One-class Classification by Combining Density and Class Probability Estimation

Kathryn Hempstalk, Eibe Frank, and Ian H. Witten

Department of Computer Science, University of Waikato, Hamilton, NZ
{kah18,eibe,ihw}@cs.waikato.ac.nz

Abstract. One-class classification has important applications such as outlier and novelty detection. It is commonly tackled using either density estimation techniques or by adapting a standard classification algorithm to the problem of carving out a decision boundary that describes the location of the target data. In this paper we present a simple method for one-class classification that combines the application of a density estimator, used to form a reference distribution, with the induction of a standard model for class probability estimation. In our method, the reference distribution is used to generate artificial data that is employed to form a second, artificial class. In conjunction with the target class, this artificial class is the basis for a standard two-class learning problem. We explain how the density function of the reference distribution can be combined with the class probability estimates obtained in this way to form an adjusted estimate of the density function of the target class. Using UCI datasets, and data from a typist recognition problem, we show that the combined model, consisting of both a density estimator and a class probability estimator, can improve on using either component technique alone when used for one-class classification. We also compare the method to one-class classification using support vector machines.

1 Introduction

In most classification problems, training data is available for all classes of instances that can occur at prediction time. In this case, the learning algorithm can use the training data for the different classes to determine decision boundaries that discriminate between these classes. However, there are some problems that are suited to machine learning, but exhibit only a single class of instances at training time. At prediction time, new instances with unknown class labels can either belong to this target class or a new class that was not available during training. In this scenario, two different predictions are possible: *target*, meaning an instance belongs to the class learned during training, and *unknown*, where the instance does not appear to belong to the previously learned class. This type of learning problem is known as one-class classification.

In many cases, it may seem sensible to suggest that one-class problems should be reformulated into two-class ones because there is actually data from other classes that can be used for training. However, there are genuine one-class applications where it is inappropriate to make use of negative data during training.

For example, consider password hardening, which is a biometric system that strengthens the login process on a computer by not only requiring the correct password to be typed, but also requiring it to be typed with the correct rhythm. Password hardening is clearly a one-class problem; a single user must be verified and during training time only data from that user is available—we cannot ask anyone else to provide data without supplying them with the password.

Even in applications where instances from multiple classes are available at training time, it may be opportune to focus solely on the target class under consideration and contemplate a one-class set-up. In these applications, new classes may occur at prediction time that are different from *all* classes available during the training process. This is the case in the continuous typist recognition problem that motivated the work presented in this paper. Continuous typist recognition is similar to password hardening, only the text underlying the patterns is not fixed: the current typist is verified on a block of free text. In this situation we rely on one-class classification because we need to be able to refuse an attacker the system has never seen before.

One-class classification is often called outlier (or novelty) detection because the learning algorithm is being used to differentiate between data that appears normal and abnormal with respect to the distribution of the training data. A common statistical approach to this view on one-class classification is to identify outliers as instances that are greater than a distance, d, to a percentage, p, of the training data [2, 8]. Machine learning techniques that have been employed include clustering the data and determining a suitable boundary that encloses all the clusters [15], adapting kernel-based support vector machine classifiers [11], and utilising densities to estimate target class membership [12].

In this paper we present a principled approach for applying two-class classification algorithms to one-class classification. The only requirement is that these algorithms can produce class probability estimates at prediction time. This is not an impediment in general because most algorithms either provide these estimates directly or can be modified to do so.

Our approach is based on the generation of artificial data that comes from a known reference distribution such as a multi-variate normal distribution, which can be estimated from the training data for the target class. In our method, this artificial data takes the role of a second class in the construction of a two-class classification model. Using Bayes' rule, we show how the density function of the reference distribution can be combined with the class probability estimates of this classification model to yield a description of the target class. We show that the combined model, which employs both the density function and the classification model, can yield improved performance when compared to using the density function alone for one-class classification; the latter being a standard technique that has been used for one-class classification in the past [12, 13]. It also improves on using the classification model alone. The resulting method yields numeric membership scores that can be used to rank test instances according to their predicted likelihood of belonging to the target class. This property can be an advantage when compared to techniques that provide a single decision

boundary, because it makes it possible to adjust the trade-off between false positives and false negatives at prediction time.

The next section discusses previous approaches to one-class classification. Following this, we introduce our new method for one-class classification. In Sections 4 and 5 we present the evaluation of this new technique, on both standard UCI datasets and our continuous typist recognition data, and compare it to existing approaches. The final section concludes the paper and proposes some future work.

