Concerning the Potential of Evolutionary Support Vector Machines

Ruxandra Stoean Mike Preuss Catalin Stoean D. Dumitrescu

Abstract— **Within the present paper, we put forward a novel hybridization between support vector machines and evolutionary algorithms. Evolutionary support vector machines consider the classification task as in support vector machines but use an evolutionary algorithm to solve the optimization problem of determining the decision function. They can explicitly acquire the coefficients of the separating hyperplane, which is often not possible within the classical technique. More important, evolutionary support vector machines obtain the coefficients directly from the evolutionary algorithm and can refer them at any point during a run. In addition, they do not require properties of positive (semi-)definition for kernels within non-linear learning. The concept can be furthermore extended to handle large amounts of data, a problem frequently occurring e.g. in spam mail detection, one of our test cases. An adapted chunking technique is therefore alternatively used. In addition to two different representations, a crowding variant of the evolutionary algorithm is tested in order to investigate whether the performance of the algorithm is maintained; its global search capabilities would be important for the prospected coevolution of non-standard kernels. Evolutionary support vector machines are validated on four real-world classification tasks; obtained results show the promise of this new approach.**

I. Introduction

Support vector machines (SVMs) represent a state-of-the-art learning technique that has managed to reach 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). This is not the first attempt to hybridize SVMs and EAs. Existing alternatives are discussed in §III-B. Nevertheless, we claim that our approach is significantly different.

ESVMs as presented here are constructed solely based on SVMs applied for classification. Two kinds of possible representations are considered. 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. The behaviour of a crowding-based EA on preserving the performance of the algorithm is examined with the purpose of the its future employment for the coevolution of non-standard kernels. Obtained results prove suitability and competitiveness of the new approach, so ESVMs qualify as a viable simpler alternative to standard SVMs in this context. However, there is still room for improvement.

The paper is organized as follows. §2 outlines the concepts of classical SVMs. §3 describes the existing evolutionary approaches aimed at enhancing the performance of the classical architecture and explains the concept of the new hybridization. §4 puts forward a first considered representation of an EA inside proposed ESVMs. Validation is achieved on real world examples. An alternative version of the ESVM which is endowed with a new mechanism for reducing problem size in case of large data sets is also presented. §5 brings a second, simpler, representation of the proposed EA, which also speeds up runtime; the crowding variant is subsequently illustrated. §6 exhibits the comparison with results of canonical SVMs implemented in R on the same data sets and in equivalent conditions. The last section comprises conclusions and outlines ideas for further improvement.

II. 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 sample 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 samples and the actual classes produced by the learning machine. Finally, accuracy of the machine is computed on previously unseen test data samples. In the classical architecture, SVMs reduce k-class classification problems to many binary classification tasks that are separately considered and solved. Different systems then decide the class for data samples 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. the determination of its coefficients.

A. Support Vector Machines

If training data is known to be linearly separable, then there exists a linear hyperplane that performs the partition, i.e.

R. Stoean is with the Department of Computer Science, Faculty of Mathematics and Computer Science, University of Craiova, Str. Al. I. Cuza, No. 13, 200585, Craiova, Romania (e-mail: ruxandra.stoean@inf.ucv.ro).

M. Preuss is with the Chair of Algorithm Engineering, Department of Computer Science, University of Dortmund, Otto-Hahn-Str, No. 14, 44221, Dortmund, Germany, (e-mail: mike.preuss@uni-dortmund.de).

C. Stoean is with the the Department of Computer Science, Faculty of Mathematics and Computer Science, University of Craiova, Str. Al. I. Cuza, No. 13, 200585, Craiova, Romania (e-mail: catalin.stoean@inf.ucv.ro).

D. Dumitrescu is with the Department of Computer Science, Faculty of Mathematics and Computer Science, Babes-Bolyai University, Str. M. Kogalniceanu, No. 1, 400084, Cluj, Romania (e-mail: ddumitr@cs.ubbcluj.ro).

$\langle w, x\rangle - b = 0$, 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 samples 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 samples each lie on the corresponding side of a matching supporting hyperplane for the respective class (Figure 1a) [2], written in brief as (1):

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

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 [15], separation must be performed with a maximal margin between classes. Summing up, classification of linear separable data with a linear hyperplane through SVMs leads to the following optimization problem (2):

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

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 samples 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 sample 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 (3):

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

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 samples 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 therefore leads 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 samples 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 $\langle\Phi(w), \Phi(x_i)\rangle - b = 0$.

