The Odd One Out: Identifying and Characterising Anomalies

Koen Smets Jilles Vreeken

Department of Mathematics and Computer Science
Universiteit Antwerpen, Belgium
{koen.smets,jilles.vreeken}@ua.ac.be

Abstract

In many situations there exists an abundance of positive examples, but only a handful of negatives. In this paper we show how in binary or transaction data such rare cases can be identified and characterised.

Our approach uses the Minimum Description Length principle to decide whether an instance is drawn from the training distribution or not. By using frequent itemsets to construct this compressor, we can easily and thoroughly characterise the decisions, and explain what changes in an example would lead to a different verdict. Furthermore, we give a technique through which, given only a few negative examples, the decision landscape and optimal boundary can be predicted—making the approach parameter-free.

Experimentation on benchmark and real data shows our method provides very high classification accuracy, thorough and insightful characterisation of decisions, predicts the decision landscape reliably, and can pinpoint observation errors. Moreover, a case study on real MCADD data shows we provide an interpretable approach with state-of-the-art performance for screening newborn babies for rare diseases.

1 Introduction

In many situations there is an abundance of samples for the positive case, but none, or only a handful, for the negative case. Examples of such situations include, intrusion detection [10,13], screening for rare diseases [1, 11], monitoring in health care [6] and industry [22], fraud detection [2,12], as well as predicting the lethality of chemical structures in pharmaceutics [5,14]. In all these cases it is either very expensive, dangerous, or virtually impossible to acquire (many) negative examples. This means that standard classification techniques cannot be applied, as there is not enough training data for each of the classes. Since there are only enough examples for one class, this problem setting is typically known as one-class classification, but for obvious reasons it can also be regarded as outlier detection. The goal is simple: given sufficient training data for only the positive class, reliably detect the rare negative cases in unseen data. That is, to point the odd ones out.

Identification alone is not enough, however: explanations are also very important. This goes for classification in general, but in the one-class setup descriptions are especially important; we are deciding over rare events with possibly far-reaching consequences. A human operator will not follow advice to shut down a complex chemical installation if there is no good explanation to do so. Similarly, medical doctors are ultimately responsible for their patients, and hence will not trust a black-box telling them a patient has a rare disease if it cannot explain why this must be so.

In this paper we give an approach for identifying negative examples in transaction data, with immediate characterisation of the why. Our approach uses the Minimum Description Length principle to decide whether an instance is drawn from the training distribution or not; examples that are very similar to the data the compressor was induced on will require only few bits to describe, while an anomaly will take many bits. By using a pattern-based compressor, thorough characterisation of its decisions is made possible. As such, our method can explain what main patterns are present/missing in a sample, identify possible observation errors, and show what changes would lead to a different decision, that is, show how strong the decision is. Furthermore, given only few negative examples the decision landscape can be estimated well.

We are not the first to address the one-class classification problem. However, important distinctions can be made between our approach and that of previous proposals. Here we give an overview of these differences, in Section 3 we discuss related work in more detail.

Most existing methods for one-class classification focus on numeric data. However, in many cases events

are discrete (e.g. alarms do or do not go off, chemical sub-structures exist or not, etc.) and therefore are naturally stored in a binary or transaction database. Applying these methods on binary data is not trivial.

In classification research, high accuracy is generally the main goal. However, as pointed out above, for an expert the explanation of the decision is equally important. By their black-box nature, existing methods typically do not offer this. Our approach does, as it uses discovered patterns to describe and classify instances.

Also, by their focus on accuracy, most existing methods require the user to set a number of parameters to maximise their performance. Whereas this has obvious merit, in practice the expert will then need to fine-tune the method, while the effect and interplays of the parameters is often unclear. Our approach does not have such parameters, making it more easily applicable.

Besides method-specific parameters, a key parameter in one-class classification is the specificity/sensitivity threshold. As there are not enough negative examples available, the decision landscape is unknown, and hence, it is difficult to set this threshold well. Our method also requires such a decision threshold. However, given only a couple of outlying examples, our method can estimate the decision landscape. As such, we provide the user with an effective way to set the decision threshold, as well as a way to see whether it is possible to identify negative examples at all.

As the compressor, here we use KRIMP [21], which describes binary data using itemsets. The high quality of these descriptions, or *code tables*, has been well established [15, 27, 28]. Alternatively, however, other compressors can be used in our framework, e.g. to apply it on other data types or with different pattern types.

Summarising, the main contributions of this work are two-fold. First, we provide a compression-based one-class classification method for transaction data that allows for thorough inspection of decisions. Second, we give a method that estimates the distribution of encoded lengths for outliers very well, given only few anomalous examples. This allows experts to fine-tune the decision threshold accordingly—making our approach parameter-free for all practical purposes.

Experimentation on our method shows it provides competitive classification accuracies, reliably predicts decision landscapes, pinpoints observation errors, and most importantly, shows why decisions are made.

The remainder of the paper is organised as follows. Next, Section 2 covers the theory of using MDL for the one-class classification problem. Related work is discussed in Section 3. We experimentally evaluate our method in Section 4. We round up with discussion in Section 5 and conclude in Section 6.

2 One-Class Classification by Compression

In this section we first give some preliminaries, and then detail the theory of using the Minimum Description Length principle for one-class classification and characterisation.

2.1 Preliminaries Throughout this paper we consider transaction databases. Let $\mathcal{I}$ be a set of items, e.g. the products for sale in a shop. A transaction $t \in \mathcal{P}(\mathcal{I})$ is a set of items that, e.g. representing the items a customer bought in the store. A database D over $\mathcal{I}$ is then a bag of transactions, e.g. the different sale transactions on a given day. We say that a transaction $t \in D$ supports an itemset $X \subseteq \mathcal{I}$, if $X \subseteq t$. The support of X in D is the number of transactions in the database in which X occurs. Note that categorical datasets can be trivially converted into transaction databases.

All logarithms are to base 2, and by convention $0 \log 0 = 0$.

2.2 One-Class Classification In one-class classification, or outlier detection, the training database D consists solely (or, overwhelmingly) of samples drawn from one distribution $\mathcal{D}_p$. The task is to correctly identify whether an unseen instance $t \notin D$ was drawn from $\mathcal{D}_p$ or not. We refer to sample being from the *positive* class if they were drawn from distribution $\mathcal{D}_p$, and to the *negative* class if they were drawn from any other distribution $\mathcal{D}_n$. We explicitly assume the Bayes error between $\mathcal{D}_p$ and $\mathcal{D}_n$ to be sufficiently low. That is, we assume well-separated classes—an unavoidable assumption in this setup. (Section 2.8 gives a technique to evaluate whether the assumption is valid.)

