The Partial Weighted Set Cover Problem with Applications to Outlier Detection and Clustering

Sebastian Bothe[1] and Tamás Horváth[2,1]

[1]Fraunhofer IAIS, Schloss Birlinghoven, 53754 St. Augustin, Germany
[2]Dept. of Computer Science, University of Bonn, Germany
{firstname.lastname}@iais.fraunhofer.de

Abstract. We define the partial weighted set cover problem, a generic combinatorial optimization problem, that includes some classical data mining problems as special cases. We prove that it is computationally intractable and give a local search algorithm for this problem. As application examples, we then show how to translate clustering and outlier detection problems into this generic problem. Our experiments on synthetic and real-world datasets indicate that the quality of the solution produced by the generic local search algorithm is comparable to that obtained by state-of-the-art clustering and outlier detection algorithms.

1 Introduction

Let $\mathcal{S}$ be a set system over a finite ground set such that for all $X \in \mathcal{S}$ and for all $x \in X$, x is associated with a real-valued *relative weight* with respect to X. That is, x can have as many different relative weights as many sets it belongs to. In addition to the relative weights, $\mathcal{S}$ is equipped with two real-valued set functions measuring the *weight* and the *generality* of the elements of $\mathcal{S}$. The weight of a set in $\mathcal{S}$ is defined as the sum of the relative weights of its elements. In this paper we consider the following *partial weighted set cover problem*: *Given* a *weighted* set system $\mathcal{S}$ over some finite ground set as described above, a positive integer k, and a generality function on $\mathcal{S}$, *find* k elements of $\mathcal{S}$ maximizing a utility function composed of a reward and two penalty terms. While the reward term gives preference to sets with the highest weights, the penalty terms discourage the selection of overlapping sets, as well as sets with high generality. Intuitively, our aim is to select k sets that cover as many elements of the ground set as possible, subject to the constraints that the sets have as small pairwise overlap as possible and are as specific as possible.

We show that there is a polynomial reduction from the decision version of the classical set cover problem, implying that the partial set cover problem is NP-hard. To overcome the computational limitation of finding the optimal solution, we resort to a generic local search algorithm. In each iteration of the algorithm, the current k sets are updated by applying the two steps below:

(i) In the first step we proceed as follows: For each of the k sets, we select one of its supersets from $\mathcal{S}$ that, together with the remaining $k-1$ sets, maximizes

the utility function and fulfills the following conditions: The new utility value is greater than the old one and the new candidate set does not cover any new element already contained by any of the other $k-1$ sets. If all these conditions are satisfied, we replace the set by this maximal superset selected.

(ii) In the second step we then select $O(\log k)$ sets from the k sets obtained after step (i) uniformly and independently at random. For each set selected, we replace it either with one of its direct subsets (i.e., maximal proper subsets) or direct supersets (i.e., minimal proper supersets) in $\mathcal{S}$ selected uniformly at random. If the new k sets obtained in this way have a better utility or they have the same utility and the outcome of a biased coin flip is head, we keep the new k sets; otherwise we use the ones obtained after step (i).

Regarding (i), note that by construction, if at least one of the k sets will be changed after this step, we have a strict increase in the utility. Furthermore, this step is greedy, as the k sets are processed separately. Regarding (ii), this step may not result in strict increase in the utility with a certain probability specified by the user.

In the second part of the paper, we consider set systems $\mathcal{S}$ over some finite set $S \subseteq \mathbb{R}^d$. More precisely, a subset of S belongs to $\mathcal{S}$ if and only if it can be realized by a d-dimensional ball of radius r around a point $p \in S$, where r is the element of a finite geometric progression for some scale factor specified by the user. The *relative weights* of the elements in a ball are defined by a function monotonically decreasing in their distances to the center. If a set can be realized by more than one ball, we take the ball with the highest weight. Using these weights, the *reward* term for a family of k sets in $\mathcal{S}$ is defined as the sum of their weights. Defining the generality of a set by the radii of the ball representing it, the *penalty* terms are given by the sum of the radii of the k balls and by the sum of all relative weights of the elements except for the highest one.