As in the training algorithm vectors appear only as part of dot products, the issue can be further on simplified by substituting it by what is called a kernel, i.e. a function with the property that $K(x, y) = \langle\Phi(x), \Phi(y)\rangle,$ where $x, y \in R^n$. SVMs request that the kernel is a positive (semi-)definite function in order for the standard solving approach to find a solution to the optimization problem [12]. Such a kernel is one that satisfies Mercer's theorem from functional analysis and is, therefore, a dot product in some space [3].

The problem with this restriction is twofold [12]. On the one hand, Mercer's condition is very difficult to check for a newly constructed kernel. On the other hand, kernels that fail the theorem could prove to achieve a better separation of the training samples.

Applied SVMs consequently use 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 the place of the canonical solving of the optimization problem, a direct search algorithm would however disregard whether the kernel is positive (semi-)definite or not.

In conclusion, classification of linear nonseparable data with a nonlinear hyperplane through SVMs leads to the same optimization problem as in (3) which is now considered in the feature space and with the use of a kernel function (4):

$$\begin{cases} \text{find } w \text{ and } b \text{ as to minimize } \frac{K(w,w)}{2} + C\sum_{i=1}^{m}\xi_i, \\ C > 0 \\ \text{subject to } y_i(K(w, x_i) - b) \geq 1 - \xi_i, \xi_i \geq 0, \\ i = 1, 2, ..., m. \end{cases} \tag{4}$$

This generalized formulation is called the primal problem.

After the optimization problem is solved, the class of every test sample is calculated, i.e. the side of the decision boundary on which every new data sample lies is determined (5):

$$class(x) = sgn(K(w, x) - b). \tag{5}$$

It is seldom the case that coefficients can be explicitly determined following the standard solving of the primal problem, as the map Φ cannot be always specifically determined. In this situation, the class for a new sample follows from computational artifices.

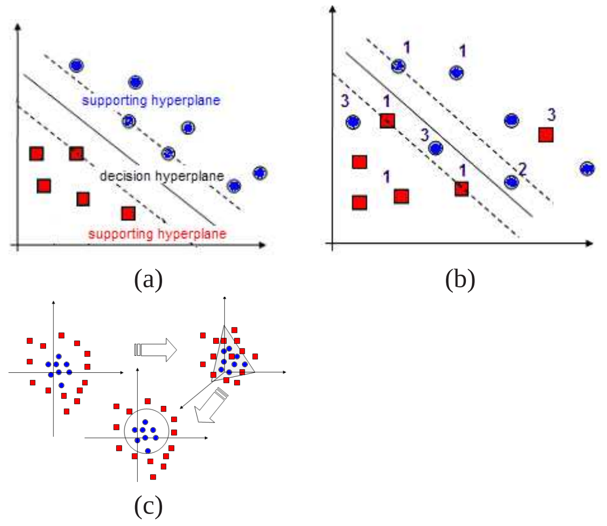

Fig. 1. (a) Decision hyperplane (continuous line) that separates between circles (positive) and squares (negative) and supporting hyperplanes (dotted lines). (b) Position of data and corresponding indicators for errors - correct placement, $\xi_i = 0$ (label 1) margin position, $\xi_i < 1$ (label 2)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

B. Multi-class Support Vector Machines

Multi-class SVMs build several 2-class classifiers that separately solve the matching tasks. Resulting decision functions are then considered as a whole and the class for each sample in the test set is decided by different systems, i.e. one-against-all, one-against-one or decision directed acyclic graph.

1) One-against-all: The one-against-all (1aa) technique [10] builds k classifiers. Every i^{th} SVM considers all training samples labelled with i as positive and all the remaining as negative.

Consequently, the aim of every i^{th} SVM is to determine the optimal coefficients, w and b, of the decision hyperplane which best separates the samples with outcome i from all the other samples in the training set, such that (6):

$$\begin{cases} \text{minimize } \frac{K(w^i,w^i)}{2} + C\sum_{j=1}^{m}\xi_j^i, \\ \text{subject to } y_j(K(w^i,x_j)-b) \geq 1-\xi_j^i, \\ \xi_j^i \geq 0 \\ j=\overline{1,m}, i=\overline{1,k}. \end{cases} \tag{6}$$

Once the coefficients w^i and b^i of all k hyperplanes are determined, the following decision system to label new test data is employed. The class for a test sample x is given by the category that has the maximum value for the learning function, as in (7):

$$class(x) = argmax_{i=1,2,\ldots,k}(K(w^i,x)-b^i). \tag{7}$$

2) One-against-one: The one-against-one (1a1) technique [10] builds $\frac{k(k-1)}{2}$ SVMs. Every i^{th} machine is trained on data from every two classes, *i* and *j*, where samples labelled with *i* are considered positive while those in class j are taken as negative.