Next, we formalise this problem in terms of the Minimum Description Length principle.

2.3 MDL, a brief introduction The Minimum Description Length principle (MDL) [8], like its close cousin MML (Minimum Message Length) [29], is a practical version of Kolmogorov Complexity [16]. All three embrace the slogan *Induction by Compression*. For MDL, this principle can be roughly described as follows.

Given a set of models $\mathcal{M}$, the best model $M \in \mathcal{M}$ is the one that minimises

$$L(M) + L(D \mid M) \ ,$$

in which $L(M)$ is the length in bits of the description of M, and $L(D \mid M)$ is the length of the description of the data when encoded with model M.

The MDL principle implies that the optimal compressor induced on database D drawn from a distribution $\mathcal{D}$ will encode transactions drawn from this distri-

bution more succinct than any other compressor.

More in particular, let $L(t \mid M)$ be the length, in bits, of a random transaction t, after compression with the optimal compressor M induced from database D, then

$$L(t \mid M) = -\log(\Pr(t \mid D)) \ ,$$

if we assume that the patterns that encode a transaction are independent [15]. That is, under the Naïve Bayes assumption, given dataset D_1 drawn from distribution $\mathcal{D}_1$ and dataset D_2 drawn from $\mathcal{D}_2$, the MDL-optimal models M_1 and M_2 respectively induced on these datasets, and an unseen transaction t, we have the following implication

$$L(t \mid M_1) < L(t \mid M_2) \Rightarrow \Pr(t \mid D_1) > \Pr(t \mid D_2) \ .$$

Hence, it is the Bayes-optimal choice to assign t to the class of the compressor that encodes it most succinct [15].

2.4 MDL for One-Class Classification In our current setup, however, we only have sufficient training data for the positive class. That is, while we can induce M_p, we cannot access M_n, and hence require a different way to decide whether an unseen t was drawn from $\mathcal{D}_p$ or $\mathcal{D}_n$. At the same time, however, we do know that the MDL-optimal compressor M_p will encode transactions drawn from $\mathcal{D}_p$ shorter than transactions drawn from any other distribution, including $\mathcal{D}_n$. As such, we have the following theorem.

THEOREM 2.1. *Let t_1 and t_2 be two transactions over a set of items $\mathcal{I}$, respectively sampled from distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, with $\mathcal{D}_1 \neq \mathcal{D}_2$. Further, let D be a bag of transactions sampled from $\mathcal{D}_1$, and M be the MDL-optimal compressor induced on D. Then, by the MDL principle we have*

$$L(t_1 \mid M) < L(t_2 \mid M) \Rightarrow \Pr(t_1 \mid D) > \Pr(t_2 \mid D) \ .$$

With this theorem, and under the assumption that $\mathcal{D}_p$ and $\mathcal{D}_n$ are dissimilar, we can use the encoded size of a transaction to indicate whether it was drawn from the training distribution or not. By MDL we know that if $L(t \mid M)$ is small, $\Pr(t \mid D)$ is high, and hence t was likely generated by the distribution $\mathcal{D}$ underlying D. Otherwise, if $L(t \mid M)$ is (very) large, we should regard t an outlier, as it was likely generated by a another distribution than $\mathcal{D}$. Crucial, of course, is to determine when $L(t \mid M)$ is small enough.

The standard approach is to let the user define a cut-off value determined by the false-negative rate, i.e. the number of positive samples that will be classified as negatives. For our setting, this would mean setting a decision threshold θ on the encoded sizes of transactions, $L(t \mid M)$, such that at least the given number of training instances are misclassified. Clearly, this approach has a number of drawbacks. First, it definitely incorrectly marks a fixed percentage of training samples as outliers. Second, it does not take the distribution of the compressed lengths into account, and so gives an unrealistic estimate of the *real* false negative rate.

To take the distribution of encoded sizes into account, we can consider its first and second order moments. That is, its mean and standard deviation. Chebyshev's inequality, given in the theorem below, smooths the tails of the distribution and provides us a well-founded way to take the distribution into account for setting θ. It expresses that for a given random variable—in our case the compressed length, $L(t \mid M)$—the difference between an observed measurement and the sample mean is probability-wise bounded, and depends on the standard deviation.

THEOREM 2.2. (CHEBYSHEV'S INEQUALITY [7]) *Let X be a random variable with expectation μ_X and standard deviation σ_X. Then for any $k \in \mathbb{R}^+$,*

$$\Pr(|X - \mu_X| \geq k\sigma_X) \leq \frac{1}{k^2} \ .$$

Note that this theorem holds in general, and can be further restricted if one takes extra assumptions into account, e.g. whether random variable X is normally distributed or not.

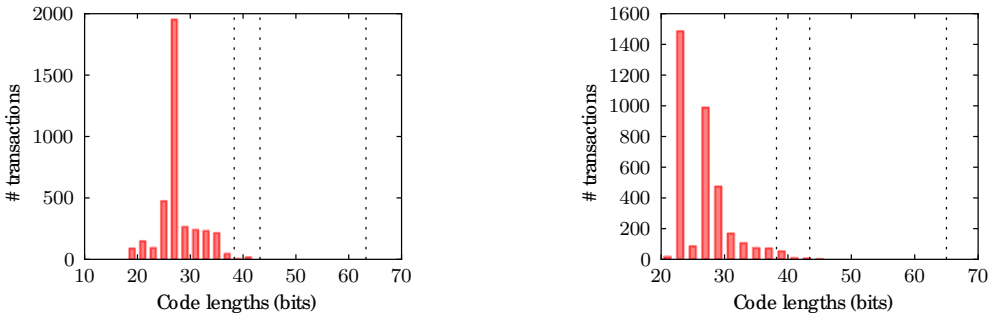
Given that M is the MDL-optimal compressor for a transaction database D over a set of items $\mathcal{I}$, M encodes D most succinct amongst all possible compressors. Hence we know that those transactions t over $\mathcal{I}$ with

$$L(t \mid M) < \frac{1}{|D|} \sum_{d \in D} L(d \mid M) \ ,$$

will have high $\Pr(t \mid D)$. In other words, if M requires fewer bits to encode t than it requires on average for transactions from D, it is very likely that t was sampled from the same distribution as D. In order to identify anomalies, we are therefore mainly concerned with transactions that are compressed significantly worse than average. To this end, we employ Cantelli's inequality, the one-sided version of Chebyshev's inequality.

