Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance

V. Kecman[1], T.-M. Huang[1], and M. Vogt[2]

[1] School of Engineering, The University of Auckland, Auckland, New Zealand
v.kecman@auckland.ac.nz
[2] Institute of Automatic Control, TU Darmstadt, Darmstadt, Germany
mvogt@iat.tu-darmstadt.de

Abstract. The chapter introduces the latest developments and results of Iterative Single Data Algorithm (ISDA) for solving large-scale support vector machines (SVMs) problems. First, the equality of a Kernel AdaTron (KA) method (originating from a gradient ascent learning approach) and the Sequential Minimal Optimization (SMO) learning algorithm (based on an analytic quadratic programming step for a model without bias term b) in designing SVMs with positive definite kernels is shown for both the nonlinear classification and the nonlinear regression tasks. The chapter also introduces the classic Gauss-Seidel procedure and its derivative known as the successive over-relaxation algorithm as viable (and usually faster) training algorithms. The convergence theorem for these related iterative algorithms is proven. The second part of the chapter presents the effects and the methods of incorporating explicit bias term b into the ISDA. The algorithms shown here implement the single training data based iteration routine (a.k.a. per-pattern learning). This makes the proposed ISDAs remarkably quick. The final solution in a dual domain is not an approximate one, but it is the optimal set of dual variables which would have been obtained by using any of existing and proven QP problem solvers if they only could deal with huge data sets.

Key words: machine learning, huge data set, support vector machines, kernel machines, iterative single data algorithm

1 Introduction

One of the mainstream research fields in learning from empirical data by support vector machines (SVMs), and solving both the classification and the regression problems, is an implementation of the incremental learning schemes when the training data set is huge. The challenge of applying SVMs on huge data sets comes from the fact that the amount of computer memory required

V. Kecman, T.-M. Huang and M. Vogt: *Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance*, StudFuzz **177**, 255-274 (2005)
www.springerlink.com

for a standard quadratic programming (QP) solver grows exponentially as the size of the problem increased. Among several candidates that avoid the use of standard QP solvers, the two learning approaches which recently have drawn the attention are the Iterative Single Data Algorithms (ISDAs), and the sequential minimal optimization (SMO) [9, 12, 17, 23].

The ISDAs work on one data point at a time (per-pattern based learning) towards the optimal solution. The Kernel AdaTron (KA) is the earliest ISDA for SVMs, which uses kernel functions to map data into SVMs' high dimensional feature space [7] and performs AdaTron learning [1] in the feature space. The Platt's SMO algorithm is an extreme case of the decomposition methods developed in [10, 15], which works on a working set of two data points at a time. Because of the fact that the solution for working set of two can be found analytically, SMO algorithm does not invoke standard QP solvers. Due to its analytical foundation the SMO approach is particularly popular and at the moment the widest used, analyzed and still heavily developing algorithm. At the same time, the KA although providing similar results in solving classification problems (in terms of both the accuracy and the training computation time required) did not attract that many devotees. There are two basic reasons for that. First, until recently [22], the KA seemed to be restricted to the classification problems only and second, it "lacked" the fleur of the strong theory (despite its beautiful "simplicity" and strong convergence proofs). The KA is based on a gradient ascent technique and this fact might have also distracted some researchers being aware of problems with gradient ascent approaches faced with possibly ill-conditioned kernel matrix. In the next section, for a missing bias term b, we derive and show the equality of two seemingly different ISDAs, which are a KA method and a without-bias version of SMO learning algorithm [23] in designing the SVMs having positive definite kernels. The equality is valid for both the nonlinear classification and the nonlinear regression tasks, and it sheds a new light on these seemingly different learning approaches. We also introduce other learning techniques related to the two mentioned approaches, such as the classic Gauss-Seidel coordinate ascent procedure and its derivative known as the successive over-relaxation algorithm as a viable and usually faster training algorithms for performing nonlinear classification and regression tasks. In the third section, we derive and show how explicit bias term b can be incorporated into the ISDAs derived in the second section of this chapter. Finally, the comparison in performance between different ISDAs derived in this chapter and the popular SVM software LIBSVM [2] is presented. The goal of this chapter is to show how the latest developments in ISDA can lead to the remarkable tool for solving large-scale SVMs as well as to present the effect of an explicit bias term b within the ISDA.

In order to have a good understanding on these algorithms, it is necessary to review the optimization problem induced from SVMs. The problem to solve in SVMs classification is [3, 4, 20, 21]

$$\min \ \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}, \quad i = 1, \ldots, l\,, \tag{1}$$

$$\text{s.t.} \ \ y_i\big[\mathbf{w}^{\mathrm{T}}\boldsymbol{\Phi}(\mathbf{x}_i) + b\big] \geq 1, \quad i = 1, \ldots, l\,, \tag{2}$$

which can be transformed into its dual form by minimizing the primal Lagrangian

$$L_{\mathrm{p}}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} - \sum_{i=1}^{l} \alpha_i \big\{ y_i\big[\mathbf{w}^{\mathrm{T}}\boldsymbol{\Phi}(\mathbf{x}_i) + b\big] - 1 \big\}\,, \tag{3}$$

in respect to $\mathbf{w}$ and b by using $\partial L_{\mathrm{p}}/\partial \mathbf{w} = 0$ and $\partial L_{\mathrm{p}}/\partial b = 0$, i.e., by exploiting

$$\frac{\partial L_{\mathrm{p}}}{\partial \mathbf{w}} = 0 \qquad \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{\Phi}(\mathbf{x}_i)\,, \tag{4}$$

$$\frac{\partial L_{\mathrm{p}}}{\partial b} = 0 \qquad \sum_{i=1}^{l} \alpha_i y_i\,. \tag{5}$$

The standard change to a dual problem is to substitute $\mathbf{w}$ from (4) into the primal Lagrangian (3) and this leads to a dual Lagrangian problem below,

$$L_{\mathrm{d}}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{l} \alpha_i y_i b\,, \tag{6}$$

subject to the box constraints (7) where the scalar $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\Phi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\Phi}(\mathbf{x}_j)$. In the standard SVMs formulation, (5) is used to eliminate the last term of (6) that should be solved subject to the following constraints

$$\alpha_i \geq 0, \quad i = 1, \ldots, l \quad \text{and} \tag{7}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0\,. \tag{8}$$

As a result the dual function to be maximized is (9) with box constraints (7) and equality constraint (8).

$$L_{\mathrm{d}}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)\,. \tag{9}$$

An important point to remember is that without the bias term b in the SVMs model, the equality constraint (8) does not exist. This association between bias b and (8) is explored extensively to develop ISDA schemes in the rest of the chapter. Because of the noise, or due to the generic class' features, there will be an overlapping of training data points. Nothing, but constraints, in solving (9) changes and, for the overlapping classes, they are

$$C \geq \alpha_i \geq 0, \quad i = 1, \ldots, l \quad \text{and} \tag{10}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \,, \tag{11}$$

where $0 < C < \infty$, is a penalty parameter trading off the size of a margin with a number of misclassifications. This formulation is often referred to as the soft margin classifier.

