A unifying framework for greedy mining approximate top-k binary patterns and their evaluation

Claudio Lucchese[1], Salvatore Orlando[2], and Raffaele Perego[1]

[1] ISTI-CNR, Pisa, Italy
[2] DAIS - Università Ca' Foscari Venezia, Italy

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 algorithms, and also discuss PANDA$^+$, an enhanced version of a previously proposed algorithm, which is able to greedily optimize several cost functions generalized into a unifying formulation.
In evaluating the set of mined patterns, we aim at complementing the usual assessment methodology, which only measures the given cost function. Thus, we also evaluate how good are the models/patterns extracted in unveiling supervised knowledge on the data, i.e. the class labels of the data instances. We tested state-of-the-art algorithms and diverse cost functions on several datasets from the UCI repository. As expected, internal (cost function) and external (classification accuracy) indices of quality provide contrasting results. Nevertheless, PANDA$^+$ performs best, since the classifiers, built over the mined patterns used as record features, are in the majority of the cases 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.

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, i.e. 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 [3]. Our analysis explores the cost functions used by such greedy algorithms and focuses on the evaluation of the extracted patterns. We show that the cost functions adopted by all these algorithms share important aspects that can be generalized into a unique formulation. We thus extend PaNDa, by plugging into its framework such generalized formulation, which makes it possible to greedily mine approximate patterns according to several cost functions. We also included into this framework noise constraints inspired to [4] in order to improve its greedy heuristics. We named the resulting algorithm PaNDa$^+$.

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 more the effectiveness of the greedy strategy 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). As expected, experiments show that internal and external quality indices provide contrasting results. Nevertheless, PaNDa$^+$ is able to perform well with several cost function and quality measures. More specifically, the classifiers built over the mined patterns are in the majority of the cases the most accurate.

In summary, the contributions of this work are the following:

- a unifying formulation of several cost functions that are used by state-of-the-art algorithms to drive their greedy heuristic strategies and to evaluate the quality of the mined patterns;
- a new algorithm, named PaNDa$^+$, that thanks to the unified formulation, and by extending PaNDa [3], can deal with a variety of cost functions;
- PaNDa$^+$ also improves over [5], since it can directly optimize the Minimum Description Length (MDL) encoding cost proposed therein, and it is capable to deal with error noise thresholds [4], thus improving the accuracy of each mined pattern;
- we show the goodness of PaNDa$^+$ by means of both an unsupervised, based on an internal quality measures, and a supervised assessment, where the

accuracy of a classifier built over the pattern-based transaction features is considered a proxy for the quality of the mined patterns.

The rest of the paper is organized as follows. We first introduce some notation (Sec. 2) and formalize the approximate top-k pattern mining problem also illustrating state-of-the-art algorithms (Sec. 3). Then we discuss the framework of the PaNDa$^+$ algorithm (Sec. 4). Finally, we report on the conducted experiments (Sec. 5), and we discuss other related works (Sec. 6) and concluding remarks (Sec. 7).

2 Notation

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 it occurs, and it is represented by the pair of 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 $P_T \cdot P_I$ of the two binary vectors identifies the sub-matrix of $\mathcal{D}$ which is *approximately covered* by pattern P.

Being each pattern approximate, it may cover some *empty* occurrences $\mathcal{D}(i,j) = 0$ (*false positives*). Analogously, some occurrences $\mathcal{D}(i,j) = 1$ may not be covered by any pattern in a given pattern set Π (*false negatives*). Indeed, we can say that a set of patterns $\Pi = \{P_1, \ldots, P_{|\Pi|}\}$ approximately covers dataset $\mathcal{D}$, except for some noisy item occurrences identified by the matrix $\mathcal{N} \in \{0,1\}^{N \times M}$:

$$\mathcal{N} \;=\; \bigvee_{P \in \Pi} (P_T \cdot P_I) \quad \veebar \; \mathcal{D}. \tag{1}$$

where $\vee$ and $\veebar$ are respectively the element-wise *logical or* and *xor* operators.

Indeed, this formulation of the noise 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 $\mathcal{N}(i,j) = 1$.

