Computing and Informatics, Vol. 35, 2016, 615–634

EVOLUTIONARILY TUNED GENERALIZED PSEUDO-INVERSE IN LINEAR DISCRIMINANT ANALYSIS

Tomasz Górecki

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

Maciej Łuczak

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

Abstract. Linear Discriminant Analysis (LDA) and the related Fisher's linear discriminant are very important techniques used for classification and for dimensionality reduction. A certain complication occurs in applying these methods to real data. We have to estimate the class means and common covariance matrix, which are not known. A problem arises if the number of features exceeds the number of observations. In this case the estimate of the covariance matrix does not have full rank, and so cannot be inverted. There are a number of ways to deal with this problem. In our previous paper, we proposed improving LDA in this area, and we presented a new approach which uses a generalization of the Moore–Penrose (MP) pseudo-inverse to remove this weakness. However, for data sets with a larger number of features, our method was computationally too slow to achieve good results. Now we propose a model selection method with a genetic algorithm to solve this problem. Experimental results on different data sets demonstrate that the improvement is efficient.

Keywords: Linear discriminant analysis, Moore–Penrose pseudo-inverse, genetic algorithm

Mathematics Subject Classification 2010: 62-H30

1 INTRODUCTION

Linear Discrimination Analysis is still being intensively developed by many researchers and has recently been extended to various generalized LDA methods that can be applied [13, 34, 37, 33, 36]. This method is also widely used in practice, e.g. in face/image recognition [27, 35, 10], medicine [17, 24], genetics [15], biometry [4, 26, 23, 28], and many other areas. LDA is also often used as a component of a (much) more complex classifier [19, 16, 11, 2]. Additionally, LDA is a very popular and effective technique for feature extraction (dimensionality reduction), which is important in data mining and machine learning [3, 25, 21].

Although relying on strong assumptions (multivariate normal distributions, equal covariance matrices in groups) which do not hold in many real problems, LDA has been proved to be effective [18]. This is mainly because LDA is a simple, linear model that is quite robust against noise, and most likely will not overfit. The linear discriminant function is a linear combination of the measured variables, being easy to interpret. Classical LDA involves a sample covariance matrix, required to be nonsingular. However, in many real-world applications, such as text mining, microarray data classification or image recognition, the number of training examples is too small relative to the number of dimensions. Under such circumstances, the covariance matrix is singular and cannot be accurately estimated, and so cannot be inverted. This is known as the singularity (undersampled) problem or small sample size (SSS) problem. One method to deal with this problem is to use a pseudo-inverse instead of the usual matrix inverse [29, 7].

In a previous article [12] we proposed an extension of this approach.

Classically the MP inverse is used to find the inverse of a covariance matrix. We decided to use a specific variant of the generalized MP inverse. We constructed a parametrical family of generalized MP inverses and used it in LDA. Using the cross-validation (leave-one-out) error rates, we chose the best models and combined them by a mean rule. In this way we obtained, in addition to the possibility of working with any data sets (not only of SSS), a substantial decrease in the classification error rate compared with basic LDA.

We showed that this method works well when the number of features is less than 15. In this situation we can check all models and choose the best one. However, for data sets with more than 15 features, our method was computationally too slow (too many models to check) to achieve good results. The problem is too complex to find an exact solution (or it takes too long to calculate the solution exactly). The most feasible approach, then, is to use a heuristic method [20]. A genetic algorithm

(GA) is heuristic, which means that it estimates a solution. Therefore we propose the use of GA to solve our problem. Genetic algorithm is a popular technique commonly used in classification (e.g. [5, 31]). 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 execution technique of the genetic algorithm is not dependent on the error surface, we can solve multi-dimensional, non-differential, non-continuous, and even non-parametrical problems. It is a method which is very easy to understand, and it requires practically no mathematical knowledge. GAs have drawbacks too, of course. Certain optimization problems (called variant problems) cannot be solved by means of GAs. There is no absolute assurance that a GA will find a global optimum. Like other artificial intelligence techniques, GA cannot guarantee constant optimization response times.

In this paper, we first present the main ideas of LDA (Section 2). In the same section we describe generalized inverses of matrices. At the end of that section we explain our previous idea of extended LDA and present our new genetic approach. In this paper the performances of the described methods are compared and the error of classification is analyzed. The methods and data sets used are described in Section 3. The results of the research are explained using graphs showing the differences between the classifiers. Section 4 contains the results of our experiments on the described data sets, as well as statistical analysis of the results. We conclude with discussion in Section 5.

2 METHODS

Let P be a data set (population) consisting of K classes (subpopulations, groups) denoted by $G_1, \ldots, G_K$. The classes are disjoint subsets and cover the whole data set P. This means that every element $\boldsymbol{x}$ of P belongs to exactly one of the classes. From the data set P we sample a subset called the training data set. The training data consist of pairs $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ is a feature vector in a d-dimensional space and y_i is a label corresponding to one of the classes $G_1, \ldots, G_K$. Suppose we have a new item $\boldsymbol{x}$ from the population P. The goal of (supervised) classification is to find a label y for the vector $\boldsymbol{x}$ using only information from the training data. If a predicted label is incorrect, we say that an error occurs. A classifier is a method for predicting the membership of an unclassified feature vector $\boldsymbol{x} \in P$ in one of the classes $G_1, \ldots, G_K$. A classifier can be viewed as a rule for estimating the posterior probability of membership in a class G_k. A reasonable classification strategy is to assign $\boldsymbol{x}$ to the class with the highest posterior probability. This strategy is called the Bayes' rule classifier. We denote the posterior probability of membership in G_k by

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

The probability that a randomly selected observation belongs to class G_k is called a prior probability, and denoted by π_k. Let $f_k(\boldsymbol{x})$ be the conditional multivariate probability density for the k^{th} class. There is no requirement that the densities have to be continuous; they may be discrete or be finite mixture distributions, or even have singular covariance matrices. The posterior probability $p_k(\boldsymbol{x})$ that the observed vector $\boldsymbol{x}$ belongs to the class G_k follows from Bayes' theorem, and can be written as

$$p_k(\boldsymbol{x}) = \frac{f_k(\boldsymbol{x})\pi_k}{\sum_{i=1}^{K} f_i(\boldsymbol{x})\pi_i}. \tag{2}$$

If two (or more) classes have the same posterior probabilities for a vector $\boldsymbol{x}$, then a random assignment is used.