Accordingly, the aim of every SVM is to determine the optimal coefficients, w and b, of the decision hyperplane which best separates the samples with outcome i from the samples with outcome j, such that (8):

$$\begin{cases} \text{minimize } \frac{K(w^{ij},w^{ij})}{2} + C\sum_{l=1}^{m}\xi_l^{ij}, \\ \text{subject to } y_l(K(w^{ij},x_l)-b) \geq 1-\xi_l^{ij}, \\ \xi_l^{ij} \geq 0 \\ l=\overline{1,m}, i,j=\overline{1,k}, i\neq j. \end{cases} \tag{8}$$

Once the coefficients of the $\frac{k(k-1)}{2}$ SVMs are found, a *voting* method is used to determine the class for a test sample *x*. For every SVM, the class of each sample *x* is computed by following the sign of the corresponding decision function applied to x. Subsequently, if the sign says *x* is in class *i*, the vote for the *i*-th class is incremented by one; conversely, the vote for class *j* is increased by unity. 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.

3) Decision Directed Acyclic Graph: Training in the decision directed acyclic graph (DDAG) technique [14] happens in an identical manner to that of 1a1.

Once the coefficients of the $\frac{k(k-1)}{2}$ SVMs are evolved, the following graph system is used to determine the class for a test sample *x* (Figure 2). Each node of the graph has a list of classes attached and considers the first and last elements of the list. The list that corresponds to the root node contains all k classes. When we evaluate a test instance x we descend from node to node, i.e. we eliminate one class from each corresponding list, until the leaves are reached.

The algorithm starts at the root node which considers the first and last classes. At each node, i vs j, we refer to the SVM that was trained on data from classes i and j. The class of *x* is computed by following the sign of the corresponding decision function applied to x. Subsequently, if the sign says *x* is in class *i*, the node is exited via the right edge; conversely, we exit through the left edge. We thus eliminate the wrong class from the list and proceed via the corresponding edge to test the first and last classes of the new list and node. The class is given by the leaf that x eventually reaches.

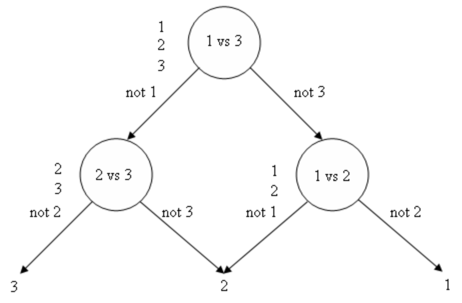

Fig. 2. DDAG for labelling a test sample in 3-class problems

C. The Canonical Solving of the Primal Problem

Classical SVMs approach the optimization problem that is reached through a dualization method utilizing Lagrange multipliers [8]. Nevertheless, the mathematics of the technique can be found to be very difficult both to grasp and apply. As a consequence, a new approach that would simplify (and improve) the solving of the optimization problem would be desirable.

III. Evolutionary Support Vector Machines

In the following, we develop a new straightforward hybridization of SVMs and EAs, put it into context with other existing combinations of these two paradigms, and finally verify it by targeted experimentation.

A. Main Question and Aim

Can we provide an evolutionary solution to the primary optimization problem?
We put forward a new hybridized approach where separation of positive and negative samples proceeds as in standard SVMs, while the optimal values for the coefficients of the separating hyperplane (w and b) are directly determined by an EA. 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 general form of the optimization problem (4) within SVMs. Although the suggested representation appears to be straightforward, determining several other algorithm details (operators, parameters) is not; these are to be explored. In other words, we target at attaining a suitable EA for solving the primal problem of finding the separation hyperplane.

B. Evolutionary Approaches to Support Vector Machines

Note that this is not the first attempt to hybridize SVMs and EAs. Existing alternatives are numerous and recent, of which some are presented further on. Their combination envisaged four different directions: model and feature selection, kernel evolution and evolutionary detection of the Lagrange multipliers. Model selection concerns adjustment of hyperparameters (free parameters) within SVMs, i.e. the penalty for errors and parameters of the kernel which, in standard variants, is performed through grid search or gradient descent methods. Evolution of hyperparameters can be achieved through evolution strategies [7]. 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 [6] and genetic programming [5]. Evolution of kernel functions to model training data is performed by means of genetic programming [9]. Finally, the Lagrange multipliers involved in the expression of the dual problem can be evolved by means of evolution strategies and particle swarm optimization [11]. To the best of our knowledge, evolution of the coefficients of the decision function within SVMs has not been accomplished yet.

