An Evolutionary Approximation for the Coefficients of Decision Functions within a Support Vector Machine Learning Strategy

Ruxandra Stoean, Mike Preuss, Catalin Stoean, Elia El-Darzi, and D. Dumitrescu

1 Introduction

This chapter puts forward a hybrid approach which embraces the geometrical consideration of learning within support vector machines (SVMs) while it considers the estimation for the coefficients of the decision surface through the direct search capabilities of evolutionary algorithms (EAs).

The SVM framework views artificial learning from an interesting perception: A hyperplane geometrically discriminates between training samples and the coefficients of its equation have to be determined, with respect to both the particular prediction ability and the generalization capacity. On the other hand, EAs are universal optimizers that generate solutions based on abstract principles of evolution and heredity. The aim of this work thus becomes to approximate the coefficients of the decision hyperplane through a canonical EA.

The motivation for the emergence of this combined technique resulted from several findings. SVMs are a top performing tool for data mining, however, the inner-workings of the optimization component are rather constrained and very complex. On the other hand, the adaptable EAs achieve learning relatively difficult from a standalone perspective. Taking advantage of the original interpretation of learning within SVMs and the flexible optimization nature of EAs, hybridization aims to accomplish an improved methodology. The novel approach augments support vector learning to become more 'white-box' and to be able to converge independent of the properties of the underlying kernel for a potential decision function. Apart from the straightforward evolution of hyperplane coefficients, an additional aim of the chapter is to investigate the treatment of several other variables involved in the learning process. Furthermore, it is demonstrated that the transparent evolutionary alternative is performed at no additional effort as regards the parametrization of the EA. Last but not least, on a different level from the theoretical reasons, the hybridized technique offers a simple and efficient tool for solving practical problems. Several real-world test cases served not only as benchmark, but also for application of the proposed architecture, and results bear out the initial assumption that an evolutionary approach is useful in terms of deriving the coefficients of such learning structures.

The research objectives and aims of this chapter will be carried out through the following original aspects:

- The hybrid technique will consider the learning task as in SVMs but use an EA to solve the optimization problem of determining the decision function.
- Classification and regression particularities will be treated separately. The optimization problem will be tackled through two possible EAs: One will allow for a more relaxed, adaptive evolutionary learning condition, while the second will be more similar to support vector training.
- Validation will be achieved by considering five diverse real-world learning tasks.
- Besides comparing results, the potential of the utilized, simplistic EA through parametrization is to be investigated.
- To enable handling large data sets, the first adaptive EA approach will be enhanced by the use of a chunking technique, with the purpose of resulting in a more versatile approach.
- The behavior of a crowding-based EA on preserving the performance of the technique will be examined with the purpose of a future employment for the coevolution of nonstandard kernels.

- The second methodology, which is more straightforward, will be generalized through the additional evolution of internal parameters within SVMs; a very general method of practical importance is therefore desired to be achieved.

The chapter contributes some key elements to both EAs and SVMs:

- The hybrid approach combines the strong characteristics of the two important artificial intelligence fields, namely: The original learning concept of SVMs and the flexibility of the direct search and optimization power of EAs.
- The novel alternative approach simplifies the support vector training.
- The proposed hybridization offers the possibility of a general evolutionary solution to all SVM components.
- The novel technique opens the direction towards the evolution and employment of nonstandard kernels.

The remainder of this chapter is organized as follows: Section 2 outlines the primary concepts and mechanisms underlying SVMs. Section 3 illustrates the means to achieve the evolution of the coefficients for the learning hyperplane. Section 4 describes the insides of the technique and its application to real-world problems. The chapter closes with several conclusions and ideas for future enhancement.

2 The SVM Learning Scheme

SVMs are a powerful approach to data mining tasks. Their originality and performance emerge as a result of the inner learning methodology, which is based on the geometrical relative position of training samples.

2.1 A Viewpoint on Learning

Given $\{(x_i, y_i)\}_{i=1,2,\ldots,m}$, a training set where every $x_i \in R^n$ represents a data sample and each y_i corresponds to a target, a learning task is concerned with the discovery of the optimal function that minimizes the discrepancy between the given targets of data samples and the predicted ones; the outcome of previously "unknown" samples, $\{(x_i^{'}, y_i^{'})\}_{i=1,2,\ldots,p}$, is then tested.

The SVM technique is equally suited for classification and regression problems. The task for classification is to achieve an optimal separation of given samples into classes. SVMs assume the existence of a separating surface between every two classes labelled as -1 and 1. The aim then becomes the discovery of the appropriate decision hyperplane.

The standard assignment of SVMs for regression is to find the optimal function to be fitted to the data such that it achieves at most ϵ deviation

from the actual targets of samples; the aim is thus to estimate the optimal regression coefficients of such a function.

2.2 SVM Separating Hyperplanes

If training examples are known to be linearly separable, then there exists a linear hyperplane of equation (1), which separates the samples according to classes. In (1), w and b are the coefficients of the hyperplane and $\langle\rangle$ denotes the scalar product.

$$\langle w, x_i \rangle - b = 0, w \in \Re^n, b \in \Re, x_i \in R^n, i = 1, 2, ..., m \; . \tag{1}$$

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 [1], each of the positive and negative samples lies on the corresponding side of a matching supporting hyperplane for the respective class (denoted by y_i) (2).

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

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 [2], separation must be performed with a maximal margin between classes. This high generalization ability implies a minimal $\|w\|$. In summary, the SVM classification of linear separable data with a linear hyperplane leads to the optimization problem (3).

$$\begin{cases} \min_{w,b} \|w\|^2 \\ \text{subject to } y_i(\langle w, x_i \rangle - b) \geq 1, i = 1, 2, ..., m \; . \end{cases} \tag{3}$$

Generally, the training samples are not linearly separable. In the nonseparable case, 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 applied to derive a solution to the classification problem [3]. The idea is to relax the separability statement through the introduction of slack variables, denoted by ξ_i for every training example. This relaxation can be achieved 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 $\frac{\pm\xi_i}{\|w\|}$, $\xi_i \geq 0$ [4]. These values may indicate different nuanced digressions, but only a ξ_i higher than unity signals a classification error (Fig. 1). Minimization of training error is achieved by adding the indicator of error for every data sample into the separability statement while minimizing their sum. Hence, the SVM classification of linear nonseparable data with a linear hyperplane leads to the primal

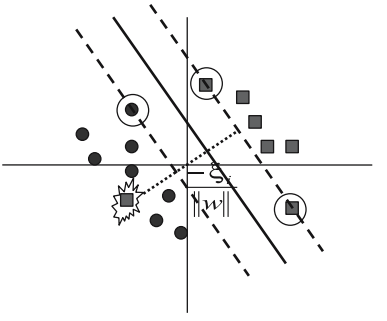


Fig. 1 The separating and supporting linear hyperplanes for the nonseparable training subsets (*squares* denote positive samples, while *circles* stand for the negative ones). The support vectors are *circled* and the misclassified data point is *highlighted*

optimization problem (4), where C represents the penalty for errors and is what is called a hyperparameter (free parameter) of the SVM method.

$$\begin{cases} \min_{w,b} \|w\|^2 + C \sum_{i=1}^{m} \xi_i, C > 0 \\ \text{subject to } y_i(\langle w, x_i \rangle - b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, 2, ..., m \, . \end{cases} \tag{4}$$

If a linear hyperplane does not provide satisfactory results for the classification task, then a nonlinear decision surface can be formulated. The initial space of training data samples can be nonlinearly mapped into a higher dimensional one, called the feature space and further denoted by H, where a linear decision hyperplane can be subsequently built. The separating hyperplane can achieve an accurate classification in the feature space which corresponds to a nonlinear decision function in the initial space (Fig. 2). The procedure therefore leads to the creation of a linear separating hyperplane that would, as before, minimize training error; however, in this case, it will 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.

