Improving the Qualities of SVM Classifiers with Evolutionary Algorithms

V. Nachev, T. Titova, Ch. Damyanov

Key Words: *Pattern recognition; support vector machines; evolutionary algorithms; feature selection; kernel function selection.*

Abstract. *This paper concerns the use of evolutionary algorithms for improving the support vector machine (SVM) classification. Method genetic algorithms and genetic programming in the design of an SVM in learning stages: feature selection and kernel function selection are summarized. Experimental results of the SVM method application and adaptation to the recognition of unidimensional spectral realization of potatoes have been presented. In this practical non-destructive quality estimation task, the purpose of this work was to achieve maximum objectivity and accuracy by reducing the heuristic approach to feature selection and kernel SVM classifier synthesis.*

Introduction

The first paper that presented the currently familiar form of the Support Vector Machines (SVM) method was published by Vapnik and colleagues (B. Boser and I. Guyon) in 1992. Having started in the 1960s, Vapnik and Chervonenkis' collective work on the statistical theory of training, capacity estimation of class decision function (VCdimension) [1,16,17] and the Generalized Portrait method of optimal linear classifier synthesis played a major role in the method development. It is necessary to point out that this presupposes the linear class separability hypothesis which is rather seldom applied in practice. This condition is comparable to the conditions imposed in the application of the maximum posterior probability principle in Bayesian methods. The problem is eventually solved with the help of the so-called „kernelbased methodology" which maps the input space into a space of considerably higher dimension. The linear separation hyperplane obtained in the induced space via this approach has already become non-linear in the input space, which partially solves the problem with the non-linear classification. The solution is partial since the new space again presupposes linear separability. Among the significant advantages of this method are the good theoretical validity and the existence of statistical methods of assessment as well as criteria for quality evaluation of classifier learning and generalization capacity.

Under these conditions, the successful application of the method to specific applied problems depends mostly on individual parameters which are not defined by the algorithm (as weight factors of the decision function) but by parameters depending on the user (hyperparameters). These parameters are as follows: determining the type of transformation kernel, its parameters, regularization parameter, models for multivariate classification. In this case, the application of evolutionary algorithms aims at finding the above parameters while searching in an extended multidimensional space, which cannot be achieved with heuristic methods or the classical grid search method.

Evolutionary Algorithms

Evolutionary algorithms are optimization algorithms based on mechanisms reflecting the natural process of evolution. At the core of this process lies learning or certain adaptation to the environment, the difference consisting in the fact that natural evolution affects generations (individuals) spanning over more than one lifecycle. As is the case with nature, the infinite process of evolution includes selection, recombination, mutation, and reproduction. The following stages are common to all algorithms of this type:

1. The synthesis of a set X of random solutions to the problem (*population*).

2. The reckoning of the *fitness function* for every solution.

3. The establishment of solutions Y (*parents*) for X to generate new solutions (*offspring*).

4. The generation of new X solutions on the basis of Y (*breeding*).

5. The repetition of the process starting with Stage 2 until a specific number of generations, preset time or a preset value of the target function are reached.

The major conceptual framework and the stages upon which algorithms of this type are based can be used in several, differing spheres of application. The Genetic Algorithms introduced by Holland [7] in the 1960s and the Genetic Programming introduced by Koza [9] in the 1970s have developed as basic strategies.

Genetic Algorithms. Genetic algorithms (GA) work with model parameters expressed as a binary string [6,7]. These parameters are numerical variables, constants, setting parameters, etc. Certain parameters may be missing but the length of the binary string is fixed. Individual binary strings (chromosomes) have two basic properties. The first one is a genotype characterized by a specific sequence and determining a chromosome. The second one is a phenotype which constitutes a decoded version of the genotype and defines the unique traits of the individual. The parameters of each individual in the population are decoded by the chromosome and evaluated with a fitness function. The evolutionary process occurs via the creation of new generations with the help of a crossover operator to combine the chromosomes identified as most suitable. The mutation operator is applied with a moderate frequency - in this case, randomly selected bytes from the binary string have their values altered.