2.1 Linear Discriminant Analysis

Using Bayes' rule directly is impractical. To obtain the densities $f_k(\boldsymbol{x})$ we would need very much data to obtain the relative frequencies of all groups for each measurement. It is more convenient to assume that we know the distribution and obtain the probability theoretically. So we now make the Bayes' rule classifier more specific. Let all probability densities be multivariate Gaussian (normal) with mean vectors $\boldsymbol{\mu}_k$ and a common covariance matrix $\boldsymbol{\Sigma}$. Thus f_k is an $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ density with the equation

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}. \tag{3}$$

Under the above assumptions we can write a linear Bayesian classifier as

$$d_B(\boldsymbol{x}) = \arg\max_k \delta_k(\boldsymbol{x}), \tag{4}$$

where

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln\pi_k \tag{5}$$

is a linear discriminant function (details in [32]). We assign a vector $\boldsymbol{x}$ to the class G_k for which the value of $\delta_k(\boldsymbol{x})$ is maximal.

In practice, the class means $\boldsymbol{\mu}_k$ and the covariance matrix $\boldsymbol{\Sigma}$ are not known. We can estimate them from the training data. Usually the maximum likelihood (plug-in) estimate may be used in place of the exact value in the above equations. Although the estimates of the covariance may be considered optimal in some sense, this does not mean that the resulting discriminant obtained by substituting these values is optimal in any sense, even if the assumption of normally distributed classes is correct [1]. Also, any sensible Bayesian rule will not lead to this approach, except either asymptotically or under very restrictive conditions [8]. Additionally, we have to

estimate a priori probabilities. These are usually estimated simply by the empirical frequencies of observations in the training set.

2.2 Algorithm

The regular matrix inverse $\boldsymbol{A}^{-1}$ or the Moore–Penrose inverse $\boldsymbol{A}^{\dagger}$ is a standard method used to compute an inverse of the covariance matrix (see Equation (3)). In [12] another generalized pseudo-inverse was introduced and used in Linear Discriminant Analysis. For the convenience of readers we recall the definitions below.

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

$$\boldsymbol{A}^{\dagger}_{\boldsymbol{MN}} = \hat{\boldsymbol{N}}^{-1} \left(\hat{\boldsymbol{M}} \boldsymbol{A} \hat{\boldsymbol{N}}^{-1} \right)^{+} \hat{\boldsymbol{M}} \tag{6}$$

satisfies the condition

$$\begin{aligned} &\|\boldsymbol{A}^{\dagger}_{\boldsymbol{MN}} \boldsymbol{y}\|_N \leq \|\boldsymbol{x}\|_N \\ &\forall\, \boldsymbol{x} \in \{\boldsymbol{x} \colon \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_M \leq \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{y}\|_M \,\forall\, \boldsymbol{z} \in \mathbb{R}^n\}\,, \end{aligned}$$

where $\|\boldsymbol{x}\|_N = \sqrt{\boldsymbol{x}^T \boldsymbol{N} \boldsymbol{x}}$ and $\|\boldsymbol{y}\|_M = \sqrt{\boldsymbol{y}^T \boldsymbol{M} \boldsymbol{y}}$ are norms in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. $\boldsymbol{A}^{\dagger}_{\boldsymbol{MN}}$ is referred to as the minimum $\boldsymbol{N}$-norm $\boldsymbol{M}$-least-squares g-inverse of $\boldsymbol{A}$. When $\boldsymbol{M}, \boldsymbol{N}$ are identity matrices, we use the notation $\boldsymbol{A}^{\dagger}$ and call it the Moore–Penrose inverse (pseudo-inverse). For a larger survey and more details we refer readers to [22].

If $\boldsymbol{M}$ is positive semi-definite, then $\|\boldsymbol{y}\|_M$ is a seminorm (i.e. $\|\boldsymbol{y}\|_M$ can be zero for nonzero $\boldsymbol{y}$) and the right side of Equation (6) does not need to be a g-inverse. We denote this by $\boldsymbol{A}^{*}_{\boldsymbol{MN}}$, and by $\boldsymbol{A}^{*}_{\boldsymbol{M}}$ if $\boldsymbol{N} = \boldsymbol{I}$.

We use $\boldsymbol{A}^{*}_{\boldsymbol{M}}$ with a special form of matrix $\boldsymbol{M}$. More precisely, we use Equation (6) with the assumptions

$$\hat{\boldsymbol{N}} = \boldsymbol{N} = \boldsymbol{I}, \quad \hat{\boldsymbol{M}} = \boldsymbol{M} = \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_m \end{bmatrix} \tag{7}$$

where $a_i = 0$ or 1 for $i = 1, \dots, m$. This leads to the seminorm

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}} = \sqrt{x_{j_1}^2 + x_{j_2}^2 + \dots + x_{j_k}^2},\ 1 \leq k \leq m$$

for $\boldsymbol{x} = (x_1, x_2, \dots, x_m) \in \mathbb{R}^m$ ($a_i x_i^2 = 0$ for $a_i = 0$, $j_s = i$ for $a_i = 1$). Then Equation (6) assumes the form

$$\boldsymbol{A}^{*}_{\boldsymbol{M}} = (\boldsymbol{M}\boldsymbol{A})^{\dagger} \boldsymbol{M}. \tag{8}$$

Thus we can use $\mathbf{\Sigma}^*_{\boldsymbol{M}}$ instead of $\mathbf{\Sigma}^{-1}$ to compute the inverse of the covariance matrix $\mathbf{\Sigma}$ (see Equation (3)). Note that we do not have to compute the determinant of the covariance matrix $\mathbf{\Sigma}$ to obtain posterior probabilities (see Equation (2)).

We take only ones and zeros in the diagonal of the matrix $\boldsymbol{M}$, because it has been proved [12] that the value of $\boldsymbol{A}^*_{\boldsymbol{M}}$ depends only on whether the coefficients a_i are zero or nonzero. We study two algorithms for choosing ones and zeros in the diagonal of the matrix $\boldsymbol{M}$. In the first one (ALG1) we pass through all combinations of ones and zeros in the diagonal of matrix $\boldsymbol{M}$. In the second one (ALG2) we take only diagonals with at most one zero. Then, in both cases, we choose the linear discriminant models with the lowest cross-validation (leave-one-out) error rates.

In [12], for data sets with more than 15 features we used only ALG2, because of the computational complexity of ALG1. In this paper we use a genetic algorithm to find the best (or almost the best) diagonal of matrix $\boldsymbol{M}$ (among all possible diagonals). Thanks to this genetic approach we can handle data sets with a large number of features (many more than 15).