3 Problem Statement and Algorithms

In general we can state the top-k pattern discovery problem as an optimization one, where the goal is to minimize a given cost function.

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 $\overline{\Pi}_k$, $|\overline{\Pi}_k| \leq k$, that minimizes the given cost function $J(\Pi_k, \mathcal{D})$:

$$\overline{\Pi}_k = \operatorname*{argmin}_{\Pi_k} J(\Pi_k, \mathcal{D}) \tag{2}$$

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.

3.1 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:

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

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.

Asso has been proved to perform better than other matrix decomposition approaches, such as principal component analysis and non-negative matrix factorization, even if these last methods were not specifically tailored for boolean matrices.

3.2 Minimizing the pattern set complexity (Hyper+)

The Hyper+ [2] algorithm works in two phases. In the first phase (corresponding to the covering algorithm Hyper [6]), 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:

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

During this first phase, the algorithm aims to cover in the best way all the items occurring in $\mathcal{D}$, without 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 Sec. 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.

3.3 Minimizing both pattern set complexity and noise (PANDA)

PANDA [3] minimizes a cost function that combines the two functions $J_A(\cdot)$ and $J_H(\cdot)$. Given $\mathcal{N}$ as defined in Eq. (1), PANDA minimizes the following cost:

$$J_P(\Pi_k, \mathcal{D}) = J_A(\Pi_k, \mathcal{D}) + J_H(\Pi_k, \mathcal{D}) = \|\mathcal{N}\| + \sum_{P \in \Pi_k} (\|P_T\| + \|P_I\|) \quad (5)$$

PANDA adopts a greedy strategy, by exploiting a two-stage heuristics to iteratively select each pattern. The problem of discovering an approximate pattern is in fact decomposed into two simpler problems: (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. A number of different heuristic strategies are proposed with PANDA. In this work we assume *correlation ordering* with 30 randomization rounds, as described in [3].

Even if we do not fix the input parameter k, PANDA can stop producing further patterns when the cost of a new pattern is larger than the corresponding noise reduction. The cost function of PANDA is indeed inspired by the MDL principle. By looking at Eq. (5) we can in fact recognize the cost of the model (the cost of pattern set) and the the cost of the dataset given the model (the cost of the noise matrix).

3.4 Minimizing the MDL encoding

In [5] the MDL principle [7] is adopted to evaluate a pattern set Π_k. According to the MDL principle, 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 $\mathcal{D}$. More formally, given a collection of pattern sets, the best Π_k is the one that minimizes the following cost function:

$$J_E(\Pi_k, \mathcal{D}) = \mathsf{enc}(\Pi_k) + \mathsf{enc}(\mathcal{D}|\Pi_k) = \sum_{P \in \Pi_k} \mathsf{enc}(P) + \mathsf{enc}(\mathcal{N}) \tag{6}$$

in which $\mathsf{enc}(\Pi_k)$ is the length, in bits, of the description of Π_k, and $\mathsf{enc}(\mathcal{D}|\Pi_k)$ is the length, in bits, of the description of the data when encoded with Π_k. Optimal codes are assumed. Note that $\mathsf{enc}(\Pi_k)$ is computed in terms of the encoding costs of all the single patterns, whereas $\mathsf{enc}(\mathcal{D}|\Pi_k)$, that is the residual information we need to derive $\mathcal{D}$ when Π_k is known, which exactly corresponds to encoding $\mathcal{N}$. Hereinafter, we refer to J_E as the Typed-XOR cost described in [5].

The authors do not provide an algorithm for mining directly the patterns that minimize J_E, but rather they use J_E to select only the best top-listed patterns returned by Asso. We show that PaNDa can be easily extended to directly optimize the MDL cost $J_E(\Pi_k, \mathcal{D})$, rather than performing a post-pruning as done in [5].

4 Generalized PaNDa$^+$ framework

In this section we discuss the improvements introduced in the PaNDa algorithm aimed at producing the generalized PaNDa$^+$ framework. First, we generalize its cost function in order to deal in a flexible way with a wider class of optimization problems. Second, we introduce noise constraints to improve the quality of the extracted patterns.