As in the classical SVM solving procedure, vectors appear only as part of scalar products, the issue can be further simplified by substituting the scalar product by what is referred to as kernel.

A kernel is defined as a function with the property given by (5).

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle, x_i, x_j \in R^n \, . \tag{5}$$

The kernel can be perceived as to express the similarity between samples. SVMs require the kernel to be a positive (semi-)definite function in order for the standard approach to find a solution to the optimization problem [5]. Such a kernel satisfies Mercer's theorem below and is, therefore, a scalar product in some space [6].

Theorem 1. *(Mercer) [3], [7], [8], [9] Let K(x,y) be a continuous symmetric kernel that is defined in the closed interval $a \leq x \leq b$ and likewise for y. The kernel K(x,y) can be expanded in the series*

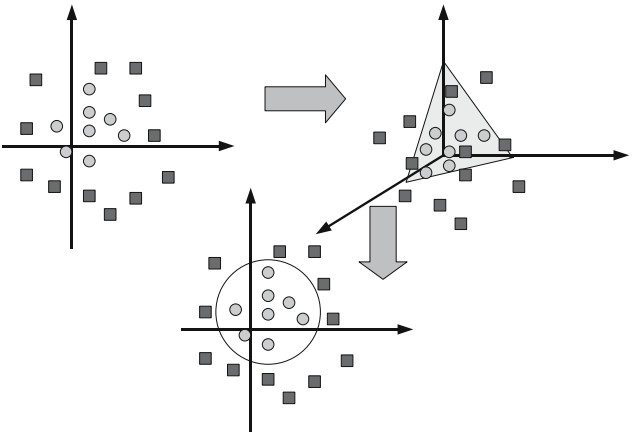


Fig. 2 Initial data space (*left*), nonlinear map into the higher dimension where the objects are linearly separable/the linear separation (*right*), and corresponding nonlinear surface (*bottom*)

$$K(x,y) = \sum_{i=1}^{\infty} \lambda_i \Phi(x)_i \Phi(y)_i$$

with positive coefficients, $\lambda_i > 0$ *for all i. For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary that the condition*

$$\int_a^b \int_a^b K(x,y)\psi(x)\psi(y)dxdy \geq 0$$

holds for all $\psi(\cdot)$ *for which*

$$\int_a^b \psi^2(x)dx < \infty$$

The problem with this restriction is twofold [5]. Firstly, Mercer's condition is very difficult to check for a newly constructed kernel. Secondly, 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:

- the polynomial classifier of degree p: $K(x,y) = \langle x,y \rangle^p$
- the radial basis function classifier: $K(x,y) = e^{\frac{\|x-y\|^2}{\sigma}}$, where p and σ are also hyperparameters of SVMs.

However, as a substitute for the original problem solving, a direct search algorithm does not depend on the condition whether the kernel is positive (semi-)definite or not.

After the optimization problem is solved, the class of every test sample is calculated: The side of the decision boundary on which every new data example lies is determined (6).

$$class(x_i^{'}) = sgn(\langle w, \Phi(x_i^{'})\rangle - b), i = 1, 2, ..., p\ . \tag{6}$$

As it is not always possible to determine the map Φ and, as a consequence of the standard training methodology, either to explicitly obtain the coefficients, the class follows from further artifices.

The classification accuracy is then defined as the number of correctly labelled cases over the total number of test samples.

2.3 Addressing Multi-class Problems through SVMs

k-class SVMs build several two-class classifiers that separately solve the matching tasks. The translation from multi-class to two-class is performed through different systems, among which one-against-all, one-against-one or decision directed acyclic graph are the most commonly employed.

One-against-all Approach

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

Consequently, by placing the problem in the initial space, the aim of every j^{th} SVM is to determine the optimal coefficients w^j and b^j of the decision hyperplane which best separates the samples with outcome j from all the other samples in the training set, such that (7) holds.

$$\begin{cases} \min_{w^j, b^j} \frac{\|w^j\|^2}{2} + C \sum_{j=1}^{m} \xi_i^j, \\ \text{subject to } y_i(\langle w^j, x_i\rangle - b^j) \geq 1 - \xi_i^j, \\ \xi_i^j \geq 0, i = 1, 2, ..., m, j = 1, 2, ..., k\ . \end{cases} \tag{7}$$

Once all hyperplanes are determined following the classical SVM training, the label for a test sample $x_i^{'}$ is given by the class that has the maximum value for the learning function, as in (8).

$$class(x_i^{'}) = argmax_{j=1,2,...,k}(\langle w^j, \Phi(x_i^{'})\rangle) - b^j), i = 1, 2, ..., p\ . \tag{8}$$

One-against-one Approach

The one-against-one (1a1) technique [10] builds $\frac{k(k-1)}{2}$ SVMs. Every machine is trained on data from every two classes, l and j, where samples labelled with l 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^{lj} and b^{lj} of the decision hyperplane which best separates the samples with outcome l from the samples with outcome j, such that (9).

$$\begin{cases} \min_{w^{lj},b^{lj}} \frac{\|w^{lj}\|^2}{2} + C\sum_{i=1}^{m} \xi_i^{lj}, \\ \text{subject to } y_i(\langle w^{lj}, x_i\rangle - b) \geq 1 - \xi_i^{lj}, \\ \xi_i^{lj} \geq 0, i = 1, 2, ..., m, l, j = 1, 2, ..., k, l \neq j \ . \end{cases} \tag{9}$$

Once the hyperplanes of the $\frac{k(k-1)}{2}$ SVMs are found, a *voting* method is used to determine the class for a test sample $x_i^{'}, i = 1, 2, ..., p$. For every SVM, the label is computed following the sign of the corresponding decision function applied to $x_i^{'}$. Subsequently, if the sign says $x_i^{'}$ is in class l, the vote for the l-th class is incremented by one; conversely, the vote for class j is increased by unity. Finally, $x_i^{'}$ is taken to belong to the class with the largest vote. In case two classes have an identical number of votes, the one with the smaller index is selected.

Decision Directed Acyclic Graph

Learning within the decision directed acyclic graph (DDAG) technique [11] follows the same procedure as in 1a1. After the hyperplanes of the $\frac{k(k-1)}{2}$ SVMs are discovered, a graph system is used to determine the class for a test sample $x_i^{'}, i = 1, 2, ..., p$. 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 a test instance $x_i^{'}$ is evaluated, it is descended from node to node, in other words, one class is eliminated from each corresponding list, until the leaves are reached. The mechanism starts at the root node which considers the first and last classes. At each node, l vs j, we refer to the SVM that was trained on data from classes l and j. The class of x is computed by following the sign of the corresponding decision function applied to $x_i^{'}$. Subsequently, if the sign says x is in class l, the node is exited via the right edge while, conversely, through the left edge. The wrong class is thus eliminated from the list and it is proceeded 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 $x_i^{'}$ eventually reaches.

2.4 SVM Regression Hyperplanes

SVMs for regression must find a function $f(x)$ that has at most ϵ deviation from the actual targets of training samples and, simultaneously, is as flat as possible [12]. In other words, the aim is to estimate the regression coefficients of $f(x)$ with these requirements.

