Evolutionary Support Vector Machines and their Application for Classification

Ruxandra Stoean, Mike Preuss, Catalin Stoean and D. Dumitrescu

Nr. CI-212/06

Interner Bericht ISSN 1433-3325 June 2006

Sekretariat des SFB 531 · Universität Dortmund · Fachbereich Informatik/XI
44221 Dortmund · Germany

Diese Arbeit ist im Sonderforschungsbereich 531, „Computational Intelligence“, der Universität Dortmund entstanden und wurde auf seine Veranlassung unter Verwendung der ihm von der Deutschen Forschungsgemeinschaft zur Verfügung gestellten Mittel gedruckt.

Evolutionary Support Vector Machines and their Application for Classification

Ruxandra Stoean[1], Mike Preuss[2], Catalin Stoean[1], and D. Dumitrescu[3]

[1] Department of Computer Science, Faculty of Mathematics and Computer Science, University of Craiova, Romania
{ruxandra.stoean,catalin.stoean}@inf.ucv.ro

[2] Chair of Algorithm Engineering, Department of Computer Science, University of Dortmund, Germany
mike.preuss@uni-dortmund.de

[3] Department of Computer Science, Faculty of Mathematics and Computer Science, "Babes-Bolyai" University of Cluj-Napoca, Romania
ddumitr@cs.ubbcluj.ro

Abstract. We propose a novel learning technique for classification as result of the hybridization between support vector machines and evolutionary algorithms. Evolutionary support vector machines consider the classification task as in support vector machines but use evolutionary algorithms to solve the optimization problem of determining the decision function. They can acquire the coefficients of the separating hyperplane, which is often not possible within classical techniques. More important, ESVMs obtain the coefficients directly from the evolutionary algorithm and can refer them at any point during a run. The concept is furthermore extended to handle large amounts of data, a problem frequently occurring e.g. in spam mail detection, one of our test cases. Evolutionary support vector machines are validated on this and three other real-world classification tasks; obtained results show the promise of this new technique.

Keywords: support vector machines, coefficients of decision surface, evolutionary algorithms, evolutionary support vector machines, parameter tuning

1 Introduction

Support vector machines (SVMs) represent a state-of-the-art learning technique that has managed to reach very competitive results in different types of classification and regression tasks. Their engine, however, is quite complicated, as far as proper understanding of the calculus and correct implementation of the mechanisms are concerned. This paper presents a novel approach, *evolutionary support vector machines* (ESVMs), which offers a simpler alternative to the standard technique inside SVMs, delivered by *evolutionary algorithms* (EAs). Note that this is not the first attempt to hybridize SVMs and EAs. Existing alternatives are discussed in §2.2. Nevertheless, we claim that our approach is significantly different from these.

ESVMs as presented here are constructed solely based on SVMs applied for classification. Validation is achieved by considering four real-world classification tasks. Besides comparing results, the potential of the utilized, simplistic EA through parametrization is investigated. To enable handling large data sets, the first approach is enhanced by use of a chunking technique, resulting in a more versatile algorithm. A second solution for dealing with a high number of samples is brought by a reconsideration of the elements of the first evolutionary algorithm. Obtained results prove suitability and competitiveness of the new approach, so ESVMs qualify as viable simpler alternative to standard SVMs in this context. However, this is only a first attempt with the new approach. Many of its components still remain to be improved.

The paper is organized as follows: §2 outlines the concepts of classical SVMs together with existing evolutionary approaches aimed at boosting their performance. §3 presents the new approach of ESVMs. Their validation is achieved on real-world examples in §4. Improved ESVMs with two new mechanisms for reducing problem size in case of large data sets are presented in §5. The last section comprises conclusions and outlines ideas for further improvement.

2 Prerequisites

Given $\{f_{t \in T}|, f_t : R^n \rightarrow \{1, 2, ..., k\}\}$, a set of functions, and $\{(x_i, y_i)\}_{i=1,2,...,m}$, a training set where every $x_i \in R^n$ represents a data vector and each y_i corresponds to a class, a classification task consists in learning the optimal function f_{t^*} that minimizes the discrepancy between the given classes of data vectors and the actual classes produced by the learning machine. Finally, accuracy of the machine is computed on previously unseen test data vectors. In the classical architecture, SVMs reduce k-class classification problems to many binary classification tasks that are separately considered and solved. A voting system then decides the class for data vectors in the test set. SVMs regard classification from a geometrical point of view, i.e. they assume the existence of a separating surface between two classes labelled as -1 and 1, respectively. The aim of SVMs then becomes the discovery of this decision hyperplane, i.e. its coefficients.