In the case of the *nonlinear regression* the learning problem is the maximization of a dual Lagrangian below

$$\begin{aligned} L_{\mathrm{d}}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = &- \varepsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) y_i \\ &- \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{12}$$

$$\text{s.t.} \quad \sum_{i=1}^{l} \alpha_i = \sum_{i=1}^{l} \alpha_i^* \,, \tag{13}$$

$$0 \leq \alpha_i^* \leq C, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l \,. \tag{14}$$

Again, the equality constraint (13) is the result of including bias term in the SVMs model.

2 Iterative Single Data Algorithm for Positive Definite Kernels without Bias Term *b*

In terms of representational capability, when applying Gaussian kernels, SVMs are similar to radial basis function networks. At the end of the learning, they produce a decision function of the following form

$$f(\mathbf{x}) = \sum_{j=1}^{l} v_j K(\mathbf{x}, \mathbf{x}_j) + b \,. \tag{15}$$

However, it is well known that *positive definite kernels* (such as the most popular and the most widely used RBF Gaussian kernels as well as the complete polynomial ones) do not require bias term b [6, 12]. This means that the SVM learning problems should maximize (9) with box constraints (10) in classification and maximize (12) with box constraints (14) in regression. In this section, the KA and the SMO algorithms will be presented for such a fixed (i.e., no-) bias design problem and compared for the classification and regression cases. The equality of two learning schemes and resulting models will be established. Originally, in [18], the SMO *classification* algorithm was developed for solving (9) including the equality constraint (8) related to the

bias b. In these early publications (on the classification tasks only) the case when bias b is fixed variable was also mentioned but the detailed analysis of a fixed bias update was not accomplished. The algorithms here extend and develop a new method to regression problems too.

2.1 Iterative Single Data Algorithm without Bias Term b in Classification

2.1.1 Kernel AdaTron in Classification

The classic AdaTron algorithm as given in [1] is developed for a linear classifier. As mentioned previously, the KA is a variant of the classic AdaTron algorithm in the feature space of SVMs. The KA algorithm solves the maximization of the dual Lagrangian (9) by implementing the gradient ascent algorithm. The update $\Delta\alpha_i$ of the dual variables α_i is given as

$$\Delta\alpha_i = \eta \frac{\partial L_{\mathrm{d}}}{\partial \alpha_i} = \eta \left(1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) = \eta (1 - y_i f_i) \,, \tag{16a}$$

where f_i is the value of the decision function f at the point $\mathbf{x}_i$, i.e., $f_i = \sum_{j=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)$, and y_i denotes the value of the desired target (or the class' label) which is either $+1$ or -1. The update of the dual variables α_i is given as

$$\alpha_i \leftarrow \min(\max(0, \alpha_i + \Delta\alpha_i), C) \quad (i = 1, \ldots, l) \,. \tag{16b}$$

In other words, the dual variables α_i are clipped to zero if $(\alpha_i + \Delta\alpha_i) < 0$. In the case of the soft nonlinear classier ($C < \infty$) α_i are clipped between zero and C, ($0 \leq \alpha_i \leq C$). The algorithm converges from any initial setting for the Lagrange multipliers α_i.

2.1.2 SMO without Bias Term in Classification

Recently [23] derived the update rule for multipliers α_i that includes a detailed analysis of the Karush-Kuhn-Tucker (KKT) conditions for checking the optimality of the solution. (As referred above, a fixed bias update was mentioned only in Platt's papers). The following update rule for α_i for a no-bias SMO algorithm was proposed

$$\Delta\alpha_i = -\frac{y_i E_i}{K(\mathbf{x}_i, \mathbf{x}_i)} = -\frac{y_i f_i - 1}{K(\mathbf{x}_i, \mathbf{x}_i)} = \frac{1 - y_i f_i}{K(\mathbf{x}_i, \mathbf{x}_i)} \,, \tag{17}$$

where $E_i = f_i - y_i$ denotes the difference between the value of the decision function f at the point $\mathbf{x}_i$ and the desired target (label) y_i. Note the equality of (16a) and (17) when the learning rate in (16a) is chosen to be $\eta_i = 1/K(\mathbf{x}_i, \mathbf{x}_i)$. The important part of the SMO algorithm is to check the KKT conditions

with precision τ (e.g., $\tau = 10^{-3}$) in each step. An update is performed only if

$$\begin{aligned} &\alpha_i < C \quad \wedge \quad y_i E_i < -\tau \,, \quad \text{or} \\ &\alpha_i > 0 \quad \wedge \quad y_i E_i > \tau \,. \end{aligned} \tag{17a}$$

After an update, the same clipping operation as in (16b) is performed

$$\alpha_i \leftarrow \min(\max(0, \alpha_i + \Delta\alpha_i), C) \quad (i = 1, \ldots, l) \,. \tag{17b}$$

It is the nonlinear clipping operation in (16b) and (17b) that strictly equals the KA and the SMO without-bias-term algorithm in solving nonlinear classification problems. This fact sheds new light on both algorithms. This equality is not that obvious in the case of a "classic" SMO algorithm with bias term due to the heuristics involved in the selection of active points which should ensure the largest increase of the dual Lagrangian L_d during the iterative optimization steps.

2.2 Iterative Single Data Algorithm without Bias Term *b* in Regression

Similarly to the case of classification, for the models without bias term b, there is a strict equality between the KA and the SMO algorithm when positive definite kernels are used for nonlinear regression.