2 Related Work

Existing models for one-class classification either extend current methods for multi-class classification or are based on density estimation. In the latter approach, density estimation is performed by fitting a statistical distribution, such as a Gaussian, to the target data; any instances with a low probability of appearing (more precisely, low density value) can be marked as outliers [8]. This is a sensible approach in cases where the target data follows the selected distribution very closely. The challenge is to identify an appropriate distribution for the data at hand. Alternatively one can use a non-parametric approach, such as kernel density estimation, but this can be problematic because of the curse-of-dimensionality and the resulting computational complexity.

Extensions of current multi-class classifiers to one-class classification involve fitting a boundary around the target data, such that instances that fall outside the boundary are considered outliers. The boundary can be generated by adapting the inner workings of an existing multi-class classifier [11], or by using artificial data as a second class, in conjunction with a standard multi-class learning technique [1]. Methods in the former category generally rely heavily on a parameter that defines how much of the target data is likely to be classified as outlier [13]. This parameter defines how conservative the boundary around the target class will be. If it is chosen too liberally, then the model will overfit and we risk identifying too much legitimate target data as outliers. A drawback of these techniques is that an appropriate parameter value needs to be manually chosen at training time.

In contrast, when density estimation is used for one-class classification, a threshold on the density can be adjusted at prediction time to obtain a suitable rate of outliers. In some situations, where parametric density estimation fails, using classification-based methods may be favourable; these techniques are generally able to define boundaries on data that cannot be tightly modelled by a standard statistical distribution. It is possible to use density estimation and classification-based methods sequentially [10], first using the density estimate to identify possible outliers in the training data, then relabelling outliers and using a classification method such as a kernel-based method to learn the target versus the outlier class. The issue with this approach is that it relies on two important properties for accurate classification: that real outliers are present in the training data and that the density estimate is able to define them.

The method presented in this paper is based on the generation of artificial data to form a two-class classification problem. However, unlike earlier work [1], our method is based on using a density estimate adapted to the target class to generate the artificial data for a two-class probability estimator, and the estimated reference density function is combined with the resulting class probability estimator to form an overall prediction.

Our method is generic in the sense that it is applicable in conjunction with an arbitrary density estimator and an arbitrary class probability estimation technique. An alternative generic approach is ensemble learning, where the predictions of different one-class classifiers are combined in an ensemble [14]. It is also possible to use predictions sequentially [10], as described earlier. In contrast, we combine a density estimator and a class probability estimator to form a single one-class classifier.

3 Combining Density Functions and Class Probability Estimators

Given the large number of classification algorithms that have been developed, it would be useful to be able to utilize them for one-class problems. A possible approach for doing this is to generate artificial data to take the role of the second class. The most straightforward method for implementing this is to generate uniformly distributed data and learn a classifier that can discriminate this data from the target. A problem with this method is that different decision boundaries are obtained for different amounts of artificial data: if too much artificial data is generated, then the target class will be overwhelmed by it, and, assuming the learning algorithm aims to minimize classification error, it will learn to always predict the artificial class. However, this problem can be avoided when the objective of learning is viewed as accurate class probability estimation rather than minimization of classification error, and a suitable configuration of the learning algorithm is chosen. An example of a suitable inductive approach is bagging of unpruned decision trees, which has been shown to yield good class probability estimators [9].

Once a class probability estimation model has been obtained in this fashion, different thresholds on the resulting class probability estimates for the target class correspond to different decision boundaries surrounding the instances of the target class. This means that, as in the density estimation approach to one-class classification, the rate of obtaining "outliers" can be adjusted at prediction time to yield an outcome appropriate for the application at hand.

There is one significant problem with the approach that we have just described: as the number of attributes in the learning problem increases (i.e. as the dimensionality of the instance space grows), it quickly becomes infeasible to generate enough artificial data to obtain sufficient coverage of the instance space, and the probability that a particular artificial instance occurs inside or close to the target class diminishes to a point which makes any kind of discrimination impossible.

The solution to this problem is to generate artificial data that is as close as possible to the target class. In this case, because the data is no longer uniformly distributed, it becomes necessary to take the distribution of this artificial data into account when computing the membership scores for the resulting one-class model. We call the distribution that is used to generate the artificial data the "reference" distribution. In the following we show how the class probability estimates of the two-class classifier are combined with the values of the density function of this reference distribution to obtain membership scores for the target class.