Genetic Programming. In contrast to GA, Genetic Programming (GP) does not aim at determining the parameters of

a preset model generating a solution but rather at determining a so-called program (model, function) [9]. The search goes on in a space made up of models leading to a solution to the problem. Functions and input variables (primitives) are defined in advance. To encode a program, a dendritic structure (a type of tree) is used, where the operators or functions are nonterminals (*figure 1*) whereas the parameters or input data for the program

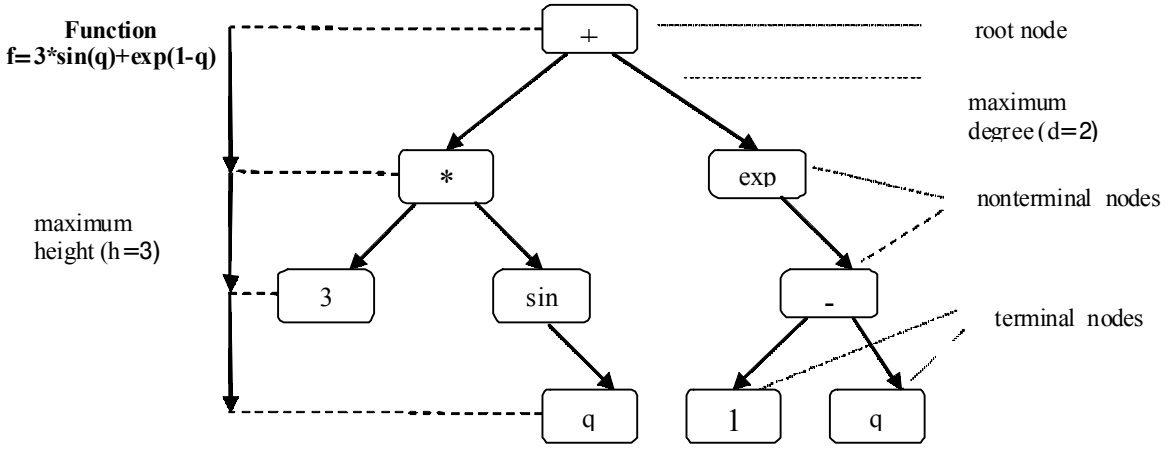


Figure 1. Representing an example of a function through a tree structure

are terminals. Traversing the binary tree properly leads to the encoded function. In Genetic Programming, trees are recombined via the crossover operation. In this process, the two selected tree structures exchange randomly selected subtrees. The mutation operator is also applied in this case - it involves changing a function or a variable in a given node of the graph.

Evolutionary algorithms are successfully integrated via methods for image recognition. [5] uses GA for Feature Selection while [10] also applies parameters directly to the decision function. In [11], GP has been used for feature reduction in transmission mechanisms fault detection.

Support Vector Machines

SVM is one of the recently established methods in Statistical Learning Theory for regression and classification problems [3,12]. The definition of the classification problem involving two classes is the following: there are N-class learning images, represented by the vectors $(y_i, \mathbf{x}_i)$, $i = 1...N$ where: y_i is a class label; $\mathbf{x}_i \in \Re^d$ - a feature vector. The classifier is represented by the function $f(\mathbf{x}, \mathbf{w}, b) : \mathbf{x} \to y$. SVM is used to determine an optimal separating hyperplane so that the learning examples with notations for two classes + 1 and - 1 are situated on both sides of the hyperplane at a maximum distance (margin) between the boundary support vectors for the two classes *(figure 2*).

