Regression Models for Classification with Foreign Patterns Rejection

Wladyslaw Homenda[1], Agnieszka Jastrzebska[1], and Piotr Waszkiewicz[1]

[1]Faculty of Mathematics and Information Science, Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland
{homenda,A.Jastrzebska}@mini.pw.edu.pl,waszkiewiczp@student.mini.pw.edu.pl

Abstract. In the article we discuss the issue of contaminated data sets that which, apart from proper patterns, contain improper elements. To distinguish between those two kinds of patterns we use terms: native (proper patterns) and foreign (garbage patterns). We do not know neither origins, nor characteristics of foreign patterns. They can appear in the data, for example when signal segmentation algorithm is poorly, or when input streams get mixed. Foreign patterns do not belong to any native class, and therefore we do not want to account them as native. Ideally, we wish to remove them from the data entirely. However, due to the mentioned obstacles (unknown characteristics), foreign patterns rejection mechanism should not be trained based on some synthetic substitute foreign patterns as such course of actions will limit rejection mechanism's capability for generalization to various unexpected foreign patterns. To deal with the issue of contaminated data sets in this paper we propose to build foreign patterns rejection mechanism based on a collection of regression models. We propose a scheme for a classifying/rejecting model for a multi-class data set trained entirely on native patterns. The developed approach is tested in a study of handwritten digits recognition.

1 Introduction

Pattern recognition is a classical machine learning problem. Its aim is to form a model (called classifier) that, after initial training, can assign correct class labels to new patterns. The training process usually consists of feeding example data to classifier and providing answers in form of proper labels for each pattern in a training set. The ideal situation occurs when trained model is properly classifying patterns from outside of the training dataset, which means it gained ability to generalise information.

It is important to have in mind that patterns in their original form are usually some sort of a signal, for instance images or voice recordings. Due to the fact, that original patterns are often collected automatically with some signal-acquiring device, patterns that should not be accounted to any of proper classes may be recorded. Such situation may happen, when a device that is used to acquire data was automatically reset due to a power outage and a poor default calibration hinders signal segmentation process. Another scenario is when we collect data in a noisy (out of lab) environment and, apart from proper patterns, there is a lot of unexpected and unwanted input. The main problem with such erroneous patterns, called foreign patterns, is that their characteristics cannot be easily predicted and therefore cannot be included in the model training process.

The motivation for our study is to provide algorithmic approaches used for distinguishing proper patterns (called *native patterns*) from garbage and unwanted ones (called *foreign patterns*) by using only classifiers for both rejection and classification. The design assumption is to provide methods based on native patterns only so that the approach could be truly versatile and could be easily adapted to any pattern recognition problem in an uncertain environment, where foreign patterns may appear. The study focuses on a multi-class pattern recognition problem, for which we employ specifically constructed classification models that together perform actions of classification and rejection. It should be emphasised that novelty of the contribution presented in this paper is in the application of otherwise well-known methods to a new area of study.

The remainder of this paper is organized as follows. Section 2 presents the background knowledge on foreign patterns detection present in the literature. Section 3.1 presents the backbone algorithms, known in the literature, that were used to construct our models. Section 3.6 presents the proposed approach. Section 4 discusses a series of experiments. Section 5 concludes the paper and highlights future research directions.

2 Literature Review

The problem of contaminated data sets has been reappearing in research on pattern recognition in various contexts. First studies, which we may account as tightly related to foreign patterns rejection, were dealing with outliers detection. Outliers are proper native patterns, whose characteristics significantly differ from the majority of data. Outliers, due to their atypical character, may cause problems at a model training stage. Therefore, we are often advised to apply a variety of methods to detect them and, if needed, to modify or remove from the set of native patterns before further processing. Literature provides us with a selection of statistical tests, for instance discussed in [7, 21], which act as decision rules for outlier detection. Apart from statistical tests, we find more convoluted approaches. In particular, there is a wide variety of methods related to the notion of distance. Those methods evaluate either proximity of data points or their density in order to detect outliers. Popular examples of such approaches have been presented in [3, 12].

We shall stress that foreign patterns are not outliers. When it comes to outliers, we can actually determine their proper class belongingness, even though it is difficult. In contrast, foreign patterns do not belong to any class and should be removed from the data.

