Combining One-Class Classifiers via Meta Learning

Eitan Menahem, Lior Rokach and Yuval Elovici
Telekom Innovation Laboratories
Department of Information Systems Engineering
Ben-Gurion University of the Negev
Be'er Sheva, 84105, Israel
{eitanme,liorr,elovici}@post.bgu.ac.il

ABSTRACT

Selecting the best classifier among the available ones is a difficult task, especially when only instances of one class exist. In this work we examine the notion of combining one-class classifiers as an alternative for selecting the best classifier. In particular, we propose two one-class classification performance measures to weigh classifiers and show that a simple ensemble that implements these measures can outperform the most popular one-class ensembles. Furthermore, we propose a new one-class ensemble scheme, TUPSO, which uses meta-learning to combine one-class classifiers. Our experiments demonstrate the superiority of TUPSO over all other tested ensembles and show that the TUPSO performance is statistically indistinguishable from that of the hypothetical best classifier.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Selection process; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

Keywords

Ensemble of Classifiers, One-Class Ensemble, Meta Learning

1. INTRODUCTION AND BACKGROUND

In regular classification tasks we aim to classify an unknown instance into one class from a predefined set of classes. One-class classification aims to differentiate between instances of class of interest and all other instances. The one-class classification task is of particular importance to information retrieval tasks [12]. Consider, for example, trying to identify documents of âĂIJinterestâĂİ to a user, where the only information available is the previous documents that this user has read (i.e. positive examples), yet another example is citation recommendation, in which the system helps authors in selecting the most relevant papers to cite, from a potentially overwhelming number of references [1]. Again, one can obtain representative positive examples by simply going over the references, however it would be hard to identify typical negative examples (the fact that a certain paper is not cited by another paper does not necessarily indicate it is irrelevant). Many one-class classification algorithms have been investigated [2, 16, 9]. While there are plenty of learning algorithms to choose from, identifying the one that performs best in relation to the problem at hand is difficult. This is because evaluating a one-class classifier's performance is problematic. By definition, the data collections only contain one-class examples and thus, performance metrics, such as false-positive (FP), and true negative (TN), cannot be computed. In the absence of FP and TN, derived performance metrics, such as classification accuracy, precision, among others, cannot be computed. Moreover, prior knowledge concerning the classification performance on some previous tasks may not be very useful for a new classification task because classifiers can excel in one dataset and fail in another, i.e., there is no consistent winning algorithm.

This difficulty can be addressed in two ways. The first option is to select the classifier assumed to perform best according to some heuristic estimate based on the available positive examples (i.e., TP and FN). The second option is to train an ensemble from the available classifiers. To the best of our knowledge, no previous work on selecting the best classifier in the one-class domain has been published and the only available one-class ensemble technique for diverse learning algorithms is the fixed-rule ensemble, which in many cases, as we later show, makes more classification errors when compared to a random-selected classifier.

In this paper we search for a new method for combining one-class classifiers. We begin by presenting two heuristic methods to evaluate the classification performance of one-class classifiers. We then introduce a simple heuristic ensemble that uses these heuristic methods to select a single base-classifier. Later, we present TUPSO, a general meta-learning based ensemble, roughly based on the Stacking technique [19] and incorporates the two classification performance evaluators. We then experiment with the discussed ensemble techniques on forty different datasets. The experiments show that TUPSO is by far the best option to use when multiple one-class classifiers exist. Furthermore, we show that TUPSO's classification performance is strongly correlated with that of the actual best ensemble-member.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505619 .

1.1 One-Class Ensemble

The main motivation behind the ensemble methodology is to weigh several individual classifiers and combine them to obtain a classifier that outperforms them all. Indeed, previous work in supervised ensemble learning shows that combining classification models produce a better classifier in terms of prediction accuracy [13].

Compared to supervised ensemble learning, progress in the one-class ensemble research field is limited [7]. Specifically, the Fix-rule technique was the only method which was considered for combining one-class classifiers [18, 8, 17]. In this method, the combiner regards each participating classifier's output as a single vote upon which it applies an aggregation function (a combining rule), to produce a final classification. In the following few years, further research was carried out and presently there are several applications reaching domains, such as information security (intrusion detection), remote sensing, image retrieval, image segmentation, on-line signature verification, and fingerprint matching.

Fixed-rule ensemble techniques, however, are not optimal as they use combining rules that are assigned statically and independently of the training data. As a consequence, as we will show later, the fixed rule ensembles produce inferior classification performance in comparison to the best classifier in the ensemble.

In the following lines we use the notation $P_k(x|\omega_{Tc})$ as the estimated probability of instance x given the target class ω_{Tc}, $fr_{(T,k)}$ as the fraction of the target class, which should be accepted for classifier $k = 1 \ldots R$, N as number of features, and θ_k notates the classification threshold for classifier k. A list of fixed combining rules is presented in Table 1.

