Efficient Binary Classification Through Energy Minimisation of Slack Variables

Margarita Kotti[1], Konstantinos I. Diamantaras[2]

Abstract

Slack variables are utilized in optimisation problems in order to build soft margin classifiers that allow for more flexibility during training. A robust binary classification algorithm that is based on the minimisation of the energy of slack variables, called the Mean Squared Slack (MSS), is proposed in this paper. Initially, the algorithm is analysed for the linear case, where the minimum mean squared slack is attained as a separating vector. Next, the kernel trick is exploited to facilitate computation of non-linear separating hyperplanes. For this paper, two kernels are tested, namely the radial basis function (RBF) and the polynomial kernel. In order to ensure a time and memory efficient system that converges in a few iterations four strategies are applied so as to withhold just a subset of feature vectors that are misclassified during training. Aiming to the automatic optimisation of the kernel parameters a modern combination of particle swarm optimisation (PSO) with artificial immune system (AIS) is tested. The aforementioned evolutionary methods are combined in a parallel architecture. Four datasets of diverse nature are exploited for performance evaluation, namely the iris, the SPECTheart, the vertebral column, and the wine quality datasets. Simulation experiments demonstrate high classification accuracy in a number of benchmark datasets.

Keywords:
Slack minimisation, Binary classification, Kernel methods, Genetic optimisation

[1]Corresponding author, Department of Surgery and Cancer, Faculty of Medicine, Imperial College London, Charing Cross Hospital, London W6 8RF email: m.kotti@imperial.ac.uk

[2]K. I. Diamantaras is with the Information Technology Department, ATEI of Thessaloniki, Sindos 57400, Greece e-mail: kdiamant@it.teithe.gr

1. Introduction

Binary classification is a central problem in machine learning. Boser et al. [1] were the first to use kernels to construct a non-linear estimation algorithm, which is the hard margin predecessor of Support Vector Machines (SVM) [2]. The substitution of kernels for dot products transforms a linear geometric algorithm into a non-linear one. This way, hyperplane classifiers evolved to SVMs [3]. SVMs are binary maximum margin classifiers that try to find the hyperplane which optimally separates the data. The feature vectors at the margin are called support vectors and they define the classifier's hyperplane. SVMs present the ability to generalize well even with a limited number of training data. In this paper we employ an SVM alternative which minimises the energy of the slack variables directly, even though that solution may not necessarily yield the maximum margin classifier.

With respect to applications, SVMs are commonly used for example in bio-informatics and natural language processing. This may be partially attributed to the fact that both fields deal with high-dimensional problems, such as micro array processing tasks, fault diagnosis, and categorisation. Additionally, SVMs have been employed in a variety of applications including speech and speaker recognition, emotion classification, e-learning, database marketing, intrusion detection, geo- and environmental sciences, finance time series forecasting, and high energy physics.

Many of the aforementioned fields exploit non-linear SVMs. Non-linear SVMs transform the feature space into a higher dimension one using a set of non-linear basis functions. Hopefully, in the higher dimension feature space the training feature vectors may be separated linearly. An advantage of the SVM is that it is not necessary to explicitly implement this transformation. Instead a kernel representation can be used, where the solution is written as a weighted sum of the values of certain kernel function evaluated at the support vectors. Most recent methods exploit the idea of constructing kernel algorithms where the starting point is a linear criterion instead of a linear algorithm [3]. For example, a linear criterion may be that two samples have identical means or two random variables present zero covariance. Other alternatives aim to improved scalability by utilizing parallel SVM (PSVM) [4]. Parallel SVMs loads only essential data to each machine, which reduces memory use through performing a row-based, approximate matrix factori-

sation. Additional recent theoretical advances include the bounds generalisation based on Rademacher complexity theory for model selection and error estimation [5]. Furthermore, probably approximately correct (PAC) Bayesian theory is utilized to compute a dimension-independent bound of the generalisation error [6].

This paper presents a binary classification algorithm which is based on the minimisation of the energy of the slack variables. A slack variable is defined as zero if the training feature vector is classified correctly and as a small positive value if the training feature vector is classified incorrectly. A maximum margin classifier, such as an SVM seeks to put a soft penalty on the sum of the slack variables, whereas in the approach presented in this paper we attack directly the slack variables of the misclassified patterns. Since many patterns may be classified incorrectly during training, four different strategies are presented in this paper in order to sustain just a subset of the aforementioned training feature vectors. This way time and memory efficiency is achieved. The first strategy retains a subset of the misclassified training feature vectors in a "first come-first kept" approach, the second one retains those patterns in a stochastic manner, the third aims to retain only the "worst" patterns, whereas the final one sustains those patterns whose slack variables attain values within a predefined range.

Next, the kernel trick is utilized in order to facilitate the computation of non-linear separating hyperplanes. Specifically, 2 types of kernels are tested for this paper: the radial basis function (RBF) kernel and the polynomial one. It is widely accepted that the parameters which are related to the aforementioned kernels -that is σ for the RBF kernel and power/offset for the polynomial kernel- may have crucial influence on the classification efficiency. Obviously, the optimal classification accuracy is obtained by optimal parameters setting. Aiming to reach the best performing parameters in an automatic manner, a evolutionary algorithm is employed. The aforementioned algorithm is the combination of particle swarm optimisation (PSO) [7] and artificial immune systems (AIS) [8]. The combination is achieved in a manner of a parallel network. Specifically, in every iteration a pool of u memory cells is constructed and their respective affinity (testing accuracy) is calculated. Best half $u/2$ of the memory cells are selected and given as input to PSO and AIS, independently. New memory cells are produced by each evolutionary algorithm and the best-performing of them are added to the memory cell pool. Also, at each iteration random memory cells are generated to ensure that the size of population is u. The evolutionary algorithm

terminates after a user-defined number of iterations. The proposed approach is tested on four datasets of different nature to verify its robustness using a 5-fold cross-validation experimental protocol. High classification accuracy is achieved, 99.2% for the iris dataset, 80.140%, for the SPECTheart dataset, and 85.769% for the vertebral column dataset.

In summary, our contributions are as follows:

- A novel binary classification algorithm is presented which attacks the slack variables directly.
- It is demonstrated that in the linearly separable case the minimum mean squared slack is attained at a separating vector.
- It is proven that the minimiser in the linearly non-separable case is bounded, but not zero.
- The algorithm is an EM algorithm, so there is convergence, at least in the local sense. Additionally, the evolutionary nature of the parameter estimation algorithm facilitates the escape from local minima. Moreover, one of the proposed strategies is based on the stochastic selection of the subset which also aims at the same direction.
- The algorithm is time and memory efficient since it converges in just a few iterations.
- The algorithm is stable, regardless of the subset retainment strategy that is employed.
- A hybrid evolutionary system is tested, as a parallel network of PSO and AIS in order to select the kernel parameters in an automatic manner.
- The algorithm proves to handle efficiently datasets of highly diverse nature.

The rest of this paper is organized as follows. In Section 2 the proposed algorithm is analysed. In Section 4 we extend from the linear case to the kernel case and the subset retainment strategies are detailed. The evolutionary algorithm for the automatic selection of kernel parameters is detailed in Section 5. Experimental evaluation in demonstrated in Section 6, whereas the presented results are discussed in Section 7. Finally, conclusions are drawn in Section 8.

2. Mean Squared Slack Minimisation

2.1. Problem Formulation

Let us consider the classification task for a set of training data

$$\mathcal{X} = \{(\mathbf{x}(i), t(i)) \mid i = 1, \cdots, N\} \tag{1}$$

where $\mathbf{x}(i) \in \mathbb{R}^n$ is a feature vector, $t(i) \in \{-1, 1\}$ is the class label of $\mathbf{x}(i)$, and N is the size of $\mathcal{X}$. The classification task is described as the search for a proper weight vector $\mathbf{w} \in \mathbb{R}^n$ and bias b that solve the set of inequalities:

$$t(i)y(i) \geq \gamma, \quad i = 1, \cdots, N \tag{2}$$

where

$$y(i) = \mathbf{w}^T\mathbf{x}(i) + b \tag{3}$$

is the output of the classifier for pattern i.

In general, there may not exist any feasible solution for (2). In this case, it is useful to define a slack variable $\xi(i)$, associated with pattern i,

$$\xi(i) = \max\{\gamma - t(i)y(i),\ 0\} \tag{4}$$

so that

$$\xi(i) = 0 \quad \text{iff} \quad t(i)y(i) \ \geq \ \gamma, \tag{5}$$

$$\xi(i) > 0 \quad \text{iff} \quad t(i)y(i) \ = \ \gamma - \xi(i) < \gamma. \tag{6}$$

Thus, the slack variable is positive only for the misclassified patterns, i.e. those with output y less than γ. Typically a maximum margin classifier, such as an SVM, seeks to minimise the norm of the weight vector $\mathbf{w}$ while putting a soft penalty on the sum of the slack variables. Nevertheless, the computational complexity of the resulting quadratic problem may be high.

An alternative approach would be to attack the slack variables directly [9]. Since for misclassified patterns ξ is positive, we can, in fact, define a whole family of cost functions of the form

$$J_p = \frac{1}{2}\bar{E}\left\{\xi^p \,|\, \xi > 0\right\} \tag{7}$$

where $\bar{E}\{X \mid Y\}$ is the empirical average of the sequence $X(i)$ under condition Y:

$$\bar{E}\{X \mid Y\} = \frac{1}{N_Y} \sum_{\substack{\text{all } i \text{ where} \\ Y \text{is true}}} X(i) \tag{8}$$

and N_Y is the number of instances where Y is true. It is not difficult to see that for $p = 1$ we obtain the Perceptron cost function J_1 [10, chapter 5].

Defining

$$S = \{i \ : \ \xi(i) > 0\}, \tag{9}$$

to be the set of indexes of the patterns with positive margin we obtain

$$J_p = \frac{1}{2|S|} \sum_{i \in S} \xi(i)^p.$$

For the case of $p = 2$, we define the **Mean Squared Slack** (**MSS**) as [9]:

$$J_{MSS} = \frac{1}{2|S|} \sum_{i \in S} \big(\gamma - t(i)\mathbf{w}^T\mathbf{x}(i) - bt(i)\big)^2 \tag{10}$$

where $|\cdot|$ stands for the cardinality. Note that the average operator in (10) involves only the patterns which give $x(i) > 0$ or $t(i)(\mathbf{w}^T\mathbf{x}(i) + b) < \gamma$. This is reasonable, since, in the classification context, *only* the misclassified patterns that fail to satisfy inequality (2) should contribute to the cost, while the correctly classified should not.

