

Supervised Evaluation of Top- k Itemset Mining Algorithms

Claudio Lucchese¹, Salvatore Orlando²(✉), and Raffaele Perego¹

¹ ISTI-CNR, Pisa, Italy

² DAIS - Università Ca' Foscari, Venice, Italy

orlando@unive.it

Abstract. A major mining task for binary matrixes is the extraction of approximate top- k patterns that are able to concisely describe the input data. The top- k pattern discovery problem is commonly stated as an optimization one, where the goal is to minimize a given cost function, e.g., the accuracy of the data description.

In this work, we review several greedy state-of-the-art algorithms, namely ASSO, HYPER+, and PANDA⁺, and propose a methodology to compare the patterns extracted. In evaluating the set of mined patterns, we aim at overcoming the usual assessment methodology, which only measures the given cost function to minimize. Thus, we evaluate how good are the models/patterns extracted in unveiling supervised knowledge on the data. To this end, we test algorithms and diverse cost functions on several datasets from the UCI repository. As contribution, we show that PANDA⁺ performs best in the majority of the cases, since the classifiers built over the mined patterns used as dataset features are the most accurate.

1 Introduction

Binary matrixes can be derived from several typologies of datasets, collected in diverse and popular application domains. Without loss of generality, we can think of a binary matrix as a representation of a transactional database, composed of a multi-set of transactions (matrix rows), each including a set of items (matrix columns). An *approximate pattern* extracted from a binary matrix thus corresponds to a pair of sets, items and transactions, where the items of the former set are mostly included in all the transactions of the latter set. The cardinality of the transaction set (matrix rows) can also be seen as the approximate support of the corresponding set of items.

Top- k pattern mining is an alternative approach to pattern enumeration. It aims at discovering the (small) set of k patterns that best *describes*, or *models*, the input dataset. State-of-the-art algorithms differ in the formalization of the above concept of *dataset description*. For instance, in [1] the goodness of the description is given by the number of occurrences in the dataset incorrectly modeled by the extracted patterns, while shorter and concise patterns are promoted in [2,3]. The goodness of a description is measured with some cost function,

and the top- k mining task is casted into an optimization of such cost. In most of such formulations, the problem is demonstrated to be NP-hard, and therefore greedy strategies are adopted. At each iteration, the pattern that best optimizes the given cost function is added to the solution. This is repeated until k patterns have been found or until it is not possible to improve the cost function.

In this paper we analyze in depth three state-of-the-art algorithms for mining approximate top- k pattern from binary data: ASSO [1], HYPER+ [2] and PANDA+ [4]. Indeed, the cost functions adopted by ASSO [1] and HYPER+ [2] share important aspects that can be generalized into a unique formulation. The PANDA+ framework can be plugged with such generalized formulation, which makes it possible to greedily mine approximate patterns according to several cost functions, including the ones proposed in [3, 6]. PANDA+ also permits expressing noise constraints [5].

Concerning the evaluation methodology, we observe that state-of-the-art algorithms for approximate top- k mining measure the goodness of the discovered patterns by their capability of minimizing the same cost function optimized by the greedy algorithm. This simple assessment methodology captures the effectiveness of the greedy strategy rather than the quality of the extracted patterns. In this paper, we want to go beyond this common evaluation approach by adopting an assessment methodology aimed at measuring also how good are the concise models extracted in unveiling some hidden supervised knowledge, in particular the class labels associated with transactions.

We test this capability by using the algorithms for approximate top- k mining as a sort of feature extractors. The accuracy of the classifiers built on top of the extracted features is then considered a proxy for the quality of the mined patterns. In this way, we are able to complement internal indices of quality (cost function), with external ones (classification accuracy). PANDA+, which is able to optimize several cost function, permits to extract those patterns/features that when used to train a classifier, produces models that are in the majority of the cases the most accurate.

2 Problem Statement and Algorithms

In this sections we first introduce some notation and the problem statement, and then we briefly discuss the main features and peculiarities of three state-of-the-art algorithms solving our approximate top- k pattern mining problem.

