Pattern Recognition with Rejection

Combining Standard Classification Methods with Geometrical Rejecting

Wladyslaw Homenda[1,2], Agnieszka Jastrzebska[1], Piotr Waszkiewicz[1] and Anna Zawadzka[1]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland
[2] Faculty of Economics and Informatics in Vilnius, University of Bialystok
Kalvariju g. 135, LT-08221 Vilnius, Lithuania

Abstract. The motivation of our study is to provide algorithmic approaches to distinguish proper patterns, from garbage and erroneous patterns in pattern recognition problem. The design assumption is to provide methods based on proper patterns only. In this way the framework that we propose is truly versatile and it can be adapted to any pattern recognition problem in an uncertain environment, where garbage patterns may appear. The proposed attempt to recognition with rejection combines known classifiers with geometric methods used for separating native patterns from foreign ones. The proposed methods are empirically verified on datasets of handwritten digits classification (native patterns) and handwritten letters of Latin alphabet (foreign patterns).

Key words: pattern recognition, classification, rejecting option, geometrical methods

1 Introduction

The task of pattern recognition is a classical machine learning problem. On the input we pass a training dataset, consisting of labelled patterns belonging to c classes. In the process we expect to form a model that would be able to assign correct labels to new patterns (new observations).

It is important to have in mind that patterns in their original form are some sort of signals, for instance images or voice recordings. Due to the fact that the original patterns are often collected using some signal-acquiring devices, we may encounter patterns that do not belong to any of proper classes. An example of such situation may happen, when the device that we use to acquire data has been automatically reset due to power outage and poor default calibration distorts the segmentation process. Another scenario may happen when we collect data in a noisy (out-of-lab) environment and apart from proper patterns there is a lot of unexpected residual elements. The problem with such patterns, say garbage patterns, is that we cannot predict their characteristic and therefore we cannot include information about them in the model training process.

The motivation of our study is to provide algorithmic approaches to distinguish proper patterns, that we call **native patterns** from garbage and erroneous patterns, that we call **foreign patterns**. The task that we describe we call **foreign patterns rejection**. The design assumption is to provide methods based on native patterns only. In this way the framework that we propose is truly versatile and it can be adapted to any pattern recognition problem in an uncertain environment, where foreign patterns may appear.

In particular, the objectives of this paper are focused on designing methods for recognition with rejection and employ them to (a) distinguish native patterns from foreign ones and (b) improve classification quality of native patterns. The proposed attempt to recognition with rejection combines known classifiers with geometric methods used for separating native patterns from foreign ones. The proposed methods are empirically verified on datasets of handwritten digits (native patterns) and handwritten Latin letters (foreign patterns).

We would like to emphasis that the novelty of the contribution presented in this paper is not in the methods that we use, but in how we employ them and on what we achieve with them. Our main purpose is to examine the cooperation of standard classifiers with geometric methods used for rejection, especially influence of rejection on native patterns classification, eg., improvement of Fine Accuracy. We also test the effectiveness of foreign patterns rejection and differences between used classifiers.

The remainder of this paper is organized as follows. Section 2.1 presents the background knowledge on foreign elements detection present in the literature. Sections 2.2 and 2.3 presents the backbone algorithms, known in the literature, that we use to construct our models. Section 3 presents the proposed approach. In Section 4 we discuss a series of experiments. Section 5 concludes the paper and highlights future research directions.

2 Preliminaries

Data collection and processing is a vital study problem across multiple domains of science. Along with a substantial automation of data acquiring we meet with difficulties that appear due to poor data quality. The research we present in this paper has been motivated by the issue of contaminated datasets, that apart from proper patterns contain garbage.