IV. A Naive Representation

A. Research question

How do we refer errors (slack variables) in the optimization problem? Evolve them?

B. Preexperimental Planning

First 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[1], 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, diabetes diagnosis is a better-known benchmark, but spam 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, while soybean has a high number of attributes, iris has only four, but a larger number of samples. 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 is given in Table I.

C. Task

We want to evaluate whether the suggested hybrid ESVM with evolved errors produces competitive classifiers if compared to standard approaches (results for these are given in §6) and how appropriate parameters will be chosen.

D. Algorithm Setup

1) Representation: An individual encodes the coefficients of the separating hyperplane, w and b. Since indicators for errors of classification, $\xi_i,\ i = 1, 2, ..., m$, appear in the conditions for hyperplane optimality, ESVMs may handle them through inclusion in the structure of an individual, as well (9):

$$c = (w_1, ..., w_n, b, \xi_1, ..., \xi_m). \tag{9}$$

After termination of the algorithm, the approximately optimal values for the coefficients of the decision hyperplane are obtained.
2) Initial population: Individuals are randomly generated such that $w_i \in [-1, 1], i = 1, 2, ..., n,\ b \in [-1, 1]$ and $\xi_j \in [0, 1], j = 1, 2, ..., m$.
3) Fitness assignment: 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 through 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 (10):

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

$$f(w,b,\xi) = K(w,w) + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}[t(y_i(K(w,x_i)-b)-1+\xi_i)]^2 \quad (10)$$

4) Search and variation 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.

5) 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 sample can be determined directly following (5).

E. Problem Setup

For each data set, 30 runs of the ESVM were conducted; in every run approx. 70% random cases were appointed to the training set and the remaining 30% went into test. Experiments showed the necessity for data normalization in diabetes, spam and iris. No further modification of the data was carried out and all data was used in the experiments.

SVM and EA parameter values are given in the naive representation section of Table I. As training in DDAG is identical to that of the 1a1 multi-class approach, the same parameter values are employed in either situation and we therefore refer only the latter in the table.

The error penalty was invariably set to 1. Since this is the first ESVM approach, we have used only the traditional polynomial and radial kernels.

TABLE I

DATA SET PROPERTIES AND MANUALLY TUNED PARAMETER VALUES.

	Diabetes	Iris 1a1/1aa	Soybean	Spam
Data				
Number of samples	768	150	47	4601
Number of attributes	8	4	35	57
Number of classes	2	3	4	2
Naive rep.				
p or σ	$p = 2$	$\sigma = 1$	$p = 1$	$p = 1$
Population size	100	100/100	100	100
Generations	250	100/100	100	250
Crossover prob.	0.40	0.30/0.70	0.30	0.30
Mutation prob.	0.40	0.50/0.50	0.50	0.50
ξ mutation prob.	0.50	0.50	0.50	0.50
Mutation strength	0.10	0.10/4	0.10	0.10
ξ mutation strength	0.10	0.10/0.10	0.10	0.10
Hyperplane rep.				
Population size	100	100/100	100	150
Generations	250	100/100	100	300
Crossover prob.	0.4	0.30/0.70	0.30	0.80
Mutation prob.	0.4	0.50/0.50	0.50	0.50
Mutation strength	0.1	4/4	0.1	3.5

In order to validate the manually found EA parameter values, the parameter tuning method SPO [1] was applied with a budget of 1000 optimization runs. Parameter bounds were set as follows:

- Population size - 10/200
- Number of generations - 50/300
- Crossover probability - 0.01/1
- Mutation probability - 0.01/1
- Error mutation probability - 0.01/1
- Mutation strength - 0.001/5
- Error mutation strength - 0.001/5

Since the three multi-class techniques behave similarly in all our manual multi-class experiments (Table III), we run automatic tuning only for the most widely used case of 1a1. Parameter settings of the best parameter configurations as determined by SPO are depicted in the naive representation section of Table II.