2.1 Notation and Problem Statement

A *transactional dataset* of N transactions and M items can be represented by a *binary matrix* $\mathcal{D} \in \{0, 1\}^{N \times M}$ where $\mathcal{D}(i, j) = 1$ if the i -th item occurs in the j -th transaction, and $\mathcal{D}(i, j) = 0$ otherwise. An *approximate pattern* P is identified by the set of items it contains and the set of transactions where these items (partially) occur. We represent these two sets as binary vectors $P = \langle P_I, P_T \rangle$, where $P_I \in \{0, 1\}^M$ and $P_T \in \{0, 1\}^N$. The outer product $P_T \cdot$

$P_I^T \in \{0, 1\}^{N \times M}$ identifies a sub-matrix of \mathcal{D} . Being each pattern approximate, the sub-matrix should mainly cover 1-bits in \mathcal{D} (*true positives*), but it may also cover a few 0-bits too (*false positives*).

Finally, let $\Pi = \{P_1, \dots, P_{|\Pi|}\}$ be a set of *overlapping patterns*, which approximately cover dataset \mathcal{D} , except for some noisy item occurrences, identified by matrix $\mathcal{N} \in \{0, 1\}^{N \times M}$:

$$\mathcal{N} = \bigvee_{P \in \Pi} (P_T \cdot P_I^T) \vee \mathcal{D}. \quad (1)$$

where \vee and \bigvee are respectively the element-wise *logical or* and *xor* operators. Note that some 1-bits in \mathcal{D} may not be covered by any pattern in Π (*false negatives*). Indeed, our formulation of noise (matrix \mathcal{N}) models both *false positives* and *false negatives*. If an occurrence $\mathcal{D}(i, j)$ corresponds to either a false positive or a false negative, we have that $\mathcal{N}(i, j) = 1$.

On the basis of this notation, we can state the top- k pattern discovery problem as an optimization one:

Problem 1 (Approximate Top- k Pattern Discovery Problem). Given a binary dataset $\mathcal{D} \in \{0, 1\}^{N \times M}$ and an integer k , find the pattern set $\bar{\Pi}_k$, $|\bar{\Pi}_k| \leq k$, that minimizes the given cost function $J(\Pi_k, \mathcal{D})$:

$$\bar{\Pi}_k = \underset{\Pi_k}{\operatorname{argmin}} J(\Pi_k, \mathcal{D}) \quad (2)$$

In the following we review some cost functions and the algorithms that adopt them. These algorithms try to optimize specific functions J with some greedy strategy, since the problem belongs to the NP class. In addition, they exploit some specific *parameters*, whose purpose is to make the pattern set Π_k subject to particular *constraints*, with the aim of (1) reducing the algorithm search space, or (2) possibly avoiding that the greedy generation of patterns brings to local minima. As an example of the former type of parameters, we mention the frequency of the pattern. Whereas, for the latter type of parameters, an example is the amount of false positives we can tolerate in each pattern.

2.2 Minimizing Noise (ASSO)

ASSO [1] is a greedy algorithm aimed at finding the pattern set Π_k that minimizes the amount of noise in describing the input data matrix \mathcal{D} . This is measured as the L^1 -norm $\|\mathcal{N}\|$ (or Hamming norm), which simply counts the number of 1 bits in matrix \mathcal{N} as defined in Eq. (1). ASSO is thus a greedy algorithm minimizing the following function:

$$J_A(\Pi_k, \mathcal{D}) = \|\mathcal{N}\|. \quad (3)$$

Indeed, ASSO aims at finding a solution for the *Boolean matrix decomposition problem*, thus identifying two low-dimensional factor binary matrices of rank k , such that their *Boolean product* approximates \mathcal{D} . The authors of ASSO called this matrix decomposition problem the Discrete Basis Problem (DBP). It can

be shown that the DBP problem is equivalent to the approximate top- k pattern mining problem when optimizing J_A . The authors prove that the decision version of the problem is NP-complete by reduction to the set basis problem, and that J_A cannot be approximated within any factor in polynomial time, unless $P = NP$.

ASSO works as follows. First it creates a set of candidate item sets, by measuring the correlation between every pair of items. The minimum confidence parameter τ is used to determine whether two items belong to the same item set. Then ASSO iteratively selects a pattern from the candidate set by greedily minimizing the J_A .

