1. Grab It

Soham Roy
Martikulation Number: 7028704
soro00002@stud.uni-saarland.de

1 Overview

All three papers, Smets and Vreeken 2012 called as Slim, Vreeken, Van Leeuwen, and Siebes 2011 called as Krimp, and Fischer and Vreeken 2019 called as Grab, have the primary goal of developing and presenting efficient and effective methods for mining patterns and association rules, from data. However, each paper focuses on different aspects of this overall goal.

Krimp introduces an algorithm to generate a minimal set of high-quality codes and patterns from a given dataset. It primarily focuses on creating a compressed representation of the original dataset by leveraging the Minimum Description Length (MDL) principle. Krimp aims to find a code table that most efficiently compresses the dataset, which is thought to best capture the underlying regularities in the data. Slim introduces a method that, like Krimp, leverages the MDL principle and applies it in a similar way. It views the problem of pattern selection as a problem of model selection, focusing on finding the smallest set of patterns (code table) that describes the data best. However their is a subtle difference in the algorithms of how krimp and slim go about achieving their objectives, thereby reducing the issue of "pattern explosion", i.e., finding an unmanageably large number of patterns in the data.

Grab proposes a novel method for finding association rules in data using the MDL principle, with a unique focus on dealing with noisy data. Grab introduces the concept of an error matrix to handle noise in the data explicitly. This results in a more robust pattern selection in the presence of noise. Grab aims to find a small, non-redundant set of rules that together describe the data well, providing a balance between capturing the main trends in the data and overfitting to the noise.

In summary, while all three papers have the primary goal of mining high-quality association rules from data, they approach the problem from slightly different angles: Slim and Krimp focuses on compression and pattern selection through compression, and Grab focuses on finding a set of association rules of the kind X–>Y which is robust in the presence of noise.

2 Analysis

2.1 General procedure of the Algorithms

At a broad level, all three algorithms follow a similar procedure which is:

1. Start with a rule or pattern set consisting only of singletons.
2. Incrementally add the most effective candidates to the set in each iteration.
3. Perform pruning and retain the set which achieves best compression.
4. Repeat the above steps until the algorithm reaches convergence.

2.2 Comparative Analysis of Algorithms

Though the input to Krimp and Slim is the same, the way they tackle the problem is different. Slim finds patterns within the data that describe the data well. It mines the patterns by iteratively

Preprint. Under review.

combining the existing patterns in the code table and checking if the combination explains the data better. However, it does not blindly try all combinations, but makes a smart choice of selecting a combination based on usage, thus making an optimistic estimator. On the other hand, Krimp chooses its candidates from a candidate generation program, which spits out a list of all the possible candidates that could be helpful. This candidate generation program is based upon support (minsup), thus also needs to calculate the usage. Calculating usage of the entire dataset for all the codes is difficult because of the immense size and complexity of the potential search space. Given that the number of possible itemsets grows exponentially with the number of items, attempting to compute and keep track of usage for each one would be computationally prohibitive and inefficient, potentially hindering the effectiveness of the Krimp algorithm.Grab bears a resemblance to Krimp in that it cultivates its own rule set from the initial collection, incrementally building upon it. As such, it does not depend on an external process for generating candidates.

2.3 Issues with Candidate Generation

Generating candidates based on minimum support and usage parameters can present a few disadvantages. First, it can generate an exceedingly large number of potential patterns, especially in cases where the data set is large and dense. This could be computationally expensive and could lead to a significant amount of time spent on calculating the usage of all possible patterns across the entire data set, which may not be feasible for larger datasets. Additionally, the quality of the candidates generated is tied to the threshold set for the minimum support. If the threshold is set too high, potentially interesting patterns may be overlooked. On the other hand, a low threshold may result in the generation of too many trivial or uninteresting patterns. Furthermore, by tying the generation of candidates to such parameters, we risk missing patterns that might be meaningful but do not meet the set threshold values. The parameter selection is also data-dependent, adding an extra layer of complexity to the process. Lastly, this approach inherently assumes that patterns with high support are more interesting or useful, which is not necessarily always the case.This brings up another issue: there's a lack of a feedback mechanism in the candidate selection process. The F set selected by minimum support doesn't provide guarantees on the quality or relevance of the selected candidates.