Let T denote the target class for which we want to build a one-class model. We have training data for this class. Let A be the artificial class, for which we generate artificial data using a known reference distribution. Let X denote an instance and let $P(X|A)$ denote the density function of the reference distribution.

What we would like to obtain is $P(X|T)$, the density function for the target class. If we had this density function, we could use it for one-class classification by imposing a threshold on its values. Let us assume for the moment that we know the true class probability function $P(T|X)$. In practice, we need to estimate this function using a class probability estimator learned from the training data.

The following shows how we can compute the density function for T, namely $P(X|T)$, given the class probability function $P(T|X)$, the reference density $P(X|A)$, and $P(T)$, which is the prior probability of observing an instance of the target class. We start with Bayes' theorem:

$$P(T|X) = \frac{P(X|T)P(T)}{P(X)}$$

For a two-class situation, the probability of X is the probability of seeing an instance of X with either class label, so the equation becomes:

$$P(T|X) = \frac{P(X|T)P(T)}{P(X|T)P(T) + P(X|A)P(A)}$$

Now we solve for $P(X|T)$, the density function for the target class, which we want to use for one-class classification. We first bring the denominator on the right to the left:

$$(P(X|T)P(T) + P(X|A)P(A))P(T|X) = P(X|T)P(T)$$

Now we expand the product on the left, and bring the term involving $P(X|T)$ to the right:

$$P(X|T)P(T)P(T|X) + P(X|A)P(A)P(T|X) = P(X|T)P(T)$$

$$P(X|A)P(A)P(T|X) = P(X|T)P(T) - P(X|T)P(T)P(T|X)$$

Then we extract out $P(X|T)$ and bring the remainder to the left:

$$P(X|A)P(A)P(T|X) = P(X|T)(P(T) - P(T)P(T|X))$$

$$\frac{P(X|A)P(A)P(T|X)}{P(T) - P(T)P(T|X)} = P(X|T)$$

We swap the two sides and extract $P(T)$ in the denominator:

$$P(X|T) = \frac{P(X|A)P(A)P(T|X)}{P(T)(1 - P(T|X)}$$

Now we make use of the fact that $P(A) = 1 - P(T)$, because there are only two classes, and rearrange:

$$P(X|T) = \frac{(1 - P(T))P(T|X)}{P(T)(1 - P(T|X))}P(X|A) \tag{1}$$

This equation relates the density of the artificial class $P(X|A)$ to the density of the target class $P(X|T)$ via the class probability function $P(T|X)$ and the prior probability of the target class $P(T)$.

To use this equation in practice, we choose $P(X|A)$ and generate a user-specified amount of artificial data from it. Each instance in this data receives the class label A. Each instance in the training set for the target class receives class label T. Those two sets of labeled instances are then combined. The proportion of instances belonging to T in this combined dataset is an estimate of $P(T)$, and we can apply a learning algorithm to this two-class dataset to obtain a class probability estimator that takes the role of $P(T|X)$. Assuming we know how to compute the value for $P(X|A)$ given any particular instance X—and we can make sure that this is the case by choosing an appropriate function—we then have all the components to compute an estimate of the target density function $\hat{P}(X|T)$ for any instance X.

Note that, because we are using estimates for $P(T|X)$ and $P(T)$, the function $\hat{P}(X|T)$ will not normally integrate to one and is thus not a proper density. However, this is not a problem for one-class classification because we can empirically choose an appropriate threshold on $\hat{P}(X|T)$ to perform classification, and we can adjust this threshold to tune the probability of an instance being identified as an outlier. Obviously, we can also use this function to rank instances. Note that this also means the prior odds ratio $\frac{(1-P(T))}{P(T)}$ is irrelevant in practice, because it corresponds to a constant factor, and it is sufficient to use an estimate of the posterior odds $\frac{P(T|X)}{1-P(T|X)}$ in conjunction with $P(X|A)$.

It is instructive to relate this method to the simple approach discussed at the beginning of this section, which was based on generating uniformly distributed data for the artificial class. This corresponds to using the uniform distribution as the reference distribution, i.e. using the uniform density for $P(X|A)$. Based on Equation 1, this means that $P(X|T)$ becomes proportional to the posterior odds ratio $P(T|X)/(1-P(T|X))$—which is monotonic in $P(T|X)$—meaning that test cases can simply be ranked according to the class probability function $P(T|X)$, and classification can be performed based on an appropriately chosen threshold for this function. This means the two-class classification model that is learned from the target data and the uniformly distributed data can directly be used

for one-class classification; for the reason given above, this is only a worthwhile option to consider for very low-dimensional learning problems.

