Rule-based OneClass-DS Learning Algorithm

Dat T. Nguyen and Krzysztof J. Cios

Abstract—One-class learning algorithms are used in situations when training data are available only for one class, called target class. Data for other class(es), called outliers, are not available. One-class learning algorithms are used for detecting outliers, or novelty, in the data. The common approach in one-class learning is to use density estimation techniques or adapt standard classification algorithms to define a decision boundary that encompasses only the target data. In this paper, we introduce OneClass-DS learning algorithm that combines rule-based classification with greedy search algorithm based on density of features. Its performance is tested on 25 data sets and compared with eight other one-class algorithms; the results show that it performs on par with those algorithms.

Index Terms—One-class learning algorithm: OneClass-DS, outlier detection, anomaly detection, novelty detection

-----◆-----

1 INTRODUCTION

In a classical supervised learning problem, data instances that represent several classes are available for designing a classifier and a learning algorithm uses this information to discriminate between the classes. In a single-instance learning scenario, the simplest, we have class label for each instance, while in multiple-instance learning scenario we have class label for a group of instances, called bags, [1, 2]. In the third, more difficult, learning scenario we have data available only for a single class, called Target (each instance belongs to Target). Once a classifier is designed it can be used for predicting whether a new instance belongs to the Target class or not. If an instance does not belong to a Target class it is called an outlier, also known as “ignorance” [3]. Therefore this type of learning problem is known as outlier, or novelty, or anomaly detection. One-class learning algorithms are often designed using information about distribution of the Target training data. Several methods have been developed to solve this challenging problem and they can be divided into density estimation, boundary, reconstruction, and rule-based methods. They differ in their ability to cope with, or exploit, different characteristics of data such as feature scaling, grouping of data into clusters, or taking advantage of data distribution convexity. These methods are briefly described next.

1.1 Density estimation methods

The common and most straightforward method is to estimate density of the training data [4] and then use some threshold to encompass the data. When data are sufficiently large and a density model, such as Parzen, is used this approach works quite well. The drawback, however, is that it requires a large number of instances to overcome the curse of dimensionality [5]. If the dimensionality of data and complexity of the density model are restricted then a large bias is introduced resulting in the model that does not fit the data well. Finding the right model to describe Target data distribution is a bias-variance dilemma. Using the density approach one needs to assume a distribution type, such as Gaussian [6, 7, 8], or mixture of Gaussians [5, 6, 8], or Parzen [9, 10, 11]. Algorithm introduced in [1] builds density function from a chosen distribution and then combines this function with a class probability to form an adjusted estimate of the density function of the Target class, which is then used for constructing a decision tree.

1.2 Boundary methods

Boundary methods define a closed boundary around the target data first and then it is optimized. These methods rely heavily on distances between instances and are sensitive to feature scaling [8, 12]. Although the volume of data is not always minimized, most of these methods have a strong bias towards a minimal-volume solution. How small the volume is to be depends on the model. The advantage of the boundary methods is that the number of instances required for training is smaller than one required by the density methods. The difficulty, however, is shifted into defining appropriate distance measures. One such example method is the k-center, which covers the data with k small balls with equal radii [13]. The ball centers, μ_k , are

• DT Nguyen is with the Department of Computer Science, School of Engineering, Virginia Commonwealth University, Richmond, USA, and School of Science & Technology, Hoa-Sen University, HCMC, Vietnam. E-mail: dat.nguyentien1780@hoasen.edu.vn.

• KJ Cios is with Department of Computer Science, School of Engineering, Virginia Commonwealth University, Richmond, USA, and IITIS Polish Academy of Sciences, Gliwice, Poland. E-mail: kcios@vcu.edu.

placed on training instances such that the maximum of all minimum distances between training instances and the centers is minimized. To fit the model the following error is minimized:

$$\varepsilon_{k-centre} = \max_i \left(\min_k \|x_i - \mu_k\|^2 \right)$$

It uses a greedy search strategy, starting with random initialization. The radius is determined by maximum distance to the instances that the corresponding ball captures. Because of this, the method is sensitive to outliers possibly present in the target data but works well when data have good (compact) clustering structure. However, the user needs to specify in advance both the number of balls, k , and the maximum number of tries (the number of runs with random initialization).

Another such method is the nearest-neighbor, NN-d, which is designed from local density estimation by the nearest neighbor classifier[5]. It avoids explicit density estimation and uses only distances to the first nearest neighbors. It is similar to the methods of[14] and[15] that were used for outlier detection in large databases. In the nearest neighbor density estimation a cell, often a hypersphere in a d dimensional space, is centered around the training instance z . The volume of this cell is grown until it captures k instances of the training data. Support vector machine with RBF kernel was also used for anomaly detection[16]. For each new instance, the method determines if it falls within the learned region: if it does then the instance is considered to be a target instance, otherwise it is an outlier instance. Similar approach, called Robust SVM, was used for intrusion detection[17].

1.3 Reconstruction methods

Reconstruction methods use prior knowledge about data and make assumptions about the data-generating process to build a classifier. The assumption is that a compact representation of the target data can be obtained that decreases noise influence[6, 18]. The reconstruction methods assume that outlier instances do not satisfy assumptions about the target class distribution. During testing a new instance may have low or large noise component and thus the corresponding low or high reconstruction error, calculated as a distance to the target training data. Users need to choose appropriate thresholds when using these methods. For example, in learning vector quantization[18] and in k -means clustering[6] it is the number of clusters. In self-organizing feature maps[19] it is the dimensionality of the manifold, the number of prototypes per manifold, and the learning rate. In Principal Component Analysis (PCA) [6] it is the mean and basis vectors for each of the subspaces and the noise variance outside of the subspaces. In diabolito networks[20, 21] and auto-encoder networks [22] it is the number of layers and neurons, learning rates, and stopping criteria.

1.4 Rule-based methods

Rule-based methods are of key interest here as the introduced below algorithm is rule-based. Rule-based algorithms generate rules that capture target behavior of a system[23, 24, 25]. An instance that is not covered by any of the rules is considered an outlier. Different techniques generate rules in different ways. Classification techniques such as IREP and RIPPER[26] learn rules from data by adding outlier class data into the training data. Outlier class is artificially generated so a binary classifier (such as RIPPER) can learn the boundaries between the two classes. In [27, 25] supervised outlier detection method was introduced to detect network intrusions while [28] adapted C4.5 algorithm for outlier detection. A similar approach was used in [29] where Query by Bagging method was used.

A similar approach is to use association rule mining algorithm for rule generation, which requires users to specify minimum support and confidence thresholds[30, 31, 32]. The advantage of this approach is that it utilizes the fact that outliers occur very rarely in the data, and can be dealt with by choosing appropriate support threshold to ensure that outliers are not taken into account in the process of rule generation. To ensure that rules correspond to strong patterns the rules with low support values are pruned. An application of this technique for intrusion detection was used in ADAM system[30, 31] and in [33] for intrusion detection embedded on the network interface card. LERAD method [34] generates association rules from data in the form $P(\text{not}W|U)$, which is conditional probability of one subset of attributes taking on a particular set of values (denoted by $\text{not}W$) given that a disjoint subset of attributes takes on a particular set of values (denoted by U). In order to deal with possibly high number of rules that can be generated the authors used sampling and randomization techniques. Similar approach was used for credit card fraud detection and for fraud detection in spacecraft house-keeping data[35]. Observing that an outlying transaction occurs in fewer frequent itemsets when compared with a normal transaction, in [36, 37] an approach similar to an outlier de-

tection algorithm for categorical data sets was proposed. The authors also introduced a measure called Frequent Pattern Outlier Factor that ranked the transactions based on the number of frequent itemsets they occur in. A different approach, based on fuzzy classifiers [38, 39, 40], uses a fuzzy classifier learning algorithm to create rules from data with noise. For instance, fuzzy classifiers eClass and FLEXFIS-Class [41] can be used with different model architectures for solving one-class problems.

