Implementation of the FP-Growth Algorithm

Raiyan Abdul Baten
University of Rochester
Rochester, NY
rbaten@ur.rochester.edu

ABSTRACT

This document explains my approach for mining frequent itemsets from the adult dataset. In particular, the report sheds light on data preparation, python implementation and result analysis of the FP-Growth algorithm.

ACM Reference Format:
Raiyan Abdul Baten. 2018. Implementation of the FP-Growth Algorithm. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

FP-Growth is a popular algorithm for mining frequent itemsets from transaction databases. In this project, I have implemented the algorithm as specified in Chapter 6 of Han et al.'s book [1].

2 DATASET

I have used the adult dataset[1] in this project.

2.1 Dataset Summary

The dataset was originally intended for classification tasks, and therefore comes pre-split into training and test sets, in files named 'adult.data' and 'adult.test' respectively. The training set contains $N_{\text{training}} = 32561$ data tuples, and the test set contains $N_{\text{test}} = 16281$ tuples. In this project, I have analyzed the two sets separately to obtain the results.

Both of the dataset files contain the same 14 attributes and 1 class label: age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country, and the class label of yearly income. Since this is a frequent itemset mining task rather than classification, I have treated all of the attributes and class labels as 'items', making all 15 of the information entries conceptually equivalent.

2.2 Pre-processing

A quick exploration reveals that the attributes 'education' and 'education-num' have one-to-one mapping. For example, 'Bachelors' in 'education' is coded with '13' in 'education-num', 'Masters' is coded with '14', so on and so forth. Hence, the attribute 'education-num' is just a proxy for 'education', and therefore I removed the 'education-num' attribute altogether from my analyses.

The attributes 'workclass', 'occupation' and 'native-country' have blank entries in them, labeled with '?' marks. I decided to treat the '?' entries as categorical values of the respective attributes, since all of the entries are eventually treated as 'items' anyway. To differentiate between the '?' entries of one attribute from those of others, I concatenated all of the entries of these three attributes with their respective attribute names. Therefore, the workclass 'Private' became 'Private_workclass', the entry '?' became '?_workclass', so on and so forth for occupation and native-country as well.

The attributes 'age', 'fnlwgt', 'capital-gain', 'capital-loss' and 'hours-per-week' are numeric. I split each of them into 10 equal-width bins, and took the bin boundary information as categorical values. I further concatenated these categorical names with the attributes names for clarity. Therefore, the age '39' became 'age_(38.9,46.2]', holding the attribute and binning information.

Finally, I converted the dataset into a dictionary, with keys corresponding to 'transaction ID's, and values corresponding to the lists of 'items' (holding attribute values). Therefore, every tuple in the original dataset became a (processed) list of items. This dictionary was fed into the code as the input database.

3 IMPLEMENTATION

I implemented the algorithm presented in Figure 6.9 of Han et al.'s textbook [1]. I chose a support of 23% .

Data structures: I have used a list of dictionaries, 'frequent_patterns', to hold all the generated frequent itemsets. Each dictionary in the list holds an itemset and its support. I have further used dictionaries to hold the tree data structures: the original FP-tree and the subsequent conditional FP-trees. These 'tree' dictionaries have unique IDs, given to each node, as keys; and dictionaries of node attributes as values. These attributes include the item represented by the node, its count, and the ID of its parent node. The null node is assigned an ID of 0. For example, an entry in the FP-tree dictionary can be $1 : \{``item" : ``HS-grad", ``count" : 4, ``parent" : 0\}$, which would mean that this 'HS-grad' node is connected to the root of the tree (since parent ID is 0), has a count of 4, and any entry having a parent of 1 will be its child (since its key is 1).

4 RESULTS

The 'adult.data' and 'adult.test' datasets gave 581 and 587 frequent itemsets respectively.

REFERENCES

[1] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques.* Elsevier.

[1] http://archive.ics.uci.edu/ml/datasets/Adult