While the former condition is straightforward, errors are allowed as long as they are less than ϵ, the latter needs some further explanation [13]. Resulting

values of the regression coefficients may affect the model in the sense that it fits current training data but has low generalization ability, which would contradict the principle of Structural Risk Minimization for SVMs [2]. In order to overcome this limitation, it is required to choose the flattest function in the definition space. Another way to interpret SVMs for regression is that training data are constrained to lie on a hyperplane that allows for some error and, at the same time, has high generalization capacity.

Suppose a linear regression model can fit the training data. Consequently, function f has the form (10).

$$f(x) = \langle w, x \rangle - b \; . \tag{10}$$

The conditions for the flattest function (smallest slope) that approximates training data with ϵ precision can be translated into the optimization problem (11).

$$\begin{cases} \min_{w,b} \|w\|^2 \\ \text{subject to } \begin{cases} y_i - \langle w, x_i \rangle + b \leq \epsilon \\ \langle w, x_i \rangle - b - y_i \leq \epsilon \end{cases} \\ i = 1, 2, ..., m \; . \end{cases} , \tag{11}$$

It may very well happen that the linear function f is not able to fit all training data and consequently SVMs will again allow for some relaxation, analogously to the corresponding situation for classification.

Therefore, the positive slack variables ξ_i and ξ_i^*, both attached to each sample, are introduced into the condition for approximation of training data and, also as before, the sum of these indicators for errors is minimized. The primal optimization problem in case of regression then translates to (12).

$$\begin{cases} \min_{w,b} \|w\|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \\ \text{subject to } \begin{cases} y_i - \langle w, x_i \rangle + b \leq \epsilon + \xi_i \\ \langle w, x_i \rangle - b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \\ i = 1, 2, ..., m \; . \end{cases} , \tag{12}$$

If a linear function is not at all able to fit training data, a nonlinear function has to be chosen, as well. The procedure follows the same steps as before in SVMs for classification.

When a solution for the optimization problem is reached, the predicted target for a test sample is computed as (13).

$$f(x_i^{'}) = \langle w, \Phi(x_i^{'}) \rangle - b, i = 1, 2, ..., p \; . \tag{13}$$

Also as in the classification problem, the regression coefficients are rarely transparent and the predicted target actually is derived from other computations.

In order to verify the accuracy of the technique, the value of the root mean square error (RMSE) is computed as in (14).

$$RMSE = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(f(x_i^{'}) - y_i^{'})^2} \ . \tag{14}$$

2.5 Solving the Optimization Problem within SVMs

The standard algorithm of finding the optimal solution relies on an extension of the Lagrange multipliers technique. Corresponding dual problem may be expressed as (15) for classification.

$$\begin{cases} \min_{\{\alpha_i\}_{i=1,2,..,m}} Q(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j K(x_i, x_j) \\ \text{subject to} \begin{cases} \sum_{i=1}^{m} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases} \\ i = 1, 2, ..., m \ . \end{cases} \tag{15}$$

Conversely, in the most general case of nonlinear regression, the dual problem is restated as (16). For reasons of a shorter reference, $\alpha_i(^*)$ denotes both α_i and α_i^*, in turn.

$$\begin{cases} \min_{\{\alpha_i(^*)\}_{i=1,2,..,m}} Q(\alpha(^*)) = \sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) \\ -\epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \begin{cases} \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0 \\ 0 \leq \alpha_i(^*) \leq C \end{cases} \\ i = 1, 2, ..., m \ . \end{cases} \tag{16}$$

The optimum Lagrange multipliers $\alpha_i(^*)$s are determined as the solutions of the system by setting the gradient of the objective function to zero. For more mathematical explanation, see [3], [14].

3 Evolutionary Adaptation of the Hyperplane Coefficients to the Training Data

Apart from its emergence as a simple complementary method for solving the SVM derived optimization problem, the EA-powered determination of hyperplane coefficients had to be explored and improved with respect to runtime, prediction accuracy and adaptability.

3.1 Motivation and Aim

Despite the originality and performance of the learning vision of SVMs, the inner training engine is intricate, constrained, rarely transparent and able to converge only for certain particular decision functions. This has brought the motivation to investigate and put forward an alternative training approach that benefits from the flexibility and robustness of EAs.

The technique adopts the learning strategy of the SVMs but aims to simplify and generalize its solving methodology, by offering a transparent substitute to the initial 'black-box'. Contrary to the canonical technique, the evolutionary approach can at all times explicitly acquire the coefficients of the decision function, without any further constraints. Moreover, in order to converge, the evolutionary method does not require the positive (semi-)definition properties for kernels within nonlinear learning. Eventually, the evolutionary approach demonstrates to be an efficient tool for real-world application in vital domains like disease diagnosis and prevention or spam filtering.

There have been numerous attempts to combine SVMs and EAs, however, this method differs from the reported ones: The learning path remains the same, except that the coefficients of the decision function are now evolved with respect to the optimization objectives regarding accuracy and generalization. Several potential structures, enhancements and additions had been proposed, tested and confirmed using available benchmarking test problems.

3.2 Literature Review: Previous EA-SVM Interactions

The chapter focuses on the evolution of the coefficients for decision functions of a learning strategy similar to that of SVMs. However, there are other works known to combine SVMs and EAs. One evolutionary direction tackles model selection, which concerns the adjustment of hyperparameters within SVMs, i.e. the penalty for errors C and parameters of the kernel, and is generally performed through grid search or gradient descent methods. Alternatively, determination of hyperparameters can be achieved through evolution strategies [15]. Another aspect envisages the evolution of the form for the kernel, which can be performed by means of genetic programming [16]. The Lagrange multipliers involved in the expression of the dual problem can be evolved by means of evolution strategies and particle swarm optimization [5]. Inspired by the geometrical SVM learning, [17] also reports the evolution of w and C, while using erroneous learning ratio and lift values as the objective function. Current paper therefore extends the work in the hybridization between EAs and SVMs by filling the gap of a direct evolutionary solving of the primal optimization problem of determining the decision hyperplane, which has never been performed before, to the best of our knowledge.

3.3 Evolving the Coefficients of the Hyperplane

The evolutionary approach for support vector training (ESVM) considers the adaptation of a flexible hyperplane to the given training data through the evolution of the optimal coefficients for its equation. After the necessary revisions in the learning objectives due to a different way of solving the optimization task, the corresponding EA is adopted in a canonical formulation for real-valued problems [18]. For the purpose of a comparative analysis between ESVMs and SVMs, there are solely the classical polynomial and radial kernels that are used in this chapter to shape the decision functions.

Representation

An individual c encodes the coefficients of the hyperplane, w and b (17). Individuals are randomly generated such that $w_{i'} \in [-1, 1], i' = 1, 2, ..., n$, $b \in [-1, 1]$.

$$c = (w_1, ..., w_n, b) \ . \tag{17}$$

Fitness Assignment

Prior to deciding on a strategy to evaluate individuals, the objective function must be established in terms of the new approach to address the optimization goals.