2.3 Minimizing the Pattern Set Complexity (HYPER+)

The HYPER+ [2] algorithm works in two phases. In the first phase (corresponding to the covering algorithm HYPER+ [7]), given a collection of frequent item sets, the algorithm greedily selects a set of patterns Π^* by minimizing the following cost function that models the pattern set complexity:

$$J_H(\Pi^*, \mathcal{D}) = \sum_{P \in \Pi^*} (\|P_I\| + \|P_T\|). \quad (4)$$

During this first phase, the algorithm aims to cover in the best way all the items occurring in \mathcal{D} , with neither false negatives nor positives, and thus without any noise. The rationale is to promote the simplest description of the input data \mathcal{D} . Note that the size of Π^* is unknown, and depends on the amount of patterns that suffice to cover the 1-bits in \mathcal{D} . The minimum support parameter σ is used by HYPER+ to select an initial set of frequent item sets for starting the greedy selection phase, thus reducing the search space of the greedy optimization strategy.

Concerning the second phase of the algorithm, pairs of patterns in Π^* are recursively merged as long as a new collection Π' , with a reduced number of patterns, can be obtained without generating an amount of *false positive* occurrences larger than a given budget β . Finally, since the pattern set Π' is ordered (from most to least important), we can simply select Π_k as the top-listed k patterns in Π' , as done by the algorithm authors in Sect. 7.4 of [2]. Note that this also introduces false negatives, corresponding to all the occurrences $\mathcal{D}(i, j) = 1$ in the dataset that remain uncovered after selecting the top- k patterns Π_k only.

2.4 Minimizing Multiple Cost Functions (PANDA+ Framework)

PANDA+ is a framework recently proposed [4] that inherits the optimization engine of PANDA [3]. Thus it adopts a greedy strategy by exploiting a two-stage heuristics to iteratively select each pattern: (a) discover a noise-less pattern that covers the yet uncovered 1-bits of \mathcal{D} , and (b) extend it to form a good approximate pattern, thus allowing some false positives to occur within the pattern. Rather than considering all the possible exponential combinations of items, these are sorted to maximize the probability of generating large cores, and processed one at the time without backtracking.

Table 1. Objective functions for Top- k pattern discovery problem.

Cost Function	Description
$J_A(\Pi_k, \mathcal{D}) = J^+(\Pi_k, \mathcal{D}, \gamma_{\mathcal{N}}(\mathcal{N}) = \ \mathcal{N}\ , \gamma_P(P) = 0, \rho = 0)$	Minimize noise [1]
$J_H(\Pi_k, \mathcal{D}) = J^+(\Pi_k, \mathcal{D}, \gamma_{\mathcal{N}}(\mathcal{N}) = 0, \gamma_P(P) = \ P_T\ + \ P_I\ , \rho = 1)$	Minimize pattern set complexity [7]
$J_P(\Pi_k, \mathcal{D}) = J^+(\Pi_k, \mathcal{D}, \gamma_{\mathcal{N}}(\mathcal{N}) = \ \mathcal{N}\ , \gamma_P(P) = \ P_T\ + \ P_I\ , \rho = 1)$	Minimize noise and pattern set complexity [3, 9]
$J_P^{\bar{\rho}}(\Pi_k, \mathcal{D}) = J^+(\Pi_k, \mathcal{D}, \gamma_{\mathcal{N}}(\mathcal{N}) = \ \mathcal{N}\ , \gamma_P(P) = \ P_T\ + \ P_I\ , \rho = \bar{\rho})$	Extend J_P to leverage the trade-off between noise and pattern set complexity
$J_E(\Pi, \mathcal{D}) = J^+(\Pi, \mathcal{D}, \gamma_{\mathcal{N}}(\mathcal{N}) = \text{enc}(\mathcal{N}), \gamma_P(P) = \text{enc}(P), \rho = 1)$	Minimize the encoding length [8] of the pattern model according to [6]

The PANDA⁺ [3] original cost function J_P can be replaced without harming its greedy heuristics. Indeed, J_P , which is simply the sum of $J_A(\cdot)$ and $J_H(\cdot)$ – see Eqs. (3) and (4) – is generalized as follows:

$$J^+(\Pi_k, \mathcal{D}, \gamma_{\mathcal{N}}, \gamma_P, \rho) = \gamma_{\mathcal{N}}(\mathcal{N}) + \rho \cdot \sum_{P \in \Pi_k} \gamma_P(P) \quad (5)$$

where \mathcal{N} is the noise matrix defined by Eq. (1), $\gamma_{\mathcal{N}}$ and γ_P are user defined functions measuring the cost of the noise and patterns descriptions respectively, and $\rho \geq 0$ works as a regularization factor weighting the relative importance of the patterns cost.