One question remains, namely how to choose the reference density $P(X|A)$. Essential requirements for this function are that (a) we need to be able to generate artificial data from it and (b) we need to be able to compute its value for any instance X. Another requirement, as indicated at the beginning of this section, is that the data generated based on this distribution is close to the target class. In fact, based on Equation 1 we now know that, ideally, the reference density is identical to the target density, in which case $P(T|X)$ becomes a constant function that any well-behaved learning algorithm should be able to induce (i.e. the resulting two-class learning problem would become trivial). This is obviously not realistic because it would essentially require us to know (or somehow obtain) the density of the target class. However, this observation gives us a clue as to how we can go about getting a useful density function $P(X|A)$ for the problem at hand: we can apply *any* density estimation technique to the target data for class T and use the resulting density function to model the artificial class A. The more accurately this initial density estimate models $P(X|T)$, i.e. the better the match between $P(X|A)$ and $P(X|T)$, the easier the resulting two-class class probability estimation task should become. This discussion implies that Equation 1 can also be viewed as a mechanism for improving an initial density estimate for a dataset using a class probability estimation technique.

In practice, given the availability of powerful methods for class probability estimation, and the relative lack of such techniques for density estimation, it makes sense to apply a simple density estimation technique to the target data first, to obtain $P(X|A)$, and then employ a state-of-the-art class probability estimation method to the two-class problem that is obtained by joining the artificial data generated using $P(X|A)$ and the data from the target class. This is the strategy that we evaluate in this paper.

Because Equation 1 is used to estimate a density function for the target class, our approach is most closely related to the standard density-estimation-based approach to one-class classification. However, given that a class probability estimation model is used as part of the estimation process, which is obtained from a standard technique for classification learning, the method is also related to approaches that adapt standard two-class learning techniques to the one-class learning task: in a sense, it straddles the boundary between these two groups of approaches to one-class classification.

4 Evaluation Method

The development of the one-class learning technique presented in this paper was sparked by our interest in the domain of continuous typist recognition. In the next section we present empirical results for our method on datasets from this domain. However, we also present results for standard multi-class UCI datasets.

Evaluating one-class classifiers on datasets with multiple classes is straightforward. Each class in turn is treated as the target class and the other classes are

joined into an "outlier" class. In our experiments, we ran a standard stratified 10-fold cross-validation, repeated 10 times, to estimate the AUC for a particular target class. The one-class learning methods simply ignore the data for the outlier class that occurs in the training sets. The instances in the test sets are ranked according to their predicted density values so that AUC can be computed. In this fashion we obtain one AUC value for each class in a dataset. To summarize performance for a particular UCI dataset, we report a weighted average of these AUC values. For the weighted AUC, each two-class AUC is calculated, then weighted by the prevalence of the target class and summed. The formula for the weighted AUC is:

$$AUC_{weighted} = \sum_{\forall c_i \in C} AUC(c_i) \times p(c_i) \tag{2}$$

where C is the full set of classes in the dataset, $p(c_i)$ is the prevalence of the target class c_i in the full dataset, and $AUC(c_i)$ is the AUC value for target class c_i. Using a weighted average rather than an unweighted one prevents target classes with smaller instance counts from adversely affecting the results.

Our method combines the output of a density estimator with that of a class probability estimator. In our evaluation we used bagged unpruned C4.5 decision trees with Laplace smoothing as the probability estimator $P(T|X)$. Ten bagging iterations were used throughout. We evaluated two different simple density estimation models: a Gaussian density with a diagonal co-variance matrix containing the observed variance of each attribute in the target class, and a product of mixture of Gaussian distributions with one mixture per attribute. Each mixture is fitted to the target data for its corresponding attribute using the EM algorithm. The amount of artificial data generated using the reference distribution, which determines the estimate of $P(T)$ in Equation 1, was set to the size of the target class. Hence the data used to build the bagged unpruned decision trees was exactly balanced.