4.1 Cost functions generalization

The PaNDa original cost function J_P can be replaced without harming its greedy heuristics. Indeed, J_P can be generalized so as to include all of the aforementioned cost functions as follows:

$$J^+(\Pi_k, \mathcal{D}, \gamma_\mathcal{N}, \gamma_P, \rho) \;=\; \gamma_\mathcal{N}(\mathcal{N}) \;+\; \rho \cdot \sum_{P \in \Pi_k} \gamma_P(P) \tag{7}$$

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. (7) 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 $J_P^{\overline{\rho}}$ is a generalization of the function J_P already proposed for PaNDa, with parameter $\overline{\rho}$ that determines a different trade-off between patterns description cost and noise cost.

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 patter set complexity [6]
$J_P(\Pi_k, \mathcal{D}) = J^+(\Pi_k, \mathcal{D}, \gamma_\mathcal{N}(\mathcal{N}) = \|\mathcal{N}\|, \gamma_P = \|P_T\| + \|P_I\|, \rho = 1)$	Minimize noise and pattern set complexity [8, 3].
$J_P^{\overline{\rho}}(\Pi_k, \mathcal{D}) = J^+(\Pi_k, \mathcal{D}, \gamma_\mathcal{N}(\mathcal{N}) = \|\mathcal{N}\|, \gamma_P(P) = \|P_T\| + \|P_I\|, \rho = \overline{\rho})$	Extend J_P to leverage the trade-off between noise and pattern set complexity.
$J_E(\Pi, \mathcal{D}) = J^+(\Pi, \mathcal{D}, \gamma_\mathcal{N}(\mathcal{N}) = \mathsf{enc}(\mathcal{N}), \gamma_P(P) = \mathsf{enc}(P), \rho = 1)$	Minimize the encoding length [7] of the pattern model according to [5].

As a result, PANDA$^+$ is the first approximate top-k pattern mining algorithm directly optimizing the MDL-based cost function J_E.

4.2 Introducing noise thresholds into PANDA$^+$

Depending on the functions $\gamma_\mathcal{N}$ and γ_P, PANDA may extend a pattern by adding an item or a transaction even if a significant amount of noise is introduced. For instance, J_P allows to add a transaction to a pattern P if it contains at least half of its items plus one. This behavior may lead the algorithm to fall into a local minimum.

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 [4], aimed at bounding the maximum amount of noise generated by adding a new item or a new transaction to a pattern. Given a pattern $P = \langle P_I, P_T \rangle \in \Pi_k$, the following constraints must hold:

1. every item j (*column*) of the pattern (s.t. $P_I(j) = 1$) must be included in at least $\epsilon_c \cdot \|P_T\|$ transactions of the pattern:
$$\forall j \in [1, M], \text{ such that } P_I(j) = 1, \quad \sum_{i=1}^{N} (P_T(i) \cdot \mathcal{D}(i, j)) \;\geq\; \epsilon_c \cdot \|P_T\|$$
2. every transaction i (*row*) of the pattern (s.t. $P_T(i) = 1$) must include at least $\epsilon_r \cdot \|P_I\|$ items of the pattern:
$$\forall i \in [1, N], \text{ such that } P_T(i) = 1, \quad \sum_{j=1}^{M} (P_I(j) \cdot \mathcal{D}(i, j)) \;\geq\; \epsilon_r \cdot \|P_I\|$$

Note that PANDA$^+$ optimizing J_P extracts different patterns from PANDA due to these noise constraints.

5 Experiments

We used 24 datasets from the UCI repository[3] which have been discretized for pattern mining[4]. The main characteristics of the datasets are reported in Table 2. For all the experiments, the class labels were removed before feeding any top-k approximate item sets mining algorithm.

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

We compared the performance of HYPER+, ASSO, and PANDA$^+$, where the last one is able to minimize the different objective functions illustrated in this work. In all the experiments, we required the mining algorithms to extract a number of patterns equal to *twice* the number of classes present in the data. We believe that the number of classes is a good proxy for the actual number of interesting patterns occurring in the data. We used twice the number of classes to deal with the presence of multiple dense subgroups that characterize a single class, as found in [5]. 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

[3] http://archive.ics.uci.edu/ml/

[4] http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/

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.