Previously [9], it had been shown that if the problem is linearly separable, then the minimiser of J_{MSS} is a separating vector, otherwise it is a non-zero, bounded vector. We extend these results here for the more general cost function J_p:

Lemma 2.1. *Consider the cost function J_p defined in* (7)*. The following statements are true:*

(a) if problem (2) *is linearly separable, then the minimum $J_p = 0$ is attained by $\boldsymbol{\omega} = [\mathbf{w}^T, b]$ iff $\boldsymbol{\omega}$ is a separating vector;*

(b) if the problem is not linearly separable, then the cost function J_p attains its minimum for some $\boldsymbol{\omega} = [\mathbf{w}^T, b]$ with $0 < \|\boldsymbol{\omega}\| < \infty$.

Proof.

(a) Clearly, if $\boldsymbol{\omega} = [\mathbf{w}^T, b]$ is a separating vector then $\forall i \ \xi(i) = 0$, and so $J_p = 0$, which is the absolute minimum since $J_p \geq 0$. Reversely, if $J_p = 0$ for some $\boldsymbol{\omega} = [\mathbf{w}^T, b]$ then $\xi(i) = 0$ for all i and so $\boldsymbol{\omega}$ is a separating vector.

(b) First we'll show that the vector $\boldsymbol{\omega} = 0$ cannot be the minimiser of J_p. To that end, select any vector $\boldsymbol{\omega}_1 = \varepsilon[\mathbf{w}_1, b_1] \neq 0$. For sufficiently small $\varepsilon > 0$ we have $\xi(i) = \gamma - \varepsilon t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1) > 0$ for all i, and thus we can write

$$J_p(\mathbf{0}) - J_p(\varepsilon\boldsymbol{\omega}_1) = \sum_{i=1}^{N} \gamma^p - \xi(i)^p = \sum_{i=1}^{N} \gamma^p - \left[\gamma - \varepsilon t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1)\right]^p$$
$$= \varepsilon \sum_{i=1}^{N} t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1) R(i) \tag{11}$$

where

$$R(i) = \sum_{k=0}^{p-1} \xi(i)^k \gamma^{p-1-k}. \tag{12}$$

Since $\xi(i), \gamma > 0$, we have $R(i) > 0$. Further, let $\mathbf{w}_1$, b_1, satisfy

$$\sum_{i=1}^{N} t(i)\mathbf{w}_1^T \mathbf{x}(i) \neq -\sum_{i=1}^{N} t(i) b_1 R(i)$$

(this is always possible, for example, by changing b_1, if necessary). Then, according to (11) either $J_p(\mathbf{0}) - J_p(\varepsilon\boldsymbol{\omega}_1) > 0$ or $J_p(\mathbf{0}) - J_p(\varepsilon\boldsymbol{\omega}_1) < 0$ in which case $J_p(\mathbf{0}) - J_p(-\varepsilon\boldsymbol{\omega}_1) > 0$. In either case, $J_p(\mathbf{0})$ is not the minimum.

Next we'll show that the minimum is attained for a bounded vector $\boldsymbol{\omega}$. Call $\mathcal{I}_+$, $\mathcal{I}_-$ the set of indices, i, for which

$$t(i)[\mathbf{w}_1^T \mathbf{x}(i) + b_1] > 0, \quad \forall i \in \mathcal{I}_+ \tag{13}$$
$$t(i)[\mathbf{w}_1^T \mathbf{x}(i) + b_1] < 0, \quad \forall i \in \mathcal{I}_- \tag{14}$$

It is always possible to select $\mathbf{w}_1$, b_1 such that neither set is empty due to the assumption that the problem is not linearly separable. For example, if $\mathcal{I}_-$ were empty, we could shift b_1 by a small amount so that $t(i)[\mathbf{w}_1^T \mathbf{x}(i) + b_1] < 0$, for some i, without making $\mathcal{I}_+$ empty. Then, for any

$$\varepsilon > \frac{\gamma}{\min_{i \in \mathcal{I}_+}\{t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1)\}} \doteq \varepsilon_{L_1} \tag{15}$$

we'll have $\varepsilon t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1) > \gamma$ so $\xi(i) = 0$, for all $i \in \mathcal{I}_+$. It follows that,

$$J_p(\mathbf{0}) - J_p(\varepsilon\boldsymbol{\omega}_1) = \sum_{i \in \mathcal{I}_+} \gamma^p + \sum_{i \in \mathcal{I}_-} \gamma^p - \xi(i)^p$$
$$= \sum_{i \in \mathcal{I}_+} \gamma^p + \varepsilon \sum_{i \in \mathcal{I}_-} t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1) R(i) \tag{16}$$

with $R(i)$ defined again as in (12). Due to (14) and the fact that $R(i) > 0$, we have

$$\sum_{i \in \mathcal{I}_-} t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1) R(i) < 0 \tag{17}$$

Therefore, for any ε satisfying (15) and

$$\varepsilon > -\frac{\sum_{i \in \mathcal{I}_+} \gamma^p}{\sum_{i \in \mathcal{I}_-} t(i)(\mathbf{w}_1^T \mathbf{x}(i) + b_1) R(i)} \doteq \varepsilon_{L_2} \tag{18}$$

Eq. (16) yields $J_p(\mathbf{0}) < J_p(\varepsilon \boldsymbol{\omega}_1)$ and $\varepsilon \boldsymbol{\omega}_1$ cannot be a minimiser. We conclude that for any line $\boldsymbol{\omega}(\varepsilon) = \varepsilon[\mathbf{w}_1, b_1]$ the part for $\varepsilon > \max\{\varepsilon_{L_1}, \varepsilon_{L_2}\}$ does not contain the minimiser of J_p, so the minimiser must be bounded. □

In the sequel, we shall deal with the quadratic cost J_{MSS}, although most results extend trivially for J_p, $p > 2$. In order to define the binary classifier one needs an algorithm to minimise J_{MSS}. To that end, one may exploit the Karush-Kuhn-Tucker (KKT) conditions $\nabla_{\mathbf{w}} J_{MSS} = 0$, $\partial J_{MSS}/\partial b = 0$. The gradients of J_{MSS} with respect to $\mathbf{w}$ and b, can be computed as follows:

$$\mathbf{g}_w = \mathbf{R}_x \mathbf{w} + b\mathbf{m}_x - \gamma \mathbf{m}_{tx} \tag{19}$$

$$g_b = \mathbf{m}_x^T \mathbf{w} + b - \gamma \mathbf{m}_t \tag{20}$$

where:

$$\mathbf{R}_x = \frac{1}{|S|} \sum_{i \in S} \mathbf{x}(i)\mathbf{x}(i)^T, \tag{21}$$

$$\mathbf{m}_{tx} = \frac{1}{|S|} \sum_{i \in S} t(i)\mathbf{x}(i), \tag{22}$$

$$\mathbf{m}_x = \frac{1}{|S|} \sum_{i \in S} \mathbf{x}(i), \tag{23}$$

$$m_t = \frac{1}{|S|} \sum_{i \in S} t(i). \tag{24}$$

The system of equations that come from the KKT conditions $\mathbf{g}_w = 0$ and $g_b = 0$, is not linear w.r.t. $\mathbf{w}$ and b since $\mathbf{R}_x$, $\mathbf{m}_x$, $\mathbf{m}_{tx}$, and m_t are implicit functions of $\mathbf{w}$ and b through S. Therefore, an iterative solution is proposed. In each iteration, r, we compute the statistics $\mathbf{R}_x$, $\mathbf{m}_x$, $\mathbf{m}_{tx}$, and m_t, using the

values $\mathbf{w}_{r-1}$, b_{r-1}, from the previous iteration. Then we treat the statistics as constants and solve the linear system $\mathbf{g}_w = 0$, $g_b = 0$, to obtain

$$\begin{bmatrix} \mathbf{w}_r \\ b_r \end{bmatrix} = \gamma \begin{bmatrix} \mathbf{R}_x & \mathbf{m}_x \\ \mathbf{m}_x^T & 1 \end{bmatrix}^+ \begin{bmatrix} \mathbf{m}_{tx} \\ m_t \end{bmatrix} \tag{25}$$

where $^+$ stands for the pseudo-inverse matrix. With the new values for the weight vector and bias we compute a new set of statistics and the process is repeated until convergence or until a maximum number of iterations is reached. The method is summarized in Algorithm 1. For the remainder of this work, without loss of generality, we set $\gamma = 1$.

3. Relation with the EM algorithm

The quadratic cost function J_{MSS} in (10) can be expressed as a likelihood function of the "bad" patterns. To see that, let us first define

$$\bar{\mathbf{x}}(i) \stackrel{\text{def}}{=} t(i)[\mathbf{x}(i)^T, 1]^T \text{ and } \boldsymbol{\omega} \stackrel{\text{def}}{=} [\mathbf{w}^T, b]^T.$$

A pattern $\bar{\mathbf{x}}$ will be labelled "bad" if $\bar{\mathbf{x}}^T\boldsymbol{\omega} < 1$, otherwise it will be labelled "good". Consider the probability distribution function

$$\begin{aligned} p(\bar{\mathbf{x}}\,;\boldsymbol{\omega}) &= Z \exp\left\{-\frac{1}{2}\Big[\frac{(\bar{\mathbf{x}}^T\boldsymbol{\omega} - 1)^2}{\sigma^2} + \|\bar{\mathbf{x}}\|^2\Big]\right\} \\ &= Z \exp\left\{-\frac{1}{2}(\bar{\mathbf{x}} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\bar{\mathbf{x}} - \boldsymbol{\mu})\right\} \end{aligned} \tag{26}$$

where $\boldsymbol{\Sigma}^{-1} = [\sigma^{-2}\boldsymbol{\omega}\boldsymbol{\omega}^T + \mathbf{I}]$, $Z = \big[(2\pi)^{-(n+1)}|\boldsymbol{\Sigma}^{-1}|\big]^{1/2}$, and $\boldsymbol{\mu}$ is any vector such that $\boldsymbol{\mu}^T\boldsymbol{\omega} = 1$ (e.g. $\boldsymbol{\mu} = \boldsymbol{\omega}/\|\boldsymbol{\omega}\|^2$). It is easy to see that the eigenvalues of $\boldsymbol{\Sigma}^{-1}$ are: $\lambda_1 = \sigma^{-2}\|\boldsymbol{\omega}\|^2 + 1$, and $\lambda_2 = \cdots = \lambda_{n+1} = 1$, hence, the determinant is $|\boldsymbol{\Sigma}^{-1}| = (\sigma^{-2}\|\boldsymbol{\omega}\|^2 + 1)$.