2.2.1 Kernel AdaTron in Regression

The first extension of the Kernel AdaTron algorithm for regression is presented in [22] as the following gradient ascent update rules for α_i and α_i^*

$$\begin{aligned} \Delta\alpha_i &= \eta_i \frac{\partial L_\mathrm{d}}{\partial \alpha_i} = \eta_i \left(y_i - \varepsilon - \sum_{j=1}^{l} (\alpha_j - \alpha_j^*) K(\mathbf{x}_j, \mathbf{x}_i) \right) = \eta_i (y_i - \varepsilon - f_i) \\ &= -\eta_i (E_i + \varepsilon) \,, \end{aligned} \tag{18a}$$

$$\begin{aligned} \Delta\alpha_i^* &= \eta_i \frac{\partial L_\mathrm{d}}{\partial \alpha_i^*} = \eta_i \left(-y_i - \varepsilon + \sum_{j=1}^{l} (\alpha_j - \alpha_j^*) K(\mathbf{x}_j, \mathbf{x}_i) \right) = \eta_i (-y_i - \varepsilon + f_i) \\ &= \eta_i (E_i - \varepsilon) \,, \end{aligned} \tag{18b}$$

where y_i is the measured value for the input $\mathbf{x}_i$, ε is the prescribed insensitivity zone, and $E_i = f_i - y_i$ stands for the difference between the regression function f at the point $\mathbf{x}_i$ and the desired target value y_i at this point. The calculation of the gradient above does not take into account the geometric reality that no training data can be on both sides of the tube. In other words, it does not use the fact that either α_i or α_i^* or both will be nonzero. i.e., that $\alpha_i \alpha_i^* = 0$ must be fulfilled in each iteration step. Below we derive the gradients of the dual Lagrangian L_d accounting for geometry. This new formulation of the KA algorithm strictly equals the SMO method and it is given as

$$\begin{aligned}\frac{\partial L_{\mathrm{d}}}{\partial \alpha_i} =& - K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i - \sum_{j=1, j\neq i}^{l} (\alpha_j - \alpha_j^*)K(\mathbf{x}_j, \mathbf{x}_i) + y_i \\ & - \varepsilon + K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i^* - K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i^* \\ =& - K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i - (\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}_i) \\ & - \sum_{j=1, j\neq i}^{l} (\alpha_j - \alpha_j^*)K(\mathbf{x}_j, \mathbf{x}_i) + y_i - \varepsilon \\ =& - K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i^* + y_i - \varepsilon - f_i \\ =& - (K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i^* + E_i + \varepsilon) \,.\end{aligned} \tag{19a}$$

For the α_i^* multipliers, the value of the gradient is

$$\frac{\partial L_{\mathrm{d}}}{\partial \alpha_i} = -K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i + E_i - \varepsilon \,. \tag{19b}$$

The update value for α_i is now

$$\Delta\alpha_i = \eta_i \frac{\partial L_{\mathrm{d}}}{\partial \alpha_i} = -\eta_i (K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i^* + E_i + \varepsilon) \,, \tag{20a}$$

$$\alpha_i \leftarrow \alpha_i + \Delta\alpha_i = \alpha_i + \eta_i \frac{\partial L_{\mathrm{d}}}{\partial \alpha_i} = \alpha_i - \eta_i (K(\mathbf{x}_i, \mathbf{x}_i)\alpha_i^* + E_i + \varepsilon) \tag{20b}$$

For the learning rate $\eta_i = 1/K(\mathbf{x}_i, \mathbf{x}_i)$ the gradient ascent learning KA is defined as,

$$\alpha_i \leftarrow \alpha_i - \alpha_i^* - \frac{E_i + \varepsilon}{K(\mathbf{x}_i, \mathbf{x}_i)} \,. \tag{21a}$$

Similarly, the update rule for α_i^* is

$$\alpha_i^* \leftarrow \alpha_i^* - \alpha_i + \frac{E_i - \varepsilon}{K(\mathbf{x}_i, \mathbf{x}_i)} \,. \tag{21b}$$

Same as in the classification, α_i and α_i^* are clipped between zero and C,

$$\alpha_i \leftarrow \min(\max(0, \alpha_i + \Delta\alpha_i), C), \quad (i = 1, \ldots, l) \,, \tag{22a}$$

$$\alpha_i^* \leftarrow \min(\max(0, \alpha_i^* + \Delta\alpha_i^*), C), \quad (i = 1, \ldots, l) \,. \tag{22b}$$

2.2.2 SMO without Bias Term b in Regression

The first algorithm for the SMO without-bias-term in regression (together with a detailed analysis of the KKT conditions for checking the optimality of the solution) is derived in [23]. The following learning rules for the Lagrange multipliers α_i and α_i^* updates were proposed

$$\alpha_i \leftarrow \alpha_i - \alpha_i^* - \frac{E_i + \varepsilon}{K(\mathbf{x}_i, \mathbf{x}_i)} \,, \tag{23a}$$

$$\alpha_i^* \leftarrow \alpha_i^* - \alpha_i + \frac{E_i - \varepsilon}{K(\mathbf{x}_i, \mathbf{x}_i)} \,. \tag{23b}$$

The equality of (21) and (23) is obvious when the learning rate, as presented above in (21), is chosen to be $\eta_i = 1/K(\mathbf{x}_i, \mathbf{x}_i)$. Note that in both the classification and the regression, the optimal learning rate is not necessarily equal for all training data pairs. For a Gaussian kernel, $\eta = 1$ is same for all data points, and for a complete nth order polynomial each data point has different learning rate $\eta_i = 1/K(\mathbf{x}_i, \mathbf{x}_i)$. Similar to classification, a joint update of α_i and α_i^* is performed only if the KKT conditions are violated by at least τ, i.e. if

$$\begin{aligned} &\alpha_i < C \quad \wedge \quad \varepsilon + E_i < -\tau \,, \quad \text{or} \\ &\alpha_i > 0 \quad \wedge \quad \varepsilon + E_i > \tau \,, \quad \text{or} \\ &\alpha_i^* < C \quad \wedge \quad \varepsilon - E_i < -\tau \,, \quad \text{or} \\ &\alpha_i^* > 0 \quad \wedge \quad \varepsilon - E_i > \tau \end{aligned} \tag{24}$$

After the changes, the same clipping operations as defined in (22) are performed

$$\alpha_i \leftarrow \min(\max(0, \alpha_i + \Delta\alpha_i), C) \quad (i = 1, \ldots, l) \,, \tag{25a}$$

$$\alpha_i^* \leftarrow \min(\max(0, \alpha_i^* + \Delta\alpha_i^*), C) \quad (i = 1, \ldots, l) \,. \tag{25b}$$

The KA learning as formulated in this section and the SMO algorithm without bias term for solving regression tasks are strictly equal in terms of both the number of iterations required and the final values of the Lagrange multipliers. The equality is strict despite the fact that the implementation is slightly different. In every iteration step, namely, the KA algorithm updates both weights α_i and α_i^* without any checking whether the KKT conditions are fulfilled or not, while the SMO performs an update according to (24).