Table 1: Fix combining rules.

Combining Rule	Combination Rule Formula
Majority voting	$y(x) = I_{>k/2}\left(\sum_k I\left(P_k\left(x\middle\|\omega_{Tc}\right) \geq \theta_k\right)\right)$
Mean vote	$y(x) = \frac{1}{R}\sum_{k=1}^{R} I\left(P_k\left(x\middle\|\omega_{Tc}\right) \geq \theta_k\right)$
Weigh. mean vote	$y(x) = \frac{1}{R}\sum_{k=1}^{R}\left[f_{T,k} I\left(P_k\left(x\middle\|\omega_{Tc}\right) \geq \theta_k\right) + \left(1 - f_{T,k}\right) I\left(P_k\left(x\middle\|\omega_{Tc}\right) \geq \theta_k\right)\right]$
Avg. rule	$y(x) = \frac{1}{R}\sum_{k=1}^{R} P_k\left(x\middle\|\omega_{Tc}\right)$
Max rule	$y(x) = argmax_k[P_k\left(x\middle\|\omega_{Tc}\right)]$
Product rule	$y(x) = \prod_{k=1}^{R}[P_k\left(x\middle\|\omega_{Tc}\right)]$
Exclusive voting	$y(x) = I_1\left(\sum_k I\left(P_k\left(x\middle\|\omega_{Tc}\right) \geq \theta_k\right)\right)$
Weighted vote product	$y(x) = \frac{\prod_{k=1}^{R}[fr_{(T,k)} I(P_k(x\|\omega_{Tc}) \geq \theta_k)]}{\prod_{k=1}^{R}[f_{T,k} I(P_k(x\|\omega_{Tc}) \geq \theta_k)] + \prod_{k=1}^{R}[(1-f_{T,k}) I(P_k(x\|\omega_{Tc}) < \theta_k)]}$

Instead of using the fix-rule (e.g., weighting methods), technique to combine one-class classifiers, the meta-learning approach can be used.

1.2 Meta Learning

Meta-learning is the process of learning from basic classifiers (ensemble members); the inputs of the meta-learner are the outputs of the ensemble-member classifiers. The goal of meta-learning ensembles is to train a meta-model (meta-classifier), which will combine the ensemble members' predictions into a single prediction. To create such an ensemble, both the ensemble members and the meta-classifier need to be trained. Since the meta-classifier training requires already trained ensemble members, these must be trained first. The ensemble members are then used to produce outputs (classifications), from which the meta-level dataset (meta-dataset) is created. The basic building blocks of meta-learning are the meta-features, which are measured properties of the ensemble members output, e.g., the ensemble members' predictions. A vector of meta-features and a classification k comprise a meta-instance, i.e., meta-instance $\equiv < f_1^{meta}, \ldots, f_k^{meta}, y >$, where y is the real classification of the meta-instance that is identical to the class of the instance used to produce the ensemble members' predictions. A collection of meta-instances comprises the meta-dataset upon which the meta-classifier is trained.

2. ESTIMATING THE CLASSIFICATION QUALITY

Traditional classifier evaluation metrics, such as true negative and false positive, cannot be computed in the one-class setup, since only positive examples exist. Consequently, measures, such as a classifier's accuracy, precision, AUC, F-score, and MatthewâĂŹs correlation coefficient (MCC), cannot be computed, since $accuracy = (TP + \mathbf{TN})/(TP + \mathbf{TN} + \mathbf{FP} + FN)$, $Precision = TP/(TP + \mathbf{FP})$ and $F\text{-}score = 2 * \mathbf{P} * R/(\mathbf{P} + R)$, where P is precision and R is recall. Instead of computing the aforementioned metrics, [11] and [10], proposed heuristic methods for estimating, rather than actually measuring, the classifier's accuracy and F-score, respectively. Next, we describe the two performance estimators.

2.1 Heuristic Based Classification Performance Measures

Liu et al. [11] demonstrated that by rewriting the error probability, one can estimate the classification error-rate, in the one-class paradigm, given a prior on the target-class : $Pr[f(x) \neq y] = Pr[f(x) = 1] - Pr[Y = 1] + 2Pr[f(x) = 0|Y = 1]Pr[Y = 1]$

where $f(x)$ is the classifier's classification result for the examined example x, $Pr[f(x) = 1]$ is the probability that the examined classifier will classify *Positive*, $Pr[f(x) = 0|Y = 1]$ is the probability that the classifier will classify *Negative* when given a *Positive* example, and lastly, $Pr[Y = 1]$ is the prior on the target-class probability.

Naturally, we define the one-class accuracy (OCA), estimator as follows: $OCA = 1 - Pr[f(x) \neq y]$. Note that the probabilities $Pr[f(x) = 1]$ and $[f(x) = 0|Y = 1]$ can be estimated for any one-class problem at hand using a standard cross-validating procedure.