2.3 Genetic Algorithm

The main scheme of the genetic algorithm is shown in Figure 1. The population consists of individuals (genotypes) which are diagonals of the matrix $\boldsymbol{M}$. Each individual is a vector of positions (genes) that correspond to numbers (ones or zeros) in the diagonal of $\boldsymbol{M}$. All populations in the algorithm have a constant number n of individuals.

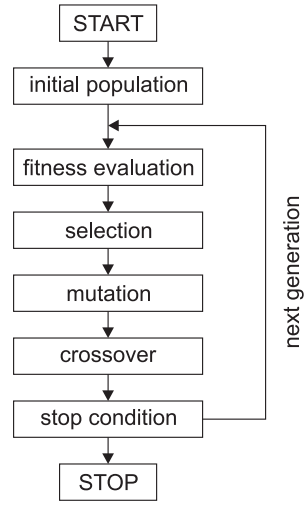


Figure 1. The main scheme of the genetic algorithm

Initial population: This is generated randomly. We construct n individuals such that each position in the vector (diagonal) can be 0 or 1 with probabilities of 0.5.

Fitness evaluation: The fitness function value is computed by the leave-one-out crossvalidation (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 involves changing (0 or 1) in the relevant position (Figure 2). This 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 for 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 that point are exchanged with one another (Figure 2). The operation is repeated an appropriate number of times to make a new population of size n.

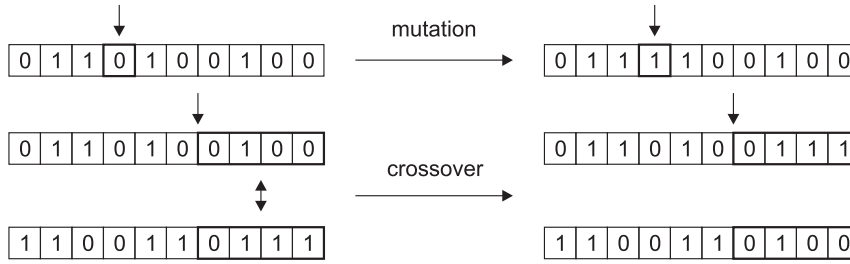


Figure 2. Reproduction of individuals in a population in the genetic algorithm. Mutation (top) of one individual, and crossover (bottom) of two individuals.

Stop condition: We do not use a fixed number of generations in the genetic algorithm. 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 is satisfied. The stop condition depends on the behavior of the mean fitness value in the populations over k steps of the algorithm. If in k steps the mean is not 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

3.1 Data Sets

We performed experiments on 5 data sets with fewer than 15 features and 10 data sets with more than 15 features. The data sets were chosen in such a way that they had different numbers of features of particular types and different numbers of examples; also there were some data sets with two-class distribution and some with more than two classes. In Table 1 the characteristics of the data sets are given, showing the variety of training set sizes, numbers of classes, and dimensionalities. All data sets come from the UCI Machine Learning Repository [9].

Name of data set	Number of features	Number of classes	Number of instances
german	24	2	1 000
hepatitis	16	2	137
ionosphere	33	2	351
libras	91	15	360
lungcancer	55	3	32
musk	166	2	476
sonar	60	2	208
spectf	44	2	267
vote	16	2	300
wave	21	3	125
breast_w	9	2	683
glass	9	6	214
hartC	13	5	297
heartS	10	5	105
vowel	10	11	990

Table 1. Information about the data sets used

3.2 Experimental Setup

The classification errors were estimated by the cross-validation (leave-one-out) and bootstrap methods. Leave-one-out was used to train the model, i.e. to find the "best" diagonals (those with the smallest error rates) of the matrix $\boldsymbol{M}$. This method was used to compute the value of the fitness function in the genetic algorithm. 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 used tournament selection. Different stop condition numbers were tried, $k = 0, \ldots, 10$. For the final result of our method we took the best case $k = 10$. In the next step, we tested the model. The mean classifier was performed for models with each of these "best" diagonals. We calculated the bootstrap classification error rate (1 000 repetitions).

For each data set we repeated the algorithm 10 times. We finally fixed as the error rate of our method the mean of these bootstrap error rates. For all methods being compared, the same bootstrap samples were used.

The algorithms ALG1 and ALG2 used in the experiments are the versions ALG1-3 and ALG2-3 from [12].

In the computational process we used the PRTools 4.2.4 program (`http://www.prtools.org`). This is a Matlab (version R2011a) based toolbox for pattern recognition [30]. In each procedure we used the default parameters.

4 RESULTS

4.1 Classification Error Rates

The results of the process are given in Table 2. For our method, ALGG, the CV error rate among all (10) runs of the genetic algorithm is presented. This is compared with the CV error rates of the LDA and ALG2 methods. The next columns contain the bootstrap error rates of the methods. The last columns show the relative errors of our method calculated by $\frac{\text{ALGG}-\text{Alg}}{\text{Alg}}$, where ALGG is the bootstrap error of our method, and Alg can be the bootstrap error of LDA or ALG2.

We can see that in the training phase, the genetic algorithm finds the best diagonal (the smallest CV error) on almost all data sets (excluding *glass*). Comparing the algorithms in the testing phase (bootstrap errors), ALGG is the best out of all the methods 10 times out of 15. In terms of means of relative error rates for all data sets, ALGG is 6.68 % better than LDA and 5.19 % better than ALG2.

The computation time of the compared algorithms is directly proportional to the number of computations of pseudo-inverses $\boldsymbol{A}_{\boldsymbol{M}}^{*}$. This means that computation time depends on the number of diagonals of the matrix $\boldsymbol{M}$ used by the algorithms in the learning phase (cross-validation). The number of diagonals is $c+1$ for ALG2, 2^c for ALG1, where c is the number of features of the data set, and $20 \cdot g$ for ALGG, where 20 is the number of individuals in the population of the genetic algorithm and g is the average number of generations. A comparison of the numbers of diagonals for algorithms ALG1 and ALGG is given in Table 3. We can see that the number of potential pseudo-inverse computations is extremely large in the case of ALG1. For this reason, we are not able to compute ALG1 for the "large" data sets (more than 15 features) listed in Table 1.