In this section we start discussion with review of relevant literature positions in machine learning that tackle the issue of contaminated datasets. Then, in order to provide a self-contained description of employed methods we present backbone literature algorithms applied. In what follows we present the Minimum Volume Enclosing Ellipsoid (MVEE) algorithm and a suite of 3 classification methods: Random Forests (RF), Support Vector Machines (SVM), and K-Nearest Neighbors algorithm (K-NN). Listed methods are employed in various configurations to native elements classification with foreign elements rejection. Our approach, based on those algorithms, is discussed in Section 3.

2.1 Literature Review

The rejecting option in pattern recognition problem has gained rather weak attention despite its importance in practice. Also, there is a relatively short list of papers raising the problem of rejecting foreign patterns, cf. [8] for a short survey.Here we only hint some issues present in literature neither with claiming to comprehensive coverage the subject, not giving deep background of methods employed in this study.

Discussion on approaches related to foreign elements rejection should also mention one-class classification methods. Especially, there are two noteworthy examples: centroid-based methods and One-Class Support Vector Machine.

Centroid-based methods rely on distinguishing cluster centres (centroids). Region reserved for proper elements is usually defined by the distance between centre and the furthest proper pattern.

One-Class SVM has been introduced in [12]. While "regular" SVM algorithm forms hyperplane separating two classes, the One-Class SVM separates data points from the entire feature space. Notably, the One-Class SVM provides a soft decision rules, as there is a ν parameter determining the fraction for outliers.

When it comes to the study on foreign elements rejection, there is relatively few papers to review. This issue, in spite of its importance, remains somehow neglected. Among noteworthy studies one may mention rank-based methods, for instance ones described in [2, 4, 6, 7, 11, 13, 15]. In a nutshell, mentioned papers propose to attach confidence scores along with class labels. Rejection occurs when none of native class labels was assigned with a satisfying confidence.

In our paper we assume an approach that in its background resembles more model-based outlier detection techniques than reported in the literature foreign elements rejection schemes. The objective of our study is to form a model that would be able to distinguish between the region reserved for native patterns and the remainder of the feature space.

2.2 Ellipsoids for Foreign Elements Rejection

Both native and foreign patterns are represented by certain vector of features extracted from pattern of interest. Features are usually represented as real numbers, therefore every pattern is a point in multidimensional Euclidean space. What's more, if it turned out that the sets of those points representing symbols, each forms a tight cluster in space, it might be possible to find their respective minimal enclosing boxes.

In computational geometry, the smallest enclosing box problem is that of finding the oriented minimum bounding box enclosing a set of points. As opposed to convex hull, which is the most accurate point set container with smallest volume and which is enclosed by linear hyper planes. Bounding boxes are far less complex. In many cases, when there's a need for computing convex hull and testing inclusions of other points, an approximation of such hull can be used, which helps to reduce time needed for computations, since most of alternative methods have lower construction and inclusion-testing complexities. Some of such approaches include using figures like hypercubes, diamonds, spheres or ellipsoids to successfully enclose given set of points.

When comparing highlights and drawbacks of each method, from computational complexity perspective, ease of testing point inclusion and algorithm implementation point of view, ellipsoids seem to be a reasonable choice. Constructed ellipsoid is superior to the minimal cuboid in many ways. It is unique, gives better approximation of the object it contains and if $E(S)$ is the bounding ellipsoid for a point set S with convex hull $C(S)$ in dimension d, then:

$$\frac{1}{d}E(S) \subseteq C(S) \subseteq E(S)$$

where scaling is with respect to the center of $E(S)$.

Adaptation of the smallest enclosing box problem to foreign elements rejection, or native elements identification, seems to be a very natural approach. The justification is fairly simple: if we enclose elements belonging to native classes, using for instance ellipsoids, formed geometrical model will discriminate a region of the features space reserved for native patterns between a region where we may encounter foreign patterns. With this premise in mind, let us present a detailed description of the MVEE algorithm.