Consider now the likelihood function

$$\begin{aligned} L &= \sum_i p(\bar{\mathbf{x}}(i), bad\,;\boldsymbol{\omega}) \\ &= \sum_i p(\bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}_k)p(bad\,|\,\bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}) \end{aligned} \tag{27}$$

Clearly,

$$\begin{aligned} p(bad \,|\, \bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}) &= prob(\bar{\mathbf{x}}(i)^T\boldsymbol{\omega} < 1 \,|\, \bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}) \\ &= \begin{cases} 1 & \bar{\mathbf{x}}(i)^T\boldsymbol{\omega} < 1 \\ 0 & \bar{\mathbf{x}}(i)^T\boldsymbol{\omega} \geq 1 \end{cases} \\ &= I_{\bar{\mathbf{x}}(i)^T\boldsymbol{\omega}<1}(\bar{\mathbf{x}}(i)) \end{aligned} \tag{28}$$

where $I_{\bar{\mathbf{x}}(i)^T\boldsymbol{\omega}<1}(\bar{\mathbf{x}}(i))$ is the indicator function for the condition $\bar{\mathbf{x}}(i)^T\boldsymbol{\omega} < 1$ which is equivalent to the condition $\xi(i;\boldsymbol{\omega}) > 0$. The EM algorithm can be employed for the maximization of L. At iteration r, the algorithm makes the following two steps

Expectation step: Compute the expectation

$$\begin{aligned} Q(\boldsymbol{\omega}\,;\boldsymbol{\omega}_r) &= \sum_i p(bad \,|\, \bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}_r) \ln p(\bar{\mathbf{x}}(i), bad\,;\boldsymbol{\omega}) \\ &= \sum_i I_{\bar{\mathbf{x}}(i)^T\boldsymbol{\omega}_r<1}(\bar{\mathbf{x}}(i)) \ln \left[p(\bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}) I_{\bar{\mathbf{x}}(i)^T\boldsymbol{\omega}<1}(\bar{\mathbf{x}}(i))\right] \\ &= \sum_{i:\ \xi(i;\boldsymbol{\omega}_r)>0} \ln \left[p(\bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}) I_{\xi(i;\boldsymbol{\omega})>0}(\bar{\mathbf{x}}(i))\right] \end{aligned} \tag{29}$$

Maximization step:

$$\boldsymbol{\omega}_{r+1} = \arg\max_{\boldsymbol{\omega}} Q(\boldsymbol{\omega}\,;\boldsymbol{\omega}_r) \tag{30}$$

The expectation function Q will contain terms equal to $-\infty$ when $I_{\xi(i;\boldsymbol{\omega})>0}(\bar{\mathbf{x}}(i)) = 0$ and $\xi(i;\boldsymbol{\omega}_r) > 0$. For this reason it is convenient to ignore those cases and focus on maximizing the following function

$$\tilde{Q}(\boldsymbol{\omega}\,;\boldsymbol{\omega}_r) = \sum_{i:\ \xi(i;\boldsymbol{\omega}_r)>0} \ln p(\bar{\mathbf{x}}(i)\,;\boldsymbol{\omega}) \ \geq\ Q(\boldsymbol{\omega}\,;\boldsymbol{\omega}_r) \tag{31}$$

Obviously $S = \{i\ :\ \xi(i;\boldsymbol{\omega}_r) > 0\}$ so we can write

$$\begin{aligned} \tilde{Q}(\boldsymbol{\omega}\,;\boldsymbol{\omega}_r) &= \ln Z + \sum_{i\in S} -\frac{1}{2\sigma^2}\left[(\bar{\mathbf{x}}(i)^T\boldsymbol{\omega} - 1)^2 + \|\bar{\mathbf{x}}\|^2\right] \\ &= \frac{1}{2}\ln\left(\|\sigma^{-1}\boldsymbol{\omega}\|^2 + 1\right) + \boldsymbol{\omega}^T \sum_{i\in S} \bar{\mathbf{x}}(i) \\ &\quad - \boldsymbol{\omega}^T \sum_{i\in S} \bar{\mathbf{x}}(i)\bar{\mathbf{x}}(i)^T\boldsymbol{\omega} + \text{other terms} \end{aligned} \tag{32}$$

where "other terms" stands for terms that are not functions of $\boldsymbol{\omega}$, hence irrelevant to the optimization. For large σ, the term $\ln Z$ can be ignored (zero-order approximation) leading to the function:

$$\tilde{Q}_0(\boldsymbol{\omega}\,;\boldsymbol{\omega}_r) = \mathbf{m}\boldsymbol{\omega} - \boldsymbol{\omega}^T\mathbf{R}\boldsymbol{\omega} \tag{33}$$

where

$$\mathbf{m} = \sum_{i\in S}\bar{\mathbf{x}}(i) = |S|\begin{bmatrix}\mathbf{m}_{tx}\\ m_t\end{bmatrix} \tag{34}$$

$$\mathbf{R} = \sum_{i\in S}\bar{\mathbf{x}}(i)\bar{\mathbf{x}}(i)^T = |S|\begin{bmatrix}\mathbf{R}_x & \mathbf{m}_x\\ \mathbf{m}_x^T & 1\end{bmatrix}. \tag{35}$$

The maximization of $\tilde{Q}_0$ in the M-step yields the same results as (25) (with $\gamma = 1$). Therefore, the proposed algorithm is an approximate EM procedure maximizing the probability of the "bad" data, assuming that the patterns follow a normal distribution as described by (26). We get an insight why such an optimization works by noting that the classifier output $y(i)$ is Gaussian with mean equal to the corresponding target $t(i)$. Indeed, according to (26) the variable $t(i)y(i) = \bar{\mathbf{x}}^T\boldsymbol{\omega}$ follows the Gaussian distribution with mean $\boldsymbol{\mu}^T\boldsymbol{\omega} = 1$. Therefore, the maximization of the probability for the "bad" patterns pushes $t(i)y(i)$ towards 1, ie. it attempts to turn the "bad" patterns into "good".

4. The Kennel Trick

4.1. Problem Formulation

The "kernel trick" can be applied with the help of a non-linear mapping $\boldsymbol{\Phi}$ from the input space $\mathbb{R}^n$ to a higher dimensional space $\mathbb{R}^m$ ($m > n$ or even $m = \infty$). This way the computation of non-linear separating hyperplane is facilitated. We shall avoid the explicit computation of $\boldsymbol{\Phi}()$ using the scalar kernel function

$$K(\mathbf{x},\mathbf{y}) = \boldsymbol{\Phi}(\mathbf{x})^T\boldsymbol{\Phi}(\mathbf{y}). \tag{36}$$

Then, (25) becomes [9]

$$\begin{bmatrix}\sum_{i\in S}\boldsymbol{\Phi}(\mathbf{x}(i))\boldsymbol{\Phi}(\mathbf{x}(i))^T & \sum_{i\in S}\boldsymbol{\Phi}(\mathbf{x}(i))\\ \sum_{i\in S}\boldsymbol{\Phi}(\mathbf{x}(i))^T & |S|\end{bmatrix}\cdot \begin{bmatrix}\sum_{j\in G}\boldsymbol{\Phi}(\mathbf{x}(j))a(j)\\ b\end{bmatrix} = \begin{bmatrix}\sum_{i\in S}t(i)\boldsymbol{\Phi}(\mathbf{x}(i))\\ \sum_{i\in S}t(i)\end{bmatrix}. \tag{37}$$

Call

$$K_{ij} = K(\mathbf{x}(i), \mathbf{x}(j)) \tag{38}$$

and define the row vectors

$$\mathbf{k}_i^T = [K_{ij}]_{j \in G} \qquad \forall i = 1, \cdots, N. \tag{39}$$

Then we have [9]

$$\begin{bmatrix} \sum_{i \in S} \mathbf{\Phi}(\mathbf{x}(i)) \mathbf{k}_i^T & \sum_{i \in S} \mathbf{\Phi}(\mathbf{x}(i)) \\ \sum_{i \in S} \mathbf{k}_i^T & |S| \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i \in S} t(i) \mathbf{\Phi}(\mathbf{x}(i)) \\ \sum_{i \in S} t(i) \end{bmatrix} \tag{40}$$

The novelty of the kernel extension is that we retain a subset G of S [11]. Actually, $|G|$ may be much smaller that $|S|$. Our goal is to find the best strategy for selecting G so that execution is accelerated and use of memory is minimised without compromising performance. To that end, we shall fix an upper limit to $|G|$. Remember that in SVM theory the number of support vectors is typically much smaller than $|S|$. The advantages of the aforementioned approach are two-fold:

- Less computational time is needed since the size of the linear system to be solved in each iteration is smaller.
- Less memory is needed, a feature especially important when handling large-scale problems.

We take the separating vector $\mathbf{w}$ *in* $\mathbb{R}^m$ as a linear combination of the mapped inputs [11]

$$\mathbf{w} = \sum_{j \in G} a(j) \mathbf{\Phi}(\mathbf{x}(j)) \tag{41}$$

where the summation is over a subset G of S. This is a well justified approach since only a few terms in (41) are enough to form a solution. Therefore, the output $y(i) = \mathbf{w}^T \mathbf{\Phi}(\mathbf{x}(i)) + b$ reads

$$y(i) = \sum_{j \in G} K(\mathbf{x}(i), \mathbf{x}(j)) a(j) + b. \tag{42}$$

Thus, next we assume that the parameters $\mathbf{R}_x$, $\mathbf{m}_{tx}$, $\mathbf{m}_x$, m_t, in (25) are produced by summation over the set G (rather than S). So (40) may be simplified into

$$\begin{bmatrix} \mathbf{\Phi}_G \\ \mathbf{1}^T \end{bmatrix} [\mathbf{K}\ \mathbf{1}] \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma \begin{bmatrix} \mathbf{\Phi}_G \\ \mathbf{1}^T \end{bmatrix} \mathbf{t}_G \tag{43}$$

where

$$\mathbf{a} = [a(i)]_{i \in G} \in \mathbb{R}^{|G| \times 1}, \tag{44}$$

$$\mathbf{\Phi}_G = [\mathbf{\Phi}(\mathbf{x}(i))]_{i \in G} \in \mathbb{R}^{m \times |G|}, \tag{45}$$

$$\mathbf{K} = [K_{ij}]_{i \in G, j \in G} = [\mathbf{k}_i]_{j \in G}^T \in \mathbb{R}^{|G| \times |G|}, \tag{46}$$

$$\mathbf{1} = [1, \cdots, 1]^T \in \mathbb{R}^{|G| \times 1}, \tag{47}$$

and

$$\mathbf{t}_G = [t(i)]_{i \in G} \in \mathbb{R}^{|G| \times 1}. \tag{48}$$

It is sufficient, that [9]

$$[\mathbf{K}\ \mathbf{1}] \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma \mathbf{t}_G \tag{49}$$

hence

$$\begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma\, [\mathbf{K}\ \mathbf{1}]^+ \mathbf{t}_G. \tag{50}$$

For this particular paper, we simplify the model by setting $b = 0$ (and γ equal to 1, as it is already stated). Under this setup, Eq. (25) in conjunction with Eq. (41) becomes

$$\mathbf{a} = \mathbf{K}^+ \mathbf{t}_G, \tag{51}$$

4.2. Strategies for forming G

There is a range of ways one may form G. Below we examine four specific approaches for forming the subset G.

