

COMBINING CLASSIFIERS FOR FOREIGN PATTERN REJECTION

Władysław Homenda^{1,2,*}, Agnieszka Jastrzębska^{1,3}, Witold Pedrycz^{4,3}, Fusheng Yu⁵

¹*The Faculty of Mathematics and Information Science
Warsaw University of Technology, Poland*

²*The Faculty of Economics and Informatics in Vilnius
University of Białystok, Vilnius, Lithuania*

³*The Systems Research Institute
Polish Academy of Sciences, Warsaw, Poland*

⁴*The University of Alberta, Edmonton, Canada*

⁵*The Beijing Normal University, Beijing, China*

*E-mail: homenda@mini.pw.edu.pl

Submitted: 9th October 2019; Accepted: 16th February 2020

Abstract

In this paper, we look closely at the issue of contaminated data sets, where apart from legitimate (proper) patterns we encounter erroneous patterns. In a typical scenario, the classification of a contaminated data set is always negatively influenced by garbage patterns (referred to as foreign patterns). Ideally, we would like to remove them from the data set entirely. The paper is devoted to comparison and analysis of three different models capable to perform classification of proper patterns with rejection of foreign patterns. It should be stressed that the studied models are constructed using proper patterns only, and no knowledge about the characteristics of foreign patterns is needed. The methods are illustrated with a case study of handwritten digits recognition, but the proposed approach itself is formulated in a general manner. Therefore, it can be applied to different problems. We have distinguished three structures: global, local, and embedded, all capable to eliminate foreign patterns while performing classification of proper patterns at the same time. A comparison of the proposed models shows that the embedded structure provides the best results but at the cost of a relatively high model complexity. The local architecture provides satisfying results and at the same time is relatively simple.

Keywords: data mining, knowledge engineering

1 Introduction

Classification is one of the most popular problems in the area of machine learning. There is a vast amount of literature devoted exclusively to various classification methods. However, it is worth to approach the task of classification from the point of

view of practical problems. Poor data quality is one of the issues that affect classification results. A specific example of data quality issue is the contamination of data sets. Contaminated data sets consist not only of relevant (proper) patterns but also of irrelevant (erroneous) patterns, which, for instance, appeared by an error. In this paper, we use names of

native patterns and *foreign patterns* to distinguish between legitimate (proper) and erroneous, abnormal patterns. It is worth to notice, that foreign patterns cannot be seen as outliers, novelties, etc., because we cannot assume that their origins, characteristics, etc. are known.

It is worth to underline, that foreign patterns are any abnormal input data presented to a classifiers already constructed and that no assumption about foreign patterns can be made at the stage of the classifier construction.

In a standard pattern recognition task, all patterns are assigned to one of the available classes. Foreign patterns negatively affect the quality of classification, as they do not belong to any class, but a classifier assigns a class label anyway. One may suggest a straightforward solution, to treat foreign patterns as yet another class and train a classifier on such an extended set of classes. Unfortunately, this approach is not feasible, because, as we mentioned, we cannot assume that representatives of foreign patterns are known at the stage of classifier construction. Moreover, even when some foreign patterns are available at the stage of classifier construction, we cannot assume that they all will form a coherent class or classes in the future. We have to mind, that foreign patterns may be highly dissimilar to patterns belonging to other classes. Therefore, it is necessary to propose a generalised solution, which is not limited to rejecting certain kinds of foreign patterns.

Having the above in mind, we see the need for classification models reinforced with a foreign pattern rejection option. Such models should be trained on native data exclusively, and they should be able to:

- a) reject foreign patterns, what is the main goal of the research presented in this study,
- b) classify native patterns, what can be seen as a complementary goal.

The objective of the study presented in this paper is to propose various classifier compositions (cascading classifiers) that put together at a specific order and trained according to specific schemes are able to achieve the above-mentioned tasks. We intended to make use of already existing data processing algorithms in order to provide ensemble models

that are capable not only of classification but also of foreign patterns of rejection.

This research was motivated by our experiences with various real-world applications dealing with pattern recognition. Contaminations in data originate due to many reasons: blurry images, not properly isolated signals, multiple objects surrounding an object of interest, noisy environments, technicians' errors, etc. It is worth to stress that no assumption is made on contamination reason, anyone listed here is possible. We believe, that applying and building on known algorithms has its practical benefits. Implementation of processing schemes introduced in this paper does not require a change of data analysis environment and programming language. We see the designed rejection tools as an additional component in a broader scheme of pattern recognition. Therefore, implementation of the proposed methods can be performed based on a classifier that one planned to use anyway. Alternatively, one can employ any other classifiers as a base for the rejection mechanism.

The novelty of the presented study is not in the particular algorithms we use, but in the way how we use them and what we achieve. The proposed schemes for foreign patterns rejection based on compositions of classifiers is a novel contribution. In contrast to the methods available in the literature, the choice of a particular classification algorithm performing classification and rejection tasks depends on the model designer, what assures substantial flexibility.

In this paper, we summarise progress in the area of foreign patterns rejection based on ensemble classifiers. We present a thorough overview and comparison of various models. The study addressed in this paper is a continuation of our work reported in [1].

The paper is structured as follows. In Section 2, we present a brief literature review on data processing methods that deal with foreign patterns. Section 3 is devoted to a theoretical presentation of different classifier compositions: global, local, and embedded. In Sections 4 and 5, we present experiments. Section 6 concludes the paper.

2 Literature Review

Research on the issue of contaminated data sets evolved over time. Studies on outliers were one of the most important start points. Among early works on outliers, we find [2] and [3]. Outliers belong to a native data set, but they differ substantially from the majority of data. It was an apparent observation, that some processing methods, including popular least squares estimates for regression models or some centroid-based clustering techniques, were not robust with respect to outliers. Hence, researchers developed a range of methods for outlier detection. Many of these approaches tell to eliminate observations, that significantly differ from some central tendency in data. A suitable cut threshold could be based on the data dispersion measure. A simple example is to apply the Chauvenet's criterion for discarding outlying observations based on mean and standard deviation, which was recently elaborated on in [4]. Another simple test for outlier detection is the Grubb's test also based on sample mean and on the maximal distance between the mean and a data point, which was recently applied in an interesting study in [5]. Another well-known statistical test for outlier identification is the Tukey's HSD (honest significant difference) test which uses pairwise comparisons of means, recently revisited in [6]. It shall be stressed, that the recalled research has been conceived on the grounds of statistics and there are certain assumptions on data properties, which have to be satisfied, in order to conduct those tests.

The variety of problems that fall into the scope of machine learning demanded new solutions to deal with contaminated data sets. Just removing outliers turned out to be insufficient. The main drawback of removing outliers is that it decreases the size of a native data set, what in some applications is not acceptable. Native data points that are dissimilar to the majority of data could represent infrequent, but very valuable subjects. Removing such data points from a training set makes it impossible for a recognition algorithm to learn to classify them. The necessity to keep outliers in a data set is especially crucial in active learning schemes, in which we want a learning algorithm to gradually adjust to an incrementally changing data stream.

In addition, we must acknowledge, that in a challenging data set, apart from outliers, which are actually native patterns but highly dissimilar to the majority of data, we may encounter other kinds of patterns. In particular, let us now focus on the novelty detection task, which could be viewed as a special case of the foreign pattern rejection problem.

Novelty detection is the task of identifying unseen data, that differs from the data available during training, [7]. In other words, we aspire to form a classifying model, that has been trained based on samples from the proper class only but is capable to distinguish proper class samples from previously unseen novelty patterns. A typical scenario, in which we employ a novelty detection algorithm, is when a minority class is extremely infrequent, and we are unable to form a classifying model capable to correctly recognise the infrequent class. A domain, in which novelty classification is frequently applied, is text mining. It is worth to mention, that in the literature novelty detection is sometimes termed as one-class classification. Novelty detection, even when it concerns a data set with multiple classes, can be reduced to the task of constructing a one-class model. Elements not accounted to this one class are treated as novelties.

There is a wide spectrum of probabilistic approaches to novelty detection. This group of methods links past theoretical research on outliers with modern data processing algorithms. The underlying objective of these methods is to estimate a generative probability density function of the training data. Assuming that we have found out data distribution, we may propose a threshold-based method that distinguishes regular data from novelties. Since challenging data require sophisticated methods, it became a popular technique to represent data distribution using a mixture of models. Particular examples of such approach are based on Gaussian distributions (the so-called Gaussian mixture models), [8], but other distributions were considered as well. For instance, mixtures based on gamma distribution are considered in [9], while Poisson mixtures are addressed in [10]. Model parameters need to be estimated, for instance, using maximum likelihood methods. Mixture-based methods rely on a relatively small number of distinct distributions, usually fewer than the number of classes in a data. In con-

trast, a related group of the so-called density estimation methods requires a large number of distinct kernels to cover the data, [11].