The solution is defined by the function $\mathbf{wx} + b = 0$ with

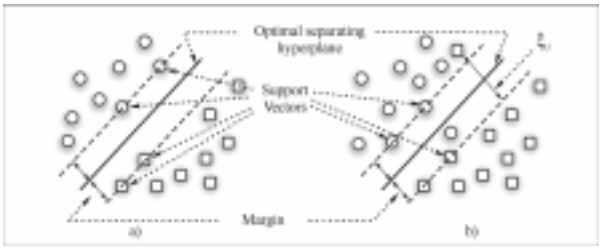


Figure 2. An optimal hypersurface, linear (a) and non-linear (b) class separation

parameters $(\mathbf{w}, b)$. For vectors which do not cut across the hyperplane, the following conditions are met: $\mathbf{wx} + b > 0$ or $\mathbf{wx} + b < 0$. Thus, the classifier is defined as $f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{wx} + b)$. Support vectors lie on two parallel hypersurfaces $\mathbf{wx} + b = \pm 1$. The maximization of the margin with these two equations results in an optimization problem with the following additional conditions:

$$(1)\ \min\left\{\frac{1}{2}\|w\|^2\right\} \text{ for } y_i(\mathbf{w}.\mathbf{x} + b) \geq 1, i = 1 \ldots N.$$

Since the learning vectors are not always linearly separable, in (1) a regularization parameter C is introduced as well as variables which account for errors ξ_i in the decision function. In this case, the problem is expressed in the following way:

$$(2)\quad \begin{cases} \min\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i;\} \\ y_i(\mathbf{w}\mathbf{x}+b) \geq 1-\xi_i; \\ \xi_i \geq 0,\ i=1\ldots N. \end{cases}$$

To solve the optimization problem (2), a Lagrange form of dual is used to get:

$$(3)\quad \begin{cases} \max\{\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \langle \mathbf{x}_i \mathbf{x}_j \rangle; \\ C \geq \lambda_i \geq 0,\ i=1\ldots N; \\ \sum_{i=1}^{N}\lambda_i y_i = 0,\ i=1\ldots N. \end{cases}$$

Lagrange variables λ_i have a non-zero value only for the support vectors.

SVM can be generalized in the case of non-linearly separable surfaces. Typical for this method is the projection of data into a new high-dimensional space where the data are linearly separable. The operation in this space is simulated via the so-called Kernel Method. If the vectors $\mathbf{x} \in \Re^d$ map into a new space H, through the function $\Phi: \Re^d \to H$, then the product $\langle \mathbf{x}_i.\mathbf{x}_j \rangle$ from (3) is transformed with $\langle \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j) \rangle$. The product $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j) \rangle$ is a kernel which is directly used in (3), without explicit computing of Φ (*kernel trick*).

The SVM analytic function is defined only for the separation of two classes. For multi-class separation (*M*-class), two approaches are used: One-Against-All - *M* classifiers are applied iteratively to separate the point of each class from all the others; One-Against-One - $M(M-1)/2$ classifiers separate all class pairs, the final solution being the class determined most frequently.

One of the important characteristics of the Support Vector Machines method is the possibility for reducing the free parameters of the model while preserving the levels of the classification error. With SVM, the learning procedure automatically determines the decisive rule for a function with a specific number (of support vectors). This contributes to the simplification of the model (the discovery of the so-called *sparse structure*). One aspect of the problem is the more precise definition of support vectors. The research along these lines is concentrated on the optimization of the learning sample used. A contemporary method derived from SVM is the Relevance Vector Machines method (RVM) [15] which resorts to Bayesian evaluation as a selection mechanism for support vectors. Another possibility for simplifying the model is the reduction of the feature space which accounts for feature selection [10]. A great number of applied problems prove that 4-18% of all features defined are informative and independent.

With regard to the so-called *kernel methods*, which include SVM, it is well known that the transformation function plays a vital role. The inadequate selection of a kernel function results in undesirable characteristics. In view of this, recent years saw the appearance of various kernel selection methods (*kernel learning*). For the most part, these methods are restricted to learning the parameters of some standard kernel functions. In this respect, the application of genetic programming allows for the elimination of this condition and the organization of the more general approach of non-parametric learning.

A Genetic Algorithm for Feature Selection

A major issue in classifier synthesis is the selection of informative features. One of the methods for selection utilizes GA. In SVM, feature selection is not critical to the generalization capacity and classification precision of the classifier. Feature reduction here is important with respect to the working time of the algorithm which is determined by the time needed to compute kernels.