Using the set system with the utility function as well as the generic algorithm sketched above, we arrive at a combinatorial optimization problem and at an algorithm solving it that are highly relevant e.g. for (soft) clustering and outlier detection problems. In fact, as we experimentally demonstrate on synthetic and real-world data, our empirical results on these two particular data mining problems are comparable to those obtained by state-of-the-art algorithms. This is especially remarkable because, as we will discuss in detail, the balls inducing the set system are split into concentric annuli and two points belonging to the same ball have the same relative weight if and only if they belong to the same annulus. According to our experimental results, this type of *distance discretization* does not seem to have a strong impact on the quality of the output.

One of the main advantages of our approach is its *generality*: After the domain *dependent* step of specifying the set system and the utility function for a broad class of problems, we can solve the problem with a domain *independent* algorithm. Further potential advantages will be discussed in the last section.

The rest of the paper is organized as follows. In Section 2 we formally define the partial weighted set cover problem, show its computational intractability, and give a local search algorithm for approximating its solution. In Section 3

we adapt our generic approach to set systems defined by d-balls and empirically demonstrate the usefulness of our method on clustering and outlier detection problems. Finally, in Section 4 we discuss our results and present some interesting problems for future work.

2 The Partial Weighted Set Cover Problem

In this section we define the partial weighted set cover problem, show that it is computationally intractable, and present a generic local search algorithm for this problem.

Let S be some finite set and $\mathcal{S} \subseteq 2^S$ be a set system over S. We assume that there is a function $w_X : X \to R_{\geq 0}$ for all $X \in \mathcal{S}$, where $R_{\geq 0}$ denotes the set of non-negative real numbers. Thus, for all sets X in $\mathcal{S}$ and for all $x \in X$, $w_X(x)$ defines the *relative weight* of x with respect to X. In addition to the relative weights on the elements of X, we assume that there are two set functions W and G, both mapping $\mathcal{S}$ to $\mathbb{R}_{\geq 0}$. While

$$W(X) = \sum_{x \in X} w_X(x)$$

defines the *weight* for all $X \in \mathcal{S}$, $G(X)$ specifies the *generality* of X. In the applications we consider, $G(X)$ will be defined by the *size* of X, for some appropriate notion of size depending on the particular problem at hand. Using the definitions above we are ready to define the following combinatorial optimization problem considered in this work:

The Partial Weighted Set Cover (PWSC) Problem: Given a weighted set system $\mathcal{S}$ over some finite set as defined above, a positive integer k, and a function $P_O : \mathcal{S}^k \to R_{\geq 0}$, *find*

$$\operatorname*{argmax}_{\mathcal{S}_k \subseteq \mathcal{S}, |\mathcal{S}_k| = k} \mathcal{U}(\mathcal{S}_k) \ ,$$

with

$$\mathcal{U}(\mathcal{S}_k) = \sum_{X \in \mathcal{S}_k} W(X) - \sum_{X \in \mathcal{S}_k} G(X) - P_O(\mathcal{S}_k) \tag{1}$$

In the definition above, P_O is used to penalize the elements of S that are covered by more than one set from $\mathcal{S}_k$. Thus, our goal is to select k sets from $\mathcal{S}$ with maximum weights subject to the constraints that the sets must be as specific as possible and the overlap amongst the k sets must be as small as possible. There are many problems that can be regarded as special cases of the PWSC problem. As an example, in the next section we show that soft clustering and outlier detection problems can be viewed as such special cases, allowing these classical data mining problems to be translated into combinatorial optimization problems.

Before giving our algorithm for the PWSC problem, we first discuss its complexity. Not surprisingly, the PWSC problem is computationally intractable. This is stated in the proposition below.