An additional performance criteria, $\frac{r^2}{Pr[f(x)=1]}$, denoted as One-Class F-score (OCF), is given in [10]. Using this criteria, one can estimate the classifier's F-score in the semi-supervise paradigm. However, when only positive-labeled instances exist, the recall, $r = Pr[f(x) = 1|y = 1]$, equals to $Pr[f(x) = 1]$ (because $Pr[y = 1] = 1$), which only measures the fraction of correct classifications on positive test examples, i.e., true-positive rate (TPR). Using the TPR to measure the classification performance makes sense, because the TPR is strongly correlated with the classification accuracy when negative examples are very rare, such as in the case of most one-class problems.

2.2 Best-Classifier By Estimation

Using the discussed classification performance estimators, we define a new and very simple ensemble: Estimated Best-Classifier Ensemble (ESBE). This ensemble is comprised of an arbitrary number of one-class ensemble-members (classifiers). During the prediction phase, the ensemble's output is determined by a single ensemble-member, denoted as the *dominant classifier*. The ensemble's dominant member

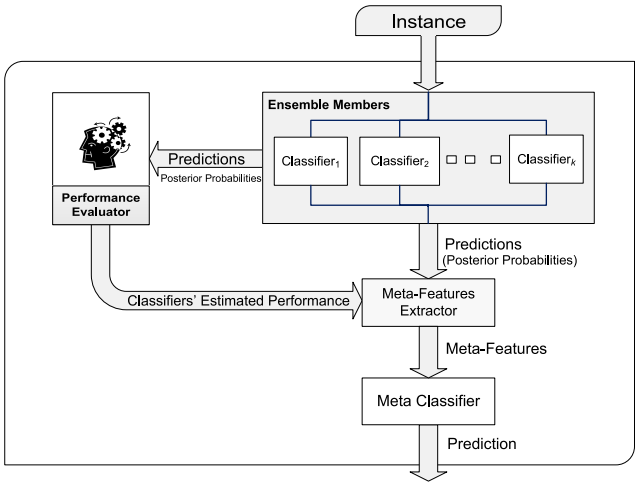

Figure 1: The TUPSO ensemble scheme.

is selected during the training phase. This is achieved by evaluating the performance of the participating ensemble-members using a 5x2 cross-validation procedure, as described in [6]. During this procedure only the training-set's instances are used, and the metric used to measure the ensemble-members' performance is either OCA or OCF.

3. TUPSO: META-LEARNING-BASED ENSEMBLE

In this section, we introduce the TUPSO ensemble scheme. The main principle of TUPSO is combining multiple and possibly diverse one-class classifiers using the meta-learning technique. TUPSO is roughly based on the Stacking technique, and as so, it uses a single meta-classifier to combine the ensembles' members. As opposed to Stacking, however, where the meta-classifier trains directly from the ensemble-members' outputs, TUPSO's meta-classifier trains on a series of aggregations from the ensemble-members' outputs. To elevate the effectiveness of some of the aggregations used by TUPSO, and with that improve the ensemble's overall performance, during the training phase, the ensemble-members are evaluated using the aforementioned one-class performance evaluators. The performance estimates are then translated into static weights, which the meta-learning algorithm uses during the training of the meta-classifier, and during the prediction phases.

The TUPSO ensemble, as shown in Figure 1, is made up of four major components: (1) Ensemble-members, (2) Performance evaluator, (3) Meta-features extractor, and (4) Meta-classifier. Next, we describe each component.

Ensemble Members

In TUPSO, the ensemble members are one-class, machine-learning-based, classifiers. TUPSO regards its ensemble members as black boxes, in order to avoid any assumption regarding their inducing algorithm, data structures or methods for handling missing values and categorical features. During the ensemble's training phase, the ensemble-members are trained several times, as part of a cross-validation process, which is required for generating the meta-classifier's dataset. This process is described later in Section 4.

Meta-Feature Name	Aggregation Function
Sum-Votes	$f_1(P_m) = \sum_{i=1}^{k} 1_{\{p_{m_i} \geq 0.5\}}(P_{m_i})$
Sum-Predictions	$f_2(P_m) = \sum_{i=1}^{k} P_{m_i}$
Sum-Weighted-Predictions	$f_3(P_m) = \sum_{i=1}^{k} \alpha_i * P_{m_i}$
Sum-Power-Weighted-Predictions	$f_4(P_m) = \sum_{i=1}^{k} \alpha_i * (P_{m_i})^2$
Sum-Log-weighted-Predictions	$f_5(P_m) = \sum_{i=1}^{k} \alpha_i * \log(P_{m_i})$
Var-Votes	$f_6(P_m) = Var(1_{\{p_{m_i} \geq 0.5\}}(P_{m_i}))$
Var-Predictions	$f_7(P_m) = Var(P_m)$
Var-Weighted-Predictions	$f_8(P_m) = Var(\alpha * P_m)$

Table 2: Aggregate functions which generate TUPSO's meta features.