2 ONECLASS-DS ALGORITHM

In a classical binary classification problem there are positive and negative instances, representing two classes, which are available for training. When rules for the positive (or negative) class are generated they are found based on comparisons between positive and negative instances. During testing, an instance is assigned to either the positive or the negative class. This simple scheme, however, cannot be used for solving one-class classification problems since only instances representing one Target class are available. Moreover, Target instances may contain noise so they can be either true targets or false outliers (FO); this is illustrated in the left rectangle in Figure 1. Similarly, the Outlier instances (to be identified only during testing) can be either true outliers or false targets; this is illustrated in the right rectangle in Figure 1.

Figure 1. One-class training and test scenarios.

OneClass-DS algorithm generates rules by performing greedy hill-climbing search in a manner similar to the DataSqueezer algorithm [42]. However, DataSqueezer was designed only for binary classification problems and generates rules by comparing the positive class instances with the negative class instances. Thus, OneClass-DS uses four user-specified heuristic parameters to guide the rule generation process to generate rules from one-class data only. They are: Threshold, MinCoverTager, MinAttribute and MaxAttribute. The last two are used for accepting the generated rules only if the total number of selectors in a rule is in the range between MinAttribute and MaxAttribute. The OneClass-DS generated rules are in the format:

IF (Feature x = value a) and (Feature y = value b)... THEN Target

Furthermore, all selectors in a rule must satisfy two conditions: they must cover a minimum number of target instances (MinCoverTarget) and have the summed-up value equal to or larger than the Threshold. The summed-up value of a selector is calculated by multiplying the number of values a feature takes on by the number of times it appears in the target training data. The pseudocode of the OneClass-DS algorithm is shown in Figure 2.

Generate Rules**Input:** *TargetTable, MinAttribute, MaxAttribute, MinCoverTarget, Threshold**Rules* $\leftarrow []$ *While* *TargetTable* $\neq$ empty Create list *L* of selectors, *s*, each selector satisfying two conditions:
 its summed-up-value $\geq$ *Threshold* and its cover $\geq$ *MinCoverTarget* If (list *L* = empty) break; rule *r* $\leftarrow []$ Select *s* from *L* which has the maximum summed-up-value and add it to rule *r*, and delete *s* from *L* While (total number of selectors in rule *r* < *MinAttribute*) and (list *L* not empty) Choose the next selector, *s*, and add it to rule *r*: *r* $\leftarrow r \cup \{s\}$ Delete *s* from *L* If (total number of selectors in rule *r* = *MaxAttribute*) break

End While

 Delete all instances in *TargetTable* covered by rule *r* *Rules* $\leftarrow Rules \cup \{r\}$ *End While***Output:** *Rules* (for *Target class*)

Figure 2. Pseudocode of the OneClass-DS algorithm.

The process of generating rules is repeated several times with different settings of the four parameters. To choose the best model from the generated ones a Target Rate (sensitivity) measure is used in the following way.

$$TargetRate = \frac{TrueTargets(TT)}{TotalTargets(TT + FO)} = Sensitivity$$

Although target data instances should consist only of true target instances, some of the target instances may be false outliers (FO), as illustrated in Figure 1. To identify the FO instances we use 10-FCV as follows. The model is built from 9/10 of the target data but when testing it on the 1/10 of the remaining data, some of the instances that are not recognized as true targets and are identified as false outliers (FO). At the end of the 10-FCV process all the instances that were recognized as FO (in ten runs) are subtracted from the Target data instances. The model that has the largest Target Rate is then chosen as the best one. The pseudocode for building the model is shown in Figure 3, while Figure 4 illustrates the process.

Input: Data set *D* with *N* classes (from 1 to *N*)*Build model:* Choose class *i* as a target class *T* Denote all instances in class *i* as belonging to class *T*, and all other instances as belonging to class *O* – this forms data set *D*₁ Divide data set *D*₁ into *k* parts *P*_{*j*} (each part may include instances belonging to class *T* and/or class *O*) For each part *P*_{*j*} Combine (*K*-1) parts (except for part *P*_{*j*}) to form data set *D*₂ Delete all instances of class *O* from *D*₂ (becomes one-class problem) Build model for the reduced data set *D*₂ (now contains only class *T* instances) Test on part *P*_{*j*} (may contain instances belonging to *T* and *O*) Choose the best model and test it on *D*₁

Figure 3. Pseudocode for building OneClass-DS model.

Figure 4. Building and choosing the best model.

Computational complexity of OneClass-DS algorithm is approximately $O(RKN \log N)$, where R is the number of rules in Rules, N is the number of instances, and K is the number of attributes.

To summarize, OneClass-DS is simple, greedy, effective rule learner that generates a set of production rules. The rules are compact, easy to comprehend, and have accuracy on par with rules (models) generated by other one-class rule-based algorithms. The main advantages of the algorithm are its high efficiency and robustness to missing data (shown later). In general, using higher values of the four algorithm's parameters results in generation of "stronger" rules that may predict new test instances with higher accuracy, while lower values of the parameters result in generating "weaker" rules.

Data shown in Table 1 are used to illustrate the working of the algorithm.

TABLE 1. TARGET DATA.

F1	F2	F3	F4	Class
1	1	1	1	T
1	2	1	2	T
1	4	2	2	T
1	4	4	1	T
4	5	5	4	T

Suppose we choose the following values of the parameters: $MinCoverTarget=25\%$, $Threshold=0$, $MaxAttribute=MinAttribute=2$. The summed-up values (v_{ij}) are calculated as shown in Table 2, where i denotes a feature, j is feature number, and $(v_{ij}) = (\text{the number of times a value appears in target data}) \times (\text{the number of values the feature takes on})$.

TABLE 2. CALCULATION OF THE SUMMED-UP VALUES FOR FEATURES SHOWN IN TABLE 1.

Feature	Number of values each feature takes on	Summed-up values
F1	2 values {1,4}	$v_{11}=4 \times 2$, $v_{41}=1 \times 2$
F2	4 values {1,2,4,5}	$v_{12}=1 \times 4$, $v_{22}=1 \times 4$, $v_{42}=2 \times 4$, $v_{52}=1 \times 4$
F3	4 values {1,2,4,5}	$v_{13}=2 \times 4$, $v_{23}=1 \times 4$, $v_{43}=1 \times 4$, $v_{53}=1 \times 4$
F4	3 values {1,2,4}	$v_{14}=2 \times 3$, $v_{24}=2 \times 3$, $v_{44}=1 \times 3$

Notice that F1, F2 and F3 have the same maximal summed-up values, namely, $v_{11} = v_{42} = v_{13} = 8$. We choose the first feature, F1, and add selector "F1=1" to start the process of generating the first rule: **IF F1=1** Table 3 shows instances covered (all but one) by F1=1. We continue calculating summed-up values for features F2 through F4, shown in Table 4, in order to choose the next selector for the first rule (because $MinAttribute=2$).

TABLE 3. INSTANCES COVERED BY F1=1.

F1	F2	F3	F4	Class
1	1	1	1	T
1	2	1	2	T
1	4	2	2	T
1	4	4	1	T

TABLE 4. SUMMED-UP VALUES FOR ALL FEATURES BUT F1.

