

Efficient Search for Association Rules

Authors :

Geoffrey I. Webb

Deakin University



Contents

- Introduction to the problem
- Key Contribution
- Relevant Prior Work
- Methodology
- Results
- Opinion of the paper



Problem

When apriori imposes computational burden ?

Apriori can impose large computational overheads when the number of frequent item set is very large.

- Cases when apriori performed on domains other than market basket
- Cases when market basket information is augmented by the customer information

Basic Problems

- Maintenance and manipulation of the item set impose computational burden.
- Large amount of time required to complete the computation.



Key Contribution

- New alternative approach in “Association Rule discovery” using direct approach rather than two stage approach of apriori.
- More efficient association rule discovery when all the data retained in the memory and number of item set can not be adequately constrained by considering individual item set in isolation.
- Use of inter-association-rule constraint to find the association rule effectively.



Related work

Approches :

➤ Machine Learning

It's an early approach to identify an interesting data dominated by attempts to form small set of rules to classify unseen data.

➤ Knowledge based Discovery

Identifying large numbers of the classification rules distinguished by the removal of the objective of using rules for classification and hence the requirement that a small number of rules be identified.

➤ Association rule Discovery

It's been motivated by finding rules that predict increased frequency of an attribute value without limitations on the values that may appear in the consequent of a rule.



New algorithm: OPUS

What is OPUS ?

- Optimum Search for Unsorted Search spaces is the approach to bring two divergent branches of the rule discovery research back together.
- OPUS is distinguished by its ability to efficiently find the pre specified number of rules that maximize an arbitrary function measuring rule quality.



OPUS Strategy

- Any condition to be pruned at a node must precede all conditions not to be pruned.
- OPUS guarantees that every pruning approximately halves the remaining search spaces.



Understanding some terms

Inclusive Pruning: Exclusion from the search space all those nodes that do not contain given condition.

Exclusive Pruning: Exclusion from the search space all those nodes containing a particular condition.

Coverage: The proportion of examples covered by the LHS.

Support: The proportion of examples jointly covered by the LHS and RHS.

Strength: The proportion of examples covered by the LHS that are also covered by the RHS.

Lift: The strength divided by the proportion of all examples that are covered by the RHS.



OPUS : search Algorithm

OPUS_AR(CurrentLHS, AvailableLHS, AvailableRHS)

1. So Far := {}
2. FOR EACH P in AvailableLHS
 - (a) NewLHS := AvailableLHS – P
 - (b) AvailableLHS := AvailableLHS – P
 - (c) IF cover(NewLHS) < minLHScover. THEN
 - i. NewAvailableRHS = AvailableRHS
 - ii. FOR EACH Q in AvailableRHS
 - A. IF Strength > minStrength and able to fit into m best rules THEN
Record NewLHS → Q
 - B. IF any RHS condition Q for which
 1. cover(NewLHS U {Q}) < minRHScover OR
 2. opt_strength < min_strength OR
 3. opt_lift < min_lift THEN
NewAvailableRHS := NewAvailableRHS – Q
 - iii. If NewAvailableRHS != {} THEN
OPUS_AR(NewLHS, SoFar, NewAvailableRHS)
 - iv. SoFar := SoFar U {P}