Performance Evaluator

The Performance Evaluator estimates the ensemble members' classification performance during the ensemble's training phase. To fulfill its task, the Performance Evaluator uses one of the available classification performance estimators, i.e., OCA or OCF.

Meta-Features Extractor

The meta-features are measured properties of one or more ensemble-members' output. A collection of meta features for a single instance makes a meta-instance. A collection of meta-instances is called a meta-dataset. The meta-dataset is used to train the meta-classifier. The Meta Features Extractor computes the meta-features by using multiple aggregations of the ensemble-members' output. Let $P_m =< p_{(m_1)}, \ldots, p_{(m_k)} >$ be the vector containing the ensemble-members' outputs $p_{(m_1)}, \ldots, p_{(m_k)}$, where k is the number of members in the ensemble. A set of aggregate features is computed for each instance in the training set. A single set makes a single meta-instance, which will later be used either as a training instance for the meta-learner or as a test meta-instance.

Table 2 defines eight experimental aggregate meta-features. The aggregate functions $f_2 \ldots f_5$ and $f_6 \ldots f_8$ are based on the first and second moments, respectively. The first moment computes the "average" ensemble-members' prediction, whereas the second moment computes the variability among the ensemble-members' predictions. The first moment based aggregation, a subtle version of the mean voting rule, is motivated by *Condorcet's Jury Theorem*, and is used in several supervised-learning ensembles, e.g., Distribution-Summation [4]. Furthermore, the second moment based aggregation is motivated by the knowledge it elicits over the first moment, i.e., the level of consent among the ensemble-members. From this information, unique high-level patterns of ensemble members' predictions can be learned by the meta-learner, and thereafter be at the disposal of the meta-classifier. Table 3 shows the resulted structure of TUPSO's meta-dataset.

Instance	$f_1(P_m)$	$f_2(P_m)$	$f_3(P_m)$	...	$f_7(P_m)$	$f_8(P_m)$
1	$ma_{1,1}$	$ma_{1,2}$	$ma_{1,3}$	...	$ma_{1,7}$	$ma_{1,8}$
2	$ma_{2,1}$	$ma_{2,2}$	$ma_{2,3}$	...	$ma_{2,7}$	$ma_{2,8}$
...	...	...	...	...	...	...

Table 3: The training-set of the meta-classifier. Each column represents one aggregate feature over the ensemble members' predictions and $ma_{i,j}$ denotes the value of meta-feature j for meta-instance i

Meta-Classifier

The meta-classifier is the ensemble's combiner, thus, it is responsible for producing the ensemble's prediction. Similar to the ensemble-members, the meta-classifier is a one-class classifier; it learns a classification model from meta-instances, which consist of meta-features. Practically, the meta-features used in training the meta-classifier can be either aggregate features, raw ensemble-members' predictions or their combination. However, preliminary experimentsshowed that training the meta-classifier using the raw ensemble-members' predictions alone or alongside the aggregate meta-features yielded less accurate ensembles.

3.1 Training Process

The training process of TUPSO begins with training the ensemble-members followed by training the meta-classifier. The ensemble-members and the meta-classifier are trained using an inner k-fold cross-validation training process. First, the training-set is partitioned into k splits. Then, in each fold, the ensemble-members are trained on k-1 splits. Afterwards, the trained ensemble-members classify the remaining split to produce the instances for training the meta-classifier. The meta-instances in each fold are added to a meta-dataset. After k iterations, the meta-dataset will contain the same amount of instances as the original dataset. Lastly, the ensemble-members are re-trained using the entire training-set and the meta-classifier is trained using the meta-dataset.

3.2 Weighting the Ensemble Members

In order to calculate certain meta-features, e.g., f_3, the ensemble-members' predictions have to be weighed. To do so, a set of weights, one per ensemble-member, are learned as part of the ensemble training process. During the meta-classifier training, the ensemble-members predict the class of the evaluated instances. The predictions are fed to the Performance Evaluator, which calculates either OCA or OCF estimations for each of the ensemble-members, $Perf_{vect} =< Perf_1, \ldots, Perf_m >$, where $Perf_i$ is the estimated performance of ensemble-member$_i$. Finally, a set of weights, $\alpha_1, \alpha_2, \cdots, \alpha_m$, is computed as follows:

$$\alpha_i = \frac{Perf_i}{\Sigma_{j=1}^{m} Perf_j}, \forall i = 1 \ldots m$$

4. METHODS

We now specify the methods and conditions in which we investigated the presented ensemble schemes. First, we indicate the ensemble-members that participate in the ensembles. Next, we discuss the evaluated ensemble schemes.