For "small" data sets (fewer than 15 features) we can also compare our method with ALG1, which is a method covering all possible models (diagonals). A detailed comparison of CV error rates is contained in Table 4. The genetic algorithm ALGG, as a random method, does not always find the best solution. The mean of relative errors compared with ALG1 is slightly higher than zero. However, considering minimal errors, we see that for every data set there are runs of the genetic algorithm that equal the optimal error (ALG1). In fact, for the *vowel* data set, all runs find the best optimal error. Comparing with ALG2 we can clearly see that the genetic algorithm is better. The only exception, for the *glass* data set, is because in this

Name of Data Set	CV Error Rate LDA	ALG2	ALGG	Bootstrap Error Rate LDA	ALG2	ALGG	Relative Bootstrap Error Rate $\frac{ALGG-LDA}{LDA}$	$\frac{ALGG-ALG2}{ALG2}$
german	23.20	22.60	**22.20**	24.16	**23.78**	24.25	0.35	1.97
hepatitis	13.14	12.41	**10.00**	15.41	**14.89**	18.77	21.79	26.05
ionosphere	13.68	11.68	**9.86**	14.17	13.68	**12.56**	−11.36	−8.22
libras	33.89	32.50	**25.67**	41.11	40.74	**33.83**	−17.71	−16.96
lungcancer	53.13	46.88	**17.19**	**49.94**	52.57	52.85	5.81	0.52
musk	18.70	17.23	**14.14**	26.40	23.95	**20.80**	−21.22	−13.18
sonar	24.52	22.60	**16.06**	28.59	27.83	**25.99**	−9.10	−6.62
spectf	25.09	23.22	**17.19**	26.95	26.67	**20.94**	−22.29	−21.47
vote	6.33	5.33	**4.77**	6.04	5.83	**5.81**	−3.80	−0.28
wave	24.80	23.20	**13.76**	31.80	30.96	**20.78**	−34.66	−32.88
glass	35.05	**34.11**	34.63	**38.47**	39.46	38.81	0.87	−1.67
breast_w	3.95	3.81	**3.50**	4.03	3.96	**3.86**	−4.32	−2.68
hart_c	40.07	39.06	**38.82**	42.04	**41.58**	41.68	−0.85	0.24
heart_s	57.14	56.19	**54.67**	62.41	61.45	**60.12**	−3.67	−2.17
vowel	**45.25**	**45.25**	**45.25**	**46.52**	46.77	**46.52**	0.00	−0.53
MEAN							−6.68	−5.19

Table 2. Cross-validation and bootstrap error rates on all data sets. The best results in each group of error rates are bolded.

Name of Data Set	#features	#generations (mean)	#diagonals to compute ALG1	ALGG
german	24	45.3	16 777 216	906
hepatitis	16	34.9	65 536	698
ionosphere	33	52.1	8 589 934 592	1 042
libras	91	37.0	$2.47 \cdot 10^{27}$	740
lungcancer	55	36.1	$3.60 \cdot 10^{16}$	722
musk	166	39.5	$9.35 \cdot 10^{49}$	790
sonar	60	46.9	$1.15 \cdot 10^{18}$	938
spectf	44	45.8	$1.75 \cdot 10^{13}$	916
vote	16	30.4	65 536	608
wave	21	38.4	2 097 152	768
glass	9	28.9	512	578
breast_w	9	29.6	512	592
hart_c	13	32.4	8 192	648
heart_s	10	31.7	1 024	634
vowel	10	34.6	1 024	692

Table 3. Performance of algorithms ALG1 and ALGG. Computation time is directly proportional to the number of diagonals.

	CV Error Rate									Relative CV Error Rate		
Name				ALGG			#generations					
of Data Set	LDA	ALG2	ALG1	max	mean	min	max	mean	min	$\frac{ALGG-LDA}{LDA}$	$\frac{ALGG-ALG2}{ALG2}$	$\frac{ALGG-ALG1}{ALG1}$
glass	35.05	**34.11**	**34.11**	35.51	34.63	**34.11**	46	29	19	−1.20	1.51	1.51
breast_w	3.95	3.81	**3.22**	3.81	3.50	**3.22**	47	30	19	−11.48	−8.08	8.64
hart_c	40.07	39.06	**38.05**	40.07	38.82	**38.05**	65	32	21	−3.11	−0.61	2.03
heart_s	57.14	56.19	**54.29**	56.19	54.67	**54.29**	47	32	19	−4.33	−2.71	0.69
vowel	**45.25**	**45.25**	**45.25**	**45.25**	**45.25**	**45.25**	53	35	19	0.00	0.00	0.00
MEAN								31		−4.02	−1.98	2.57

Table 4. Cross-validation error rates on "small" data sets (fewer than 15 features). The best results are bolded.

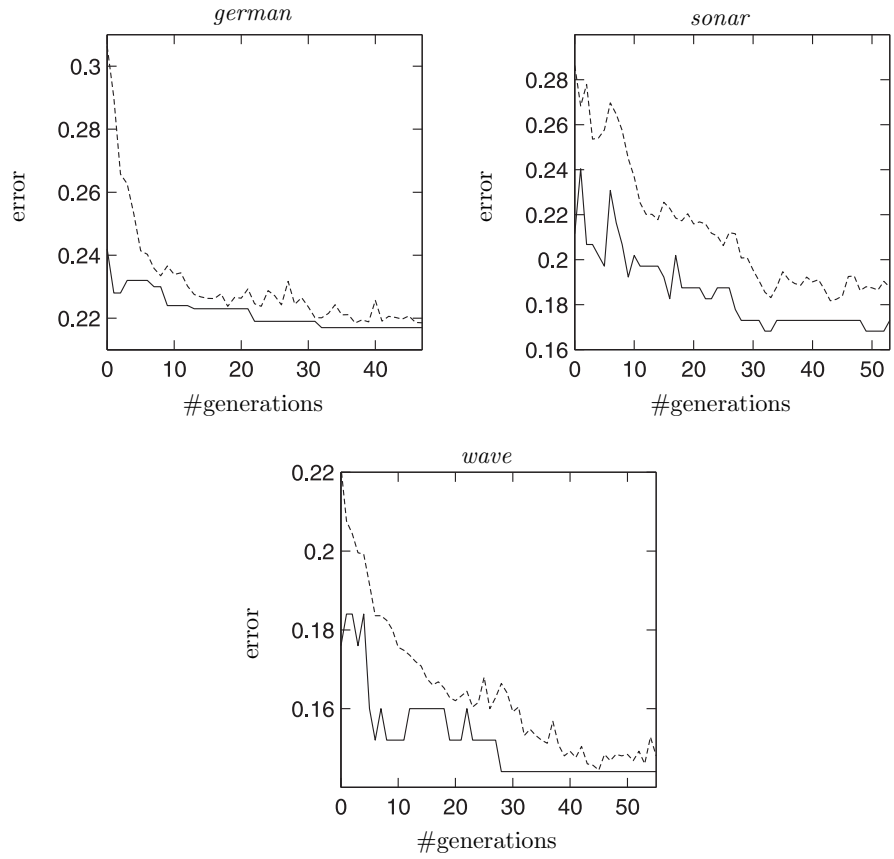


Figure 3. Runs of the genetic algorithm for the "large" (more than 15 features) example data sets. Fitness function value (mean $(\cdots)$ and minimum (–) of CV error rate) depending on the number of generations. From left: *german*, *sonar*, *wave*.

case even ALG2 reaches the optimal error; similarly for the *vowel* data set, where all algorithms find the optimal error. We can say that the behavior of the genetic algorithm is typical. It finds results that are close to the best one, and sometimes even the optimal one (ALG1). When possible, the genetic algorithm is distinctly better than the ALG2 method.

