
0	1	0
0	0	0

In a case like the one above, among the positive instances and among all instances the instance at index 0 occurs most frequently, both cases being respectively 100% frequency among the occurrences.

If you were to attempt to generate a heuristic for predicting when the label is positive for a given instance in the original dataset, a good heuristic would be if the column X in the instance is equal to 1. Because the value counts dataset shows that for all instances in the original dataset, whenever x is equal to 1, y is also equal to 1.

Table 3. 2 Column Value Counts Dataset Positive Instance Denial

X	Y	Number of Occurrences
1	1	50
1	0	50
0	1	0
0	0	0

If there were a case like the one above, there wouldn't be a good heuristic for when y is equal to one for a given instance in the original dataset. This is because although among the positive instances one instance is overrepresented, among all instances both the instance at indices 0 and 1 occur most frequently at an equal number of occurrences.

The rule at index 1 is the logical denial of the rule at index 0. This is the reason why a heuristic to predict if a given instance is positive using the value count dataset aforementioned would not work.

This same phenomenon would occur if indices 0 and 2 had an equal number of occurrences in the value counts dataset.

3.4.3 Negative Heuristic Generation

Negative heuristic generation is the opposite of positive heuristic generation. Negative instances are defined as those instances where the label is equal to 0.

Table 4. 2 Column Value Counts Dataset Negative Instance

X	Y	Number of Occurrences
1	1	0
1	0	50
0	1	0
0	0	0

Table 5. 2 Column Value Counts Dataset Negative Instance Denial

X	Y	Number of Occurrences
1	1	50
1	0	50
0	1	0
0	0	0

3.4.4 Combinatorial Heuristic Generation

Combinatorial heuristic generation is specifically when an instance and its inverse occur at an equal number of occurrences in the value counts dataset and at a respective number of occurrences over all numbers of occurrences in the value counts dataset of greater than 25%.

Table 6. 2 Column Value Counts Dataset Combinatorial

X	Y	Number of Occurrences
1	1	50
1	0	0
0	1	0
0	0	50

In the example above, the 0 and 3 indices occur the same number of times and combined occur greater than 50% of the time in the dataset. Logically a conditional statement does not imply it's inverse, unless it can be shown empirically to be true. This is that case.

Table 7. 2 Column Value Counts Dataset Combinatorial Dataset

X	Y	Number of Occurrences
1	1	50
1	0	50
0	1	50
0	0	50

The above is a denial dataset, it tells us that we cannot make a combinatorial rule.

4. Related Work
Unlike other presentations, on the day of presenting my research, and even now we have not reviewed any scholarly material related to the work that we've been doing. Of what we've seen, it seems we're operating in the field of machine learning and particularly conjunctive concept learning. We're not sure if my particular implementation is novel.

5. Analysis and Results
Only from those demonstrations we've given do we have any results, they're rather narrow in scope. The algorithm appears to work flawlessly for X-vector sizes of 1.

6. Discussion
Some talks were had between my independent study advisor and other computer science faculty about the validity of the brute force aspect of the method, perhaps there are better ways to select columns than that particular way. False negatives and false positives might be something able to be collected, disjunctive rules, etc.

7. Conclusion
The algorithm has at least partially theoretically shown to be viable, further intensive testing will be required to make anything of it.

8. ACKNOWLEDGMENTS
Our thanks to ACM SIGCHI for allowing us to modify templates they had developed. Thank you to my independent study advisor Collin Engstrom, Winona State University, and particularly the WSU Computer Science department.

9. REFERENCES
[1] ACM SIG PROCEEDINGS template. http://www.acm.org/sigs/pubs/proceed/template.html.