Table 1 shows how the cost function defined by Eq. (5) can be instantiated to obtain all the functions discussed above, and allows for new functions to be introduced, by fully leveraging the trade-off between patterns description cost and noise cost (thanks to parameter ρ). Note the $J_P^{\bar{\rho}}$ is a generalization of the function J_P already proposed for PANDA, with parameter $\bar{\rho}$ that determines a different trade-off between patterns description cost and noise cost. In addition, one instance of the generalized cost function is J_E , originally proposed in [6] to evaluate a pattern set Π_k . According to the MDL principle [8], the regularities in \mathcal{D} , corresponding to the discovered patterns Π_k , can be used to *lossless compress* \mathcal{D} : thus the best pattern set Π_k is the one that induces the smallest encoding of \mathcal{D} . Finally, in order to avoid the greedy search strategy accepting too noisy patterns, we introduced into PANDA⁺ two maximum noise thresholds $\epsilon_r, \epsilon_c \in [0, 1]$, inspired by [5], aimed at bounding the maximum amount of noise generated by adding a new item or a new transaction to a pattern.

3 Evaluation Methodology and Experiments

We observe the lack of common real-world benchmarks, e.g., datasets for which the most important embedded patterns are known, and that can be used as a

Table 2. Characteristics of the datasets used for the experiments

Dataset	# Classes	# Items	# Transactions	Avg. trans. length
Abalone	3	40	4177	8.0
Anneal	5	108	898	38.0
Audiology	24	154	226	67.6
Auto	6	129	205	24.7
Congres	2	32	434	15.1
Credita	2	70	690	14.9
CylBands	2	120	540	33.2
Dermatology	6	43	366	12.0
Diabetes	2	40	768	8.0
Ecoli	8	26	336	7.0
Flare	8	30	1389	10.0
Glass	6	40	214	9.0
Heart	5	45	303	13.0
Hepatitis	2	50	155	17.9
HorseColic.D85	2	81	368	16.8
Ionosphere	2	155	351	34.0
Iris	3	16	150	4.0
Mushroom	2	88	8124	21.7
Pima	2	36	768	8.0
Sick	2	75	3772	27.4
Soybean-large	19	99	683	31.6
Vehicle	4	90	846	18.0
Wine	3	65	178	13.0
Zoo	7	35	101	16.0

ground truth to evaluate and compare algorithms. We thus propose a *supervised evaluation methodology* of the extracted top- k patterns that measures and compares the ability of each algorithm in discovering interesting transaction features that we can use to train accurate classifier.

To this end we focus on 24 datasets from the UCI repository¹ which have been discretized for pattern mining². The main characteristics of the datasets are reported in Table 2. In all the experiments, we thus remove class labels before feeding any top- k approximate item sets mining algorithm. Such patterns are used as features to train and test an SVM classifier.

3.1 Parameter Setting of Pattern Mining Algorithms

We apply HYPER+, ASSO, and PANDA+ over the 24 UCI datasets, by requiring the mining algorithms to extract a number of patterns equal to *twice* the number

¹ <http://archive.ics.uci.edu/ml/>.

² http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/.

of classes present in each distinct dataset. Although the number of classes is a good proxy for the actual number of interesting patterns occurring in the data, we doubled this number to deal with the presence of multiple dense subgroups that characterize a single class, as found in [6]. Also, note that all the three algorithms may produce less patterns than required if none is found to improve the objective function.

Although HYPER+ allows for setting the maximum number of patterns k to extract, unfortunately we found this option to perform poorly, resulting in excessive noise. This is due to the covering constraint of the algorithm: false negatives are not allowed, and the extracted k patterns must cover all the occurrences $\mathcal{D}(i, j) = 1$ in the dataset. It is possible to achieve much better results by tuning the algorithm noise budget β , and then accepting only the k top-listed patterns. We fine-tuned the β parameter on every single dataset by choosing β in the set $\{1\%, 10\%\}$. Moreover, we used frequent closed item sets in order to take advantage of a lower minimum support thresholds, which is the most sensitive parameter of HYPER+. We swept the minimum support threshold in the interval $[10\%, 90\%]$ by increments of 10%.

