STATISTICS IN TRANSITION new series, September 2017
Vol. 18, No. 3, pp. 443–458, DOI 10. 21307

STACKED REGRESSION WITH A GENERALIZATION OF THE MOORE-PENROSE PSEUDOINVERSE

Tomasz Górecki[1], **Maciej Łuczak**[2]

ABSTRACT

In practice, it often happens that there are a number of classification methods. We are not able to clearly determine which method is optimal. We propose a combined method that allows us to consolidate information from multiple sources in a better classifier. Stacked regression (SR) is a method for forming linear combinations of different classifiers to give improved classification accuracy. The Moore-Penrose (MP) pseudoinverse is a general way to find the solution to a system of linear equations.

This paper presents the use of a generalization of the MP pseudoinverse of a matrix in SR. However, for data sets with a greater number of features our exact method is computationally too slow to achieve good results: we propose a genetic approach to solve the problem. Experimental results on various real data sets demonstrate that the improvements are efficient and that this approach outperforms the classical SR method, providing a significant reduction in the mean classification error rate.

Key words: stacked regression, genetic algorithm, Moore-Penrose pseudoinverse.

1. Introduction

Suppose that a training sample has been collected by sampling from a population P consisting of K subpopulations or classes $G_1,\dots,G_K$. The ith observation is a pair denoted by $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is a d-dimensional feature vector and y_i is the label for recording class membership. The corresponding pair for an unclassified observation is denoted by $(\mathbf{x}, y)$. In this case $\mathbf{x}$ is observed, but the class label y is unobserved. The goal of classification is to construct a classification rule for predicting the membership of an unclassified feature vector $\mathbf{x} \in P$. An automated classifier can be viewed as a method of estimating the posterior probability of membership of G_j. The classification rule assigns $\mathbf{x}$ to the group with the largest posterior probability estimate. We denote the posterior probability of membership of G_k by

$$p_k(\mathbf{x}) = P(y = k|\mathbf{x}). \tag{1}$$

[1]Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Umultowska 87, 61-614 Poznań, Poland. E-mail: tomasz.gorecki@amu.edu.pl.

[2]Faculty of Civil Engineering, Environmental and Geodetic Sciences, Koszalin University of Technology, Śniadeckich 2, 75-453 Koszalin, Poland. E-mail: mluczak@wilsig.tu.koszalin.pl.

In practice, it is not clear how one should choose a classifier. An even more practical difficulty in choosing a classifier is that different classifiers have different merits and, as a result, in a given situation one classifier may perform better than another. Consider the following typical situation (Mojirsheibani (2002)). Suppose that there are 3 classes, two of which are approximately multivariate normal distributions, while the third class is non-normal. Then linear or quadratic classification function might work best for separating the first two classes (the normal distributions), while the nearest neighbor rule is perhaps more appropriate in the non-normal case. This example suggests that perhaps one should consider methods that somehow combine the best features of different individual classifiers. Some possible benefits of such combined methods are as follows:

1. Lowering the risk of choosing the wrong classifier.

2. Obtaining more stable prediction performance, since in combining different methods certain biases inherited from particular models could be offset.

3. Producing a better prediction of the classification of new observations, since the combined method gives decision-makers additional information from different sources.

The purpose of ensemble learning is to construct a learning rule which combines a number of base methods, so that the final classifier gives better performance than any individual classifier (Rokach (2010)). Three groups of combining methods could be distinguished as follows (Duin and Tax (2000)):

- Parallel combining of classifiers computed for different feature sets. Parallel classifiers are often of the same type.

- Stacked combining of different classifiers computed for the same feature space. Stacked classifiers may be of a different nature, e.g. the combination of a neural network, a nearest neighbor classifier and a parametric decision rule.

- Combining weak classifiers. In this case, large sets of simple classifiers are trained on modified versions of the original data set.

For all cases, the question arises how the classifiers should be combined. The most intuitive approach is a simple majority vote (Kuncheva (2004)), whereby every classifier computes a class label and the label that receives the most votes is the output of the ensemble. In addition, one may also train a classifier using, e.g. the BKS method (Huang and Suen (1995)), Wernecke's method (Wernecke (1992)) or the fuzzy integral (Cho and Kim (1995)). Currently, the most interesting ensemble methods are bagging (Breiman (1996b)) and boosting (Schapire (1990)), random forests (Breiman (2001)), and finally SR, introduced by Wolpert (1992). In SR, the posterior probability estimates are combined by weighted sums, where the weights are obtained by classical least squares regression. Stacking is still used in practice (Sehgal et al. (2005), Doumpos and Zopounidis (2007), Marqués et al. (2012)). Although SR is applied to real-world problems less frequently than other ensemble methods, such as bagging or boosting, the exponential growth of data as well as the diversity of these data continue to make SR an interesting alternative (Sesmero et al. (2015)). There are also some new papers which propose extensions to SR (Džeroski and Ženko (2004), Rooney et al. (2004a), Rooney et al. (2004b), Xu et al. (2007), Ozay and Vural (2008), Ni et al. (2009)), Ledezma et al. (2010), Shunmugapriya and Kanmani (2013). An informative overview of SR methods can be found in Sesmero et al. (2015).