An area of studies, which is even more similar to foreign patterns rejection, is the so called novelty detection. Novelty elements are patterns that belong to extremely infrequent native classes. A typical domain, in which we recognize the importance of novelty elements, is text processing. Even large corpora of texts often contain meaningful, but rare key phrases. Studies on novelty detection typically utilize either a probabilistic approach or are based on the notion of distance, [15]. Among successful probabilistic approaches we can find kernel density estimators, as for example reported in [10, 11], and mixture models (for instance discussed in [14, 19]). The goal of these methods is to perform probability density estimation of the data. In contrast, distance-based methods for novelty detection rely on the assumption that novel elements are located far from majority of data. In this line of study, we find methods resolving to various sophisticated distance measures, for instance [4, 6] or even methods based on well-known clustering algorithms, as for example in [8, 20].

Foreign elements detection, alike novelty detection task, has to assume that native patterns, alike majority class, is the only available information. Therefore, model construction has to rely on native patterns only. In this paper we present a method for foreign patterns rejection based on classifying models. The approach that we work on is fit to solve multi-class classification problems (when more than two native classes are present). The proposed technique uses a collection of specifically trained binary classifiers, which set together to form a specific structure provide capability not only to classify native patterns, but also to reject foreign patterns.

3 Preliminaries

The issue of contaminated data sets frequently emerges from a substantial automation of data acquisition and processing, that results in poor data quality. The research that we present in this paper has been motivated by the need for algorithmically-aided methods for data sets purging: accepting proper patterns and rejecting foreign. We deal with possibly contaminated multi-class data sets (this is the input). As an output we wish to obtain a classified data set of native patterns, without foreign elements. In the study addressed in this paper we concentrated on enhancing well-known classifiers in order to perform a rejection task. We focused on regression models, but we also applied other popular classification algorithms, such as Support Vector Machines (SVM), random forest and k-Nearest Neighbours (kNN). In what follows we discuss applied algorithms. First, we address well-known classification models applied in the study. Second, in Section 3.6 we discuss our approach to rejection mechanism construction, based on those algorithms. Finally, we address quality measures.

3.1 The Task of Classification with Rejection

The task of classification aims at categorising unknown elements to their appropriate groups. The procedure is based on quantifiable characteristics obtained from the source signal. Those characteristics, i.e. features, are gathered in a feature vector (a vector of independent variables) and each pattern is described with one feature vector. We expect that patterns accounted to the same category are in a relationship with one another. In other words, subjects and objects of knowledge accounted to the same category are expected to be in some sense similar. There are many mathematical models that can be used as classifiers, such as SVM, random forest, kNN, regression models, or Neural Networks. Their main disadvantage lies in their need to be trained prior to usage, which makes them unable to recognize elements from a new class, not present during the training process. This behaviour can be especially troublesome in an unstable, noisy environment, where patterns sent for classification can be corrupted, distorted or otherwise indistinguishable. In such situation a proper mechanism for rejecting garbage patterns could be used to provide correct results.

3.2 Support Vector Machines

Support Vector Machines (SVM) are a collection 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 patterns 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 the 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.

3.3 Random Forest

Random forest 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 forest algorithm [2] the weak learners are decision trees, which are used to predict class labels. For a feature vector representing one pattern a decision tree calculates its class label 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 subsets. In the random forest we form a large number of classification trees, which altogether serve as a classifier. In order to grow each tree, we draw with replacement a random selection of rows from the training set. Random sampling with replacement is also called bootstrap sampling. In addition, when constructing trees for a random forest at each node we select randomly m variables out of the set of all 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.

3.4 k-Nearest Neighbors

The k-Nearest Neighbours algorithm, denoted as kNN, is an example of a "lazy classifier", where the entire training dataset is the 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 choice that the model designer has to make is selection of k and distance metrics. Both are often determined experimentally with a help of supervised learning procedures.

3.5 Regression

Regression analysis is a statistical method for describing relationships between variables. It is applied in order to predict future values of a given dependent variable based on known values of other variables (independent variables). Regression model construction resolves to formation of a function that describes how expected value of the dependent variable is related with one or more independent variables. When we deal with more than one independent variable, we have the case of multiple regression. The function can literally be a mathematical formula, but it can also be an algorithm taking us from the space of independent variables values to the space of depended variables values. Examples of algorithms building a regression model are: regression trees, neural networks, etc.