2.1 Classical Support Vector Machines

If training data is known to be linearly separable, then there exists a linear hyperplane that performs the partition, i.e. $\langle w, x \rangle - b = 0$, where $w \in R^n$ is the normal to the hyperplane and represents hyperplane orientation and $b \in R$ denotes hyperplane location. The separating hyperplane is thus determined by its coefficients, w and b. Consequently, the positive data vectors lie on the corresponding side of the hyperplane and their negative counterparts on the opposite side. As a stronger statement for linear separability, the positive and negative vectors each lie on the corresponding side of a matching supporting hyperplane for the respective class (Figure 1a) [1], written in brief as:

$$y_i(\langle w, x_i \rangle - b) > 1, i = 1, 2, ..., m.$$

In order to achieve the classification goal, SVMs must determine the optimal values for the coefficients of the decision hyperplane that separates the training data with as few exceptions as possible. In addition, according to the principle of Structural Risk Minimization from Statistical Learning Theory [2], separation must be performed with a maximal margin between classes. Summing up, classification of linear separable data with a linear hyperplane through SVMs leads thus to the following optimization problem:

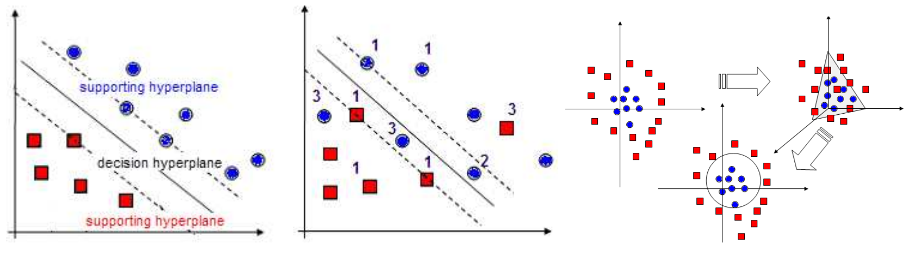
$$\begin{cases} \text{find } w^* \text{ and } b^* \text{ as to minimize } \frac{\|w^*\|^2}{2} \\ \text{subject to } y_i(\langle w^*, x_i \rangle - b^*) \geq 1, i = 1, 2, ..., m. \end{cases} \tag{1}$$

Training data are not linearly separable in general and it is obvious that a linear separating hyperplane is not able to build a partition without any errors. However, a linear separation that minimizes training error can be tried as a solution to the classification problem. Training errors can be traced by observing the deviations of data vectors from the corresponding supporting hyperplane, i.e. from the ideal condition of data separability. Such a deviation corresponds to a value of $\frac{\pm\xi_i}{\|w\|}$, $\xi_i \geq 0$. These values may indicate different nuanced digressions (Figure 1b), but only a ξ_i higher than unity signals a classification error. Minimization of training error is achieved by adding the indicator of error for every training data vector into the separability statement and, at the same time, by minimizing the sum of indicators for errors. Adding up, classification of linear nonseparable data with a linear hyperplane through SVMs leads to the following optimization problem, where C is a hyperparameter representing the penalty for errors:

$$\begin{cases} \text{find } w^* \text{ and } b^* \text{ as to minimize } \frac{\|w^*\|^2}{2} + C\sum_{i=1}^{m} \xi_i, C > 0 \\ \text{subject to } y_i(\langle w^*, x_i \rangle - b^*) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, 2, ..., m. \end{cases} \tag{2}$$

If a linear hyperplane does not provide satisfactory results for the classification task, then a nonlinear decision surface can be appointed. The initial space of training data vectors can be nonlinearly mapped into a high enough dimensional feature space, where a linear decision hyperplane can be subsequently built. The separating hyperplane will achieve an accurate classification in the feature space which will correspond to a nonlinear decision function in the initial space (Figure 1c). The procedure leads therefore to the creation of a linear separating hyperplane that would, as before, minimize training error, only this time it would perform in the feature space. Accordingly, a nonlinear map $\Phi : R^n \rightarrow H$ is considered and data vectors from the initial space are mapped into H. w is also mapped through Φ into H. As a result, the squared norm that is involved in maximizing the margin of separation is now $\|\Phi(w)\|^2$. Also, the equation of the hyperplane consequently changes to $\langle \Phi(w), \Phi(x_i) \rangle - b = 0$.