MVEE problem is solved by several known algorithms that can be categorized as first-order, second-order interior-point or combination of the two. For small dimensions d, the MVEE problem can be solved in $O(d^{O(d)}m)$ operations using randomized or deterministic algorithms [14]. In this paper the algorithm based on Khachiyan solution is used.
An ellipsoid in center form is given by

$$E = \{x \in \mathbb{R}^n | (x-c)^T A(x-c) \leq 1\}$$

where $c \in \mathbb{R}^n$ is the center of the ellipse E and $A \in \mathbb{S}^n_{++}$. Points lying inside the ellipse satisfy

$$(x_i - c)^T A(x_i - c) \leq 1 + \mathit{acceptance}$$

where acceptance parameter defines the error margin in determining point belonging to ellipsoid, i.e. it alows to enlarge the ellipsoid.

However, presented equation, is not a convex optimization problem and it has to be changed. It turns out that even then, the solution is not easily obtainable so the dual problem has to be found. For more precise and in depth solution description see [14]. The main problem, when using ellipsoids as identifiers, lies in constructing them. Two main factors that decide about identification effectiveness are tolerance and acceptance parameters. Tolerance can be viewed as a threshold for ellipsoid construction accuracy. The lower the parameter is, the better minimal volume ellipsoid is created. On the other hand, even with good training set, there's a risk of including native elements that lie outside of created ellipsoid. To prevent such unwanted behaviour acceptance has been introduced. It defines threshold for point rejection for points lying outside of created figure.

2.3 Native Elements Classification

The task of native elements classification relies on forming a model based on a labelled training dataset that assigns proper class labels to new problem instances. In a conventional scenario the training dataset consists of m instances of n-dimensional feature vectors. Feature vectors are numerical descriptors of actual patterns. There is a multitude of classification algorithms, among which we have selected 3 different ones that are applied in our methods. It is necessary to emphasize that if someone would like to adapt our method to their own domain, those algorithms could be substituted with some other classification tools that may be more efficient in that domain. Without further ado let us move towards a brief description of the methods that we apply in our study.

Support Vector Machines are a set of supervised learning methods used for classification, regression and outliers detection. The SVM algorithm relies on a construction of hyperplane with a maximal margin that separates instances of two classes, [5]. SVMs are effective in high dimensional spaces, memory efficient and quite versatile with many kernel functions that can be specified for the decision function. Although in some cases, where number of features is much greater than the number of samples, this method can give poor results, and is not cost-efficient when calculating probability estimates, it is well suited for problem presented in this paper. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the another. For multi-class classification "one-against-one" approach is used. For n classes n * (n - 1) / 2 classifiers are constructed, each one is trained with data from two different classes. In our study we use decimal digits as classes. Therefore, the following 45 class-against-class SVMs are built: "0 vs 1", "0 vs 2", . . . "0 vs *9*", "1 vs 2", . . . "1 vs 9", . . . "*8* vs *9*". Classification decision is taken by voting method, i.e. a new pattern subjected to classification is counted to the most frequent class among these 45 binary classifiers. The case when two or more classes are most frequent, a second choice decision is made for actual classification. For instance, the closest pattern from most popular classes or minimal sum of distances from the processed pattern to ones from most popular classes may decide.

Random Forests is a popular ensemble method. The main principle behind ensemble methods, in general, is that a group of "weak learners" can come together to form a "strong learner". In the Random Forests algorithm the weak learners are decision trees, which are used to predict the membership of objects in the classes. For vector of independent variables representing one object they calculate the value of the class the object belongs to by dividing value space into two or more subspaces. More precisely, an input data is entered at the top of the tree and as it traverses down the tree the data gets bucketed into smaller and smaller sets. In this method a large number of classification trees is generated. To grow each tree a random selection with replacement is made from the examples in the training set D. Those subsets D_k are called bootstrap training sets. At each node m variables are selected at random out of the set of input variables and the best split on these m is used to split the node. After a relatively large number of trees is generated, they vote for the most popular class. Random Forests join few important benefits: (a) they are relatively prone to the influence of outliers, (b) they have an embedded ability of feature selection, (c) they are prone to missing values, and (d) they are prone to overfitting, [3].