Typically, we deal with parametric regression models, where the shape of the regression function is known beforehand and our goal is to determine its components (parameters). When we apply the same set of parameters for all input variables, no matter their particular values, we have the case of global parametric regression. A general formula for parametric regression is given as follows:

$$Y = f(X, \beta) + \epsilon \tag{1}$$

where Y is the dependent variable, X is the vector of independent variables, β is the vector of regression coefficients, f is regression function with values in real numbers, ϵ is random error. The estimation target is a function of the independent variables f. The goal is to find such a function that minimizes loss.

There are many different regression models: linear regression, polynomial regression, logistic regression, Bayesian Ridge regression, etc. In this paper we will focus mainly on two regression variants: logistic and polynomial regression (please note that linear regression is in fact polynomial regression using polynomial of degree 2).

Linear regression is the simplest regression model. It describes the relationship between scalar dependent variable and one or more explanatory variables. For example, if we have three independent variables, multiple regression model looks as follows:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \tag{2}$$

Polynomial regression can be viewed as a model that uses linear regression extended by constructing polynomial features from the coefficients. This approach maintains the generally fast performance of linear methods, while allowing them to fit to a much wider range of data. The predictors resulting from the polynomial expansion of the "baseline" predictors are known as interaction features, and can be used in classification problem. Just like linear regression, polynomial regression models are usually fit using the method of least squares. Polynomial regression does not contain in its formula interactions between predictors. For instance, if we have three independent variables, polynomial regression quadratic model looks as follows:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2 + \beta_5 x_3 + \beta_6 x_3^2 \tag{3}$$

Logistic regression is a regression variant applicable to a case, when dependent variable is dichotomous (assumes only two values). However, independent variables do not need to be dichotomous, they can be numerical, categorical, interval-based, etc. Logistic regression, instead of using "classical" definition of probability, uses odds. Say that p is "classical" probability of a success, then $odds = p/(1-p)$. Logarithm of odds assumes values from the $(-\infty, +\infty)$, and thanks to this we can use any regression method that we like (not restricted to $[0, 1]$) in order to estimate the logarithm of odds. In other words, technically the logistic regression is a Generalized Linear Model with logit link function.

3.6 Rejection Mechanism for Classifiers

The proposed approach is based on an assumption that at the stage of model construction we do not have information about foreign patterns. We aim at constructing a model relying only on native patterns only. The model shall be able to:

- classify native patterns,
- reject foreign patterns.

We are investigating a multi-class classification scheme, what means that we have more than two distinct classes, say class_1, class_2, ..., class_c, c is the number of all classes.

We propose two methods based on an ensemble of specifically trained binary classifiers:

- "one-versus-all",
- "one-versus-one".

The "one-versus-all" method requires creating an array of c binary classifiers constructed using specially selected training data. Together, they provide a way to classify and reject. Training data set for each classifier in this method is consisting of two sets: the first set (denoted as "class_i") holding all training data for certain i-th native class, and the second one (denoted as "rest") being the result of a subset sum operation performed on the rest of the classes except for the class used in class_i set. Please note that as a result of the subset sum, class "rest" could contain significantly more samples than "class_i", especially when we deal with a large c. In such case, it is a viable option to undersample set "rest".

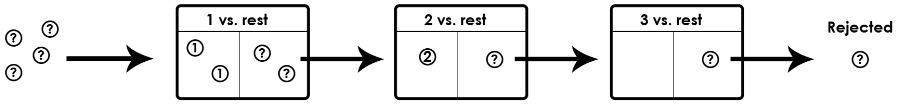

Fig. 1. "one-versus-all" rejection method. Unknown pattern passes through an array of specially prepared classifiers, one for each class. If each classifier says it is not native, it is rejected.

The actual classification with rejection is performed by presenting the unknown pattern to each of the classifiers from the array. When any classifier recognizes this element as a native one (belonging to class_i), then the pattern is treated as a recognized one, and it is assumed to be native. In a case when all classifiers reject a pattern (all binary classifiers say that it belongs to set "rest"), it is treated as a foreign pattern and it is rejected. It is worth to notice that there is a possibility that more than one classifier recognizes the pattern as a native element. In such case randomly chosen class label is assigned to this pattern. The scheme for this method is sketched in Figure 1.