THEOREM 2.3. (CANTELLI'S INEQUALITY [7]) *Let X be a random variable with expectation μ_X and standard deviation σ_X. Then for any $k \in \mathbb{R}^+$,*

$$\Pr(X - \mu_X \geq k\sigma_X) \leq \frac{1}{1 + k^2} \ .$$



(a) *Class 1* (b) *Class 2*

Figure 1: Code length histograms for the 2 different classes of *Mushroom*. Shown are the compressed sizes of transactions of the training class, together with the decision thresholds at false positive rates of respectively 10%, 5% and 1% estimated using Cantelli's inequality.

Again, like Theorem 2.2, this theorem holds in the general case, and we can restrict it depending on the knowledge we have on the distribution of the positive samples $\mathcal{D}_p$. Cantelli's inequality gives us a well-founded way to determine a good value for the threshold θ; instead of having to choose a pre-defined *amount* of false-negatives, we can let the user choose a confidence level instead. That is, an upper bound for the false-negative *rate* (FNR). Then, by $\theta = \mu + k\sigma$, we set the decision threshold accordingly.

Example. Consider Figure 1, depicting the histograms of the encoded sizes for the *Mushroom* database. This is normally the only information the user is able to derive based on samples from the positive class. The standard procedure to set the decision threshold is to choose a fixed number of known false-negatives. In our setup, we can choose the expected false-negative rate instead, for which Cantelli's gives us the correct value for θ. The dashed lines in Figure 1 show the thresholds for confidence levels of respectively 10%, 5% and 1%. The confidence level of 10%, for instance, corresponds setting θ at 3 standard deviations to the right of the average. This means the user has less than 10% chance of observing a future positive transaction that lies further than the decision threshold.

Note that using only the empirical cumulative distribution, the decision threshold would always fall inside the observed range, while by using Cantelli's inequality we are able to exceed this range: in the case above from 5% onward. Obviously, the user has no guarantees on the rate that outliers will be classified as positive samples, i.e. the false positive rate (FPR). We will discuss that particular problem in Section 2.8.

2.5 Introducing KRIMP In the previous subsections, we simply assumed access to the MDL-optimal compressor. Obviously, we have to make a choice for which compressor to use in practice. In this work we employ KRIMP, an itemset-based compressor [21], to approximate the optimal compressor for a transaction database. As such, it aims to find that set of itemsets that together describe the database best. The models KRIMP considers, *code tables*, have been shown to be of very high quality [15, 27, 28]. In this section we will give a quick overview, and refer to the original publication [21] for more details.

The key idea of KRIMP is the code table. A code table has itemsets on the left-hand side and a code for each itemset on its right-hand side. The itemsets in the code table are ordered first descending on itemset length, second descending on support and third lexicographically. The actual codes on the right-hand side are of no importance: their lengths are. To explain how these lengths are computed, the coding algorithm needs to be introduced. A transaction t is encoded by KRIMP by searching for the first itemset X in the code table for which $X \subseteq t$. The code for X becomes part of the encoding of t. If $t \setminus X \neq \emptyset$, the algorithm continues to encode $t \setminus X$. Since it is insisted that each code table contains at least all single items, this algorithm gives a unique encoding to each (possible) transaction. The set of itemsets used to encode a transaction is called its *cover*. Note that the coding algorithm implies that a cover consists of non-overlapping itemsets. The length of the code of an itemset in a code table CT depends on the database we want to compress; the more often a code is used, the shorter it should be. To compute this code length, we encode each transaction in the database D. The *usage* of an itemset $X \in CT$ is the number of transactions $t \in D$ which have X in their cover. The

relative usage of $X \in CT$ is the probability that X is used in the encoding of an arbitrary $t \in D$. For optimal compression of D, the higher $\Pr(X \mid D)$, the shorter its code should be. In fact, from information theory [8], we have the Shannon entropy, that gives us the length of the optimal prefix code for X, by

$$L(X \mid CT) = -\log(\Pr(X \mid D)) = \frac{usage(X)}{\sum_{Y \in CT} usage(Y)} .$$

The length of the encoding of transaction is now simply the sum of the code lengths of itemsets in its cover,

$$L(t \mid CT) = \sum_{X \in cover(t)} L(X \mid CT) .$$

The size of the encoded database is the sum of the sizes of the encoded transactions,

$$L(D \mid CT) = \sum_{t \in D} L(t \mid CT) .$$

To find the optimal code table using MDL, we need to take both the compressed size of the database, as described above, and the size of the code table into account. For the size of the code table, we only consider those itemsets that have a non-zero usage. The size of the right-hand side column is obvious; it is simply the sum of all the different code lengths. For the size of the left-hand side column, note that the simplest valid code table consists only of the single items. This is the standard code table ST, of which we use the codes to compute the size of the itemsets in the left-hand side column. Hence, the size of the code table is given by:

$$L(CT) = \sum_{\substack{X \in CT \\ usage(X) \neq 0}} L(X \mid ST) + L(X \mid CT) .$$

Siebes et al. [21] define the optimal set of (frequent) itemsets as the one whose associated code table minimises the total compressed size:

$$L(CT) + L(D \mid CT) .$$

Krimp starts with the standard code table and the frequent itemsets up to a given *minsup* as candidates. These candidates are ordered first descending on support, second descending on itemset length and third lexicographically. Each candidate itemset is considered by inserting it in CT and calculating the new total compressed size. A candidate is only kept in the code table iff the resulting total size is smaller. If it is kept, all other elements of CT are reconsidered to see if they still contribute positively to the compression. Note that the *minsup* parameter is only used to control the number of candidates: it holds in general that the lower the *minsup*, the larger the number of candidates, the larger the search space and the better the final code table approximates the optimal code table. In MDL we are after the optimal compressor, for Krimp this means *minsup* should be set as low as practically feasible.

For more details, we refer to the original paper [21].

2.6 Krimp for One-Class Classification By running Krimp on training database D, consisting of positive examples, we obtain an approximation of the MDL-optimal compressor for D. To employ this compressor for one-class classification, or outlier detection, we combine the insights from Sections 2.4 and 2.5. Formally, this means that given a decision threshold θ, a code table CT for database D, both over a set of items $\mathcal{I}$, for an unseen transaction t also over $\mathcal{I}$, we decide that t belongs to the distribution of D iff

$$L(t \mid CT) \leq \theta .$$

In other words, if the encoded length of the transaction is larger than the given threshold value, we decide it is an outlier. We will refer to our approach as OC^3, which stands for One-Class Classification by Compression.

The MDL-optimal compressor for the positive class can obviously best be approximated when D consists solely of many samples from $\mathcal{D}_p$. In practice, however, these demands may not be met.

Firstly, D may not be very large. The smaller D is, the less well the compressor will be able to approximate the MDL-optimum. We thus especially expect good performance for large training databases. Note that MDL inherently guards against overfitting: adding too much information to a model would make it overly complex, and thus degrade compression.