Proposition 1. *The PWSC problem is NP-hard.*

Proof. We prove the claim by using the following polynomial reduction from the decision version of the set cover problem[1]. Let $\mathcal{S}$ be a set system over some finite ground set S and define $w_X(x) = 1$, $W(X) = |X|$, $G(X) = 0$, and

$$P_O(\mathcal{S}_k) = \sum_{Y \in \mathcal{S}_k} |Y| - \left| \bigcup_{Y \in \mathcal{S}_k} Y \right|$$

for all $X \in \mathcal{S}$, for all $x \in X$, and for all $\mathcal{S}_k \subseteq \mathcal{S}$ with $|\mathcal{S}_k| = k$. For the output $\mathcal{S}_k$ of the PWSC problem for the weighted set system constructed we have that $\mathcal{U}(\mathcal{S}_k) = |S|$ if and only if there exist k sets in $\mathcal{S}_k$ that cover S. □

To overcome the computational limitation stated in Proposition 1 above, we resort to a local search algorithm for finding some approximate solution of the PWSC problem (see Alg. 1). The parameters of the algorithm are a weighted set system $\mathcal{S}$ over some finite set S, a positive integer k, and a set function P_O on $\mathcal{S}_k$. In lines 1 and 2 of the main algorithm, we greedily select k sets maximizing the utility function given in (1). Then, until some termination condition holds, we iteratively call two update functions (lines 3–6).

The input to the first update function (UPDATE_A) is a family $\mathcal{S}_k \subseteq \mathcal{S}$ with $\mathcal{S}_k = \{S_1, \ldots, S_k\}$. For every $i = 1, \ldots, k$, it selects a proper superset S_i' of S_i from $\mathcal{S}$ such that the replacement of S_i by S_i' in $\mathcal{S}_k$ maximizes the utility function. If the utility of the k sets after the replacement is greater than that of $\mathcal{S}_k$, we take the new configuration and continue the process with the next set (lines 2–4 of UPDATE_A). Note that this function is greedy, as it updates the k sets in $\mathcal{S}_k$ separately.

In function UPDATE_B we select $O(\log k)$ sets uniformly at random from the input family $\mathcal{S}_k = \{S_1, \ldots, S_k\}$ (line 3 of UPDATE_B) and try to shrink/enlarge them without any decrease in the utility. More precisely, for each set $S_i \in \mathcal{S}_k$ selected, we first calculate the family $\mathcal{F}_1$ of maximal proper subsets (resp. the family $\mathcal{F}_2$ of minimal proper supersets) of S_i in $\mathcal{S}$ that minimize the L_1-distance of the relative weights of the elements belonging to the intersection (line 5 resp. line 6). We then select a set from $\mathcal{F}_1 \cup \mathcal{F}_2$ uniformly at random (line 7) and replace S_i in $\mathcal{S}_k$ by the set selected. After having processed all $O(\log k)$ sets in this way, we compare the utility of the new k sets with that of the input ones. We keep the new configuration if its utility is greater or it has the same utility and the outcome of a biased coin flip is HEAD (lines 9–11). The rationale behind the definition of $\mathcal{F}_1$ and $\mathcal{F}_2$ is that we would like to avoid big steps during local search.

[1] The decision version of the set cover problem is defined as follows: *Given* a set system $\mathcal{S}$ over some finite set S and a positive integer k, *decide* whether there exist k sets in $\mathcal{S}$ that cover S. This problem is known to be NP-complete.

Algorithm 1 MAIN

Parameters: weighted set system $\mathcal{S}$ over some finite set S, $k \in \mathbb{N}$, and $P_O : \mathcal{S}^k \to R_{\geq 0}$

set $\mathcal{S}_0 = \emptyset$
for $i \in [k]$ **do** $\mathcal{S}_i = \mathcal{S}_{i-1} \cup \{ \operatorname{argmax}_{X \in \mathcal{S} \setminus \mathcal{S}_{i-1}} \mathcal{U}(\mathcal{S}_{i-1} \cup \{X\}) \}$
repeat
    $\mathcal{S}_k = \texttt{Update\_A}(\mathcal{S}_k)$
    $\mathcal{S}_k = \texttt{Update\_B}(\mathcal{S}_k)$
until some termination condition holds
return $\mathcal{S}_k$