Please modify Fig 1 - we'll speak in person.

The "one-versus-one" method requires preparing an array of classifiers, but this time it consists of $\binom{c}{2}$ classifiers, where c is the number of native classes. Each classifier is trained on data consisting of two sets: the first one (denoted as class_i) holding all training data entries for i-th native class, and the second one (denoted as class_o) holding all training data entries for some other class (not the same as class_i). In the end, there is one classifier for each pair of classes: 1 vs. 2, 1 vs. 3, $\ldots$, 1 vs. c, $\ldots$, $(c-1)$ vs. c. Classification with rejection mechanism is based on presenting unknown pattern to each classifier in the vector and remembering their its (e.g. classifier constructed for 1 vs. c classes can classify the pattern as belonging to class 1 or class c). In the end, those answers can be summarized and for each pattern we can form a c-elements array with numbers saying how many times this pattern was classified as belonging to class $1, 2, 3, \ldots, c$. The pattern is rejected when the difference between two biggest values in the result array is smaller than two. In such case, it is assumed that classifiers were highly uncertain as to which class should this unknown element belong to. Otherwise, the pattern is classified as an element belonging to the class, which had the biggest value in the result array. The general scheme for this method is presented in Figure 2.

Please modify Fig 2 - we'll speak in person. Prosze zmienic na obrazku oznaczenie z n na c

The modified "one-versus-one" method is the last rejection mechanism studied in this paper. It is based on the"one-versus-one" method discussed in the previous paragraph. The difference between those two methods lies in a rejection mechanism. In this method an unknown pattern is treated as a foreign element

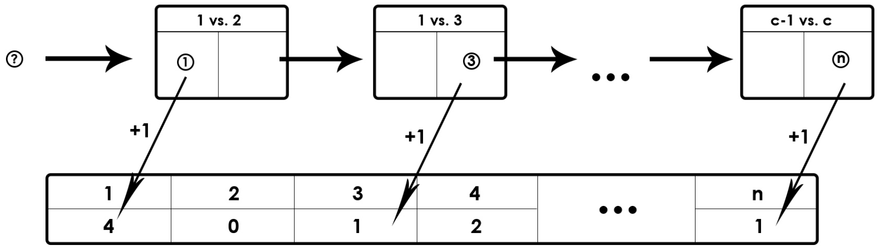

Fig. 2. The "one-versus-one" rejection method. Each element passes through a collection of "one-versus-one" classifiers and has a counter summing up classification outcomes (we sum up how many times the pattern was classified to each native class). There are two rejection rules applied in this study that use the values stored in this counter array.

if the biggest value in the result array is smaller than $(c-1)$. What it actually means, is that there must be a certain class that has always been chosen by a classifier from the vector what provides a more strict decision rule.

3.7 Quality Evaluation

In order to evaluate the quality of the proposed methods we use a set of measures. Below we list basic notions applied in the formulas for those measures, while measures themselves are placed in Table 1.

- *Correctly Classified* is the number of native patterns classified as native with a correct class label.
- *True Positives* is the number of native patterns classified as native (no matter, into which native class).
- *False Negatives* is the number of native patterns incorrectly classified as foreign.
- *False Positives* is the number of foreign patterns incorrectly classified as native.
- *True Negatives* is the number of foreign patterns correctly classified as foreign.

4 Experiments

4.1 Presentation of Datasets

Let us now move to the empirical evaluation of the proposed models. In what follows we present a study on handwritten digits recognition.

The native dataset consisted of 10,000 handwritten digits with approximately 1,000 observations in each class taken from publicly available MNIST database.

Table 1. Quality measures for classification with rejection.

$\text{Native Precision} = \dfrac{\text{TP}}{\text{TP+FP}}$	$\text{Accuracy} = \dfrac{\text{TP+TN}}{\text{TP+FN+FP+TN}}$
$\text{Foreign Precision} = \dfrac{\text{TN}}{\text{TN+FN}}$	$\text{Strict Accuracy} = \dfrac{\text{CC+TN}}{\text{TP+FN+FP+TN}}$
$\text{Native Sensitivity} = \dfrac{\text{TP}}{\text{TP+FN}}$	$\text{Fine Accuracy} = \dfrac{\text{CC}}{\text{TP}}$
$\text{Foreign Sensitivity} = \dfrac{\text{TN}}{\text{TN+FP}}$	$\text{Strict Native Sensitivity} = \dfrac{\text{CC}}{\text{TP+FN}}$

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

The MNIST dataset is publicly available on-line under [13]. We split each class in proportion ca. 7:3 and as a result we got two sets. The first one included 6,996 patterns and was used for training. The second set, the test set (the hold-out set), contained 3,004 patterns. The native data set contains ten classes ($c = 10$), one class for each digit from 0 to 9.