Sigletos et al. (2005) pointed out that stacking using probabilities performs comparably or significantly better than voting. This result has inspired us to consider some extension of SR. The classical stacked regression method uses the MP inverse of a matrix to solve a set of normal equations, whereas we try to find a specific generalization of the MP inverse. We construct a parametric family of generalized MP inverses and use it in the SR model. Then we choose models with the lowest cross-validation (leave-one-out) error rate and combine them by a mean rule (Kuncheva (2004)).

However, for most datasets there are too many models to compute the cross-validation (CV) error for all of them. The problem is too complex to find an exact solution (or if done, it takes too long to calculate the solution exactly). The most feasible approach, then, is to use a meta-heuristic method (Michalewicz, Fogel (2004)). A genetic algorithm (GA) is meta-heuristic, which means it estimates a solution. Therefore, we propose GA to solve our problem. GA has a number of advantages. It can quickly scan a vast solution set. Bad proposals do not negatively affect the end solution, as they are simply discarded. It can solve every optimization problem which can be described with the chromosome encoding. It solves problems with multiple solutions. Since the genetic algorithm execution technique is not dependent on the error surface, we can solve multi-dimensional, non-differential, non-continuous, and even non-parametric problems. It is a method which is very easy to understand and it demands practically no mathematical knowledge.

In this paper, we first present the main ideas of SR (Section 2). In the same section we describe generalized inverses of matrices. At the end of this section we explain our concept for extended SR and we precisely describe the genetic approach to our extension. In the paper the performances of the methods are compared and the bootstrap error of classification is considered. A total of 15 real data sets are used. The methods and data sets used are described in Section 3. Section 4 contains the results of our experiments on the described real data sets. The results of the research are explained, the differences between the classifiers being shown accurately. The same section contains a statistical comparison of the described methods. Final conclusions are given in Section 5.

2. Methods

2.1. Stacked regression

Wolpert (1992) presented an interesting idea for the combining of classifiers, known as stacked generalization. He was not searching for the best classifier in the set of all c classifiers, but for a linear combination of them. Since each single one has some advantages, combining them is reasonable. Wolpert's proposal was translated into the language of statistics by Breiman (1996a). He called it SR. Then, Leblanc and Tibshirani (1996) took advantage of it to construct a combined classifier in discriminant analysis. Stacking was shown by them theoretically to be a bias-reducing technique. A combined classifier is a linear combination of estimated posterior probabilities. An estimate of $p_k(\mathbf{x})$ obtained by the jth classifier is denoted by

$$\hat{p}_k^j(\mathbf{x}); \quad k = 1,2,\ldots,K; \quad j = 1,2,\ldots,c. \tag{2}$$

We have c classifiers and K classes, so we have $K \cdot c$ estimates, which are arranged in the vector:

$$\hat{\mathbf{p}}(\mathbf{x}) = (\hat{p}_1^1(\mathbf{x}),\ldots,\hat{p}_K^1(\mathbf{x}),\ldots,\hat{p}_1^c(\mathbf{x}),\ldots,\hat{p}_K^c(\mathbf{x}))'. \tag{3}$$

These estimates are arranged in the stack as rows of the matrix $\mathbf{P}$. Let $\mathbf{u}_k$ be a vector having a 1 in the ith position if the observation belongs to class k and 0 otherwise, so

$$u_{i,k} = \begin{cases} 1, & \text{if } y_i = k, \\ 0, & \text{if } y_i \neq k. \end{cases} \tag{4}$$

The SR model has the form:

$$\mathbf{u}_k = \mathbf{P}\boldsymbol{\beta}_k + \boldsymbol{\varepsilon}_k, \tag{5}$$

where $\boldsymbol{\beta}_k$ is a $K \cdot c \times 1$ vector of unknown SR coefficients and $\boldsymbol{\varepsilon}_k$ a vector of errors with zero mean. A least-squares estimate of $\hat{\boldsymbol{\beta}}_k$ can be obtained by solving the following equation:

$$\mathbf{P}'\mathbf{P}\boldsymbol{\beta}_k = \mathbf{P}'\mathbf{u}_k \tag{6}$$

with respect to $\boldsymbol{\beta}_k$.

