## REFERENCES

[1] T. Pappas, "An adaptive clustering algorithm for image segmentation," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 901–914, Apr. 1992.

[2] N. K. Pal and S. K. Pal, "A review of image segmentation techniques," *Pattern Recognit.*, vol. 26, pp. 1277–1294, 1993.

[3] K. Chen, D. Wang, and X. Liu, "Weight adaptation and oscillatory correlation for image segmentation," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1106–1123, Sep. 2000.

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press, 1995.

[5] D. Ormoneit and V. Tresp, "Averaging, maximum penalized likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates," *IEEE Trans. Neural Netw.*, vol. 9, no. 4, pp. 639–650, Jul. 1998.

[6] S. Sanjay-Gopal and T. J. Hebert, "Bayesian pixel classification using spatially variant finite mixtures and the generalized EM algorithm," *IEEE Trans. Image Process.*, vol. 7, no. 7, pp. 1014–1028, Jul. 1998.

[7] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain MR images through a hidden Markov random field model and the expectation–maximization algorithm," *IEEE Trans. Med. Imag.*, vol. 20, no. 1, pp. 45–57, Jan. 2001.

[8] P. J. Green, "Bayesian reconstructions from emission tomography data using a modified EM algorithm," *IEEE Trans. Med. Imag*, vol. 9, no. 1, pp. 84–93, Jan. 1990.

[9] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer-Verlag, 1999.


# An Improved Conjugate Gradient Scheme to the Solution of Least Squares SVM

Wei Chu, Chong Jin Ong, and S. Sathiya Keerthi

***Abstract*—The least square support vector machines (LS-SVM) formulation corresponds to the solution of a linear system of equations. Several approaches to its numerical solutions have been proposed in the literature. In this letter, we propose an improved method to the numerical solution of LS-SVM and show that the problem can be solved using one reduced system of linear equations. Compared with the existing algorithm for LS-SVM, the approach used in this letter is about twice as efficient. Numerical results using the proposed method are provided for comparisons with other existing algorithms.**

***Index Terms*—Conjugate gradient (CG), least square support vector machines (LS-SVM), sequential minimal optimization (SMO).**

## I. INTRODUCTION

As an interesting variant of the standard support vector machines (SVMs) [2], least squares support vector machines (LS-SVM) have been proposed by Suykens and Vandewalle [3] for solving pattern recognition and nonlinear function estimation problems. The links between LS-SVM classifiers and kernel Fisher discriminant analysis have also been established by Van Gestel *et al.* [4]. The LS-SVM formulation has been further extended to kernel principal component analysis, recurrent networks and optimal control [5]. As for the training of the LS-SVM, Suykens *et al.* [1] proposed an iterative algorithm based on the conjugate gradient (CG) algorithm. Keerthi and Shevade [6] adapted the sequential minimal optimization (SMO) algorithm for SVM [7] for the solution of LS-SVM.

In this letter, we propose an improved algorithm with CG methods for LS-SVM. We first show the optimality conditions of LS-SVM, and establish its equivalence to a reduced linear system. CG methods can then be employed for its solution. Compared with the algorithm proposed by Suykens *et al.* [1], our algorithm is equally robust and is at least twice as efficient.

We adopt the following notations. $x \in R^d, D \in R^{n\times m}$ are $d$-dimensional column vector and $n \times m$ matrix of real entries, respectively; $x^T$ is the transpose of $x$; $\mathbf{1_n}$ and $\mathbf{0_n}$ are $n$-column vectors of entries 1 and 0, respectively. This letter is organized as follows. In Section II, we review the optimization formulation of LS-SVM, and then show the simplification of the optimality conditions to a reduced linear system. In Section III, we present the results of numerical experiments using our proposed algorithm on some benchmark data sets of different sizes, and compare with the results obtained using the conjugate method by Suykens *et al.* [1] and the SMO algorithm by Keerthi and Shevade [6]. We conclude in Section IV.

## II. LS-SVM AND ITS SOLUTION

Suppose that we are given a training data set of $n$ data points $\{x_i, y_i\}_{i=1}^n$, where $x_i \in R^d$ is the $i$th input vector and $y_i$ is the corresponding $i$th target. For binary classification problems $y_i$ takes only two possible values $\{-1, +1\}$, whereas $y_i \in R$ for regression problems. We employ the idea to transform the input patterns into the reproducing kernel Hilbert space (RKHS) by a set of mapping functions $\boldsymbol{\phi}(x)$ [5]. The reproducing kernel $K(x, x')$ in the RKHS is the dot product of the mapping functions at $x$ and $x'$, i.e.

$$K(x, x') = \langle \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x') \rangle. \tag{1}$$