In order to provide foreign patterns we used a set of handwritten Latin alphabet letters (26 symbols from a to z). The dataset of foreign patterns contained 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.

The justification to assume such foreign dataset for testing purposes is that appearance of other real-world symbols, not belonging to any proper class, is a common issue in a character recognition problem. Let us stress again, that foreign patterns do not participate in the model building phase. The entire scheme is trained on native patterns set only and the foreign set is involved after the model has been built. We use it for quality evaluation only. Samples of processed patterns are displayed in Figure 3.

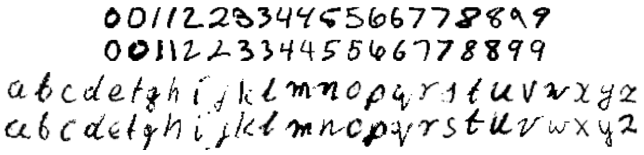

Fig. 3. Samples of native patterns – handwritten digits and foreign patterns – handwritten Latin alphabet letters.

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. We considered features standardization but the training data is sufficiently consistent (there are no outliers and the data is balanced), so we normalized those features to bring linearly all values into the range $[0, 1]$. Best first search for the optimal feature subset has been performed using the FSelector R package, [16]. The best first search is a greedy search-based feature selection method. It is a so called wrapper method, what means that we build multiple classification models based on various subsets of features and evaluate feature subset quality by checking classification accuracy on test set. We applied a 10-class SVM as an evaluator in this process. Next, we performed analysis of variance what led to further reduction of the feture set. Finally, the ready-to-use feature vector contained 24 elements.

4.2 Experimental Settings

We have formed the described three rejection mechanisms based on popular classification methods: SVM, random forest, kNN and we compare results obtained with those methods with results obtained with regression models: logistic regression and polynomial regression.

The experiment presented in this paper has been conducted in Python programming language, using scientific libraries [17, 18] and their implementations for SVM, Random Forests, kNN, logistic and polynomial regression. Several tests have been performed in order to find best suited method parameters for all classifiers by using Grid Search [18] algorithm. Final results were obtained for SVM model with the Radial Basis Function kernel, cost parameter was tuned to be eight and delta=0.5 ?? γ. When using Random Forests a total of 100 trees were used. kNN method used five neighbours when classifying.

4.3 Model Quality

Experimental results are summarized in Tables 2 (classification quality) and Table 3 (rejection quality).

czemu jest ”– ” w komorce Method2-kNN Fine accuracy?

Prosze umiescic dane z Tab 2 i Tab3 na jednym wykresie slupkowym - pewnie bedzie lepiej wygladalo i zaoszczedzimy troche miesjca. Prosze jeszcze sie upewnic czy wartosci sa dobrze przepisane. Rysujemy tylko dane estowe.

Prosze uzupelnic Tabelki 4 i 5. Prosze ”wkleic” dane z tych tabelek tez do jednego excela i sprobowac narysowac z nich wykres (na razie pierwsza wersje – zobaczymy czy zostaniemy przy tabelkach czy przy wykresie/wykresach). Rysujemy tylko dane testowe.

Tu skonczylam czytac – dokoncze sprawdzanie jak beda uzupelnione tabelki/wykresy, bo chce sie do nich odniesc w tekscie.

As it can be seen, polynomial regression model achieved better accuracy in classifying native patterns and was better at rejecting the foreign ones than other tested models. It turned out that the third approach (Method 3) yielded the best results whereas the second one performed the worst. Method 2 was the one that “destroyed” classification capabilities of classifiers and had tendency to reject native patterns. On the other hand Method 1 accepted most of the foreign elements. Those results become more obvious if one looks at them from the methods’ construction point of view.