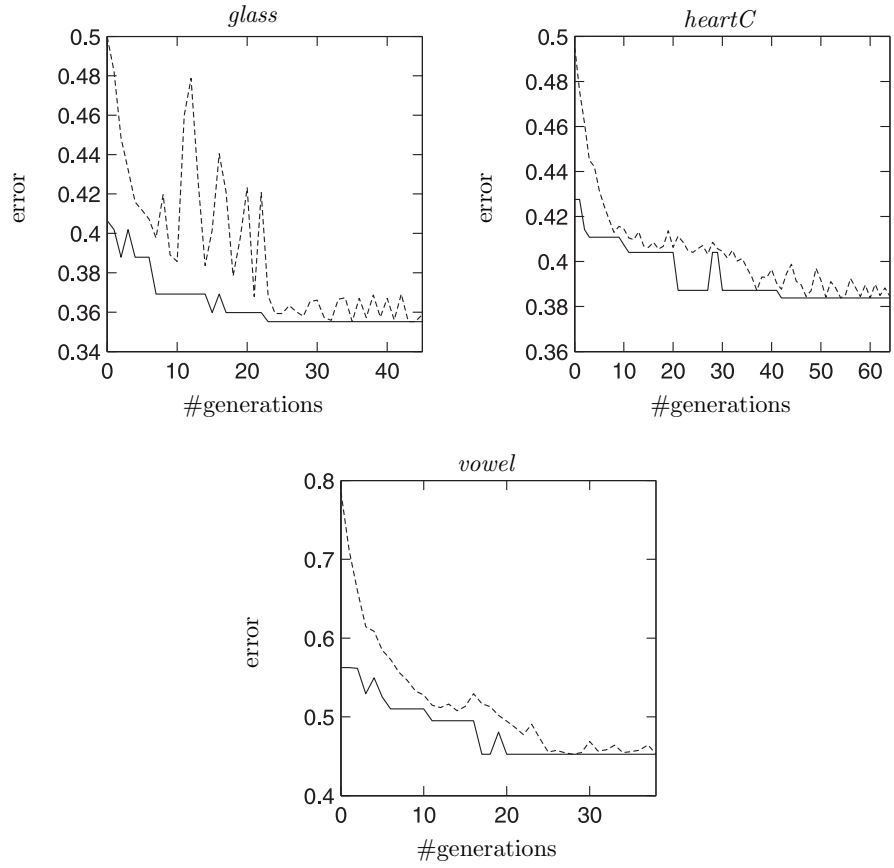
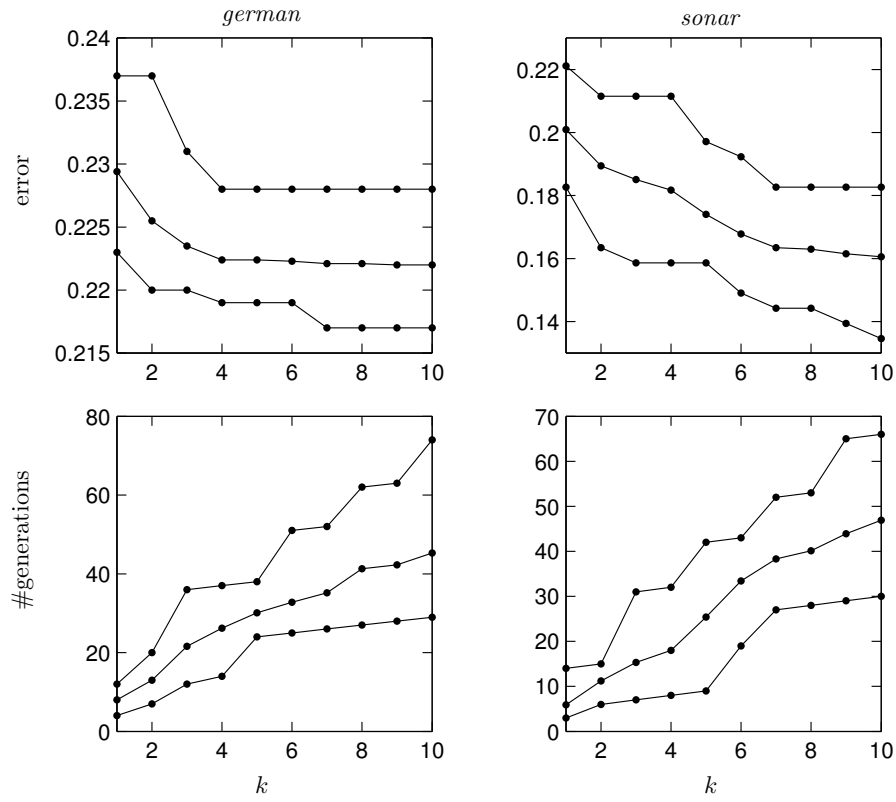


Figure 4. Runs of the genetic algorithm for the "small" (fewer than 15 features) example data sets. Fitness function value (mean $(\cdots)$ and minimum $(-)$ of CV error rate) depending on the number of generations. From left: *glass*, *heartC*, *vowel*.

Graphs of example runs of our algorithm are shown in Figure 3 and Figure 4. We can observe the rather normal behavior of the genetic algorithm. The tournament selection used is not an elitist selection method, 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 some number of generations.

4.2 Stop Condition Analysis



The results in Tables 2 and 4 are shown for the stop condition number $k = 10$, i.e. for the best case, where the algorithm runs for the longest time. It is interesting how the error rate and the number of generations change if we take a lower value of k. We performed experiments for $k = 0, \ldots, 10$. Example graphs are shown in Figure 5 ("large" data sets) and Figure 6 ("small" data sets). The upper graph shows error rates (max, mean, min) depending on the stop condition number k. The lower one shows the number of generations (max, mean, min) depending on k. For the *wave* data set we see that from $k = 4$ the mean error stops decreasing, while for the minimum error this is true even for $k = 2$. On the other hand the number of generations increases practically linearly from $k = 4$. The error of the data set *german* stabilizes from $k = 7$. For the *sonar* data set we see that we could obtain even lower errors with $k > 10$. We have only considered an approximately linear increase in iterations of the genetic algorithm. The graphs for "small" data sets are similar. The errors stabilize faster, sometimes even with min = mean = max, as for the data set *vowel*.