In feature selection, the chromosome in the genetic algorithm constitutes a binary string (*table 1*). The genes (bytes) with a value 1 testify to the presence of the given feature in the complex.

Table 1

A complete feature complex	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	...
Chromosome	1	1	0	1	1	0	0	0	1	1	1	...
A tested subset	s_1	s_2		s_4	s_5				s_9	s_{10}	s_{11}	...

The genetic algorithm can be generally defined by the following sequence:

1. An initial population is randomly generated

$$\begin{array}{l} 10110100011001001000 10 \\ 00100101001001011010 11 \rightarrow Feature\ \ masks \\ \cdot\qquad\qquad\cdot\qquad\qquad\cdot \\ \cdot\qquad\qquad\cdot\qquad\qquad\cdot \\ \cdot\qquad\qquad\cdot\qquad\qquad\cdot \\ 11110111101101111110 01 \end{array}$$

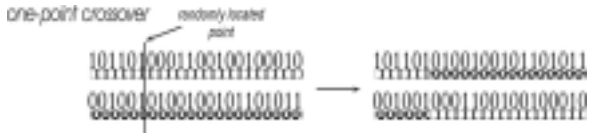
Table 2

Support Vector Machines (k_{RBF}) – *One-Against-One* approach, $C = 3.6, \sigma = 1.7$				
Errors			Number of support vectors	Number of features
Class 1/3	Class 3/1	Mean		
1.15	0.10	0.63	14	7
Features: [s6,s8,s10,s20]; Average learning time: 2.14s Average classification time (100 items): 0.01 s				
Class 1/2	Class 2/1	Mean	Number of support vectors	Number of features
2.29	6.89	4.59	48	6
Features: [s2 s6 s8 s10 s12 s20]				
Class 2/3	Class 3/2	Mean	Number of support vectors	Number of features
1.29	2.59	1.94	37	5
Features: [s4 s6 s10 s12 s14]				
Errors from *One-Against-One* multi-class approach:				
Class 1(86)			Class 2 (88)	Class 3 (66)
82 2 2			2 82 4	1 3 62
4.65			6.81	6.06

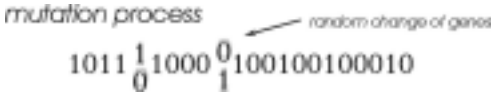
2. Until stopping criteria (a specified time or generation number) are met, the following steps occur:

2.1. Selection of parents from the population - the *selection operator*. The most common variants of this operator are proportional selection and tournament selection. With proportional selection, individuals with a maximum fitness are chosen - in this case, this corresponds to the maximum frequency of correct classification.

2.2. Generation of a new population - the *crossover operator*. The function of the operator is to create offspring. This happens with a given probability P_c and at a randomly located point of separation. If this operator is not applied (with a probability $1 - P_c$), the offspring inherits the parents' characteristics.



2.3. Modification of certain genes - *the mutation operator*.



A random change of a coordinate is carried out (with probability P_m).

2.4. The population is renewed

Table 2 presents the results from a crossvalidation test in the case of classification and search for optimal feature vectors through GA. Images from a database were classified, the database containing unidimensional realizations of potatoes (*figure 3*).

The spectral permeability of the tubers was measured during their passage at a fixed velocity through a photometric camera. Each potato was evaluated by an expert and, depending on the organoleptic properties, classified into one of the three quality classes - first, second, and third quality. The initial

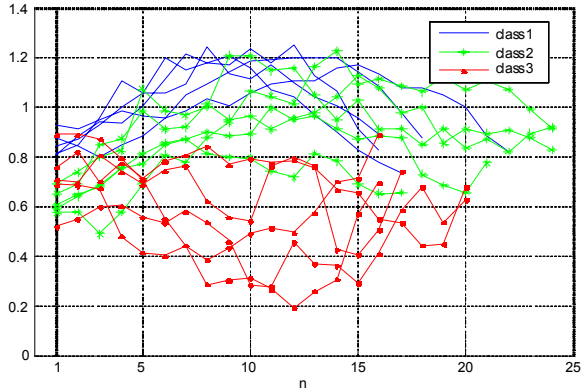


Figure 3. Unidimensional spectral realization of potatoes for three classes

feature complex comprised 20 features. Feature selection can be carried out for each classifier within the general classification scheme for more than two classes.