Table 2. Classification quality obtained with an ensemble of binary classifiers based on popular models like SVM, random forests (RF) and kNN for the test (hold-out) set of native patterns.

	Method 1			Method 2			Method 3		
Classifier	SVM	RF	kNN	SVM	RF	kNN	SVM	RF	kNN
Strict Native Sens.	0.903	0.883	0.806	0.030	0.097	0	0.964	0.949	0.926
Strict Accuracy	0.141	0.178	0.082	0.737	0.781	0.8977	0.259	0.233	0.094
Fine Accuracy	0.910	0.893	0.806	0.970	0.963	–	0.969	0.955	0.926
Accuracy	0.150	0.190	0.103	0.737	0.781	0.898	0.262	0.239	0.102

Table 3. Evaluation of classification quality for a classifying/rejecting model based on polynomial and logistic regression achieved on the test set.

	Method 1		Method 2		Method 3	
Regression type	Polynomial	Logistic	Polynomial	Logistic	Polynomial	Logistic
Strict Native Sens.	0.601	0.727	0.494	0.030	0.811	0.920
Strict Accuracy	0.064	0.114	0.719	0.820	0.853	0.143
Fine Accuracy	0.602	0.738	0.969	0.882	0.959	0.925
Accuracy	0.105	0.141	0.720	0.821	0.587	0.151

The first approach rejected input only after n failed tests which makes rejection less probable the more tests are performed (more native classes means more tests). Provided pattern was treated as native and labelled as an element

Table 4. Quality of foreign patterns rejection/native patterns acceptance using proposed three methods based on popular algorithms: SVM, random forest and kNN. Results concern test set of native patterns mixed with the set of foreign elements (handwritten Latin alphabet letters).

	Method 1			Method 2			Method 3		
Classifier	SVM	RF	kNN	SVM	RF	kNN	SVM	RF	kNN
Native Precision	0.107	0.111	0.102	0.057	0.076	–	0.121	0.118	0.102
Native Sensitivity	0.993	0.988	1.000	0.101	0.102	0.000	0.995	0.994	1.000
Native F–measure	0.193	0.200	0.186	0.073	0.087	–	0.216	0.211	0.185
Foreign Precision	0.986	0.987	1.000	0.888	0.894	0.898	0.997	0.996	–
Foreign Sensitivity	0.054	0.099	0.000	0.810	0.859	1.000	0.179	0.152	0.000
Foreign F–measure	0.103	0.179	0.001	0.847	0.876	0.946	0.304	0.264	–

Table 5. Quality of foreign patterns rejection/native patterns acceptance for the three methods based on polynomial and logistic regression. Results concern test (hold-out set) of native patterns (handwritten digits) with the foreign set of handwritten letters.

	Method 1		Method 2		Method 3	
Regression type	Polynomial	Logistic	Polynomial	Logistic	Polynomial	Logistic
Native Precision	0.102	0.105	0.185	0.041	0.405	0.107
Native Sensitivity	0.998	0.984	0.509	0.034	0.846	0.995
Native F–measure	0.186	0.190	0.271	0.037	0.548	0.193
Foreign Precision	0.929	0.962	0.930	0.892	0.980	0.990
Foreign Sensitivity	0.003	0.045	0.744	0.911	0.858	0.055
Foreign F–measure	0.006	0.086	0.827	0.901	0.915	0.103

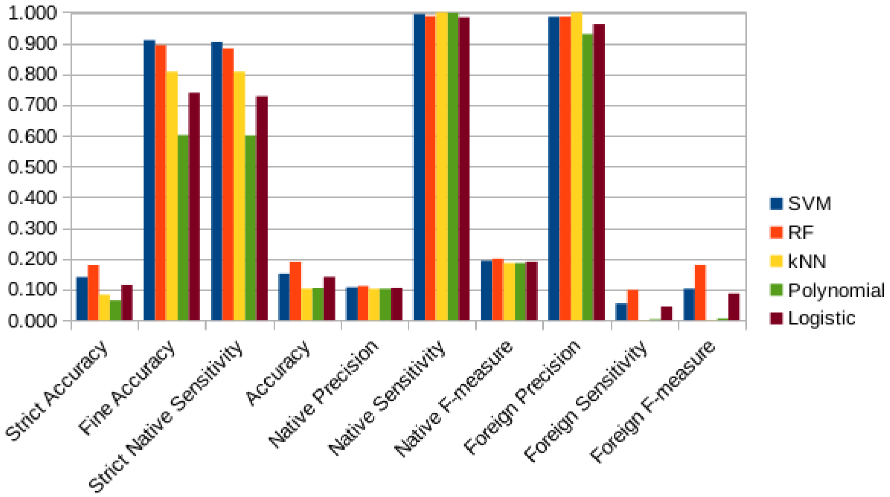

Fig. 4. Charts holding the values of quality measurements for each one of the classifiers, while using "one-versus-all" rejection method.