Feature	Total number of values	Summed-up values
F2	3 values {1,2,4}	$v_{12}=1 \times 4$, $v_{22}=1 \times 4$, $v_{42}=2 \times 4$
F3	2 values {1,2,4}	$v_{13}=2 \times 4$, $v_{23}=1 \times 4$, $v_{43}=1 \times 4$
F4	2 values {1,2}	$v_{14}=2 \times 3$, $v_{24}=2 \times 3$

Features F2 and F3 have the same max summed-up values, namely, $v_{42}=v_{13}=8$. Thus, we choose F2 with value 4 as the next selector and finish (because $MaxAttribute=2$) generation of the first rule:

IF F1=1 AND F2=4

This rule (IF F1=1 AND F2=4 THEN Class=T) covers 40% of Target data (instances 3 and 4 in Table 1) so we need to continue rule generation. After all instances covered by the first rule are deleted the reduced data set is shown in Table 5.

TABLE 5. REMAINING TARGET DATA.

F1	F2	F3	F4	Class
1	1	1	1	T
1	2	1	2	T
4	5	5	4	T

We repeat the process and the second generated rule is: **IF F1=1 AND F3=1**

This rule (IF F1=1 AND F3=1 THEN Class=T) also covers 40% of Target data (instances 1 and 2 in Table 1).

After deleting two instances covered by the second rule we are left with just one data instance shown in Table 6.

TABLE 6. TRAINING DATA INSTANCES AFTER DELETING INSTANCES COVERED BY THE FIRST TWO RULES.

F1	F2	F3	F4	Class
4	5	5	4	T

Because $MinCoverTarget=25\%$ (i.e., we allowed for 25% of target instances not to be covered) the OneClass-DS algorithm converges and stops generating rules.

In the next section, in order to perform comparisons of the results we use performance measures typically used in evaluating one-class algorithms, namely, the FAR and IPR measures [11]:

$$FAR = \frac{FalseOutliers(FO)}{TrueOutliers(TO) + FalseOutliers(FO)}$$

$$IPR = \frac{FalseTargets(FT)}{TrueTargets(TT) + FalseTargets(FT)}$$

where, FAR (False Alarm Rate) specifies the number of true Target instances incorrectly identified as outliers (false negatives), and IPR (Impostor Pass Rate) specifies the number of outlier instances that are wrongly classified (false positives) as belonging to the Target class. A higher FAR corresponds to a lower IPR and vice versa. We will also use measures of Precision and Accuracy:

$$Precision = \frac{TrueTargets(TT)}{TrueTargets(TT) + FalseTargets(FT)}$$

$$Accuracy = \frac{TrueTargets(TT) + TrueOutliers(TO)}{Totalinstances\ in\ test\ data\ (TT + FO + TO + FT)}$$

3 RESULTS

The OneClass-DS algorithm was implemented as a function of the WEKA software to allow for consistent testing and comparisons with other algorithms. We conducted four different types of experiments to compare OneClass-DS results with results of other algorithms.

For all four types of experiments the results are reported using a modified 10-fold cross-validation procedure. Namely, after the data are divided into ten parts, all instances not belonging to Target class are deleted from the nine parts used for training (representing the Target class). For example, if the data had three classes (One, Two and Three), and if the Target was chosen as class One, all the instances belonging to classes Two and Three were deleted from the nine parts used for training. Similarly, models for classes Two and Three are generated, and the average of the three runs is reported as the final result.

In the first experiment we compare OneClass-DS with five WEKA algorithms on 11 data sets, in the WEKA environment. Second, OneClass-DS is compared with three algorithms: OneClass SVM[43, 44], OCC-Gauss and OCC-EM using the results published in [11] on the same data sets. Third, we compare performance of OneClass-DS on data with missing values with REP-Tree, Decision Stump, and Random Tree algorithms, again in the WEKA environment. Fourth, OneClass-DS is used on highly unbalanced multi-class data, which was done by converting the multi-class problem into several one-class problems.

Before the experiments are performed all continuous features are discretized using CAIM algorithm[45] as follows. Suppose that a data set consists of M instances and that each instance belongs to only one of the S classes. F denotes continuous feature that is divided into n discrete intervals bounded by the pairs of num-

bers (boundary points) $D: \{[d_0, d_1], (d_1, d_2], \dots, (d_{n-1}, d_n]\}$ where d_0 is the minimal value and d_n is the maximal value of attribute F , after arranging the values in the ascending order. These values constitute the boundary set $\{d_0, d_1, d_2, \dots, d_{n-1}, d_n\}$ for discretization D , which gives rise to a quanta matrix shown in Table 7.

TABLE 7. NORMAL QUANTA MATRIX.

Class	Interval					Class Total
	$[d_0, d_1]$		$(d_{r-1}, d_r]$		$(d_{n-1}, d_n]$	
C_1	q_{11}		q_{1r}		q_{1n}	M_{1+}
.....						
C_i	q_{i1}		q_{ir}		q_{in}	M_{i+}
.....						
C_S	q_{s1}		q_{sr}		q_{sn}	M_{S+}
Interval Total	M_{+1}		M_{+r}		M_{+n}	M

In Table 7, q_{ir} is the total number of continuous values belonging to the i^{th} class that are within interval $(d_{r-1}, d_r]$. M_{i+} is the total number of instances belonging to the i^{th} class, and M_{+r} is the total number of continuous values of feature F that are within the interval $(d_{r-1}, d_r]$, for $i=1, 2, \dots, S$ and, $r=1, 2, \dots, n$. For one-class problem we build a quanta matrix with a dummy instance for the outlier class in each interval $(d_{r-1}, d_r]$ as shown in Table 8.

TABLE 8. QUANTA MATRIX IN ONECLASS-DS

Class	Interval					Class Total
	$[d_0, d_1]$		$(d_{r-1}, d_r]$		$(d_{n-1}, d_n]$	
Target class	q_{11}		q_{1r}		q_{1n}	$q_{11}+..+q_{1r}+..+q_{1n}$
Outlier class	1		1		1	n
Interval Total	$q_{11}+1$		$q_{1r}+1$		$q_{1n}+1$	$q_{11}+..+q_{1r}+..+q_{1n}+ n$

Using this quanta matrix, we discretize feature F using the CAIM algorithm. Note that in experiments I and III we have re-run the experiments and tuned the values of parameters for each algorithm used in comparisons (to achieve their best results) for each data set. Only in experiment II we used the published results [9], assuming that the authors showed their best results as we did not have enough information to re-run them ourselves.

3.1 Experiments I

In these experiments we compare OneClass-DS with five essentially binary algorithms from WEKA, namely, REP-Tree, J48, Random Tree, Decision Stump, and Random Forest. However, they can be used for one-class learning via WEKA's OneClassClassifier[11] software, which creates and uses artificial data set as "the other" class to allow for generation of "one-class" rules. 11 data sets from the UCI repository <http://www.ics.uci.edu/~mllearn/MLRepository.html> are used. Four of them have nominal feature values; the number of instances ranges from 151 to 12,960, the number of features from 5 to 23, and the number of classes from 2 to 5. Details of the data are shown in Table 9.

TABLE 9. DATA SETS USED IN EXPERIMENTS.

Name	Description	# of Instances	# of features	# of classes
Car	Car Evaluation-Nominal	1728	7	4
Mrm	Mushroom -Nominal	8124	23	2
Nur	Nursery-Nominal	12960	9	5
Bre	Breast Cancer	699	10	2
Sca	Balance Scale Weight	625	5	3
Bup	BUPA livers disorder	345	7	2
Hea	Heart Dataset	270	14	2
Pim	Pima Indians Diabetes Database	768	9	2
Spc	SPECT heart data-Nominal	267	23	2
Cmc	Contraceptive Method Choice	1473	10	3
Tae	Teaching Assistant Evaluation	151	6	3

Evaluating performance of OneClass-DS on a data set with N classes is performed by treating each class as the Target class, and all other classes as the “outlier” class. The results are shown in Table 10. The four parameters used in OneClass-DS algorithm, namely, *Threshold*, *MaxAttribute*, *MinAttribute*, and *MinCoverTarget* are optimized to choose the best model – one that maximizes the Target Rate.