K-Nearest Neighbors is an example of a ”lazy classifier”, where the entire training dataset is a model. There is no typical model building phase, hence the name. Class membership is determined based on class labels encountered in K closest observations in the training dataset, [1]. In a typical application, the only choices the model designer has to make are selection of K and distance metrics. Both are often extracted with a help of a supervised learning procedures.

3 Methodology

There are two approaches used to determine whether an object is rejected (classified as foreign). First one assumes the use of classification methods, which originally were not designed for elements rejection. The second approach involves using classifiers as the only classification tool, whereas rejecting is realized by ellipsoids. In this section let us present the second approach to foreign elements rejection.

3.1 External Global and Local Rejecting

Whereas ellipsoids are good identifiers, they lack in element classification quality. This is caused by the fact that ellipsoids may overlap each other, which results in points belonging to both classes at the same time. Although this can be solved by calculating distance between those elements and each ellipsoid centre, or taking value of ellipsoid-inclusion equation as a classification measure, tests have proven that such approaches are more prone to errors than other classifiers mentioned in this paper. Taking into considerations both strengths and weaknesses of classifiers and identifiers, the combined solution has been prepared that employs both tools (classifiers and ellipsoids), making use of their advantages.

Classifiers have high success rate but cannot distinguish between foreign elements and native ones. Contrary to that, ellipsoids tend to be better at rejecting foreign patterns, which makes them good at identifying points that should not be classified. The natural way of dealing with that problem would be to use ellipsoids as first-entry identifier that purifies input set by removing foreig patterns. The result of such rejection would be sent to the chosen classifier that would classify remaining native elements. Schema of this method is presented on the left part of Figure 1

Another way of using ellipsoids as identifiers is to treat them as correctors. That means there’s no need to remove foreign elements from the set before classification, as rejection is done later. After all elements have been classified, identification for each class is done by using class-corresponding ellipsoid. There’s a possibility that some foreign elements will be assigned to a class with a corresponding ellipsoid that will reject them. This is somewhat different from previous approach because elements can be rejected even if there is an ellipsoid that would pass inclusion test. The schema can be seen on the right part of Figure 1

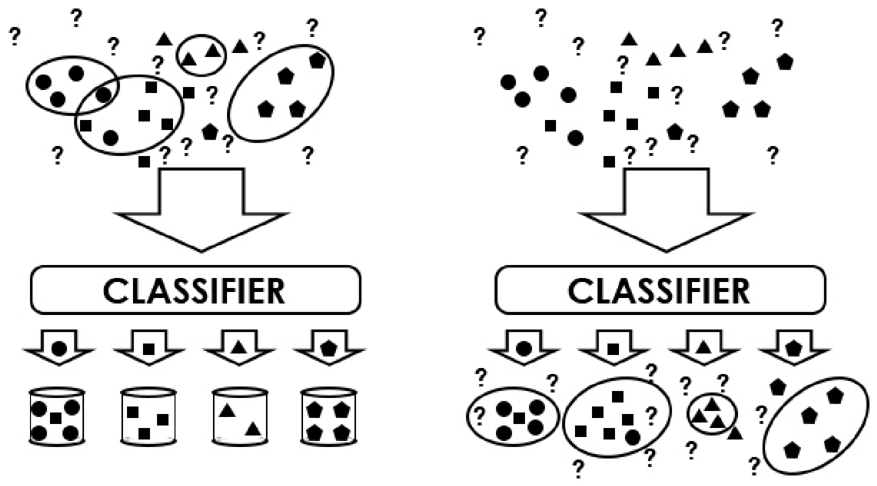

Fig. 1: Recognition with ellipsoids employed for rejecting: ellipsoids as identifiers reject foreign elements (left part) and ellipsoids as correctors (right part)

3.2 Quality Evaluation

In order to evaluate the quality of proposed methods the patterns from the following groups are counted:

- CC (Correctly Classified) - correctly classified patterns, i.e. native patterns classified as native ones with the correct class,
- TP (True Positives) - native patterns classified as native (no matter, into which native class),
- FN (False Negatives) - native patterns incorrectly classified as foreign,
- FP (False Positives) - foreign patterns incorrectly classified as native,
- TN (True Negatives) - foreign patterns correctly classified as foreign.