2.4 Neglect of Frequency Information

A potential limitation of these algorithms is their lack of attention to frequency information when establishing the code table or the rule sets. For any $t \in D$ its encoded length, in bits, denoted by $L(t|\text{CT})$, is summation of code lengths of X which covers t , but suppose for a code of ABCABC , then ABC covers it twice, which is not captured with the assumption that each code covers a "t" at most once! So for the code ABCABC , the algorithm will only consider ABC (unique items only) and disregard frequency information.

Let's consider an everyday shopping scenario for clarity. Generally, when a customer purchases a pack of two burger patties, they're likely to also buy a pair of hamburger buns (Y). Burger patties often come in packs of two or four, matching the number of buns needed to prepare complete burgers. Consequently, buying a pack of burger patties usually correlates with purchasing hamburger buns. However, if only one burger patty is bought, the customer might not buy hamburger buns, as the patty might be intended for a different recipe. The current algorithms are unable to incorporate this kind of frequency-related information.

If we had altered the main algorithm , then another aspect that needs consideration is how the Cover function processes the data. The Cover function does not count the frequency. or instance, in the case of 'ABCABC', it will be treated as a new code of length 6, regardless of the presence of 'ABC' or the singletons already in the code table. only singletons wont work here as it would violate the cover constraint that the disjoint union would recover the t. The frequency information would allow for better compression and more efficient representation of the patterns in data.

2.5 Unpacking the Rule Set

The 'rule set' yielded by Grab can be understood as an enriched version of the code tables produced by algorithms like Krimp and Slim. While these earlier methods identify a combination of patterns that compresses the data into a single 'sack', they do not offer any insight into the causal relationships within that sack. Grab, on the other hand, provides this and more. It not only condenses the data, but

also elucidates causal relationships within the pattern sets, described as rules. However the causal relationship it describes should not be confused with the causal direction of the features. It has the power to ascertain that the presence of A also causes the presence of B , and not that the event A causes event B. A simple example would be swimmingvests and sunscream, the algorithm can deduce that the presence of swimmingvests also causes the presence of sunscream in the same transaction but it does not imply any direction of causation and in this case this is a confounding scenario with summer being the confounder. For instance, where Slim and Krimp might simply produce 'AB' in their code table, Grab would deliver 'A→B', providing an additional layer of interpretability. This means that while we could interpret Grab's result in the context of Slim or Krimp as 'AB', Grab's approach also provides something more.

2.6 Effects of Singleton Volume on Candidates

The number of candidates ih F for Grab and Slim is directly dependent on the number of singletons or features in the dataset. In circumstances where a dataset contains numerous singletons, the initial pool of candidates for both Slim and Grab would be considerably large. However, in Krimp, the variable 'minsup' can be used to control this. Thus, in datasets with a high number of singletons, Krimp might offer an added advantage by preventing a combinatorial explosion of candidate generation. However it has been claimed that Grab scales very well to thousands of features . Nevertheless, the evaluations performed do not encompass datasets of this type. Consequently, Krimp's ability to manage this potential drawback stands as an untested, albeit logical, benefit.

2.7 Optimization of Pruning

The author considers a greedy approach to pruning the code/rule sets to get rid of the redundancy caused by adding of new codes/rules. The author checks on subsets of the already existing table and calculates the true compression value on the considered subset, however this can further be optimized , After including a rule or itemset into the code table and then considering all the subsets for pruning, arranged by usage, the author could just consider the subsets that include items from the newest addition only. So if the addition was ABC, then they should just take subsets that have A, B or C. This is because this is just a relative comparison to check if the new addition of ABC is better than the existing ones having A , B or C and we don't need to know the actual compressed value, so this should be enough. In this scenario, a code with DEF won't be pruned, so we don't need to think about it from the start.

2.8 Potential for Anti-Correlation Exploration

An aspect that could potentially enhance the algorithm is to delve into the territory of anti-correlation. This challenges our conventional understanding of patterns as being entities that exist in the data, suggesting that we could also consider the absence of certain associations as patterns. For instance, we could perceive '$\neg A \rightarrow \neg B$' or '$A \rightarrow \neg B$' as a rule or pattern defining the data.To illustrate, if we have transactions ACD and BCD, this could be succinctly explained using one of a pattern like '$A \rightarrow \neg B$' by Grab or '$A\,\neg B$' by Slim/Krimp. Such an approach could potentially compress code tables even further and require fewer rules than the current algorithm. However, this comes with increased complexity and necessitates careful consideration of numerous edge cases, but it certainly opens up an exciting avenue for further exploration.