Figure5 shows impacts of varying one parameter at a time while having the other three parameters fixed. It is shown only for the Heart data to illustrate the process.

Figure5. Influence of changing the four parameters on the Target rate, Precision, IPR and FAR, on the Heart data set.

TABLE 10. RESULTS OF ONECLASS-DS ALGORITHM ON THE 11 DATA SETS.

	Class	# of targets	# of outliers	Time to build model (sec)	# of rules	Target rate	Accuracy	Precision	FAR	IPR
Car	unacc	1210	518	0.02	11	0.97	0.91	0.91	0.09	0.09
	acc	384	1344	0.01	11	0.80	0.55	0.30	0.11	0.70
	vgood	65	1663	0.01	1	0.02	0.95	0.05	0.04	0.95
	good	69	1659	0.01	2	0.42	0.88	0.15	0.03	0.85
	Average Weight					0.87	0.82	0.60	0.05	0.40
Mrm	Class p	3916	4208	0.22	31	0.89	0.91	0.91	0.10	0.09
	Class e	4208	3916	0.34	32	0.96	0.95	0.95	0.05	0.05
	Average Weight					0.93	0.93	0.93	0.07	0.07
Nur	recom	2	12958							
	prior	4266	8694	0.48	139	0.98	0.60	0.45	0.02	0.55
	not_recom	4320	8640	0.34	54	0.96	0.32	0.33	0.82	0.67
	very_recom	328	12632	0.12	27	0.89	0.80	0.10	0.00	0.90
	spec_prior	4044	8916	0.45	124	0.99	0.66	0.48	0.01	0.52
	Average Weight					0.98	0.59	0.38	0.02	0.62
Bre	Class 2	458	241	0.04	26	0.95	0.93	0.95	0.10	0.05
	Class 4	241	458	0.02	7	0.98	0.98	0.98	0.01	0.02
	Average Weight					0.96	0.96	0.96	0.04	0.04
Sca	Class B	49	576	0.1	13	0.67	0.66	0.14	0.04	0.86
	Class R	288	337	0.01	29	0.88	0.69	0.61	0.16	0.39
	Class L	288	337	0.01	28	0.91	0.69	0.61	0.14	0.39
	Average Weight					0.88	0.68	0.51	0.10	0.49
Bup	Class 1	145	200	0.03	1	0.94	0.46	0.43	0.28	0.57
	Class 2	200	145	0.05	1	0.95	0.59	0.59	0.40	0.41
	Average Weight					0.94	0.53	0.52	0.33	0.48
Hea	Class 1	150	120	0.05	4	0.71	0.69	0.73	0.35	0.27
	Class 2	120	150	0.05	4	0.67	0.68	0.63	0.28	0.37
	Average Weight					0.69	0.69	0.69	0.31	0.31
Pim	Class 0	500	268	0.4	7	0.99	0.66	0.66	0.30	0.34
	Class 1	268	500	0.39	7	0.97	0.35	0.35	0.54	0.65
	Average Weight					0.99	0.50	0.50	0.43	0.50
Spc	Class 0	55	212	0.1	2	0.87	0.45	0.26	0.09	0.74
	Class 1	212	55	0.1	4	0.96	0.76	0.79	1.00	0.21
	Average Weight					0.94	0.61	0.57	0.17	0.43
Cmc	Class 1	629	844	0.02	9	0.99	0.43	0.43	0.44	0.57
	Class 2	333	1140	0.01	10	0.99	0.25	0.23	0.10	0.77
	Class 3	511	962	0.02	8	0.99	0.37	0.35	0.13	0.65
	Average Weight					0.99	0.35	0.34	0.15	0.66
Tae	Class 1	49	102	0.17	7	0.94	0.42	0.35	0.15	0.65
	Class 2	50	101	0.01	7	0.96	0.43	0.36	0.11	0.64
	Class 3	52	99	0.01	7	0.96	0.42	0.37	0.13	0.63
	Average Weight					0.95	0.42	0.36	0.13	0.64

Comparison of results,in terms of IPR and FAR measures, is shown in Table 11 while Table 12 shows it in terms of accuracy.

TABLE 11. COMPARISON OF ONECLASS-DS WITH 5 OTHER ALGORITHMS IN TERMS OF IPR AND FAR MEASURES; THE BEST RESULTS ARE SHOWN IN BOLD.

Data set	IPR					
	REP-Tree	J48	Random Tree	Decision Stump	Random Forest	OneClass-DS
Car	0.28	0.20	0.49	0.20	0.43	0.40
Mrm	0.02	0.04	0.01	0.30	0.00	0.07
Nur	0.62	0.62	0.62	0.58	0.58	0.62
Bre	0.50	0.50	0.42	0.10	0.25	0.04
Sca	0.62	0.63	0.64	0.53	0.63	0.49
Bup	0.49	0.49	0.50	0.49	0.50	0.48
Hea	0.41	0.47	0.39	0.41	0.37	0.31
Pim	0.49	0.50	0.49	0.49	0.48	0.50
Spc	0.34	0.36	0.40	0.34	0.35	0.43
Cmc	0.64	0.65	0.65	0.63	0.63	0.66
Tae	0.65	0.67	0.65	0.66	0.64	0.64
Data set	FAR					
	REP-Tree	J48	Random Tree	Decision Stump	Random Forest	OneClass - DS
Car	0.03	0.03	0.04	0.04	0.03	0.05
Mrm	0.04	0.17	0.04	0.14	0.17	0.07
Nur	0.05	0.06	0.02	0.05	0.04	0.02
Bre	0.53	0.50	0.10	0.11	0.12	0.04
Sca	0.16	0.18	0.18	0.13	0.16	0.10
Bup	0.44	0.45	0.49	0.41	0.47	0.33
Hea	0.33	0.46	0.21	0.23	0.19	0.31
Pim	0.40	0.45	0.42	0.46	0.37	0.43
Spc	0.18	0.16	0.20	0.19	0.16	0.17
Cmc	0.18	0.24	0.23	0.17	0.17	0.15
Tae	0.28	0.33	0.26	0.24	0.19	0.13

TABLE 12. COMPARISON OF ONECLASS-DS WITH FIVE OTHER ALGORITHMS IN TERMS OF ACCURACY.

Data Set	REP-Tree	J48	Random Tree	Decision Stump	Random Forest	OneClass-DS
Car	0.89	0.92	0.76	0.92	0.8	0.82
Mrm	0.97	0.89	0.97	0.75	0.9	0.93
Nur	0.61	0.6	0.68	0.67	0.66	0.59
Bre	0.5	0.5	0.63	0.89	0.8	0.96
Sca	0.47	0.45	0.42	0.63	0.45	0.68
Bup	0.51	0.51	0.5	0.52	0.51	0.53
Hea	0.61	0.54	0.65	0.63	0.68	0.69
Pim	0.52	0.51	0.52	0.51	0.53	0.5
Spc	0.71	0.7	0.65	0.71	0.71	0.61
Cmc	0.44	0.42	0.41	0.47	0.44	0.35
Tae	0.41	0.42	0.42	0.39	0.44	0.42

Recall, that the results shown in Table 10 were generated by OneClass-DS algorithm using only Target class instances, and that the best model was chosen by optimizing the target rate.

Figure 6 shows graphical comparison (for just one of the runs of the 10-FCV) of Target Rate and Precision while changing the *MaxAttribute* from 1 to 9 (or 12 or 15). It is done on the Cmc, Bre, Mrm, and Spc data sets and compared with REP Tree, Random Tree, Decision Stump, and Random Forest algorithms. Precision (red line) and Target Rate (blue line) are shown for different *MaxAttribute* values, i.e., the number of features to be used in the rules. The best results in terms of Target Rate (indicated as blue triangles) and Precision (indicated as red triangles) for the best performing (from among the five) algorithm are also shown in Figure 6.