The ASSO algorithm has a minimum correlation parameter τ which determines the initial patterns candidate set. Indeed, ASSO is very sensitive to this parameter. We fine-tuned the algorithm independently on every single dataset, by tuning τ in the range $[0.5, 1]$ with steps of 0.05. In our experiments we always tested the best performing variant of the algorithm which is named ASSO + *iter* in the original paper.

We evaluated four variants of PANDA+ optimizing different cost functions: J_A , J_P , J_E , $J_P^{1.2}$. Recall that J_A measures the noise cost only, J_P adds the cost of each pattern, J_E measures the cost of an optimal MDL encoding, and $J_P^{1.2}$ mimics J_E by using a larger weight for the cost of the patterns cost. In addition, PANDA+ uses two maximum noise thresholds ϵ_r and ϵ_c , to control the maximum amount of noise on each pattern row or column. Also in this case, on each dataset we swept parameters ϵ_r and ϵ_c in the range $[0.0, 0.5]$, with steps of 0.05, and also considering the value 1.0 which is equivalent to ignoring the thresholds.

3.2 Supervised Evaluation of Pattern Set

Each binary UCI dataset, from which the class labels is removed in advance, is first subdivided into training and test set. Indeed, we use 10-fold cross validation, so a training set includes 9/10 of all transactions, and the corresponding test set the remaining 1/10 ones. We employ the three algorithms, instantiated with a given parameter set, to extract a distinct a pattern set Π_k from the training set. Then, every transaction in the training and test sets is mapped into the pattern space by considering a binary feature for each approximate pattern in Π_k indicating its presence/absence in the transaction. Note that the algorithms we evaluated produce, for each pattern $\langle P_I, P_T \rangle \in \Pi_k$, the set of transactions P_T in the training where P_I it occurs, and this set is used for the mapping. For what regard the test set, we say that an approximate pattern $P \in \Pi_k$ occurs in an unseen test transaction t iff $|P_I \cap t|/|P_I| \geq \eta$, where η is the minimum intersection

ratio $|P_I \cap t^*|/|P_I|$ for every training transaction $t^* \in P_T$. The rationale is to accept a pattern P for a test transaction t if it does not generate more noise than what it has been observed in the training set.

After mapping the transactions into such pattern space, the class labels are restored in the transformed training and test sets. The training set is thus used to train an SVM classifier. Finally, the classifier is evaluated on the mapped test set. Specifically we adopt the implementation provided by [10], in which we use a radial basis function as SVM kernel, and estimate its crucial regularization parameter as in [11].

It is worth noting that, unlike [12], the features extracted from each transaction only correspond to the patterns discovered by the given algorithm: we do not consider the singletons as a transaction feature in classifier training/test data, unless such patterns composed of single items are actually extracted by the mining algorithm. This is because we want to evaluate exclusively the predictive power of the mined collection of patterns Π_k .

Since the various algorithms take as input several parameters summarized in Fig. 1(a), we exploit a parameter sweeping technique, like in [1] and [6], aimed at understanding the full potential of the algorithms taken into consideration.

Specifically, we adopt two distinct approaches to derive the best classifier for each dataset, and thus testing its accuracy. In the former approach – see Fig. 1(b) – we sweep over the input parameters of the given algorithm, and select 'ex ante' the best parameters on the basis of the specific cost function J used to evaluate the quality of the mined pattern sets. We thus conduct 10-fold cross validation to evaluate the goodness of the algorithm as follows: at each fold the top- k patterns are extracted from the training set using the best parameters previously found, then they are exploited as features to map the data in a new feature space, and finally, the mapped data is used to train and test an SVM classifier. In the latter approach – see Fig. 1(c) – we use parameter sweeping to explore the maximum accuracy achievable by a given algorithm. The classifier resulting from every combination of the algorithm parameters is evaluated with