Since ESVMs depart from the standard mathematical treatment of SVMs, a different general (nonlinear) optimization problem is derived [19] . Accordingly, w is also mapped through Φ into H. As a result, the squared norm that is involved in the generalization condition is now $\|\Phi(w)\|^2$. At the same time, the equation of the hyperplane consequently changes to (18).

$$\langle \Phi(w), \Phi(x_i) \rangle - b = 0. \tag{18}$$

The scalar product is used in the form (19) and, besides, the kernel is additionally employed to address the norm in its simplistic equivalence to a scalar product.

$$\langle u, w \rangle = u^T w \ . \tag{19}$$

In conclusion, the optimization problem for classification is reformulated as (20).

$$\begin{cases} \min_{w,b} K(w, w) + C \sum_{i=1}^{m} \xi_i, C > 0 \\ \text{subject to } y_i(K(w, x_i) - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1, 2, ..., m \ . \end{cases} \tag{20}$$

At the same time, the objectives for regression are transposed to (21).

$$\begin{cases} \min_{w,b} K(w,w) + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ \text{subject to } \begin{cases} y_i - K(w,x_i) + b \le \epsilon + \xi_i \\ K(w,x_i) - b - y_i \le \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \\ i = 1,2,...,m\ . \end{cases} \quad , \tag{21}$$

The ESVMs targeting multi-class situations must undergo similar transformation with respect to the expression of the optimization problem [20], [21]. As 1aa is concerned, the aim of every j^{th} ESVM is expressed as to determine the optimal coefficients, w^j and b^j, of the decision hyperplane which best separates the samples with outcome j from all the other samples in the training set, such that (22) takes place.

$$\begin{cases} \min_{w^j,b^j} K(w^j,w^j) + C\sum_{i=1}^{m}\xi_i^j, \\ \text{subject to } y_i(K(w^j,x_i) - b^j) \ge 1 - \xi_i^j, \\ \xi_i^j \ge 0, i = 1,2,...,m, j = 1,2,...,k\ . \end{cases} \tag{22}$$

Within the 1a1 and DDAG approaches, the aim of every ESVM becomes to find the optimal coefficients w^{lj} and b^{lj} of the decision hyperplane which best separates the samples with outcome l from the samples with outcome j, such that (23) holds.

$$\begin{cases} \min_{w^{lj},b^{lj}} K(w^{lj},w^{lj}) + C\sum_{i=1}^{m}\xi_i^{lj}, \\ \text{subject to } y_i(K(w^{lj},x_i) - b^{lj}) \ge 1 - \xi_i^{lj}, \\ \xi_i^{lj} \ge 0, i = 1,2,...,m, l,j = 1,2,...,k, i \ne j\ . \end{cases} \tag{23}$$

The fitness assignment now derives from the objective function of the optimization problem and is minimized. Constraints are handled by penalizing the infeasible individuals through the introduction of a function $t : R \to R$ which returns the value of the argument, if negative, while zero otherwise.

Classification and regression variants simply differ in terms of objectives and constraints. Thus the expression of the fitness function for the former is (24), with the corresponding indices in the multi-class situations [22], [23], [24].

$$f(w,b,\xi) = K(w,w) + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}[t(y_i(K(w,x_i) - b) - 1 + \xi_i)]^2. \tag{24}$$

As for the latter, the fitness assignment is defined in the form (25) as found in [25], [26], [27].

$$f(w,b,\xi) = K(w,w) + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) + \sum_{i=1}^{m}[t(\epsilon + \xi_i - y_i + K(w,x_i) - b)]^2 + \sum_{i=1}^{m}[t(\epsilon + \xi_i^* + y_i - K(w,x_i) + b)]^2. \quad (25)$$

Selection and Variation Operators

The efficient tournament selection and the common genetic operators for real encoding, i.e. intermediate crossover and mutation with normal perturbation, are applied.

Stop Condition

The EA stops after a predefined number of generations and outputs the optimal coefficients for the equation of the hyperplane. Moreover, ESVM is transparent at all times during the evolutionary cycle, thus w and b may be observed as they adapt throughout the process.

Test Step

Once the coefficients of the hyperplane are found, the class for an unknown data sample can be determined directly following (26) .

$$class(x_i^{'}) = sgn(K(w,x_i^{'}) - b), i = 1,2,...,p\,. \quad (26)$$

Conversely for regression, the target of test samples can be obtained through (27) .

$$f(x_i^{'}) = K(w,x_i^{'}) - b, i = 1,2,...,p\,. \quad (27)$$

For the multi-class tasks, the label is found by employing the same specific mechanisms, only this time the resulting decision function applied to the current test sample takes the form (28) .

$$f(x_i^{'}) = K(w^j,x_i^{'}) - b^j, i = 1,2,...,p, j = 1,2,...,k\,. \quad (28)$$

3.4 Preexperimental Planning: The Test Cases

Experimentation had been conducted on five real-world problems, coming from the UCI Repository of Machine Learning Databases[1], i.e. diabetes mellitus diagnosis [28] , spam detection [29] , [30], iris recognition [20] , soybean

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

Table 1 Data set properties

	Diabetes	Iris	Soybean	Spam	Boston
Data					
No. of samples	768	150	47	4601	506
No. of features	8	4	35	57	13
No. of classes	2	3	4	2	-

disease diagnosis [31] and Boston housing [25], [26] (see Table 1). 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. Finally, Boston housing is a representative regression task. For all reasons mentioned above, the selection of test problems certainly contained all the variety of situations that had been necessary for the objective validation of the ESVM approach. The experimental design was set to employ holdout cross-validation: For each data set, 30 runs of the ESVM were conducted – in every run, approximately 70% random cases were appointed to the training set and the remaining 30% went into test. The necessity for data normalization in diabetes, spam and iris was also observed.

4 Discovering ESVMs

While the target of the EA was straightforward, addressing several inside interactions had been not. Moreover, application of the ESVM to the practical test cases had yielded more research questions yet to be resolved. Finally, further implications of being able to instantly operate on the evolved coefficients had been realized.

4.1 A Naïve Design

As already stated, the coefficients of the hyperplane, w and b, are encoded into the structure of an individual. But, since the conditions for hyperplane optimality additionally refer the indicators for errors, ξ_i, $i = 1, 2, ..., m$, the problem becomes how to comprise them in the evolutionary solving. One simple solution could be to depart from the SVM geometrical strict meaning of a deviation and simply evolve the factors of indicators for errors. Thus, the structure of an individual changes to (29), where $\xi_j \in [0, 1], j = 1, 2, ..., m$.

Table 2 Manually tuned SVM hyperparameter values for the evolutionary and canonical approach

	Diabetes	Iris	1a1/1aa	Soybean	Spam	Boston
ESVMs						
p (σ)	$p = 2$	$\sigma = 1$	$p = 1$	$p = 1$	$p = 1$	
SVMs						
p (σ)	$p = 1$	$\sigma = 1/m$	$p = 1$	$p = 1$	$\sigma = 1/m$	

Table 3 Manually tuned EA parameter values for the naïve construction

	Diabetes	Iris	1a1	Soybean	Spam	Boston
ps	100	100	100	100	200	
ng	250	100	100	250	2000	
cp	0.40	0.30	0.30	0.30	0.50	
mp	0.40	0.50	0.50	0.50	0.50	
emp	0.50	0.50	0.50	0.50	0.50	
ms	0.10	0.10	0.10	0.10	0.10	
ems	0.10	0.10	0.10	0.10	0.10	

$$c = (w_1, ..., w_n, b, \xi_1, ..., \xi_m) \ . \tag{29}$$

The evolved values of the indicators for errors can now be addressed in the proposed expression for fitness evaluation. Also, mutation of errors is now constrained, preventing the ξ_is from taking negative values.

Once the primary theoretical aspects had been completed, the experimental design had to be inspected and the practical evaluation of the viability of ESVMs had to be conducted. The hyperparameters both approaches share were manually chosen (Table 2). The error penalty C was invariably set to 1. For certain (e.g. radial, polynomial) kernels, the optimization problem is relatively simple, due to Mercer's theorem, and is also implicitly solved by classical SVMs [32]. Note that ESVMs are not restricted to using the traditional kernels, but these had been employed within to enable comparison with classical SVMs. Therefore, as a result of multiple testing, a radial kernel was used for the iris data set, a polynomial one was employed for diabetes, while for spam, soybean and Boston, a linear surface was applied. In the regression case, ϵ was set to 0.