Secondly, D may contain some (unidentified) samples from $\mathcal{D}_n$. However, under the assumption that $\mathcal{D}_p$ and $\mathcal{D}_n$ are well separated, the MDL-optimal compressor for a dataset D with a strong majority of samples from $\mathcal{D}_p$, and only few from $\mathcal{D}_n$, will typically encode future samples from $\mathcal{D}_p$ in fewer bits than those from $\mathcal{D}_n$. As such, the classifier will also work when the training data contains anomalies. Unless stated otherwise, in the remainder of this paper we assume that D is sampled solely from the positive class.

2.7 Characterising Decisions One of the main advantages of using a pattern-based compressor like Krimp, is that we can *characterise* decisions.

As an example, suppose a transaction t is classified as an outlier. That is, $L(t \mid CT) > \theta$. To inspect this decision, we can look at the itemsets by which

the transaction was covered; this gives us information whether the outlier shows patterns characteristic for the positive class. That is, the more t resembles the patterns of the positive class, the more it will be covered by long itemsets and less by singletons. On the other hand, patterns that are highly characteristic for D that are *missing* from the transaction cover are equally informative; they pinpoint where t is essentially different from the positive class.

Since code tables on average contain up to a few hundred of elements, this analysis can easily be done by hand. In addition, we can naturally rank these patterns on encoded size, to show the user what most characteristic, or frequently used, patterns are missing or present. As such, decisions can easily be thoroughly inspected.

2.8 Estimating the Decision Landscape For many situations it is not unrealistic to assume that, while not abundant, *some* example outliers are available besides the training data (e.g. less than 10). Even if these examples are not fully representative for the whole negative class $\mathcal{D}_n$, we can use them to make a more informed choice for the threshold parameter.

To this end, we propose to generate artificial outliers, based on the given negatives, to estimate the number of bits our positive-class compressor will require to encode future samples from $\mathcal{D}_n$; given this estimated distribution of encoded lengths, and the encoded lengths for the training data, we can set the decision threshold θ to maximise expected accuracy—as well as to inspect whether it is likely we will see good classification scores.

For this, we have to make one further assumption that builds on the one underlying one-class classification, i.e. that the positive and negative distributions are essentially different, and hence, that the MDL-optimal compressor for the positive class will badly compress samples from the negative class. Now, in addition, we assume that by slightly altering a known outlier it will still be an outlier. Note that if the positive and negative distributions are not well-separated, this assumption will not hold.

More formally, let us consider the MDL-optimal compressor M for a training database D of samples from $\mathcal{D}_p$, all over a set of items $\mathcal{I}$. Further, we have a known outlier $t \in \mathcal{D}_n$ for which $L(t \mid M)$ is large, compared to $L(d \mid M)$ for random $d \in D$. Next, let us construct transactions t' by removing few (one, two, ...) items X from t, and replacing these with equally many items Y not in t, i.e. $t' \leftarrow (t \setminus X) \cup Y$, with $|t'| = |t|$, $Y \subseteq \mathcal{I}$ and $X \setminus t = \emptyset$. Now, the main assumption is that on average, t' is about as likely as t in $\mathcal{D}_n$, i.e. we have $\Pr(t' \mid \mathcal{D}_n) \approx \Pr(t \mid \mathcal{D}_n)$, and t' is unlikely to be drawn from $\mathcal{D}_p$, i.e. $\Pr(t' \mid \mathcal{D}_p)$ is small and $L(t' \mid M)$ relatively large. Under this assumption, $L(t' \mid M)$ gives us a good estimate of the encoded sizes of real future outliers.

Naturally, the quality of the estimate is strongly influenced by how we swap items. One option is random change. Alternatively, we can take the compressor and the training data into account. Through these, we can identify those X and Y that will maximally change $L(t' \mid M)$; by choosing X and Y such that t' is compressed badly, we will (likely) overestimate the separation between $\mathcal{D}_p$ and $\mathcal{D}_n$, and analogously we underestimate when we minimise $L(t' \mid M)$.

We argue that in this setup the latter option is preferred. First of all, it is possible that the identified outliers are extremes—otherwise they might not have been discovered. Second, it is not unlikely the two distributions share basic characteristics. A pattern very common in D will likely also occur in samples from $\mathcal{D}_n$; we should take this into account when sampling t's.

Given some prototype outliers, we generate new samples according to the following distribution. First, we uniformly choose a transaction t among the given prototype outliers. Next, from t we select an item i to remove, using the following exponential distribution,

$$\Pr(i) = \frac{2^{-1/l(i)}}{\sum_{j \in t} 2^{-1/l(j)}} ,$$

where, $l(i) = \frac{L(Z|CT)}{|Z|}$ and $i \in Z \in \mathit{cover}(t)$. By this choice, we prefer to remove those items that require the most bits to be described—that is, those that fit the patterns from the positive class least. To complete the swap, we choose an item from $\mathcal{I} \setminus t$ to add to t. (Note that if the original dataset is categorical, it only makes sense to swap to items corresponding to the same category.) We choose the item j to swap to according to the following distribution, similar to how we previously [27] imputed missing values,

$$\Pr(j) = \frac{2^{-L(t_j|CT)}}{\sum_{k \in \mathcal{I} \setminus (t \setminus i)} 2^{-L(t_k|CT)}} ,$$

with $t_j = (t \setminus \{i\}) \cup \{j\}$. This distribution generates transactions t' with preference to short encoding.

To estimate the expected false positive rate, we generate a large number of samples and calculate mean and standard deviation. One can use Cantelli's inequality, or assume the encoded lengths of the outliers to follow a normal distribution. Then, one can update θ by taking both FPR and FNR into account, e.g. choose the intersection between the two distributions.

2.9 Measuring Decision Certainty Item swapping is also useful to show how a transaction t needs to be modified in order to change the classification verdict. Or, the other way around, to show what items are most important with regard to the decision of t. However, we can go one step further, and look at the certainty of a decision by considering the encoded lengths of altered transactions. The rationale is that the more we need to change t to let its encoded length reach below the decision threshold, the more likely it is this example is indeed an outlier. Alternatively, for a sample with an observation error, a small change may be enough to allow for a correct decision.

So, the goal is, given a transaction t, to maximally reduce the encoded size $L(t \mid CT)$ with a limited number of changes δ. In general, transactions may have different cardinality, so up to δ elements can be added—in categorical databases transactions are of the same size and up to δ items need to be swapped. Clearly, with $\binom{|\mathcal{I}\setminus t|}{\delta} \times \binom{|t|}{\delta}$ possible altered transactions, solving this problem exhaustively quickly becomes infeasible for larger δ and $\mathcal{I}$. However, in our setup we can exploit the information in the code table to guide us in choosing those swaps that will lead to a short encoding.