One of the main objectives of our empirical study was to ascertain that the combined model can indeed improve on its components in terms of predictive performance. Hence we also evaluated one-class classification using the Gaussian density and the EM-based density directly, with the same cross-validation-based experimental set-up described above. This means we use the reference density $P(X|A)$ from Equation 1 to rank test instances, rather than $\hat{P}(X|T)$. Additionally, we also measured the performance obtained when using only the class probability estimator from Equation 1, i.e. the bagged unpruned decision trees in our experiments. This means we only use the estimate for $P(T|X)$ from Equation 1 for ranking. Note that in this case the reference density is still used to generate the data for the artificial second class.[1]

As an optimistic baseline we also show the performance obtained on the UCI datasets when treating the learning problem as a standard classification problem,

[1] We also tried using uniformly distributed data as the artificial class, but the results were poor, so they are not shown in what follows.

using the data for all classes at training time, by building a one-vs-rest model using bagged decision trees.

For completeness, we also compare our method to the one-class support vector machine described by [11], as it is implemented in libSVM [3]. We used RBF kernels and set the value of ν to 0.1. The parameter ν determines how much of the target data is likely to be classified as outlier. We adjusted the γ value for the RBF kernel for each dataset to obtain a false alarm rate (FAR) as close as possible to 0.1. The false alarm rate is the number of legitimate target instances incorrectly identified as outliers (also known as the false negative rate).

When comparing libSVM to our combined one-class classifier we cannot use AUC because the one-class implementation in libSVM does not return membership scores, just a yes/no decision. Hence our comparison is based on attempting to achieve a fixed FAR, namely 0.1, for both techniques, by choosing an appropriate threshold for our model, and evaluating the corresponding impostor pass rate (IPR). The impostor pass rate is the number of outlier instances that are wrongly classified as belonging to the target class (also known as the false positive rate). FAR and IPR are often used in domains such as biometrics. A higher FAR results in a lower IPR and vice versa. Note that, to calculate FAR and IPR in our experiments, false negatives and false positives were simply accumulated across all one-class learning problems that resulted from processing a multi-class dataset. As in the case of AUC, 10-fold cross-validation, repeated 10 times, was used for a single one-class learning problem.

5 Results

In the following we present the experimental results obtained using the methodology described above. We first discuss the performance of the combined classifier, its components, and the baseline multi-class classifier on standard multi-class UCI datasets. In the second subsection we introduce the typist dataset, which motivated this work, and use it to show the performance of the combined classifier on individual classes. Finally, we present a comparison between the combined one-class classifier and the one-class support vector machine [3, 11].

5.1 UCI Datasets

Table 1 contains the results we obtained for the UCI datasets, comparing weighted AUC values for the baseline multi-class classifier, in the left-most column, and the variants of one-class classifiers discussed above, excluding the SVM. More specifically, there are two groups of three columns for the one-class classifiers. The first group is based on using the Gaussian density as the reference density and the second one based on using the EM method. Each group of three columns has results for (a) our combined classifier, consisting of both the reference density and the class probability estimation model (i.e. bagged trees), (b) the reference density $P(X|A)$ used directly, and (c) the class probability estimator $P(T|X)$ used directly.

From the results presented in Table 1 it is clear that the combined classifier performs much better than its component probability estimator—i.e. the estimate of $P(T|X)$—for all of the UCI datasets. Hence it is essential to take the reference density into account when making a prediction. It is not sufficient to simply generate artificial data based on the reference density, build a class probability estimator, and then use that for prediction.

The picture is not so clear when comparing the combined classifier to the reference density. Considering the Gaussian case, there are four datasets where the latter produces better results: diabetes, ecoli, iris, and waveform-5000. This indicates that the combined model is too complex in those cases; the simple reference distribution is sufficient to model the distribution of the target class for those datasets, and adding bagged trees into the model is detrimental. On the other hand, there are six datasets where the combined model performs better: heart-statlog, letter, mfeat-karhunen, pendigits, sonar, and vehicle. In the case of the vehicle and letter datasets the difference is substantial—and is so even for the AUC values of every individual class label occurring in these datasets (for brevity these results are not shown here).

Considering the case of using the EM-based density estimate as the reference density, the overall picture is similar, but there is only one tie in this case (sonar). There are eight wins for the combined classifier (heart-statlog, letter, the three mfeat datasets, pendigits, vehicle, and waveform-5000) and six losses (diabetes, ecoli, glass, ionosphere, iris, and pendigits). The biggest wins in absolute terms for the combined method occur again on the vehicle and letter datasets.