These numbers are then used to form measures reflecting specific aspects of classification and rejection, cf. Table 1. Notions that we use are well-known in the domain of pattern recognition. We have included a detailed description of those measures in our previous paper [8].

- *Strict Accuracy* measures classifier's performance. It is the ratio of the number of all *correctly classified* patterns to the number of all patterns being processed.
- *Accuracy* is a "softer" characteristic derived from the Strict Accuracy. Accuracy describes the ability to distinguish between native and foreign patterns. The difference is that we do not require that the native patterns are labelled with their proper class label.
- *Native Precision* is the ratio of the number of not rejected native patterns to the number of all not rejected patterns (i.e. all not rejected native and foreign

Table 1: Quality measures for classification and rejection

$$\text{Native Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad \text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FN+FP+TN}}$$

$$\text{Foreign Precision} = \frac{\text{TN}}{\text{TN+FN}} \qquad \text{Strict Accuracy} = \frac{\text{CC+TN}}{\text{TP+FN+FP+TN}}$$

$$\text{Native Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} \qquad \text{Fine Accuracy} = \frac{\text{CC}}{\text{TP}}$$

$$\text{Foreign Sensitivity} = \frac{\text{TN}}{\text{TN+FP}} \qquad \text{Strict Native Senssitivity} = \frac{\text{CC}}{\text{TP+FN}}$$

$$\text{F–measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

ones). The higher the value of this measure, the better ability to distinguish foreign elements from native ones. Native Precision does not evaluate how effective identification of native elements is.

- *Native Sensitivity* is the ratio of the number of not rejected native patterns to all native ones. The higher the value of Native Sensitivity, the more effective identification of native elements. Unlike the Native Precision, this measure does not evaluate the effectiveness of separation between native and foreign elements.
- *Strict Native Sensitivity* takes only correctly classified native patterns and does not consider native patterns, which are not rejected and assigned to incorrect classes, unlike Native Sensitivity, where all not rejected native patterns are taken into account.
- *Fine Accuracy* is the ratio of the number of native patterns classified to correct classes, i.e. assigned to their respective classes, to the number of all native patterns not rejected. This measure conveys how precise is correct classification of not rejected patterns.
- *Foreign Precision* corresponds to Native Precision.
- *Foreign Sensitivity* corresponds to Native Sensitivity.
- Precision and Sensitivity are complementary and there exists yet another characteristic that combines them: the *F–measure*. It is there to express the balance between precision and sensitivity since these two measures affect each other. Increasing sensitivity can cause a drop in precision since, along with correctly classified elements, there might be more incorrectly classified.

4 Experiments

In this section we move towards description of a series of experiments where we apply rejection strategies discussed theoretically in Sections 3.1. In what follows we describe the datasets, experiments' settings and the results.

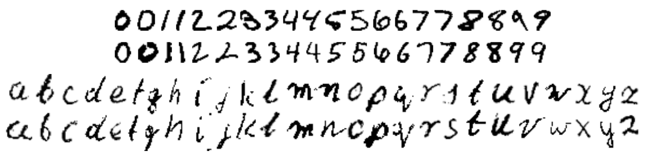

Fig. 2: Sample of: native patterns (top) and foreign patterns (bottom).

4.1 Datasets

The Experiments' section is devoted to a study on handwritten digits recognition and handwritten letters rejection.

Foreign patterns are usually unknown. In this study as foreign patterns we assume a dataset of handwritten letters from the Latin alphabet. The justification to assume such foreign dataset for testing purposes is that appearance of other real symbols, not belonging to any proper class, is a common issue in a character recognition problem.