Nevertheless, as simple in theory, the appointment of an appropriate map Φ with the above properties is a difficult task. As in the training algorithm vectors appear only as part of dot products, the issue would be simplified if there were a kernel function K that would obey $K(x, y) = \langle \Phi(x), \Phi(y) \rangle,$ where $x, y \in R^n$. In this way, one would use K everywhere and would never need to explicitly even know what Φ is. The remaining problem is that kernel functions with this property are those that obey the corresponding conditions of Mercer's theorem from functional analysis, which are not easy to check. There are, however, a couple of classical kernels that had been demonstrated to meet Mercer's condition, i.e. the polynomial classifier of degree p: $K(x, y) = \langle x, y \rangle^p$ and the radial basis function classifier: $K(x, y) = e^{\frac{\|x-y\|^2}{\sigma}}$, where p and σ are also hyperparameters of SVMs. In conclusion, classification of linear nonseparable data with a



(a) (b) (c)

Fig. 1: (a) Decision hyperplane (continuous line) that separates between circles (positive) and squares (negative) and supporting hyperplanes (dotted lines). (b) Position of data and corresponding indicators for errors - correct placement, $\xi_i = 0$ (label 1) margin position, $\xi_i < 1$ (label 2) and classification error, $\xi_i > 1$ (label 3). (c) Initial data space (left), nonlinear map into the higher dimension/its linear separation (right), and corresponding nonlinear surface (bottom).

nonlinear hyperplane through SVMs leads to the same optimization problem as in (2) which is now considered in the feature space and with the use of a kernel function:

$$\begin{cases} \text{find } w^* \text{ and } b^* \text{ as to minimize } \frac{K(w^*,w^*)}{2} + C\sum_{i=1}^{m}\xi_i, C > 0 \\ \text{subject to } y_i(K(w^*,x_i) - b^*) \geq 1 - \xi_i, \xi_i \geq 0, i = 1,2,...,m. \end{cases} \quad (3)$$

The optimization problem (corresponding to either situation above) is subsequently solved. Accuracy on the test set in then computed, i.e. the side of the decision boundary on which each new data vector lies is determined. Classical SVMs approach the optimization problem through a generalized form of the method of Lagrange multipliers [3]. But the mathematics of the technique can be found to be very difficult both to grasp and apply. This is the reason why present approach aims to simplify (and improve) SVMs through a hybridization with EAsby utilizing these in order to determine optimal values for the coefficients of the separating hyperplane (w and b) directly.

2.2 Evolutionary Approaches to Support Vector Machines

EAs have been widely used in hybridization with SVMs in order to boost performance of classical architecture. Their combination envisaged two different directions: model selection and feature selection. Model selection concerns adjustment of hyperparameters (free parameters) within SVMs, i.e. the penalty for errors, C, and parameters of the kernel, p or σ which, in standard variants, is performed through grid search or gradient descent methods. Evolution of hyperparameters can be achieved through evolution strategies [4]. When dealing with high dimensional classification problems, feature selection regards the choice of the most relevant features as input for a SVM. The optimal subset of features can be evolved using genetic algorithms in [5] and genetic programming in [6]. To the best of our knowledge, evolution of coefficients of the separating hyperplane within SVMs has not been accomplished yet.

3 Evolutionary Support Vector Machines for Classification

Within the new hybridized technique of ESVMs separation of positive and negative vectors proceeds as in standard SVMs, while the optimization problem is solved by means of EAs. Therefore, the coefficients of the separating hyperplane, i.e. w and b, are encoded in the representation of the EA and their evolution is performed with respect to the objective function and the constraints in the optimization problem (3) within SVMs, which is considered for reasons of generality.

3.1 Evolving the Coefficients of the Separating Hyperplane

Representation An individual encodes the coefficients of the separating hyperplane, w and b. Since indicators for errors of classification, ξ_i, $i = 1, 2, ..., m$, appear in the conditions for hyperplane optimality, ESVMs handle them through inclusion in the structure of an individual, as well:

$$c = (w_1, ..., w_n, b, \xi_1, ..., \xi_m). \tag{4}$$

After termination of the algorithm, the best individual from all generations gives approximately optimal values for the coefficients of the decision hyperplane. If proper values for parameters of the evolutionary algorithm are chosen, training errors of classification can also result from the optimal individual (those indicators whose values are higher than unity) but with some loss in accuracy; otherwise, indicators grow in the direction of errors driving the evolutionary cycle towards its goal, but do not reach the limit of unity when the evolutionary process stops. An example of an ESVM which also provides the errors of classification is nevertheless exhibited for simple artificial 2-dimensional data sets separated by various kernels (Figure 2). In such a situation, the number of samples and the dimensionality of data are both low, thus accuracy and runtime are not affected by the choice of parameters which leads to the discovery of all training errors.