It is instructive to compare the performance of the two different combined models. The EM-based model wins on nine datasets and losses on only five. Hence the combined model often receives a boost when the more powerful EM-based density estimator is used to obtain the reference density. Note also that the EM-based density often has an edge compared to using the Gaussian density when both are used in stand-alone mode: the former wins nine times and losses six times.

Not surprisingly, standard bagged decision trees outperform all one-class classifier variants on all datasets, often by a significant margin. The reason for this is that this learner actually gets to see data for the outlier class (i.e. the union of the non-target classes) at training time. It can thus focus on discriminating the target class against all those specific non-target classes, and its performance can be considered an optimistic target for that of corresponding one-class classifiers. We would like to emphasize here that in practical applications of one-class classifiers this kind of data is not available. Even if there is some negative data available at training time, the expectation is that completely new classes of data will be encountered at prediction time.

5.2 Typist Dataset

The work in this paper was motivated by the need to find an appropriate classification method for a continuous typist recognition problem. As mentioned previously, continuous typist recognition is akin to password hardening, only the

	Bagged	One-class Classifier Gaussian			One-class Classifier EM		
Dataset	Trees	Combined	$P(X\|A)$	$P(T\|X)$	Combined	$P(X\|A)$	$P(T\|X)$
diabetes	0.818	0.626	0.653	0.511	0.639	0.669	0.510
ecoli	0.953	0.928	0.930	0.516	0.928	0.931	0.542
glass	0.878	0.698	0.698	0.624	0.731	0.735	0.593
heart-statlog	0.880	0.796	0.790	0.671	0.768	0.751	0.629
ionosphere	0.965	0.697	0.697	0.587	0.726	0.727	0.580
iris	0.985	0.974	0.977	0.628	0.972	0.976	0.662
letter	0.996	0.904	0.887	0.701	0.931	0.921	0.783
mfeat-karhunen	0.984	0.957	0.955	0.524	0.960	0.959	0.577
mfeat-morphological	0.959	0.941	0.941	0.804	0.940	0.939	0.836
mfeat-zernike	0.959	0.898	0.898	0.418	0.904	0.902	0.622
optdigits	0.995	0.959	0.959	0.562	0.954	0.955	0.645
pendigits	0.998	0.958	0.953	0.845	0.938	0.933	0.821
sonar	0.867	0.588	0.587	0.484	0.612	0.612	0.501
vehicle	0.919	0.705	0.657	0.656	0.781	0.765	0.700
waveform-5000	0.951	0.863	0.864	0.415	0.864	0.863	0.466

Table 1. Weighted AUC results on UCI data, for standard bagged decision trees, the combined one-class classifier, and its components

patterns presented to the system may contain any number of characters and sample lengths may vary. During our search for a method for improving the state-of-the-art in this research area, we investigated several different one-class classifiers—all of which were customised to the task of typist recognition [4–7]. We felt that this problem would benefit from the extensive research performed on multi-class classifiers and directed our efforts towards creating a dataset that could be used by standard machine learning algorithms.

Unfortunately, with the ethical issues surrounding key logging data, and the omission of key release events from one of the datasets we had access to, we were unable to transform an existing dataset for use in our experiments. Instead, we recorded 3000 emails from 19 people in the Computer Science Department at the University of Waikato over a period of 3 months. After technical and ethical issues were addressed, a dataset of 15 emails for each of 10 participants was created. This dataset only contained the raw sequences of input from the recordings. Each sample was broken down further into blocks of 400 events—a size that roughly equates to a small paragraph of text and is similar in size to other typist datasets [5]—resulting in between 24 and 75 samples per user. Every 400-event sample had a number of attributes calculated; these attributes formed the dataset for use with a standard machine learning algorithm.

In total, there are 8 attributes in our typist dataset.[2] Most of the attributes are based around the typist speed (average words-per-minute (WPM) rate, peak WPM, trough WPM) or error rate (backspaces, paired backspaces, average backspace block length). There are also two attributes that relate to the slurring of key press and release events (press/release ordering, press/release rate).

[2] This number is likely to change in future for the purposes of typist recognition.