2.3 The Coordinate Ascent Based Learning for Nonlinear Classification and Regression Tasks

When positive definite kernels are used, the learning problem for both tasks is same. In a vector-matrix notation, in a dual space, the learning is represented as:

$$\max \quad L_\mathrm{d}(\boldsymbol{\alpha}) = -0.5\boldsymbol{\alpha}^\mathrm{T}\mathbf{K}\boldsymbol{\alpha} + \mathbf{f}^\mathrm{T}\boldsymbol{\alpha} \,, \tag{26}$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad (i = 1, \ldots, n) \,, \tag{27}$$

where, in the classification $n = l$ and the matrix $\mathbf{K}$ is an (l, l) symmetric positive definite matrix, while in regression $n = 2l$ and $\mathbf{K}$ is a $(2l, 2l)$ symmetric semipositive definite one. Note that the constraints (27) define a convex subspace over which the convex dual Lagrangian should be maximized. It is very well known that the vector α may be looked at as the *iterative* solution of a system of linear equations

$$\mathbf{K}\boldsymbol{\alpha} = \mathbf{f} \,, \tag{28}$$

subject to the same constraints given by (27), namely $0 \leq \alpha_i \leq C$, $(i = 1, \ldots, n)$.

Thus, it may seem natural to solve (28), subject to (27), by applying some of the well known and established techniques for solving a general linear system of equations. The size of training data set and the constraints (27) eliminate direct techniques. Hence, one has to resort to the *iterative approaches* in solving the problems above. There are three possible iterative avenues that can be followed. They are; the use of the Non-Negative Least Squares (NNLS) technique [13], application of the Non-Negative Conjugate Gradient (NNCG) method [8] and the implementation of the Gauss-Seidel i.e., the related Successive Over-Relaxation technique. The first two methods solve for the non-negative constraints only, but the bound $\alpha_i \leq C$, for solving "soft" tasks can be readily incorporated into both the NNLS and NNCG. In the case of nonlinear regression, one can apply NNLS and NNCG by taking $C = \infty$ and compensating (i.e. smoothing or "softening" the solution) by increasing the sensitivity zone ε.

Here we show how to extend the application of Gauss-Seidel and Successive Over-Relaxation to both the nonlinear classification and to the nonlinear regression tasks. The Gauss-Seidel method solves (28) by using the ith equation to update the ith unknown doing it iteratively, i.e., starting in the kth step with the first equation to compute the α_1^{k+1}, then the second equation is used to calculate the α_2^{k+1} by using new α_1^{k+1} and α_i^k $(i > 2)$ and so on. The iterative learning takes the following form,

$$\begin{aligned} \alpha_i^{k+1} &= \left(f_i - \sum_{j=1}^{i-1} K_{ij}\alpha_j^{k+1} - \sum_{j=i+1}^{n} K_{ij}\alpha_j^k \right) \Big/ K_{ii} \\ &= \alpha_i^k - \frac{1}{K_{ii}} \left(\sum_{j=1}^{i-1} K_{ij}\alpha_j^{k+1} + \sum_{j=i}^{n} K_{ij}\alpha_j^k - f_i \right) \\ &= \alpha_i^k - \frac{1}{K_{ii}} \frac{\partial L_{\mathrm{d}}}{\partial \alpha_i}\bigg|_{k+1}, \end{aligned} \tag{29}$$

where we use the fact that the term within a second bracket (called the residual r_i in mathematics' references) is the ith element of the gradient of a dual Lagrangian L_{d} given in (26) at the $k+1$th iteration step. The (29) above shows that Gauss-Seidel method is a *coordinate* gradient ascent procedure as well as the KA and the SMO are. *The KA and SMO for positive definite kernels equal the Gauss-Seidel!* Note that the optimal learning rate used in both the KA algorithm and in the SMO without-bias-term approach is exactly equal to the coefficient $1/K_{ii}$ in a Gauss-Seidel method. Based on this equality, the convergence theorem for the KA, SMO and Gauss-Seidel (i.e., successive over-relaxation) in solving (26) subject to constraints (27) can be stated and proved as follows:

Theorem: *For SVMs with positive definite kernels, the iterative learning algorithms KA i.e., SMO i.e., Gauss-Seidel i.e., successive over-relaxation, in solving nonlinear classification and regression tasks* (26) *subject to constraints* (27)*, converge starting from any initial choice of* $\boldsymbol{\alpha}_0$.

Proof: The proof is based on the very well known theorem of convergence of the Gauss-Seidel method for symmetric positive definite matrices in solving (28) without constraints [16]. First note that for positive definite kernels, the matrix $\mathbf{K}$ created by terms $y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ in the second sum in (9), and involved in solving classification problem, is also positive definite. In regression tasks $\mathbf{K}$ is a symmetric positive semidefinite (meaning still convex) matrix, which after a mild regularization given as $(\mathbf{K} \leftarrow \mathbf{K} + \lambda\mathbf{I}, \quad \lambda \sim 1\mathrm{e}-12)$ becomes positive definite one. (Note that the proof in the case of regression does not need regularization at all, but there is no space here to go into these details). Hence, the learning without constraints (27) converges, starting from any initial point α_0, and each point in an n-dimensional search space for multipliers α_i is a viable starting point ensuring a convergence of the algorithm to the maximum of a dual Lagrangian L_d. This, naturally, includes all the (starting) points within, or on a boundary of, any convex subspace of a search space ensuring the convergence of the algorithm to the maximum of a dual Lagrangian L_d over the given subspace. The constraints imposed by (27) preventing variables α_i to be negative or bigger than C, and implemented by the clipping operators above, define such a convex subspace. Thus, each "clipped" multiplier value α_i defines a new starting point of the algorithm guaranteeing the convergence to the maximum of L_d over the subspace defined by (27). For a convex constraining subspace such a constrained maximum is unique.

Due to the lack of the space we do not go into the discussion on the convergence rate here and we leave it to some other occasion. It should be only mentioned that both KA and SMO (i.e. Gauss-Seidel and successive overrelaxation) for positive definite kernels have been successfully applied for many problems (see references given here, as well as many other, benchmarking the mentioned methods on various data sets). Finally, let us just mention that the standard extension of the Gauss-Seidel method is the method of successive over-relaxation that can reduce the number of iterations required by proper choice of a relaxation parameter ω significantly. The successive over-relaxation method uses the following updating rule

$$\alpha_i^{k+1} = \alpha_i^k - \omega \frac{1}{K_{ii}} \left(\sum_{j=1}^{i-1} K_{ij} \alpha_j^{k+1} + \sum_{j=i}^{n} K_{ij} \alpha_j^k - f_i \right) = \alpha_i^k + \omega \frac{1}{K_{ii}} \frac{\partial L_\mathrm{d}}{\partial \alpha_i} \bigg|_{k+1}, \tag{30}$$

and similarly to the KA, SMO, and Gauss-Seidel its convergence is guaranteed.