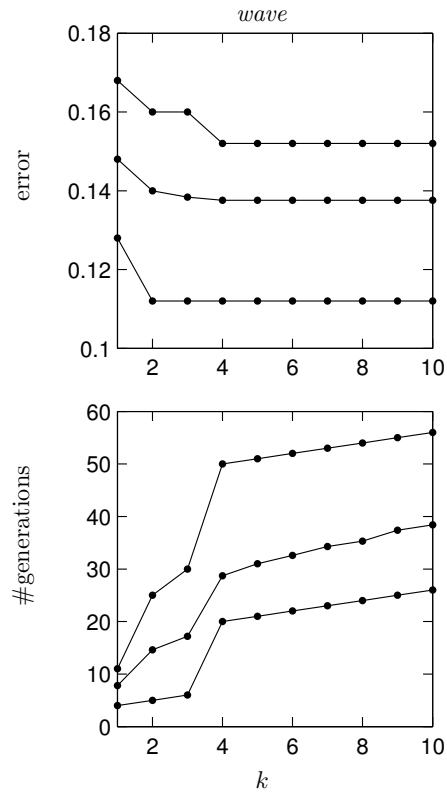
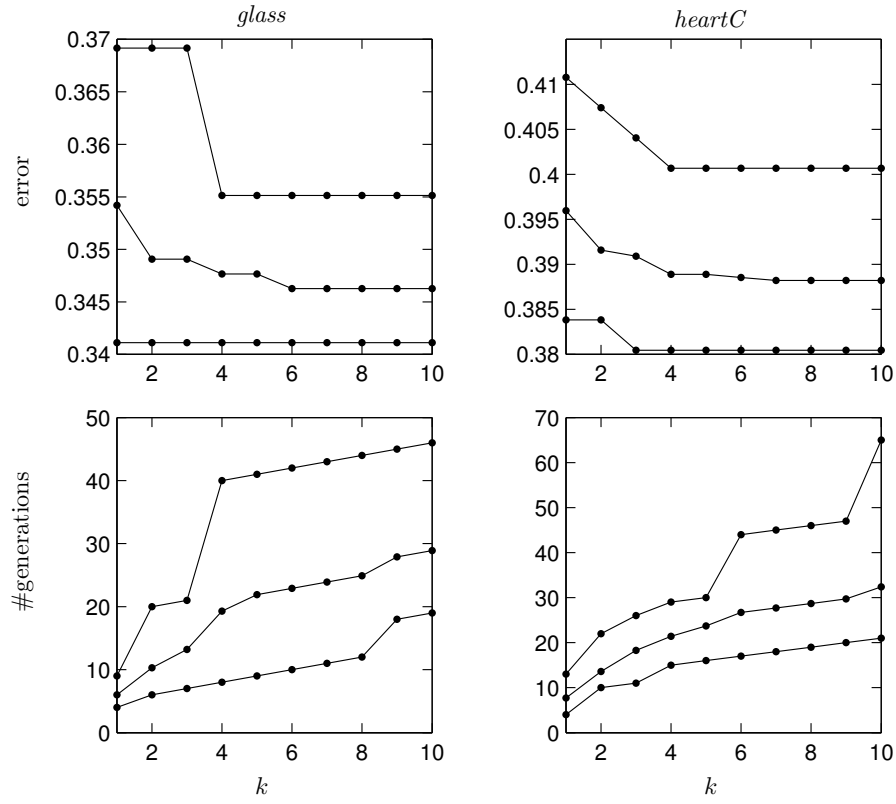


Figure 5. Error rates (max, mean, min) depending on the stop condition number k (top), and the number of generations (max, mean, min) depending on the stop condition number k (bottom), for "large" data sets (more than 15 features). From left: *german*, *sonar*, *wave*.

In Figure 7 a summary of the relationship is given. It shows the mean of relative errors (top) and the mean of the number of generations (bottom), depending on the stop condition number k, for all data sets (left-hand side – "small", right-hand side – "large"). It is clearly seen that the error rate curve becomes saturated for higher values of k, but the number of iterations (generations) increases practically linearly. For example, for "large" data sets, we can observe that a decrease in k to about 7 or 8 results in a very small decrease in the error rate (from $-18\,\%$ to about $-19\,\%$). On the other hand the number of iterations increases significantly (from 30 to about 40), and this influences the computation time of the algorithm. For "small" data sets this effect is even more clearly visible.



4.3 Statistical Comparison of Classifiers

Finally, to confirm that the new ALGG method is superior to ALG2, we present a statistical comparison of their bootstrap error rates on all 15 data sets.

To statistically compare two classifiers over multiple data sets, [6] recommends the Wilcoxon signed-ranks test. The Wilcoxon signed-ranks test is a non-parametric alternative to the paired t-test, which ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences.

Let d_i be the difference between the performance scores of the two classifiers on the i^{th} out of N data sets (15 in our comparison). The differences are ranked according to their absolute values (average ranks are assigned in case of ties). Let R^+ be the sum of ranks for the data sets on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite case. Ranks of $d_i = 0$ are split evenly among the sums (if there are an odd number of them, one is ignored):

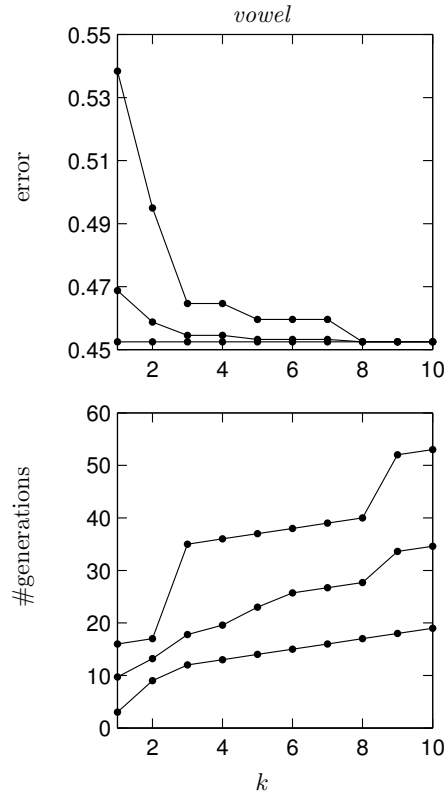


Figure 6. Error rates (max, mean, min) depending on the stop condition number k (top), and the number of generations (max, mean, min) depending on the stop condition number k (bottom), for "small" data sets (fewer than 15 features). From left: *glass*, *heartC*, *vowel*.