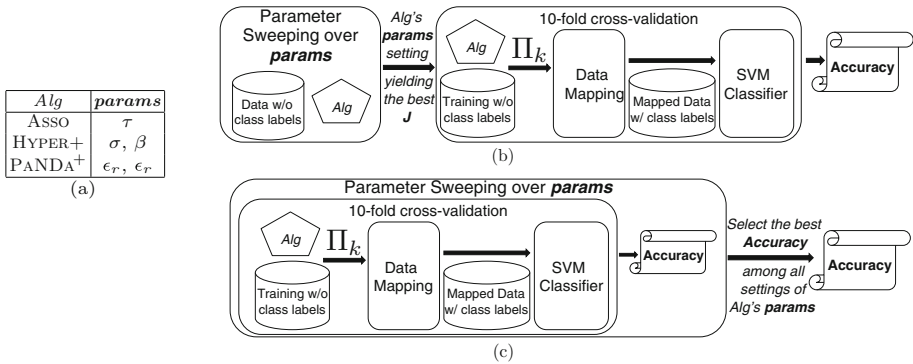


Fig. 1. Supervised evaluation: (a) algorithms' parameters, (b) unsupervised and (c) supervised selection of the algorithms' parameters.

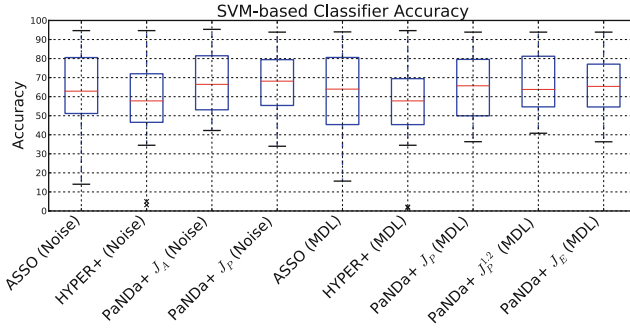


Fig. 2. Box-plots of the accuracy of the SVM-based classifier with pre-optimized algorithm parameters across the various UCI datasets (HYPER+ did not complete on audiology). Leftmost data correspond to noise-based parameter tuning and rightmost data to MDL-based parameter tuning.

same 10-fold cross validation process, and the best accuracy is used to measure the goodness of the algorithm.

3.3 Experimental Results

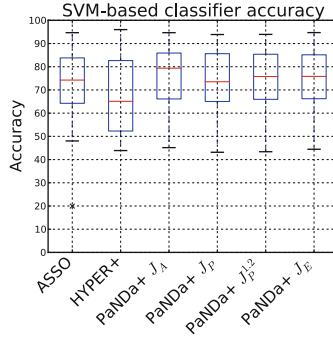
The first of our experiments compares ASSO, HYPER+ and PANDA+ when their parameters are chosen according to their ability to optimize J_A or J_E , as illustrated in Fig. 1(b). Results reported in Fig. 2 show that the performance of HYPER+ are better than expected. Even if HYPER+ is not able to optimize well neither J_A nor J_E , the resulting patterns provide a reasonably good accuracy. We observe that the J_E cost function leads to poorer patterns on average. This is quite surprising, since we expected that a purely noise-based cost function would lead to over-fitting patterns and that the MDL-based cost would promote patterns with a larger generalization power. The best median is achieved by PANDA+ J_P , which looks to be a good compromise between noise minimization and generalization power. In Table 3 we report the results of the pairwise comparison, which confirm the good performance of PANDA+ J_P when its parameters are determined on the basis of J_A .

The above experiments, show that the approximate frequent pattern mining algorithms we took into consideration provide similar performance, with a slight preference for PANDA variants, and they also show that optimizing J_A is preferable to optimizing J_E .

We also measured the maximum accuracy that each algorithm can achieve, as illustrated in Fig. 1(c). The results of this experiment are shown in Fig. 3 and in Table 4. Except for HYPER+, all the algorithms significantly increase their median accuracy with 10% improvement, the best performing being PANDA+ J_A with a median accuracy close to 80%. Recall that PANDA+ J_A was not the best algorithm at optimizing J_A . Again, we can state that neither J_A nor J_E are good hints for the discovery of predictive patterns. Moreover, PANDA+ J_A performs almost always better the HYPER+, and beats ASSO in

Table 3. Number of times an algorithm (column) generated better/equal/worse patterns than the baseline (row) based on the corresponding SVM accuracy.