4.2.1. The "first come-first kept" approach

For this case we retain the *first* $Z_{\mathbf{a}}$ training feature vectors that are misclassified [11]. In other words, the cardinality of G is $Z_{\mathbf{a}}$. Here, $Z_{\mathbf{a}}$ initially equals n and it may reach a maximum value $Z_{\mathbf{a}}\_max$ through a scale up factor $Z_{\mathbf{a}}\_scale$.

The respective algorithm is summarized in Algorithm 2.

Algorithm 1 Slack minimisation algorithm (linear case)

Input: $MAX\_ITERATIONS$

{INITIALISATION}

Initialize $\mathbf{w}_0, b_0$ to random values

{TRAINING}

for $r = 1 : MAX\_ITERATIONS$ **do**

$y_{r-1}(i) = \mathbf{w}_{r-1}^T \mathbf{x}(i) + b_{r-1}, \quad i = 1, \cdots, N$

$S_{r-1} = \{i \ : \ 1 > t(i) y_{r-1}(i)\}$

{UPDATE RULES}

$\mathbf{X}_r = [\mathbf{x}(i), \ i \in S_{r-1}]$

$\mathbf{t}_r = [t(i), \ i \in S_{r-1}]$

$l_{r-1} \rightarrow$ length of S_{r-1}

$\mathbf{R}_x = \frac{1}{l_{r-1}} [\mathbf{X}_r \ \mathbf{1}][\mathbf{X}_r \ \mathbf{1}]^T$

$\mathbf{m}_x = \frac{1}{l_{r-1}} [\mathbf{X}_r \ \ \mathbf{1}]\mathbf{t}_r$

$res = \mathbf{R}_x^+ \mathbf{m}_x$

$\mathbf{w}_r = res(1 : n)$

$b_r = res(n + 1)$

{TERMINATING CONDITION}

if (misclassified=0 **or** l_{r-1}=0 **or** $\|\mathbf{w}_r\| \geq threshold$) **then**

break

end if

end for

{TESTING ($r = MAX\_ITERATIONS$)}

$y_r(i) = \mathbf{w}_r^T \mathbf{x}(i) + b_r, \quad i = 1, \cdots, N$

return $\mathbf{y}_r$

Algorithm 2 Slack minimisation algorithm (kernel case) - The "first come-first kept" approach

Input: $MAX\_ITERATIONS$, $Z_{\mathbf{a}}\_max$, $Z_{\mathbf{a}}\_scale$

{INITIALISATION}
Initialize $\mathbf{a}_0 = \mathbf{0}$, b=0, and $Z_{\mathbf{a}}$=n
{TRAINING}
for $r = 1 : MAX\_ITERATIONS$ **do**
  $G_{r-1} = [\,]$
  for p=1 TO N **do**
    $\mathbf{k}_{T(p,r-1)} = [K_{pi}]$, $i = 1, \cdots, N$
    $\mathbf{y}(p)_{r-1} = \mathbf{k}_{T(p,r-1)}\mathbf{a}_{r-1} + b$
    $S_{(p,r-1)} = \{p_{r-1} \ : \ 1 > \mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p)\}$
    if $S_{(p,r-1)}$ is true **then**
      $l_{S(p,r-1)} \rightarrow$ length of $S_{(p,r-1)}$
      if $l_{S(p,r-1)} \leq Z_{(\mathbf{a},r-1)}$ **then**
        $G_{(p,r)} = [G_{(p,r-1)} \ \ p_{r-1}]$
      else
        $increaseZ_{\mathbf{a}}flag$ = true
      end if
    end if
  end for
  Compute $\mathbf{K}_r = [K_{ij}]_{i \in G_r, j \in G_r}$
  if $increaseNZ_{\mathbf{a}}flag$ = true **then**
    $Z_{\mathbf{a},r} = min((Z_{\mathbf{a}}\_scale \times Z_{(\mathbf{a},r-1)}), Z_{\mathbf{a}}\_max)$
  end if
  {UPDATE RULES}
  $\mathbf{a}_r = \mathbf{K}_r^+ \mathbf{t}_{(G,r)}$
  {TERMINATING CONDITION}
  if (misclassified=0 **or** $l_{(S,p,r-1)}$=0 **or** $norm(\mathbf{a}_r) \geq threshold$) **then**
    break
  end if
end for
{TESTING ($r = MAX\_ITERATIONS$)}
$\mathbf{K'}_r = [K_{ij}]_{i \in N, j \in N}$
$\mathbf{y}_r = \mathbf{K'}_r \mathbf{a}_r + b$
return $\mathbf{y}_r$

4.2.2. Stochastic formulation of G

This strategy is stochastic in the sense that it produces at each iteration a random number $rand\_num$ which is compared to a user defined parameter $retain\_prob$. For a training feature vector to be retained at G there are two prerequisites.

- First, the current training feature vector should be "worse" that the one encountered at the previous iteration, i.e.

$$t(i)y(i) < t(i-1)y(i-1). \tag{52}$$

  It is reminded that both training feature vectors belong to S.

- Second,

$$rand\_num > retain\_prob. \tag{53}$$

  In other words we retain a training feature vector that belongs to S with probability $rand\_num$ given that (52) is true.

The proposed algorithm is summarized in Algorithm 3.

4.2.3. Retaining the "worst" patterns

In this case a user-defined parameter idx_max determines the maximum number of training feature vectors that satisfy $t(i)y(i) < 1$ and may be retained. That is the the cardinality of G is idx_max. Initially we retain the first idx_max ones, so as it holds $f = \{1, \ldots, idx\_max\}$ training feature vectors that belong to S along with their corresponding values

$$\mathbf{ty} = [t(1)y(1) \quad \ldots \quad t(idx\_max)y(idx\_max)]. \tag{54}$$

Let as also define $max_f = max(\mathbf{ty})$ and min_f=$min(\mathbf{ty})$. In case a new training feature vector j arrives that has a value of $t(j)y(j)$ which is greater that the max_f it substitutes the training feature vector whose index corresponds to min_f. In other words, we retain the idx_max training feature vectors which are closer to the separating hyperplane. The proposed algorithm for the kernel case is summarized in Algorithm 4.

Algorithm 3 Slack minimisation algorithm (kernel case) - Stochastic formulation of G

Input: $MAX\_ITERATIONS$, $Z_{\mathbf{a}}\_max$, $Z_{\mathbf{a}}\_scale$, $retain\_prob$

{INITIALISATION}
Initialize $\mathbf{a}_0 = \mathbf{0}$, b=0, and $Z_{\mathbf{a}}$=n
{TRAINING}
for $r = 1 : MAX\_ITERATIONS$ **do**
  $G_{r-1} = [\,]$
  for p=1 TO N **do**
    $\mathbf{k}_{T(p,r-1)} = [K_{pi}]$, $i = 1, \cdots, N$
    $\mathbf{y}(p)_{r-1} = \mathbf{k}_{T(p,r-1)}\mathbf{a}_{r-1} + b$
    $S_{(p,r-1)} = \{p_{r-1} \ : \ 1 > \mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p)\}$
    if $S_{(p,r-1)}$ is true **then**
      if $rand\_num_{r-1} < retain\_prob$ AND $\mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p) < \mathbf{t}_{r-1}(p-1)\mathbf{y}_{r-1}(p-1)$ **then**
        $l_{S(p,r-1)} \rightarrow$ length of $S_{(p,r-1)}$
        if $l_{S(p,r-1)} \leq Z_{(\mathbf{a},r-1)}$ **then**
          $G_{(p,r)} = [G_{(p,r-1)} \ \ p_{r-1}]$
        else
          $increaseZ_{\mathbf{a},r-1}flag$ = true
        end if
      end if
    end if
  end for
  Compute $\mathbf{K}_r = [K_{ij}]_{i \in G_r, j \in G_r}$
  if $increaseZ_{\mathbf{a},r-1}flag$ = true **then**
    $Z_{\mathbf{a},r} = min((Z_{\mathbf{a}}\_scale \times Z_{\mathbf{a},r-1}), Z_{\mathbf{a}}\_max)$
  end if
  {UPDATE RULES}
  $\mathbf{a}_r = \mathbf{K}_r^{+}\mathbf{t}_{(G,r)}$
  {TERMINATING CONDITION}
  if (misclassified=0 **or** $l_{(S,p,r-1)}$=0 **or** $norm(\mathbf{a}_r) \geq threshold$) **then**
    break
  end if
end for
{TESTING ($r = MAX\_ITERATIONS$)}
$\mathbf{K'}_r = [K_{ij}]_{i \in N, j \in N}$
$\mathbf{y}_r = \mathbf{K'}_r\mathbf{a}_r + b$
return $\mathbf{y}_r$

4.2.4. Retaining those patterns that belong to a range

For this case, we retain those training feature vectors that satisfy $t(i)y(i) \in [MIN, MAX]$. The idea is that if $\xi(i)$ is small enough, then those training feature vectors are closer to the separating hyperplane, and can be chosen to be retained. In other words, possible outliers that have very large $\xi(i)$ are dismissed.

The respective algorithm is summarized in Algorithm 5.