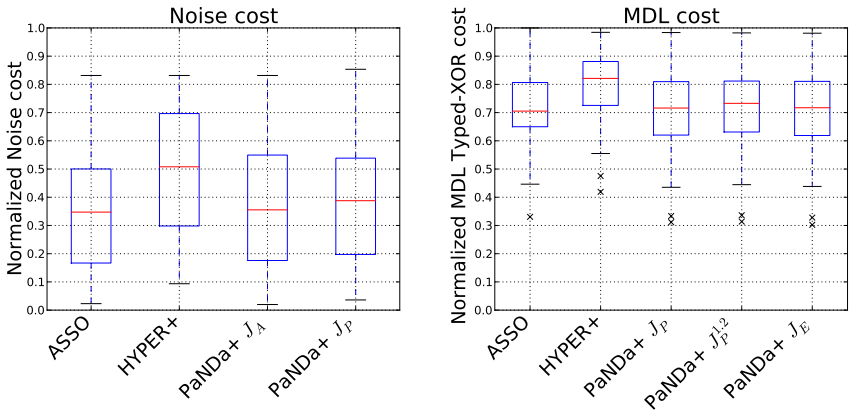
We evaluated four variants of PANDA$^+$ optimizing different cost functions: J_A, J_P, J_E, $J_P^{1.2}$. Recall that J_A measures the noise cost only, J_P adds the cost of each pattern, J_E measures the cost of an optimal MDL encoding, and $J_P^{1.2}$ mimics J_E by using a larger weight for the cost of the patters semi-perimeter. The PANDA$^+$ algorithm 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 sweept the 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.

The objective of our experimental evaluation is twofold. First, we want to evaluate the capability of the various algorithms in optimizing two specific cost functions, i.e., noise (J_A) and MDL-based encoding (J_E). These metrics capture the goodness of the patterns extracted at describing the input data. We consider this as an *unsupervised evaluation*, since no external knowledge is used to choose the best extracted patterns and evaluate their goodness. Indeed, this kind of evaluation tells us the capability of a greedy algorithm at optimizing the given cost function.

In addition to this, we propose a *supervised evaluation* of the extracted patterns. Even if the patterns are mined without any knowledge on the class labels of each transaction, we can measure how well the extracted patterns may describe, or predict, such class labels. We believe that this second kind of evaluation is more relevant, and we illustrate in the following the behavior of the various algorithms.

5.1 Unsupervised evaluation

In the first experiment, we conducted parameter sweeping for each algorithm and we selected the best outcome according to the cost function J_A, i.e. the pattern set generating the smallest amount of noise. In Tab. 3 we report the pairwise comparison of each algorithm, i.e. the number of times each algorithm better/equally/worse optimized J_A. HYPER+ was not able to run to completion on the audiology dataset, since its memory footprint exceeded the 4GBs available on the test machine. This is because it needs to process all the patterns extracted by the underlying closed frequent item set mining algorithm. Overall, HYPER+ is the worst performing in terms of noise reduction. The first phase



(a) (b)

Fig. 1. Box-plots of the normalized (a) J_A and (b) J_E costs across the various UCI datasets (HYPER+ did not complete on audiology).

of the algorithm tries to cover every occurrence in $\mathcal{D}$, and this probably leaves little degrees of freedom for the merging phase. ASSO is clearly the best performing algorithm, as it is exactly designed to optimize J_A. In Fig. 1(a) we report the normalized noise, i.e. $\|\mathcal{N}\|/\|\mathcal{D}\| = J_A(\Pi, \mathcal{D})/J_A(\emptyset, \mathcal{D})$, across the various datasets. We tested the two variants PANDA$^+$ J_A and PANDA$^+$ J_P, the first optimizing directly J_A while the second using the same cost function as PANDA. We can see that PANDA$^+$ J_A improves over PANDA$^+$ J_P and that the absolute difference between PANDA$^+$ J_A and ASSO is very low.