$$R^+ = \sum_{d_i>0} \operatorname{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \operatorname{rank}(d_i),$$

$$R^- = \sum_{d_i<0} \operatorname{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \operatorname{rank}(d_i).$$

Let T be the smaller of the sums:

$$T = \min(R^+, R^-).$$

As the number of data sets N increases, the distribution of T tends towards a normal distribution. We reject the hypothesis that classifiers do not differ if $T = 25$ is less

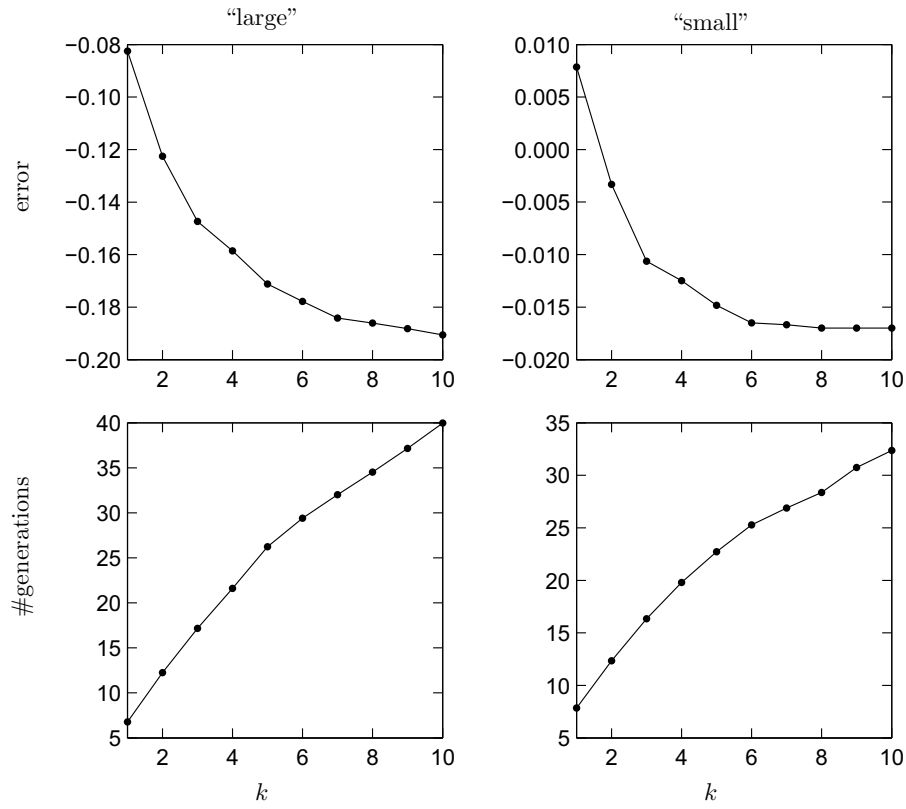


Figure 7. Mean of relative errors (top) and mean of the number of generations (bottom), depending on the stop condition number k, for all data sets (left-hand side – "large", right-hand side – "small").

than or equal to the critical value (25). In our case we obtain a p-value equal to 0.0479. We see that ALGG is significantly better than ALG2 at a significance level of $\alpha = 0.05$.

5 CONCLUSIONS

In this paper we have introduced and studied a new approach (a genetic approach) to the use of a generalization of the Moore–Penrose pseudo-inverse of a matrix in the LDA method of object classification. Our study showed that this method leads to very good results on "large" data sets (having from 15 to over 100 features). Our technique outperforms LDA, as well as our previous, classical, approach. The major disadvantage of our previous method was high computational complexity for "large" data sets. The new method removes this inconvenience.

Due to the high nonlinearity and complicated dynamics in genetic algorithms, the method does not easily lead to a rigorous theoretical analysis. However, the experiments that we have conducted provide evidence of the potential and usefulness of our method.

Of course, the classification performance of the new algorithm needs to be further evaluated, considering additional real and artificial data. In our technique we can use methods of combining classifier ensembles other than the mean method. This will be a topic of our future research.

REFERENCES

[1] Anderson, T.: An Introduction to Multivariate Analysis. Wiley, 1984.

[2] Chau, A. L.—Li, X.—Yu, W.: Support Vector Machine Classification for Large Datasets Using Decision Tree and Fisher Linear Discriminant. Future Generation Computer Systems, Vol. 36, 2014, No. 7, pp. 57–65.

[3] Chen, S.—Li, D.: Modified Linear Discriminant Analysis. Pattern Recognition, Vol. 38, 2005, No. 3, pp. 441–443.

[4] Cozzolino, D.—Restaino, E.—Fassio, A.: Discrimination of Yerba Mate (Ilex Paraguayensis St. Hil.) Samples According to Their Geographical Origin by Means of Near Infrared Spectroscopy and Multivariate Analysis. Sensing and Instrumentation for Food Quality and Safety, Vol. 4, 2002, No. 2, pp. 67–72.

[5] Dehuri, S.—Mall, R.: Predictive and Comprehensible Rule Discovery Using a Multi-Objective Genetic Algorithm. Knowledge Based Systems, Vol. 19, 2006, pp. 413–421.

[6] Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, Vol. 7, 2006, pp. 1–30.

[7] Duda R.—Hart P.—Stork D.: Pattern Classification. Wiley, 2000.

[8] Enis, P.—Geisser, S.: Optimal Predictive Linear Discriminants. Annals of Statistics, Vol. 1, 1986, No. 2, pp. 403–410.

[9] Frank, A.—Asuncion, A.: UCI Machine Learning Repository. School of Information and Computer Science, University of California, Irvine, CA, 2000. `http://archive.ics.uci.edu/ml`.

[10] Gao, Q.—Liu, J.—Zhang, H.—Hou, J.—Yang, X.: Enhanced Fisher Discriminant Criterion for Image Recognition. Pattern Recognition, Vol. 45, 2012, No. 10, pp. 3717–3724.

[11] Górecki, T.: Sequential Correction of Linear Classifiers. Journal of Applied Statistics, Vol. 40, 2013, No. 4, pp. 763–776.