An issue of crucial importance for demonstrating the feasibility of any EA alternative relies on the simplicity to determine appropriate parameters. The EA parameter values were initially manually determined (Table 3).

In order to validate the manually found EA parameter values and to probe the ease in their choice, the tuning method of Sequential Parameter Optimization (SPO) [33] was applied. The SPO builds on a quadratic regression model, supported by a Latin hypercube sampling (LHS) methodology and noise reduction, by incrementally increased repetition of runs. Parameter bounds were set as follows:

Table 4 SPO tuned EA parameter values for the naïve representation

	Diabetes	Iris	Soybean	Spam	Spam +Chunks	Boston
ps	198	46	162	154	90	89
ng	296	220	293	287	286	1755
pc	0.87	0.77	0.04	0.84	0.11	0.36
pm	0.21	0.57	0.39	0.20	0.08	0.5
epm	0.20	0.02	0.09	0.07	0.80	0.47
ms	4.11	4.04	0.16	3.32	0.98	0.51
ems	0.02	3.11	3.80	0.01	0.01	0.12

- Population size (*ps*) - 5/2000
- Number of generations (*ng*) - 50/300
- Crossover probability (*pc*) - 0.01/1
- Mutation probability (*pm*) - 0.01/1
- Error mutation probability (*epm*) - 0.01/1
- Mutation strength (*ms*) - 0.001/5
- Error mutation strength (*ems*) - 0.001/5

Since the three multi-class techniques behave similarly in the manually tuned multi-class experiments (Table 5), automatic adjustment was run only for the most widely used case of 1a1. The best parameter configurations for all problems as determined by SPO are depicted in Table 4.

Test accuracies/errors obtained by manual tuning are presented in Table 5. Differentiated (spam/non spam for spam filtering and ill/healthy for diabetes) results are also depicted.

Table 5 Accuracy/RMSE of the manually tuned naïve ESVM version on the considered test sets, in percent

	Average	Worst	Best	StD
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
Boston	**4.78**	5.95	3.96	0.59

Table 6 Accuracies of the SPO tuned naïve ESVM version on the considered test sets, in percent

	$\mathbf{LHS}_{best}$	**StD**	**SPO**	**StD**
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	**91.04**	0.80
Spam (spam)	80.63	3.51	84.72	1.59
Spam (non spam)	94.82	0.94	95.10	0.81
Boston	5.41	0.65	**5.04**	0.52

Table 6 summarizes the performances and standard deviations of the best configuration of an initial LHS sample and of the SPO.

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 LHS 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 more effort, i.e. increasing population size or number of generations.

The computational experiments show that the proposed technique produces equally good results as compared to the canonical SVMs and it is further explained in subsection 4.6. Furthermore, the smaller standard deviations prove the higher stability of ESVMs.

As concerns the difficulty in setting the EA, SPO confirms: 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, spam and Boston. Especially for the small problems, well performing parameter configurations are seemingly easy to find. This brings evidence in support of the (necessary) simplicity in tuning the parameters inside the evolutionary alternative solving.

It must be stated that for the standard kernels, one cannot expect the ESVM to be better than the standard SVM, since the kernel transformation that induces learning is the same. However, the flexibility of the EAs as optimization tools makes ESVMs an attractive choice from the performance perspective, due to their prospective ability to additionally evolve problem-tailored kernels, regardless of whether they are positive (semi-)definite or not, which is impossible under SVMs.

4.2 Chunking within ESVMs

A first 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, this problem was tackled by an adaptation of a chunking procedure [34] inside ESVM.

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 and 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 (Algorithm 1).

Algorithm 1. ESVM with Chunking

Require: The training samples
Ensure: Best obtained coefficients and corresponding accuracy
begin
Randomly choose N training samples, 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 exist) N/2 samples, equally distributed;
Randomly choose the rest up to N from the current chunk and add all to a new one;
if obtained training accuracy if higher than the best one obtained so far **then**
Update best accuracy and hyperplane coefficients; set improvement to true;
end if
end while
Apply best obtained coefficients on the test set and compute accuracy
return accuracy
end

ESVM with chunking was applied to the spam data set. Manually tuned parameters had the same values as before, except the number of generations

Table 7 Accuracy/RMSE of the manually tuned ESVM with chunking version on the considered test sets, in percent

	Average	Worst	Best	StD
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

Table 8 Accuracies of the SPO tuned ESVM with chunking version on the considered test sets, in percent

	LHS_{best}	StD	SPO	StD
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

for each run of the EA which is now set to 100. The chunk size, N, was chosen as 200 and the number of iterations with no improvement, i.e. repeats of the chunking cycle, was designated to be 5. Values derived from the SPO tuning are presented in the chunking column from Table 4.

Results of manual and SPO tuning are shown in Tables 7 and 8. The novel approach of ESVM with chunking produced good results in a much smaller runtime; it runs 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.

4.3 A Pruned Variant

Although already a good alternative approach, the ESVM may still be improved concerning simplicity. The current optimization problem requires to treat the error values, which in the present EA variant are included in the representation. These severely complicate the problem by increasing the genome length (variable count) by the number of training samples. Moreover, such a methodology strongly departs from the canonical SVM concept. Therefore, it had been investigated whether the indicators for errors could be computed instead of evolved.

Since ESVMs directly and interactively provide hyperplane coefficients at all times, the generic EA representation (17) can be kept and the indicators can result from geometrical calculations. In case of classification, the procedure follows [1]. The current individual, which is the current separating

hyperplane, is considered and supporting hyperplanes are determined through the mechanism below. One first computes (30).

$$\begin{cases} m_1 = min\{K(w, x_i)|y_i = +1\} \\ m_2 = max\{K(w, x_i)|y_i = -1\} \ . \end{cases} \tag{30}$$

Then (31) proceeds.

$$\begin{cases} p = |m_1 - m_2| \\ w' = \frac{2}{p} w \\ b' = \frac{1}{p}(m_1 + m_2) \ . \end{cases} \tag{31}$$

For every training sample x_i, the deviation to its corresponding supporting hyperplane (32) is obtained.

$$\delta(x_i) = \begin{cases} K(w', x_i) - b' - 1, \text{ if } y_i = +1 \\ K(w', x_i) - b' + 1, \text{ if } y_i = -1 \\ i = 1, 2, ..., m \ . \end{cases} \tag{32}$$

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 no normalization. The different behavior can be explained by the fact that the first three data sets have a larger number of training samples. The sum of 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 form of the fitness function remains as before (24), obviously without taking the ξ_is as arguments.