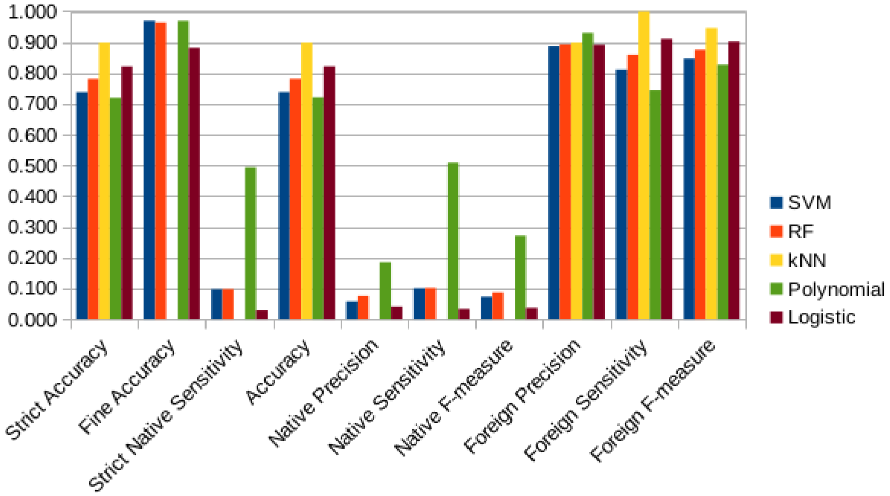

Fig. 5. Charts holding the values of quality measurements for each one of the classifiers, while using "one-versus-one" rejection method.

from the first class that passed its corresponding test. Because tests were done in order (from class 1 to class n) we observed increased number of classified elements being treated as members from the first few classes.

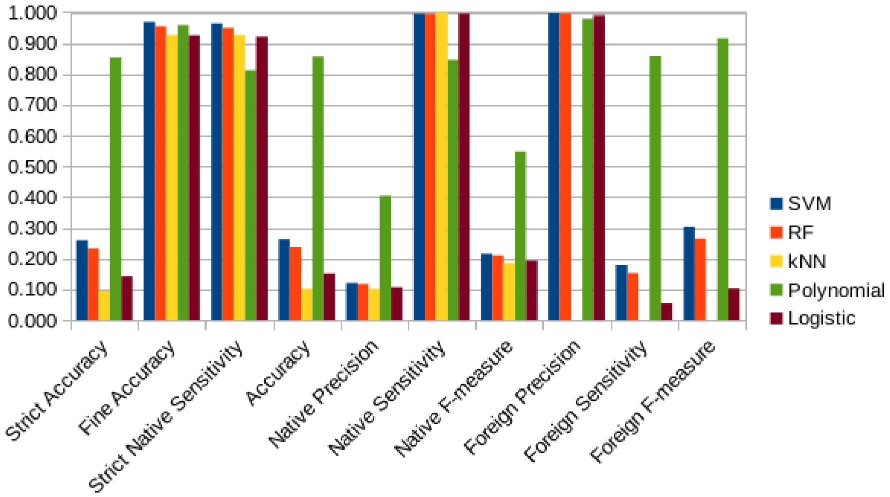

Fig. 6. Charts holding the values of quality measurements for each one of the classifiers, while using "one-versus-one version 2" rejection method.

The second method required certain class to be obviously "stronger" than the others in terms of classification. All patterns that had similarities between two or more different classes were rejected which resulted in poor classification capabilities. This result is not surprising given how often even people can be unsure what kind of handwritten symbol they see. Although it's not too hard for them to distinguish between a letter and a digit, correct classification of certain digits can be sometimes troublesome. Method 2 automatically rejected all symbols that were similar to two or more different classes.