In case of the Cmc data (upper left graph) Random Forest gave the best result with precision of 0.27 (red triangle) and target rate of 0.91 (blue triangle). OneClass-DS for *MaxAttribute*=8 has precision of 0.38 and target rate of 0.45. Note that if *MaxAttribute* was chosen to be 6 it would obtain precision higher than Random Forest but at the expense of Target Rate. Upper right graph for Bre data shows that Decision Stump performed best with precision of 0.95 and target rate of 0.88. OneClass-DS with *MaxAttribute*=7 has precision of 0.86 and target rate of 0.89. Notice that *MaxAttribute* parameter influences both precision and target rate and that we choose the values of precision and target rate being both relatively high and close to the intersection point of the two curves. Similarly for Mrm data (lower left graph), for which the best was Random Tree with 0.97 precision and 0.95 target rate, OneClass-DS with *MaxAttribute*=7 has precision of 0.89 and target rate of 0.95. In case of Spc data set (lower right graph) the best was REP Tree with precision=0.39 and target rate=0.78, while OneClass-DS with *MaxAttribute*=10 has precision of 0.51 and target rate of 0.60.

Figure 6. Comparison of OneClass-DS results on four data sets, with Random Forest, Decision Stump, Random Tree and REP Tree algorithms, in terms of Precision and Target rate (y-axis) vs. its MaxAttribute (x-axis) value.

To check statistical significance of the obtained results Wilcoxon and Friedman tests are performed. First, it is done in terms of IPR and FAR measures (using Table 11 results) and, second, in terms of accuracy (using Table 12 results). The results for the two tests are shown in Table 13. Notice that in terms of Wilcoxon test using accuracy and IPR, OneClass-DS performs on par with other algorithms. In terms of the FAR measure, OneClass-DS performs better than J48 and Decision Stump, and on par with the remaining three algorithms. Friedman tests show that OneClass-DS performs at the same level as the other algorithms in terms of all three: accuracy, IPR, and FAR.

TABLE 13. WILCOXON AND FRIEDMAN TESTS RESULTS FOR COMPARING ONECLASS-DS WITH FIVE ALGORITHMS ON 11 DATA SETS

WILCOXON PAIREDTEST			
Accuracy $\alpha = 0.05, N = 11$ The null-hypothesis that a given pair of algorithms performs equally well is rejected when $T \leq 11$ (in 0 cases).In other words, OneClass-DS performs on par with other algorithms.		IPR $\alpha = 0.05, N = 11$ The null-hypothesis that a given pair of algorithms performs equally well is rejected when $T \leq 11$ (in 0 cases). In other words, OneClass-DS performs on par with other algorithms.	
Pair of compared Algorithms	R+	R-	T=min(R+, R-)
OneClass-DSvs. REP-Tree	34	32	32
OneClass-DSvs. J48	26.5	39.5	26.5
OneClass-DSvs. Random Tree	30	36	30
OneClass-DSvs. Decision Stump	35.5	30.5	30.5
OneClass-DSvs. Random Forest	33	33	33
FAR $\alpha = 0.05, N = 11$ The null-hypothesis that a given pair of algorithms performs equally well is rejected when $T \leq 11$ (in 2 cases). In other words, OneClass -DS performs better than J48 and Decision Stump algorithms.			
Pair of compared Algorithms	R+	R-	T=min(R+, R-)
OneClass-DS vs. REP-Tree	13	53	13
OneClass-DS vs. J48	3	63	3
OneClass-DS vs. Random Tree	19.5	46.5	19.5
OneClass-DS vs. Decision Stump	11	55	11
OneClass-DS vs. Random Forest	20	46	20
FRIEDMAN TEST			
Accuracy $\alpha = 0.05, N = 11$ The null hypothesis that all algorithms perform at the same level is accepted because the calculated Fr value (4.3) is smaller than the critical value (11.07). In other words, OneClass-DS performs on par with other algorithms.		IPR $\alpha = 0.05, N = 11$ The null hypothesis that all algorithms perform at the same level is accepted because the calculated Fr value (8.25) is smaller than the critical value (11.07). In other words, OneClass -DS performs on par with other algorithms.	
FAR $\alpha = 0.05, N = 11$ The null hypothesis that all algorithms perform at the same level is accepted because the calculated Fr value (10.58) is smaller than the critical value (11.07). In other words, OneClass -DS performs on par with other algorithms.			

3.2 Experiments II

These experiments compare OneClass-DS with three algorithms: One-Class SVM[43, 44], and two OCC (One-Class Classification) algorithms: OCC-Gauss and OCC-EM. The results are shown in Table 14. Results for One-Class SVM and OCC algorithms are repeated here after [11]. In addition, we show comparison of time and average number of rules needed to construct a model. As expected, OneClass SVM was the fastest while OCC-EM was the slowest. Notice that the total number of rules generated by OneClass-DS was smaller than ones by OCC-Gauss and OCC-EM algorithms on all data sets but Ecoli and Glass.

TABLE 14. COMPARISON OF ONECLASS-DS WITH ONECLASS-SVM, OCC-GAUSS, AND OCC-EM ON NINE DATA SETS.

Data set	# of features	# of instances	# of classes	FAR				IPR			
				OneClass-DS	OneClass SVM	OCC Gaussian	OCC EM	OneClass-DS	OneClass SVM	OCC Gaussian	OCC EM
Diabetes	8	768	2	0.264	0.111	0.098	0.109	0.482	0.514	0.857	0.779
Ecoli	7	336	8	0.241	0.137	0.129	0.136	0.111	0.068	0.088	0.083
Glass	9	214	7	0.064	0.154	0.147	0.18	0.473	0.412	0.434	0.331
Heart-statlog	13	270	2	0.295	0.122	0.14	0.141	0.46	0.624	0.507	0.504
Ionosphere	34	351	2	0.211	0.128	0.15	0.169	0.458	0.738	0.732	0.697
Iris	4	150	3	0.096	0.12	0.125	0.137	0.13	0.073	0.076	0.077
Sonar	60	208	2	0.282	0.12	0.123	0.163	0.455	0.705	0.815	0.751
Vehicle	18	846	4	0.208	0.103	0.109	0.13	0.342	0.629	0.645	0.494
Waveform	21	5000	3	0.126	0.103	0.075	0.11	0.451	0.307	0.411	0.354
				Time to build a model (sec)				Average # of rules			
				OneClass-DS	OneClass SVM	OCC Gaussian	OCC EM	OneClass-DS	OneClass SVM	OCC Gaussian	OCC EM
				0.52	0.37	3.89	13.9	11	N/A	14	27
				0.82	0.13	1.18	0.62	8	N/A	3	5
				0.38	0.09	0.25	1.41	7	N/A	5	5
				0.1	0.06	0.58	5.23	5	N/A	4	8
				9.9	0.08	1.65	30.64	5	N/A	3	5
				0.1	0.02	0.13	0.26	4	N/A	7	14
				10.3	0.13	5.71	21.44	7	N/A	11	19
				1.5	0.68	0.83	13.4	20	N/A	25	21
				77	4.07	47.5	106.17	12	N/A	27	71

The Wilcoxon and Friedman tests are performed using time to build a model, IPR and FAR measures (using Table 14 results). The results are shown in Table 15. Notice that in terms of Wilcoxon test, using the IPR measure, OneClass-DS performs on par with the other algorithms. Using the FAR measure OneClass-DS performs on par with the OneClass SVM and OCC-EM. Wilcoxon test performed using time to build the model shows that OneClass-DS performed on par with OCC-Gauss.