[12] Górecki, T.—Łuczak, M.: Linear Discriminant Analysis with a Generalization of the Moore–Penrose Pseudoinverse. International Journal of Applied Mathematics and Computer Science, Vol. 23, 2013, No. 2, pp. 463–471.

[13] Hubert, M.—van Driessen, K.: Fast and Robust Discriminant Analysis. Computational Statistics and Data Analysis, Vol. 45, 2004, No. 2, pp. 301–320.

[14] Iman, R.—Davenport, J.: Approximations of the Critical Region of the Friedman Statistic. Communications in Statistics – Theory and Methods, Vol. 9, 1980, No. 6, pp. 571–595.

[15] Jin, J.—An, J.: Robust Discriminant Analysis and Its Application to Identify Protein Coding Regions of Rice Genes. Mathematical Biosciences, Vol. 232, 2011, No. 2, pp. 96–100.

[16] Kim, H.—Drake, B.—Park, H.: Multiclass Classifiers Based on Dimension Reduction with Generalized LDA. Pattern Recognition, Vol. 40, 2007, No. 11, pp. 2939–2945.

[17] Kwak, N.—Kim, S.—Lee, C.—Choi, T.: An Application of Linear Programming Discriminant Analysis to Classifying and Predicting the Symptomatic Status of HIV/AIDS Patients. Journal of Medical Systems, Vol. 26, 2002, No. 5, pp. 427–438.

[18] Lim, T.—Loh, W.—Shih, Y.: A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms. Machine Learning, Vol. 40, 2000, No. 3, pp. 203–228.

[19] Lin, C.—Chen, A.: Fuzzy Discriminant Analysis with Outlier Detection by Genetic Algorithm. Computers and Operations Research, Vol. 31, 2004, No. 6, pp. 877–888.

[20] Michalewicz Z.—Fogel, D. B.: How to Solve It: Modern Heuristics. Springer, 2004.

[21] Peng, H.—Jiang, X.—Fang, X.—Liu, J.: Variable Selection for Fisher Linear Discriminant Analysis Using the Modified Sequential Backward Selection Algorithm for the Microarray Data. Applied Mathematics and Computation, Vol. 238, 2014, No. 7, pp. 132–140.

[22] Rao, C. R.—Mitra, S. K.: Generalized Inverse of Matrices and Its Applications. Wiley, 1971.

[23] Santos, V. S.—Nardini, V.—Cunha Jr., L. C.—Barbosa Jr., F.—Teixeira, G. H.: Identification of Species of the Euterpe Genus by Rare Earth Elements Using Inductively Coupled Plasma Mass Spectrometry and Linear Discriminant Analysis. Food Chemistry, Vol. 153, 2014, No. 6, pp. 334–339.

[24] Satheeshkumar, P. S.—Balan, A.: Linear Discriminant Analysis for Differentiation of Odontogenic Cyst and Tumours in Computerised Tomographic Findings, Vol. 47, 2011, pp. S95.

[25] Shin, Y.—Park, C.: Analysis of Correlation Based Dimension Reduction Methods. International Journal of Applied Mathematics and Computer Science, Vol. 21, 2011, No. 3, pp. 549–558.

[26] Sikorska, E.—Chmielewski, J.—Górecki, T.—Khmelinskii, I.—Sikorski, M.—De Keukeleire, D.: Discrimination of Beer Flavours by Analysis of Volatiles Using the Mass Spectrometer as an Electronic Nose. Journal of the Institute of Brewing, Vol. 113, 2007, No. 1, pp. 110–116.

[27] Song, F.—Zhang, D.—Chen, Q.—Wang, J.: Face Recognition Based on a Novel Linear Discriminant Criterion. Pattern Analysis and Applications, Vol. 10, 2007, No. 3, pp. 165–174.

[28] Takizawa, K.—Nakano, K.—Ohashi, S.—Yoshizawa, H.—Wang, J.—Sasaki, Y.: Development of Nondestructive Technique for Detecting Internal De-

fects in Japanese Radishes. Journal of Food Engineering, Vol. 126, 2014, No. 4, pp. 43–47.

[29] Tian, Q.—Fainman, Y.—Lee, S. H.: Comparison of Statistical Pattern-Recognition Algorithms for Hybrid Processing. Journal of the Optical Society of America A, Vol. 5, 1988, No. 10, pp. 1670–1682.

[30] van der Heijden, F.—Duin, R.—de Ridder, D.—Tax, D.: Classification, Parameter Estimation and State Estimation: An Engineering Approach Using Matlab. Wiley, 2004.

[31] Vivekanandan, P.—Rajalakshmi, R.—Nedunchezhian, R.: An Intelligent Genetic Algorithm for Mining Classification Rules in Large Datasets. Computing and Informatics, Vol. 32, 2013, No. 1, pp. 1–22.

[32] Weeb, A.: Statistical Pattern Recognition. Wiley, 2002.

[33] Villegas, M.—Paredes, R.: On Improving Robustness of LDA and SRDA by Using Tangent Vectors. Pattern Recognition Letters, Vol. 34, 2013, No. 9, pp. 1094–1100.

[34] Xu, Y.—Yang, J.—Jin, Z.: A Novel Method for Fisher Discriminant Analysis. Pattern Recognition, Vol. 37, 2004, No. 2, pp. 381–384.

[35] Yang, J.—Liu, C.—Yang, J.: What Kind of Color Spaces is Suitable for Color Face Recognition? Neurocomputing, Vol. 73, 2010, No. 10–12, pp. 2140–2146.

[36] Yang, W.—Wu, H.: Regularized Complete Linear Discriminant Analysis. Neurocomputing, Vol. 137, 2014, No. 5, pp. 185–191.

[37] Zhang, X.—Jia, Y.: A Linear Discriminant Analysis Framework Based on Random Subspace for Face Recognition. Pattern Recognition, Vol. 40, 2007, No. 9, pp. 2585–2591.

Tomasz Górecki received his M.Sc. degree in mathematics from the Faculty of Mathematics and Computer Science of Adam Mickiewicz University in Poznań, Poland, in 2001. He also received his Ph.D. degree there in 2005. He is currently Assistant Professor at the same university. His research interests include machine learning, time series classification and data mining.

Maciej Łuczak received his M.Sc. and Ph.D. degrees in mathematics from the Faculty of Mathematics and Computer Science of Adam Mickiewicz University in Poznań, Poland, in 2001 and 2005, respectively. He is currently Assistant Professor at the Faculty of Civil Engineering, Environmental and Geodetic Sciences of Koszalin University of Technology, Poland. His research interests are in the area of machine learning and evolutionary algorithms.