		Noise-based parameters				MDL-based parameters				
		ASSO	HYPER+	PANDA ⁺ J_P	PANDA ⁺ J_A	ASSO	HYPER+	PANDA ⁺ J_P	PANDA ⁺ $J_P^{1,2}$	PANDA ⁺ J_E
Noise-based	ASSO	(24)	10/2/11	16/1/7	11/1/12	6/9/9	10/1/12	11/1/12	11/1/12	13/2/9
	HYPER+	11/2/10	(23)	15/1/7	14/0/9	11/2/10	7/6/10	13/1/9	13/1/9	12/2/9
	PANDA ⁺ J_P	7/1/16	7/1/15	(24)	9/2/13	7/2/15	5/1/17	5/2/17	8/2/14	7/2/15
	PANDA ⁺ J_A	12/1/11	9/0/14	13/2/9	(24)	11/1/12	9/0/14	7/1/16	9/1/14	10/1/13
MDL-based	ASSO	9/9/6	10/2/11	15/2/7	12/1/11	(24)	10/1/12	12/2/10	10/2/12	12/3/9
	HYPER+	12/1/10	10/6/7	17/1/5	14/0/9	12/1/10	(23)	13/1/9	15/1/7	13/1/9
	PANDA ⁺ J_P	12/1/11	9/1/13	17/2/5	16/1/7	10/2/12	9/1/13	(24)	12/2/10	11/2/11
	PANDA ⁺ $J_P^{1,2}$	12/1/11	9/1/13	14/2/8	14/1/9	12/2/10	7/1/15	10/2/12	(24)	12/2/10
	PANDA ⁺ J_E	9/2/13	9/2/12	15/2/7	13/1/10	9/3/12	9/1/13	11/2/11	10/2/12	(24)

**Fig. 3.** Box-plots of the accuracy of the SVM-based classifier with parameter sweeping across the various UCI datasets (HYPER+ did not complete on audiology).**Table 4.** Number of times an algorithm (column) achieves better/equal/worse MDL score than the baseline (row) on the test datasets. Between parentheses the number of datasets where the algorithm succeeded in extracting the top- k patterns. In boldface the best results.

	ASSO	HYPER+	PANDA ⁺ J_P	PANDA ⁺ $J_P^{1,2}$	PANDA ⁺ J_E
ASSO	(24)	0/0/21	13/0/11	9/0/15	12/0/12
HYPER+	21/0/0	(21)	21/0/0	21/0/0	21/0/0
PANDA ⁺ J_P	11/0/13	0/0/21	(24)	8/0/16	15/0/9
PANDA ⁺ $J_P^{1,2}$	15/0/9	0/0/21	16/0/8	(24)	20/0/4
PANDA ⁺ J_E	12/0/12	0/0/21	9/0/15	4/0/20	(24)

17 datasets out of 24. The difference between PANDA⁺ J_A and ASSO is statistically significant according to the two-tailed sign test with $p = 0.023$, and according to the two-tailed Wilcoxon signed-rank test with $p = 0.008$.

4 Related Work

We classify related works in three large categories: matrix decomposition based, database tiling and Minimum Description Length based.

Matrix Decomposition Based. The methods in this class aim at finding a product of matrices that describes the input data with a smallest possible amount of error. These methods include Probabilistic latent semantic indexing (PLSI) [13], Latent Dirichlet allocation (LDA) [14], Independent Component Analysis (ICA) [15], Non-negative Matrix Factorization, etc. However, ASSO was shown to outperform such methods on binary datasets.

Database Tiling. The maximum k -tiling problem introduced in [16] requires to find the set of k tiles having the largest coverage of the given database \mathcal{D} . However, this approach is not able to handle the false positives present in the data, similarly to HYPER [7]. According to [17], tiles can be hierarchical. Unlike our approach, low-density regions are considered as important as high density ones, and inclusion of tiles is preferred instead of overlapping.

Minimum Description Length Principle. In [18] a set of item sets, called *cover* or *code table*, is used to encode all the transactions in the database, meaning that every transaction is represented by the union of some item sets in the cover. The MDL principle is used to choose the best code table. The proposed KRIMP algorithm selects the item sets of the cover from a pool of candidates. In their experiments, the authors exploited the collection of all the item sets occurring in the dataset to achieve good results. However, patterns that cover a given transaction must be disjoint, thus increasing the size and redundancy of the model, and noisy occurrences are not allowed.