The estimates of posterior probability obtained from the classifiers sum to one, so

$$\sum_{k=1}^{K} \hat{p}_k^j = 1; \quad j = 1,2,\ldots,c. \tag{7}$$

Hence, the columns of matrix $\mathbf{P}$ are subject to c linear constraints, $\mathbf{P}$ is not full column rank and $\mathbf{P}'\mathbf{P}$ is a singular matrix. We can use the MP generalized inverse of the matrix $\mathbf{P}'\mathbf{P}$ (Breiman (1996)), denoted by $(\mathbf{P}'\mathbf{P})^+$, and

$$\hat{\boldsymbol{\beta}}_k = (\mathbf{P}'\mathbf{P})^+\mathbf{P}'\mathbf{u}_k. \tag{8}$$

Given the estimates $\hat{\boldsymbol{\beta}}_1,\ldots,\hat{\boldsymbol{\beta}}_K$, we classify $\mathbf{x}$ using the dot product:

$$\hat{u}_{0,k} = \hat{\mathbf{p}}'(\mathbf{x})\hat{\boldsymbol{\beta}}_k. \tag{9}$$

We select the class with the largest values of $\hat{u}_{0,k}$. These scalar products are called discriminant indices.

2.2. Algorithm

In SR, the MP generalized inverse $\mathbf{A}^+$ is used to compute the coefficients $\hat{\boldsymbol{\beta}}_k$ (see Equation (8)). The main idea of this paper is to use another generalized inverse. The MP pseudoinverse is a general way to find the solution to a system of linear equations (eg. Ben-Israel and Greville (2003), Kyrchei (2015)).

We consider a general (real) matrix $\boldsymbol{A}$ of order $m \times n$ and rank which may be less than $\min(m,n)$. If $\boldsymbol{M},\boldsymbol{N}$ are positive definite matrices, and there exist factorizations $\hat{\boldsymbol{N}}'\hat{\boldsymbol{N}} = \boldsymbol{N}$, $\hat{\boldsymbol{M}}'\hat{\boldsymbol{M}} = \boldsymbol{M}$, then

$$\boldsymbol{A}_{\boldsymbol{MN}}^{+} = \hat{\boldsymbol{N}}^{-1}(\hat{\boldsymbol{M}}\boldsymbol{A}\hat{\boldsymbol{N}}^{-1})^{+}\hat{\boldsymbol{M}}, \tag{10}$$

satisfies the condition

$$\begin{gathered} ||\boldsymbol{A}_{\boldsymbol{MN}}^{+}\boldsymbol{y}||_n \le ||\boldsymbol{x}||_n \\ \forall \boldsymbol{x} \in \{\boldsymbol{x}\colon ||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}||_m \le ||\boldsymbol{A}\boldsymbol{z}-\boldsymbol{y}||_m \forall \boldsymbol{z} \in \mathbb{R}^n\}, \end{gathered} \tag{11}$$

where $\|\boldsymbol{x}\|_n = \sqrt{\boldsymbol{x}'\boldsymbol{N}\boldsymbol{x}}$ and $\|\boldsymbol{y}\|_m = \sqrt{\boldsymbol{y}'\boldsymbol{M}\boldsymbol{y}}$ are norms in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. $\boldsymbol{A}^+_{\boldsymbol{MN}}$ is referred to as the minimum $\boldsymbol{N}$-norm $\boldsymbol{M}$-least-squares g-inverse of $\boldsymbol{A}$. When $\boldsymbol{M}$ and $\boldsymbol{N}$ are identity matrices, we use the notation $\boldsymbol{A}^+$ and call it the MP inverse (pseudoinverse). The matrix $\boldsymbol{A}^+_{\boldsymbol{MN}}$ is also called the weighted Moore-Penrose inverse of $\boldsymbol{A}$. The weighted MP inverse of a matrix has many important applications eg. in statistics, prediction theory and curve fitting. For a wider survey and more details we refer readers to Rao and Mitra (1971).

If $\boldsymbol{M}$ is positive semi-definite, then $\|\boldsymbol{y}\|_m$ is a seminorm and the right side of Equation (10) does not need to be a g-inverse. We denote it by $\boldsymbol{A}^*_{\boldsymbol{MN}}$ and $\boldsymbol{A}^*_{\boldsymbol{M}}$ if $\boldsymbol{N} = \boldsymbol{I}$.

