LINFO2364 - Project 2 - Group 48
Algorithms for sequence mining & classification

Bourez Nicolas
UCLouvain
Ecole polytechnique de Louvain-la-Neuve
Louvain-la-Neuve, Belgium
28462000

Demaret Dimitri
UCLouvain
Ecole polytechnique de Louvain-la-Neuve
Louvain-la-Neuve, Belgium
38502000

I. Introduction

Frequent sequence mining is a crucial task in data mining, with wide-ranging applications across various domains. It involves identifying sequential patterns or subsequences that occur frequently within a dataset. Among the array of algorithms designed to address this challenge, the SPADE algorithm is a very effective one. In the first part of this report, we will explain how we implement this algorithm and justify our implementation choices.

This algorithm allows us then to perform a classification task using the most present patterns and the patterns with the highest WRACC score. Many real-world datasets, such as DNA sequences, text documents, clickstream data, and financial transactions, are sequential. Traditional classification methods may not effectively capture the sequential nature of these datasets. Sequence mining techniques, however, are specifically designed to handle sequential data, making them more suitable for classification tasks in such domains.

Through rigorous analysis and experimentation on multiple datasets, this report aims to provide a comprehensive comparison to understand the strengths and weaknesses of the different scores used in sequence classification. By examining factors such as computational efficiency, scalability, and accuracy, we seek to uncover the conditions under which score excels.

II. Description of the SPADE algorithm

A. Most frequent sequences mining

This task aimed to find the k most frequent sequence in a given dataset. However, the first subtility of the task was that the data was split into two parts: the negative transactions and the positives one. To simplify the task and go back to a one-build dataset, we decided to flatten these datasets, keeping in my that the indexes between `[0, len(positive_transactions)]` correspond to the positive and the rest to the negative.

First, we started by implementing a dummy version of the SPADE algorithm. We chose to build a tree data structure, starting from all the singleton itemsets and then merging the one with the same prefix. At each depth, we iterate two times to find all the possible sequences. It was a very inefficient version but was theoretically correct, as we tested it on a toy dataset.

We then choose to improve several points:

- Instead of a simple double for loop, we decided to implement a depth-first search, as presented in the slides of the course. This implementation choice allows the use of a more efficient search of the frequent itemsets and prunes the tree at some step of the iteration.
- We decide then to use the anti-monotonicity property that frequent sequence mining offers us. Indeed, if a sequence A is included in a sequence B, and if this sequence A is unfrequent, we are sure that the frequency B will be unfrequent. This nice property allows us to skip a huge part of the tree and improve the complexity of the algorithms significantly.

  To prune the tree we decided to sort the sequence in reverse order and took the count of the k-th as a minimum bound at each iteration of the depth-first. If the count of a sequence is lower than this bound, we do not take it into account in the next recursion of the search. If the sequence bound is higher, then we add this sequence and update the bound. The search stops when no sequence is higher than the bound anymore.

B. Supervised sequences mining

What can be interesting is changing the score (the criterion that our bound is based on) and choosing a more appropriate one for classification than just looking for the most frequent sequences. We decided to use the Weighted relative accuracy which consists in :

$$WRACC(x) = \left(\frac{P}{P+N} * \frac{N}{P+N}\right) * \left(\frac{p(x)}{P} - \frac{n(x)}{N}\right)$$

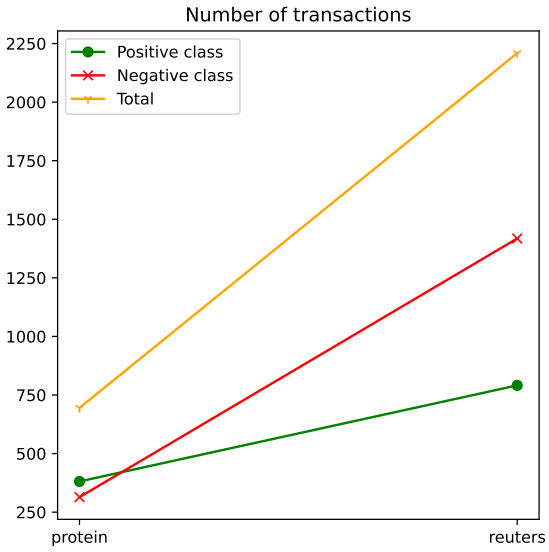
The sequence with a high positive WRACC will be highly present in the positive class, and conversely. The particularity of this score is that we are losing the anti-monotonicity. Indeed, if we take our previous example, the sequence A

might have a low WRACC but the sequence B a high one. It depends on the number of elements present in the positive and negative sets.