TABLE II

SPO TUNED PARAMETER VALUES

	Diabetes	Iris	Soybean	Spam	Spam
Naive representation					**+Chunks**
Population size	198	46	162	154	90
Generations	296	220	293	287	286
Crossover prob.	0.87	0.77	0.04	0.84	0.11
Mutation prob.	0.21	0.57	0.39	0.20	0.08
ξ mutation prob.	0.20	0.02	0.09	0.07	0.80
Mutation strength	4.11	4.04	0.16	3.32	0.98
ξ mutation strength	0.02	3.11	3.80	0.01	0.01
Hyperplane representation					
Population size	190	17	86	11	
Generations	238	190	118	254	
Crossover prob.	0.13	0.99	0.26	0.06	
Mutation prob.	0.58	0.89	0.97	0.03	
Mutation strength	0.15	3.97	0.08	2.58	
Hyperplane representation with Crowding					
Population size	92	189	166	17	
Generations	258	52	201	252	
Crossover prob.	0.64	0.09	0.77	0.42	
Mutation prob.	0.71	0.71	0.92	0.02	
Mutation strength	0.20	0.20	0.04	4.05	

F. Results

Test accuracies obtained by manual tuning are presented in the naive representation section of Table III. Differentiated (spam/non spam for spam filtering and ill/healthy for diabetes) accuracies are also depicted. The naive representation section of Table IV holds performances and standard deviations of the best configuration of an initial latin hypersquare design (LHD) sample of size 4×100.

G. Observations

SPO indicates that for all cases, except for the soybean data, crossover probabilities were dramatically increased, while often reducing mutation probabilities, especially for errors. However, the relative quality of SPO's final best configurations against the ones found during the initial LHD phase increases with the problem size. It must be stated that in most cases, results achieved with manually determined parameter values are only improved by SPO – if at all – by increasing effort (population size or number of generations). A problem appears for large data sets, i.e. spam filtering, where the amount of runtime needed for training is very large. 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

TABLE III

ACCURACIES OF DIFFERENT MANUALLY TUNED ESVM VERSIONS ON THE CONSIDERED TEST SETS, IN PERCENT.

	Average	**Worst**	**Best**	**StD**
Naive representation				
Diabetes (overall)	**76.30**	71.35	80.73	2.24
Diabetes (ill)	50.81	39.19	60.27	4.53
Diabetes (healthy)	90.54	84.80	96.00	2.71
Iris 1aa (overall)	**95.85**	84.44	100.0	3.72
Iris 1a1 (overall)	**95.18**	91.11	100.0	2.48
Iris DDAG (overall)	**94.96**	88.89	100.0	2.79
Soybean 1aa (overall)	**99.22**	88.24	100	2.55
Soybean 1a1 (overall)	**99.02**	94.11	100.0	2.23
Soybean DDAG (overall)	**98.83**	70.58	100	5.44
Spam (overall)	**87.74**	85.74	89.83	1.06
Spam (spam)	77.48	70.31	82.50	2.77
Spam (non spam)	94.41	92.62	96.30	0.89
ESVMs with Chunking				
Spam (overall)	**87.30**	83.13	90.00	1.77
Spam (spam)	83.47	75.54	86.81	2.78
Spam (non spam)	89.78	84.22	92.52	2.11
Hyperplane representation				
Diabetes (overall)	**74.60**	70.31	82.81	2.98
Diabetes(ill)	45.38	26.87	58.57	6.75
Diabetes (healthy)	89.99	86.89	96.75	2.66
Iris 1aa (overall)	**93.33**	86.67	100	3.83
Iris 1a1 (overall)	**95.11**	73.33	100	4.83
Iris DDAG (overall)	**95.11**	88.89	100	3.22
Soybean 1aa (overall)	**99.22**	88.24	100	2.98
Soybean 1a1 (overall)	**99.60**	94.12	100	1.49
Soybean DDAG (overall)	**99.60**	94.12	100	1.49
Spam (overall)	**86.19**	82	100	3.12
Spam (spam)	70.54	62.50	77.80	4.55
Spam (non spam)	95.39	92.66	97.44	1.09

TABLE IV

ACCURACIES OF DIFFERENT SPO ESVM VERSIONS ON THE CONSIDERED TEST SETS, IN PERCENT.