We would like to stress again, that foreign patterns do not participate in model building phase. The entire scheme is based on native patterns only. Handwritten letters are used only for rejection mechanisms quality evaluation. Samples of processed patterns are displayed in Figure 2.

The training dataset was made of 10,000 handwritten digits with approximately 1,000 observations in each class taken from publicly available MNIST database, [9]. We split each class in proportion ca. 7:3 and in a process we got two sets. The first one includes 6,996 patterns and is used for training. The second set, the test set, contains 3,004 patterns. The dataset of foreign patterns contains 26,383 handwritten Latin letters, ca. 1,000 letters in each class. This dataset was created by 16 students, writing about 70 copies of each letter.

All patterns were normalized and feature vectors comprising of 106 numerical features were created. Examples of features are: maximum/position of maximum values of projections, histograms of projections, transitions, offsets; raw moments, central moments, Euler numbers etc. The best first search for the optimal feature subset has been performed using FSelector R package, [10] and then ANOVA was employed to select independent features. The final feature vector contained 24 features. We considered features standardization but the training data is sufficiently consistent (there is no outliers), so we normalized those features to bring linearly all values into the range [0,1].

4.2 Experimental Settings

Solutions presented in this paper has been implemented in Python programming language, using scientific libraries [16–18]. The MVEE algorithm, available as MATLAB code [3] has been rewritten in Python language, using NumPy library for matrix representation and operations. Several tests have been performed in order to find best suited method parameters for both classifiers and identifiers. For finding those values the Grid Search [18] has been used for SVM and Random Forests. Ellipsoids and KNN methods had their values assigned manually.

[3] http://www.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid/content/MinVolEllipse.m

MVEE method parameters Each ellipsoid was created and used with two parameters: tolerance and acceptance. The tolerance argument was used during creation phase, as "accuracy measurement". Lower value means that created enclosing figure is more fitted to the construction set. Acceptance parameter defines how far can point lie outside the ellipsoid to still be identified as belonging to it. In other words, it treats enclosing ellipsoid as being bigger than it really is. Parameters tests involved computing effectiveness of MVEE algorithm for certain tolerance and acceptance values. We tested values from such ranges:

- tolerance = [0.5, 0.2, 0.1, 0.01]
- accuracy = [0.1, 0.01, 0.001, 0.0005]

The results revealed that for given training and test sets the best parameter combination was tolerance=0.5 and accuracy=0.1 and those values were used in final, combined method described later in this document.

SVM method parameters SVM method available in Scikit package offers a few built-in kernels that were used during computations: radial basis function and polynomial. Additionally, there are two more parameters that were tested: C (described as penalty parameter C of the error term), and γ (known as kernel coefficient). Values that were tested:

- kernel = ['rbf', 'poly']
- C = [1, 2, 4, 5, 16]
- $\gamma = [2^{-1}, 2^{-2}, 2^{-3}, 0, 00025]$

The best found combination of those parameters used rbf kernel along with C=8 and $\gamma = 2^{-1}$ values.

Random Forests method parameters Scikit library was used to test random forests. Random forests with the following number of trees were tested: 1, 2, 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150. The best number of trees to build the forest is 100.

KNN method parameters The package was used for the k-NN classifier testing. We tested values of K (number of neighbors) such as: 1, 2, 3, 4, 5, 10, 20, 30, 40, 50. The best found value of k is 4. There is also one parameter - metric, but we use default value, which is standard Euclidean metric.

4.3 Results of Experiments

To determine effectiveness of parameter values for each method, confusion matrix has been calculated. Each row of this matrix represents information about distribution of patterns from certain class. For instance, 213 zeros were correctly classified to their class, 4 zeros were incorrectly classified as ones, 2 zeros - as sixes, 17 zeros as eights and 58 zeros were incorrectly rejected as non-digit patters, cf. row "0" in Table 2. Numbers in each column hold information about quantity of patterns that were classified as belonging to the corresponding class.