UPDATE_A($\mathcal{S}_k$) with $\mathcal{S}_k = \{S_1, \ldots, S_k\}$:
$U_{\max} = \mathcal{U}(\mathcal{S}_k)$
for $i \in [k]$ **do**
    $S_i' = \operatorname{argmax}_{X \in \mathcal{S}, X \supsetneq S_i} \mathcal{U}(\mathcal{S}_k \setminus \{S_i\} \cup \{X\})$
    if $\mathcal{U}(\mathcal{S}_k \setminus \{S_i\} \cup \{S_i'\}) > U_{\max}$ **then** $\mathcal{S}_k = \mathcal{S}_k \setminus \{S_i\} \cup \{S_i'\}$, $U_{\max} = \mathcal{U}(\mathcal{S}_k)$
return $\mathcal{S}_k$

UPDATE_B($\mathcal{S}_k$) with $\mathcal{S}_k = \{S_1, \ldots, S_k\}$:
$\mathcal{S}_k' = \mathcal{S}_k$
for all $i \in [k]$ **do**
    flip a biased coin with $\mathbf{Pr}(\text{HEAD}) = \frac{\log(k)}{k}$
    if the outcome is HEAD **then**
        $\mathcal{F}_1 = \{X \in \mathcal{F}_1' : \nexists Y \in \mathcal{F}_1' \text{ with } Y \supsetneq X\}$ with

$$\mathcal{F}_1' = \{Z \in \mathcal{S} : Z \subsetneq S_i \text{ and } \sum_{x \in Z} |w_Z(x) - w_{S_i}(x)| \text{ is minimum}\}$$

        $\mathcal{F}_2 = \{X \in \mathcal{F}_2' : \nexists Y \in \mathcal{F}_2' \text{ with } Y \subsetneq X\}$ with

$$\mathcal{F}_2' = \{Z \in \mathcal{S} : Z \supsetneq S_i \text{ and } \sum_{x \in S_i} |w_Z(x) - w_{S_i}(x)| \text{ is minimum}\}$$

        select a set S_i' uniformly at random from $\mathcal{F}_1 \cup \mathcal{F}_2$
        set $\mathcal{S}_k' = \mathcal{S}_k \setminus \{S_i\} \cup \{S_i'\}$
if $\mathcal{U}(\mathcal{S}_k') > \mathcal{U}(\mathcal{S})$ **then** $\mathcal{S}_k = \mathcal{S}_k'$
else if $\mathcal{U}(\mathcal{S}_k') = \mathcal{U}(\mathcal{S})$ **then**
    set $\mathcal{S}_k = \mathcal{S}_k'$ if the outcome of a biased coin flip is HEAD
return $\mathcal{S}_k$

3 Applications

To demonstrate the practical usefulness of our approach, in this section we present the applications of the PWSC problem to two classical problems of data mining: To clustering and outlier detection. In case of clustering, the task is to identify subsets of observations (i.e., *clusters*) minimizing the inter-cluster

distances (i.e., the distance between instances within the same subset) and maximizing the inter-cluster distances (i.e., the distance between clusters). Note that this informal definition applies to soft clustering as well. Regarding outlier detection, we use the following definition: "*An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism*" [1]. Thus, the goal of outlier detection is to distinguish the set of outlier observations from that of the inlier ones. Similarly for example to DBSCAN [2], we reduce the outlier detection problem to clustering; all instances not belonging to any of the clusters are regarded as outliers. In order to model the two problems above by the PWSC problem and to apply Algorithm 1, we need to construct an appropriate weighted set system and define the reward function P_O.

3.1 The PWSC Problem for Clustering and Outlier Detection

Both clustering and outlier detection use a concept of similarity between observations. We consider the case that the observations form a finite set $S \subseteq \mathbb{R}^d$ for some d and that the similarity between observations is defined by some metric D on $\mathbb{R}^d$. For each point $P \in S$, we consider a set of d-balls around P for all radii defined by the elements of a finite geometric progression for some scale factor. The set system $\mathcal{S}$ over S is then defined by the family of subsets of S, each covered by such a ball centered around a point P for some $P \in S$. We will refer to the resulting PWSC as *BallCover*.