The proposed method for acquiring the errors for the regression situation is as follows. For every training sample, one firstly calculates the difference between the actual target and the predicted value that is obtained with the coefficients of the current individual (regression hyperplane), as in (33).

$$\delta_i = |K(w, x_i) - b - y_i|, i = 1, 2, ..., m \ . \tag{33}$$

Secondly, one tests the difference against the ϵ threshold, following (34).

$$\begin{cases} \text{if } \delta_i < \epsilon & \text{then } \xi_i = 0 \\ \text{else} & \xi_i = \delta_i - \epsilon \\ i = 1, 2, ..., m \ . \end{cases} \tag{34}$$

The newly obtained indicators for errors can now be employed in the fitness evaluation of the corresponding individual, which changes from (25) to (35):

Table 9 Manually tuned parameter values for the pruned approach

	Diabetes	Iris	Soybean	Spam	Boston
ps	100	100	100	150	200
ng	250	100	100	300	2000
pc	0.4	0.30	0.30	0.80	0.50
pm	0.4	0.50	0.50	0.50	0.50
ms	0.1	4	0.1	3.5	0.1

Table 10 SPO tuned parameter values for the pruned representation

	Diabetes	Iris	Soybean	Spam	Boston
ps	190	17	86	11	100
ng	238	190	118	254	1454
pc	0.13	0.99	0.26	0.06	0.88
pm	0.58	0.89	0.97	0.03	0.39
ms	0.15	3.97	0.08	2.58	1.36

$$f(w,b) = K(w,w) + C\sum_{i=1}^{m}\xi_i \ . \tag{35}$$

The function to be fitted to the data is thus still required to be as flat as possible and to minimize the errors of regression that are higher than the permitted ϵ. Experiments on the Boston housing problem demonstrated that the specific method for computing the deviations does not require any additional normalization.

The problem related settings and SVM hyperparameters were kept the same as for naïve approach, except ϵ which was set to 5 for the regression problem, which reveals that the pruned representation apparently needs a more generous deviation allowance within training.

The EA first proceeded with the manual values for parameters from Table 9. Subsequent parameter values derived from SPO on the pruned variant are shown in Table 10.

Results obtained after manual and SPO tuning are depicted in Tables 11 and 12. The automated performance values were generated by 30 validation runs for the best found configurations after the initial design and SPO, respectively.

Results of automated tuning are similar to those of the manual regulation which once again demonstrates the easy adjustability of the ESVM. Additionally, the performance spectra of LHS was plotted in order to compare the hardness of finding good parameters for our two representations on the spam and soybean problems (Figs. 3 and 4). The Y axis represents the fractions of all tried configurations; therefore the Y value corresponding to each bar denotes the percentage of configurations that reached the accuracy of the X axis where the bar is positioned.

Table 11 Accuracy/RMSE of the manually tuned pruned ESVM version on the considered test sets, in percent

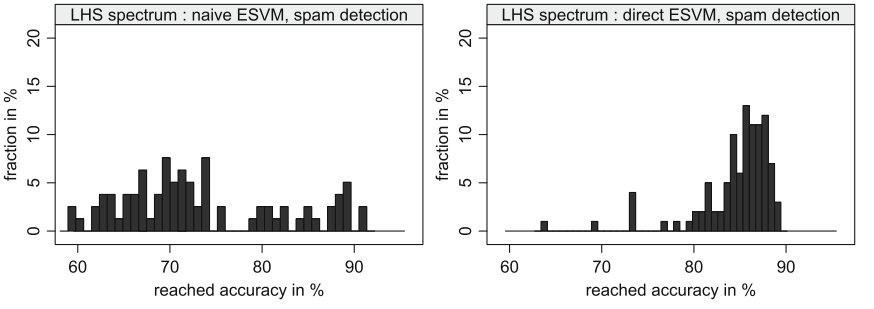
	Average	**Worst**	**Best**	**StD**
Diabetes (overall)	**74.60**	70.31	82.81	2.98
Diabetes(ill)	45.38	26.87	58.57	6.75
Diabetes (healthy)	89.99	86.89	96.75	2.66
Iris 1aa (overall)	**93.33**	86.67	100	3.83
Iris 1a1 (overall)	**95.11**	73.33	100	4.83
Iris DDAG (overall)	**95.11**	88.89	100	3.22
Soybean 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)	**85.68**	82	88.26	1.72
Spam (spam)	70.54	62.50	77.80	4.55
Spam (non spam)	95.39	92.66	97.44	1.09
Boston	**5.07**	6.28	3.95	0.59

Table 12 Accuracies of the SPO tuned pruned ESVM version on the considered test sets, in percent

	LHS$_{best}$	**StD**	**SPO**	**StD**
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
Boston	4.99	0.66	**4.83**	0.45

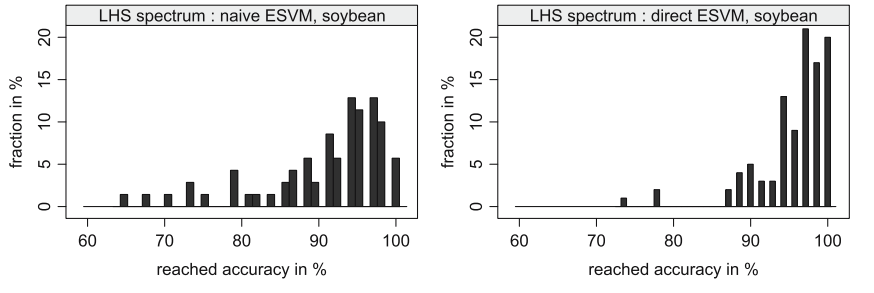
The diagrams illustrate the fact that naïve ESVM is harder to parameterize than the pruned approach: When SPO finds a configuration for the latter, it is already a promising one, as it can be concluded from the higher corresponding bars.

It is interesting to remark that the pruned 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 scalar products that must be calculated due to (30), (32) and (33). As run length itself is a parameter in present studies, an upper bound of the necessary effort is rather obtained. Closer investigation may lead to a better understanding of suitable run lengths, e.g. in terms of fitness evaluations. However, the pruned representation has its advantages. Besides featuring smaller genomes, less parameters are needed, because the slack variables are not evolved and thus



(a) (b)

Fig. 3 Comparison of EA parameter spectra, LHS with size 100, 4 repeats, (a) for the naïve (7 parameters) and (b) the pruned (5 parameters) representation on the spam problem



(a) (b)

Fig. 4 Comparison of EA parameter spectra, LHS with size 100, 4 repeats, (a) for the naïve (7 parameters) and (b) the pruned (5 parameters) representation on the soybean problem

two parameters are eliminated. As a consequence, it can be observed that this representation is easier to tune.

The best configurations for the pruned representation perform slightly worse as compared to the results recorded for the naïve representation. The independent evolution of the slack variables seems to result in a better adjustment of the hyperplane as opposed to their strict computation. 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, spam and Boston. This indicates once more that parameter setting for the ESVM 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.

Table 13 SPO tuned parameter values for the pruned representation with crowding

	Diabetes	Iris	Spam
ps	92	189	17
ng	258	52	252
pc	0.64	0.09	0.42
pm	0.71	0.71	0.02
ms	0.20	0.20	4.05

4.4 A Crowding Variant

In addition to the direct pruned representation, a crowding [35] variant of the EA had also been tested. Within crowding, 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 nonstandard kernels. It is desirable, however, to investigate whether the employment of a crowding-based EA on the pruned representation would maintain the performance of the technique or not. All the other elements of the EA remained the same and the values for parameters as determined by SPO are shown in Table 13. The crowding experiment was chosen to be run only on the representative tasks for many samples (diabetes and spam), features (spam) and classes (iris).

Note that only automated tuning was performed for the pruned crowding ESVM and results can be found in Table 14.

Table 14 Accuracies of the SPO tuned pruned version with crowding on the considered test sets, in percent

	LHD_{best}	StD	SPO	StD
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
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

SPO revealed that for the crowding variant, some parameter interactions dominate the best performing configurations: For larger population sizes, smaller mutation steps and larger crossover probabilities are better suited, and with greater run lengths, performance increases with larger mutation step sizes. For the original pruned variant, no such clear interactions could be attained. However, in both cases, many good configurations were detected.