Initial population Individuals are randomly generated such that $w_i \in [-1, 1], i = 1, 2, ..., n$, $b \in [-1, 1]$ and $\xi_j \in [0, 1], j = 1, 2, ..., m$.

Fitness evaluation The fitness function derives from the objective function of the optimization problem and has to be minimized. Constraints are handled by penalizing the infeasible individuals by appointing $t : R \to R$ which returns the value of the argument if negative while otherwise 0. The expression of the function is thus as follows:

$$f(w, b, \xi) = K(w, w) + C \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} [t(y_i(K(w, x_i) - b) - 1 + \xi_i)]^2. \tag{5}$$

Genetic operators Operators were chosen experimentally. Tournament selection, intermediate crossover and mutation with normal perturbation are applied. Mutation of errors is constrained, preventing the ξ_is from taking negative values.

Stop condition The algorithm stops after a predefined number of generations. As the coefficients of the separating hyperplane are found, the class for a new, unseen test data vector can be determined, following $class(x) = sgn(K(w, x) - b)$. This is unlike classical SVMs where it is seldom the case that coefficients can be determined following the standard technique, as the map Φ cannot be always explicitly determined. In this situation, the class for a new vector follows from computational artifices.

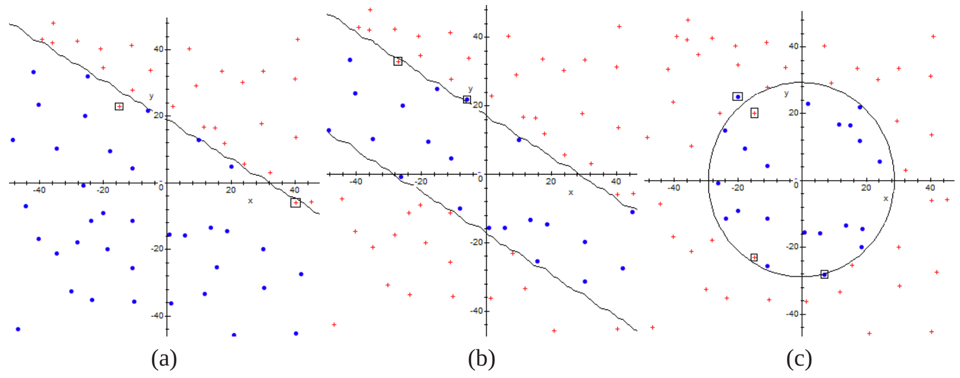


Fig. 2: *Vizualization of ESVMs on 2-dimensional points. Errors of classification are squared. Odd (a), even (b) polynomial and radial (c) kernels*

3.2 ESVMs for Multi-class Classification

Multi-class ESVMs employ one classical and very successful SVM technique, the ONE-AGAINST-ONE (1-a-1) [7]. As the classification problem is *k*-class, $k > 2$, 1-a-1 considers $\frac{k(k-1)}{2}$ SVMs, where each machine is trained on data from every two classes, *i* and *j*, where *i* corresponds to *1* and *j* to *-1*. For every SVM, the class of *x* is computed and if *x* is in class *i*, the vote for the *i*-th class is incremented by one; conversely, the vote for class *j* is added by one. Finally, *x* is taken to belong to the class with the largest vote. In case two classes have identical number of votes, the one with the smaller index is selected. Consequently, 1-a-1 multi-class ESVMs are straightforward. $\frac{k(k-1)}{2}$ ESVMs are built for every two classes and voting is subsequently conducted.