Synthesis of a Kernel

SVM makes use of the so-called kernel. The most commonly used kernels are linear kernels, polynomial kernels, and radial basis functions:

$$K(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z} \rangle;\ \ K(\mathbf{x},\mathbf{z}) = (1 + \langle \mathbf{x},\mathbf{z} \rangle)^n;$$

$$(4)\quad K(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|/2\sigma^2).$$

Each kernel represents a given space and, depending on the data distribution in the problem, linear separation cannot be guaranteed. An alternative is to synthesize a specific kernel suitable for the concrete example or to ensure a general automation of the search for kernels and their testing. The literature published on this new development is connected with the application of GP. The classic approach to this kind of search of the kernel is presented in [8]. The paper [14] proposes an algorithm for kernel synthesis using genetic programming. The method guarantees the finding of a kernel satisfying Mercer's theorem. This results from the requirement for the formation of new functions which are a sum of the standard ones. The article [4] suggests the Evolutionary Kernel Machine method. Assessment of the kernel function is performed by the Nearest Neighbor classification algorithm integrated with the method of generating functions using kernel genetic programming.

With this hybrid approach to learning, each chromosome encodes a given kernel function which is SVM-tested and the accuracy of the resultant classification is evaluated. The representation of the kernel function is made through a tree structure (*figure1*). Respective tree nodes correspond to primitive functions or preset variables. *Table 3* illustrates a possible set of functions.

Basic Operators

Initialization. It consists in the generation of trees. One of the methods for the initialization of a non-AVL tree is the *GROW* Method [9,13]. With this method, a local root is determined at a preset depth. If the local root is generated when a maximum depth has been obtained, then it is defined as terminal and represented by a randomly selected symbol. If a maximum depth has not been reached, the node is defined as terminal or nonterminal. If it is terminal, the tree does not grow beyond this node. In the case of a nonterminal node, the algorithm is recursively applied to it. To adapt the algorithm in view of synthesizing a kernel, some additional conditions are to be met in order to obtain syntactically correct chromosomes:

- The root of each tree is determined by the set of scalar functions (*table 3*) or a norm depending on vector $\Re$ (the program is supposed to return a value).
- Vector norm functions ($\Re$) inherit scalar functions.
- If a given node includes a vector function ($\Re$ or $\Re^d$), its heirs are initialized by a vector function - $\Re^d$ or a terminal symbol.

Table 3

Scalar $\Re$ - valued functions	Vector functions		Terminal variables
	$\Re$ -valued function	$\Re^d$ - valued function	$\Re^d$
+ - * / exp	- Euclid norm: $En = \sum_{i=1}^{d}(x_i - z_i)^2$ - scalar multiplication: $Sp = \sum_{i=1}^{d} x_i z_i$ - Gaussian function: $Gn = \exp(-\gamma\|x-z\|^2)$	- elementary vector operators: addition - $Sum = \mathbf{x}. + \mathbf{z}$ ($Sum_i = x_i + z_i, i = 1,..,d$); subtraction $Sum = \mathbf{x}. - \mathbf{z}$; multiplication $Pdt = \mathbf{x}. * \mathbf{z}$. *(Corresponding to the notation of "element-wise" MATLAB operations)*	Conventionally assumed to be two because kernels are computed for vector pairs (3) - $\mathbf{x}$ - $\mathbf{z}$