The idea is to cover the transaction t using the most specific elements, i.e., the itemsets $X \in CT$ with highest cardinality $|X|$, while tolerating up to δ missing items. The reason to choose the most specific elements is that we cover the most singletons by one itemset, and therewith replace many (on average long) codes by just one (typically short) code. Note that, alternatively, one could attempt to greedily cover with the elements with the shortest codes, or even minimise the encoded length by dynamic programming.

We present the pseudo-code for finding a δ-fault-tolerant cover for a transaction t and a code table CT as Algorithm 1. In order to calculate the resulting encoded length of t, we simply use the code lengths in the code table. In the algorithm, while covering t, we keep track of the number of swaps made so far, denoted as ϵ. Once the number of uncovered items in t is smaller or equal than ϵ (line 2) we can stop: the items remaining in t are the items we have to remove, the items $S \setminus t$ are the ones we have to add. We only use Algorithm 1 as a step during analysis of decisions; it would require non-trivial adaptations to Krimp to let it consider missing items when constructing the optimal code table, and our focus here is clearly not to construct a new compressor.

Algorithm 1 Fault-Tolerant Cover

Input: Transaction $t \in D$ and code table CT, with CT and D over a set of items $\mathcal{I}$. Maximum number of faults δ, and current number of faults ϵ.

Output: Fault-tolerant cover of t using the most specific elements of CT, tolerating at most δ faults.

1. $S \leftarrow$ smallest X of CT in **Standard Cover Order** with $|X \setminus t| \leq \delta$, and $X \in t$ if $|X| = 1$
2. **if** $|t \setminus S| \leq \epsilon$ **then**
3. $\quad Res \leftarrow \{S\}$
4. **else**
5. $\quad Res \leftarrow \{S\} \cup$ **Fault-Tolerant Cover**$(t \setminus S, CT, \delta - |S \setminus t|, \epsilon + |S \setminus t|)$
6. **end if**
7. **return** Res

3 Related Work

Much of the existing work on one-class classification targets record data constructed with numerical attributes. For a general overview, we refer to [17, 24]. Very few of these studies are applicable to transaction data, as many of the these methods rely on density estimations (e.g. Parzen-windows or mixture of Gaussians) to model the positive class. Two state-of-the-art algorithms that, by respectively using the appropriate kernel or distance measure, are applicable to binary data are Support Vector Data Description (SVDD) [26], or one-class SVM [20], and Nearest Neighbour Data Description (NNDD) [24].

He et al. [9] study outlier detection for transaction databases. The authors assume that transactions having less frequent itemsets are more likely to be outliers. Narita et al. [18], on the other hand, use information of association rules with high confidence for the outlier degree calculation. Both these approaches were formulated to detect outliers in a database, but can also be used for single-class problems. While both methods use patterns and share the transparency of our approach, their performance is very parameter-sensitive. For the former, the authors restrict themselves to the top-k frequent itemsets. In the latter, besides a minimum support threshold, the user also needs to specify minimum confidence. Both papers notice large changes in accuracy depending on the parameter settings, but postpone insight in how to set these optimally to future work.

We are not the first to employ the MDL principle for one-class classification. However, to the best of our knowledge, we are the first to apply it in a binary/transaction database setting. Bratko et al. [3] and Nisenson et al. [19] consider streams of character data (e.g. text, streams of bytes or keyboards events, etc.). In these approaches, the compressor is immaterial; that is, well-known universal compression algorithms, such as gzip, are used, which do not allow for inspection. The algorithms compress common shared (sub)strings of characters that occur often together in the streams.

Table 1: Statistics of the datasets used in the experiments. Given are, per dataset, the number of rows, the number of distinct items, the number of classes and the *minsup* thresholds for the KRIMP-compressor.

				KRIMP
Dataset	$\|D\|$	$\|\mathcal{I}\|$	$\|\mathcal{K}\|$	*minsup*
Adult	48842	95	2	50
Anneal	898	66	5	1
Breast	699	14	2	1
Chess (k-k)	3196	75	2	400
Chess (kr-k)	28056	58	18	1
Connect-4	67557	129	3	1
Led7	3200	14	10	1
Mushroom	8124	117	2	1
Nursery	12960	27	4	1
Pageblocks	5473	39	5	1
Pen Digits	10992	76	10	10
Pima	768	36	2	1
Typist	533	40	10	1

In transaction databases one is not interested in sequences of items, as items are unordered.

We use KRIMP, introduced by Siebes et al. [21], to build a compression model relying on the (frequent) itemsets to encode transactions. Van Leeuwen et al. show that these models are able to compete with the state-of-the-art multi-class classifiers [15]. Alternatively, one could use PACK [23], or any other suited transaction data compressor, in our framework.

4 Experiments

In this section we experimentally evaluate our approach. First, we discuss the experimental setup, then investigate classification accuracy. Next, we show how classification decisions can be characterised, and observation errors in transactions can be detected. Finally, we estimate the distribution of encoded lengths for outliers to improve classification results.

4.1 Experimental Setup For the experimental validation of our method we use a subset of publically available discretised datasets from the LUCS-KDD repository [4]. In addition to these datasets, shown in Table 1, we also consider the *Typist* recognition problem provided by Hempstalk et al. [10], discretised and normalised using the LUCS-KDD DN software [4].

We turn this selection of multi-class classification datasets into several one-class classification problems. For each dataset, one class at a time, we consider a particular class $K \in \mathcal{K}$ as the positive class and the samples from the remaining classes $\mathcal{K} \setminus K$ as outliers. All results reported in this section are 10-fold cross-validated.

To obtain the KRIMP code table CT_K for a dataset D_K, we use (closed) frequent itemsets in D_K mined with *minsup* set as low as practically feasible. The actual values for *minsup* are depicted in Table 1.

4.2 Classification We compare our method to two state-of-the-art [10, 26] one-class classifiers: Support Vector Data Description (SVDD) [20, 26] and Nearest Neighbour Data Description (NNDD) [24], both of which are publically available in DDtools [25].

For the kernel in SVDD, or one-class SVM, we use the polynomial kernel, and optimise degree parameter d for high accuracy. While more generally employed in one-class classification, for binary data the RBF kernel leads to worse results. One other advantage of using polynomial kernels with binary data is the close relatedness to itemsets: the attributes in the feature space induced by this kernel indicate the presence of itemsets up to length d.