Friedman tests show that OneClass-DS performs on par with the other three algorithms in terms of IPR, FAR and time to build a model measures.

TABLE 15. WILCOXON AND FRIEDMAN TESTS RESULTS FOR COMPARING ONECLASS-DS WITH THREE ALGORITHMS ON 9 DATA SETS.

WILCOXON PAIREDTEST																																				
Time to build model $\alpha = 0.05, N = 9$ The null-hypothesis that a given pair of algorithms performs equally well is rejected when $T \leq 6$ (in 2 cases). In other words, OneClass-DS performs on par only with OCC- Gauss algorithm.				IPR $\alpha = 0.05, N = 9$ The null-hypothesis that a given pair of algorithms performs equally well is rejected when $T \leq 6$ (in 0 cases). In other words, OneClass -DS performs on par with other algorithms.																																
<table><tr><td>Pair of compared Algorithms</td><td>R+</td><td>R-</td><td>T=min(R+, R-)</td></tr><tr><td>OneClass-DSvs. OneClass-SVM</td><td>45</td><td>0</td><td>0</td></tr><tr><td>OneClass-DSvs. OCC-Gauss</td><td>31</td><td>14</td><td>14</td></tr><tr><td>OneClass-DSvs. OCC-EM</td><td>2</td><td>43</td><td>2</td></tr></table>	Pair of compared Algorithms	R+	R-	T=min(R+, R-)	OneClass-DSvs. OneClass-SVM	45	0	0	OneClass-DSvs. OCC-Gauss	31	14	14	OneClass-DSvs. OCC-EM	2	43	2	<table><tr><td>Pair of compared algorithms</td><td>R+</td><td>R-</td><td>T=min(R+, R-)</td></tr><tr><td>OneClass-DSvs. OneClass-SVM</td><td>14</td><td>31</td><td>14</td></tr><tr><td>OneClass-DSvs. OCC-Gauss</td><td>11</td><td>34</td><td>11</td></tr><tr><td>OneClass-DSvs. OCC-EM</td><td>13</td><td>32</td><td>13</td></tr></table>				Pair of compared algorithms	R+	R-	T=min(R+, R-)	OneClass-DSvs. OneClass-SVM	14	31	14	OneClass-DSvs. OCC-Gauss	11	34	11	OneClass-DSvs. OCC-EM	13	32	13
Pair of compared Algorithms	R+	R-	T=min(R+, R-)																																	
OneClass-DSvs. OneClass-SVM	45	0	0																																	
OneClass-DSvs. OCC-Gauss	31	14	14																																	
OneClass-DSvs. OCC-EM	2	43	2																																	
Pair of compared algorithms	R+	R-	T=min(R+, R-)																																	
OneClass-DSvs. OneClass-SVM	14	31	14																																	
OneClass-DSvs. OCC-Gauss	11	34	11																																	
OneClass-DSvs. OCC-EM	13	32	13																																	
FAR $\alpha = 0.05, N = 9$ The null-hypothesis that a given pair of algorithms performs equally well is rejected when $T \leq 6$ (in 1 cases). In other words, OneClass -DS performs on par with OneClass-SVM and OCC-EM algorithms.																																				
<table><tr><td>Pair of compared Algorithms</td><td>R+</td><td>R-</td><td>T=min(R+, R-)</td></tr><tr><td>OneClass-DS vs. OneClass-SVM</td><td>39</td><td>7</td><td>7</td></tr><tr><td>OneClass-DS vs. OCC-Gauss</td><td>40</td><td>5</td><td>5</td></tr><tr><td>OneClass-DS vs. OCC-EM</td><td>37</td><td>8</td><td>8</td></tr></table>					Pair of compared Algorithms	R+	R-	T=min(R+, R-)	OneClass-DS vs. OneClass-SVM	39	7	7	OneClass-DS vs. OCC-Gauss	40	5	5	OneClass-DS vs. OCC-EM	37	8	8																
Pair of compared Algorithms	R+	R-	T=min(R+, R-)																																	
OneClass-DS vs. OneClass-SVM	39	7	7																																	
OneClass-DS vs. OCC-Gauss	40	5	5																																	
OneClass-DS vs. OCC-EM	37	8	8																																	
FRIEDMAN TEST																																				
Time to build model $\alpha = 0.05, N = 9$ The null hypothesis that all algorithms perform at the same level is rejected because the calculated Fr value (21.13) is larger than the critical value (12.59).				IPR $\alpha = 0.05, N = 9$ The null hypothesis that all algorithms perform at the same level is accepted because the calculated Fr value (9.93) is smaller than the critical value (12.59). In other words, OneC-lass -DS performs on par with other algorithms.																																
FAR $\alpha = 0.05, N = 9$ The null hypothesis that all algorithms perform at the same level is accepted because the calculated Fr value (3.8) is smaller than the critical value (12.59). In other words, OneClass -DS performs on par with other algorithms.																																				

3.3Experiments III

The purpose of these experiments is to show how OneClass-DS performs on data with missing values, and later compare it with REP-Tree, Decision Stump, and RandomTree algorithms. The experiments are performed on four data sets from the UCI data repository, and are shown in Table 16. The number of instances ranges from 57 to 32,561 and the number of features from 6 to 17. Note that these data sets are very different in terms of the number of missing values, ranging from 48 to 1843. The results are shown in Table 17.

TABLE 16. DATA SETS WITH MISSING VALUES.

Data set	Description	# of instances	# of missing values	# of features	# of classes
Vot	US Congressional Voting	435	104	17	2
Adt	Dataset Adult From UCI	32561	1843	15	2
Lab	Labor negotiations in Canada	57	48	17	2
Mam	Mammographic Mass	961	76	6	2

TABLE 17. ONECLASS-DS RESULTS ON DATA WITH MISSING VALUES.

Data Set	Classes	# of targets	# of outliers	Time to build model (sec)	# of rules	Target rate	Accuracy	Precision	FAR	IPR
Vot	Democrat	267	168	0.02	39	0.80	0.78	0.84	0.30	0.16
	Republican	168	267	0.01	20	0.85	0.92	0.95	0.09	0.05
	Average			0.02	30	0.82	0.85	0.89	0.19	0.11
Adt	Less50K	24720	7841	1253	158	0.83	0.68	0.77	0.72	0.23
	Over50K	7841	24720	1225	61	0.72	0.72	0.45	0.11	0.55
	Average			1239	110	0.77	0.70	0.61	0.41	0.39
Lab	Bad	20	37	0.01	6	0.70	0.74	0.61	0.18	0.39
	Good	37	20	0.01	2	0.54	0.70	1.00	0.46	0.01
	Average			0.01	4	0.62	0.72	0.80	0.32	0.20
Mam	0	516	445	0.03	4	0.59	0.72	0.83	0.35	0.17
	1	445	516	0.02	4	0.49	0.74	0.90	0.31	0.10
	Average			0.03	4	0.54	0.73	0.86	0.33	0.14

Figure 7 illustrates influence of *MaxAttribute* (in a similar way as is shown in Figure 6) on precision and target rate, on the four data sets. To choose the best model for each data set we ran 10-FCV. Upper left graph (Vot data) indicates acceptable model with *MaxAttribute*=8, which corresponds to the intersection of Target rate and Precision curves. For Mam data (upper right graph) *MaxAttribute*=4 gives a good model. For Lab data (lower left graph) it is *MaxAttribute*=3, and for Adt data (lower right graph) it is *MaxAttribute*=11.