2.4 Discussions

Both the KA and the SMO algorithms were recently developed and introduced as alternatives to solve quadratic programming problem while training support vector machines on huge data sets. It was shown that when using positive definite kernels the two algorithms are identical in their analytic form and numerical implementation. In addition, for positive definite kernels both algorithms are strictly identical with a classic iterative Gauss-Seidel (optimal coordinate ascent) learning and its extension successive over-relaxation. Until now, these facts were blurred mainly due to different pace in posing the learning problems and due to the "heavy" heuristics involved in the SMO implementation that shadowed an insight into the possible identity of the methods. It is shown that in the so-called no-bias SVMs, both the KA and the SMO procedure are the coordinate ascent based methods and can be classified as ISDA. Hence, they are the inheritors of all good and bad "genes" of a gradient approach and both algorithms have same performance.

In the next section, the ISDAs with explicit bias term b will be presented. The motivations for incorporating bias term into the ISDAs are to improve the versatility and the performance of the algorithms. The ISDAs without bias term developed in this section can only deal with positive definite kernel, which may be a limitation in applications where positive semi-definite kernel such as a linear kernel is more desirable. As it will be discussed shortly, ISDAs with explicit bias term b also seems to be faster in terms of training time.

3 Iterative Single Data Algorithms with an Explicit Bias Term *b*

Before presenting iterative algorithms with bias term b, we discuss some recent presentations of the bias b utilization. As mentioned previously, for positive definite kernels there is no need for bias b. However, one can use it and this means implementing a different kernel. In [19] it was also shown that when using positive definite kernels, one can choose between two types of solutions for both classification and regression. The first one uses the model without bias term (i.e., $f(\mathbf{x}) = \sum_{j=1}^{l} v_j K(\mathbf{x}, \mathbf{x}_j)$), while the second SVM uses an explicit bias term b. For the second one $f(\mathbf{x}) = \sum_{i=1}^{l} v_i K(\mathbf{x}, \mathbf{x}_i) + b$, and it was shown that $f(\mathbf{x})$ is a function resulting from a minimization of the functional shown below

$$I[f] = \sum_{j=1}^{l} V(y_j, f(\mathbf{x}_j)) + \lambda \|f\|_{K^*}^2 \,, \tag{31}$$

where $K^* = K - a$ (for an appropriate constant a) and K is an original kernel function (more details can be found in [19]). This means that by adding a constant term to a positive definite kernel function K, one obtains the solution to the functional $I[f]$ where K is a conditionally positive definite kernel.

Interestingly, similar type of model was also presented in [14]. However, their formulation is done for the classification problems only. They reformulated the optimization problem by adding the $b^2/2$ term to the cost function $\|\mathbf{w}\|^2/2$. This is equivalent to an addition of 1 to each element of the original kernel matrix $\mathbf{K}$. As a result, they changed the original classification dual problem to the optimization of the following one

$$L_{\mathrm{d}}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (K(\mathbf{x}_i, \mathbf{x}_j) + 1) \,. \tag{32}$$

3.1 Iterative Single Data Algorithms for SVMs Classification with Bias Term b

In the previous section, for the SVMs' models when positive definite kernels are used without a bias term b, the learning algorithms for classification and regression (in a dual domain) were solved with box constraints only, originating from minimization of a primal Lagrangian in respect to the weights w_i. However, there remains an open question - how to apply the proposed ISDA scheme for the SVMs that do use explicit bias term b. Such general nonlinear SVMs in classification and regression tasks are given below,

$$f(\mathbf{x}_i) = \sum_{j=1}^{l} y_j \alpha_j \boldsymbol{\Phi}(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\Phi}(\mathbf{x}_j) + b = \sum_{j=1}^{l} v_j K(\mathbf{x}_i, \mathbf{x}_j) + b \,, \tag{33a}$$

$$f(\mathbf{x}_i) = \sum_{j=1}^{l} (\alpha_j^* \alpha_j) \boldsymbol{\Phi}(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\Phi}(\mathbf{x}_j) + b = \sum_{j=1}^{l} v_j K(\mathbf{x}_i, \mathbf{x}_j) + b \,, \tag{33b}$$

where $\boldsymbol{\Phi}(\mathbf{x}_i)$ is the l-dimensional vector that maps n-dimensional input vector $\mathbf{x}_i$ into the feature space. Note that $\boldsymbol{\Phi}(\mathbf{x}_i)$ could be infinite dimensional and we do not have necessarily to know either the $\boldsymbol{\Phi}(\mathbf{x}_i)$ or the weight vector $\mathbf{w}$. (Note also that for a classification model in (33a), we usually take the sign of $f(\mathbf{x})$ but this is of lesser importance now). For the SVMs' models (33), there are also *the equality constraints* originating from minimizing the primal objective function in respect to the bias b as given in (8) for classification and (13) for regression. The motivation for developing the ISDAs for the SVMs with an explicit bias term b originates from the fact that the use of an explicit bias b *seems to lead to the SVMs with less support vectors*. This fact can often be very useful for both the data (information) compression and the speed of learning. Below, we present an iterative learning algorithm for the classification SVMs (33a) with an explicit bias b, subjected to the equality constraints (8). (*The same procedure is developed for the regression SVMs but due to the space constraints we do not go into these details here. However we give some relevant hints for the regression SVMs with bias b shortly*).

There are *three major avenues* (procedures, algorithms) possible in solving the dual problem (6), (7) and (8).

The first one is the standard SVMs algorithm which imposes the equality constraints (8) during the optimization and in this way ensures that the solution never leaves a feasible region. In this case the last term in (6) vanishes. After the dual problem is solved, the bias term is calculated by using *unbounded* Lagrange multipliers α_i [11, 20] as follows

$$b = \frac{1}{\#\ unboundedSVecs} \left(\sum_{i=1}^{\#\ UnboundedSVecs} \left(y_i - \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \right) . \quad (34)$$

Note that in a standard SMO iterative scheme the minimal number of training data points enforcing (8) and ensuring staying in a feasible region is two.