In our method we use $\mathbf{A}^*_{\mathbf{M}}$ with a special form of matrix $\mathbf{M}$ instead of $\mathbf{A}^+$. Precisely, we use Equation (10) with the assumptions

$$\hat{\mathbf{N}} = \mathbf{N} = \mathbf{I}, \qquad \hat{\mathbf{M}} = \mathbf{M} = \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_m \end{bmatrix} \tag{12}$$

where $a_i = 0$ or 1 for $i = 1, \dots, m$ $(m = K \cdot c)$. This leads to the seminorm

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}'\mathbf{M}\mathbf{x}} = \sqrt{x_{i_1}^2 + x_{i_2}^2 + \dots + x_{i_k}^2}, \qquad 1 \le k \le m \tag{13}$$

for $\mathbf{x} = (x_1, x_2, \dots, x_m) \in \mathbf{R}^m$. Then Equation (10) has the form

$$\mathbf{A}^*_{\mathbf{M}} = (\mathbf{MA})^+\mathbf{M}. \tag{14}$$

Thus, we can calculate SR coefficients $\hat{\boldsymbol{\beta}}_k$ by the formula

$$\hat{\boldsymbol{\beta}}_k = (\mathbf{P}'\mathbf{P})^*_{\mathbf{M}}\mathbf{P}'\mathbf{u}_k = (\mathbf{MP}'\mathbf{P})^+\mathbf{MP}'\mathbf{u}_k. \tag{15}$$

In the algorithm the matrix $\mathbf{N}$ corresponds to the norm $\|\cdot\|_n$ in Equation (11). In the case of SR the norm operates on the space of probabilities, so it seems that the simplest choice is to take the Euclidean norm, i.e. $\mathbf{N} = \mathbf{I}$.

We only take ones and zeros in the diagonal of the matrix $\boldsymbol{M}$ because it has been proven (Górecki, Łuczak (2013)) that the value of $\boldsymbol{A}^*_{\boldsymbol{M}}$ depends only on whether the coefficients a_i are zeros or not. Each zero in the diagonal trims a part (but not all) of the information about one pair consisting of a class and a classifier.

The number of models (diagonals) which have to be tested by the CV process at the learning phase is equal to $2^{K \cdot c}$, where c and K depends neither on the number of elements of the learning data set nor on the number of features of the data.

In our algorithm we choose the best combinations of ones and zeros in the diagonal of matrix **M** using the genetic algorithm, and form the SR model with the lowest CV error rate. If there is more than one best model the mean classifier is performed for them; the classification index of our method is the mean of indices for the best models (in the sense of CV) that joins all the information from them. We will call this method generalized stacked regression (GSR).

2.3. Genetic algorithm

The sequence of steps in a basic GA is shown in Fig. 1. The population consists of individuals (genotypes) which are diagonal of matrix $\boldsymbol{M}$. Each individual is a binary vector (genes) that corresponds to numbers (ones or zeros) in the diagonal of $\boldsymbol{M}$. All populations in the algorithm have a constant number n of individuals.

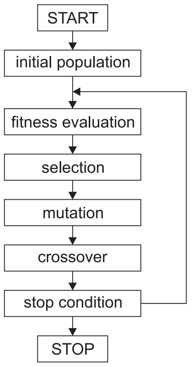

Figure 1: Genetic algorithm

Initial population: This is generated randomly. We construct n individuals where each position in the vector (diagonal) may be 0 or 1 with probability of 0.5. *Fitness evaluation*: The fitness function value is computed by the leave-one-out CV method. The CV error rate is the fitness value of any individual. The smaller the value, the better fitness an individual has. *Selection*: We use tournament selection. Two individuals are chosen from the population at random. The one with higher fitness is selected for mutation and crossover. This is repeated n times to make a new population. *Mutation*: We use standard one-point mutation. For each individual each position in the vector has the same probability of mutation p_m. The mutation is negation of the number (0 or 1) in the position (Fig. 2). It is repeated an appropriate number of times to make a new population of size n. *Crossover*: We use a standard one-point crossover operation. Each individual can be chosen to crossover with constant probability p_c. For every pair of chosen individuals, the point of crossing is fixed at random. Then the positions to the right of the point are exchanged with one another (Fig. 2). The operation is repeated an appropriate number of times to make a new population of size n.