Participant	Bagged Trees	One-class Classifier Gaussian			One-class Classifier EM		
		Combined	$P(X\|A)$	$P(T\|X)$	Combined	$P(X\|A)$	$P(T\|X)$
A	0.938	0.924	0.932	0.312	0.923	0.931	0.326
B	0.970	0.934	0.931	0.365	0.929	0.930	0.433
C	0.915	0.707	0.665	0.677	0.786	0.788	0.594
D	0.958	0.924	0.928	0.335	0.902	0.916	0.456
E	0.994	0.973	0.974	0.795	0.971	0.971	0.793
F	0.913	0.852	0.843	0.619	0.867	0.862	0.636
G	0.962	0.942	0.943	0.367	0.952	0.951	0.418
H	0.892	0.909	0.909	0.613	0.914	0.913	0.618
I	0.939	0.956	0.958	0.591	0.950	0.949	0.449
J	0.975	1.000	1.000	0.792	1.000	1.000	0.747
Weighted Avg.	0.941	0.897	0.891	0.540	0.908	0.910	0.547

Table 2. AUC results for the typist dataset, for standard bagged decision trees, the combined one-class classifier, and its components

The final typist dataset used here contains these 8 attributes and 10 class labels (Participants A–J).[3]

Table 2 shows the results obtained for the 10 different typist classes, in each case treating one of these classes as the target class, and the union of the other classes as the outlier class. Each row states AUC values for one particular target class. The bottom row has the weighted AUC value, calculated according to Equation 2.

The results from the typist dataset are similar to those from the UCI datasets: using only the class probability estimator, i.e. $P(T|X)$, from the combined model, results in poor performance, whereas using just the reference density, i.e. $P(X|A)$, sometimes performs better than the combined model. More specifically, considering the case of the Gaussian reference density, the win/loss ratio for the combined model vs. the reference density is 3/5; considering the case of the EM-based density it is 4/4. According to overall weighted AUC, the combined model has an edge in the Gaussian case, but is outperformed slightly in the case of EM.

Considering performance relative to the baseline multi-class classifier, the overall picture is similar as in the case of the UCI datasets: multi-class classification outperforms one-class classification. However, surprisingly, for three of the user classes—H, I and J—the combined one-class classifier and the reference densities have a higher AUC than the baseline classifier. This is not unusual; we experienced similar results on individual class labels on many of the UCI datasets. In the context of our target application, typist recognition, this is exciting because it demonstrates that one-class classification can be a very satisfactory substitute for multi-class classifiers, and one that is better suited to solving the problem at hand because it does not require any training data from the outlier classes in order to achieve good results: when using typist recognition

[3] Available for download at http://www.cs.waikato.ac.nz/ml/data/typist.arff

	libSVM			OCC Gaussian		OCC EM	
Dataset	γ	FAR	IPR	FAR	IPR	FAR	IPR
diabetes	0.00005	0.111	0.514	0.098	0.857	0.109	0.779
ecoli	0.1	0.137	0.068	0.129	0.088	0.136	0.083
glass	0.005	0.154	0.412	0.147	0.434	0.180	0.331
heart-statlog	0.0001	0.122	0.624	0.140	0.507	0.141	0.504
ionosphere	0.00005	0.128	0.738	0.150	0.732	0.169	0.697
iris	0.0005	0.120	0.073	0.125	0.076	0.137	0.077
letter	0.000005	0.101	0.516	0.100	0.291	0.105	0.215
mfeat-karhunen	0.0001	0.131	0.034	0.094	0.114	0.105	0.092
mfeat-morphological	0.0000001	0.110	0.206	0.068	0.128	0.075	0.134
mfeat-zernike	0.000001	0.116	0.253	0.085	0.324	0.099	0.276
optdigits	0.00005	0.106	0.087	0.107	0.084	0.122	0.087
pendigits	0.000001	0.103	0.203	0.100	0.116	0.102	0.137
sonar	0.001	0.120	0.705	0.123	0.815	0.163	0.751
vehicle	0.00005	0.103	0.629	0.109	0.645	0.130	0.494
waveform-5000	0.001	0.103	0.307	0.075	0.411	0.110	0.354
typist	0.00005	0.113	0.331	0.113	0.204	0.147	0.157

Table 3. Results for one-class support vector machines (obtained with libSVM) versus the combined one-class classifier (OCC)

for computer security we simply do not have access to training data for new classes of typists that correspond to impostors (or "attackers").

5.3 LibSVM

To compare our method—again, using bagged unpruned decision trees—to an established one-class classifier, we now discuss results obtained in a comparison to the one-class classifier in libSVM [3]. The results are shown in Table 3. As mentioned in Section 4, we could not use AUC for comparison, and instead resort to reporting FAR and IPR. For each dataset, the table reports the value of the γ parameter for the RBF kernel that was used to obtain the results shown.