	LHD$_{best}$	**StD**	**SPO**	**StD**
Naive representation				
Diabetes (overall)	75.82	3.27	**77.31**	2.45
Diabetes (ill)	49.35	7.47	52.64	5.32
Diabetes (healthy)	89.60	2.36	90.21	2.64
Iris (overall)	**95.11**	2.95	**95.11**	2.95
Soybean (overall)	99.61	1.47	**99.80**	1.06
Spam (overall)	89.27	1.37	**90.59**	0.98
Spam (spam)	80.63	3.51	83.76	2.21
Spam (non spam)	94.82	0.94	95.06	0.62
ESVMs with Chunking				
Spam (overall)	87.52	1.31	**88.37**	1.15
Spam (spam)	86.26	2.66	86.35	2.70
Spam (non spam)	88.33	2.48	89.68	2.06
Hyperplane representation				
Diabetes (overall)	72.50	2.64	**73.39**	2.82
Diabetes(ill)	35.50	10.14	43.20	6.53
Diabetes (healthy)	92.11	4.15	89.94	3.79
Iris (overall)	**95.41**	2.36	**95.41**	2.43
Soybean (overall)	**99.61**	1.47	99.02	4.32
Spam (overall)	89.20	1.16	**89.51**	1.17
Spam (spam)	79.19	3.13	82.02	3.85
Spam (non spam)	95.64	0.90	94.44	1.42
Hyperplane representation with Crowding				
Diabetes (overall)	74.34	2.30	**74.44**	2.98
Diabetes(ill)	43.68	6.64	45.32	7.04
Diabetes (healthy)	90.13	3.56	90.17	3.06
Iris (overall)	**95.63**	2.36	94.37	2.80
Soybean (overall)	99.61	2.11	**100**	0.00
Spam (overall)	88.72	1.49	**89.45**	0.97
Spam (spam)	80.14	5.48	80.79	3.51
Spam (non spam)	94.25	1.66	95.07	1.20

problem with an adaptation of a chunking procedure [13] inside ESVMs.

A chunk of N training samples is repeatedly considered. Within each chunking cycle, the EA (with a population of half random individuals and half previously best evolved individuals) runs and determines the coefficients of the hyperplane. All training samples are tested against the obtained decision function and a new chunk is constructed based on N/2 randomly (equally distributed) incorrectly placed samples and half randomly samples from the current chunk. The chunking cycle stops when a predefined number of iterations with no improvement in training accuracy passes.

ESVM with chunking was applied to the spam data set. Manually tuned parameters had the same values as before, except the number of generations for each run of the 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 (repeats of the chunking cycle) was designated to be 5. Values derived from the SPO tuning are presented in the ESVMs chunking section of Table II.

Results of manual and SPO tuning are shown in the ESVMs chunking sections of Tables III and IV. The novel algorithm of ESVM with chunking reached its goal, running 8 times faster than the previous one, at a cost of a small loss in accuracy.

Besides solving the EA genome length problem, proposed mechanism additionally reduces the large number of computations that derives from the reference to the many training samples in the expression of the fitness function.

H. Discussion

Obtained results for the classification tasks we have undertaken to solve have proven to be competitive as compared to accuracies of the canonical SVMs on the same test problems (see §6). Note that discrete result values lead to high standard deviations, limiting the use of hypothesis tests. Only for the largest problem (spam), standard deviation intervals of algoritm variants do not always overlap. For SPO, a similar hardness occurs: Distinguishing the performance of different configurations is difficult even after computing a large number of repeats. Consequently, the 'parameter optimization potential' justifies employing a tuning method only for the larger problems, diabetes and spam. Especially for the small problems, well performing parameter configurations are seemingly easy to find.

It must be stated that for the standard kernels, we cannot expect ESVMs to be better than standard SVMs. However, in future work, we can profit from the flexibility of the EAs as optimization tools, by being able to additionally evolve kernels that achieve a better separation, regardless of whether they are positive (semi-)definite or not.

V. HYPERPLANE REPRESENTATION

Although already a viable alternative approach, the ESVMs may still be improved concerning simplicity.

A. Research question

The current optimization problem requires to treat the error values, which in the present EA variant are included in the representation. These can be expected to severely complicate the problem by increasing the genome length (variable count) by the number of training samples. We propose to tackle this issue by a reconsideration of the elements of the EA as follows. Can we represent only the hyperplane coefficients and compute the errors instead of evolving them?

B. Preexperimental Planning

For reasons of comparison between the two representations, we keep the same data sets for application.

C. Task

It will be investigated if a representation without errors can perform as well as the naive representation of §4.

D. Algorithm Setup

Since ESVMs directly and interactively 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, as in (11):

$$c = (w_1, ..., w_n, b). \tag{11}$$

Additionally, all indicators $\xi_i,\ i = 1, 2, ..., m$ will have to be computed in order to be referred in the fitness function (10), which remains as before. The procedure follows [2] and takes place as follows.
The current individual (which is the current separating hyperplane) is considered and supporting hyperplanes are determined through the mechanism below. One first computes (12):

$$\begin{cases} m_1 = min\{K(w, x_i)|y_i = +1\} \\ m_2 = max\{K(w, x_i)|y_i = -1\} \end{cases} \tag{12}$$