4.5 Integration of SVM Hyperparameters

For practical considerations, a procedure for a dynamic choice of model hyperparameters was further included within the pruned construction. Having judged from performed experiments, the parameter expressing the penalty for errors C seemed of no significance within the ESVM technique; it was consequently dropped from the parameters pool. Further on, by simply inserting one more variable to the genome, the kernel parameter (p or $sigma$) could also be evolved. In this way, benefiting from the evolutionary solving of the primal problem, model selection was actually performed at the very same time.

The idea was tested through an immediate manual tuning of the EA parameters; the values are depicted in Table 15. For reasons of generality with respect to the new genomic variable, an extra mutation probability (hpm) and mutation strength (hms) respectively, were additionally set. The corresponding gene also had a continuous encoding, the hyperparameter being rounded at kernel application. The soybean task was not considered for this experiment anymore, as very good results had already been achieved.

Table 15 Manually tuned parameter values for the all-in-one pruned representation

	Diabetes	**Iris**	**Boston**	**Spam**
ps	100	50	100	5
ng	250	280	2000	480
pc	0.4	0.9	0.5	0.1
pm	0.4	0.9	0.5	0.1
hpm	0.4	0.9	0.9	0.1
ms	0.1	1	0.1	3.5
hms	0.5	0.1	0.1	0.1

The resulting values for the SVM hyperparameters were identical to our previous manual choice (Table 2), with one exception in the diabetes task, where sometimes a linear kernel is obtained.

Results of the all-inclusive technique (Table 16), similar in accuracy or regression error to the prior ones and obtained at no additional cost, sustain the inclusion of model selection and point to the next extension, the coevolution of nonstandard kernels.

4.6 ESVMs Versus SVMs

In order to validate the aim of this work, that is to offer a simpler, yet equally performing alternative to SVM training, this section compares the ESVM results with those of canonical SVMs run in R on the same data sets. The reasons for this choice of a contrast were twofold: The R software already

Table 16 Accuracy/RMSE of the manually tuned all-in-one pruned ESVM version on the considered test sets, in percent

	Average	**Worst**	**Best**	**StD**
Diabetes (overall)	**74.20**	66.66	80.21	3.28
Diabetes(ill)	46.47	36.99	63.08	6.92
Diabetes (healthy)	89.23	81.40	94.62	3.46
Iris (overall)	**96.45**	93.33	100	1.71
Spam (overall)	**88.92**	85.39	91.48	1.5
Spam (spam)	79.98	68.72	94.67	5.47
Spam (non spam)	94.79	84.73	96.91	2.22
Boston	**5.06**	6.19	3.97	0.5

contains a standard package for a SVM implementation and objectivity is achieved only in similar experimental setup and test cases. However, search for the outcome of the application of other related approaches (as described in subsection 3.2) on the same data sets revealed only results on the diabetes task: A classification accuracy of 74.48% and a standard deviation of 4.30% came out of a 20-fold cross-validation within evolution of Lagrange multipliers in [32] and an accuracy of 76.88% and a standard deviation of 0.25% averaged over 20 trials was obtained through the evolution of the SVM hyperparameters in [15].

The results, obtained after 30 runs of holdout cross-validation, are illustrated in Table 17. After having performed manual tuning for the SVM hyperparameters, the best results were obtained as in the corresponding row of Table 2. It is worthy to note a couple of differences between our ESVM and the SVM implementation: In the Boston housing case, despite the employment of a linear kernel in the former, the latter produces better results for a radial function, while, in the diabetes task, the ESVMs employ a degree two polynomial and SVMs target it linearly.

The results for each problem were compared via a Wilcoxon rank-sum test. The p-values (see Table 17) suggest to detect significant differences only in the cases of Soybean and Boston data sets. However, the absolute difference is not large for Boston housing, rendering SVM a slight advantage. It may be more relevant for the Soybean task, where ESVM is better.

Table 17 Accuracy/RMSE of canonical SVMs on the considered test sets, in percent, as compared to those obtained by ESVM and p-values from a Wilcoxon rank-sum test

	SVM	**StD**	**ESVM**	**StD**	**p-value**
Diabetes	**76.82**	1.84	**77.31**	2.45	0.36
Iris	**95.33**	3.16	**96.45**	2.36	0.84
Spam	**92.67**	0.64	**91.04**	0.80	0.09
Soybean	**92.22**	9.60	**99.80**	1.06	3.98×10^{-5}
Boston	**3.82**	0.7	**4.78**	0.59	1.86×10^{-6}

Although, in terms of accuracy, the ESVM approach had not achieved better results for some of the test problems, it has many advantages: The decision surface is always transparent even when working with kernels whose underlying transformation to the feature space cannot be determined. The simplicity of the EA makes the solving process easily explained, understood, implemented and tuned for practical usage. Most importantly, any function can be used as a kernel and no additional constraints or verifications are necessary.

From the opposite perspective, the training is relatively slower than that of SVM, as the evaluation always relates to the training data. However, in practice (often, but not always), it is the test reaction that is more important. Nevertheless, by observing the relationship between each result and the corresponding size of the training data, it is clear that SVM performs better than ESVM for larger problems; this is probably due to the fact that, in these cases, much more evolutionary effort would be necessary. The problem of handling large data sets is thus worth investigating deeper in future work.

5 Conclusions and Outlook

The evolutionary learning technique proposed in this chapter resembles the vision upon learning of SVMs but solves the inherent optimization problem by means of an EA. An easier and more flexible alternative to SVMs is put forward and undergoes several enhancements in order to provide a viable alternative to the classical paradigm. These developments are summarized below:

- Two possible representations for the EA (one simpler, and a little faster, and one more complicated, but also more accurate) that determines the coefficients are imagined.
- In order to boost the suitability of the new technique for any issue, a novel chunking mechanism for reducing size in large problems is also proposed; obtained results support its employment.
- The use of a crowding-based EA is inspected in relation to the preservation of performance. Crowding would be highly necessary in the immediate coevolution of nonstandard kernels.
- Finally, an all-inclusive ESVM construction for the practical perspective is developed and validated.
- On a different level, an additional aim was to address and solve real-work tasks of high importance.

Several conclusions can be eventually drawn and the potential of the technique can be further strengthened through the application of two enhancements:

- As opposed to SVMs, ESVMs are much easier to understand and use.
- ESVMs do not impose any kind of constraints or requirements.

- Moreover, the evolutionary solving of the optimization problem enables the acquirement of function coefficients directly and at all times within a run.
- SVMs, on the other hand, are somewhat faster, as the kernel matrix is computed only once.
- Performances are comparable, for different test cases ESVMs and SVMs take the lead, in turn.

Although already a suitable alternative, the novel ESVM can still be enhanced in several ways:

- The requirement for an optimal decision function actually involves two criteria: the surface must fit to training data but simultaneously generalize well. So far, these two objectives are combined in a single fitness expression. As a better choice for handling these conditions, a multicriterial approach could be tried instead.
- Additionally, the simultaneous evolution of the hyperplane and of non-standard kernels will be achieved. This approach is highly difficult by means of SVM standard methods for hyperplane determination, whereas it is straightforward for ESVMs. A possible combination can be achieved through a cooperative coevolution between the population of hyperplanes and that of GP-evolved kernels.

6 Appendix - Definition of Employed Concepts

There are a series of notions that appear throughout the chapter. Their meaning, use and reference are explained in what follows:

Evolutionary algorithm (EA) [18] - a metaheuristic optimization algorithm in the field of artificial intelligence that finds its inspiration in what governs nature: A population of initial individuals (genomes) goes through a process of adaptation through selection, recombination and mutation; these phenomena encourage the appearance of fitter solutions. The best performing individuals are selected in a probabilistic manner as parents of a new generation and gradually the system evolves to the optimum. The fittest individual(s) obtained after a certain number of iterations is (are) the solution to the problem.