To tackle this problem, we decide to relax our pruning condition; indeed, instead of taking only the sequence above the given threshold, we accept the sequence with a lower WRACC if its score with zero elements in the negative class is higher than the threshold (`compute_wracc(count_positive,0) > threshold`). Indeed, even if our sequence A has a low score, we know that the positive count of the sequence B will be bound by the count positive of A, but we allow the fact that the count negative might be lower (to max zero) and thus have a higher WRACC score than A.

C. Datasets analysis

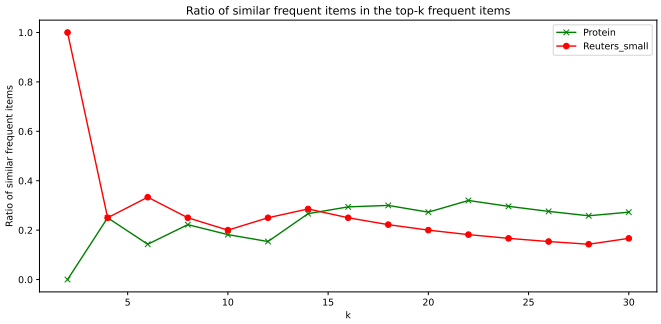
Before analyzing the algorithm's performance, we should first look at the datasets over which they will run. We have at our disposal 2 datasets, The Figure II-C below allows us to obtain precious information about them. All of our plots are saved under `.pdf` format, it is thus possible to zoom to see them clearly.



The plot shows the transaction's repartition into positive and negative transactions. We observe two different repartitions, `Protein` has nearly as many negative transactions as positive (more or less $50\backslash 50$) while the `Reuters_small` dataset has closer to a $2\backslash 3$ fraction of negative transactions. Also, we see that there are nearly 3 times more transactions in the `Reuters_small` datasets than the `Protein`.

D. Score analysis - Comparison Frequent sequence and WRACC

Now that we know the composition of the datasets, let us look for similarities and differences between the scoring we implemented before analyzing their performances. To do so, we plotted a graph of the ratio of frequent sequences that are taken by both the frequency mining and the WRACC algorithm for different values of k (see Figure II-D).



We observe that this ratio is quite similar for both datasets and is quite bad (nearly never above 40%) implying that those scoring look for criterion not too much correlated. We will now interest ourselves in the repartition of the count and the WRACC score over the top 20 items for each of those scoring. This can be seen in Figure 1.

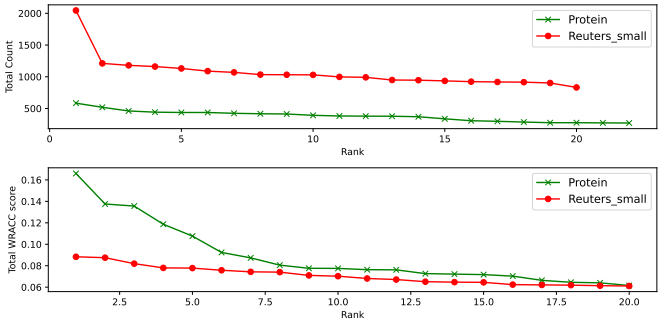


Fig. 1. Scores of the top 20

The first plot shows the evolution of the total count of the top 20 most frequent sequences (which is the 22 best sequences in the case of the `Protein` dataset). The second one shows the WRACC score of the top 20 using weighted relative accuracy. The total count seems more uniform through the top 20 members, while the WRACC score tends to decrease significantly, especially in the first sequences, those with the best score.

III. Classification task

A. Cross-validation

1) Implementation: We will only go through the main steps of the implementation :

1) We create copies of our positive and negative transactions, then we start a for boucle from 1 to $kfold$ (number of folds).
2) We get the values of the transactions in the training for both the positive and negative transactions (also for the testing).

3) We look for our top-k sequences according to the chosen algorithm.
4) We store in the function `get_feature_matrices` a binary matrix where the entries are for each sequence of the top-n frequencies if it is in each transaction.
5) Finally, we create a classifier using our binary matrix and `DecisionTreeClassifier` we get the prediction and compute the accuracy over each fold.