Among other noteworthy approaches to novelty detection, we shall mention methods tightly connected to the notion of distance. They operate in a multidimensional space of features and rely on the assumption that proper data forms clusters in this space, while novel elements are scattered far from these clusters. In contrast to statistical approaches, application of these methods is not limited to data following a specific distribution. However, it is necessary to select a distance measure and an evaluation method. For example, in [12], we read about a method evaluating distance of a potential novelty to its k nearest neighbours. In [13], we find an approach relying on a distance to an average of \underline{k} nearest neighbours. Another noteworthy novelty detection method, the so-called Local Outlier Factor, has been presented in [14]. This method calculates ratios of local density of an area around a potential novelty element and local densities of its neighbours. Later, the Local Outlier Factor method has been modified and improved, in order to efficiently deal with more challenging and large data sets. Furthermore, a variety of approaches to novelty detection is based on clustering algorithms, for instance, the fuzzy c -means [15] or the agglomerative clustering [16].

A well-known novelty detection algorithm is one-class SVM (Support Vector Machines), often referred to as ν -SVM or novelty detection SVM. An in-depth elaboration on this method is presented in [17] and [18]. The one-class SVM detects a soft boundary of a set. In consequence, we are able to assess, which elements surely belong to this class, and which do not. A training procedure of the ν -SVM is governed by the $0 < \nu \leq 1$ parameter, which should be tuned in an appropriate validation procedure. Too small value of the ν increases the risk of overfitting, too large may cause underfitting.

An interesting approach to anomaly detection based on a multiple kernel learning approach for the One-class Classification task is outlined in [19]. Localized Multiple Kernel learning approach for Anomaly Detection with One-class Classification method is proposed as an extension of known Multi Kernel Anomaly Detection, LMKAD, algorithm. LMKAD provides a localized formulation

for multi-kernel learning method by local assignment of weights to each kernel.

Moreover, the literature offers approaches to foreign patterns rejection, in which model training relies on a synthesised set of foreign patterns. In other words, foreign patterns are treated as an additional class. Hence, by executing an extended classification procedure, we identify them. Among papers discussing this method, we find [20]. In our opinion, this approach has a limited practical potential. It is quite unrealistic to expect, that we know in advance features of foreign patterns. If we do know them, then are they really foreign?

It has to be stressed, that our ultimate goal was to propose methods based on native patterns only. Hence, the ideas discussed in this paper are closer to the research on novelty detection than to outliers patterns rejection approaches with synthesised additional patterns. The closest counterpart to the approach presented in this paper could be found in the paper by Tax and Duin [21]. In the cited paper, the authors proposed to combine one-class classifiers, in order to solve the outliers identification problem in a data set of handwritten digits. The study shows, that the best combination of individual one-class classifiers used the Parzen density estimator. A noteworthy observation was made, that combining classifiers trained in different feature spaces increased chances for outlier detection.

In light of the described developments in the areas of foreign patterns rejection and the related fields, let us highlight the advantages of the contribution introduced in this paper.

- We propose an approach to foreign patterns rejection, which could be applied to a multi-class data set.
- A model is trained only on native data. No knowledge about foreign patterns is needed to construct the model.
- Rejection method relies on standard classifiers, which are trained and set together in a specific way.
- The proposed approach can be used in order to improve classification quality in a dataset of purely native patterns.

3 Pattern Recognition with Foreign Patterns Rejection

3.1 Classification – Basic Notions

Let us define a classification problem as an action of dividing a set of patterns into subsets based on their similarity. Let us assume, that $\mathbf{S} = S_1 \cup S_2 \cup \dots \cup S_{|\mathbf{C}|}$, where $|\mathbf{C}|$ is the number of all classes in the data set, \mathbf{S} is the set of objects and $S_1, \dots, S_{|\mathbf{C}|}$ are its subsets that are pairwise disjoint: $(\forall i \neq j)(S_i \cap S_j = \emptyset)$. A mapping $\sigma : \mathbf{S} \rightarrow \mathbf{C}$, where $\mathbf{C} = \{1, 2, \dots, |\mathbf{C}|\}$ is the goal of pattern recognition (goal of classification).

We represent a pattern which we want to classify with a vector of measurable characteristics, which are called features, and then perform classification on such description format. This can be expressed as two mappings: $\phi : \mathbf{S} \rightarrow \mathbf{X}$ and $\omega : \mathbf{X} \rightarrow \mathbf{C}$. The first mapping ϕ is from the space of objects to the space of features. The second mapping ω is from the space of features to the space of classes. It can be seen, that $\sigma = \omega \circ \phi$ and from now on σ can be referred to as the classifier. Moreover, let us mention, that the space of features is the real-coordinate space of n dimensions: \mathbb{R}^n .

It should be mentioned, that the set denoted as \mathbf{S} contains all patterns. Obviously, it is not possible to obtain such a set. Therefore, we construct a classification mechanism on an available subset of \mathbf{S} . The available subset is denoted as \mathbf{L} , $\mathbf{S} \supset \mathbf{L} = L_1 \cup L_2 \cup \dots \cup L_{|\mathbf{C}|}$ such that $(\forall i \in \langle 1, |\mathbf{C}| \rangle)(L_i \subset S_i)$. \mathbf{L} is called a learning set. Furthermore, we split the learning set into a training set (\mathbf{Tr}) and a test set (\mathbf{Ts}) as follows: $\mathbf{L} = \mathbf{Tr} \cup \mathbf{Ts}$. Each class from the learning set is split into the training set and the test set, namely: $(\forall i \in \langle 1, |\mathbf{C}| \rangle)(Tr_i \cup Ts_i = L_i$ and $Tr_i \cap Ts_i = \emptyset)$ and $\mathbf{Tr} = Tr_1 \cup Tr_2 \cup \dots \cup Tr_{|\mathbf{C}|}$ along with $\mathbf{Ts} = Ts_1 \cup Ts_2 \cup \dots \cup Ts_{|\mathbf{C}|}$. The training set, comprising of native patterns only, is used for model construction. The test set is the so-called unseen data and it is used for model quality assessment.

In this paper, we expand this model, by considering an extra set of foreign patterns S_f , which extends the set of patterns $\mathbf{S} = \{S_1, S_2, \dots, S_{|\mathbf{C}|}\}$ to the set $\mathbf{S}_f = \{S_1, S_2, \dots, S_{|\mathbf{C}|}, S_f\}$. Notice, that the set \mathbf{S} is in fact the set of native patterns.

The fundamental assumption is that foreign patterns are not available at the stage of recogniser construction. A standard classifier assigns class label to every pattern presented to it. Only these labels, which have appeared in the training set of native patterns are available. This implies, that foreign patterns will be always incorrectly classified at the stage of classification.

3.2 Rejecting Foreign Patterns: Ideas and Architectures

We envision the classification task as a partitioning of a set of patterns into subsets of patterns that belong to the same class. When we deal with a contaminated data set, classification should be supplemented with a procedure of foreign patterns removal (rejection) so that foreign patterns do not end up in subsets reserved for native patterns. A “physical” output of a classifying and rejecting mechanism is a labelling of input patterns. The mechanism assigns either one of the native class labels or a special label marking a foreign pattern.

The approach, that we propose, is based on specifically trained classifiers. We compose them to a certain structure (one may say: architecture) and they together provide a mechanism for native patterns classification with foreign patterns rejection. Depending on the order of actions (classification/rejection) we distinguish three different structures (architectures):

- global,
- local,
- embedded.

Proposed names reflect the level at which we perform rejection of foreign patterns. The three structures are illustrated in Figure 1.

It is of an utmost importance to distinguish between the terms “architecture” and “rejection mechanism”. Architectures (global, local, and embedded) are scenarios (in other words: schemes, orders of actions), in which we perform classification and rejection. In contrast, a rejection mechanism is a data processing mechanism, which after an appropriate training is able to process a contaminated set and reject foreign patterns from it. Section 3.3

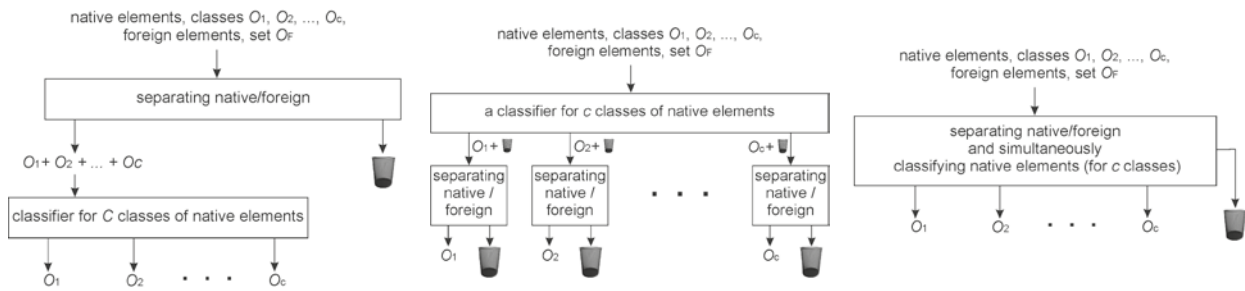


Figure 1. Rejecting architectures: global (left diagram), local (middle diagram) and embedded (right diagram). Bins represent rejected patterns.

is devoted to the construction of rejection mechanisms.