Then, we set (13):

$$\begin{cases} p = |m_1 - m_2| \\ w' = \frac{2}{p}w \\ b' = \frac{1}{p}(m_1 + m_2) \end{cases} \tag{13}$$

Finally, for every training sample x, deviation to the corresponding supporting hyperplane, following its class, is calculated, as in (14):

$$deviation(x_i) = \begin{cases} K(w', x_i) - b' - 1, y_i = +1, \\ K(w', x_i) - b' + 1, y_i = -1, \\ i = 1, 2, ..., m. \end{cases} \tag{14}$$

If sign of deviation equals class, corresponding $\xi_i = 0$; else, the (normalized) absolute deviation is returned as the indicator for error. Experiments showed the need for normalization of the computed deviations in the cases of diabetes, spam and iris, while, on the contrary, soybean requires without. The different behaviour can be explained by the fact that the first three data sets have a larger number of training samples. The sum of the deviations is subsequently added to the expression of the fitness function. As a consequence, in the early generations, when the generated coefficients lead to high deviations, their sum, considered from 1 to the number of training samples, takes over the whole fitness value and the evolutionary process is driven off the course to the optimum. The discussed diverse choices of actions concerning the normalization of data and errors and the kernel selection bring experimental evidence for the crucial importance of proper data preparation parameters prior to the actual application of SVM learning.
In addition to the different representation, we also test a crowding [4] variant of the EA. Here, test for replacement is done against the most similar parent of the current population. Crowding based EAs are known to provide good global search capabilities. This is of limited value for the kernel types employed in this study, but it is important for non-standard kernels. For now however, we want to investigate whether the employment of a crowding-based EA on the hyperplane representation would worsen the performance of the algorithm or not. All the other elements of the EA remain the same.
The EA proceeds with the values for parameters from the hyperplane representation section in Table I and, in the end of the run, hyperplane coefficients are again directly acquired. Resulting parameter values for SPO on the hyperplane and the hyperplane crowding variant are shown in the appropriate sections of Table II. Note that only automated tuning is performed for the hyperplane crowding ESVM.

E. Problem Setup

The problem related settings are kept the same as for the naive representation.

F. Results

Manual and SPO tuning based results are depicted in the hyperplane and hyperplane with crowding representation sections of Tables III and IV. The given SPO performance values are generated by (30) validation runs for the best found configurations after initial design and after finishing SPO, respectively.

G. Observations

Automated tuning revealed that for the crowding variant, some parameter interactions dominate the best performing configurations (not depicted due to space limitations): For larger population size, smaller mutation step sizes and larger crossover probability are better suited, and with greater run lengths, performance increases with larger mutation step sizes. For the original hyperplane variant, no such clear interactions can be attained. However, in both cases, many good configurations are detected.

H. Discussion

It is interesting to remark that the hyperplane representation is not that much faster. Although the genome length is drastically reduced, the runtime consequently gained is however partly lost again when computing the values for the slack variables. This draws from the extra number of dot products that must be calculated due to (12) and (14). As run length itself is a parameter in our studies, we rather obtain an upper bound of the necessary effort. Closer investigation may lead to a better understanding of suitable run lengths, e.g. in terms of fitness evaluations. However, the hyperplane representation has its advantages. Besides featuring smaller genomes, less parameters are needed, because the slack variables are not evolved and thus 2 parameters vanish.

The best configurations for hyperplane representation with and without crowding perform similarly and not significantly worse as compared to the results recorded for the naive representation, except for the diabetes test case, where they are weaker. Parameter tuning beyond a large initial design appears to be infeasible, as performance is not significantly improved in most cases. If at all, it is successful for the larger problems of diabetes and spam. This indicates that parameter setting for the ESVMs is rather easy, because there is a large set of good performing configurations. Nevertheless, there seems to be a slight tendency towards fewer good configurations (harder tuning) for the large problems.

VI. Comparison to Canonical Support Vector Machines

In order to make a direct comparison with the results of the canonical SVMs, we used the R environment and available related packages (e1071, mlbench and kernlab) for applying them to all considered data sets. 1a1 SVM is the only existing R implementation for multi-class tasks. The results, obtained after 30 runs, are illustrated in Table V. For the iris data set a radial kernel was used, while for the other three a polynomial one was employed. After performing manual tuning for the SVM parameters, the best results were obtained for $C = 1$, $\sigma = 1$ and $p = 1$ in all corresponding cases.

TABLE V

Accuracies of canonical SVMs on the considered test sets, in percent, as opposed to those obtained by ESVMs.