4 Experimental Evaluation: Real-World Classification

Experiments have been conducted on four data sets (with no missing values) concerning real-world problems coming from the UCI Repository of Machine Learning Databases[4], i.e. diabetes mellitus diagnosis, spam detection, iris recognition and soybean disease diagnosis. The motivation for the choice of test cases was manifold. Diabetes and spam are two-class problems, while soybean and iris are multi-class. Differentiating, on the one hand, diagnosis is a better-known benchmark, but filtering is an issue of current major concern; moreover, the latter has a lot more features and samples, which makes a huge difference for classification as well as for optimization. On the other hand, iris has a lot more samples while soybean has a lot more attributes. For all reasons mentioned above, the selection of test problems certainly contains all the variety of situations that is necessary for the objective validation of the new approach of ESVMs. Brief information on the classification tasks and SVM and EA parameter values are given in Table 1. The error penalty was invariably set to 1.

For each data set, 30 runs of the ESVM were conducted; in every run 75% random cases were appointed to the training set and the remaining 25% went into test. Experiments showed the necessity for data normalization in diabetes, spam and iris.

[4] Available at http://www.ics.uci.edu/ mlearn/MLRepository.html

Table 1: *Data set properties and ESVM algorithm parameter values. Rightmost columns hold tuned parameter sets for modified ESVMs, ESVMs with chunking, and utilized parameter bounds.*

	Diabetes		**Iris**		**Soybean**		**Spam**				
Data set description									modif.	chunk.	bounds
Number of samples	768		150		47		4601			(434)	(10/500)
Number of attributes	8		4		35		57				
Number of classes	2		3		4		2				
ESVM parameter values	man.	tun.	man.	tun.	man.	tun.	man.	tun.	tun.	tun.	
Kernel parameter (p or σ)	$p = 2$		$\sigma = 1$		$p = 1$		$p = 1$		$p = 1$		
Population size	100	198	100	46	100	162	100	154	51	90	10/200
Number of generations	250	296	100	220	100	293	250	287	180	286	50/300
Crossover probability	0.40	0.87	0.30	0.77	0.30	0.04	0.30	0.84	0.95	0.11	0.01/1
Mutation probability	0.40	0.21	0.50	0.57	0.50	0.39	0.50	0.20	0.03	0.08	0.01/1
Error mutation probability	0.50	0.20	0.50	0.02	0.50	0.09	0.50	0.07	—	0.80	0.01/1
Mutation strength	0.10	4.11	0.10	4.04	0.10	0.16	0.10	3.32	3.75	0.98	0.001/5
Error mutation strength	0.10	0.02	0.10	3.11	0.10	3.80	0.10	0.01	—	0.01	0.001/5

No further modification of the data was carried out and all data was used in the experiments. Obtained test accuracies are presented in Table 2. Differentiated (spam/non spam for spam filtering and ill/healthy for diabetes) accuracies are also depicted. In order to validate the manually found EA parameter values, the parameter tuning method SPO [8] was applied with a budget of 1000 optimization runs. The last 4 columns of Table 2 hold performances and standard deviations of the best configuration of an initial latin hypersquare sample (LHS) with size $10 \times$ #parameters, and the finally found best parameter setting. Resulting parameter values are depicted in Table 1. They indicate that for all cases, except for soybean data and chunking enhanced ESVM on the spam data, crossover probabilities were dramatically increased, while often reducing mutation probabilities, especially for errors. It must be stated that in most cases, results achieved with manually determined parameter values are only improved by increasing effort (population size or number of generations).

Comparison to worst and best found results of different techniques, either SVMs or others, was conducted. Assessment cannot be objective, however, as outlined methods either use different sizes for the training/test sets or other types of cross-validation and number of runs or employ various preprocessing procedures for feature or sample selection. *Diabetes diagnosis* was approached in SVM^{light} [9] where an accuracy of 76.95% was obtained and Critical SVMs [10] with a result of 82.29%. Accuracy on *spam detection* is reported in [11] where k-Nearest Neighbourhood on non preprocessed data resulted in 66.5% accuracy and in [12] where functional link network wrapped into a genetic algorithm for input and output feature selection gave a result of 92.44%. 1-a-1 multi-class SVMs on the *Iris data set* were perfomed in [7] (accompanied by a shrinking technique) and [13]; obtained results were of 97.33% and 98.67%, respectively. Results for the *Soybean small data* set were provided in [14], where, among others, Naive Bayes was applied and provided an accuracy of 95.50%, reaching 100% when a pair-wise classification strategy was employed.