3.2.1 The Global Architecture

The global architecture is presented in the left diagram in Figure 1. In this scenario, at first, we reject, ideally all, foreign patterns from the input data set. Then, we classify patterns that were not rejected. The rejection mechanism has to be able to identify all native patterns and distinguish them from foreign patterns. This is a challenging task, because native patterns may be very different. If the rejection mechanism is not as good as we wish, there is no other option to recover a rejected native pattern or to reject an accepted foreign pattern.

3.2.2 The Local Architecture

The local architecture is presented in the middle diagram in Figure 1. In this scenario, at first, we classify all incoming patterns. In this case, it is necessary to form c rejection mechanisms, one per each native class.

We expect, that the classifier will split the incoming data into c subsets. Ideally, all native patterns will be classified correctly. However, foreign patterns will get classified as well, because a classifier always assigns a class label. We may assume, that native patterns belonging to the same class are in some sense similar. Therefore, a rejection mechanism positioned in a leaf in the local architecture has to distinguish foreign patterns (that could be very diverse) from native patterns of just one class (that should form a consistent set).

Let us note, that it is also possible to implement the local architecture with more than c rejection mechanisms. In such case, we would have to

perform a fine-grained analysis of native patterns in each class and split a single class of native patterns into a few subclasses based on pattern similarity. In this scenario, we shall employ unsupervised learning in order to split a single class. However, we have to be aware that this action may lead to overfitting. Implementing the local architecture with c leaves is a justified choice, especially for balanced data.

It is also possible and desirable, that a rejection mechanism rejects not only foreign patterns, but also misclassified native patterns. The capability of a rejection mechanism to reduce the number of misclassified native patterns is a natural property of the local architecture as rejection mechanisms may be constructed on more coherent data than in the case of the global architecture.

3.2.3 The Embedded Architecture

The embedded architecture is depicted in the right diagram in Figure 1. In this scheme, each processed pattern is pushed down into a c -class classifier until it reaches a leaf, where a class label is assigned. Actions performed by the c -class classifier are followed by a rejection procedure, where we have a chance to remove foreign patterns. The embedded architecture is the most fine-grained.

Let us go ahead the main narration at this point and present a particular example of an embedded architecture implemented in experiments discussed later in this paper. It is a model based on binary classifiers organised in a binary tree. We are dealing with the problem of handwritten digits recognition, and the data set we have is contaminated with various foreign patterns. In a similar way as in [22], we applied spectral clustering to evaluate

the similarity of native classes. It revealed, that the best binary split for the entire native data set is into the following subsets of classes: $\{0, 2, 3, 5, 6, 8\}$ and $\{1, 4, 7, 9\}$. Then, the first subset is split into $\{0, 6\}$ and $\{2, 3, 5, 8\}$, and so on. The full structure is displayed in Figure 2. Let us reiterate, that in the embedded architecture a tree of binary classifiers, which consecutively splits data into smaller subsets, is the base for classifying and rejecting mechanism. A formed tree is supplemented with rejection mechanisms, which are placed after each binary split except the split in the root. The reason is that a rejecting mechanism designed as described in Section 3.3 is useless in the root.

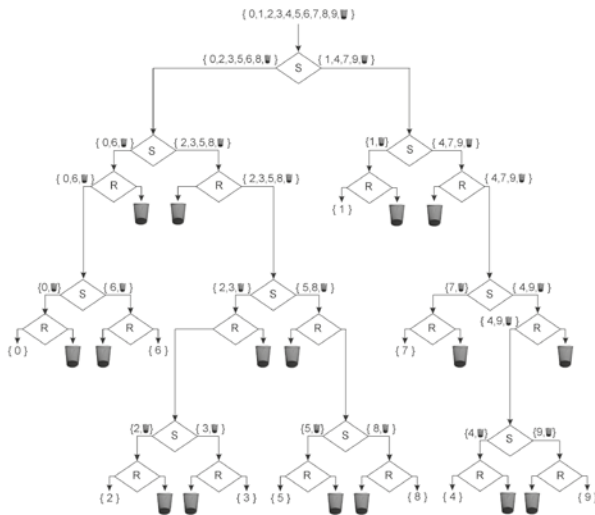


Figure 2. An illustration of the embedded architecture implemented for the case study of handwritten digits recognition. “S” denotes a binary split into two subsets, “R” denotes a rejection mechanism. A bin represents foreign patterns.

As we see in Figure 2, even for a relatively simple data set with only ten classes, the embedded architecture is relatively complex. It is possible to reduce this complexity by removing some rejection mechanisms.

The proposed architectures differ in their complexity. The simplest is the global rejection architecture. In this case, we have one rejection mechanism and one c -class classifier (c denotes the number of native classes). The local rejection architecture has one c -class classifier and c rejection mechanisms. The most complicated is the embedded ar-

chitecture, where we have $c - 1$ binary classifiers and in the most complex case $2 \cdot (c - 1)$ rejection mechanisms.

3.3 Construction of Rejection Mechanisms

Up to this point, we have only discussed scenarios, in which we perform actions leading to the classification of native patterns and rejection of foreign patterns. In order to adapt the method for a particular data processing problem, it is necessary to:

- select classifiers (a c -class classifier for the global and the local architecture, a tree of binary classifiers for the embedded architecture),
- select rejection mechanisms.

Selection and training of classifiers for the global and the local architecture do not differ from a usual multi-class classification scenario. It is up to the model designer to choose an appropriate classification method. A non-typical classifier is required in the embedded architecture. However, combining binary classifiers for a multi-class classification is an approach already present in the literature, for instance in [23] and [24].

The second important component of the proposed architectures, that needs customization is a rejection mechanism. Alike classifiers, rejection mechanisms have to be formed based on native patterns only. It is worth to underline, that we have no knowledge about foreign patterns at the stage of model construction. Therefore, rejection mechanisms are realised with appropriately trained classifiers. There are two classifier learning strategies, which we may choose: one-class and binary.

3.3.1 One-class Classifiers for Rejection

One-class rejection mechanisms, by analogy to novelty detection methods presented in Section 2, aim at designing a model that describes native patterns. A one-class rejection mechanism treats all native patterns as if they belong to a single class and provides a decision rule for the identification of native patterns. Let us recall, that these approaches involve approximations of native data distribution, often apply notions of similarity or proximity.

One-class rejection mechanisms have not been considered in the study presented in this paper, be-

cause they are too well-known. However, we have applied them alone (not as an element of an architecture) for the sake of comparison. In Section 5.5, we contrast our results with literature-based approaches.

3.3.2 Binary Classifiers for Rejection

Binary rejection method aims at splitting a set of patterns into two sets: native and foreign. It is composed of one or a few binary classifiers trained only on native patterns. Let us reiterate, we never involve foreign patterns in the training process. The ability to discriminate between native and foreign patterns is gained due to a specially conducted training procedure. In a binary classifier trained for rejection, we distinguish a native class (let us call it a “pro” class), and the second class, which is a special substitute (let us call it a “contra” class). A rejection mechanism based on a collection of binary classifiers is suited well to deal with multi-class classification problem.

In this paper, we assume that a given set of native patterns creates the class “pro”, while remaining native patterns form the class “contra”. For example, let us consider classes $\{2, 3, 5, 8\}$ in one of the nodes of the tree, cf. Figure 2. This set of classes is split to two subsets of similar classes ($\{2, 3\}$ and $\{5, 8\}$). Thus, the native set $\{2, 3\}$ creates the class “pro”, while the set of the other classes ($\{0, 1, 4, 5, 6, 7, 8, 9\}$) plays the role of the “contra” class. In other words, the rejection mechanism accompanying native classes $\{2, 3\}$ is a binary classifier that distinguishes native classes $\{2, 3\}$ versus all other native classes in the data set. Alike, the set of digits $\{5, 8\}$ creates the class “pro”, while the set of digits $\{0, 1, 2, 3, 4, 6, 7, 9\}$ forms the class “contra”. Binary rejection mechanisms in the embedded architecture need to reject foreign patterns from subsets that are designed to contain more than one class of native patterns.

The described approach is straightforward for the local rejection architecture, where rejection is launched separately for each single native class (see the middle diagram in Figure 1). In the local architecture, we construct c rejection mechanisms, so we train c binary classifiers one-versus-all-else for rejecting.

The situation is slightly different in the global architecture, where a rejection mechanism needs to reject foreign patterns from a set of all native patterns (cf. the left diagram in Figure 1). In such a case, we build c binary classifiers, one for each native class, alike in the local architecture, and then employ a simple voting rule. If we want to determine whether a given pattern is foreign, we look if any rejection mechanism accounted it as native. Only if none of the rejection mechanisms say that the pattern is native, we reject it. Since this method is firmly based on known (native) classes, it may be seen as a supervised scheme. This method may be also implemented in an unsupervised mode, when no split of native patterns to classes is known. We discuss such example in Section 5.4.