One-Class Learning Algorithms

For evaluation purposes, we made use of four, one-class algorithms: OC-SVM [15], OC-GDE, OC-PGA, and ADIFA [14]. We selected these ensemble-members because they represent the prominent families of one-class classifiers, i.e., nearest-neighbor (OC-GDE, OC-PGA), density (ADIFA), and boundary (OC-SVM). The first two algorithms are our adaptations of two well-known supervised algorithms to one class learning.

We used a static pool of six ensemble-members for all the evaluated ensembles: (i) ADIFA$_{HM}$, (ii) ADIFA$_{GM}$, (iii) OC-GDE, (iv) OC-PGA, OC-SVM$_1$, and (vi) OC-SVM$_2$. The ensemble-members properties, illustrated in Table 4, were left unchanged during the entire evaluation.

Base Classifier	Algorithm	Parameters
ADIFA$_{HM}$	$ADIFA$	$\Psi = HarmonicMean, s = 2\%$
ADIFA$_{GM}$	$ADIFA$	$\Psi = GeometricMean, s = 1\%$
OC-GDE	$OC\text{-}GDE$	n/a
OC-PGA	$OC\text{-}PGA$	$k = 3$, $p_\alpha = 0.01$
OC-SVM$_1$	$OC\text{-}SVM$	$k = linear$, $\nu = 0.05$
OC-SVM$_2$	$OC\text{-}SVM$	$k = polynomial$, $\nu = 0.05$

Table 4: ensemble-members setup parameters. The non-default parameters are illustrated.

Ensemble Combining Methods

The following evaluation includes several ensemble combining methods from three groups of algorithms: Heuristic-Ensemble: estimated best-classifier ensemble (ESBE); Fixed-rules: *majority voting, mean-voting, max-rule* and *product-rule*; and Meta-learning-based: *TUPSO*. The learning algorithm used for inducing the meta-classifier in TUPSO was ADIFA, as it outperformed the other three mentioned learning algorithms on the evaluation set.

Datasets

During the evaluation we used a total of 40 distinct datasets from two different collections, UCI and KDD-CUP99. All datasets are fully labeled and binary-class.

We selected 34 datasets from the widely used UCI dataset repository [3]. The datasets vary across dimensions, number of target classes, instances, input features, and feature type (nominal or numeric). So as to have only two classes in the UCI datasets, a pre-process was completed where only the instances of the two most prominent classes were selected. The other instances were filtered out.

The *KDD CUP 1999* dataset contains a set of instances that represent connections to a military computer network. The dataset contains 41 attributes, 34 of which are numerical and 7 of which are categorical. The original dataset contained 4,898,431 multi-class data instances. In order to divide the dataset into multiple binary-class sets, we followed the method performed in [20]. Compared with the UCI datasets, the KDD99-CUP are much more *natural* one-class datasets, as they are highly imbalanced (instances of the network's normal state make the lion's share of the derived binary datasets).

Evaluation Methodology

During the training phase, only the examples of one-class were available to the learning algorithms and to the classification performance estimators. During the testing phase, however, both positive and negative examples were available, to evaluate the classifiers in real-life conditions. The generalized classification accuracy was measured by performing a 5x2 cross-validation procedure [6]. We used the area under the ROC curve (AUC) metric to measure the classification performance of the individual classifiers and ensemble methods.

In order to conclude which ensemble performs best over multiple datasets, we followed the procedure proposed in [5]. In the case of multiple ensembles of classifiers or features, we first used the adjusted Friedman test so as to reject the null

hypothesis, followed by the Bonferroni-Dunn test to examine whether a specific ensemble or feature produces significantly better AUC results than the reference method.

5. EXPERIMENTAL RESULTS

In this section we examine the performance of the discussed ensembles. Our goal is to learn which ensemble, if any, performs at least as good as the best ensemble member. We first investigate the performance of the discussed ensembles schemes using statistic tools. Next, we determine whether any of the ensemble of classifiers performs as good as the actual best ensemble-member classifier. Note the difference between the *actual*, best ensemble member and the *estimated*, best ensemble member. The first is determined during the evaluation phase, where both positive and negative instances exist, whereas the second is computed during the training phase, where only positive instances exist, and therefore, we should expect it to be inferior to the *actual*, best ensemble-member.

5.1 One-Class Ensembles Performance

In the following experiment we examine the classification performance of the one-class ensembles using the same dataset used in the previous experiment. Next, the ensembles' classification performances are compared with the hypothetical best-classifier, to tell which of the ensembles, if any, can match its performance.