The third method provided some sort of equilibrium between classification and rejection by modifying method 2. Elements were rejected only when there was no strong evidence that they belong to certain class. This evidence was in fact the number of times pattern was assigned certain class' label. If there was no class with required number of labels the element was identified as a foreign one. This approach can be compared to real-life situation where person classifies patterns by intuition and rejects them only when is really unsure about their true nature.

5 Conclusion

Proposed solution for enhancing classifiers with rejection mechanism did not prove to yield good results for all models. Whereas SVM, Random Forests or kNN did not benefit from introduced changes, certain regression models indeed gained identification capabilities. It's worth noting that not every of the introduced methods performed well. In the conducted tests both method 1 and 2 were outperformed by method 3. This behaviour was analysed and explained in Section 4.3.

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 match wide spectrum of problems.

Let us conclude this paper by saying that various adaptations of the idea of enhancing classifiers with rejection capabilities has a vital role in modern machine learning, especially when dealing with corrupted or unknown datasets. We believe that the study in this direction is worth further efforts.

Acknowledgement

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. Breiman, L., *Random Forests.* Machine Learning 45 (1), 2001, pp. 5-32.
3. Breunig, M. M., Kriegel, H. P., Ng, R. T., and Sander, J., *OPTICS-OF: Identifying Local Outliers*, Proc. of European Conf. on Principles of Data Mining and Knowledge Discovery, 1999.
4. Chawla, S. and Sun, P., *SLOM: a new measure for local spatial outliers*, Knowl. Inf. Syst. 9(4), pp. 412–429, 2006.
5. Cortes, C., Vapnik, V., *Support-vector networks.* Machine Learning 20 (3), 1995, pp. 273-297.
6. Ghoting, A., Parthasarathy, S. and Otey, M., *Fast mining of distance-based outliers in high-dimensional datasets*, Data Min. Knowl. Discov. 16(3), pp. 349–364, 2008.
7. Grubbs, F. E., *Sample criteria for testing outlying observations*, Annals of Mathematical Statistics 21(1), pp. 27–58, 1950.
8. He, Z., Xu, X. and Deng, S., *Discovering cluster-based local outliers*, Pattern Recognit. Lett. 24(9), pp. 1641–1650, 2003.
9. Hempstalk, K., Frank, E. and Witten, I., *One-class classification by combining density and class probability estimation*, Machine Learning and Knowl. Disc. in Databases, pp. 505–519, 2008.
10. Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T., *Gaussian processes for object categorization*, Int. J. Comput. Vis. 88(2), pp. 169–188, 2010.
11. Kim, H. and Lee, J., *Pseudo-density estimation for clustering with Gaussian processes*, Adv. Neural Netw. 3971, pp. 1238–1243, 2006.
12. Knorr, E. M. and Ng, R. T. *Finding intensional knowledge of distance-based outliers*, Proc. of Int. Conf. on Very Large Data Bases, 1999.
13. LeCun, Y., Cortes, C., and Burges, C., *The MNIST database of handwritten digits*, in: http://yann.lecun.com/exdb/mnist.
14. Paalanen, P., Kamarainen, J., Ilonen, J. and Kalviainen, H., *Feature representation and discrimination based on Gaussian mixture model probability densities practices and algorithms*, Pattern Recognition 39(7), pp. 1346–1358, 2006.

15. Pimentel, M. A. F., Clifton, D. A., Clifton, L. and Tarassenko, L., *A review of novelty detection*, Signal Processing 99, pp. 215–249, 2014.
16. Romanski, P., Kotthoff, L., *Package FSelector*, http://cran.r-project.org/web/packages/FSelector/FSelector.pdf
17. http://www.numpy.org/
18. http://scikit-learn.org/stable/
19. Song, X., Wu, M., Jermaine, C., and Ranka, S., *Conditional anomaly detection*, IEEE Trans. Knowl. Data Eng. 19(5), pp. 631–645, 2007.
20. Sun, H., Bao, Y., Zhao, F., Yu, G. and Wang,D., *CD-trees: an efficient index structure for outlier detection*, Adv. Web-Age Inf. Manage. 3129, pp. 600–609, 2004.
21. Tukey, J. W., *Exploratory Data Analysis*, Addison-Wesley, 1977.