In Fig. 1(b) we report the normalized MDL cost values, i.e. $J_E(\Pi, \mathcal{D})/J_E(\emptyset, \mathcal{D})$. While also in this case HYPER+ has the worst performance, ASSO and the PANDA$^+$ variants achieve very similar results. We tested the PANDA$^+$ variants optimizing $J_P^{1.2}$, $J_P^{1.2}$, J_E, being more relevant for task of minimizing J_E. Tab. 3 reports the results of the pairwise comparison. First we can observe that PANDA$^+$ J_E performs much better than PANDA$^+$ J_P and PANDA$^+$ $J_P^{1.2}$, thus showing the benefits of directly optimizing the MDL cost function. On the other

Table 3. Results for J_A or J_E optimization. Number of times an algorithm (column) generated better/equal/worse results than the baseline (row) on the test datasets. The number of datasets for which the algorithm succeeded in extracting patterns is reported in the diagonal between parentheses. Best results are highlighted in boldface.

	Optimizing J_A				Optimizing J_E				
	ASSO	HYPER+	PANDA$^+$ J_P	PANDA$^+$ J_A	ASSO	HYPER+	PANDA$^+$ J_P	PANDA$^+$ $J_P^{1.2}$	PANDA$^+$ J_E
ASSO	**(24)**	0/1/22	1/0/23	2/1/21	(24)	2/0/21	**13/0/11**	9/0/15	12/0/12
HYPER+	**22/1/0**	(23)	**21/0/2**	**22/1/0**	21/0/2	(23)	**23/0/0**	**23/0/0**	**23/0/0**
PANDA$^+$ J_P	**23/0/1**	2/0/21	(24)	13/0/11	11/0/13	0/0/23	(24)	8/0/16	**15/0/9**
PANDA$^+$ J_A	**21/1/2**	0/1/22	11/0/13	(24)					
PANDA$^+$ $J_P^{1.2}$					15/0/9	0/0/23	**16/0/8**	(24)	20/0/4
PANDA$^+$ J_E					**12/0/12**	0/0/23	9/0/15	4/0/20	**(24)**

Alg	***params***
ASSO	τ
HYPER+	σ, β
PANDA$^+$	ϵ_r, ϵ_r

(a)

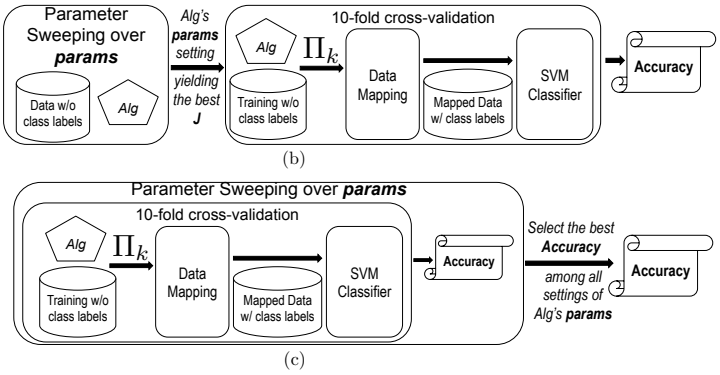


Fig. 2. Supervised evaluation: (a) algorithms' parameters, (b) unsupervised and (c) supervised selection of the algorithms' parameters.

hand, we can observe that ASSO has performance similar to PANDA$^+$ J_E, with an equal number of wins and losses. ASSO performs only slightly worse than PANDA$^+$ J_P, showing that probably both ASSO and PANDA$^+$ J_P fall into some into local minima.

5.2 Supervised evaluation

We observe the lack of common real-world benchmarks, i.e., data sets for which the most important patterns are known, and that can be used as a ground truth to evaluate and compare algorithms. We thus propose to compare different pattern mining algorithms by evaluating their ability in discovering interesting transaction features that we can use to build an accurate classifier. More specifically, we adopt two distinct approaches for our supervised evaluation. In both, since the various algorithms take as input several parameters summarized in Fig. 2(a), we exploited a parameter sweeping technique, like in [1] and [5], aimed at understanding the full potential of the algorithms taken into consideration.