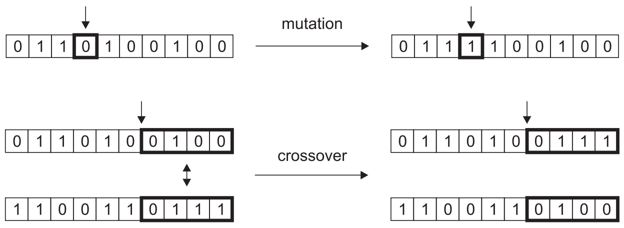

Figure 2: Reproduction

Stop condition: We do not use a fixed number of generations in GA. For so many different data sets, the algorithm needs different numbers of steps to reach a satisfactory result. The process is repeated until a stop condition has been reached. The stop condition depends on the behaviour of the mean fitness value in the populations over k steps of the algorithm. If during k steps the mean does not become smaller than the smallest value of the mean up to the current generation, the algorithm is terminated. We shall call the number k the stop condition number.

3. Computational experiments

Data sets

We performed experiments on 15 real data sets. The description of the data sets used is presented in Table 1.

The data set *beetles* comes from Seber (1984), *chemistry* and *irradiation* come from Morrison (1976), and *football* is from Gleim (1984). The other data sets originate from the UCI Machine Learning Repository (Frank and Asuncion (2010)).

Experimental setup

The classification errors were estimated by the leave-one-out and bootstrap methods. Leave-one-out was used to find the best diagonals (those with the smallest error rates) of matrix $\mathbf{M}$. The method was used to compute the value of the fitness function in the GA. The number of individuals per population was fixed at a constant value of $n = 20$. We chose probabilities of mutation $p_m = 0.01$ and crossover $p_c = 0.8$. As a selection method we use tournament selection. Different stop condition numbers were tried, $k = 0, \ldots, 10$. For the final result of our method we assumed the best case $k = 10$. For each data set we repeated the algorithm 10 times.

Table 1: The description of the data sets used.

Name	Number of features	Number of classes	Number of instances
beetles	2	3	64
breast tissue	9	6	106
chemistry	3	4	45
flags	28	6	194
football	6	3	90
glass	9	6	214
heart_c	13	5	297
heart_h	10	5	261
heart_s	10	5	105
iris	4	3	150
irradiation	3	4	45
libras	90	15	360
sonar	60	2	208
wine	13	3	178
zoo	16	7	100

In the next step, the mean classifier was performed for models with each of these diagonals. We calculated the bootstrap classification error rate (1000 repetitions). We finally fixed the mean of these bootstrap error rates as the error rate of our method.

The success of stacked generalization depends on the methods that are combined. Obviously, if all the methods provide the same class assignments, then a combined model will not provide any improvement in classification accuracy. The classification performance of the methods is of rather limited interest in this context, i.e. one is not interested in combining highly accurate methods, but in combining methods that are able to consider different aspects of the problem and the data used. Of course, it is rather difficult to find which methods meet this requirement. However, it is expected that consideration of different types of methods (e.g. methods which are not simple variations of one another) should be beneficial in stacking (Wolpert (1992)). We performed computations for three basic classifiers:

1. Nearest neighbors classifier with 5 neighbors (5NN). Objects are assigned based on a majority vote among the classes of the 5 nearest training points. The 5NN variant of the nearest neighbor classifier was chosen on the one hand to avoid an excess of zero posterior probabilities, and on the other hand because too large a number of neighbors leads to an excessive number of ties, whose resolution can be problematic (Górecki, (2005)). Too many neighbors may also be problematic for small data sets and for data sets with small classes.

2. Naive Bayes classifier (NB). We assume that the value of a particular feature is independent of the value of any other feature, given the class variable. This reduces the problem to d one-dimensional density estimation problems, within each of the K groups. We adopted a typical assumption that the continuous values associated with each class are distributed according to a Gaussian distribution. NB classifier works quite well in many complex real-world situations. In addition, Zhang (2004) investigated the optimality of NB under the Gaussian distribution, and presented the explicit sufficient condition under which NB is optimal, even though the independence assumption is violated.

3. Binary decision tree classifier (TREE). The algorithm computes a binary decision tree on a multi-class data set. Thresholds are set such that the Gini impurity is minimized in each step. Early pruning is used in order to avoid overtraining (Breiman et al. (2005)).

We focus on methods with a fast implementation (at the same time popular and relatively efficient), because GA itself is very time-consuming. The methods should be also significantly diversified in order for the ensemble method to yield better results (Kuncheva and Whitaker (2003)). Noteworthy is also Table 2 in Sesmero et al. (2015), where one can find information about base classifiers used in SR. The methods we selected are commonly used and meet the criteria of fast implementation and efficiency. More details about the methods we use can be found in Webb (2002).

In the computational process we used the program PRTools 4.2.1 (`http://www.prtools.org`). This is a Matlab (version R2011a) based toolbox for pattern recognition (van der Heijden et al. (2004)). In each procedure we used the default parameters.