For NNDD we use the Hamming distance. To determine the number of neighbouring points that are used to decide the classification, parameter k, we optimise the log-likelihood of the leave-one-out density estimation.

To compare the algorithms, we use the AUC, i.e. area under the ROC curve, as it is independent of actual decision thresholds. Table 2 shows the AUC scores averaged over 10-folds and each of the classes of each dataset. We see, and it is confirmed by pairwise Student's t-tests at α-level 5%, that, in general, the performance of OC^3 is on par with that of SVDD and both algorithms clearly outperform NNDD. Especially for datasets with high numbers of transactions, e.g. *Chess* (*kr-k*), *Connect-4*, *Mushroom*, and *Pen Digits*, OC^3 provides very good performance, while for data with very small classes, such as *Pima* and *Typist*, it does not improve over the competing methods. This is expected, as MDL requires sufficient samples to properly estimate the training distribution. In conclusion, OC^3 performs on par with the state of the art, and can improve over it for large databases.

The sub-par performance of all classifiers on *Adult*, *Pageblocks*, and *Pima*, can be explained: these datasets contain large numbers of identical transactions that only differ on class-label, making errors unavoidable.

In our setup, classification depends on code length distributions for the different classes. Figures 2a and 2b show the histograms on the training instances of two classes from the *Pen Digits* dataset. One can notice that the positive transactions (shown in hatched red) are bet-

Table 2: AUC scores for the benchmark datasets. Shown are, per dataset, mean and standard deviation of the average AUC score over the classes. The KRIMP-compressor in OC^3 ran using the *minsup* values in Table 1.

Dataset	OC^3	SVDD	NNDD
Adult	68.63 ± 3.71	**72.86** ± 6.68	65.64 ± 5.84
Anneal	95.58 ± 0.62	93.62 ± 9.95	**97.32** ± 2.36
Breast	87.12 ± 12.76	**96.47** ± 1.19	72.77 ± 33.04
Chess (k-k)	68.57 ± 0.58	65.62 ± 7.89	**82.27** ± 1.69
Chess (kr-k)	**94.89** ± 5.18	86.46 ± 8.71	80.38 ± 9.79
Connect-4	**73.73** ± 6.47	55.14 ± 6.98	62.22 ± 5.52
Led7	91.45 ± 3.50	**93.41** ± 3.45	79.43 ± 7.26
Mushroom	**100.00** ± 0.00	97.69 ± 2.89	99.92 ± 0.07
Nursery	98.43 ± 1.68	**98.68** ± 1.68	84.54 ± 7.16
Pageblocks	52.59 ± 23.87	**56.79** ± 13.91	51.13 ± 13.07
Pen Digits	98.25 ± 0.89	**98.98** ± 0.80	98.32 ± 1.00
Pima	50.94 ± 28.93	**65.32** ± 9.66	50.63 ± 12.81
Typist	87.81 ± 6.93	**92.30** ± 6.35	87.84 ± 7.73
average	84.62 ± 8.11	84.71 ± 3.35	80.11 ± 8.79

ter compressed than the outliers, i.e. the other classes, (filled blue) which is in accordance with the MDL assumption. The quality of the classification result is determined by the amount of overlap. Following, the more the two distributions are apart, the better the results are. For the sub-par performing datasets in Table 2, the histograms overlap virtually completely, and hence the separation-assumption is not met.

If memory or time constraints do not allow us to mine for frequent itemsets at low *minsup*, e.g. for the dense *Chess* (*k-k*) dataset, some characteristic patterns might be missing from the code tables, in turn leading to sub-par compression and performance. Clearly, at further expense, mining at lower *minsup* would provide better compression, which in turn should provide better performance on these datasets.

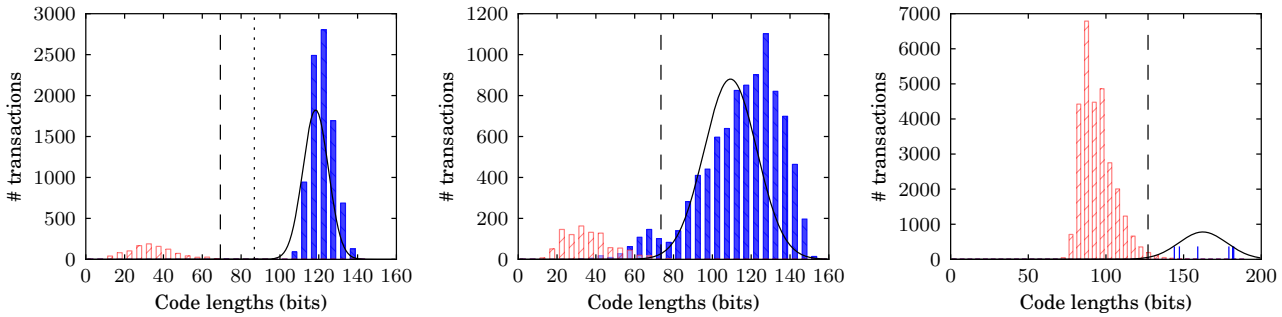
4.3 Inspection and Characterisation One of the key strengths of our approach is that we can analyse and explain the classification of transactions. Here, we investigate how well the approach Section 2.7 outlines works in practice. Note that black-box methods like SVDD and NNDD do not allow for characterisation.

To demonstrate the usability of our approach in a real problem setting, we perform a case study on real MCADD data, described in detail in Section 4.5, obtained from the Antwerp University Hospital. Medium-Chain Acyl-coenzyme A Dehydrogenase Deficiency (MCADD) [1] is a deficiency newborn babies are screened for during a Guthrie test on a heel prick blood sample. This recessive metabolic disease affects about one in 10 000 people while around one in 65 is a carrier of the responsible mutated gene. If left undiagnosed, this rare disease is fatal in 20 to 25% of the cases and many survivors are left with severe brain damage after a severe crisis.

Figure 3, which shows the covers of 4 transactions from the *MCADD* dataset, serves as our running example. A typical cover of a transaction from the positive class is shown in Figure 3a (bottom): one or more larger itemsets possibly complemented with some singletons. Also note that the lengths of the individual codes, denoted by the alternating light and dark grey bars, are short. This is in strong contrast with the covers of the other transactions, which resemble typical covers for outlier transactions, where mostly singletons are used. Also, the code lengths of itemsets in the cover of an outlier transaction are typically long.

The 'outlier' transaction at the top of Figure 3a was artificially constructed by taking a transaction from the positive class. If we use Algorithm 1, with δ, the number of mistakes allowed, set to one, we exactly recover the true positive transaction (bottom of Figure 3a). This shows that we are able to detect and correct observation errors in future samples. Or, the other way around, if many swaps are needed for the decision to change, this gives a plausible explanation why the transaction is identified as an outlier.