It shall be mentioned that the procedure of foreign patterns rejection may be imperfect. It means, that not all foreign patterns may get rejected and some of native patterns may get rejected too. What is more, native patterns classification may be imperfect as well. Not rejecting a foreign pattern is always an unfavourable situation. However, when rejecting a native pattern, we may envision two cases:

- a native pattern, which would have been correctly classified, is rejected. This is an unwelcome situation, because it decreases the number of correctly classified patterns and, thus, it worsens recognition quality.
- a native pattern, that would have been incorrectly classified, i.e. a classifier would have assigned an incorrect class label, is rejected. In this scenario, rejecting mechanism increases the quality of classification.

Therefore, adding a rejection mechanism could be justified as well, when the cost of misclassification is high, and it is better to reject a native pattern than to classify it into an incorrect class.

3.4 Model Quality Assessment

A study on foreign patterns rejection entails the need for appropriate quality assessment methods. Standard techniques, typically applied to evaluate classification effectiveness, need to be adapted, in order to describe the effectiveness of classification reinforced with rejection. It has to be stressed, that a badly designed rejection mechanism may hinder

the quality of classification, while a good rejection mechanism can improve it. Hence, we shall look closely both at classification and rejection rates. The following notions are used:

- CC (Correctly Classified) – the number of native patterns with a correct class label,
- TP (True Positives) – the number of native patterns classified as native (no matter, with which native class label),
- FN (False Negatives) – the number of rejected native patterns,
- FP (False Positives) – the number of foreign patterns incorrectly classified as native,
- TN (True Negatives) – the number of rejected foreign patterns.

Note: TP, FN, FP and TN are widely used in literature in the context of binary pattern recognition. Here, we adapt them to describe rejection quality. Based on these notions, we define the following measures to evaluate the quality of a classifying and rejecting model:

$$\begin{aligned}
 \text{Accuracy} &= (TP+TN)/(TP+FN+FP+TN) \\
 \text{Strict Accuracy} &= (CC+TN)/(TP+FN+FP+TN) \\
 \text{Fine Accuracy} &= CC/TP \\
 \text{Native Precision} &= TP/(TP+FP) \\
 \text{Native Sensitivity} &= TP/(TP+FN) \\
 \text{Strict Native Sens.} &= CC/(TP+FN) \\
 \text{Foreign Precision} &= TN/(TN+FN) \\
 \text{Foreign Sensitivity} &= TN/(TN+FP) \\
 \text{F-measure} &= 2 \cdot \frac{\text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}
 \end{aligned}$$

We have already discussed the above characteristics in [22]. Hence, we do not repeat this information in this paper.

The higher the value of these measures, the better the quality of classification with rejection. However, an increase in one measure can lead to a decrease in another. In practice, depending on application, one measure may reveal more important information than another. For instance, if a priority is given to minimization of the number of foreign patterns identified as native, then Native Precision should be of a higher importance than the other

measures and it should be maximised. On the other hand, if the highest priority is given to minimization of a loss of native patterns, then one should focus on Native Sensitivity, and so on.

Pattern recognition reinforced with a rejection mechanism employed to process a set of native patterns may be a viable choice for pattern recognition problems, in which misclassification might generate much greater loss than a lack of classification. Two measures are appropriate to evaluate the quality of a rejecting mechanism in this case: Strict Native Sensitivity and Fine Accuracy. It is worth to notice, that if foreign patterns are absent, quantities such as TN and FP are not relevant. Therefore, Accuracy becomes identical to Native Sensitivity and Strict Accuracy to Strict Native Sensitivity. Furthermore, in the case of pure recognition (that is, without rejection) only two quantities: CC and TP are relevant. Therefore, it only makes sense to calculate the ratio of correctly classified patterns to all processed patterns. If we assume that CC is the number correctly classified patterns and TP is the number of all patterns being processed, while TN, FP and FN are absent, then Strict Accuracy and Strict Native Sensitivity are identical to Fine Accuracy.

4 Empirical Study – Settings

4.1 Data Sets

4.1.1 Native Patterns

The empirical study is focused on handwritten digits recognition. The data set of native patterns consists of 10,000 images of handwritten digits, approximately equally divided into ten classes ($c = 10$). It is publicly available in the MNIST database, [25]. Figure 3 presents samples of native handwritten digits. The data was split into training and native test sets in proportion 6999 and 3001 patterns, respectively. The training set was used for model construction, while the test set for quality evaluation only. In the experiments presented in this paper, the native test set was contaminated using various methods, as described in Section 4.1.2. Contaminated data sets were presented to the input of formed rejection/classification models to evaluate the quality of the outcome. We use measures presented in Section 3.4. In order to provide a fair comparison, if the number of patterns in a data set

of foreign patterns differs from the number of patterns in the native set, we use scaling factors, so that the influence of foreign patterns on the quality measure is the same as the influence of native patterns. Some of foreign data sets (we describe them in Sections 4.1.2 – 4.1.4) contain different number of patterns than the native data set.

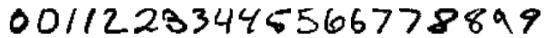


Figure 3. Samples of handwritten digits (native patterns).

Each native pattern has been described using a set of 106 numerical features, which we presented in [22] and we do not recall here due to space limitations. Features have been computed based on monochromatic (black and white) images. Out of 106 features, 26 have been selected using R package “FSelector”. We have used one of wrapper feature selection methods, namely forward search. It is a greedy search that starts from an empty set of features and adds one-by-one a new feature. Selection of a particular feature is performed using a ranking computed by training a classifier of choice on subsets of features. In our experiment, we have implemented an evaluation function based on a $c = 10$ -class SVM. Of course, it can be argued that wrapper methods for feature selection are computationally demanding and, ideally, shall be repeated when we change a classifier. On the other hand, wrapper methods are known to provide good results.

In addition, after the forward search returned 26 features, we performed analysis of variance, which led us towards elimination of two more features, which were highly correlated with others. In the end, our data set was made of 24 features.

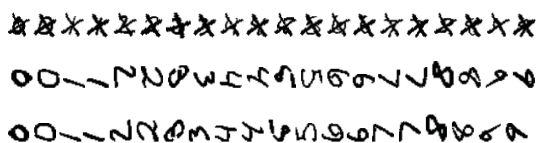


Figure 4. Samples of semi-synthetic foreign patterns. Distorted patterns, from the top: crossed out, rotated clockwise and rotated anticlockwise.

4.1.2 Semi-synthetic Foreign Patterns

For test purposes, we have prepared several sets of semi-synthetic foreign patterns. They were created based on the set of images of native patterns,

i.e. the set of handwritten digits. Images of handwritten digits were distorted, in order to obtain realistic foreign patterns, cf. next paragraph. It is worth to notice, that empty (white) patterns distorted with randomly inverted pixels are trivial to reject, as reported in [22].

In this study, the following semi-synthetic foreign patterns are considered:

- native patterns crossed out with digits 1 rotated by $\pm 45^\circ$, forming upper case “X” (we call it for short – X set),
- native patterns rotated clockwise and anticlockwise by 90° (for short – 90 set), the set of native patterns was randomly split to equal subsets to be subjected to rotations in both directions.

Samples of semi-synthetic foreign patterns are displayed in Figure 4. We may expect, that in a real-world handwritten digits processing scenario, it is also very likely, that foreign patterns would be distorted images of symbols of some kind. For instance, digits may be crossed out by an agent or a distortion may be generated by a malfunctioning optical character recognition software.

4.1.3 Handwritten Letters as Foreign Patterns

A data set of handwritten Latin letters is another kind of foreign patterns in our experiments. This set consisted of 26,383 patterns, ca. 1000 copies of each letter. It was created by 16 students, writing about 70 copies of each letter. Samples are presented in Figure 5. The foreign data set of handwritten letters was represented with the same 24-feature vector as the native set of handwritten digits.

4.1.4 Kannada Symbols as Foreign Patterns

Lastly, we consider a set of Kannada symbols as foreign patterns. Kannada is one of few Dravidian languages spoken mainly in India by ca. 50 million speakers. The set of Kannada letters comprises of 49 symbols. However, the actual number of characters in Kannada is larger, because single letters can be combined to form compound characters. Detailed information regarding used Kannada data sets, including a link to the data, is available in [26]. The set of handwritten Kannada characters contains over 650 very imbalanced classes. A few selected samples (after preprocessing) are displayed

in Figure 5. By analogy to the representation of native data, the foreign data set of Kannada characters was normalised and made monochromatic. In the end, the foreign data set of handwritten Kannada characters consisted of 9,107 samples. As in the case of the data set of native patterns, all Kannada samples were represented with a 24-dimensional feature vector.

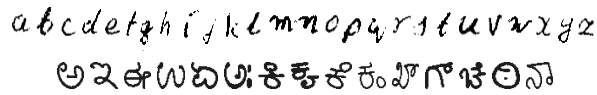


Figure 5. Samples from foreign data sets: handwritten Latin alphabet (the first row) and handwritten Kannada symbols (the second row).