Example

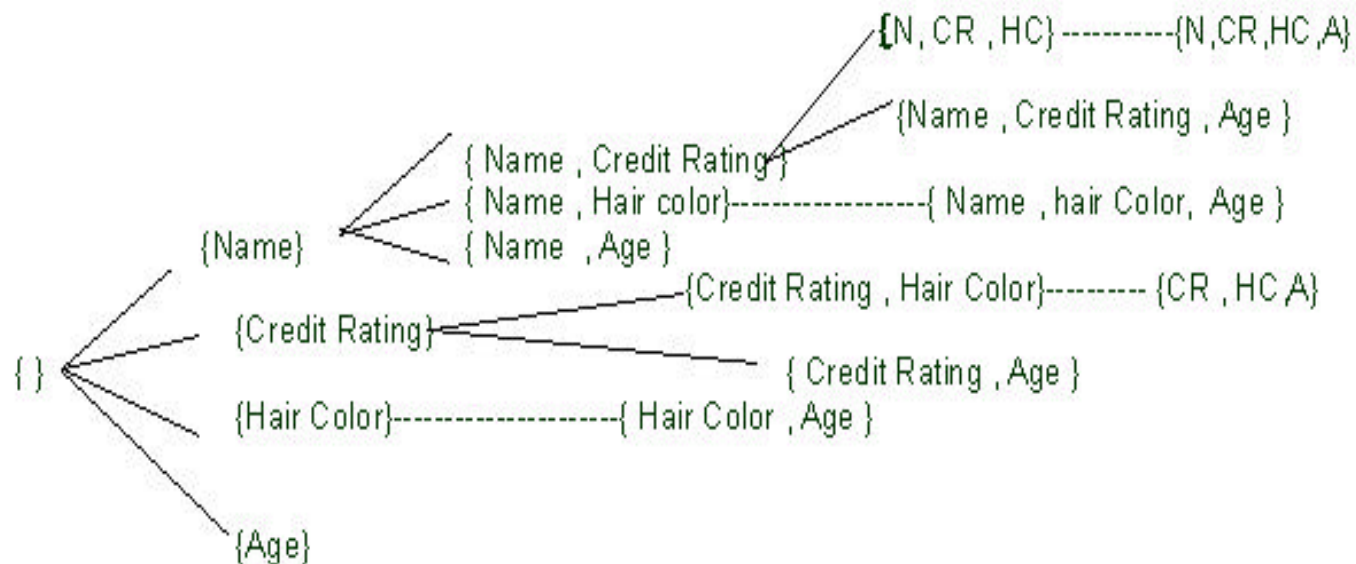


Fig. 1 : A Fixed Structure Search Space

Example

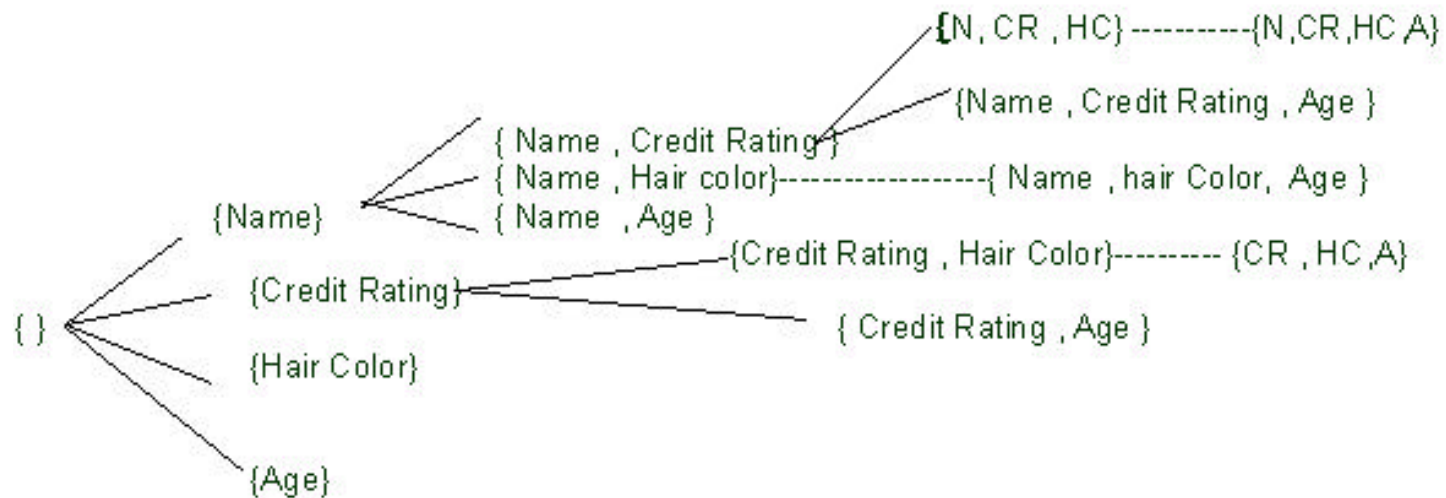
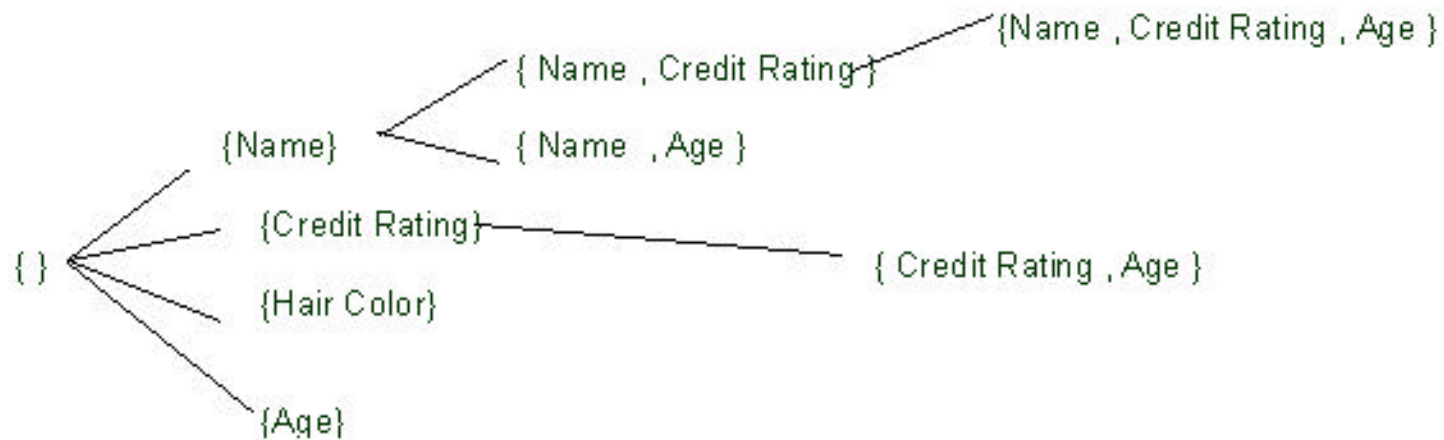