5. Automatic parameter selection

It is a fact that utilizing a kernel prerequisites the tuning of its parameters. For this work we apply the RBF kernel, as well as the polynomial kernel. Thus, the corresponding parameters are σ for the RBF kernel and power and offset for the polynomial kernel. The tuning of those parameters has proven to have a high impact on the classification accuracy, since their selection has an effect on the learning and generation performance. To tune these parameters 20% of the data are utilised. Utilizing the grid algorithm to obtain the optimal classification performance through exhaustive searching would be out of question, since this method is computationally inefficient with respect to time and memory. Accordingly, much research has been devoted to find an automatic way to select parameters in a more efficient way [12].

For the proposed hybrid PSO-AIS model, the slack minimisation algorithm parameters are dynamically and concurrently optimized via a parallel network [13]. PSO is an evolutionary computation, population-based search, iterative optimisation algorithm that is initialized with a population of random solutions. It is biologically inspired by social behaviour among individuals, such as birds flying towards a rewarding position. It has been widely successfully utilized before in order to optimise SVMs parameters [7].

Qualitative, for the PSO algorithm a potential solution to the problem is called a particle. Particles follow the currently optimum particles and also take into account the global best solution. To achieve so, each particle keeps a memory of its previous best position along with its corresponding velocity. At each iteration, each particle's velocity is computed as a combination of the previous particle velocity, the position of the currently optimum particle (that is the best value obtained so far by any particle in that particle's neighborhood), and the position of the global optimum particle. The latter considers the the entire population to be topological neighbors of this specific

Algorithm 4 Slack minimisation algorithm (kernel case) - Retaining the "worst" patterns

Input: $MAX\_ITERATIONS$, idx_max

{INITIALISATION}
Initialize $\mathbf{a}_0 = \mathbf{0}$ and b=0
{TRAINING}
for $r = 1 : MAX\_ITERATIONS$ **do**
  $G_{r-1} = [\ ]$
  for p=1 TO N **do**
    $\mathbf{k}_{T(p,r-1)} = [K_{pi}]$, $i = 1, \cdots, N$
    $\mathbf{y}(p)_{r-1} = \mathbf{k}_{T(p,r-1)}\mathbf{a}_{r-1} + b$
    $S_{(p,r-1)} = \{p_{r-1} \ : \ 1 > \mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p)\}$
    if $S_{(p,r-1)}$ is true **then**
      if $|G_{(p,r-1)}| <= idx\_max$ **then**
        $\mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p) = [\mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p) \quad t_{(p,r)}(i)y_{(p,r)}(i)]$
        $[max_{f(p,r-1)}, max_{idx(p,r-1)}] = max(\mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p))$
        $[min_{f(p,r-1)}, min_{idx(p,r-1)}] = min(\mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p))$
      else
        $\mathbf{t(p)_{r-1}y(p)_{r-1}} = \mathbf{ty} - \{min_{f(p,r-1)}\} \bigcup \{max_{f(p,r-1)}\}$
        $G_{(p,r)} = G_{(p,r-1)} - \{min_{idx(p,r-1)}\} \bigcup \{max_{idx(p,r-1)}\}$
      end if
    end if
  end for
  Compute $\mathbf{K}_r = [K_{ij}]_{i \in G_r, j \in G_r}$
  {UPDATE RULES}
  $\mathbf{a}_r = \mathbf{K}_r^+ \mathbf{t}_{(G,r)}$
  {TERMINATING CONDITION}
  if (misclassified=0 **or** $l_{(S,p,r-1)}$=0 **or** $norm(\mathbf{a}_r) \geq threshold$) **then**
    break
  end if
end for
{TESTING ($r = MAX\_ITERATIONS$)}
$\mathbf{K'}_r = [K_{ij}]_{i \in N, j \in N}$
$\mathbf{y}_r = \mathbf{K'}_r \mathbf{a}_r + b$
return $\mathbf{y}_r$

Algorithm 5 Slack minimisation algorithm (kernel case) - Retaining those patterns that belong to a range

Input: $MAX\_ITERATIONS$, MIN, MAX
{INITIALISATION}
Initialize $\mathbf{a}_0 = \mathbf{0}$, b=0
{TRAINING}
for $r = 1 : MAX\_ITERATIONS$ **do**
$G_{r-1} =$ [ ]
for p=1 TO N **do**
$\mathbf{k}_{T(p,r-1)} = [K_{pi}]$, $i = 1, \cdots, N$
$\mathbf{y}(p)_{r-1} = \mathbf{k}_{T(p,r-1)}\mathbf{a}_{r-1} + b$
$G_{(p,r-1)} = \{p_{r-1} \ : \ MAX > \mathbf{t}_{r-1}(p)\mathbf{y}_{r-1}(p) > MIN\}$
end for
Compute $\mathbf{K}_r = [K_{ij}]_{i \in G_r, j \in G_r}$
{UPDATE RULES}
$\mathbf{a}_r = \mathbf{K}_r^+ \mathbf{t}_{(G,r)}$
{TERMINATING CONDITION}
if (misclassified=0 **or** $l_{(S,p,r-1)}$=0 **or** $norm(\mathbf{a}_r) \geq threshold$) **then**
break
end if
end for
{TESTING ($r = MAX\_ITERATIONS$)}
$\mathbf{K'}_r = [K_{ij}]_{i \in N, j \in N}$
$\mathbf{y}_r = \mathbf{K'}_r \mathbf{a}_r + b$
return $\mathbf{y}_r$

particle. Next, the calculated velocity is used to compute a new position for the particle. This way, all particles share information about the search space by any particles in the swarm.

To put is in more strict mathematical terms, let $\chi_{i,d}$ be the ith particle. Here, t stands for the index of the iteration (also can be considered as time index), d stands for the dimension (that is for RBF kernel $d = 1$, whereas for the polynomial case $d = 2$). The best position of the aforementioned particle is denoted by $p_{i,d}$, whereas the global best position discovered by any particles in the swarm is denoted by $p_{g,d}$. The current velocity at time t is defined as:

$$\begin{aligned} v_{i,d}(t) &= \omega v_{i,d}(t-1) + q_1 r_1(p_{i,d} - \chi_{i,d}(t-1)) + \\ & \quad q_2 r_2(p_{g,d} - \chi_{i,d}(t-1)) \end{aligned} \tag{55}$$

where ω is a weight, q_1 is the personal learning rate, q_2 is the social learning rate, and r_1 and r_2 are random numbers. It is true that $r_1, r_2 \in [0, 1]$. The new position of the particle is defined as:

$$\chi_{i,d}(t+1) = \chi_{i,d}(t) + v_{i,d}(t+1). \tag{56}$$

Literature has proved that each particle converges to a weighted average of its personal best and neighborhood best position. The maximum number of iterations is user defined.

AIS is an emerging soft computing method and accordingly there are limited examples in the literature of SVM classifiers which employ AIS in order to optimize their parameters [8]. AIS is inspired by he natural immune system which is a remarkable information processing and self learning system. Qualitative, in AIS, an antibody is produced as a mean to deal with an antigen i.e. a foreign substance. When an antibody encounters an antigen is for the first time, it takes a long to be constructed. However, when this happens for the second time, the antibody is produced faster in greater quantities. Also, when an antibody is produced, a concentration of it remains into the body. This is equal to mature mutation. This way, the algorithm can escape the local critical points to find the globally best solution.

AIS operators comprise the selection operator, the mutation operator and the crossover operator. The selection operator refers to the way an antibody $\varphi_i(t)$ produces its offspring $\varphi_i(t+1)$. The higher the affinity of the antibody, the more effective the antibody (i.e. the better the solution), and the greater the possibility the antibody will be selected. Mutation allows the parameters

in the antibody to alter. The mutation rate m_r of antibody i is inversely proportional to the antibody's affinity with an antigen $\alpha(i)$. In specific, it is defined as:

$$m_r(i) = \vartheta \exp^{-\alpha(i)}, \tag{57}$$

where ϑ determines the shape of the mutation rate. The mutated antibody is produced according to:

$$\varphi_i(t+1) = \varphi_i(t) + r_3 m_r(i), \tag{58}$$

where r_3 denotes a random number that is normally distributed in the range from 0 to 1. Crossover is a mechanism for exchanging parameters between two antibodies randomly. Here, we apply the one point crossover. For example, if $\varphi_i(t)$ is an antibody with two parameters, i.e. $\varphi_i(t) = [\varphi_{i_1}(t) \;\; \varphi_{i_2}(t)]$ and $\varphi_j(t)$ is a second antibody so that $\varphi_j(t) = [\varphi_{j_1}(t) \;\; \varphi_{j_2}(t)]$, then it crossover is defined as:

$$\varphi_i(t+1) = \begin{cases} [\varphi_{i_1}(t+1) \;\; \varphi_{j_2}(t+1)], & \text{if } c > c_r(i) \\ \varphi_i(t+1), & \text{otherwise} \end{cases} \tag{59}$$

where $c_r(i)$ is the crossover rate and c is a randomly generated number in the range $[0, 1]$.

The proposed PSO-AIS system is initialized with a pull of random solutions, i.e. random antibodies and particles, jointly called memory cells. To begin with, we consider u such memory cells. The user defines the range for the parameters that we seek to minimise, i.e. σ for the case of the RBF kernel and offset and power for the case of the polynomial kernel. The procedure of the proposed PSO-AIS evolutionary approach is as follows:

For each iteration

Step 1 Compute the affinity for the memory cells.

Step 2 Consider the best $u/2$ memory cells with respect to classification accuracy as particles. The same memory cells also construct a population of $u/2$ antibodies.

Step 3 For all particles compute their personal best position as well as the global best position for each dimension.

Step 4 For all particles compute the corresponding velocity according to (55) for each dimension.

Step 5 Revise the position of all particles using (56).

Step 6 If the new position is outside the predefined range, then project it to the matching boundary, i.e. if the new position is greater than the maximum value, set it equal to the maximum value. Accordingly, for the minimum value.

Step 7 Compute the affinity of the updated particles.

Step 8 Move the $u/2$ best particles to the pool of memory cells.

Step 9 For every antibody calculate its respective mutation rate according to (57).

Step 10 Construct the mutated antibodies using (58).

Step 11 Sort the antibodies in a descending order with respect to their affinity. Let the list of the sorted antibodies be: $V = [\varphi_1(t)\varphi_2(t)...\varphi_{u/2}(t)]$.

Step 12 Apply crossover to all the consecutive pairs of V, i.e. $[\varphi_1(t) \ \ \varphi_2(t)]$ ... $[\varphi_{u/2-1}(t) \ \ \varphi_{u/2}(t)]$.

Step 13 Compute the affinity of the new crossed over antibodies.

Step 14 Select the $u/2$ best antibodies and move them to the pool of memory cells.