More precisely, we assume without loss of generality that the smallest distance between two different points in S is 1, i.e.,

$$\min_{P_1,P_2 \in S, P_1 \neq P_2} D(P_1, P_2) = 1 \ .$$

Given some positive real number θ defining the scale factor $1 + \theta$, we define $L \in \mathbb{N}$ by

$$L = \left\lceil \log_{1+\theta} R \right\rceil \ ,$$

where $R = \max_{P_1,P_2 \in S} D(P_1, P_2)$. Thus,

$$D(P, Q) \leq (1 + \theta)^L$$

for all $P, Q \in S$. For all $P \in S$, we determine an integer $0 \leq L_P \leq L$ that gives an upper bound on the set of balls of center P; the algorithm calculating L_P is discussed below. Using these concepts, for S and θ above we define the set system $\mathcal{S}$ over S by

$$\mathcal{S} = \{S_{P,l} : P \in S, 0 \leq l \leq L_P\}$$

with

$$S_{P,l} = \{P' \in S : D(P, P') < (1 + \theta)^l\} \ .$$

The definitions imply that $S \in \mathcal{S}$ and that $S_{P,L} = S$ for all $P \in S$.

For all $P \in S$ and for all $l = 0, 1, \ldots, L_P$, we define the relative weights of the instances in $S_{P,l}$ by

$$w_{S_{P,l}} : Q \mapsto \frac{1}{i+1}$$

for all $Q \in S_{P,i} \setminus S_{P,i-1}$ and for all $i = 0, 1, \ldots, l$, where $S_{P,-1} = \emptyset$. That is, $S_{P,l}$ is partitioned into $l+1$ annuli, where the 0th annulus is the point P, and the relative weight of a point Q belonging to the ith annulus is $1/(i+1)$, i.e., it is inversely proportional to the distance of the annulus from P. In this way, we disregard the exact distance for any two points belonging to the same annulus in $S_{P,l}$.

We now specify the generality (G) and the penalty function (P_O) for $\mathcal{S}$. Regarding the generality, we define it by

$$G(S_{P,l}) = \lambda(1+\theta)^l$$

for all $P \in S$ and for all $l = 0, 1, \ldots, L_P$, where $\lambda \geq 0$ is some user specified parameter. Regarding P_O, let $\mathcal{S}'$ be a subset of $\mathcal{S}$ and let $S' \subseteq S$ be the set of points contained by at least two sets in $\mathcal{S}'$. Then $P_O(\mathcal{S}_k)$ is defined by

$$P_O(\mathcal{S}_k) = \sum_{x \in S'} \left(\sum_{X \in \mathcal{S}'} w_X(x) - \max_{X \in \mathcal{S}'} w_X(x) \right) .$$

That is, according to the definition of the utility function in (1), we subtract all relative weights of a point x covered by more than one set, except for the highest one. Finally we note that if a subset of S has more than one ball representation, we take the ball (and the corresponding weights) that has the highest weight.

It remains to discuss the determination of the upper bound L_P for a point $P \in S$. Since balls having large annuli of low density are poor choices for clustering, we need to disregard them as candidates. To find the maximum ball around P that contains no annuli of low density, we observe the change in density as a function of the radius while growing the ball. The density is measured by the number of instances covered relative to the ball's radius. Accordingly, we sort the instances by their distance to P. The position in this list then provides the number of instances covered at the respective distance, giving rise to a monotone function sampled at finitely many non-equidistant positions. Our goal is to estimate the first plateau of this unknown continuous density function. To achieve this, we first interpolate the function value at equidistant positions using nearest neighbor interpolation. We then approximate the first derivative by folding the interpolated signal using a Sobel kern. Finally, we smooth the result and determine the first position that is numerically a zero point.This position defines the maximal radius L_P we consider for the ball around P. Due to space limitations we omit the formal definition of this algorithm.

3.2 Experiments

To demonstrate the usefulness of our *BallCover* approach to the tasks of outlier detection and clustering, we have conducted a series of experiments. We have

compared our results achieved on synthetic and real-world data from the UCI machine learning repository[3] to those obtained by state-of-the-art algorithms. The problems of outlier detection and clustering have been studied extensively in the past. As a result, a wide range of different concepts and algorithms have been proposed. For instance, there are outlier detection algorithms using information theoretical criteria, spectral decomposition, clustering, proximity, or density. For a recent overview of outlier detection algorithms, the reader is referred to [4].