Fig .:2 Pruning a branch from Fixed Structure Search Space

Example



**Fig.:3 Pruning All Nodes Containing a single operator from
Fixed Structure Search Space**

Example

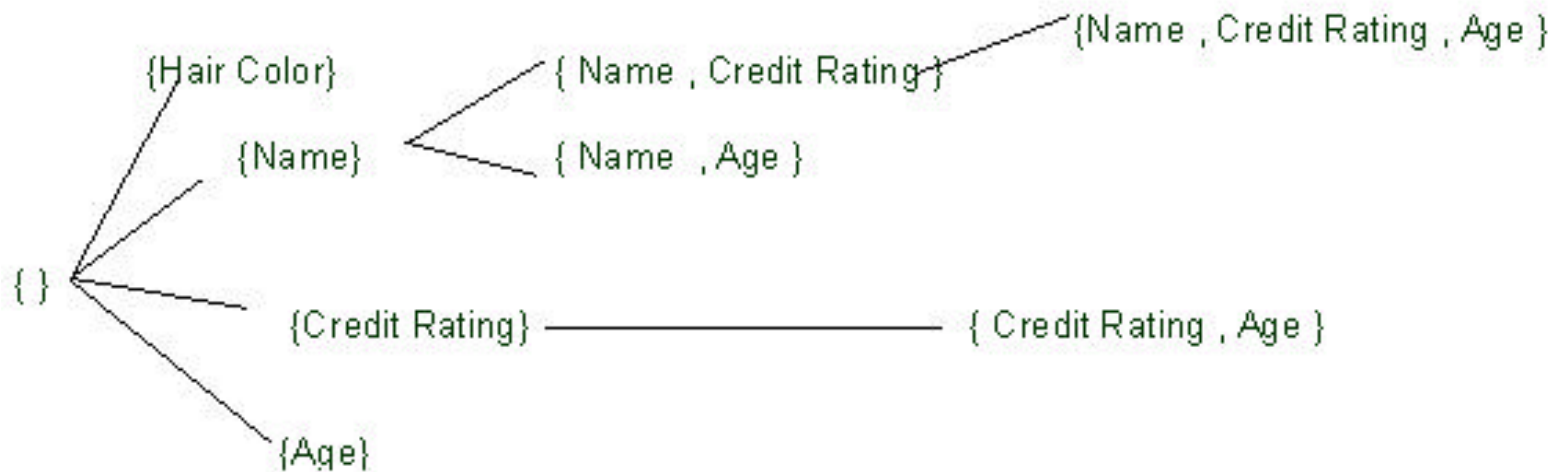


Fig.:4 Pruning with restructured search space



+ & -

Key Advantage :

Efficient in Association rule discovery particularly when all the data retained in the memory.

Main Disadvantage:

This approach has a potential disadvantage that it requires many more passes through the data than A priori. Where the data can be maintained in the main memory this need not be the serious handicap.



Results

For Cover Type Data set having 581,012 cases described by 55 attributes. The ten continuous valued attribute were discretized in to three sub ranges and remaining were all binary. Overall 120 attribute-values.

Apriori:

Requires generation and analysis of 14,567,892 itemset. Even when the item set size is restricted to five.

Total time required : 96 CPU hours.

OPUS:

Constraint: Find the 1000 association rule with the highest values of lift required evaluation of only 384,312 rules and 84,649 distinct antecedents.

Total time required: 50 CPU minutes.



Opinion on Paper

In our opinion this paper impressively explains the drawback with a priori and provides the feasible solution of the computational burden incurred in two stage search of associations using a priori. We will rank this paper 7 on a scale of 1-to-10. The only point that touches us is the big difference in computational efficiency at a cost of few gigabytes of memory. However OPUS_AR algorithm is much more complex than very straight forward a priori.

The example given here is not much more helpful to us in understanding how algorithm works actually and some of the terms used are vague in meaning.

We are also impressed by the wider application of OPUS_AR algorithm in commercial association rule discovery system called “*Magnum Opus*”.

Over all good paper to present we can gain a lot out of it.



Conclusion

- Paper presents the algorithm for association rule discovery analysis based on the efficient OPUS algorithm. This approach is distinguished from the widely utilized Apriori algorithm by its ability to use inter relation between the item sets to constrain the number of the item set that are considered.
- It is also distinguished from the number of other algorithms, that have presented as the alternatives of the Apriori algorithm, by exploring associations containing all available conditions as consequents.