Results

Graphs of example runs of our algorithm are shown in Fig. 3. We can observe rather standard behaviour of GA. We use tournament selection, which is not an elitist selection, so we can observe that the minimum of the fitness function does not decrease monotonically. The mean tends to a minimum and the algorithm is terminated if the stop condition is reached, i.e. if the mean does not decrease for a number of generations.

The results of the research are presented below in tabular form. Bootstrap error rates are presented in Table 2. From left to right the columns show the errors made by individual methods, SR, and our GSR. 5NN performed clearly the best on 2 of the data sets, NB on 3 and GSR on 10.

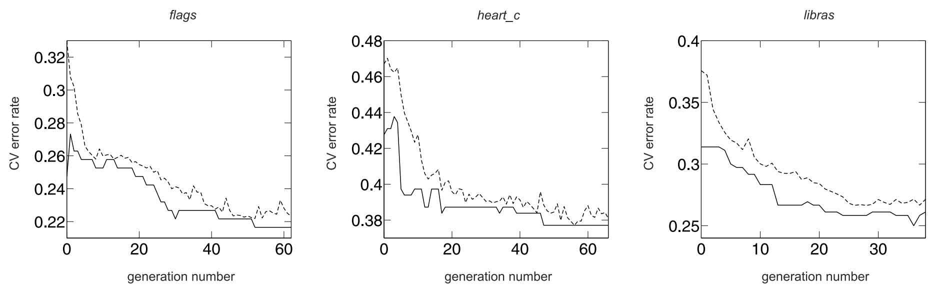

Figure 3: Runs of GA for example data sets. Fitness function value (mean $(\cdots)$ and minimum (—) of CV error rate) depending on the generation number. From left: *flags*, *heart_c*, *libras* data set.

Table 2: Bootstrap error rates (in %). Clearly the best results are marked with the symbol •.

Data set	5NN	NB	TREE	SR	GSR
beetles	6.08	6.84	5.34	4.95	• 4.34
breast tissue	48.72	38.49	43.29	41.17	• 37.47
chemistry	65.38	70.15	66.93	66.92	• 64.67
flags	66.39	35.77	43.50	40.64	• 33.47
football	40.49	• 32.60	40.51	38.99	32.65
glass	34.40	39.67	37.34	34.98	• 33.65
heart_c	57.36	41.52	52.13	51.41	• 41.48
heart_h	50.19	37.09	53.98	54.00	• 35.39
heart_s	64.86	63.76	66.53	66.55	• 63.20
iris	• 4.41	5.98	9.77	7.38	5.88
irradiation	70.62	72.06	71.76	71.79	• 70.21
libras	• 27.92	40.97	56.68	35.09	35.81
sonar	• 23.34	25.71	32.29	32.29	23.39
wine	30.67	• 3.35	11.72	5.14	3.69
zoo	10.34	8.36	10.90	9.82	• 5.54

In Table 3 we present relative differences of bootstrap error rates between SR and other methods (a positive value means that SR is better in that case). We may use the mean ratio of error rates across data sets as a measure of relative performance (Bauer and Kohavi (1998)).

Table 3: Average relative bootstrap error rates (in%) on all data sets.

	$\frac{5NN-SR}{SR}$	$\frac{NB-SR}{SR}$	$\frac{TREE-SR}{SR}$	$\frac{GSR-SR}{SR}$
MEAN	34.51	-7.02	17.65	-16.08

A direct comparison of SR with our revised version strongly favors the revised method. A graphical comparison of GSR and SR is presented in Fig. 4. We see that the new method, GSR, is clearly superior to SR on most of the examined data sets (with a 16.08% average relative error reduction for all data sets). The error rate of our method is slightly greater than for standard stacked regression in only one case (*libras*). One of the models is the standard SR (for $\mathbf{M} = \mathbf{I}$), so if it is the best model then it should be chosen. It sometimes fails because of the procedure for finding parameters. If we tried another, more sophisticated, method of finding the best model instead of CV, we would have better results.

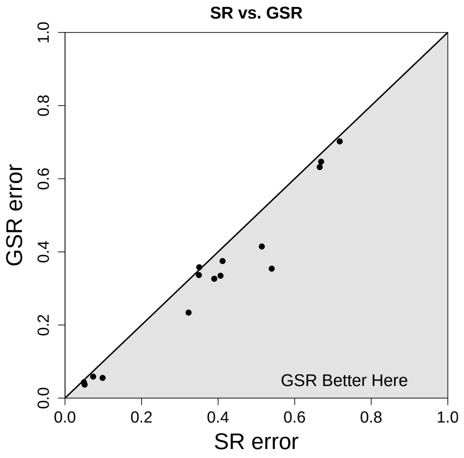

Figure 4: Comparison of test errors.