2) Classifiers: Now that we have a cross-validation algorithm, we can use it to train our previous algorithms (frequent sequences mining and best WRACC score). We ran our training and computed the mean accuracy over all the folds for different values of $kflod$. You can see the result obtained using the top 10 sequences produced by our algorithms in Figure 2.

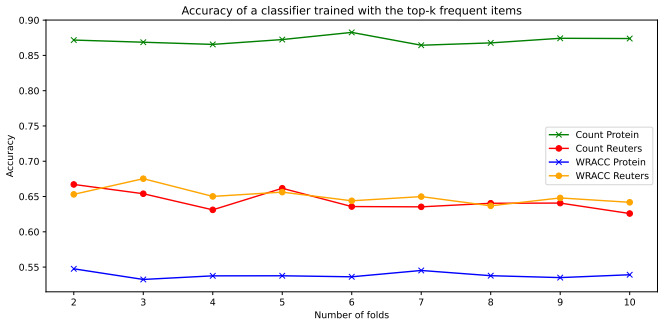


Fig. 2. Accuracies of classifiers for different numbers of folds

We observe that for $kfold$ from 2 to 10 the mean accuracy over the folds does not vary much. However, it is very interesting to see that the precision of the training using frequent sequence mining performs much better than the other for the `Protein` dataset and that their performances are nearly exactly as accurate for both algorithms in the `Reuters_small` dataset.

Let us keep $kfold = 6$ as it gave us our best accuracy result and let us look for the impact of the top-n "best" items with Figure 3.

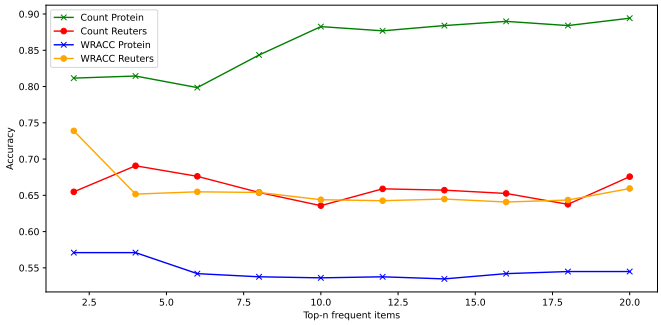


Fig. 3. Accuracy for different top n itemsets in the training

We do not see big differences apart on the `Protein` with frequent sequences mining where there is a clear increase in the accuracy when we have 10 or more top sequences.

Discussion For the protein set, we can see a huge difference between the counting and the wracc score. Has the negative and positive transactions are equally present in the dataset, if we only take the highest wracc score, we might not be able to classify well the negative value which lead to a bad classification accuracy. What might be more interesting is to take the absolute value of the wracc score. In the reuters dataset, the different of proportion between the positive and the negative transactions attenuate this bad accuracy.

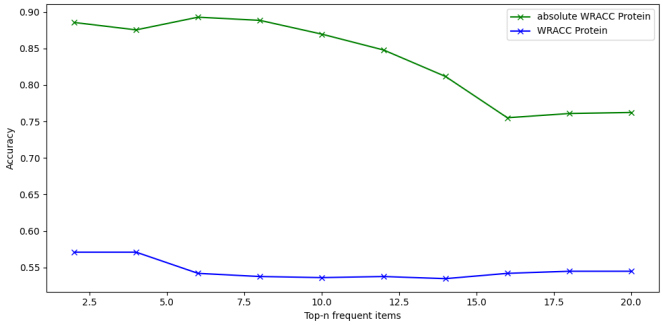


Fig. 4. abs wracc vs wracc

One last interesting analysis that we can do is compare the accuracy over the 6 folds returns when using the top 20 (frequent sequences mining) over the `Protein` dataset. Figure 5 shows the accuracies obtained for each fold. This plot allows us to conclude that there are clear differences between the fold's accuracies.