	Average	Worst	Best	StD	ESVMs Average
Diabetes	**76.82**	73.95	81.77	1.84	**77.31**
Iris	**95.33**	88.0	100.0	3.16	**95.63**
Spam	**92.67**	91.56	93.91	0.64	**90.59**
Soybean	**92.22**	66.67	100.00	9.60	**100**

VII. Conclusions and Future Work

The proposed new hybridized learning technique incorporates the vision upon classification of SVMs but solves the inherent optimization problem by means of an EA. As opposed to SVMs, ESVMs are definitely much easier to understand and use. Moreover, the evolutionary solving of the optimization problem enables the acquirement of hyperplane coefficients directly and at all times within a run. At the same time, performance remains comparable to that of canonical SVMs. Two possible representations (one simpler, and a little faster, and one more complicated, but also more accurate) for the EA that determines the coefficients are imagined. In order to enhance suitability of the new technique for any classification issue, a novel chunking mechanism for reducing size in large problems is also proposed; obtained results support its employment. Finally, the use of a crowding-based EA is inspected in relation to the preservation of the performance. Crowding would be highly necessary in the immediate coevolution of non-standard kernels.

Although already competitive, the novel ESVMs for classification can still be enhanced. The way of treating the two criteria (i.e. reduce errors and obtain a flat function) through proposed fitness evaluation may not be the best choice; instead, we could try a multicriterial approach. Additionally, we will achieve the simultaneous evolution of the hyperplane and of non-standard kernels. This approach is highly difficult by means of SVM standard methods for hyperplane determination, whereas it is straightforward for ESVMs.

References

[1] T. Bartz-Beielstein, *Experimental research in evolutionary computation - the new experimentalism*, Natural Computing Series, Berlin: Springer-Verlag, 2006.

[2] R. A. Bosch, J. A. Smith, ”Separating Hyperplanes and the Authorship of the Disputed Federalist Papers,” *American Mathematical Monthly*, vol. 105, no. 7, pp. 601–608, 1998.

[3] C. J. C. Burges, ”A Tutorial on Support Vector Machines for Pattern Recognition,” *Data Mining and Knowledge Discovery* 2, pp. 121–167, 1998.

[4] K. A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. Dissertation, University of Michigan, Ann Arbor, 1975.

[5] D. Eads, D. Hill, S. Davis, S. Perkins, J. Ma, R. Porter, J. Theiler, ”Genetic Algorithms and Support Vector Machines for Time Series Classification,” *Proc. Symposium on Optical Science and Technology*, Seattle, WA, pp. 74–85, 2002.

[6] B. Feres de Souza, A. Ponce de Leon F. de Carvalho, ”Gene selection based on multi-class support vector machines and genetic algorithms,” *Journal of Genetics and Molecular Research*, vol. 4, no. 3, pp. 599–607, 2005.

[7] F. Friedrichs, C. Igel, ”Evolutionary tuning of multiple SVM parameters,” *Proc. 12th European Symposium on Artificial Neural Networks*, pp. 519–524, 2004.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New Jersey: Prentice Hall, 1999.

[9] T. Howley, M. G. Madden, ”The Genetic Evolution of Kernels for Support Vector Machine Classifiers,” *Proc. of 15th Irish Conference on Artificial Intelligence and Cognitive Science*, $http://www.it.nuigalway.ie/m\_madden/profile/pubs.html$, 2004.

[10] C.-W. Hsu, C.-J. Lin, ”A Comparison of Methods for Multi-class Support Vector Machines,” *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[11] I. Mierswa, ”Evolutionary Learning with Kernels: A Generic Solution for Large Margin Problems,” *Proc. of the Genetic and Evolutionary Computation Conference*, Seattle, WA, pp. 1553–1560, 2006.

[12] I. Mierswa, *Making Indefinite Kernel Learning Practical*, Technical Report, Artificial Intelligence Unit, Department of Computer Science, University of Dortmund, 2006.

[13] F. Perez-Cruz, A. R. Figueiras-Vidal, A. Artes-Rodriguez, ”Double chunking for solving SVMs for very large datasets,” *LEARNING’04: Linking information and knowledge*, Elche, Spain, http://eprints.pascal-network.org/archive/00001184/01/learn04.pdf, 2004.

[14] J. C. Platt, N. Cristianini, J. Shawe-Taylor, ”Large Margin DAGs for Multiclass Classification,” *Proc. of Neural Information Processing Systems*, MIT Press, pp. 547–553, 2000.

[15] V. Vapnik, *Statistical Learning Theory*, New York: Wiley, 1998.