The top transaction in Figure 3b is a true outlier. We first of all observe that the encoded size of the transaction is large. Next, as the items are labeled descending on their support, we see that more than halve of the items belong to the 20% least frequent. Moreover, we note that by applying Algorithm 1 the



(a) *Pen Digits* - Class 1 (b) *Pen Digits* - Class 10 (c) *MCADD* - Healthy Class

Figure 2: Code length histograms for *MCADD* and *Pen Digits*. Shown are the compressed sizes of transactions of the positives (in hatched red) and the outliers (in filled blue). The solid black line depicts our estimate, using 4 counterexamples, of the encoded lengths for the outliers. The dashed line represents the decision threshold bounding the FPR at 10%, while the dotted line, if present, shows the decision threshold after updating.

gains in encoded length are negligible (see bottom of Figure 3b for one example). This trend holds in practice, and strengthens confidence in the assumption made in Section 2.8, that small variations of negative samples remain 'negative'. However, as shown above, perturbing positive samples can cause large variation in encoded size as larger patterns fall apart into singletons.

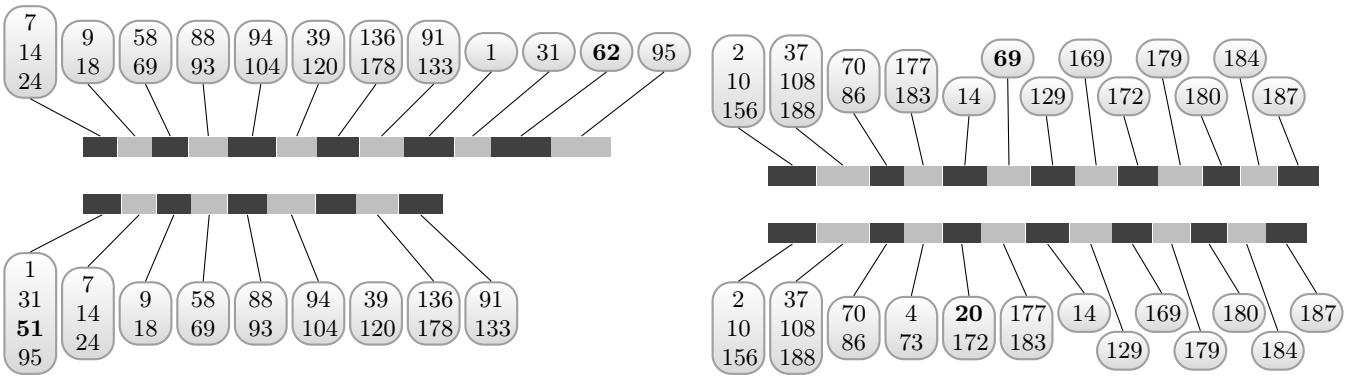
4.4 Estimating Outlier Distributions Next, we investigate how well we can estimate the distribution of encoded lengths for outliers, by generating samples from a limited number of negatives, as outlined in Section 2.8. For both the *MCADD* and *Pen Digits* dataset, we generated 10 000 samples based on 4 randomly selected outliers. As shown in Figure 2, the so-derived normal distributions, based on the sample mean and standard variation, approximate the true outlier distributions closely. Note that, as intended, the estimate is slightly conservative.

Alternatively, if we uniformly swap items, the patterns shared by both outliers and positives are more likely to be destructed. Experiments show this provides overly optimistic estimates of the separation of the distributions. Further, in many situations, it is sensible to estimate pessimistically. That is, closer to the positive samples. For example, if MCADD is left undiagnosed, it is fatal in 20% to 25% of the cases, and survivors are typically left with severe brain damage.

We will now use illustrate through Figure 2a and 2b how a user can use this information to update the decision threshold θ. Initially, the user only has information from the positive class, denoted by the hatched red histograms. By using Cantelli's inequality, the decision threshold can be set to allow up to 10% false negatives (dashed line). After estimating the negative distribution, we notice this initial choice fits perfect for class 10 in Figure 2b. However, the decision threshold for class 1 in Figure 2a is too low (5% FNR, 0.2% FPR). If we update the decision threshold (dotted line) to counterbalance both the estimated false negative and false positive rate using Cantelli's inequality, we observe that the false negative and false positive rates on the hold-out validation set are more in balance: 1.8% FNR and 1.5% FPR. So, by using information derived from the artificial outliers, the user can update the threshold and improve classification results.

4.5 Case study: MCADD In our study, the dataset contains controls versus MCADD, with respectively 32 916 negatives and only 8 positives. The instances are represented by a set of 21 features: 12 different acylcarnitine concentrations measured by tandem mass spectrometry (TMS), together with 4 of their calculated ratios and 5 other biochemical parameters. We applied k-means clustering with a maximum of 10 clusters per feature to discretise the data resulting in 195 different items. We run Krimp using a minimum support of 15, which corresponds to a relative *minsup* of 0.05%.

Repeated experiments using 10-fold cross-validation show that all 8 positive cases are ranked among the top-15 largest encoded transactions. Besides, we notice that the obtained performance indicators (100% sensitivity, 99.9% specificity and a positive predictive value of 53.3%) correspond with the state-of-the-art results [1, 11] on this problem. Moreover, analysing the positive cases by manually inspecting the patterns in the code table and covers, reveals that particular combinations of values for acylcarnitines C2, C8 and C10 together with particular following ratios $\frac{C8}{C2}$, $\frac{C8}{C10}$ and $\frac{C8}{C12}$ were



(a) (b)

Figure 3: Example transactions from *MCADD*: observation error (top left), true outlier (top right) and corrections (bottom) suggested by Algorithm 1 ($\delta = 1$). The rounded boxes visualise the itemsets that are used to cover the transaction; each itemset is linked to its code. Width of the bar represents the length of the code. Swapped items are displayed in boldface. Clearly, the observation error is correctly identified, as its encoded length can be decreased much with only swap, whereas the encoded length of the outlier cannot be improved much.

grouped together in the covers of the positive cases. Exactly these combination of variables are commonly used in diagnostic criteria by experts and were also discovered in previous in-depth studies [1, 11].

The largest negative samples stand out as a rare combination of other acetylcarnitine values. Although these samples are not MCADD cases, they are very different from the general population and are therefore outliers by definition.

5 Discussion

The experiments in the previous section demonstrate that our method works: transactions that are succinctly compressed by patterns from the positive class are indeed highly likely to belong to that class. The obtained AUC scores are on par with the state-of-the-art one-class classifiers for binary data, and especially good for large datasets.