Step 15 Produce a new random population of $u/2$ memory cells and add them to the memory pool.

Step 16 Goto Step 2.

Until end of iterations

Alternatively, the flowchart of the proposed evolutionary algorithm is depicted in Figure 1.

6. Experimental Evaluation

6.1. Databases

For our experiments we used 3 datasets of radically different nature. All three datasets are available at the UCI machine learning repository [14]. The first dataset is the benchmark iris dataset [14] available at the UCI machine

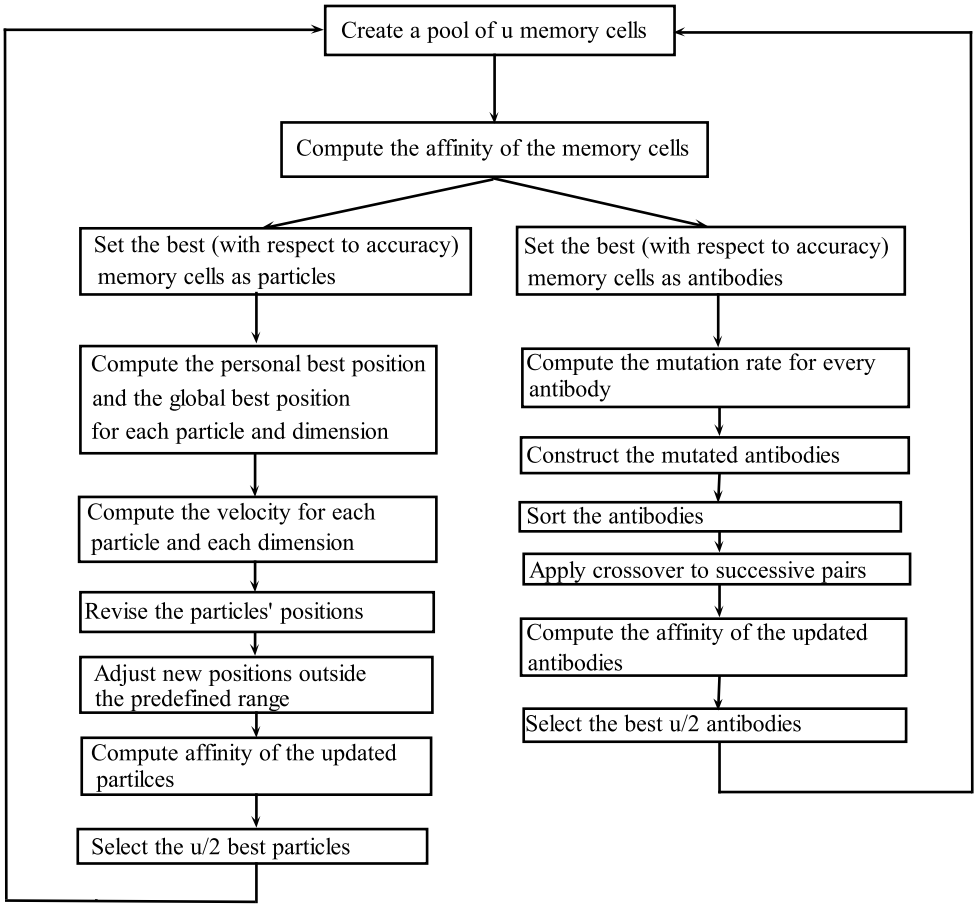


Figure 1: The flowchart of the evolutionary algorithm by means of a parallel network

Table 1: Experimental results for slackmin algorithm (linear kernel)

Dataset	Train Set				
	time elapsed (s)	$accuracy(\%)$	$PRC(\%)$	$RCL(\%)$	$F_1(\%)$
iris	0.002±0.001	73.833±3.416	63.923±3.4323	49.461±6.345	55.654±5.029
SPECTheart	0.013±0.010	91.103±1.567	93.516±1.187	95.383±1.249	94.438±1.093
vertebral column	0.006±0.003	85.565±1.004	77.136±2.077	78.405±2.023	77.756±1.802
	Test Set				
iris	0.001±0.0004	72.667±12.338	61.591±16.338	57.419±29.638	55.660±17.252
SPECTheart	0.007±0.006	80.140±3.215	87.952±4.509	86.898±4.644	87.300±2.565
vertebral column	0.007±0.006	84.839±3.534	76.324±14.443	77.881±9.639	75.925±6.788

Table 2: Experimental results for slackmin algorithm kernel case - first come first served (i.e. Algorithm 2) for the RBF kernel.

Dataset	Selected Parameter	Train Set				
	σ	time elapsed (s)	$accuracy(\%)$	$PRC(\%)$	$RCL(\%)$	$F_1(\%)$
iris	1.269	0.009±0.002	99.833±0.373	99.487±1.147	100±0	99.740±0.581
SPECTheart	13	0.062±0.020	86.679±1.271	93.456±0.944	89.478±0.902	91.423±0.846
vertebral column	3	0.098±0.073	89.794±1.458	93.973±2.894	93.264±3.495	93.538±1.013
	Test Set					
iris	1.269	0.002±0.0006	88±3.801	81.299±5.851	80.619±13.913	80.542±8.429
SPECTheart	13	0.015±0.002	77.561±6.762	84.915±7.949	87.644±7.184	85.949±4.843
vertebral column	3	0.007±0.001	83.818±2.553	73.185±10.255	78.631±9.836	75.019±5.739

Table 3: Experimental results for slackmin algorithm kernel case - first come first served (i.e. Algorithm 2) for the poly kernel.

Dataset	Selected Parameters	Train Set				
	power, offset	time elapsed (s)	$accuracy(\%)$	$PRC(\%)$	$RCL(\%)$	$F_1(\%)$
iris	2, 11	0.006 ±0.001	97.333±1.087	94.245±2.163	98.029±1.113	96.096±1.616
SPECTheart	4, 5	0.113±0.115	79.400±0.961	79.400 ±0.961	100±0	88.515±0.595
vertebral column	2, 8	0.010 ±0.001	87.419±0.776	81.351±1.713	78.674±4.693	79.946±2.975
	Test Set					
iris	2, 11	0.001±0.001	96.667 ±4.083	93.604±6.300	98.462±3.440	95.925±4.586
SPECTheart	4, 5	0.041±0.025	79.385±3.865	79.385±3.865	100±0	88.466±2.435
vertebral column	2, 8	0.003±0.0004	84.839±5.657	76.433±18.339	76.680±8.200	75.361 ±10.283

learning repository. The dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant. We form a binary problem by attempting to separate class "Versicolor", that counts a total of 50 samples, against the other two classes combined ("Virginica" and "Setosa") which contain 100 samples. A total of 4 features is available for each sample.

The second dataset refers to the diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) heart images. Each patient is classified into one of two categories: normal (212 patients) and abnormal (55 patients). That is the SPECTheart dataset has 267 instances that are described by 44 features.

An additional dataset, related to classification of pathologies of the vertebral column is considered. The dataset contains values for 6 biomedical features derived from the shape and orientation of the pelvis and lumbar spine. The aforementioned biomedical features are utilized to classify orthopaedic patients into 2 classes normal (100 patients) or abnormal (210 patients). The dataset is one of the most recent additions to UCI machine learning repository.

Table 4: Experimental results for slackmin algorithm kernel case - stochastic formulation of G (i.e. Algorithm 3) for the RBF kernel.

Dataset	Selected Parameter σ	Train Set				
		time elapsed (s)	$accuracy(\%)$	$PRC(\%)$	$RCL(\%)$	$F_1(\%)$
iris	0.971	0.003±0.001	73.833±3.416	63.923±3.432	49.461±6.345	55.654±5.029
SPECTheart	2.877	0.062±0.020	86.679±1.271	93.456±0.944	89.478±0.902	91.423±0.846
vertebral column	3	0.0196±0.002	95.565±0.638	93.224±2.784	92.978±1.215	93.074±1.227
	Test Set					
iris	0.971	0.002 ±0.0001	86.667±6.667	78.452±19.195	89±15.166	80.975±9.524
SPECTheart	2.877	0.018±0.011	73.244±10.103	86.056±6.924	78.660±8.626	82.099±7.298
vertebral column	3	0.007±0.001	83.818±2.553	73.185±10.255	78.631±9.836	75.019±5.739

Table 5: Experimental results for slackmin algorithm kernel case - stochastic formulation of G (i.e. Algorithm 3) for the polynomial kernel.

Dataset	Selected Parameter power, offset	Train Set				
		time elapsed (s)	$accuracy(\%)$	$PRC(\%)$	$RCL(\%)$	$F_1(\%)$
iris	2, 6	0.008±0.002	95±1.954	87.109±4.731	100±0	93.057±2.631
SPECTheart	12, 12.121	0.096±0.056	85.398±6.466	92.418±8.481	90.232±10.245	90.656±4.553
vertebral column	2, 7	0.014±0.003	87.177±0.721	81.360±2.611	78.297±2.623	79.743±1.156
	Test Set					
iris	2, 6	0.004±0.004	93.333±3.333	82.400 ±9.297	100±0	90.122 ±5.619
SPECTheart	12, 12.121	0.033±0.009	75.290±8.054	86.586±5.530	80.987±10.951	83.358±6.767
vertebral column	2, 7	0.003±0.0007	85.484±2.550	78.014±7.674	78.704±9.541	77.704±2.907

6.2. Experimental Protocol

As a prerequisite to classification, normalisation takes place. Feature normalisation improves the features generalisation ability, guarantees that all the features obtain the same scale, and helps to address the problem of outliers. All features are subject to mean and variance normalisation, that is each feature vector is transformed to present zero mean and unit variance hereafter.

All the experiments are carried out on on a 2.00 GHz processor with 4GB of RAM, with a Windows-7 32 bit operating system. The software platform exploited is MATLAB®R2011b [15]. For all the experiments presented in this Section, a 5-fold cross validation strategy is applied. Cross-validation helps up to verify the stability of the proposed classification method, whereas it is assured that all the feature vectors in the dataset are eventually used for both training and testing.

Classifier performance is evaluated through several sets of figures of merit to capture diverse aspects of efficiency. Let us define as tp as true positive, fn as false negative, fp as false positive, and tn as true negative. Then, is

Table 6: Experimental results for slackmin algorithm kernel case - retaining the "worst" patterns (i.e. Algorithm 4) for the RBF kernel.