		Random Classifier	ESBE	Majority Voting	Max	Mean Voting	Product	TUPSO
UCI Repository	Anneal	0.648 (4)	0.624 (5)	0.505 (6)	0.500 (7)	0.871 (2)	0.874 **(1)**	0.869 (3)
	Audiology	0.765 (5)	0.857 (2)	0.825 (3)	0.532 (7)	0.737 (6)	0.798 (4)	0.888 **(1)**
	Balance-scale	0.712 (5)	0.783 (2)	0.719 (3)	0.517 (7)	0.705 (6)	0.716 (4)	0.934 **(1)**
	Breast-cancer	0.495 (5)	0.506 **(1)**	0.499 (4)	0.500 (3)	0.469 (7)	0.476 (6)	0.504 (2)
	C.H. disease	0.560 (4)	0.490 (7)	0.512 (5)	0.500 (6)	0.646 (3)	0.685 **(1)**	0.654 (2)
	Credit-rating	0.638 (5)	0.691 (4)	0.598 (6)	0.502 (7)	0.721 (3)	0.770 (2)	0.788 **(1)**
	E-coli	0.917 (4)	0.934 (2)	0.932 (3)	0.884 (6)	0.819 (7)	0.913 (5)	0.963 **(1)**
	H. Statlog	0.569 (4)	0.500 (6)	0.520 (5)	0.500 (7)	0.640 (3)	0.705 **(1)**	0.692 (2)
	Hepatitis	0.615 (5)	0.669 (4)	0.579 (6)	0.500 (7)	0.684 (3)	0.777 **(1)**	0.745 (2)
	Horse-Colic	0.551 (5)	0.552 (4)	0.529 (6)	0.502 (7)	0.620 (3)	0.622 (2)	0.635 **(1)**
	H. Disease	0.619 (4)	0.543 (6)	0.573 (5)	0.505 (7)	0.749 (3)	0.754 (2)	0.791 **(1)**
	Hypothyroid	0.507 (5)	0.607 **(1)**	0.490 (7)	0.496 (6)	0.590 (3)	0.593 (2)	0.568 (4)
	Ionosphere	0.846 (4)	0.884 (3)	0.898 (2)	0.756 (6)	0.682 (7)	0.787 (5)	0.964 **(1)**
	Iris	0.948 (4)	0.904 (5)	0.965 (3)	0.975 (2)	0.850 (7)	0.895 (6)	0.995 **(1)**
	Chess	0.622 (5)	0.666 (4)	0.529 (6)	0.506 (7)	0.773 (3)	0.819 **(1)**	0.812 (2)
	Letter	0.762 (5)	0.876 (3)	0.830 (4)	0.502 (7)	0.752 (6)	0.906 (2)	0.958 **(1)**
	MFeature	0.815 (6)	0.924 (2)	0.830 (5)	0.608 (7)	0.871 (4)	0.873 (3)	0.972 **(1)**
	Mushroom	0.674 (4)	0.718 (3)	0.574 (6)	0.507 (7)	0.645 (5)	0.808 (2)	0.881 **(1)**
	Opt-Digits	0.842 (5)	0.927 (2)	0.871 (4)	0.658 (7)	0.768 (6)	0.897 (3)	0.979 **(1)**
	Page-Blocks	0.768 (5)	0.858 (3)	0.848 (4)	0.528 (7)	0.752 (6)	0.863 (2)	0.944 **(1)**
	Pen Digits	0.936 (4)	0.976 (3)	0.985 (2)	0.898 (6)	0.834 (7)	0.906 (5)	0.997 **(1)**
	Diabetes	0.527 (4)	0.493 (7)	0.507 (5)	0.505 (6)	0.544 (3)	0.552 (2)	0.555 **(1)**
	P-Tumor	0.964 (5)	0.975 (4)	0.983 (3)	1.000 **(1)**	0.876 (7)	0.905 (6)	1.000 **(1)**
	Segment	0.615 (5)	0.662 (4)	0.568 (6)	0.534 (7)	0.685 (3)	0.704 (2)	0.717 **(1)**
	Sonar	0.522 (4)	0.473 (7)	0.508 (5)	0.495 (6)	0.553 (3)	0.583 **(1)**	0.575 (2)
	Soybean	0.589 (4)	0.524 (5)	0.509 (6)	0.503 (7)	0.798 (2)	0.784 (3)	0.801 **(1)**
	SPAM-Base	0.584 (4)	0.676 **(1)**	0.598 (3)	0.500 (5)	0.500 (5)	0.500 (5)	0.630 (2)
	Splice	0.738 (5)	0.976 (2)	0.624 (6)	0.500 (7)	0.769 (4)	0.862 (3)	0.989 **(1)**
	Vehicle	0.608 (4)	0.595 (5)	0.525 (6)	0.497 (7)	0.731 (3)	0.793 **(1)**	0.787 (2)
	Vote	0.746 (4)	0.923 (2)	0.744 (5)	0.504 (7)	0.690 (6)	0.830 (3)	0.937 **(1)**
	Vowel	0.671 (4)	0.651 (5)	0.494 (7)	0.500 (6)	0.797 (3)	0.813 (2)	0.849 **(1)**
	Waveform	0.769 (4)	0.752 (6)	0.758 (5)	0.661 (7)	0.812 (3)	0.814 (2)	0.865 **(1)**
	WB-Cancer	0.768 (5)	0.967 (2)	0.953 (3)	0.500 (7)	0.725 (6)	0.826 (4)	0.979 **(1)**
	Zoo	0.732 (5)	0.879 (2)	0.656 (6)	0.5 (7)	0.756 (4)	0.828 (3)	0.943 **(1)**
KDD CAP 99	AUTH	0.913 (3)	0.784 (7)	0.978 (2)	0.869 (5)	0.861 (6)	0.907 (4)	0.989 **(1)**
	FTP	0.684 (5)	0.730 (4)	0.549 (6)	0.509 (7)	0.801 (3)	0.883 (2)	0.903 **(1)**
	FTP-DATA	0.655 (5)	0.781 (2)	0.602 (6)	0.506 (7)	0.713 (4)	0.794 **(1)**	0.775 (3)
	OTHER	0.825 (5)	0.967 (2)	0.748 (6)	0.600 (7)	0.895 (4)	0.897 (3)	0.995 **(1)**
	POP3	0.874 (5)	0.964 (3)	0.983 (2)	0.750 (7)	0.788 (6)	0.909 (4)	0.983 **(1)**
	SMTP	0.813 (4)	0.970 **(1)**	0.737 (7)	0.749 (5)	0.748 (6)	0.869 (3)	0.931 (2)
	Average Rank	4.5 (5)	3.6 (3)	4.8 (6)	6.3 (7)	4.5 (5)	2.9 (2)	**1.4 (1)**