Table 2: Confusion table for SVM with global rejection on training set

	0	1	2	3	4	5	6	7	8	9	*f*
0	595						1		10		80
1		670	1				1				122
2			641					1		1	79
3			3	629		1		2			72
4					598					7	82
5				6		546			4		68
6	1						567		1		102
7		1		1	1			626		5	85
8	8				2			1	589	2	79
9		1		4	3	2		6	4	607	79
f	181	44	155	16	164	104	182	9	211	61	25256

Calculations and measurements included in "Quality Evaluation" section were based on confusion matrixes. Information gathered from them will be used in the following sections of this paper. The most interesting aspects of it consist of classification quality, rejection quality and rejection impact on classification quality. Those issues are the main and most important part of problem described in this paper.

Table 3: Results for classification with rejection on train and test sets of native patterns in comparison with classification results without rejection mechanism. RF - results for random forest, SVM - results for Support Vector Machines

Strict Accuracy	1	0.985	0.955	0.941	0.938	0.936	0.942	0.942	0.942
Fine Accuracy	1	0.985	0.955	1	0.987	0.972	1	0.989	0.984
Strict Native Sens.	1	0.985	0.955	0.879	0.852	0.854	0.864	0.857	0.845
Data Set	Native Patterns, Test Set								
Strict Accuracy	0.952	0.966	0.930	0.946	0.946	0.944	0.951	0.952	0.953
Fine Accuracy	0.952	0.966	0.930	0.972	0.982	0.959	0.977	0.985	0.976
Strict Native Sens.	0.952	0.966	0.930	0.842	0.852	0.831	0.837	0.845	0.825

Influence of Rejection on Native Patterns Classification Adding a rejection mechanism, ideally, may be seen as a method for improvement of classification rates. It would be perceived as a positive side of the rejection mechanism, if it would be able to reject those native patterns, which would be incorrectly classified when there is no rejection mechanism at all. Trained models only partially fulfil this wish. Conducted tests show that performing element rejection after their initial classification (local rejection scheme), brings better results. This could be explained by the fact that in the local rejection scheme we use one ellipsoid per each class and we apply those ellipsoids after classification. In contrast, in the global scheme we have a joint set of ellipsoids that we apply to the entire dataset. In the local rejection scenario native elements identification regions are applied individually to each subset obtained from the classifier. As a result there is a chance that classification would contribute to foreign elements filtration. The same conclusion, about superiority of local rejection over global one concerns strict classification error ratios.

4.4 Separating Letters from Digits

Figure 3 presents rejection results. We compare various quality measures for models constructed based on random forest, SVM, and kNN with ellipsoids.

Results show that all classifiers behave well and provide similar quality of rejection.

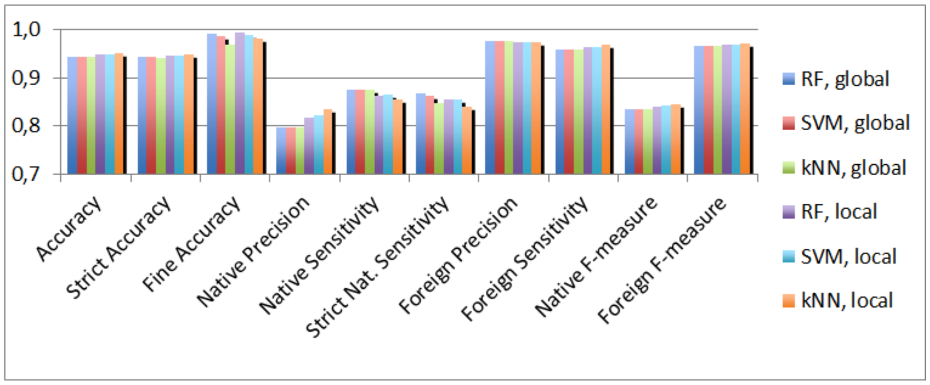

Fig. 3: Sample of: native patterns (top) and foreign patterns (bottom).