Dataset	Selected Parameter σ	Train Set				
		time elapsed (s)	*accuracy*(%)	PRC(%)	RCL(%)	F_1(%)
iris	0.007	0.003±0.001	73.833±3.416	63.923±3.432	49.461±6.345	55.654±5.029
SPECTheart	15.471	0.059±0.024	97.847±0.967	98.936±0.967	98.352±0.484	98.641±0.608
vertebral column	2.293	0.013 ±0.0003	99.677±0.180	99.012±0.553	100±0	99.503±0.2782
		Test Set				
iris	0.007	0.002 ±0.0002	98.750±2.795	100±0	96±8.944	97.778±4.969
SPECTheart		0.016±0.004	79.413±3.385	85.197±2.611	89.636±3.543	87.326±2.388
vertebral column	2.293	0.003±0.002	85.754±3.146	76.039±4.280	82.804±10.528	78.880±5.035

Table 7: Experimental results for slackmin algorithm kernel case - retaining the "worst" patterns (i.e. Algorithm 4) for the polynomial kernel.

Dataset	Selected Parameter power, offset	Train Set				
		time elapsed (s)	*accuracy*(%)	PRC(%)	RCL(%)	F_1(%)
iris	2, 15	0.006 ±0.0004	97.500±1.021	94.732±1.917	98.035±1.100	96.352±1.452
SPECTheart	10, 15	0.124±0.007	99.064±0.330	99.527±0.265	99.293±0.262	99.409±0.209
vertebral column	5, 0.0001	0.086 ±0.014	99.597±0.285	98.792 ±0.842	100±0	99.391±0.426
		Test Set				
iris	2, 15	0.001 ±0.0003	96.667±2.357	92.000±8.367	98.462±3.440	94.873±3.982
SPECTheart	10, 15	0.045±0.014	72.663±3.097	87.007±4.721	77.300±3.953	81.740±2.393
vertebral column	5, 0.0001	0.075±0.066	85.759±3.313	73.742±2.852	85.442±10.272	79.040±5.977

Table 8: Experimental results for slackmin algorithm kernel case - retaining those patterns that belong to a range (i.e. Algorithm 5) for the RBF kernel.

Dataset	Selected Parameter σ	Train Set				
		time elapsed (s)	*accuracy*(%)	PRC(%)	RCL(%)	F_1(%)
iris	0.009	0.010 ±0.006 1	100±0	100±0	100±0	100±0
SPECTheart	2.702	0.043±0.006	100±0	100±0	100±0	100±0
vertebral column	5	0.018 ±0.004	100±0	100±0	100±0	100 ±0
		Test Set				
iris	0.009	0.001 ±0.0006	99.200±1.789	100±0	98±4.472	98.947±2.354
SPECTheart	2.702	0.015±0.004	73.890±5.636	85.652 ±7.022	80.400±3.380	82.839±4.194
vertebral column	5	0.008±0.006	85.769±5.725	76.596±8.642	82.596±10.582	78.990±6.131

Table 9: Experimental results for slackmin algorithm kernel case - retaining those patterns that belong to a range (i.e. Algorithm 5) for the polynomial kernel.

Dataset	Selected Parameter power, offset	Train Set				
		time elapsed (s)	*accuracy*(%)	PRC(%)	RCL(%)	F_1(%)
iris	2, 10	0.005 ±0.0006	97.333±0.697	93.895±1.540	98.420±1.443	96.094±0.956
SPECTheart	15, 10	0.035±0.007	100±0	100±0	100±0	100±0
vertebral column	3, 15.312	0.027 ±0.0005	89.597±1.319	84.366±1.641	83.228±2.0767	83.784±1.582
		Test Set				
iris	2, 10	0.002 ±0.001	96.667±4.083	88.962±15.461	100±0	93.532±9.580
SPECTheart	15, 10	0.025±0.009	72.690±4.243	87.068±5.493	77.611±5.324	81.827±2.353
vertebral column	3, 15.312	0.009±0.001	84.516±4.785	74.616±3.628	78.701±7.005	76.480±4.266

is true that

$$\begin{aligned} accuracy &= 100 \times (tp + tn)/(tp + tn + fp + fn) \\ PRC &= 100 \times tp/(tp + fp) \\ RCL &= 100 \times tp/(tp + fn) \\ F_1 &= (2 \times PRC \times RCL)/(PRC + RCL). \end{aligned}$$

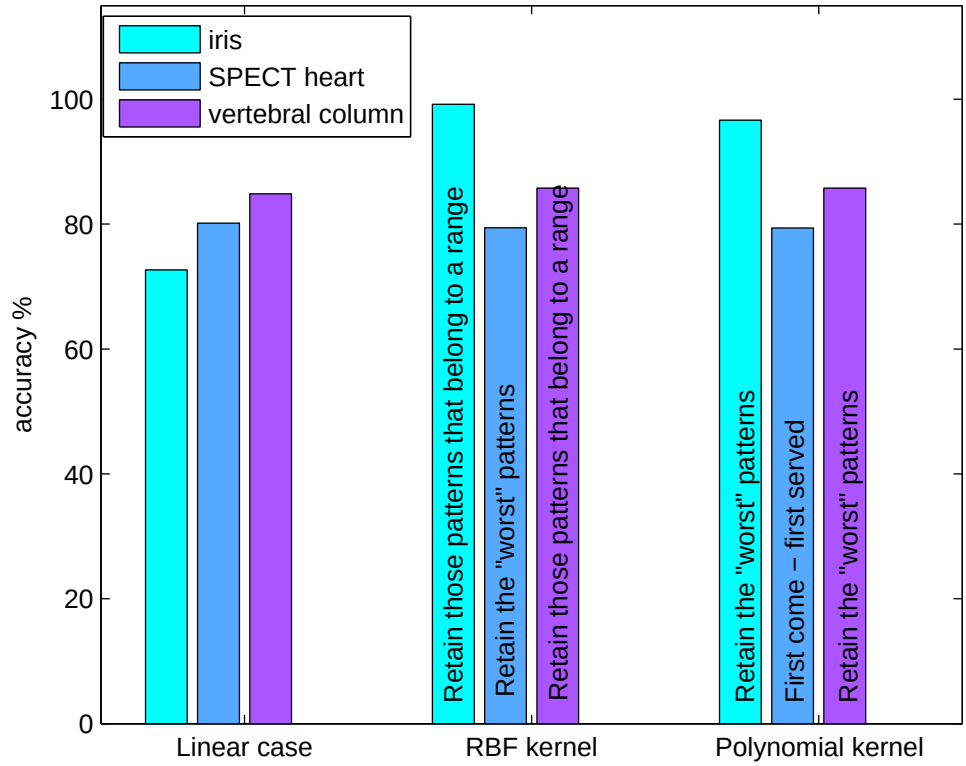
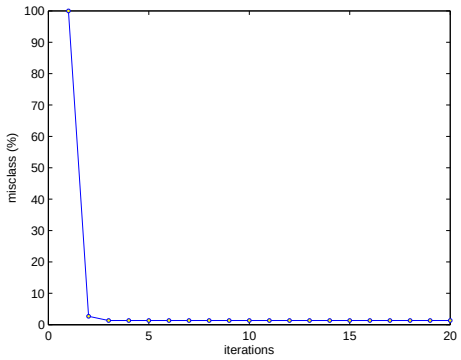


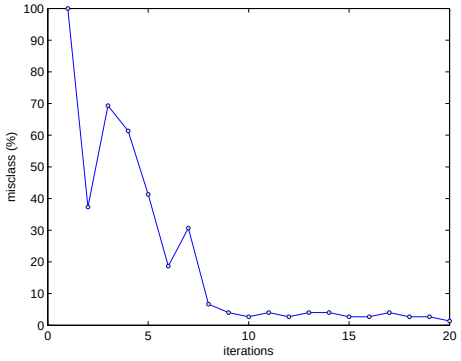
Figure 2: Best accuracies per dataset and per method (i.e. linear or kernel) achieved by the slack minimisation algorithm. The strategy which accomplish the best accuracy is also stated.

.

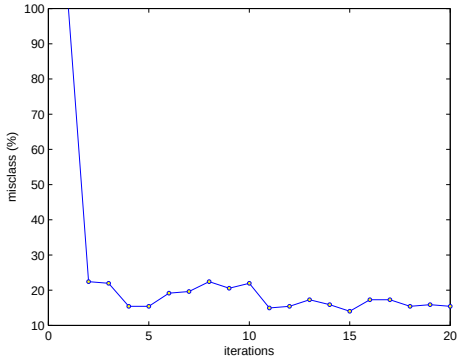
Experimental results for the linear case of proposed slack minimisation algorithm, i.e. Algorithm 1 are detailed in Table 1. The reported figures of



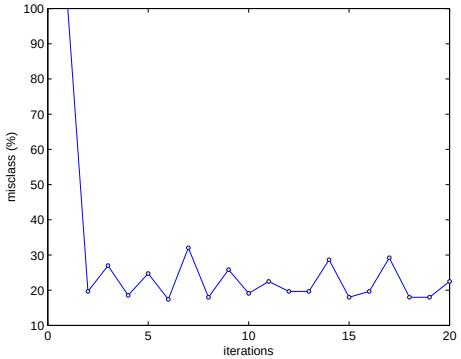
(a) The misclassification error as a function of the iterations for the iris dataset for the slack minimisation algorithm - Stochastic formulation of G.



(b) The misclassification error as a function of the iterations for the iris dataset for the slack minimisation algorithm - Retaining the "worst" patterns.



(c) The misclassification error as a function of the iterations for the SPECTheart dataset for the slack minimisation algorithm - Stochastic formulation of G.



(d) The misclassification error as a function of the iterations for the SPECTheart dataset for the slack minimisation algorithm - Retaining the "worst" patterns

Figure 3: The way the classification error develops with respect to the number of iterations

merit are averaged over the 5 folds, whereas the standard deviation is also provided for each figure of merit. Also, the execution time is provided to verify the fact that the slack minimisation algorithm presents a low execution time. For the kernel cases, the parameters selected by the PSO-AIS evolutionary algorithm are stated, as well. The respective results for Algorithm 2 are depicted in Table 2 for the RBF kernel and in Table 3 for the polynomial kernel. For Algorithm 3, the corresponding results are shown in Tables 4-5. For Algorithm 4 the results for the RBF kernel are shown in Table6 and for the polynomial case in Table 7. Finally, the results for Algorithm 5 are demonstrated in Table 8 for the RBF kernel and in Table 9 for the kernel case.