Table 5: Ensembles classification AUC results.

The results in Table 5, show that the ensemble with the highest average rank is the meta-learning based, TUPSO. The next best ensemble, by a large margin, is the product-rule, which was the only fixed-rule that was ranked higher than the random-classifier scheme. This is an indication for the high independence among the ensemble participants, induced by the heterogeneous learning algorithms. Indeed,

	Majority Voting	Max Rule	Mean Voting	Product Rule	ESBE	Random Classifier	Best Classifier
TUPSO	+ (≈ 0)	+ (≈ 0)	+ (≈ 0)	+ (< 0.01)	+ (< 0.01)	+ (≈ 0)	**(0.243)**
Best Clas.	+ (≈ 0)	+ (≈ 0)	+ (≈ 0)	+ (< 0.01)	+ (≈ 0)	+ (≈ 0)	
Rnd Clas.	(0.786)	+ (< 0.01)	(0.964)	− (< 0.01)	(0.099)		
ESBE	+ (0.057)	+ (≈ 0)	(0.108)	(0.152)			

Table 6: The statistical significance of the difference in the AUC measure of the examined ensembles. The *p-value* of the test statistic is inside the brackets. The '+' ('-') symbol indicates that the average AUC value of the ensemble, indicated at the row's beginning, is significantly higher (lower), compared to the ensemble indicated in the tableâĂŹs header with a confidence level of 95%.

[18] showed that the product-rule is motivated by independence of the combined models. In that work, however, the authors applied feature-set partitioning to decrease the dependency among the combined models.

The heuristic ensemble, ESBE, was ranked, on average, higher than the random classifier, showing, along with TUPSO, the benefits of using classification performance estimation. This simple ensemble was also ranked higher than the majority-voting, max-rule, and the mean-voting.

The statistical significance of the ranking difference between the examined ensembles is presented in Table 6

The statistical tests reveal four clusters of classifiers, each comprised of statistically comparable ensembles or classifiers. TUPSO and the best-base classifier populate the cluster that represents the top-tier classification performance. ESBE and product-rule ensembles comprise the cluster that represents the above-average classification performance. The majority-voting, mean voting and randomly selected classifier make the below-average performance cluster, and finally, the cluster that represents the lowest classification performance is comprised of the max-rule ensemble.

5.2 Classifying *Like* the Best-Classifier?

We continue further with our experiment to find out which of the ensembles can be used as an alternative for the best base-classifier. Assuming that the users require a one-class classifier for a single classification task (i.e., a single dataset), they will be more concerned by how well their classifier will perform on their classification task, rather than how it will perform on average (i.e., on many different datasets). In our case, the best performer, .i.e, TUPSO, is on average as good as the best base-classifier and therefore is the natural choice of ensemble. However, it might be the case that on several datasets TUPSO significantly outperforms the best base-classifier, while on other datasets, it significantly falls behind the best base-classifier. If this is the case, one might want to use another ensemble that has a greater chance of classifying like the best base-classifier.