In the RKHS, a linear classification/regression is performed. The discriminant function takes the form $f(x) = \sum_{i=1}^n \langle \boldsymbol{w} \cdot \boldsymbol{\phi}(x) \rangle + b$, where $\boldsymbol{w}$ is the weight vector in the RKHS, and $b \in R$ is called the bias term. The discriminant function of LS-SVM classifier [3] is constructed by solving the following minimization problem:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} P(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \tag{2}$$

$$\text{s.t. } y_i - (\langle \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i) \rangle + b) = \xi_i \quad i = 1, \ldots, n \tag{3}$$

where $C > 0$ is the regularization factor and $\xi_i$ is the difference between the output $y_i$ and $f(x_i)$. Using standard techniques [8], the Lagrangian for (2)–(3) is

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \frac{C}{2}\sum_{i=1}^n \xi_i^2 + \sum_{i=1}^n \alpha_i \left( y_i - (\langle \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i) \rangle + b) - \xi_i \right) \tag{4}$$

Manuscript received September 21, 2003; revised March 7, 2004. This work was supported in part by the National Institutes of Health (NIH) and its National Institute of General Medical Sciences (NIGMS) Division under Grant 1 P01 GM63208 (NIH/NIGMS Tools and Data Resources in Support of Structural Genomics). The work of W. Chu was supported by a National University of Singapore Research Scholarship.

W. Chu is with Gatsby Computational Neuroscience Unit, University College London, London WC1E 6BT, U.K.

C. J. Ong is with Department of Mechanical Engineering, National University of Singapore, Singapore 119260 (e-mail: mpeongcj@nus.edu.sg).

S. S. Keerthi is with Yahoo! Research Labs, Pasadena, CA 91105 USA.

Digital Object Identifier 10.1109/TNN.2004.841785

TABLE I
COMPUTATIONAL COSTS FOR SMO AND CG ALGORITHMS ($\boldsymbol{\alpha} = 0$ INITIALIZATION) ON SMALL-SIZE AND MEDIUM-SIZE DATA SETS. KERNEL DENOTES THE NUMBER OF KERNEL EVALUATIONS, IN WHICH EACH UNIT DENOTES $10^6$ EVALUATIONS. CPU DENOTES THE CPU TIME IN SECONDS CONSUMED BY THE OPTIMIZATION. $D(\boldsymbol{\alpha})$ DENOTES THE DUAL FUNCTIONAL AT THE OPTIMAL SOLUTION. $\sigma^2$ IS THE PARAMETER IN GAUSSIAN KERNEL, WHICH IS CHOSEN AS IN [9]. $C$ IS THE REGULARIZATION FACTOR IN (2)