An exhaustive comparison to all relevant outlier detection and clustering algorithms is beyond the scope of this work. Therefore, we focus only on algorithms using density criterion to identify outliers or clusters, as these are the most similar methods to the algorithm proposed in this paper. More precisely, we consider the following algorithms:

Local outlier factor (LOF) [5] The algorithm identifies outliers by comparing the density of a point with that of its surrounding points. The size of the surrounding neighborhood is specified by the user supplied parameter $MinPts$. Within the $MinPts$ neighborhood around a point, the local outlier factor (LOF) is calculated as the average density of all points in the neighborhood normalized by the points own densities. Points with density much lower than their neighbors produce a high LOF value and are considered outliers. For our experiments we use the implementation available with the ELKI [6] toolkit. To identify the set of outliers, we order the instances according to their LOF value and select the top n instances, where n is the true number of outliers. In our experiments on synthetic data we set the $MinPts$ parameter (i.e., the number of instances in a cluster) to 1000; other choices $(10, 20, 100)$ of this parameter have not lead to better results. For the UCI datasets we follow [7] and set $MinPts = 10$.

Support vector novelty detection (SVND) [8] is an extension of support vector machines (SVM) to the case of unlabeled data. In SVM the maximal separating hyperplane is determined by the location of instances with different labels in the feature space. However, there are no labels in SVND. Therefore, the goal is to find a simple subset of instances such that the probability of an instance falling into this set meets a probability threshold parameter ν. The boundary of the set is expressed in terms of a kernel expansion and its complexity is controlled by empirical risk minimization. The algorithm takes the probability threshold ν and the kernel as parameters. For our experiments we set ν to the true fraction of outliers and use the Gaussian kernel. The Gaussian kernel itself requires the specification of the variance σ, which we set to the median distance of all points in the dataset, following the recommendation in [9]. In our experiments we use the open source implementation libsvm [10].

DBSCAN [2] constructs a clustering by expanding clusters around dense points, called core points. A point is dense if it has at least $MinPts$ neighbors in a distance at maximum ϵ. All points in the neighborhood are recursively added to the cluster as long as they have $MinPts$ neighbors. Points that do not belong to any cluster are considered as outliers. For our experiments we use

the implementation available from sklearn [11], i.e., we leave the parameter $MinPts$ at the default choice of 10 and set ϵ to the medium of pairwise distances between two points in the data.

Isolation Forest [7] constructs an ensemble of trees, by randomly choosing attributes and splits at each inner node of a tree. The tree growing stops once each instance is isolated in a single leaf or the tree exceeds a height threshold. Each tree is grown on a random sample of data and the tree height is restricted to the height of a binary tree with number of leaves equal to the sample size. Each instance is scored by the expected average path length between the tree root and the node containing the instance. Instances with a short path length are isolated earlier from the rest of the data and are considered as outliers. For our experiments we use the implementation available in sklearn [11] and keep all parameters to their defaults (ensemble size 100, data sample size of 256). The authors propose instances with path length measure much smaller than 0.5 to be considered as inliers. We therefore use this threshold to determine the prediction of outliers.

For our empirical evaluation, we need datasets with known ground truth, i.e., for which we know which instances are outliers within the data. Therefore, we create synthetic datasets and use publicly available classification datasets from the UCI repository for comparison. For the synthetic data, we place k-Gaussians uniformly at random in a hypercube, each with a random diagonal co-variance matrix. The centers may be generated close to each other and the resulting distributions may have arbitrary overlap. We draw the same number of instances from each Gaussian and add uniform noise over the hypercube extended by the largest 3σ. The parameters for the data we generated here are as follows: the center coordinates of the Gaussians are drawn uniformly at random from the interval $[0, 20]$ and the variances σ^2 from $[0, 1]$. We generated ten 2-dimensional datasets with four Gaussians, having 1000 samples each and added 20% uniform noise samples as outliers. Regarding the real-world data from the UCI repository, we follow the transformation used in [7] to construct an outlier detection task from the corresponding multi-class classification task, i.e., we consider classes 3,4,5,7,8,9,14,15 as outliers for the *arrhythmia* set, classes 1,2 for *annthyroid*. The *pima* and *ionosphere* datasets are binary classification tasks, the minority class are considered as outliers. We further take *wilt* and a random sample of the *adult* dataset into account using the same proposition. Our selection includes datasets of small up to large size (350-7000) and of low and high dimensions (5-274), to cover a broad range of application scenarios.