Although we were generally unable to match FAR exactly for the methods compared, the results nevertheless enable us to make a qualitative statement about their relative performance. In particular, we can consider cases where *both* FAR and IPR are lower for one method in a pair of methods being compared. Doing this for libSVM and our Gaussian-based combined classifier, we see that there are two datasets where both FAR and IPR are lower for our method (letter and pendigits) and two cases where they are both lower for libSVM (sonar and vehicle). Repeating this exercise for libSVM and our combined model with the EM-based density, we find that there is one dataset where our method is better for both statistics (pendigits) and three datasets where this is the case for libSVM (iris, sonar, and waveform-5000). Overall, we can say that there are datasets where the SVM appears to be the better approach and other datasets where our method appears to perform better. Note that, in contrast to the SVM, FAR and

IPR can be adjusted at prediction time in our method. Moreover, our method does not require parameter tuning to return satisfactory results.

6 Conclusion

In this paper we have presented a new method for combining density estimation with class probability estimation for the purpose of one-class classification. The basic idea is to use a density estimator to build a reference density for the target class, then use this reference density to generate artificial data for a two-class learning problem suitable for a class probability estimation technique, and finally combine the predictions of the reference density and the class probability model to form predictions for new test cases.

Using experimental results obtained on UCI datasets and a continuous typist recognition problem, and using bagged unpruned decision trees as the underlying class probability estimator, we have shown that our combined model can indeed improve on both component techniques: the density estimator used to obtain the reference density and the class probability estimator trained on the semi-artificial two-class learning problem.

We have also compared our combined model to a one-class support vector machine with a RBF kernel. The results show that there are datasets where our method is superior and other datasets where the one-class SVM performs better. Our method has the advantage that, like in standard density estimation techniques for one-class classification, there is no need to specify a target rejection rate at training time.

A significant feature of our method is that it is generic, and hence can be used in conjunction with arbitrary density estimators and class probability estimation techniques. We believe that this is the primary advantage of our technique, given the availability of large collections of suitable candidate base learners in machine learning workbenches.

An interesting avenue for future work is the experimental comparison of variants of our technique that can be obtained by plugging in different types of base learners. Another important question is how the quantity of artificial data that is generated using the reference distribution influences the result. There must clearly be diminishing returns, but one would expect that, in general, more data leads to a more accurate combined model. On a more fundamental level, it would be interesting to investigate whether it is possible to avoid the artificial data generation step by adapting class probability estimators to take advantage of information in a reference distribution directly. However, it is obvious that this cannot be achieved without changing the learning algorithm involved; thus the generic aspect of the method would be lost.

References

1. 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, pages 767–772, New York, NY, USA, 2006. ACM Press.

2. V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, West Sussex, England, 1994.
3. C. Chang and C. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
4. P. Dowland, S. Furnell, and M. Papadaki. Keystroke analysis as a method of advanced user authentication and response. In *Proceedings of the IFIP TC11 17th International Conference on Information Security*, pages 215–226, Deventer, The Netherlands, 2002. Kluwer.
5. D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security*, 8(3):312–347, 2005.
6. F. Monrose and A. Rubin. Keystroke dynamics as a biometric for authentication. In *Future Generation Computer Systems 16*, pages 351–359. Elsevier Science, 2000.
7. M. Nisenson, I. Yariv, R. El-Yaniv, and R. Meir. Towards behaviometric security systems: Learning to identify a typist. In *Proceedings of the International Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 363–374, Berlin, 2003. Springer-Verlag.
8. R. Pearson. *Mining Imperfect Data*. Society for Industrial and Applied Mechanics, USA, 2005.
9. F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.
10. V. Roth. Kernel fisher discriminants for outlier detection. *Neural Computing*, 18(4):942–960, 2006.
11. B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12*, pages 582–588. MIT Press, 2000.
12. L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proceedings of the Fourth International IEEE Conference on Artificial Neural Networks*, pages 442–447, London, 1995. IEEE.
13. D. Tax. *One-class Classification, Concept-learning in the Absence of Counter-examples*. PhD thesis, Delft University of Technology, Netherlands, 2001.
14. D. Tax and R. Duin. Combining one-class classifiers. In *Proceedings of the Second International Workshop on Multiple Classifier Systems*, pages 299–308, Berlin, 2001. Springer-Verlag.
15. A. Ypma and R. Duin. Support objects for domain approximation. In *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 719–724, Berlin, 1998. Springer-Verlag.