In practice, a user lacks an overall performance measure to reliably specify the specificity/sensitivity threshold, as examples for the negative class are rare in one-class classification. Consequently, the ability to inspect classification decisions is important. In contrast to existing approaches, our pattern-based method provides the opportunity to analyse decisions in detail.

Examples in the experiments show possible use cases. First, we are able to explain why a transaction is classified as such. Transactions that are covered with itemsets that are highly characteristic for the positive class are likely positives as well, while those transactions that do not exhibit such key patterns (and thus encoded almost solely by singletons) can be considered as outliers. Next, our approach is able to detect, and correct, observation errors in test samples.

Furthermore, if some outliers are available, we propose to approximate the encoding distributions of the outliers. By using this information, a user is able to make a more informed decision when setting the decision threshold. Here, we choose to generate samples conservatively. Visualising the approximated distribution of encoded lengths for outliers shows whether one-class classification, based on compressed lengths, is actually possible.

A case study on the MCADD dataset shows that true outliers are correctly identified. Different from the setup explored here, where unseen transactions are considered as potential outliers, one could also be interested in detecting outliers *in* the dataset at hand. The method discussed in this paper may well provide a solution for this problem, that is, pointing out the most likely outlying items and transactions by compressed size. Related, as a future work, we are investigating a rigorous approach for cleaning data using MDL.

Although in this work we focus on binary data, the methods we present can be generalised as a generic approach using MDL. That is, as long as a suited compressor is employed, the theory will not differ for other data types. Variants of Krimp have already been proposed for sequences, trees, and graphs.

6 Conclusion

In this paper we introduced a novel approach to outlier detection, or one-class classification, for binary or transaction data. In this setting little or no examples

are available for the class that we want to detect, but an abundance of positive samples exists. Our method is based on the Minimum Description Length principle, and decides by the number of bits required to encode an example using a compressor trained on samples of the normal situation. If the number of bits is much larger than expected, we decide the example is an outlier.

Experiments show that this approach provides accuracy on par with the state of the art.

Most important, though, is that it holds three advantages over existing methods. First, by relying on pattern-based compression, our method allows for detailed inspection and characterisation of decisions, both by showing which patterns were recognised in the example, as well as by checking whether small changes affect the decision. Second, we show that given a few example outliers, our method can reliably estimate the decision landscape. Thereby, it can predict whether the positive class can be detected at all, and allows the user to make an informed choice for the decision threshold. Third, given this estimate the method is parameter-free.

Acknowledgements

The authors thank i-ICT of Antwerp University Hospital (UZA) for providing MCADD data and expertise. Koen Smets and Jilles Vreeken are respectively supported by Ph.D. and Postdoctoral Fellowships of the Research Foundation - Flanders (FWO).

References

[1] C. Baumgartner, C. Böhm, and D. Baumgartner. Modelling of classification rules on metabolic patterns including machine learning and expert knowledge. *J. Biomed. Inform.*, 38:89–98, 2005.

[2] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Stat. Sci.*, 17(3), 2002.

[3] A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *JMLR*, 7, 2006.

[4] F. Coenen. The LUCS-KDD discretised/normalised ARM and CARM data library, 2003. `http://www.csc.liv.ac.uk/~frans/KDD/Software/`.

[5] N. Dom, D. Knapen, D. Benoot, I. Nobels, and R. Blust. Aquatic multi-species acute toxicity of (chlorinated) anilines: Experimental versus predicted data. *Chemosphere*, 81:177–186, 2010.

[6] A. B. Gardner, A. M. Krieger, G. Vachtsevanos, and B. Litt. One-class novelty detection for seizure analysis from intracranial EEG. *JMLR*, 7, 2006.

[7] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. 2001.

[8] P. D. Grünwald. *The Minimum Description Length Principle*. 2007.

[9] Z. He, X. Xu, J. Z. Huang, and S. Deng. FP-Outlier: Frequent pattern based outlier detection. *ComSIS*, 2(1), 2005.

[10] K. Hempstalk, E. Frank, and I.H. Witten. One-class classification by combining density and class probability estimation. In *Proceedings of ECML PKDD'08*, 2008.

[11] S. Ho, Z. Lukacs, G. F. Hoffmann, M. Lindner, and T. Wetter. Feature construction can improve diagnostic criteria for high-dimensional metabolic data in newborn screening for medium-chain acyl-coa dehydrogenase deficiency. *Clin. Chem.*, 53(7), 2007.

[12] P. Juszczak, N. M. Adams, D. J. Hand, C. Whitrow, and D. J. Weston. Off-the-peg and bespoke classifiers for fraud detection. *Comput. Stat. Data Anal.*, 52(9), 2008.

[13] P. Laskov, C. Schäfer, and I. Kotenko. Intrusion detection in unlabeled data with quarter-sphere support-vector machines. In *Proceedings of DIMVA'04*, 2004.

[14] C.J. van Leeuwen and T.G. Vermeire. *Risk Assessment of Chemicals*. Springer, 2007.

[15] M. van Leeuwen, J. Vreeken, and A. Siebes. Compression picks item sets that matter. In *Proceedings of ECML PKDD'06*, 2006.

[16] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. 1993.

[17] M. Markou and S. Singh. Novelty detection: a review. *Sign. Proc.*, 83(12), 2003.

[18] K. Narita and H. Kitagawa. Outlier detection for transaction databases using association rules. In *Proceedings of WAIM'08*, 2008.

[19] M. Nisenson, I. Yariv, R. El-Yaniv, and R. Meir. Towards behaviometric security systems: Learning to identify a typist. In *Proc. of ECML PKDD'03*, 2003.

[20] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comp.*, 13(7), 2001.

[21] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *Proceedings of SDM'06*, 2006.

[22] K. Smets, B. Verdonk, and E. M. Jordaan. Discovering novelty in spatio/temporal data using one-class support vector machines. In *Proc. of IJCNN'09*, 2009.

[23] N. Tatti and J. Vreeken. Finding good itemsets by packing data. In *Proceedings of ICDM'08*, 2008.

[24] D.M.J. Tax. *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, 2001.

[25] D.M.J. Tax. DDtools 1.7.3, the data description toolbox for matlab, 2009.

[26] D.M.J. Tax and R.P.W. Duin. Support vector data description. *Mach. Learn.*, 54(1), 2004.

[27] J. Vreeken and A. Siebes. Filling in the blanks - Krimp minimisation for missing data. In *Proceedings of ICDM'08*, 2008.

[28] J. Vreeken, M. van Leeuwen, and A. Siebes. Characterising the difference. In *Proceedings of KDD'07*, 2007.

[29] C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length*. 2005.