4.2 Experiment Settings

The objective of the empirical study is to investigate and compare the three architectures: global, local, and embedded. We have selected random forest (RF) and Support Vector Machines (SVM) as particular classifiers to construct these architectures.

Both random forest and SVM are very popular machine learning methods. Hence, we do not continue with their in-depth description. In this study, they serve us as viable examples, which could be considered to construct classifying/rejecting architectures.

Random Forests

Random forest, introduced by L. Breiman and A. Cutler, is an ensemble machine learning method based on a multitude of appropriately formed decision trees, [27]. In our experiments, for each random forest, we trained by average 500 trees, the number of variables for which we obtained splits was individually tuned. We have been using a voting scheme to determine class belongingness, where each classifier in the ensemble votes for a class. We used “tuneRF” function from R package “randomForest” for tuning. The number of trees in a forest was selected experimentally from the set {300, 350, 400, 450, 500, 550, 600, 650}. Construction of random forests was realised with the function “randomForest” from the same package.

Support Vector Machines

SVMs are based on a concept, that we may define planes, in order to determine decision regions for classification. In order to adjust a “linear” (basic) SVM to a more demanding data separation tasks, we apply kernels. SVMs, in their elementary form, are binary classifiers. In order to provide multi-class classification capabilities, we form a collection of binary classifiers. Many popular implementations, including the one used in our study, apply a “one-against-one” approach. If c is the number of classes, then $c \cdot (c - 1)/2$ classifiers are constructed, and each one is for a pair of classes. Next, a voting scheme is applied to determine appropriate class label.

In our study, we used the function “svm” implemented in R package “e1071”. SVM parameter tuning has been performed with “tune” function from the same package. In all experiments, we trained SVMs based on Radial Basis Function (RBF) kernel. We have selected the RBF kernel, because our data is not linearly separable in the original feature space. In such a case, it is a common approach to apply the RBF kernel. Two parameters have to be considered and tuned: cost and γ . The cost parameter determines a trade-off between misclassification of training examples and the simplicity of a decision surface. γ determines the influence of a single training example. In all cases, 10-fold cross-validation has been applied.

Global Architecture

In global architectures instantiated in the experiment, rejection mechanisms were based on collections of $c = 10$ binary classifiers, one for each class of native patterns: either binary random forests or SVMs. Each classifier was trained to separate one class (playing the role of “pro” class) from all other classes of native patterns (simulating “contra” patterns). An unknown pattern was assumed to be foreign, if all $c = 10$ binary classifiers assigned it to the class simulating the “contra” patterns.

Local Architecture

In local architectures, either random forest or SVM was used for classification. Then, correspondingly, for each native class, a binary random forest or an SVM was employed to reject foreign patterns. A rejection mechanism is paired with each native

class. Rejecting classifiers were trained alike in the case of global architecture. However, unlike in the case of the global architecture, a pattern classified to a given native class is rejected if the corresponding rejecting mechanism rejects it, cf. Figure 1.

Embedded Architecture

In order to construct embedded architectures, a binary tree of classifiers was used, as discussed in Section 3.2.3. Let us recall, that in the embedded architecture, we start with the full set of patterns and we are sequentially performing binary splits. Splitting stops, when we end up with a subset of patterns assumed to belong to a single native class. Each split is accompanied with a rejection mechanism. In the experiments, we use SVMs and random forests as internal binary classifiers and we discuss a full embedded architecture, not a thinned one. It is worth to mention, that the local architecture could be seen as a special case of a thinned embedded architecture, in which rejection mechanisms are in leaves only. A special case of an SVM-based embedded architecture with rejection mechanisms based on one-class and two-class SVMs employed only in leaves was discussed in [22].

5 Results

5.1 Recognition with Rejection – the Perspective of Native Patterns Only

At first, let us investigate the quality of native patterns classification provided by the proposed architectures. We expect, that adding a rejection mechanism will influence the classification outcome. It would be ideal, if a rejection mechanism would be able to reject native patterns, which would have been incorrectly classified.

An overview of the results is presented in Table 1. We compare global, local, and embedded architectures based on random forests (RF) and SVMs. Table 1 contains results on training and test sets of native patterns. In addition, we present classification quality in a case, when we do not perform rejection. We see no substantial discrepancies in classification rates for a case of classification with rejection, in comparison to classification without rejection. This indicates, that adding a rejection mechanism does not hinder quality of classification.

Table 1 is divided into two parts, because we have two classifiers. Results reported in the left part of the table correspond to a ten-class classifier (based either on a random forest, or on an SVM) furnished with a rejecting mechanism available for the global and the local architectures. Results reported in the right part of the table come from a compound classifier based on binary classifiers arranged in a binary tree furnished with rejecting mechanisms, applicable in the embedded architecture.

All procedures have been conducted with a ten-fold cross-validation in order to avoid overfitting. Still, results on the training set are better than on the test set. Nonetheless, classification rates on the test set are satisfactory. It is worth to draw attention to Strict Accuracy and to Fine Accuracy. Strict Accuracy was lower for random forests than for SVMs. However, Fine Accuracy was at a comparable level for both methods. On the test set, for recognition without rejection, Strict Accuracy and Fine Accuracy (recall that Strict Accuracy, Strict Native Sensitivity and Fine Accuracy are equal in this case) were at the level of 95-96%. Adding rejection mechanisms worsened Strict Accuracy and improved Fine Accuracy. Strict Accuracy got worse by 1-6%. In contrast, Fine Accuracy was raised to the level of 98% for both classification methods and all three architectures and the increase was the most remarkable for the embedded architecture. Results show that the local architecture causes the smallest deterioration in quality of native patterns classification. On the other hand, results indicate, that the embedded architecture has the greatest capability to improve classification rates of a malfunctioning classifier.

5.2 Recognition with Rejection for Native and Semi-Synthetic Foreign Patterns

Let us compare the quality of rejection provided by the proposed architectures. Figure 6 illustrates quality measures for different models applied to various semi-synthetic foreign patterns.

Results indicate, that the prime aspect affecting the outcome is the character of a data set. The most difficult to reject were rotated digits, what is coherent with intuition, as the other distortions disrupted shapes in images to a greater extent. It is worth to emphasise, that we did not expect a perfect rejection rate for semi-synthetic patterns, because all dis-

Table 1. Comparison of classification results with rejection (global, local, and embedded architectures) on training and test sets of native patterns with classification results without rejection. RF – results for random forests, SVM – results for Support Vector Machines. Notice, that in the case of absence of foreign patterns

Strict Accuracy is equal to Strict Native Sensitivity, Accuracy is equal to Native Sensitivity and Native Precision is equal to 1 and, for classification without rejection, Strict Accuracy is equal to Fine Accuracy and it does not make sense to consider Accuracy (formally, it is equal to 1).

	$c = 10$ class classifier						A tree of binary classifiers			
Arch.	Global		Local		no rejection		Embedded		no rejection	
Base Class.	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM
Data Set	Native Patterns, Training Set									
Fine Acc.	100	99.8	100	99.8	100	99.7	100	99.8	100	99.2
Strict Acc.	100	99.5	100	99.5	100	99.7	100	99.1	100	99.2
Accuracy	100	99.7	100	99.7	–	–	100	99.3	–	–
Data Set	Native Patterns, Test Set									
Fine Acc.	97.7	97.8	97.9	98.1	95.2	96.5	98.2	98.4	88.2	94.0
Strict Acc.	88.3	94.6	88.2	94.5	95.2	96.5	87.8	93.5	88.2	94.0
Accuracy	90.3	96.7	90.1	96.3	–	–	89.4	95.0	–	–

tortions, which we introduced, were moderate. For instance, handwritten digit 0 rotated by any degree looks very similar to a “regular” 0, a crossed out handwritten digit 8 looks very similar to a “regular” 8, and so on. This intuition is confirmed by the reported values of Native Sensitivity and Foreign Precision, which are close to 1. Indeed, the smaller the number of false negatives, the higher the values of these measures. On the other hand, moderate or even small values of Native Precision and Foreign Sensitivity are caused by a higher number of false positives. The reason for that is that many foreign patterns were incorrectly accounted as native.

When we compare results obtained using the three architectures: global, local, and embedded, we see that the performance of the global model is the worst. In all cases, both the local and the embedded model provided better rejection rates. It is difficult to decide, which model was better: local or embedded. Experiments show, that they achieve comparable results. However, the embedded model is far more complex than the local model. On one hand, the embedded architecture could be subjected to custom tuning to an extent greater than the local architecture. However, the local architecture provides us with comparable results and is fairly simple to construct.

In general, models based on random forests performed on tested semi-synthetic patterns better than

architectures constructed with SVMs, except Native Sensitivity and Foreign Precision, where SVM-based models were slightly better.

5.3 Recognition with Rejection when Letters Were Used as Contaminations

Figure 7 presents the quality of classification and rejection models employed for data sets, in which handwritten digits were native patterns, while handwritten Latin letters and handwritten Kannada characters were foreign patterns.