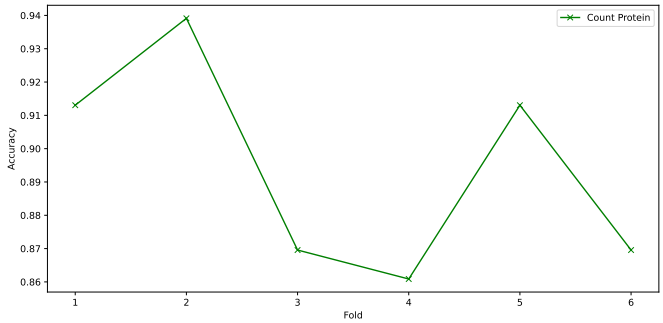


Fig. 5. Accuracy for each of the 6 folds

B. Iterative algorithm

The aim of this section is to explain how we tackle the iterative algorithm problem, meaning only one pattern will be chosen and the dataset will be updated at each iteration of the algorithm. We decide to implement the Greedy algorithm for the pattern-based decision tree, and more precisely, the

CART algorithm introduced by Pierre Dupont in the course LGBIO2010. [1]

Algorithm CART
Input: A labeled training dataset
$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ // y_i is the class of x_i
Input: The root node of the current (sub-)tree // Initially, T is a single node tree
Input: n_0 // A number of examples below which splitting is stopped
Input: imp_0 // An impurity level below which splitting is stopped
Output: A classification tree T
if $|S| \geq n_0$ *AND* $Gini(S) \geq imp_0$ **then**
  // Feature selection at current node
  Assign binarized feature x_j maximizing $Drop(S, x_j)$ to node
  foreach *value* v *of* x_j **do** // Only 2 possible values
    Add new $node_v$ as a child of node in T
    Attach $S_v^j = \{x \in S | x_j = v\}$ to $node_v$ // Split S according to x_j values
    $CART(S_v^j, node_v)$
else
  Assign majority class label to node
return T

With a list of transactions and a max_depth as input parameters, this algorithm aims to build a tree (in our case, a binary tree since it has only 2 classes), where the paths that start from the root to the leaf end with a classification of the considered transaction (0 or 1). At each iteration, we select the pattern that maximizes the drop of impurity (we can choose the gini or the entropy as the information gain score). We then drop this pattern from the transactions where it is present and we split the dataset to have two children nodes: one on the right corresponding to the transactions where the pattern is not present (so the 0 class), and one on the left corresponding to the transactions where the pattern is present (the 1 class). Where the max_depth is reached, we choose the class the most present in the remaining transactions.

One well-known problem of this algorithm is overfitting. Indeed, it is very discriminating and the training can lead to a precision of 100% in some cases which can look great apriori, but can't classify new datasets well. Choosing an appropriate max_depth helps prevent overfitting. This hyperparameter can be chosen using cross-validation (splitting the dataset into 3 parts: a training set, a validation set, and a test set).

Here are the best results we get in the function of the max_depth :

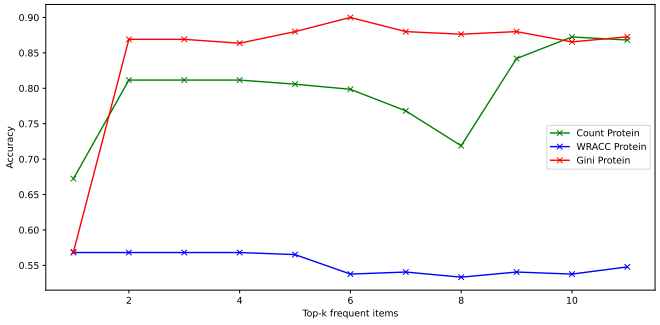


Fig. 6. Comparaison of the accuracies obtained

In Figure 6, the red line shows the different results according to the max depth k parameter using cross-validation. To get this result, we compared the accuracy obtained with the one obtained for different k best sequences with the frequent sequence mining and the supervised sequences mining. We can observe that we have a very bad result for a depth of 1 but it immediately improves significantly to reach even a better precision that we had before. We runned our algorithm on the `Protein` dataset as it gaves the best accuracies we had with the other algorithms.

IV. Conclusion

This report has demonstrated the effectiveness of the SPADE algorithm and has compared different scoring methods in classification across various datasets. Our findings suggest that while Step-Wise approach for pattern-based models are a solid base, incorporating other iterative method as CART algorithm which build a pattern based decision tree can be a good other way to classify transactions. We can also see that the information gain is giving the best result, above the wracc and the counting score.

References

[1] P.Dupont, V. Branders - Introduction to Bioinformatics LGBIO2010: Predictive Modeling, January 2024