We tested all algorithms described above by applying them to the full datasets including the outliers during the training stage. The ground truth labels are not presented to the learner; they are only used in the calculation of the performance measure. We use F_1-score, with the normal instances as positive class, to access the quality, as it accounts for the class imbalance. For our algorithm, we use the true number of balls with the synthetic datasets and report results for different choices of k on the UCI datasets. Further, we fix the parameters $\theta = 0.1$ and $\lambda = 0.05$ for all experiments. The results achieved for the different combinations

	BallCover						
UCI Dataset	k=2	k=8	k=20	LOF	SVND	DBSCAN	IFOREST
adult (sample)	0.79	0.86	0.87	0.80	0.79	0.86	0.87
arrythmia	0.93	0.92	0.87	0.88	0.57	0.89	0.92
annthyroid	0.52	0.74	0.68	0.93	0.93	0.95	0.96
ionosphere	0.70	0.84	0.68	0.91	0.88	0.83	0.87
pima	0.76	0.71	0.67	0.64	0.72	0.79	0.79
wilt	0.92	0.86	0.88	0.95	0.94	0.97	0.93
Synth. Dataset		k=4					
4-Gauss (mean)		0.95		0.98	0.95	0.97	0.97

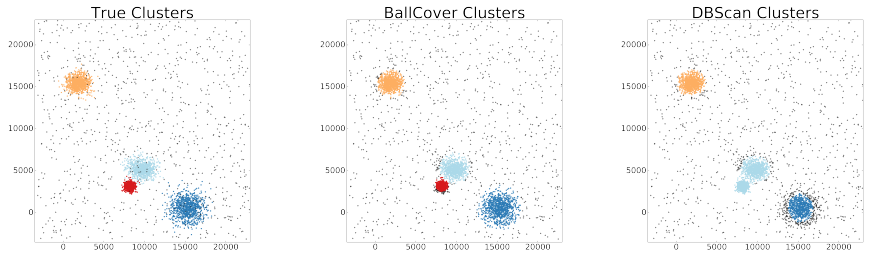
Table 1. Summary of F_1 scores of outlier detection algorithms on UCI machine learning tasks and synthetic data sets. For the ten synthetic 2-dimensional Gaussian mixtures datasets we report the mean value of F_1 scores.

of the datasets and algorithms are summarized in Table 1. As we can see, the performance of our approach depends on the choice of k, but in most cases, there is a choice that matches the quality of the competitors (*adult*, *ionosphere*, *pima*, *wilt*, synthetic data). Only for the *annthyroid* dataset, the performance of our algorithm is clearly worse than that of any reference. In turn, we can report a slightly better performance as the best competitor for the *arrhythmia* dataset.

For the clustering application we use the synthetic Gaussian datasets. Each of the Gaussians forms a separate cluster and each instance is labeled according to it. The uniform noise represents an additional component and is identified as another cluster id. For comparison, we use the DBSCAN and K-Means clustering algorithms as reference. We follow the same parameter selection scheme for DBSCAN as used within our outlier experiments. In contrast to DBSCAN, the K-Means algorithm creates a complete partition of all instances, especially without excluding any noise. Therefore we consider different choices of k setting it at least to the true number of Gaussian plus one additional for the noise. We treat the cluster id as target of a multiclass classification problem and assess the performance in terms of the weighted average over the F_1 scores for each of the classes. To match the cluster id used by the algorithm with that of the ground truth, we apply the following mapping: For the K-Means and DBSCAN algorithms we assign to each cluster the ground truth label of the majority vote of the instances in that cluster. For our algorithm, we use the center points to select the ground truth label for each ball. The results indicate, that our algorithm performs better than K-Means for small choices of k. Increasing k leads to results comparable to our approach for most datasets. There are some cases where K-Means performs better and some where our approach is slightly better (c.f. Table 2). Further, across all sets used in this experiment, the performance of our algorithm competes with the results of DBSCAN; in some cases, our algorithm performs much better. A closer look at those cases reveals that our algorithm has an advantage if the centers of the Gaussians are very close to each other, resulting in large overlaps. In such cases, those clusters are merged by DBSCAN and our algorithm is able to separate the corresponding data instances. An ex-