Support vector machine (SVM) [2] - a supervised learning method for classification and regression: Given a set of samples, the method aims for the optimal decision hyperplane to model the data and establish an equilibrium between a good training accuracy and a high generalization ability; according to [36], a possible definition of an SVM could be "a system for efficiently training linear learning machines in kernel-induced feature spaces, while respecting the insights of generalization theory and exploiting optimization theory".

Classification / regression hyperplane - the decision hyperplane whose defining coefficients must be determined; within classification, it must differentiate between samples of different classes, while, with respect to regression, it represents the surface on which the data are restrained to be positioned.

multi (k)-class SVM [10] - SVMs are implicitly built for binary classification tasks; for problems with k outcomes, $k > 2$, the technique considers the labels and corresponding samples two by two and uses common approaches like one-against-one and one-against-all to combine the obtained classifiers.

Primal problem - the direct form of the optimization task within SVMs of the determination of the decision hyperplane, while balancing between accuracy and generalization capacity. It is **dualized** in the standard SVM solving by Lagrange multipliers.

Kernel - a function of two variables that defines the scalar product between them; within SVMs, it is employed for the "kernel trick" - a technique to write a nonlinear operator as a linear one in a space of higher dimension as a result of Mercer's theorem.

Crowding-based EA [35] - a technique that was introduced as a method of maintaining diversity: new obtained individuals replace only similar individuals in the population: A percentage G (*generation gap*) of the individuals is chosen via fitness proportional selection in order to create an equal number of offspring; for each of these offspring, CF (a parameter called *crowding factor*) individuals from the current population are randomly selected – the offspring then replaces the most similar individual from these.

References

1. Bosch, R.A., Smith, J.A.: Separating hyperplanes and the authorship of the disputed federalist papers. American Mathematical Monthly 105, 601–608 (1998)
2. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
3. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, New Jersey (1999)
4. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning 1, 273–297 (1995)
5. Mierswa, I.: Evolutionary learning with kernels: A generic solution for large margin problems. In: Proc. of the Genetic and Evolutionary Computation Conference, vol. 1, pp. 1553–1560 (2006)
6. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, 121–167 (1998)
7. Boser, B.E., Guyon, I.M., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, vol. 1, pp. 11–152 (1992)

8. Courant, R., Hilbert, D.: Methods of Mathematical Physics. Wiley Interscience, Hoboken (1970)
9. Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Transactions of the London Philosophical Society (A) 209, 415–446 (1908)
10. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks 13, 415–425 (2004)
11. Platt, J.C., Cristianini, N., Shawe-Taylor, J.: Large margin dags for multiclass classification. In: Proc. of Neural Information Processing Systems, vol. 1, pp. 547–553 (2000)
12. Smola, A.J., Scholkopf, B.: A tutorial on support vector regression. Technical Report NC2-TR-1998-030. NeuroCOLT2 Technical Report Series (1998)
13. Rosipal, R.: Kernel-based Regression and Objective Nonlinear Measures to Access Brain Functioning. PhD thesis Applied Computational Intelligence Research Unit School of Information and Communications Technology University of Paisley, Scotland (2001)
14. Stoean, R.: Support vector machines. An evolutionary resembling approach. Universitaria Publishing House Craiova (2008)
15. Friedrichs, F., Igel, C.: Evolutionary tuning of multiple svm parameters. In: Proc. 12th European Symposium on Artificial Neural Networks, vol. 1, pp. 519–524 (2004)
16. Howley, T., Madden, M.G.: The genetic evolution of kernels for support vector machine classifiers. In: Proc. of 15th Irish Conference on Artificial Intelligence and Cognitive Science, vol. 1 (2004), `http://www.it.nuigalway.ie/m_madden/profile/pubs.html`
17. Jun, S.H., Oh, K.W.: An evolutionary statistical learning theory. International Journal of Computational Intelligence 3, 249–256 (2006)
18. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
19. Stoean, R., Preuss, M., Stoean, C., Dumitrescu, D.: Concerning the potential of evolutionary support vector machines. In: Proc. of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 1436–1443 (2007)
20. Stoean, R., Dumitrescu, D., Preuss, M., Stoean, C.: Different techniques of multi-class evolutionary support vector machines. In: Proc. of Bio-Inspired Computing: Theory and Applications, vol. 1, pp. 299–306 (2006)
21. Stoean, R., Stoean, C., Preuss, M., Dumitrescu, D.: Evolutionary multi-class support vector machines for classification. In: Proceedings of International Conference on Computers and Communications - ICCC 2006, Baile Felix Spa - Oradea, Romania, vol. 1, pp. 423–428 (2006)
22. Stoean, R., Dumitrescu, D., Stoean, C.: Nonlinear evolutionary support vector machines. application to classification. Studia Babes-Bolyai, Seria Informatica LI, 3–12 (2006)
23. Stoean, R., Dumitrescu, D.: Evolutionary linear separating hyperplanes within support vector machines. In: Scientific Bulletin, University of Pitesti. Mathematics and Computer Science Series, vol. 11, pp. 75–84 (2005)
24. Stoean, R., Dumitrescu, D.: Linear evolutionary support vector machines for separable training data. In: Annals of the University of Craiova. Mathematics and Computer Science Series, vol. 33, pp. 141–146 (2006)
25. Stoean, R., Preuss, M., Dumitrescu, D., Stoean, C.: ϵ - evolutionary support vector regression. In: Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2006, vol. 1, pp. 21–27 (2006)

26. Stoean, R., Preuss, M., Dumitrescu, D., Stoean, C.: Evolutionary support vector regression machines. In: IEEE Postproc. of the 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, vol. 1, pp. 330–335 (2006)
27. Stoean, R.: An evolutionary support vector machines approach to regression. In: Proc. of 6th International Conference on Artificial Intelligence and Digital Communications, vol. 1, pp. 54–61 (2006)
28. Stoean, R., Stoean, C., Preuss, M., El-Darzi, E., Dumitrescu, D.: Evolutionary support vector machines for diabetes mellitus diagnosis. In: Proceedings of IEEE Intelligent Systems 2006, London, UK, vol. 1, pp. 182–187 (2006)
29. Stoean, R., Stoean, C., Preuss, M., Dumitrescu, D.: Evolutionary support vector machines for spam filtering. In: Proc. of RoEduNet IEEE International Conference, vol. 1, pp. 261–266 (2006)
30. Stoean, R., Stoean, C., Preuss, M., Dumitrescu, D.: Evolutionary detection of separating hyperplanes in e-mail classification. Acta Cibiniensis LV, 41–46 (2007)
31. Stoean, R., Stoean, C., Preuss, M., Dumitrescu, D.: Forecasting soybean diseases from symptoms by means of evolutionary support vector machines. Phytologia Balcanica 12 (2006)
32. Mierswa, I.: Making indefinite kernel learning practical, technical report. Technical report. Artificial Intelligence Unit, Department of Computer Science, University of Dortmund (2006)
33. Bartz-Beielstein, T.: Experimental research in evolutionary computation - the new experimentalism. Natural Computing Series. Springer, Heidelberg (2006)
34. Perez-Cruz, F., Figueiras-Vidal, A.R., Artes-Rodriguez, A.: Double chunking for solving svms for very large datasets. In: Proceedings of Learning 2004, Elche, Spain, vol. 1 (2004),
`eprints.pascal-network.org/archive/00001184/01/learn04.pdf`
35. DeJong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis University of Michigan, Ann Arbor (1975)
36. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)