To distinguish between the methods, we performed a statistical comparison. We tested the hypothesis that there are no differences between the classifiers. We used Iman and Davenport's (1980) rank test, which is a less conservative variant of Friedman's ANOVA test. We compare the mean ranks of classifiers. The *p*-value from this test is equal to 5.29E-6. We can therefore proceed with the post hoc tests to detect which classifiers are significantly different from each other. Garcia and Herrera (2008) showed that the dynamic procedure of Bergmann and Hommel (1988) is the most powerful post hoc test. The results of multiple comparisons are given in Table 4 and Table 5. We finally obtained, at the significance level $\alpha = 0.05$, two homogeneous groups of classifiers: GSR and the rest of classifiers. Hence, GSR is significantly better than the other examined classifiers.

Table 4: *p*-values in the Bergmann–Hommel post hoc test.

i	Hypothesis	*p*-value
1	TREE vs. GSR	1.65×10^{-5}
2	SR vs. GSR	0.004
3	5NN vs. GSR	0.011
4	NB vs. GSR	0.032
5	NB vs. TREE	0.196
6	5NN vs. TREE	0.220
7	TREE vs. SR	0.332
8	NB vs. SR	1.000
9	5NN vs. SR	1.000
10	5NN vs. NB	1.000

Table 5: Results of the Bergmann–Hommel post hoc test.

Procedure	Ranks mean		
GSR	4.60	a	
NB	3.07		b
5NN	2.87		b
SR	2.63		b
TREE	1.83		b

4. Conclusions

Our research has shown that the use of a generalization of the MP pseudoinverse of a matrix in the SR model of object classification gives good results. In the general case our method seems to outperform SR and often even the best individual classifier. Owing to the parametric approach and the genetic optimization method, the proposed method enables one to choose an appropriate model for any data set and any individual classifiers. On the other hand, our method seems to prevent overfitting. Due to the high nonlinearity, the method does not easily lead to a rigorous theoretical analysis. However, the experiments that we have conducted provide evidence of the power and usefulness of our method.

REFERENCES

BAUER, E., KOHAVI, R., (1999). An experimental comparison of voting classification algorithms: bagging, boosting, and variants. Machine Learning, 36, 105–139.

BEN-ISRAEL, A., GREVILLE, T.N.E. (2003). Generalized inverses. Theory and applications. Springer.

BERGMANN, G., HOMMEL, G., (1988). Improvements of general multiple test procedures for redundant systems of hypotheses. In Multiple Hypotheses Testing. P. Bauer, G. Hommel and E. Sonnemann (eds.) Springer, 110–115.

BREIMAN, L., (1996a). Stacked regression. Machine Learning, 24, 49–64.

BREIMAN, L., (1996b). Bagging predictors. Machine Learning, 24, 123–140.

BREIMAN, L., (2001). Random forests. Machine Learning, 45, 5–32.

BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R.A., STONE, C.J., (1984). Classification and regression trees, Wadsworth, California.

CHO, S.B., KIM, J.H., (1995). Multiple network fusion using fuzzy logic. IEEE Transactions on Systems, Man, and Cybernetics, 6, 497–501.

DOUMPOS, M., ZOPOUNIDIS, C., (2007). Model combination for credit risk assessment: A stacked generalization approach. Annals of Operations Research, 151, 289–306.

DUIN, R., TAX, D., (2000). Experiments with classifier combining rules. Lecture Notes in Computer Science, 1857, 16–29.

DŽEROSKI, S., ŽENKO, B., (2004). Is combining classifiers with stacking better than selecting the best one? Machine Learning, 54, 255–273.

FRANK, A., ASUNCION, A., (2010). UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml` Irvine, CA: University of California, School of Information and Computer Science.

GARCIA, S., HERRERA, F., (2008). An extension on ”statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. Journal of Machine Learning Research, 9, 2677–2694.

GLEIM, G., (1984). The profiling of professional football players. Clinical Sport Medicine, 3(1), 185–97.

GÓRECKI, T., (2005). Effect of choice of dissimilarity measure on classification efficiency with nearest neighbor method. Discussiones Mathematicae Probability and Statistics, 25(2), 217–239.

GÓRECKI, T., ŁUCZAK, M., (2013). Linear discriminant analysis with a generalization of Moore-Penrose pseudoinverse. International Journal of Applied Mathematics and Computer Science, 26(2), 463–471.