Performance comparison with the other three algorithms is shown in Table 18. As shown there, OneClass-DS is better than other algorithms in terms of both FAR and Accuracy measures. In terms of IPR measure, OneClass-DS is only better on Lab and Mam data set. Note that in these experiments we only change value of *MaxAttribute*. In case of Vot and Adt data sets we can choose *MaxAttribute* values as 8 for Vot data and 11 for Adt data – at the intersection point of Target Rate and Precision curves. In case of Mam and Lab data sets, we have to choose a value for *MaxAttribute* to have some balance between Target Rate and Precision measures. For example, we can choose *MaxAttribute*=3 for Lab data and have Target Rate=0.87 and Precision=0.8 (and get better IPR, FAR and Accuracy – See Table 18). Similarly with Mam data, we choose *MaxAttribute*=4 with Target Rate=0.78 and Precision=0.59 instead of choosing value of 5 (higher Precision but lower Target Rate). Results for the selected *MaxAttribute* values are shown in Table 18.

Because of the small number of missing value data sets only the Friedman test ($\alpha = 0.05$, $N = 4$) could be used: the calculated *Fr* values (5.7 for FAR and 4.5 for accuracy) are larger than the critical value of 2.569 so the null hypothesis that all algorithms perform on par is rejected. OneClass-DS gave better results (as shown in Table 18) than the three algorithms in terms of accuracy and FAR measures; in terms of the IPR measure all algorithms performed at the same level.

Figure 7. Possible outcomes of a test in terms of Precision and Target Rate (y-axis) vs. MaxAttribute (x-axis).

TABLE 18. COMPARISON OF ONECLASS-DS WITH REP-TREE, DECISION STUMP AND RANDOM TREE ON FOUR MISSING VALUE DATA SETS.

Data set	FAR				IPR			
	OneClass-DS	REP-Tree	Decision Stump	Random Tree	OneClass-DS	REP-Tree	Decision Stump	Random Tree
Vote	0.19	0.17	0.21	0.54	0.11	0.26	0.10	0.38
Adt	0.41	0.77	0.70	0.75	0.39	0.23	0.19	0.24
Lab	0.32	0.36	0.34	0.33	0.20	0.35	0.43	0.47
Mam	0.33	0.54	0.38	0.53	0.14	0.45	0.47	0.45
Accuracy								
	OneClass-DS	REP-Tree	Decision Stump	Random Tree				
	0.85	0.77	0.85	0.55				
	0.70	0.50	0.58	0.51				
	0.72	0.64	0.64	0.63				
	0.73	0.50	0.57	0.51				

3.4 Experiment IV

This experiment is performed to show that OneClass-DS is able to generate good results even when used on highly unbalanced multi-class data. It is done by converting the multi-class data into several one-class problems. We use the Pen-Based Recognition of Handwritten Digits data set which was used in [46], in short called Pen Digits, from UCI data repository. It consists of hand-written digits (0 to 9) collected from 44 writers using a pressure-sensitive tablet for coding digits when they wrote 250 digits at random within boxes of size 500x500 pixels, for a total of 10,992 hand written digits. After preprocessing, the data are described by only 16 attributes, in the range 1-100. The second column in Table 19 shows the number of samples collected for each digit.

To solve this multi-category problem as one-class problem we transform the data into 10 one-class problems in the following way. Each class is treated in turn not as a Target class but as an Outlier class (because it has very small number of instances) while the rest of the classes are treated as the Target class. The rules are thus generated for the Outlier classes. During testing, if the rule does not cover an instance then it is recognized as belonging to the Target class. Table 19 shows that OneClass-DS generated acceptable results in terms of Accuracy and FAR measures. In fact, the accuracies are comparable to the accuracies (about 97%) reported in [46], where the authors used k-nn classifier on the same data.

TABLE 19. RESULTS ON THE PEN DIGITS DATA SET.

Class name	# of targets	# of outliers	# of rules	Time to build model (sec)	Target rate	Accuracy	Precision	FAR	IPR
0	1143	9849	36	3.8	0.70	0.96	0.89	0.03	0.11
1	1143	9849	40	3.7	0.60	0.93	0.70	0.05	0.30
2	1144	9848	38	4.4	0.69	0.95	0.82	0.04	0.18
3	1055	9937	28	3.4	0.76	0.96	0.86	0.03	0.14
4	1144	9848	50	4.1	0.67	0.95	0.78	0.04	0.22
5	1055	9937	43	3.6	0.58	0.93	0.64	0.04	0.36
6	1056	9936	36	4.0	0.71	0.96	0.83	0.03	0.17
7	1142	9850	35	5.3	0.72	0.96	0.89	0.03	0.11
8	1055	9937	42	4.4	0.41	0.93	0.72	0.06	0.28
9	1055	9937	44	4.0	0.58	0.91	0.53	0.05	0.47
Average			39	4.1	0.64	0.94	0.76	0.04	0.24

Notice that time to build the model, around 4 sec for each class, does not vary much as all training data are nearly of the same size with fixed number of attributes (16) and the number of rules generated was about 40. This is explained by recalling that the algorithm's complexity is $O(RKN \log N)$, where R is the number of rules, N is the number of instances, and K is the number of attributes. Also notice that some digits, such as 0 or 7 are slightly better recognized than digits such as 1, 5, or 9. The important point about the results shown in Table 19 is that it shows that OneClass-DS can be used for solving multi-class problems.

3. CONCLUSIONS

We have introduced a rule-based OneClass-DS learning algorithm and compared it with eight algorithms on 25 data sets. It was designed for solving one-class problems where data for other classes are not available. OneClass-DS combined rule-based classification with greedy search algorithm based on density of features. When OneClass-DS was compared with the eight one-class algorithms on 25 data sets (representing numerical, nominal, and image data) it performed on par with almost all other algorithms in terms of Accuracy, IPR (Impostor Pass Rate) and FAR (False Alarm Rate) measures.

The advantages of OneClass-DS over other algorithms, however, are that it generates rules only from Target data, including on data with missing values, without the need to create an artificial data set as "the other" class, which is typical of many other one-class algorithms. It was also shown that OneClass-DS also performed well on highly unbalanced data sets when one-class model was generated for a majority class and not the much smaller minority class.

OneClass-DS can be used to analyze one-class data in a variety of application domains such as strange traffic patterns in computer networks (possibly caused by a hacked computer which sends sensitive data to an unauthorized destination), abnormal patterns in patient medical records (that could be symptoms of a new disease), outliers in credit card transactions (that could indicate credit card theft), unusual changes in satellite images of an adversary area (that could indicate troop movements), and many more.

In the future, we plan using OneClass-DS as a general framework to be combined with other rule-based algorithms, such as fuzzy classifiers, to possibly speed up learning when solving challenging real one-class problems.

ACKNOWLEDGMENTS

This work was partially supported by the NSF GANN grant (DTN) and the National Institutes of Health grant 1R01HD056235-01A1 (KJC).

References

- [1] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez, "Solving the multi-instance problem with Axis-Parallel Rectangles," *Artificial Intelligence Journal*, vol. 89, no. 1/2, pp. 31–71, 1997.
- [2] D. T. Nguyen, C. D. Nguyen, R. Hobson, L. A. Kurgan, and K. J. Cios, "mi-DS: Multiple-instance learning algorithm," *IEEE Systems, Man, And Cybernetics—Part B: Cybernetics*, Vol. 43, No. 1, pp. 143–154, 2013.
- [3] E. Lughofer, "Single-pass active learning with conflict and ignorance," *Evolving Systems*, vol. 3 (4), pp. 251–271, 2012.
- [4] L. Tarassenko, P. Hayton, and M. Brady, "Novelty detection for the identification of masses in mammograms," in *Proc. of the Fourth International IEE Conference on Artificial Neural Networks*, vol. 409, 1995, pp. 442–447.
- [5] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons, 1973.
- [6] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.
- [7] N. Ullman, *Elementary statistics, an applied approach*. Wiley and Sons, 1978.
- [8] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2007.
- [9] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, Vol. 33, pp. 1065–1076, 1962.
- [10] M. Kraaijveld and R. Duin, "A criterion for the smoothing parameter for parzen-estimators of probability density functions," Technical report. Delft University of Technology, 1991.
- [11] K. Hempstalk, E. Frank, and I. H. Witten, "One-class Classification by Combining Density and Class Probability Estimation," *Department of Computer Science, University of Waikato*, Hamilton, NZ, 2007.
- [12] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.
- [13] Ypma and R. Duin, "Support objects for domain approximation," in *Proceedings of the International Conference on Artificial Neural Networks 1998*, Skovde, Sweden, 1998.
- [14] E. Knorr, R. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *VLDB journal*, vol. 8, no. 3, pp. 237–253, 2000.