Figure 2 graphically shows the link between the AUC performance of the best base-classifier and that of the ensembles. Each dataset is represented by a single data point. To find out which of the ensembles' AUC performance is most

	Random Classifier	Majority Voting	Max	Mean Voting	Product	ESBE	TUPSO
Pearson Correlation	0.815	0.661	0.439	0.788	0.883	0.812	0.962

Table 7: Pearson Correlation.

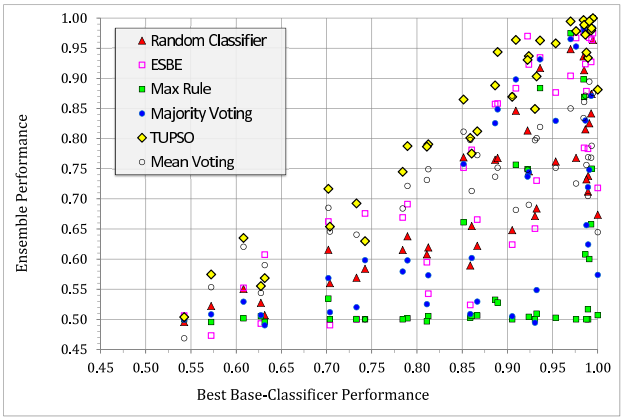

Figure 2: Classification performance: ensembles vs. actual best classifier.

tightly linked with the best base-classifier, we computed a correlation matrix using the Pearson Correlation routine. The results in Table 7 show that the TUPSO ensemble has the most correlated AUC performance with the best base-classifier.

6. CONCLUSIONS

Thus far, the only combining scheme used for diverse learning algorithms, in the context of one-class learning, is the Fix-rule. Judging by publication quantity, schemes, such as majority voting, mean-voting, and max-rule are the most popular. However, in this study we hypothesized the existence of an innate limitation with such combining methods, as they do not take into account the properties of the ensemble-members they combine. Further along, we empirically demonstrated this limitation. The fixed-rule schemes indeed produced a lower classification accuracy when compared to the best base classifier.

In this paper we searched for an improved method for combining one-class classifiers. We implemented two one-class classification performance evaluators, OCA and OCF, which evaluated ensemble-members using only the available positive labeled instances. We then introduced ESBE, a simple ensemble that uses OCA or OCF to select a single classifier from the classifiers pool. Our experiment showed that while this method is inferior to the best ensemble-member, it still outperforms the Majority voting.

Lastly, we introduced a meta-learning based ensemble, TUPSO, which learns a combining function upon aggregates of the ensemble-members' predictions. Thus, in contrast to the fix-rule scheme, TUPSO depends on the classification properties of the ensemble-members. Our experiments demonstrated the superiority of TUPSO over all other tested ensembles, both in terms of classification performance and in correlation to the best ensemble-member. Furthermore, TUPSO was found to be statistically indistinguishable from the best ensemble-member, thus, it completely removes the necessity of choosing the best ensemble-member.

7. REFERENCES

[1] S. Bethard and D. Jurafsky. Who should i cite: learning literature search models from citation behavior. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 609–618. ACM, 2010.

[2] C. M. Bishop. Novelty detection and neural network validation, 1994.

[3] C. Blake and C. Merz. UCI repository of machine learning databases. 1998.

[4] P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. In *Machine learningâĂŤEWSL-91*, pages 151–163. Springer, 1991.

[5] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[6] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[7] G. Giacinto, R. Perdisci, M. D. Rio, and F. Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008.

[8] P. Juszczak and R. P. W. Duin. Combining one-class classifiers to classify missing data. In *Multiple Classifier Systems*, pages 92–101, 2004.

[9] A. Kontorovich, D. Hendler, and E. Menahem. Metric anomaly detection via asymmetric risk minimization. In *SIMBAD*, pages 17–30, 2011.

[10] W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, pages 448–455, 2003.

[11] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML*, pages 387–394, 2002.

[12] L. M. Manevitz and M. Yousef. One-class svms for document classification. *The Journal of Machine Learning Research*, 2:139–154, 2002.

[13] E. Menahem, L. Rokach, and Y. Elovici. Troika - an improved stacking schema for classification tasks. *Inf. Sci.*, 179(24):4097–4122, 2009.

[14] E. Menahem, A. Schclar, L. Rokach, and Y. Elovici. Securing your transactions: Detecting anomalous patterns in xml documents. *CoRR*, abs/1209.1797, 2012.

[15] B. Schlkopf, J. C. Platt, J. Shawe-taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution, 1999.

[16] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[17] S. Seguí, L. Igual, and J. Vitrià. Weighted bagging for graph based one-class classifiers. In *MCS*, pages 1–10, 2010.

[18] D. M. Tax and R. P. Duin. Combining one-class classifiers. In *Proc. Multiple Classifier Systems, 2001*, pages 299–308. Springer Verlag, 2001.

[19] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[20] K. Yamanishi, J. ichi Takeuchi, G. J. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *KDD*, pages 320–324, 2000.