- The restrictions imposed on the kernel are symmetry (K(x,z) = K(z, x)) and a positive definite Gram matrix $\mathbf{K} = (K(x_i, x_j))_{i,j=1}^{n}$. The first restriction can be overcome through designing a kernel of the following type $K(\mathbf{x},\mathbf{z}) = \langle f(\mathbf{x},\mathbf{z}).f(\mathbf{z},\mathbf{x}) \rangle$ where *f (x, z)* is a function represented by the tree which is to be tested. The second restriction can be dealt with in two ways: it may be disregarded, which does not guarantee that an optimal solution will be arrived at (although it is still possible to find a near optimal solution) or new kernels may be synthesized, a composition of kernels corresponding to the condition, i.e. standard kernels (4). The research does not envisage the discovery of a positive definite Gram matrix in order to simplify the algorithm when testing kernels.

Crossover and mutation. From the selected chromosomes, a one-point crossover is carried out. After that, subtrees and the separation root are swapped. The crossover point of the first parent is random and that of the second parent is determined by it. To obtain syntactically correct chromosomes, if the local separation root is above a scalar function node, the point of

separation of the second tree is to be randomly selected above another scalar node.

The mutation operator is applied by randomly selecting a given node, the subtree beyond it being erased. The classic example utilizes the procedure for node generation applied in initialization. For the realization of the algorithm and the SVM classifier learning, the optimization MATLAB toolbox and the *GPLAB* toolbox (Genetic Programming toolbox for MATLAB) [13] were used. To decrease search time and prevent the generation of complicated models, the maximum tree height was fixed at 3. Other parameters are: population - 8; number of generations - 40; P_c = 0.7 (crossover); P_m = 0.1 (mutation). Given these parameters of the algorithm, research was carried out on a two-class problem, for Class 1 and Class 2. *Table 4* presents the results from the minimum error values obtained from standard kernels and through GP. The C parameter and the parameters of the standard kernels were determined experimentally for values at specified steps. The application of GP allows for error analysis and a choice between simpler kernels in the case of error levels with a close proximity.

Table 4

Classifier (kernel)	Mean classification error rate (testing sample), %
Linear	7.83
RBF, $C = 3.6, \sigma = 1.7$	5.67
Polynomial, $C = 3.7$, $p = 2$	6.55
Through Genetic Programming	6.42

Conclusion

This work demonstrated evolutionary algorithms for SVM machine learning. In the feature selection problem a genetic algorithm was used in order to reduce the feature set, to increase accuracy and decrease the classifier complexity. The algorithm was adapted to the synthesis of multiclass classification schemes. The genetic programming method was applied to the second problem of an automatic search for a kernel function mathematical description specific for SVM.

The experiment showed that this type of search was time-consuming. The time necessary for the quadratic optimizer to find a solution was vital in this case; with inappropriate kernels, the time increased dramatically. Therefore, errors were evaluated with the help of a testing sample (whereas a cross-validation test was used in the previous case) at a reduced volume of the learning sample.

An improvement of the classification accuracy in comparison with standard kernel functions was achieved in this classification problem.

Acknowledgements

This study was carried out in the framework of the DO 02-143/16.12.2008 project titled „Development of Intelligent Technologies for Assessment of Quality and Safety of Food and Agricultural Products“, financed by the National Science Fund of the Bulgarian Ministry of Education and Science.

References

1. Bartlett, P. L., W. Maass. Vapnik- Chervonenkis Dimension of Neural Nets. In Arbib M. A., Editor. The Handbook of Brain Theory and Neural Networks, 1188-1192. MIT Press (Cambridge), 2nd edition.

2. Boser, B. E., I. Guyon, V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. Proceeding of the 5th Annual ACM Conference on Computational Learning Theory (COLT'92), 144-152, Pittsburgh, Pa, USA, July 1992.

3. Damyanov, C., V. Nachev, M. Mukarev. On the Generalization in Learning Pattern Recognition System. - *Automatica & Informatics*, 2005, No. 4. 14-20.

4. Gagne, C., M. Schoenauer, M. Sebag, M. Tomassini. Genetic Programming for Kernelbased Learning with Co-evolving Subsets Selection. In Proceedings of Parallel Problem Solving in Nature, 2006.