In a similar way to the case of semi-synthetic foreign patterns, low values of Native Precision and Foreign Sensitivity for the data set of handwritten Latin letters were caused by a high number of false positives. It means, that many letters were not rejected.

Kannada characters turned out to be significantly easier to reject than handwritten Latin letters. This result is coherent with common sense expectations. A brief visual inspection of the selected samples shown in Figure 5 reveals, that Kannada characters are more distinct from digits than Latin alphabet letters.

The trouble of distinguishing between handwritten Latin alphabet letters from handwritten digits is comparable to the level of difficulty when we were processing a data set contaminated with semi-

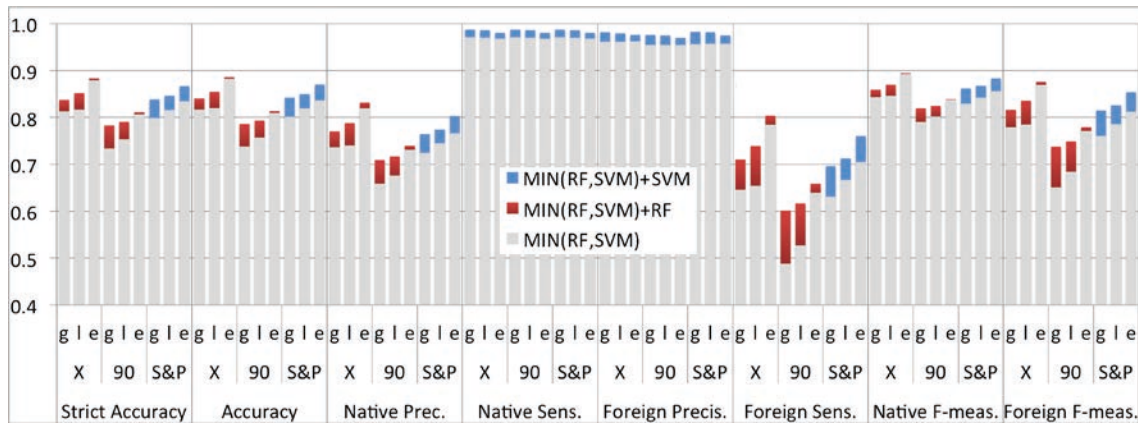


Figure 6. Quality measures of classification with rejection based on architectures instantiated with SVMs and with random forests (RF). Slate gray indicates the difference (advantage) of the SVM-based model over the corresponding random forest (RF) model. Black informs about the difference (advantage) of random forest-based model over SVM-based model. Plot presents results for all architectures: “g” stands for global, “l” – local, “e” – embedded. “X” stands for crossed out patterns, “90” are averaged results for patterns rotated anticlockwise and clockwise by 90° .

synthetic foreign patterns that originated from digits. The best performance has been achieved with the embedded rejection mechanism. The second best was the local rejection mechanism. The worst results have been achieved by the global rejection mechanism. All in all, trained mechanisms reject foreign patterns with a relatively high rate. Foreign F-measure is higher than Native F-measure, what suggests that the trained models maintain a higher capability to reject possibly distorted pattern than to accept it as native.

In almost all cases illustrated in Figure 7, models based on random forests were outperforming models based on SVMs. The advantage of random forests-based models is smaller for the Latin alphabet letters (the difficult characters in this comparison).

5.4 Further Investigation on the Global Rejection Mechanism

Since the global rejection architecture turned out to be less successful, in comparison with the local and the embedded architecture, let us now investigate a scheme for improving it.

Let us reiterate, that in the global architecture, first, we perform the action of foreign patterns rejection. Next, we classify patterns that were not rejected. In Section 3.3, we proposed to train a rejection mechanism based on binary classifiers suited

to work with a multi-class problem. A collection of c binary classifiers, one for each native class, together form a rejection mechanism. Binary classifiers are trained in a way “one-versus-all-other” classes. Thus, we can distinguish a “pro” class, which is made of patterns belonging to a single class, and a “contra” class, which is made of patterns belonging to all other classes. When a new pattern is passed into the input of a rejection mechanism made of c binary classifiers trained in this way, we ask all c binary classifiers, if this pattern belongs to one of the c classes. Only when all classifiers assume, that it belongs to the “contra” class, we reject this pattern.

We can modify the global rejection mechanism and apply clustering to construct it in a more data-aware fashion. We can easily imagine, that in a multi-class classification problem native patterns with the same class label may have very different characteristics. In order to construct a rejection mechanism, we can forgo assigned native class labels and perform unsupervised partitioning of native patterns into k clusters. Subsequently, we can assume, that cluster belongingness determines new class membership and re-label the set of native patterns using cluster labels. The number of clusters does not have to be equal to the number of native classes. Next, we can proceed with formation of a rejection mechanism based on a collection of binary classifiers using labels obtained from cluster-

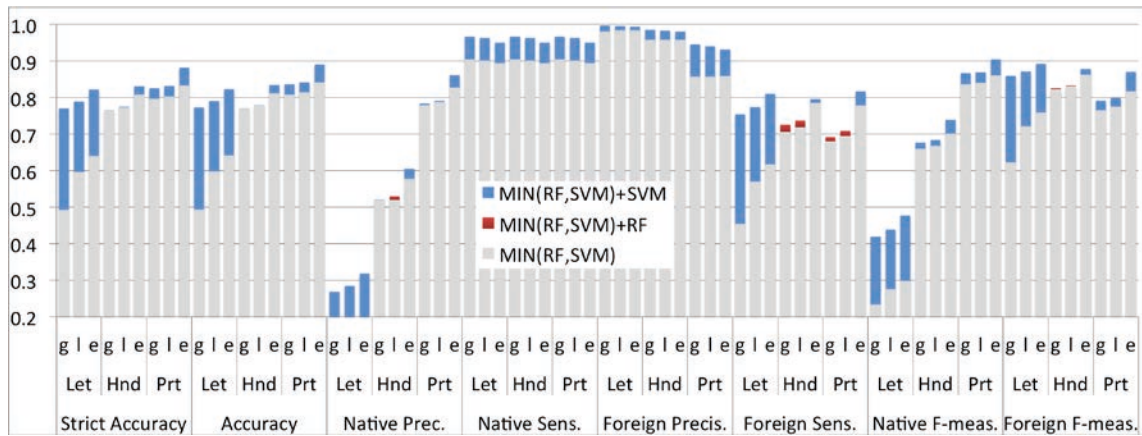


Figure 7. Rejection quality for data sets made of native handwritten digits, foreign handwritten Latin letters and handwritten Kannada symbols. “Let” stands for Latin alphabet letters, “Hnd” stands for handwritten Kannada characters. Quality measures of classification with rejection based on architectures instantiated with SVMs and with random forests (RF). Slate grey indicates the advantage of SVM-based model over corresponding random forest (RF) model. Black informs about the advantage of random forest-based model over SVM-based model.

ing. The advantage of this approach is that unsupervised clustering allows construction of a truly data-driven model. The main premise is that we should not assume that patterns with the same class label are coherent. Therefore, we may disregard true class labels and we may take an advantage of data partitioning discovered with a clustering algorithm of choice.

Let us now present results of experiments with the improved global architecture for selected data sets. We implemented the described procedure with the following settings. We have clustered native data of handwritten digits into $k = 3, 4, \dots, 20$ clusters using k-means clustering. Next, we formed k binary classifiers. We trained them to recognize class “pro” made of patterns belonging to a single cluster versus class “contra” of patterns belonging to all other clusters.

This rejection mechanism has been applied to reject foreign patterns from a native data set of handwritten digits contaminated with: handwritten Latin letters and handwritten Kannada characters. Figure 8 presents selected results.

Increasing the number of clusters on one hand lets us form more fine-grained model, fitted better to the data. On the other hand, we risk overfitting. Experiments show, that as we increase the number of clusters, increased are chances for:

- incorrectly rejecting native patterns,

- rejecting foreign patterns.

The first case is measured by Native Sensitivity, which is equal to Accuracy computed on native patterns only. The second case is reflected with Foreign Sensitivity. Indeed, Native Sensitivity slightly drops, and Foreign Sensitivity increases as k grows. All in all, the increase in Foreign Sensitivity is more substantial than the decrease in Accuracy. Looking at Figure 8, we see that the value of k between 10 and 13 is sensible. We expected so, because the data set of handwritten digits is not imbalanced. It is fairly regular, so the number of discovered groups should coincide with assigned labels. Application of the procedure investigated in this Section is especially recommended for data, where patterns belonging to the same class are irregular (as it is for instance in the case of music notation). Doubtless, the choice of the appropriate number of clusters should be based on experiments and should be performed individually for a given native data set.