Below, we show *two more possible ways* how the ISDA works for the SVMs containing an explicit bias term b too. In *the first method*, the cost function (1) is augmented with the term $0.5kb^2$ (where $k \geq 0$) and this step changes the primal Lagrangian (3) into the following one

$$L_{\mathrm{p}}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} - \sum_{i=1}^{l} \alpha_i \big\{ y_i \big[\mathbf{w}^{\mathrm{T}} \boldsymbol{\Phi}(\mathbf{x}_i) + b \big] - 1 \big\} + \frac{k}{2} b^2 . \quad (35)$$

Equation (5) also changes as given below

$$\frac{\partial L_{\mathrm{p}}}{\partial b} = 0 \qquad b = \frac{1}{k} \sum_{i=1}^{l} \alpha_i y_i . \quad (36)$$

After forming (35) as well as using (36) and (4), one obtains the dual problem without an explicit bias b,

$$\begin{aligned} L_{\mathrm{d}}(\boldsymbol{\alpha}) = & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ & - \frac{1}{k} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j + \frac{1}{2k} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \\ = & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{k} \right) . \end{aligned} \quad (37)$$

Actually, the optimization of a dual Lagrangian is reformulated for the SVMs with a bias b by applying "tiny" changes of $1/k$ only to the original matrix $\mathbf{K}$ as illustrated in (37). Hence, for the nonlinear classification problems ISDA stands for an *iterative* solving of the following linear system

$$\mathbf{K}_k \boldsymbol{\alpha} = \mathbf{1}_l \quad (38a)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l , \quad (38b)$$

where $K_k(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j (K(\mathbf{x}_i, \mathbf{x}_j) + 1/k)$, $\mathbf{1}_l$ is an l-dimensional vector containing ones and C is a penalty factor equal to infinity for a hard margin classifier. Note that during the updates of α_i, the bias term b must not be used because it is implicitly incorporated within the $\mathbf{K}_k$ matrix. Only after the solution vector α in (38) is found, the bias b should be calculated either by using *unbounded* Lagrange multipliers α_i as given in (34), or by implementing the equality constraints from $\partial L_\mathrm{p}/\partial b = 0$ and given in (36) as

$$b = \frac{1}{k} \sum_{i=1}^{\#\ SVecs} \alpha_i y_i \,. \tag{39}$$

Note, however, that all the Lagrange multipliers, meaning both bounded (clipped to C) and unbounded (smaller than C) must be used in (39). Both equations, (34) and (39), result in the same value for the bias b. Thus, using the SVMs with an explicit bias term b means that, in the ISDA proposed above, the original kernel is changed, i.e., another kernel function is used. This means that the alpha values will be different for each k chosen, and so will be the value for b. The final SVM as given in (33a) is produced by original kernels. Namely, $f(\mathbf{x})$ is obtained by adding the sum of the weighted original kernel values and corresponding bias b. The approach of adding a small change to the kernel function can also be associated with a classic penalty function method in optimization as follows below.

To illustrate the idea of the penalty function, let us consider the problem of maximizing a function $f(x)$ subject to an equality constraint $g(x) = 0$. To solve this problem using classical penalty function method, the following quadratic penalty function is formulated,

$$\max \quad P(x, \rho) = f(x) - \frac{1}{2}\rho \|g(x)\|_2^2 \,, \tag{40}$$

where ρ is the penalty parameter and $\|g(x)\|_2^2$ is the square of the L_2 norm of the function $g(x)$. As the penalty parameter ρ increases towards infinity, the size of the $g(x)$ is pushed towards zero, hence the equality constraint $g(x) = 0$ is fulfilled. Now, let us consider the standard SVMs' dual problem, which is maximizing (9) subject to box constraints (10) and the equality constraint (11). By applying the classical penalty method (40) to the equality constraint (11), we can form the following quadratic penalty function.

$$\begin{aligned} P(\mathbf{x}, \rho) &= L_\mathrm{d}(\alpha) - \frac{1}{2}\rho \left\| \sum_{i=1}^{l} \alpha_i y_i \right\|_2^2 \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{2}\rho \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (K(\mathbf{x}_i, \mathbf{x}_j) + \rho) \,. \end{aligned} \tag{41}$$

The expression above is exactly equal to (37) when ρ equals $1/k$. Thus, the parameter $1/k$ in (37) for the first method of adding bias into the ISDAs can be regarded as a penalty parameter of enforcing equality constraint (11) in the original SVMs dual problem. Also, for a large value of $1/k$, the solution will have a small L_2 norm of (11). In other words, as k approaches zero a bias b converges to the solution of the standard QP method that enforces the equality constraints. However, we do not use the ISDA with small parameter k values here, because the condition number of the matrix $\mathbf{K}_k$ increases as $1/k$ rises. Furthermore, the strict fulfilment of (11) may not be needed in obtaining a good SVM. In the next section, it will be shown that in classifying the MNIST data with Gaussian kernels, the value $k = 10$ proved to be a very good one justifying all the reasons for its introduction (fast learning, small number of support vectors and good generalization).

The *second method* in implementing the ISDA for SVMs with the bias term b is to work with original cost function (1) and keep imposing the equality constraints during the iterations as suggested in [22]. The learning starts with $b = 0$ and after each *epoch* the bias b is updated by applying a secant method as follows

$$b^k = b^{k-1} - \omega^{k-1} \frac{b^{k-1} - b^{k-1}}{\omega^{k-1} - \omega^{k-2}} , \tag{42}$$

where $\omega = \sum_{i=1}^{l} \alpha_i y_i$ represents the value of equality constraint after each epoch. In the case of the regression SVMs, (42) is used by implementing the corresponding regression's equality constraints, namely $\omega = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)$. This is different from [22] where an iterative update after each data pair is proposed. In our SVMs regression experiments such an updating led to an unstable learning. Also, in an addition to changing expression for ω, both the $\mathbf{K}$ matrix, which is now $(2l, 2l)$ matrix, and the right hand side of (38a) which becomes $(2l, 1)$ vector, should be changed too and formed as given in [12].

3.2 Performance of the Iterative Single Data Algorithm and Comparisons

To measure the relative performance of different ISDAs, we ran all the algorithms with RBF Gaussian kernels on a MNIST dataset with 576-dimensional inputs [5], and compared the performance of our ISDAs with LIBSVM V2.4 [2] which is one of the fastest and the most popular SVM solvers at the moment based on the SMO type of an algorithm. The MNIST dataset consists of 60,000 training and 10,000 test data pairs. To make sure that the comparison is based purely on the nature of the algorithm rather than on the differences in implementation, our encoding of the algorithms are the same as LIBSVM's ones in terms of caching strategy (LRU – Least Recent Used), data structure, heuristics for shrinking and stopping criterions. The only significant difference is that instead of two heuristic rules for selecting and updating two data points at each iteration step aiming at the maximal improvement of the dual

objective function, our ISDA selects the worse KKT violator only and updates its α_i at each step.