In the former approach – see Fig. 2(b) – we sweep over the input parameters of the given algorithm, and select 'ex ante' the best parameters on the basis of a cost function J which is used to evaluate the quality of the mined pattern sets (*internal unsupervised measure*). 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. The transaction mapping process is detailed later on. It is worth remarking that the final accuracy of the classifier (*external supervised measure*) can confirm or contradict the evaluation outcomes based on unsupervised cost functions.

In the latter approach – see Fig. 2(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 same the

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

In both methods, a pattern set Π_k is extracted from a training set where the class labels have been removed in advance. 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, the set of transactions in the training where 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 set, which is 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 [9], 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.

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

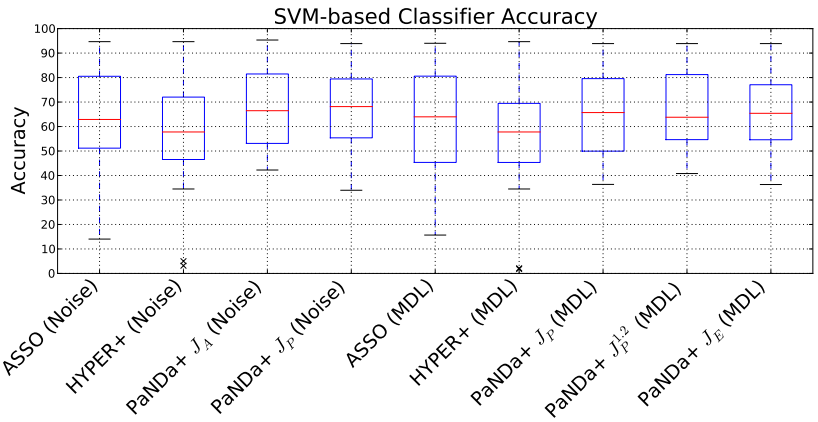


Fig. 3. Box-plots of the accuracy of the SVM-based classifier with pre-optimized algorithm parameters across the various UCI datasets (Hyper+ did not complete on audiology). Leftmost data correspond to noise-based parameter tuning and rightmost data to MDL-based parameter tuning.

Table 4. 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)

illustrated in Fig. 2(b) We notice 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 overfitting 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 Tab. 4 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. 2(c). The results of this experiment are shown in Fig. 4 and in Table 5. 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

Table 5. Number of times an algorithm (column) generated better/equal/worse patterns than the baseline (row) based on the corresponding SVM accuracy.

	Asso	Hyper+	PaNDa$^+$ J_P	PaNDa$^+$ J_A	PaNDa$^+$ $J_P^{1,2}$	PaNDa$^+$ J_E
Asso	(24)	4/3/16	13/1/10	**17/2/5**	14/2/8	14/2/8
Hyper+	16/3/4	(23)	17/1/5	**19/2/2**	16/2/5	18/1/4
PaNDa$^+$ J_P	10/1/13	5/1/17	(24)	**17/1/6**	16/1/7	13/0/11
PaNDa$^+$ J_A	5/2/17	2/2/19	6/1/17	**(24)**	6/1/17	6/0/18
PaNDa$^+$ $J_P^{1,2}$	8/2/14	5/2/16	7/1/16	**17/1/6**	(24)	14/1/9
PaNDa$^+$ J_E	8/2/14	4/1/18	11/0/13	**18/0/6**	9/1/14	(24)

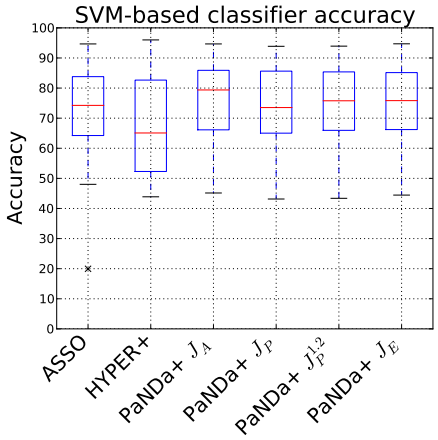


Fig. 4. Box-plots of the accuracy of the SVM-based classifier with parameter sweeping across the various UCI datasets (HYPER+ did not complete on audiology).

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 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 Wilkoxon signed-rank test with $p = 0.008$.