5.5 Comparison with Existing Methods

In this Section, let us present and compare selected results obtained with one of our rejection architectures and approaches discussed in the literature. We apply one-class-SVMs and centroid-based methods to reject foreign patterns. Both one-class-SVM and centroid-based methods are computed in two variants. The first variant is a typical nov-

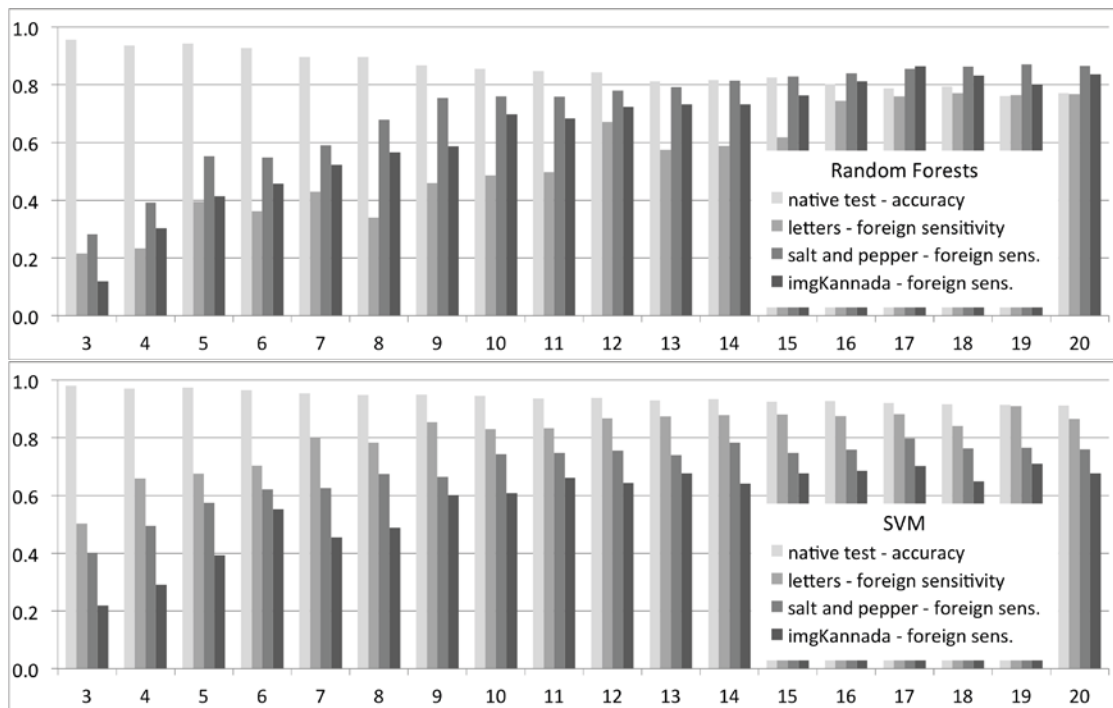


Figure 8. Accuracy on native test set, Foreign Sensitivity for handwritten Latin letters and handwritten Kannada characters. Results were achieved with an improved global rejection mechanism. Top plot: models based on random forest, bottom plot – SVMs. Horizontal axis labels inform about k – the number of clusters making the model.

elty detection scenario, when all native patterns are treated as one class, disregarding their true native class labels. A pattern is rejected, if this single rejection mechanism rejects it. The second scenario is fine-grained. We build ten models, one for each native class ($c = 10$ is the number of native classes for handwritten digits). A pattern is rejected, if all ten one-class models reject it.

The comparison concerns the local rejection architecture, which we believe is the most versatile model addressed in this paper. It is both relatively easy to construct and achieves satisfying recognition and rejection rates in frames of other architectures discussed in this study.

One-class-SVM has been trained on native training set of handwritten digits using 10-fold cross-validation. We applied an implementation available in the “e1071” R package. We selected Radial Basis Function kernel. Parameters γ and ν were individually tuned for each model.

In the centroid-based approaches, we have obtained centroids by calculating arithmetic mean of all training patterns (the case of the one-centroid)

or means in ten classes (the ten-centroid variant). Next, we have formed a region, the smallest sphere with the centroid being its centre, that enclosed all native patterns of the training set in it. In the one-centroid method, it was one region. In the ten-centroids method, there were ten regions, one per each native class.

Quality measures (in %) for one-class-SVM and for centroid-based methods are presented in Table 2. For comparison, the first column contains results for the local rejection mechanism instantiated with SVMs.

It is interesting to compare quality measures for the literature-based approaches. One-class-SVM performs considerably better than centroid-based methods. The only exception is for Foreign Precision, which is better for centroids. This effect is explained by an almost perfect identification of foreign patterns achieved by centroids. SVM-based methods incorrectly accept as native many more foreign patterns. However, the cost of this property is reflected with very low Native Precision. Centroid-based methods incorrectly reject a huge number of native patterns.

Table 2. Quality measures for semi-synthetic and handwritten foreign symbols. We compare SVM-based local architecture with literature-based approaches: one-class-SVMs and centroids. Reported are values for joined training and test sets.

		foreign set	local arch.	1-class SVM(s)		centroid(s)	
				one	ten	one	ten
Sensitivity	Native		98.7	46.2	52.7	100	99.8
	Foreign	crossed out	77.4	100	99.5	2.6	10.0
		rotated ± 90	68.3	99.1	98.1	0.5	8.5
		Latin	77.4	100	99.9	24.3	59.6
Kannada		71.9	100	98.6	0.8	9.8	
Precision	Native	crossed out	71.5	100	99.1	50.7	52.6
		rotated ± 90	54.6	98.1	96.5	50.1	52.2
		Latin	81.4	100	99.7	56.9	71.2
		Kannada	77.8	100	97.4	50.2	52.5
	Foreign	crossed out	97.3	65.0	67.8	98.5	97.9
		rotated ± 90	96.3	64.8	67.5	91.9	97.6
		Latin	98.3	65.0	67.8	99.8	99.7
		Kannada	98.2	65.0	67.6	95.1	97.9
Accuracy	Accuracy	crossed out	84.7	73.1	76.1	51.3	54.9
		rotated ± 90	76.3	72.7	75.4	50.2	54.2
		Latin	88.0	73.1	76.3	62.1	79.7
		Kannada	85.4	73.1	75.6	50.4	54.8
	Strict	crossed out	84.3		71.9		9.5
		rotated ± 90	75.9		71.2		8.8
		Latin	87.7		72.1		34.4
		Kannada	84.9		71.4		9.5
	Fine		99.2		84.0		9.1

The two literature-based approaches (a) turned out to be better than the local rejection architecture for some quality measures and (b) substantially worse for some measures. Namely, only Foreign Sensitivity and Native Precision are significantly better for the SVM-based literature methods than for the local architecture. This effect is yielded by a very good rejection of foreign patterns. This makes the TN parameter very high and FP parameter very low. In all other cases, the local architecture proves its superiority to the literature methods.

It is worth to notice, that literature-based approaches provide very low Accuracy measures (all three of them). In other words, they reject far too many native patterns. It is worth to draw attention to Fine Accuracy achieved by the collection of ten one-class-SVMs. It is relatively high, which indicates, that this mechanism was able to correctly classify a vast majority of not rejected native patterns. Severely low values of Strict Accuracy and Fine Accuracy for the ten-centroids method results from overlapping regions of different classes. Because of this, many native patterns were accounted into two or more regions. In consequence, they were not classified to any class.

6 Conclusions

In the paper, we discussed three architectures capable to perform native patterns classification with foreign patterns rejection. Studied models: global, local, and embedded differ, first and foremost, in their complexity. Proposed approaches have been tested in a series of experiments focused on handwritten digits recognition. We implemented rejecting/classifying architectures using random forests and SVMs and compared the outcomes produced by different approaches. We compared our approach with two standard novelty detection techniques present in the literature.

Let us conclude by stating that the proposed mechanisms perform very well. They are, all in all, better than popular methods available in the literature. The literature-based approaches tend to reject a lot of native patterns. Proposed architectures maintain much more reasonable balance between native patterns acceptance and foreign patterns rejection. Thus, we believe that the methods studied

in this paper are a valuable contribution to the area of pattern recognition.

The study shows, that the best performance could be achieved when using the embedded model. However, this comes at a cost of a relatively high model complexity. The embedded model requires the highest computational and design effort. The local architecture provides slightly worse performance, but it is very easy to construct. In the future, we plan to extend the study on rejection techniques onto on-line learning strategies.