HUANG, Y.S., SUEN, C.Y., (1995). A method of combining multiple experts for the recognition of unconstrained handwritten numerals. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17, 90–93.

IMAN, R., DAVENPORT, J., (1980). Approximations of the critical region of the Friedman statistic. Communications in Statistics - Theory and Methods, 9(6), 571–595.

KUNCHEVA, L.I., (2004). Combining Pattern Classifiers: Methods and Algorithms. Wiley.

KUNCHEVA, L., WHITAKER, C., (2003). Measures of diversity in classifier ensembles. Machine Learning, 51, 181–207.

KYRCHEI, I., (2015). Cramer's rule for generalized inverse solutions. In Advances in Linear Algebra Research I. Kyrchei (ed.) Nova Science Publishers, 79–132.

LEBLANC, M., TIBSHIRANI, R., (1996). Combining estimates in regression and classification. Journal of the American Statistical Association, 91, 1641–1650.

LEDEZMA, A., ALER, R., SANCHIS, A., BORRAJO, D., (2010). GA-stacking: evolutionary stacked generalization. Intelligent Data Analysis, 14, 89–119.

MARQUÉS, A., GARCÍA, V., SÁNCHEZ, J., (2012). Exploring the behavior of base classifiers in credit scoring ensembles. Expert Systems with Applications, 39(11), 10244–10250.

MICHALEWICZ, Z., FOGEL, D.B., (2004). How To Solve It: Modern Heuristics. Springer.

MOJIRSHEIBANI, M., (2002). A comparison study of some combined classifiers. Communications in Statistics - Simulation and Computation, 31(2), 245–260.

MORRISON, D.F., (1976). Multivariate Statistical Methods. McGraw-Hill.

NI, W., BROWN, S., MAN, R., (2009). Stacked partial least squares regression analysis for spectral calibration and prediction. Journal of Chemometrics, 23, 505–517.

OZAY, M., VURAL, F.T.Y., (2008). On the performance of stacked generalization classifiers. Lecture Notes in Computer Science, 5112, 445–454.

RAO, C.R., MITRA, S.K., (1971). Generalized Inverse of Matrices and its Applications. Wiley.

ROKACH, L., (2010). Ensemble-based classifiers. Artificial Intelligence Review, 33, 1–39.

ROONEY, N., PATTERSON, D., ANAND, S., TSYMBAL, A., (2004). Dynamic integration of regression models. Lecture Notes in Computer Science, 3077, 64–173.

ROONEY, N., PATTERSON, D., NUGENT, C., (2004). Reduced ensemble size stacking. Tools with Artificial Intelligence. ICTAI 6th IEEE International Conference, 266–271.

SCHAPIRE, R.E., (1990). The strength of weak learnability. Machine Learning, 5, 197–227.

SEBER, G.A.F., (1984). Multivariate Observations. New York: Wiley.

SEHGAL, M.S.B., GONDAL, I., DOOLEY, L., (2005). Stacked regression ensemble for cancer class prediction. Industrial Informatics INDIN 3rd IEEE International Conference, 831–835.

SESMERO, M., LEDEZMA, A., SANCHIS, A., (2015). Generating ensembles of heterogeneous classifiers using Stacked Generalization. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(1), 21–34.

SHUNMUGAPRIYA, P., KANMANI, S., (2013). Optimization of stacking ensemble configurations through artificial bee colony algorithm. Swarm and Evolutionary Computation, 12, 24–32.

SIGLETOS, G., PALIOURAS, G., SPYROPOULOS, C.D., HATZOPOULOS, M., (2005). Combining information extraction systems using voting and stacked generalization. Journal of Machine Learning Research, 6, 1751–1782.

WERNECKE, K., (1992). A coupling procedure for discrimination of mixed data. Biometrics, 48, 497–506.

WOLPERT, D., (1992). Stacked generalization. Neural Networks, 5, 241–259.

VAN DER HEIJDEN, F., DUIN, R.P.W., DE RIDDER, D., TAX. D.M.J., (2004). Classification, Parameter Estimation and State Estimation: An Engineering Approach Using Matlab, New York: Wiley.

WEBB, A., (2002). Statistical Pattern Recognition. New York: Wiley.

XU L., JIANG J.H., ZHOU Y.P., WU H.L., SHEN G.L., YU R.Q., (2007). MCCV stacked regression for model combination and fast spectral interval selection in multivariate calibration. Chemometrics and Intelligent Laboratory Systems, 87(2), 226–230.

ZHANG, H., (2004). The Optimality of Naive Bayes. 17. FLAIRS Conference 2004: Miami Beach, Florida, USA 562–567.