5 Conclusion

Enhancing classifiers' ability to classify objects by coupling them with ellipsoids has proven to yield better results, when working on contaminated data, than using internal rejection techniques. Although there were some differences in classification ratios between different classifiers, none of them turned out to be best when used together with global rejection method. We are aware that to truly confirm obtained results, test should be repeated on different data sets. Described in this paper set consisting of letters and digits, although being very large, might not consist of enough distinguishable classes.

Let us conclude this paper by saying that various adaptations of the idea of foreign elements rejection have a vital role in modern machine learning. It is needless to mention areas such as text mining, fraud detection, or medical diagnosis systems where we deal with various reincarnations of the foreign elements. From this perspective we believe that the study in this direction is worth further efforts.

Acknowledgment

The research is partially supported by the National Science Center, grant No 2012/07/B/ST6/01501, decision no DEC-2012/07/B/ST6/01501.

References

1. Altman N. S., *An introduction to kernel and nearest-neighbor nonparametric regression.* The American Statistician 46 (3), 1992, pp. 175-185.
2. Bertolami R., Zimmermann M., Bunke H., *Rejection strategies for offline handwritten text line recognition*, Pattern Recognition Letters 27(16), 2006, pp. 2005-2012.
3. Breiman L., *Random Forests.* Machine Learning 45 (1), 2001, pp. 532.
4. Burger T., Kessentini Y., Paquet T., *Dempster-Shafer based rejection strategy for handwritten word recognition*, Proc. of the 2011 International Conference on Document Analysis and Recognition, 2011, pp. 528-532.
5. Cortes C., Vapnik V., *Support-vector networks.* Machine Learning 20 (3), 1995, pp. 273-297.
6. Elad M., Hel-Or Y., Keshet R., *Pattern detection using maximal rejection classiffier*, C. Arcelli et al. (Eds.): Proc. International Workshop on Visual Form, Lecture Notes on Computer Science 2059, 2001, pp. 514-524.
7. Hempstalk, K., Frank, E., Witten, I., *One-class classification by combining density and class probability estimation*, Machine Learning and Knowl. Disc. in Databases, pp. 505-519, 2008.
8. Homenda, W., Jastrzebska, A., Pedrycz, W., *Rejecting Foreign Elements in Pattern Recognition Problem. Reinforced Training of Rejection Level*, in: Proc. of ICAART 2015, 2015, pp. 90-99.
9. LeCun, Y., Cortes, C., and Burges, C., *The MNIST database of handwritten digits*, in: http://yann.lecun.com/exdb/mnist.
10. Romanski, P., Kotthoff, L., *Package FSelector*, http://cran.r-project.org/web/packages/FSelector/FSelector.pdf
11. Scheme E. J., Hudgins B. S., Englehart K. B., *Confidence-based rejection for improved pattern recognition myoelectric control*, IEEE Trans. Biomed. Eng. 60(6), 2013, pp. 1563-1570.
12. Scholkopf B.et al., *Support Vector Method for Novelty Detection*, Advances in Neural Information Processing Systems 12, 1992, pp. 582 - 588.
13. Tax D. M. J., Duin R. P. W., *Growing a multi-class classifier with a reject option*, Pattern Recognition Letters 29, 2008, pp. 1565-1570.
14. Todd M. J., Yildirim E. A., *On Khachiyan's Algorithm for the Computation of Minimum Volume Enclosing Ellipsoids*, September 30, 2005, article link: http://people.orie.cornell.edu/miketodd/TYKhach.pdf
15. Wang Y., Casasent D., *A Support Vector Hierarchical Method for multi-class classification and rejection*, Proc. of International Joint Conference on Neural Networks, 2009, pp. 3281-3288.
16. http://www.mathworks.com/
17. http://www.numpy.org/
18. http://scikit-learn.org/stable/