Table 2: *Accuracies of different ESVM versions on the considered test sets, in percent.*

	Average	Worst	Best	StD	LHS$_{best}$	StD	SPO	StD
ESVMs								
Diabetes (overall)	**76.30**	71.35	80.73	2.24	75.82	3.27	**77.31**	2.45
Diabetes (ill)	50.81	39.19	60.27	4.53	49.35	7.47	52.64	5.32
Diabetes (healthy)	90.54	84.80	96.00	2.71	89.60	2.36	90.21	2.64
Iris (overall)	**95.18**	91.11	100.0	2.48	95.11	2.95	**95.11**	2.95
Soybean (overall)	**99.02**	94.11	100.0	2.23	99.61	1.47	**99.80**	1.06
Spam filtering (overall)	**87.74**	85.74	89.83	1.06	89.27	1.37	**90.59**	0.98
Spam filtering(spam)	77.48	70.31	82.50	2.77	80.63	3.51	83.76	2.21
Spam filtering (non spam)	94.41	92.62	96.30	0.89	94.82	0.94	95.06	0.62
ESVMs with Chunking								
Spam filtering (overall)	**87.30**	83.13	90.00	1.77	87.52	1.31	**88.37**	1.15
Spam filtering(spam)	83.47	75.54	86.81	2.78	86.26	2.66	86.35	2.70
Spam filtering (non spam)	89.78	84.22	92.52	2.11	88.33	2.48	89.68	2.06
Modified ESVMs								
Spam filtering (overall)	**88.40**	86.52	90.35	1.02	90.06	0.99	**91.25**	0.83
Spam filtering(spam)	79.63	75.39	84.70	2.16	82.73	2.28	85.52	2.08
Spam filtering (non spam)	94.17	91.34	95.84	1.05	94.89	0.92	95.00	0.72

5 Improving Training Time

Obtained results for the tasks we have undertaken to solve have proven to be competitive as compared to accuracies of different powerful classification techniques. However, for large problems, i.e. spam filtering, the amount of runtime needed for training is $\approx$ 800s. This stems from the large genomes employed, as indicators for errors of every sample in the training set are included in the representation. Consequently, we tackle this problem with two approaches: an adaptation of a chunking procedure inside ESVMs and a modified version of the evolutionary approach.

5.1 Reducing Samples for Large Problems

We propose a novel algorithm to reduce the number of samples for one run of ESVMs which is an adaptation of the widely known shrinking technique within SVMs, called chunking [15]. In standard SVMs, this technique implies the identification of Lagrange multipliers that denote samples which do not fulfill restrictions. As we renounced the standard solving of the optimization problem within SVMs for the EA-based approach, the chunking algorithm was adapted to fit our technique (Algorithm 1).

ESVM with chunking was applied to the spam data set. Values for parameters were set as before, except the number of generations for each EA which is now set to 100. The chunk size, i.e N, was chosen as 200 and the number of iterations with no improvement was designated to be 5. Results are shown in Table 2. Average runtime was of 103.2s/run. Therefore, the novel algorithm of ESVM with chunking reached its goal, running 8 times faster than previous one, at a cost of loss in accuracy of 0.4%.

Algorithm 1 ESVM with Chunking

```
Randomly choose N samples from the training data set, equally distributed, to make a chunk;
while a predefined number of iterations passes with no improvement do
    if first chunk then
        Randomly initialize population of a new EA;
    else
        Use best evolved hyperplane coefficients and random indicators for errors to fill half of
        the population of a new EA and randomly initialize the other half;
    end if
    Apply EA and find coefficients of the hyperplane;
    Compute side of all samples in the training set with evolved hyperplane coefficients;
    From incorrectly placed, randomly choose (if available) N/2 samples, equally distributed;
    Randomly choose the rest up to N from the current chunk and add all to the new chunk
    if obtained training accuracy if higher than the best one obtained so far then
        Update best accuracy and best evolved hyperplane coefficients; set improvement to true;
    end if
end while
Apply best obtained coefficients on the test set and compute accuracy
```

5.2 A Reconsideration of the Evolutionary Algorithm

Since ESVMs directly provide hyperplane coefficients at all times, we propose to drop the indicators for errors from the EA representation and, instead, compute their values in a simple geometrical fashion. Consequently, this time, individual representation contains only w and b. Next, all indicators ξ_i, $i = 1, 2, ..., m$ are computed in order to be referred in the fitness function (5). The current individual (which is the current separating hyperplane) is considered and, following [1], supporting hyperplanes are determined. Then, for every training vector, deviation to the corresponding supporting hyperplane, following its class, is calculated. If sign of deviation equals class, corresponding $\xi_i = 0$; else, deviation is returned. The EA proceeds with the same values for parameters as in Table 1 (certainly except probabilities and step sizes for the ξ_is) and, in the end of the run, hyperplane coefficients are again directly acquired. Empirical results on the spam data set (Table 2) and the average runtime of 600s seem to support the new approach. It is interesting to remark that the modified algorithm is not that much faster, but provides some improvement in accuracy. In contrast to the chunking approach, it also seems especially better suited for achieving high non spam recognition rates, in order to prevent erroneous deletion of good mail. Surprisingly, SPO here decreases effort while increasing accuracies (Table 1), resulting in further speedup.