Also, in order to speed up the LIBSVM's training process, we modified the original LIBSVM routine to perform faster by reducing the numbers of complete KKT checking without any deterioration of accuracy. All the routines were written and compiled in Visual C++ 6.0, and all simulations were run on a 2.4 GHz P4 processor PC with 1.5 Gigabyte of memory under the operating system Windows XP Professional. The shape parameter σ^2 of an RBF Gaussian kernel and the penalty factor C are set to be 0.3 and 10 [5]. The stopping criterion τ and the size of the cache used are 0.01 and 250 Megabytes. The simulation results of different ISDAs against both LIBSVM are presented in Tables 1 and 2, and in a Fig. 1.

The first and the second column of the tables show the performance of the original and modified LIBSVM respectively. The last three columns show the results for single data point learning algorithms with various values of constant $1/k$ added to the kernel matrix in (37). For $k = \infty$, ISDA is equivalent to the SVMs without bias term, and for $k = 1$, it is the same as the classification formulation proposed in [14].

Table 1 illustrates the running time for each algorithm. The ISDA with $k = 10$ was the quickest and required the shortest average time (T_{10}) to complete the training. The average time needed for the original LIBSVM is almost $2T_{10}$ and the average time for a modified version of LIBSVM is 10.3% bigger than T_10. This is contributed mostly to the simplicity of the ISDA. One may think that the improvement achieved is minor, but it is important to consider the fact that approximately more than 50% of the CPU time is spent on the final checking of the KKT conditions in all simulations. During

Table 1. Simulation time for different algorithms

	LIBSVM Original	LIBSVM Modified	Iterative Single Data Algorithm (ISDA)		
			$k = 1$	$k = 10$	$k = \infty$
Class	Time (sec)	Time (sec)	Time (sec)	Time (sec)	Time (sec)
0	1606	885	800	794	1004
1	740	465	490	491	855
2	2377	1311	1398	1181	1296
3	2321	1307	1318	1160	1513
4	1997	1125	1206	1028	1235
5	2311	1289	1295	1143	1328
6	1474	818	808	754	1045
7	2027	1156	2137	1026	1250
8	2591	1499	1631	1321	1764
9	2255	1266	1410	1185	1651
Time, hr	5.5	3.1	3.5	**2.8**	3.6
Time Increase	+95.3%	+10.3%	+23.9%	**0**	+28.3%

Table 2. Number of support vectors for each algorithm

	LIBSVM Original	LIBSVM Modified	Iterative Single Data Algorithm (ISDA) $k = 1$	$k = 10$	$k = \infty$
Class	#SV (BSV)	#SV (BSV)	#SV (BSV)	#SV (BSV)	#SV (BSV)
0	2172 (0)	2172 (0)	2162 (0)	2132 (0)	2682 (0)
1	1440 (4)	1440 (4)	1429 (4)	1453 (4)	2373 (4)
2	3055 (0)	3055 (0)	3047 (0)	3017 (0)	3327 (0)
3	2902 (0)	2902 (0)	2888 (0)	2897 (0)	3723 (0)
4	2641 (0)	2641 (0)	2623 (0)	2601 (0)	3096 (0)
5	2900 (0)	2900 (0)	2884 (0)	2856 (0)	3275 (0)
6	2055 (0)	2055 (0)	2042 (0)	2037 (0)	2761 (0)
7	2651 (4)	2651 (4)	3315 (4)	2609 (4)	3139 (4)
8	3222 (0)	3222 (0)	3267 (0)	3226 (0)	4224 (0)
9	2702 (2)	2702 (2)	2733 (2)	2756 (2)	3914 (2)
Av. # SVs	2574	2574	2639	2558	3151

BSV = Bounded Support Vectors

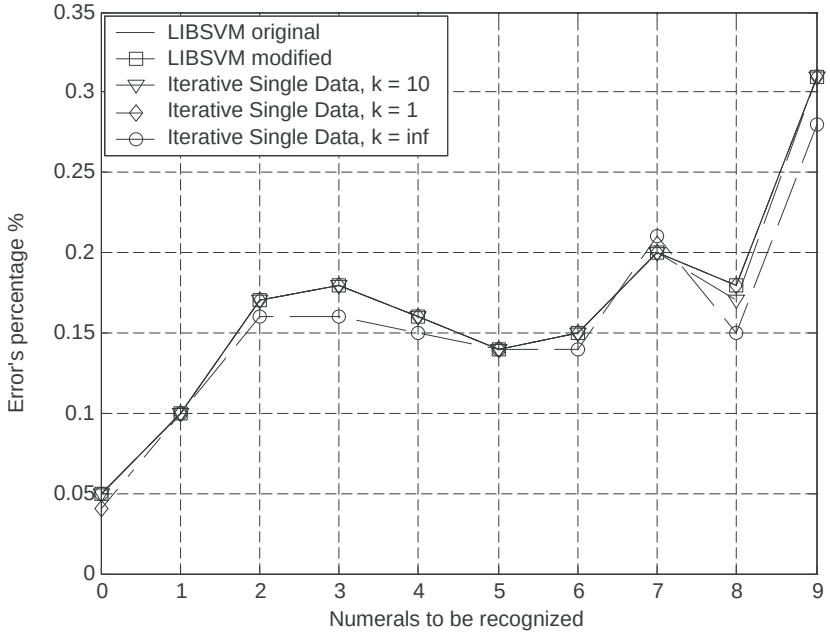

Fig. 1. The percentage of an error on the test data

the checking, the algorithm must calculate the output of the model at each datum in order to evaluate the KKT violations. This process is unavoidable if one wants to ensure the solution's global convergence, i.e. that *all the data* do satisfy the KKT conditions with precision τ indeed. Therefore, the reduction

of time spent on iterations is approximately double the figures shown. Note that the ISDA slows down for $k < 10$ here. This is a consequence of the fact that with a decrease in k there is an increase of the condition number of a matrix $\mathbf{K}_k$, which leads to more iterations in solving (38). At the same time, implementing the no-bias SVMs, i.e., working with $k = \infty$, also slows the learning down due to an increase in the number of support vectors needed when working without bias b.

Table 2 presents the numbers of support vectors selected. For the ISDA, the numbers reduce significantly when the explicit bias term b is included. One can compare the numbers of SVs for the case without the bias b ($k = \infty$) and the ones when an explicit bias b is used (cases with $k = 1$ and $k = 10$). Because identifying less support vectors speeds the overall training definitely up, the SVMs implementations with an explicit bias b are faster than the version without bias.

In terms of a generalization, or a performance on a test data set, all algorithms had very similar results and this demonstrates that the ISDAs produce models that are as good as the standard QP, i.e., SMO based, algorithms (see Fig. 1).