Acknowledgment

The research is supported by the National Science Centre, grant No 2012/07/B/ST6/01501, decision no DEC-2012/07/B/ST6/01501 and by the National Natural Science Foundation of China (No. 11971065)

References

- [1] W. Homenda and A. Jastrzebska, Global, local and embedded architectures for multiclass classification with foreign elements rejection: an overview, Proc. of the 7th International Conference of Soft Computing and Pattern Recognition, pp. 89–94, 2015.
- [2] F. J. Anscombe, Rejection of outliers, *Technometrics*, vol. 2, no. 2, pp. 123–147, 1960.
- [3] V. Barnett and T. Lewis, *Outliers in Statistical Data*, 3rd ed. Wiley, 1994.
- [4] M. P. Maples, D. E. Reichart, N. C. Konz, T. A. Berger, A. S. Trotter, J. R. Martin, D. A. Dutton, M. L. Paggen, R. E. Joyner, and C. P. Salemi, Robust Chauvenet outlier rejection, *The Astrophysical Journal Supplement Series*, vol. 238, no. 1, p. 2, 2018.
- [5] Z. Li, R. J. Baseman, Y. Zhu, F. A. Tipu, N. Slonim, and L. Shpigelman, A unified framework for outlier detection in trace data analysis, *IEEE Transactions on Semiconductor Manufacturing*, vol. 27, no. 1, pp. 95–103, 2014.
- [6] G. Yuksel and M. Cetin, Outlier detection in a preliminary test estimator of the mean, *Journal of Statistics and Management Systems*, vol. 19, no. 4, pp. 605–615, 2016.
- [7] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, A review of novelty detection, *Signal Processing*, vol. 99, pp. 215–249, 2014.
- [8] R. Rocci, S. A. Gattone, and R. Di Mari, A data driven equivariant approach to constrained gaussian

- mixture modeling, *Advances in Data Analysis and Classification*, vol. 12, no. 2, pp. 235–260, 2018.
- [9] A. Punzo, A. Mazza, and A. Maruotti, Fitting insurance and economic data with outliers: a flexible approach based on finite mixtures of contaminated gamma distributions, *Journal of Applied Statistics*, vol. 45, no. 14, pp. 2563–2584, 2018.
- [10] L. Xiang, K. K. Yau, and A. H. Lee, The robust estimation method for a finite mixture of poisson mixed-effect models, *Computational Statistics & Data Analysis*, vol. 56, no. 6, pp. 1994–2005, 2012.
- [11] H. Otneim and D. Tjøstheim, The locally gaussian density estimator for multivariate data, *Statistics and Computing*, vol. 27, no. 6, pp. 1595–1616, 2017.
- [12] J. Zhang and H. Wang, Detecting outlying subspaces for high-dimensional data: the new task, and performance, *Knowledge and Information Systems*, vol. 3, no. 10, pp. 333–355, 2006.
- [13] V. Hautamaki, I. Karkkainen, and P. Franti, Outlier detection using k-nearest neighbour graph, *Proc. of the 17th International Conference on Pattern Recognition*, vol. 3, pp. 430–433, 2004.
- [14] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, Lof: identifying density-based local outliers, *Proc. of the ACM SIGMOD International Conference on Management of Data*, vol. 29, pp. 93–104, 2000.
- [15] H. Izakian and W. Pedrycz, Anomaly detection in time series data using a fuzzy c-means clustering, *Proc. of IFSA World Congress and NAFIPS Annual Meeting*, pp. 1513–1518, 2013.
- [16] F. de Morsier, D. Tuia, M. Borgeaud, V. Gass, and J.-P. Thiran, Cluster validity measure and merging system for hierarchical clustering considering outliers, *Pattern Recognition*, vol. 48, no. 4, pp. 1478–1489, 2015.
- [17] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, New support vector algorithms, *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.
- [18] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, Estimating the support of a high-dimensional distribution, *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [19] C. Gautam, R. Balaji, S. K., A. Tiwari, and K. Ahuja, Localized multiple kernel learning for anomaly detection: One-class classification, *Knowledge-Based Systems*, vol. 165, pp. 241–252, 2019.
- [20] C. Desir, S. Bernard, C. Petitjean, and L. Heutte, One class random forests, *Pattern Recognition*, vol. 46, no. 12, pp. 3490–3506, 2013.
- [21] D. M. J. Tax and R. P. W. Duin, Combining one-class classifiers, *Proc. of Multiple Classifier Systems: Second International Workshop*, pp. 299–308, 2001.
- [22] W. Homenda, A. Jastrzebska, and W. Pedrycz, Rejecting foreign elements in pattern recognition problem. reinforced training of rejection level, *Proc. of the 7th International Conference on Agents and Artificial Intelligence*, pp. 90–99, 2015.
- [23] Y. Shiraishia and K. Fukumizu, Statistical approaches to combining binary classifiers for multi-class classification, *Neurocomputing*, vol. 74, pp. 680–688, 2011.
- [24] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes, *Pattern Recognition*, vol. 8, no. 44, pp. 1761–1776, 2011.
- [25] Y. LeCun, C. Cortes, and C. J. Burges, The mnist database of handwritten digits, <http://yann.lecun.com/exdb/mnist>.
- [26] T. E. de Campos, B. R. Babu, and M. Varma, Character recognition in natural images, in *Proc. of the International Conference on Computer Vision Theory and Applications*, 2009. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/character-recognition-in-natural-images/>
- [27] L. Breiman, Random forests, *Machine Learning*, vol. 1, no. 45, pp. 5–32, 2001.



Władysław Homenda received the M.Sc. in applied mathematics and Ph.D. in computer science degrees from Warsaw University of Technology, Warsaw, Poland, and the D.Sc. degree and Professor title in computer science from the Systems Research Institute of Polish Academy of Sciences, Poland. He has been with Warsaw Uni-

versity of Technology since graduation. He is currently Professor with the Faculty of Mathematics and Information Science, Warsaw University of Technology. He is also with the Faculty of Economics and Informatics in Vilnius (Lithuania) of the University of Białystok. He undertook organisation of graduate studies in CICESE, Ensenada, Mexico and in Faculty of Mathematics and Information Science, Warsaw, Poland.

He is the author of four books, more than 130 research articles and music processing technologies. His main research interests are in theoretical foundations of computer science and intelligent computing technologies, specifically in the areas of man-machine communication and human-centric computing. He is also active in such areas as fuzzy modeling and granular computing, knowledge discovery and data mining. He currently serves as an Associate Editor of Information Sciences and is a member of several editorial boards of other international journals.



Agnieszka Jastrzębska received the B.Sc. degree in information technology from the University of Derby, Derby, UK in 2009 and the M.Sc. Eng. degree in computer engineering from the Rzeszow University of Technology, Rzeszow, Poland in 2010 and M.A. in economics from the University of Rzeszow, Rzeszow, Poland in 2011.

She received the Ph.D. degree from the Warsaw University of Technology, Warsaw, Poland in 2016.

From 2011 to 2017, she was a Research and Teaching Assistant with the Faculty of Mathematics and Information Science, Warsaw University of Technology. Since 2017, she has been an Assistant Professor at the Faculty of Mathematics and Information Science, Warsaw University of Technology. Her research interests include machine learning, computational intelligence, and fuzzy modelling.



Witold Pedrycz (IEEE Fellow, 1998) is Professor and Canada Research Chair (CRC) in Computational Intelligence in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. He is also with the Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland. In 2009 Dr.

Pedrycz was elected a foreign member of the Polish Academy of Sciences. In 2012 he was elected a Fellow of the Royal Society of Canada. In 2007 he received a prestigious Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Society. He is a recipient of the IEEE Canada Computer Engineering Medal, a Cajastur Prize for Soft Computing from the European Centre for Soft Computing, a Killam Prize, a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society, and 2019 Meritorious Service Award from the IEEE Systems Man and Cybernetics Society.

His main research directions involve computational intelligence, fuzzy modeling and granular computing, knowledge discovery and data science, pattern recognition, data science, knowledge-based neural networks, and control engineering. He has published numerous papers in these areas; the current h-index is 111 (Google Scholar) and 82 on the list top-h scientists for computer science and electronics <http://www.guide2research.com/scientists/>. He is also an author of 21 research monographs and edited volumes covering various aspects of Computational Intelligence, Data Mining, and Software Engineering.

Dr. Pedrycz is vigorously involved in editorial activities. He is an Editor-in-Chief of Information Sciences, Editor-in-Chief of WIREs Data Mining and Knowledge Discovery (Wiley), and Co-editor-in-Chief of Int. J. of Granular Computing (Springer) and J. of Data Information and Management (Springer). He serves on an Advisory Board of IEEE Transactions on Fuzzy Systems and is a member of a number of editorial boards of international journals.



Dr. Fusheng Yu is a professor in the School of Mathematics Sciences, Beijing Normal University, Beijing, China. He received the M.S. degree and Ph.D. degree in Applied Mathematics from Beijing Normal University. From 2002 to 2004, he was a visiting scholar with the Department of Electrical and Computer Engineering, University of

Alberta, Edmonton, Alberta, Canada.

He is pursuing the research in computational intelligence, granular computing, fuzzy systems and fuzzy modeling, knowledge discovery and data mining, knowledge representation and fault diagnosis expert systems. He has presided over or participated in more than 10 projects. He has published more than 100 research papers. He (once) served as a member of the editorial committees of some journals, a member of the procedure committees or a chairman of the regional organization committee of several international conferences. He is vice chairman of the professional committee of fuzzy information and engineering branch of China operations research association, member of the special committee of non-classical logic and calculation of China logic society, and member of the professional committee of risk analysis of China disaster prevention association.