6 Conclusions and Future Work

Proposed new hybridized learning technique incorporates the vision upon classification of SVMs but solves the inherent optimization problem by means of evolutionary algorithms. ESVMs present many advantages as compared to SVMs. First of all, they are definitely much easier to understand and use. Secondly and more important, the evolutionary solving of the optimization problem enables the acquirement of hyperplane coefficients directly and at all times within a run. Thirdly, accuracy on several benchmark real-world problems is comparable to those of state-of-the-art SVM methods or to results of other powerful techniques from different other machine learning fields. In order to enhance suitability of the new technique for any classification issue, two novel mechanisms for reducing size in large problems are also proposed; obtained results support their employment.

Although already competitive, the novel ESVMs for classification can still be improved. Other kernels may be found and used for better separation; also, the two appointed classical kernels may have parameters evolved by means of evolutionary algorithms as in some approaches for model selection. Also, other preprocessing techniques can be considered; feature selection through an evolutionary algorithm as found in literature can surely boost accuracy and runtime. Definitely, other ways to handle large data sets can be imagined; a genetic algorithm can also be used for sample selection. Finally, the construction of ESVMs for regression problems is a task for future work.

References

1. Bosch, R.A., Smith, J.A.: Separating Hyperplanes and the Authorship of the Disputed Federalist Papers. American Mathematical Monthly, Vol. 105, No. 7, (1998), 601-608
2. Vapnik, V., Statistical Learning Theory. Wiley, New York (1998)
3. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall, New Jersey (1999)
4. Friedrichs, F., Igel, C.: Evolutionary tuning of multiple SVM parameters, In Proc. 12th European Symposium on Artificial Neural Networks (2004) 519-524
5. Feres de Souza, B., Ponce de Leon F. de Carvalho, A.: Gene selection based on multi-class support vector machines and genetic algorithms. Journal of Genetics and Molecular Research, Vol. 4, No. 3 (2005) 599-607
6. Eads, D., Hill, D., Davis, S., Perkins, S., Ma, J., Porter, R., Theiler, J.: Genetic Algorithms and Support Vector Machines for Time Series Classification. 5th Conf. on the Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation, Proc. Symposium on Optical Science and Technology, SPIE, Seattle, WA, 4787 (2002) 74-85
7. Hsu, C.-W., Lin, C.-J.: A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks, Vol. 13, No. 2 (2002) 415-425
8. Bartz-Beielstein, T.: Experimental research in evolutionary computation - the new experimentalism. Natural Computing Series, Springer-Verlag (2006)
9. Joachims, T.: Making Large-Scale Support Vector Machine Learning Practical. In Advances in Kernel Methods: Support Vector Learning (1999) 169-184
10. Raicharoen, T. and Lursinsap, C.: Critical Support Vector Machine Without Kernel Function. In Proc. 9th Intl. Conf. on Neural Information Processing, Singapore, Vol. 5 (2002) 2532-2536
11. Tax, D. M. J.: DDtools, the data description toolbox for Matlab (2005) http://www-ict.ewi.tudelft.nl/ davidt/occ/index.html
12. Sierra, A., Corbacho, F.: Input and Output Feature Selection. ICANN 2002, LNCS 2415 (2002) 625-630
13. Weston, J., Watkins, C.: Multi-class Support Vector Machines. Technical Report, CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science (1998)
14. Bailey, J., Manoukian, T., Ramamohanarao, K.: Classification Using Constrained Emerging Patterns. In Proc. of WAIM 2003 (2003) 226-237
15. Perez-Cruz, F., Figueiras-Vidal, A. R., Artes-Rodriguez, A.: Double chunking for solving SVMs for very large datasets. Learning'04, Elche, Spain (2004) eprints.pascal-network.org/archive/00001184/01/learn04.pdf.