5. Galvao, R. K. Variable Selection for Financial Distress Classification Using a Genetic Algorithm. IEEE, 2002.

6. Gen, M., R. Cheng. Genetic Algorithms and Engineering Optimization. New York, Wiley, 2000.

7. Goldberg, D. Genetic Algorithms in Search, Optimization, and Machine Learning, Reading. MA, Addison-Wesley, 1989.

8. Howley, T., G. Madden. The Genetic Kernel Support Vector Machine: Description and Evaluation. - *Artificial Intelligence Review*, 24 (3-4), 2005, 379 - 395.

9. Koza, J. R. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University Computer Science Department, Techn.Report, Available from <http://www.geneticprogramming. com/jkpdf/tr1314.pdf>, 1990.

10. Morariu, D., L. Vintan, V. Tresp. Feature Selection Methods for an Improved SVM Classifier. Proceedings of the 3rd International Conference on Intelligent Systems, ICIS06, Prague, August, 2006.

11. Samanta, B., Gear Fault. Detection Using Artificial Neural Networks and Support Vector Machines with Genetic Algorithms. - *Mechanical Systems and Signal Processing*, 18, 2004, 625-644.

12. Schoslkopf, B., A. Smola. Learning with Kernels, Support Vector Machines. MIT Press, London, 2002.

13. Silva, S. GPLAB - A Genetic Programming Toolbox for MATLAB. Available from <http://gplab. sourceforge.net/>.

14. Sullivan, K., S. Luke. Evolving Kernels for Support Vector Machine Classification. Genetic And Evolutionary Computation Conference, London, England, Session: Genetic Programming, 2007, 1702-1707.

15. Tipping, M. Sparse Bayesian Learning and the Relevance Vector

Machine. - *Journal of Machine Learning Research*, 1, 2001, 211-244.
16. Vapnik, V. The Natural of Statistical Theory. New York, Springer Verlag, 1995.
17. Vapnik, V. N., A. Y. Chervonenkis., On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. Theory of Probability and its Applications, 16 (2), 1971, 264-280.

Manuscript received on 14.04.2009

Veselin Nachev *(born 1975), has a doctoral degree (PhD) in automatics and computing components and devices. He is a Principal Assist. Prof. at the „Automatics, Information and Control Systems" Department at the University of Food Technologies of Plovdiv. His scientific interests are control system, pattern recognition, artificial intelligence systems and automatic quality evaluation of food and agricultural products.*

Contacts:
Computer Systems and Technologies Department
University of Food Technologies
26 Maritza Blvd.
4002 Plovdiv, Bulgaria
Phone: +359 32 603765
e-mail: v_nachevbg@yahoo.com

Tanya Titova *(born 1982), has an M.Sc. degree in Automatics, Information and Control Systems. She is an extramural PhD student. Assist. Prof. Titova works at the „Automatics, Information and Control Systems" Department at the University of Food Technologies of Plovdiv. Her scientific interests are control system, pattern recognition, artificial intelligence systems and automatic quality evaluation of food and agricultural products.*

Contacts:
Automatics, Information and Control Systems Department
University of Food Technologies
26 Maritza Blvd.
4002 Plovdiv, Bulgaria
Phone: +359 32 603765
e-mail: t_titova@abv.bg

Chavdar Damyanov *(born 1946), has an M.Sc. and PhD degree from the Technical University of Sofia. He is a D.Sc. and professor at the University of Food Technologies of Plovdiv and he is a Head of „Automatics, Information and Control Systems" Department in the same university. His scientific and research interests are in the areas of automatic control systems, pattern recognition, application of artificial intelligence and quality evaluation and nondestructive sorting of food products. He has over 180 publications and several patents.*

Contacts:
Automatics, Information and Control Systems Department
University of Food Technologies
26 Maritza Blvd.
4002 Plovdiv, Bulgaria
Phone: +359 32 603897
e-mail: chavdam@yahoo.com