A similar MDL-based approach is adopted in [19], but in this case, knowledge on the data marginal distributions is assumed to be known, thus generating a different kind of patterns. In this work, assume no knowledge on the data in evaluating the extracted patterns. In [20] a novel framework is proposed for the comparison of different pattern sets. From a given pattern set, a probability distribution over \mathcal{D} is derived according to the maximum entropy principle, and then the Kullback-Leibler distance between these two distributions is used to measure the dissimilarity of two pattern sets. After comparing several algorithms including ASSO, HYPER+, KRIMP, and others, the authors conclude that HYPER+ and ASSO are the two best performing algorithms, i.e. producing the set of patterns whose probability distribution is closer to the underlying class label distribution. Note that HYPER+ and ASSO are the algorithms analyzed in this work.

5 Conclusions

In this paper we have given a deep insight into the problem of mining approximate top- k patterns from binary matrixes. Our analysis has explored ASSO, HYPER+, and PANDA+. The main contribution is a methodology to assess the various solutions at hand. In fact, the lack of standard ground truth datasets, which we can use for evaluating and comparing the quality of pattern sets extracted by the various algorithms – namely their greedy strategies, cost functions to optimize, and parameter setting – makes very hard an objective judgement. In this paper we have thus considered the accuracy of SVM classifiers, built on top of the top- k patterns used as feature sets, as a strong signal for the quality of the pattern set extracted. The experiments conducted on 24 datasets from the UCI repository show that the patterns extracted by only optimizing J_A (noise) or J_E (pattern set complexity) exhibit limited performance. The best results achieved by full parameter sweeping show that PANDA+ outperforms all the other algorithms with a statistically significant improvement.

References

1. Miettinen, P., Mielikainen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. *IEEE TKDE* **20**(10), 1348–1362 (2008)
2. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Discov.* **23**(2), 215–251 (2011)
3. Lucchese, C., Orlando, S., Perego, R.: Mining top-k patterns from binary datasets in presence of noise. In: *SDM*, pp. 165–176. SIAM (2010)
4. Lucchese, C., Orlando, S., Perego, R.: A unifying framework for mining approximate top-k binary patterns. *IEEE TKDE* **26**, 2900–2913 (2014)
5. Cheng, H., Yu, P.S., Han, J.: AC-Close: efficiently mining approximate closed itemsets by core pattern recovery. In: *Proceedings of ICDM*, pp. 839–844. IEEE Computer Society (2006)
6. Miettinen, P., Vreeken, J.: Model order selection for boolean matrix factorization. In: *Proceedings of KDD*, pp. 51–59. ACM (2011)
7. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Succinct summarization of transactional databases: an overlapped hyperrectangle scheme. In: *Proceedings of KDD*, pp. 758–766. ACM (2008)
8. Rissanen, J.: Modeling by shortest data description. *Automatica* **14**(5), 465–471 (1978)
9. Lucchese, C., Orlando, S., Perego, R.: A generative pattern model for mining binary datasets. In: *SAC*, pp. 1109–1110. ACM (2010)
10. Joachims, T.: *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, Norwell (2002)
11. Cherkassky, V., Ma, Y.: Practical selection of svm parameters and noise estimation for SVM regression. *Neural Netw.* **17**(1), 113–126 (2004)
12. Cheng, H., Yan, X., Han, J., wei Hsu, C.: Discriminative frequent pattern analysis for effective classification. In: *Proceedings of ICDE*, pp. 716–725 (2007)
13. Hofmann, T.: Probabilistic latent semantic indexing. In: *Proceedings of SIGIR*, pp. 50–57. ACM (1999)

14. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
15. Hyvärinen, A., Karhunen, J., Oja, E.: *Independent Component Analysis*. John and Wiley, Chichester (2001)
16. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Suzuki, E., Arikawa, S. (eds.) *DS 2004. LNCS (LNAI)*, vol. 3245, pp. 278–289. Springer, Heidelberg (2004)
17. Gionis, A., Mannila, H., Seppänen, J.K.: Geometric and combinatorial tiles in 0–1 data. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) *PKDD 2004. LNCS (LNAI)*, vol. 3202, pp. 173–184. Springer, Heidelberg (2004)
18. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. *Data Min. Knowl. Discov.* **23**(1), 169–214 (2011)
19. Kontonassios, K.N., Bie, T.D.: An information-theoretic approach to finding informative noisy tiles in binary databases. In: *SDM*, pp. 153–164. SIAM (2010)
20. Tatti, N., Vreeken, J.: Comparing apples and oranges. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) *ECML PKDD 2011, Part III. LNCS*, vol. 6913, pp. 398–413. Springer, Heidelberg (2011)