			K-Means		
Dataset ID	BallCover	DBSCAN	k=5	k=8	k=16
05d...923	0.74	0.71	0.62	0.84	0.87
0c8...9d2	0.96	0.66	0.60	0.89	0.94
3ac...309	0.97	0.98	0.80	0.89	0.94
420...c47	0.64	0.68	0.63	0.67	0.69
718...f80	0.78	0.68	0.62	0.87	0.89
984...3e4	0.96	0.87	0.60	0.90	0.95
a63...013	0.69	0.69	0.60	0.66	0.88
b1a...3af	0.96	0.98	0.81	0.89	0.93
d2d...0a5	0.95	0.70	0.59	0.87	0.93
d7b...9e2	0.96	0.97	0.62	0.91	0.95

Table 2. Weighted average F_1 scores unsupervised reconstruction of class structure.



(a) (b) (c)

Fig. 1. Cluster structure of synthetic dataset with overlapping Gaussians. Colors indicate cluster memberships, noise cluster is black. True clusters are on the left (a), cluster structure found by BallCover in the middle (b), for DBSCAN on the right (c).

ample of this situation is depicted in Figure 1. The true association of points to clusters is at the left. Note the two overlapping Gaussians at the middle bottom region. Our algorithm seeks dense balls with low overlap and small radii and can detect the structure correctly. For DBSCAN, the two Gaussians are too close to each other and joined into one cluster.

4 Discussion

The approach presented in this paper is a first step towards a systematic study of its applications to other data mining/machine learning problems. The advantage of translating such problems into the partial weighted set system problem is that it allows for the application of techniques developed for combinatorial optimization problems. For example, one might transform the underlying problem into set systems of some advantageous structural properties (e.g., into matroids or greedoids) that can be utilized by the algorithm. In this way, new tractable subclasses of the problem could be identified. Another interesting research direction

is the restriction of the utility function to set function classes of advantageous algorithmic properties.

The distance discretization used in the applications considered in this work raises the question whether our approach can be combined with state-of-the-art techniques improving the speed, such as, for example, with locality sensitive hashing.

Our utility function includes two penalty terms. One of them is concerned with the generality of the elements in the set system. It is an interesting question whether and if so, how can we control the complexity of the output via these terms and the parameter k. Finally, we are going to investigate how to extend our method to an interactive one, in which the set system and the utility function are automatically adapted according to the feedback of an expert (c.f. [12]).

Acknowledgments This research was supported by the EU FP7-ICT-2013-11 project under grant 619491 (FERARI).

References

1. Hawkins, D.: Identification of Outliers. Monographs on applied probability and statistics. Chapman and Hall (1980)
2. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of 2nd International Conference on Knowledge Discovery and. (1996) 226–231
3. Lichman, M.: UCI machine learning repository (2013)
4. Aggarwal, C.: Outlier Analysis. Springer New York (2013)
5. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. SIGMOD Rec. **29**(2) (May 2000) 93–104
6. Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K.A., Zimek, A.: A framework for clustering uncertain data. PVLDB **8**(12) (2015) 1976–1979
7. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. (Dec 2008) 413–422
8. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7) (July 2001) 1443–1471
9. Caputo, B., Sim, K., Furesjo, F., Smola, A.: Appearance-based object recognition using svms: Which kernel should i use? Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler **2002** (2002)
10. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2** (2011) 27:1–27:27
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830
12. Boley, M., Mampaey, M., Kang, B., Tokmakov, P., Wrobel, S.: One click mining-interactive local pattern discovery through implicit preference and performance learning. In: KDD 2013 Workshop on Interactive Data Exploration and Analytics (IDEA). (2013)