6 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) [12], Latent Dirichlet allocation (LDA) [13], Independent Component Analysis (ICA) [14], 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 [15] requires to find the set of k tiles, possibly overlapping, 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 [6]. According to [16], 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 [17] 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 frequent item sets, mined with a very low minimum support (till a support of a single transaction) to achieve good results. There are two significant differences from our approach. First, patterns that cover a given transaction must be disjoint, increasing the size and redundancy of the model. Second, noisy occurrences are not allowed.

A similar MDL-based approach is adopted in [18], 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 [19] 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.

7 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, three state-of-the-art algorithms that differ for the greedy strategy adopted and the cost function optimized. We have shown that all these cost functions can be unified into a unique formulation and plugged into a flexible algorithmic framework, named PANDA$^+$, that allow us to greedily mine approximate patterns according to several cost functions. Moreover, we have added to this framework two other important features: the possibility of optimizing directly the MDL encoding cost of the solution, and the possibility of dealing with noise constraints in order to improve the greedy heuristics.

Particular care has been put on the assessment of the various solutions at hand. We have started from the observation that only measuring the ability of an algorithm to minimize a given cost function captures more the effectiveness of its greedy strategy than the quality of the extracted patterns. Moreover, the lack of standard benchmarking datasets that can be used as a ground truth to evaluate and compare the quality of solutions, makes very hard to objectively judge solutions adopting different cost functions. We thus chose to place side by side two different kinds of evaluations. First we measured for all the algorithms the goodness of the patterns extracted at describing the input data by means of noise (J_A) and MDL-based encoding (J_E) costs. Second, we considered the accuracy of SVM classifiers built on top of the top-k patterns as a strong signal for the quality of the pattern set extracted.

The experiments conducted on 24 datasets from the UCI repository show that PANDA$^+$ and ASSO have similar performance in minimizing the noise and the MDL-based costs. However, when evaluating the SVM-based classifiers, the patterns extracted by optimizing J_A or J_E 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) (Oct 2008) 1348–1362
2. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Summarizing transactional databases with overlapped hyperrectangles. Data Min. Knowl. Discov. **23**(2) (Sep 2011) 215–251
3. Lucchese, C., Orlando, S., Perego, R.: Mining top-k patterns from binary datasets in presence of noise. In: SDM, SIAM (2010) 165–176
4. Cheng, H., Yu, P.S., Han, J.: AC-Close: Efficiently mining approximate closed itemsets by core pattern recovery. In: Proc. of ICDM, IEEE (2006) 839–844
5. Miettinen, P., Vreeken, J.: Model order selection for boolean matrix factorization. In: Proc. of KDD, ACM (2011) 51–59
6. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Succinct summarization of transactional databases: an overlapped hyperrectangle scheme. In: Proc. of KDD, ACM (2008) 758–766
7. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5) (1978) 465–471
8. Lucchese, C., Orlando, S., Perego, R.: A generative pattern model for mining binary datasets. In: SAC, ACM (2010) 1109–1110
9. Cheng, H., Yan, X., Han, J., wei Hsu, C.: Discriminative frequent pattern analysis for effective classification. In: Proc. of ICDE, ACM (2007) 716–725
10. Joachims, T.: Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Kluwer (2002)
11. Cherkassky, V., Ma, Y.: Practical selection of svm parameters and noise estimation for svm regression. Neural Netw. **17**(1) (Jan 2004) 113–126
12. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of SIGIR, ACM (1999) 50–57
13. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3** (2003) 993–1022
14. Hyvärinen, A., Karhunen, J., Oja, E.: Independent component analysis. John and Wiley (2001)
15. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. Discovery Science (2004) 278–289
16. Gionis, A., Mannila, H., Seppänen, J.: Geometric and combinatorial tiles in 0-1 data. In: Proc. of PKDD (2004) 173–184
17. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Min. Knowl. Discov. **23**(1) (2011) 169–214
18. Kontonasios, K.N., Bie, T.D.: An information-theoretic approach to finding informative noisy tiles in binary databases. In: Proc. of SDM, SIAM (2010) 153–164
19. Tatti, N., Vreeken, J.: Comparing apples and oranges - measuring differences between data mining results. In: ECML-PKDD (2011) 398–413