- [15] M. Breunig, H.P. Kriegel, R. Ng, and J. Sander, "LOF: indentifying density-based local outliers", in *Proceedings of the ACM SIGMOD 2000 International Conference on Management of Data*, 2000.
- [16] G. Ratsch, S. Mika, B. Schokopf, and K.-R. Muller, "Constructing boosting algorithms from svms: an application to one-class clasification," *Pattern Analysis and Machine Intelligence*, Vol. 24, pp. 1184-1199, 2002.
- [17] W. Hu, Y.Liao, and V. R. Vemuri, "Robust anomaly detection using support vector machines," in *Proceedings of the International Conference on Machine Learning*, 2002.
- [18] G. Carpenter, S. Grossberg, and D. Rosen, "ART 2-A: an adaptive resonance algorithm for rapid category learning and recognition," *Neural Networks*, vol. 4, no. 4, pp. 493-504, 1991.
- [19] T. Kohonen, *Self-organizing maps*. Heidelberg : Springer-Verlag, 1995.
- [20] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation*. Addison Wesley, 1991.
- [21] P. Baldi and K. Hornik, "Neural networks and principal component analysis: learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53-58, 1989.
- [22] N. Japkowicz, C. Myers, and M. Gluck, "A novelty detection approach to classification," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995, pp. 518-523.
- [23] D. B. Skalak, and E. L. Rissland, "Inductive learning in a mixed paradigm setting," in *Proceedings of National Conference of American Association for Artificial Intelligence*, 1990, pp. 840-847.
- [24] S. Salvador and P. Chan, "Learning states and rules for detecting anomalies in time series," *Applied Intelligence*, Vol. 23, No. 3, pp. 241-255, 2005.
- [25] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction - Second Edition*, Springer, New York Berlin Heidelberg, 2009.
- [26] W. Cohen, "Fast effective rule induction," in *Proceeding 12th International Conference Machine Learning*, Lake Tahoe, CA, 1995, pp. 115-123.
- [27] W. Fan, M. Miller, S. J. Stolfo, W. Lee, and P. K. Chan, "Using artificial anomalies to detect unknown and known network intrusions," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, 2001, pp. 123-130.
- [28] G.H. John, "Robust decision trees: Removing outliers from databases," in *Proceeding of Knowledge Discovery and Data Mining*, 1995, pp. 174-179.
- [29] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 504-509.
- [30] D. Barbará, J. Couto, S. Jajodia and N. Wu, "Adam: a testbed for exploring the use of data mining in intrusion detection," in *Proceedings of the ACM SIGMOD 2001 International Conference on Management of Data*, 2001, pp. 15-24.
- [31] D. Barbará, N. Wu, and S. Jajodia, "Detecting novel network intrusions using Bayes estimators," in *Proceeding of the First SIAM Conference on Data Mining*, 2001.
- [32] G. Tandon and P. K.Chan, "Weighting versus pruning in rule validation for detecting network and host anomalies" in *Proceedings of the 13th ACM SIGKDD international conference on Knowlegde discovery and data mining*, 2007, pp. 697-706.

- [33] M. E. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula and D. Panda, "Towards NIC-based intrusion detection," in *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 723-728.
- [34] M.V. Mahoney and P. K. Chan, "Learning non-stationary models of normal network track for detecting novel attacks," in *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 376-385.
- [35] R.W. Brause, T. Langsdorf, and M. Hepp, "Neural data mining for credit card fraud detection," in *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*, 1999, pp. 103-106.
- [36] T. Yairi, Y. Kato, and K. Hori, "Fault detection by mining association rules from house-keeping data," in *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.
- [37] H. Zengyou, X. Xiaofei, J. Z. Huang, and S. Deng, "A frequent pattern discovery method for outlier detection," Springer, pp. 726-732, 2004.
- [38] J.A. Iglesias, P. Angelov, A. Ledezma and A. Sanchis, "Evolving classification of agent's behaviors: a general approach," *Evolving Systems*, vol. 1 (3), pp. 161-172, 2010.
- [39] P. Angelov, E. Lughofer and X. Zhou, "Evolving Fuzzy Classifiers using Different Model Architectures," *Fuzzy Sets and Systems*, vol. 159 (23), pp. 3160--3182, 2008.
- [40] F. Rehm, F. Klawonn and R. Kruse, "Visualization of Fuzzy Classifiers," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 15 (5), pp. 615--624, 2007.
- [41] E. Lughofer and O. Buchtala, "Reliable All-Pairs Evolving Fuzzy Classifiers," *IEEE Transactions on Fuzzy Systems*, vol. 21 (4), pp. 625-641, 2013.
- [42] L.A.Kurgan, K.J. Cios, and S. Dick, "Highly scalable and robust rule learner: Performance evaluation and comparison," *IEEE Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 36, No. 1, pp.32-53, 2006.
- [43] L.M. Manevitz and M. Yousef, "One-Class SVMs for Document Classification," *Journal of Machine Learning Research*, vol. 2, pp. 139--154, 2001.
- [44] David M.J. Tax and Robert P.W. Duin, "Support Vector Data Description," *Machine Learning*, vol. 54 (1), pp. 45-66, 2004.
- [45] L.A. Kurgan and K.J. Cios, "CAIM discretization algorithm," *IEEE Trans. on Knowledge and Data Engineering*, Vol.16, No. 2, pp. 145-153, 2004.
- [46] F. Alimoglu and E. Alpaydin, "Methods of Combining Classifiers Based on Different Representations for Pen-based Handwritten Digit Recognition," in *Proceeding of 5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN)*, 1996.

Dat T. Nguyen received the B.S. and M.S. degrees in computer science from the School of Science, Vietnam National University - Ho Chi Minh City (HCMC), Vietnam. He received Ph.D. degree in computer science, from Virginia Commonwealth University, in 2013. His research interests include machine learning, data mining and visualization of high-dimensional data. He now works as Assistant Professor in the School of Science and Technology, Hoa-Sen University, HCMC, Vietnam and also as Visiting Professor of University of Finance-Marketing, HCMC, Vietnam.

Krzysztof J. Cios received the M.S. and Ph.D. degrees from the AGH University of Science and Technology, Krakow, Poland, D.Sc. degree (habilitation) from the Polish Academy of Sciences, Warsaw, Poland, and the M.B.A. degree from the University of Toledo, Ohio. He has worked at universities in Ohio and Colorado and currently is Professor and Chair of Computer

Science at Virginia Commonwealth University, Richmond. His research interests are in the areas of machine learning, data mining, computational neuroscience and biomedical informatics. His research has been funded by the National Institutes of Health, NASA, National Science Foundation, North Atlantic Treaty Organization, and the U.S. Air Force. He published three books and about 200 journal and conference papers. Prof. Cios was a recipient of the Norbert Wiener Outstanding Paper Award, the Neurocomputing Best Paper Award and the Fulbright Senior Scholar Award. He is a Foreign Member of the Polish Academy of Arts and Sciences.