7. Discussion

7.1. Discussion on the results of the proposed method

To provide a concise overview of the reported results along the three datasets the best accuracy per method (i.e. linear and kernel - RBF and polynomial) is depicted in Figure 2. For the kernel case, we confine ourselves to the best performing strategy. In general, it seems that the retaining the "worst" patterns as well as those that belong to a range leads to improved classification accuracy. This is in line with the intuition that the closer the misclassified training feature vectors are to the boundary, i.e. the closer ξ is to 0, the more discriminating power those specific training feature vector have. Specifically, the strategy of retaining the "worst" patterns determined how many training feature vectors to chose, whereas the strategy of selecting those patterns that belong to a range specifies the "width" of the separating hyperplane.

One advantage of the proposed algorithm is the fact that the accuracy is high even for the simple linear case (with the exception of the iris dataset), which is less computationally demanding and requires no parameter tuning. A second advantage is the high efficiency of the algorithm for diverse datasets as well as its suitability for biomedical datasets, which in principle exhibit great inter-subject variability. Such datasets are the SPECTheart and the vertebral column. Slack minimisation algorithm generalises well even when a limited amount of training data is available, which is an additional advantage since obtaining more datapoints for biomedical datasets is often expensive or impractical.

Another advantage of the proposed algorithm that it manages to converge after a few iterations. It is remineded that the algorithm, as an EM algorithm, is certain to converge. Aiming to visualize the way the classification error develops with respect to the number of iterations, we provide the reader with Figure 3. The aim of the aforementioned Figure is to provide visual proof of the way that the misclassification error reduces as the number of iterations increases. To illustrate that we confine ourselves to two datasets and two strategies (since the exhaustive combination of dataset, strategies, and kernels would produce an excess of figures). In specific Figure 3(a) and 3(b) refer to the iris dataset, whereas the remaining figures are related to SPECTheart dataset. The two tested strategies are the stochastic formulation of G as well as the retaining of the "worst" patterns. From the aforementioned Figures it is clear that that the misclassification error is reduced in a high rate within just a few iterations (typically less than 5), whereas for the remaining iterations the misclassification error tends to remain at the reduced level. In other words, the proposed algorithm has the advantage of converging in just a few iterations. This is also verified by Tables 1-9. For the iris dataset for example, the maximum training time equals 0.010 s whereas the minimum training time is 0.002 s. It worth noticing that the minimum times are achieved for the tuning free linear case, since it requires less computations than the kernel case. The maximum execution times are observed for the RBF kernel case, since its quadratic form is computationally most demanding.

7.2. Comparison with SVM

For comparison reasons we compare the performance of our approach to standard SVM algorithm, as it is implemented at MATLAB®bioinformatics toolbox [15]. The exact same protocol is applied, which means that a 5-fold cross validation is applied, whereas the kernel parameters are selected automatically by the evolutionary algorithm, which means that the two approached may conclude to different parameters after tuning. In short, for all but one cases, as can be seen in Table 10 the proposed algorithm achieves a better classification accuracy that its SVM counterpart. Also, in the same table the results of the equivalent soft margin SVM are demonstrated, where the slack penalty coefficient is set to a very large number, as proposed by MATLAB®bioinformatics toolbox [15]. In that case the proposed algorithm consistently outperforms the standard SVM approach. No convergence indicates that at least one fold failed to converge. With respect to execution

Table 10: Comparison of accuracy with the standard SVM

dataset	standard SVM accuracy(%)	SVM accuracy with large penalty(%)	selected parameters	proposed algorithm accuracy(%)	selected parameters
Linear					
iris	72.000 ±7.674	No convergence	-	72.667±12.338	-
SPECT heart	76.408 ± 4.678	No convergence	-	80.140±3.215	-
vertebral column	85.161±4.174	No convergence	-	84.839±3.534	-
RBF kernel					
iris	95.333 ± 1.825	93.333 ± 0.333	2.022	99.200±1.789	0.009
SPECT heart	78.686 ± 4.476	79.406 ± 1.793	1.161	79.413±3.385	15.471
vertebral column	84.193 ± 5.275	No convergence	3.862	85.769±5.725	5
Polynomial kernel					
iris	95.334±1.826	94.000±2.789	2, 0	96.667±2.357	2, 15
SPECT heart	67.470± 8.587	75.653± 5.730	2, 0	79.385±3.865	4, 5
vertebral column	84.193 ± 2.103	No convergence	2, 0	85.759±3.313	5, 0.0001

time, the proposed method presents consistently smaller execution times when compared to SVM and soft margin SVM, as is obvious from Table 11.

7.3. Expanding to larger datasets

To check the proposed algorithm's efficiency on larger datasets, we exploited an additional dataset from the UCI machine learning repository [14], namely the wine quality dataset. The dataset contains 10 classes unbalanced classes that correspond to the quality of the wine. For this paper, we consider 2 classes, 1 for those wines that achieve a score less or equal to five and the second one for those that are attributed a score greater than 5. The number of features is 12 and the number of samples equals 4898.

With respect to the proposed algorithm's efficiency, for the linear case it equals 74.203 ± 0.001%. With respect to the "first come-first kept" strategy, for the RBF kernel accuracy is 69.288± 0.736%, for the stochastic formulation of G accuracy equals 70.458± 1.875%, for retaining the "worst" patterns accuracy rises to, 73.600± 1.177% and for retaining those patterns that be-

Table 11: Comparison of execution time with the standard SVM (time refers to the training of one fold)

dataset	standard SVM (s)	SVM with large penalty (s)	proposed algorithm (s)
Linear			
iris	0.242	No convergence	0.003
SPECT heart	1.201	No convergence	0.017
vertebral column	0.843	No convergence	0.006
RBF kernel			
iris	0.076	2.065	0.006
SPECT heart	0.358	0.586	0.019
vertebral column	0.148	No convergence	0.008
Polynomial kernel			
iris	0.030	1.625	0.001
SPECT heart	0.040	0.6832	0.045
vertebral column	0.206	No convergence	0.065

long to a range accuracy equals 74.214± 1.257%. For the polynomial kernel the respective figures are: 66.677 ± 1.368%, 74.726 ± 1.020%, 74.188 ± 1.142%, and 64.676 ± 3.916%.

The standard SVM algorithm, as it is implemented at MATLAB®bioinformatics toolbox [15] achieves no convergence for the linear and the polynomial case, whereas for the RBF kernel accuracy equals 76.496± 0.002%. For the soft margin SVM, no convergence is achieved for the linear, RBF and polynomial kernel case.

8. Conclusions

This paper presents the slack minimisation algorithm, a binary classification algorithm which minimises the energy of the slack variables to allow for a more flexible separating hyperplane. It is proven that in the linearly separable case the minimum mean squared slack is attained at a separating vector, that the minimiser in the linearly non-separable case is bounded, but not zero; and that the algorithm belongs to the EM family. Additionally, the kernel case is analysed, and specifically the RBF as well as the polynomial kernel is tested. The algorithm is computationally efficient and fast, since it manages to converge in just a few iterations. Additional time and memory efficiency is achieved by selecting a limited subset of misclassified training feature vectors via four distinct strategies. Automatic optimisation of kernel

parameters is attained via an evolutionary approach. Specifically, a computational efficient parallel combination of AIS and PSO is examined that dynamically tunes the kernel parameters for each method concurrently. hree datasets of radically different nature have been utilised namely the iris, the SPECTheart, and the vertebral column dataset to verify the high efficiency as well as the proposed approach robustness and stability. The efficiency of the algorithm is verified in terms of figures of merit as well as execution time.

In the future, the proposed system can be exploited as a base classifier of an ensemble system. Additionally, more datasets may be tested, so as to study the performance of the proposed algorithm for diverse classification problems. Furthermore, more sophisticated kernels, such as those that are based on wavelets could be tested. Alternative powers of the slsck variable ξ may be analysed to further handle the shape of the separating hyperplane. Finally, the nature of the algorithm and specifically the fact that each misclassified training feature vector is chosen independent of the others renders the algorithm suitable for parallel programming.

Acknowledgement

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALES. Investing in knowledge society through the European Social Fund.

Bibliography

[1] B. E. Boser, I. M. Guyon, V. N. Vapkin, A training algorithm for optimal margin classifiers, in: Proc. 5th annual ACM Workshop on Computational Learning Theory, ACM, 1992, pp. 144–152.

[2] V. Vapnik, Estimation of dependences based on empirical data, Springer-Verlang, 1982.

[3] T. Hofmann, B. Scholkopf, A. J. Smola, Kernel methods in machine learning, Annals of Statistics 36 (2008) 1171–1220.

[4] E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, PSVM: Parallelizing support vector on distributed computers, in: Proc. 21st

Annual Conference on Neural Information Processing Systems, Vol. 20, NIPS, 2007, pp. 144–152.

[5] L. Oneto, D. Anguita, A. Ghio, S. Ridella, Rademacher complexity and structural risk minimization: An application to human gene expression datasets, in: Proc. Int. Conf. Artificial Neural Networks, 2012, pp. 491–498.

[6] L. Tang, Z. Zhao, X. Gong, H. Zeng, On the generalization of PAC-Bayes bound for SVM linear classifier, in: Proc. Int. Conf. E-business Technology and Strategy, 2012, pp. 176–186.

[7] C.-L. Huang, J.-F. Dun, A distributed PSO-SVM hybrid system with feature selection and parameter optimization, Applied Soft Computing 8 (4) (2008) 1381 – 1391.

[8] I. Aydin, M. Karakose, E. Akin, A multi-objective artificial immune algorithm for parameter optimization in support vector machine, Applied Soft Computing 11 (1) (2011) 120 – 129.

[9] K. I. Diamantaras, M. Kotti, Binary classification by minimizing the mean squared slack, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, IEEE, 2012.

[10] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification 2nd Ed., Wiley, 2001.

[11] M. Kotti, K. I. Diamantaras, Towards minimizing the energy of slack variables for binary classificaton, in: Proc. 20th European Signal Procesing Conference (EUSIPCO), EURASIP, 2012.

[12] J. Shawe-Taylor, S. Sun, A review of optimization methodologies in support vector machines, Neurocomputing 74.

[13] R. J. Kuo, C. M. Chen, Evolutionary-based support vector machine, in: Proc. IEEE Int. Conf. Industrial Engineering and Engineering Management, IEEE, 2011, pp. 472–475.

[14] A. Frank, A. Asuncion, UCI machine learning repository (2010).
URL `http://archive.ics.uci.edu/ml`

[15] MATLAB, version 7.13.0.564 Natick (2011).