The percentages of the errors on the test data are shown in Fig. 1. Notice the extremely low error percentages on the test data sets for all numerals.

3.3 Discussions

In final part of this chapter, we demonstrate the use, the calculation and the effect of incorporating an explicit bias term b in the SVMs trained with the ISDA. The simulation results show that models generated by ISDAs (either with or without the bias term b) are as good as the standard SMO based algorithms in terms of a generalization performance. Moreover, ISDAs with an appropriate k value are faster than the standard SMO algorithms on large scale classification problems ($k = 10$ worked particularly well in all our simulations using Gaussian RBF kernels). This is due to both the simplicity of ISDAs and the decrease in the number of SVs chosen after an inclusion of an explicit bias b in the model. The simplicity of ISDAs is the consequence of the fact that the equality constraints (8) do not need to be fulfilled during the training stage. In this way, the *second choice heuristics is avoided* during the iterations. Thus, the ISDA is an extremely good tool for solving large scale SVMs problems containing huge training data sets because it is faster than, and it delivers "same" generalization results as, the other standard QP (SMO) based algorithms. The fact that an introduction of an explicit bias b means solving the problem with different kernel suggests that it may be hard to tell in advance for what kind of previously unknown multivariable decision (regression) function the models with bias b may perform better, or may be more suitable, than the ones without it. As it is often the case, the real experimental results, their comparisons and the new theoretical developments should probably be able to tell one day. As for the single data based learning

approach presented here, the future work will focus on the development of even faster training algorithms.

References

1. Anlauf, J. K., Biehl, M., The AdaTron – an adaptive perceptron algorithm. *Europhysics Letters*, 10(7), pp. 687–692, 1989
2. Chang, C., Lin, C., LIBSVM : A library for support vector machines, (available at: http://www.csie.ntu.edu.tw/∼cjlin/libsvm/), 2003
3. Cherkassky, V., Mulier, F., *Learning From Data: Concepts, Theory and Methods*, John Wiley & Sons, New York, NY, 1998
4. Cristianini, N., Shawe-Taylor, J., *An introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, Cambridge, UK, 2000
5. Dong, X., Krzyzak, A., Suen, C. Y., A fast SVM training algorithm, International Journal of Pattern Recognition and Articial Intelligence, Vol. 17, No. 3, pp. 367–384, 2003
6. Evgeniou, T., Pontil, M., Poggio, T., Regularization networks and support vector machines, Advances in Computational Mathematics, 13, pp. 1–50, 2000
7. Frieß, T.-T., Cristianini, N., Campbell, I. C. G., The Kernel-Adatron: A Fast and Simple Learning Procedure for Support Vector Machines. In Shavlik, J., editor, *Proceedings of the 15th International Conference on Machine Learning*, Morgan Kaufmann, pp. 188–196, San Francisco, CA, 1998
8. Hestenes, M., Conjugate Direction Method In Optimization: Application of Mathematics, Vol. 12, Springer-Verlag New York, Heidelberg, 1980
9. Huang, T.-M., Kecman, V., Bias Term b in SVMs Again, Proc. of ESANN 2004, 12^{th} European Symposium on Articial Neural Networks, Bruges, Belgium, (downloadable from http://www.support-vector.ws), 2004
10. Joachims, T. (1999). Making Large-scale SVM learning practical. *Advances in Kernel Methods – Support Vector Learning*. B. Schölkopf, Smola, A. J., and Burges, C. J. C. Cambridge, M.A., MIT Press: 169–184
11. Kecman, V., *Learning and Soft Computing, Support Vector Machines, Neural Networks, and Fuzzy Logic Models*, The MIT Press, Cambridge, MA, (See http://www.support-vector.ws), 2001
12. Kecman, V., Vogt, M., Huang, T.-M., On the Equality of Kernel AdaTron and Sequential Minimal Optimization in Classication and Regression Tasks and Alike Algorithms for Kernel Machines, *Proc. of the 11^{th} European Symposium on Articial Neural Networks*, ESANN 2003, pp. 215–222, Bruges, Belgium, (downloadable from http://www.support-vector.ws), 2003
13. Lawson, C. I., Hanson, R. J., *Solving Least Squares Problems*, Prentice-Hall, Englewood Clis, N.J., 1974
14. Mangasarian, O. L., Musicant, D. R., Successive Overrelaxation for Support Vector Machines, IEEE Trans. Neural Networks, 11(4), 1003–1008, 1999
15. Osuna E, Freund R, Girosi F, An Improved Training Algorithm for Support Vector Machines. In *Neural Networks for Signal Processing VII, Proceedings of the 1997 Signal Processing Society Workshop*, pp. 276–285, 1997
16. Ostrowski, A. M., Solutions of Equations and Systems of Equations, 2^{nd} ed., Academic Press, New York, 1966

17. Platt, J. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, Microsoft Research Technical Report MSR-TR-98-14, 1998
18. Platt, J. C., Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Chap. 12 in Advances in Kernel Methods – Support Vector Learning*, edited by B. Schölkopf, C. Burges, A. Smola, The MIT Press, Cambridge, MA, 1999
19. Poggio, T., Mukherjee, S., Rifkin, R., Rakhlin, A., Verri, A., *b*, CBCL Paper #198/AI Memo #2001-011, Massachusetts Institute of Technology, Cambridge, MA, 2001, also it is a Chapter 11 in "Uncertainty in Geometric Computations", pp. 131–141, Eds., J. Winkler and M. Niranjan, Kluwer Academic Publishers, Boston, MA, 2002
20. Schölkopf, B., Smola, A., *Learning with Kernels – Support Vector Machines, Optimization, and Beyond*, The MIT Press, Cambridge, MA, 2002
21. Vapnik, V. N., The Nature of Statistical Learning Theory, Springer Verlag Inc, New York, NY, 1995
22. Veropoulos, K., *Machine Learning Approaches to Medical Decision Making*, PhD Thesis, The University of Bristol, Bristol, UK, 2001
23. Vogt, M., SMO Algorithms for Support Vector Machines without Bias, Institute Report, Institute of Automatic Control, TU Darmstadt, Darmstadt, Germany, (Available at http://www.iat.tu-darmstadt.de/∼vogt), 2002
24. Vapnik, V. N., 1995. The Nature of Statistical Learning Theory, Springer Verlag Inc, New York, NY
25. Vapnik, V., S. Golowich, A. Smola. 1997. Support vector method for function approximation, regression estimation, and signal processing, In Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA
26. Vapnik, V. N., 1998. Statistical Learning Theory, J. Wiley & Sons, Inc., New York, NY