3 Considering overlapping

Overlapping patterns in data present a unique challenge for data mining, and the three algorithms - Krimp, Slim, and Grab - address this issue in their own distinctive ways. Krimp, utilizing a greedy approach to pattern discovery, doesn't directly handle overlapping patterns. Its method of single candidate pattern selection based on compression size minimization can inadvertently result in redundancies or missed information in the presence of overlaps. Slim, on the other hand, provides a more flexible approach. By iteratively combining itemsets within the code table, it evaluates the contribution of overlapping patterns to database compression. This procedure allows SLIM to better incorporate such overlaps if they help in reducing the compressed database size. Grab offers another effective solution by permitting multiple items in the consequent of a rule, thus providing

a more nuanced representation of overlapping structures. For instance, it captures dependencies between items in overlapping patterns more effectively. To sum up, although Krimp, Slim, and Grab each tackle overlapping patterns uniquely, the iterative flexibility of SLIM and the allowance for consequent complexity in GRAB equip them with better capabilities for managing these patterns.

4 Results

The study shows that the algorithm does a great job in a bunch of areas, like working fast and efficiently, packing data tightly, being able to grow with bigger tasks and sorting things into categories . This makes it a really good tool for data tasks that need to do things well and quickly, and get high-quality results. The experiments on data about Mammals and on the ICDM Abstracts by Grab did well. They showed that Grab can pick out the important stuff and spot patterns, even when there's a lot of other noise. Even though Kramp which is a version of Krimp looks at the biggest area, it doesn't always do the best. And Krimp seems to be a bit behind Slim and Slam when it comes to packing data tightly, even for bigger tasks. This might mean that the way Slim guesses to get results might be better and works faster. The overall studies performed in the evaluation of the three algorithms doesn't really talk about the problem of fitting too closely to the datasets they used for the experiment. If an algorithm works too well on these datasets, it might not work as well on other datasets. Talking about this more, or testing it, could make the study stronger. Looking at the Mammals and Plants datasets, Grab finds more than 10 length tails. It would be interesting to find out how many of these tails match up to real patterns, and how many are just noise. We're not really sure how much Grab lets noise variables into its rules and how much noise in a transaction does it tolerate before letting the rule to explain it . It would be helpful to have a part in the algorithm that makes it less good when there's too much noise in the rules.An idea could be to integrate a mechanism within the algorithm that penalizes the excessive noise into the derived rules Despite this, Grab does a great job of quickly making clear and helpful rule sets. It does really well on the experiment datasets, where it gets the real deal while ignoring the noise.

5 Conclusion

In conclusion, the algorithms presented in the papers, Slim, Krimp, and Grab, provide unique approaches to mining patterns and association rules from data. They achieve this by leveragLength principle, with each focusing on differing the Minimum Description ent aspects: Slim and Krimp tackle the challenge of pattern selection and compression, while Grab handles noisy data and find corelations association rules. The procedures they employ are broadly similar, despite varying in their methods of candidate generation, dealing with overlapping patterns and code to describe the data. Each algorithm exhibits potential advantages and limitations, such as the large number of potential patterns generated by Slim and Grab when handling numerous singletons, or the disregarding of frequency information by them. Future directions could include exploration of anti-correlation . Despite these challenges, the algorithms have shown promising results in various data contexts, providing high-quality, fast, and compact results.

References

Vreeken, Jilles, Matthijs Van Leeuwen, and Arno Siebes (2011). "Krimp: mining itemsets that compress". In: *Data Mining and Knowledge Discovery* 23, pp. 169–214.

Smets, Koen and Jilles Vreeken (2012). "Slim: Directly mining descriptive patterns". In: *Proceedings of the 2012 SIAM international conference on data mining*. SIAM, pp. 236–247.

Fischer, Jonas and Jilles Vreeken (2019). "Sets of robust rules, and how to find them". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 38–54.