| | Banana Dataset, 400 samples with 2-dimensional inputs, $\sigma^2 = 1.8221$, | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Suyken et al.'s CG | | | Our CG Approach | | | SMO | | |
| $\log_{10} C$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ |
| -4 | 0.320 | 0.080 | 0.0198 | 0.239 | 0.070 | 0.0198 | 0.902 | 0.290 | 0.0198 |
| -3 | 0.479 | 0.120 | 0.197 | 0.239 | 0.070 | 0.197 | 0.825 | 0.260 | 0.197 |
| -2 | 0.639 | 0.160 | 1.881 | 0.398 | 0.111 | 1.881 | 0.530 | 0.171 | 1.881 |
| -1 | 1.277 | 0.380 | 15.544 | 0.715 | 0.221 | 15.546 | 0.509 | 0.160 | 15.546 |
| 0 | 2.235 | 0.641 | 97.214 | 1.192 | 0.320 | 97.232 | 0.710 | 0.200 | 97.232 |
| +1 | 3.990 | 1.153 | 665.313 | 1.986 | 0.592 | 665.397 | 3.496 | 1.122 | 665.396 |
| +2 | 7.821 | 2.294 | 5668.911 | 3.733 | 1.013 | 5669.293 | 31.350 | 10.054 | 5669.291 |
| +3 | 15.641 | 4.444 | 52684.905 | 7.067 | 2.104 | 52687.397 | 319.759 | 103.199 | 52687.378 |
| +4 | 32.718 | 9.484 | 494210.847 | 15.563 | 4.616 | 494245.928 | 3306.155 | 1070.269 | 494245.799 |
| | Waveform Dataset, 400 samples with 21-dimensional inputs, $\sigma^2 = 24.5325$ | | | | | | | | |
| | Suyken et al.'s CG | | | Our CG Approach | | | SMO | | |
| $\log_{10} C$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ |
| -4 | 0.320 | 0.150 | 0.0176 | 0.239 | 0.110 | 0.0176 | 0.929 | 0.450 | 0.0176 |
| -3 | 0.479 | 0.210 | 0.173 | 0.239 | 0.090 | 0.173 | 0.910 | 0.441 | 0.173 |
| -2 | 0.639 | 0.331 | 1.477 | 0.318 | 0.140 | 1.477 | 0.517 | 0.251 | 1.477 |
| -1 | 1.118 | 0.501 | 9.398 | 0.636 | 0.291 | 9.398 | 0.413 | 0.200 | 9.398 |
| 0 | 2.394 | 1.112 | 55.415 | 1.192 | 0.560 | 55.415 | 0.557 | 0.250 | 55.415 |
| +1 | 6.225 | 3.025 | 304.430 | 2.939 | 1.485 | 304.431 | 2.355 | 1.141 | 304.430 |
| +2 | 14.684 | 6.541 | 972.925 | 7.147 | 3.183 | 972.925 | 10.600 | 5.138 | 972.925 |
| +3 | 23.462 | 10.447 | 1428.193 | 11.514 | 5.327 | 1428.192 | 21.774 | 10.575 | 1428.193 |
| +4 | 27.611 | 12.316 | 1510.109 | 13.340 | 6.101 | 1510.110 | 28.214 | 14.040 | 1510.110 |
| | Image Dataset, 1300 samples with 18-dimensional inputs, $\sigma^2 = 2.7183$ | | | | | | | | |
| | Suyken et al.'s CG | | | Our CG Approach | | | SMO | | |
| $\log_{10} C$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ |
| -4 | 3.379 | 1.983 | 0.0635 | 2.532 | 1.642 | 0.0635 | 9.301 | 6.830 | 0.0635 |
| -3 | 5.067 | 2.654 | 0.618 | 2.532 | 1.572 | 0.618 | 7.444 | 5.367 | 0.618 |
| -2 | 8.445 | 4.897 | 5.050 | 4.218 | 2.364 | 5.050 | 5.166 | 3.776 | 5.050 |
| -1 | 20.266 | 11.466 | 28.671 | 10.962 | 6.350 | 28.671 | 4.833 | 3.505 | 28.671 |
| 0 | 48.974 | 28.452 | 133.878 | 26.980 | 16.873 | 133.878 | 7.036 | 4.997 | 133.878 |
| +1 | 135.097 | 78.922 | 574.150 | 70.819 | 39.212 | 574.150 | 33.935 | 24.225 | 574.150 |
| +2 | 417.110 | 243.238 | 2554.951 | 216.667 | 137.470 | 2554.951 | 253.361 | 187.459 | 2554.950 |
| +3 | 1269.904 | 740.442 | 11554.667 | 705.636 | 450.154 | 11554.666 | 1910.307 | 2802.560 | 11554.662 |
| +4 | 4186.289 | 2446.655 | 39458.945 | 2256.850 | 1436.888 | 39458.946 | 11806.379 | 17331.135 | 39458.943 |
| | Splice Dataset, 1000 samples with 60-dimensional inputs, $\sigma^2 = 29.9641$ | | | | | | | | |
| | Suyken et al.'s CG | | | Our CG Approach | | | SMO | | |
| $\log_{10} C$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ |
| -4 | 1.000 | 2.113 | 0.0499 | 0.999 | 2.143 | 0.0499 | 4.726 | 8.262 | 0.0499 |
| -3 | 1.999 | 4.216 | 0.497 | 1.498 | 3.215 | 0.497 | 4.364 | 7.551 | 0.497 |
| -2 | 2.998 | 6.319 | 4.775 | 1.996 | 4.325 | 4.775 | 2.925 | 5.088 | 4.775 |
| -1 | 5.995 | 12.628 | 36.459 | 3.492 | 7.531 | 36.459 | 3.120 | 5.408 | 36.459 |
| 0 | 12.988 | 27.350 | 159.990 | 6.483 | 14.001 | 159.990 | 3.120 | 5.348 | 159.990 |
| +1 | 29.971 | 63.261 | 309.340 | 16.951 | 36.816 | 309.340 | 5.391 | 9.464 | 309.340 |
| +2 | 65.935 | 138.911 | 348.659 | 35.396 | 77.757 | 348.659 | 10.767 | 18.697 | 348.659 |
| +3 | 89.911 | 189.836 | 353.380 | 55.834 | 120.547 | 353.380 | 32.722 | 56.892 | 353.380 |
| +4 | 111.889 | 232.535 | 353.866 | 62.813 | 134.298 | 353.865 | 119.222 | 207.278 | 353.866 |

where $\alpha_i, i = 1, \ldots, n$ are the Lagrangian multipliers corresponding to (3). The Karush–Kuhn–Tucker (KKT) conditions (2) are

$$\begin{cases} \frac{\partial L}{\partial \boldsymbol{w}} = 0 & \rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(x_i) \\ \frac{\partial L}{\partial b} = 0 & \rightarrow \quad \sum_{i=1}^{n} \alpha_i = 0 \\ \frac{\partial L}{\partial \xi_i} = 0 & \rightarrow \quad \alpha_i = C\xi_i \qquad \forall i \\ \frac{\partial L}{\partial \alpha_i} = 0 & \rightarrow \quad \xi_i = y_i - (\langle \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i) \rangle + b) \qquad \forall i \end{cases} . \tag{5}$$

In the numerical solution proposed by Suykens *et al.* [1], the KKT conditions of (5) are reduced to a linear system by eliminating $\boldsymbol{w}$ and $\boldsymbol{\xi}$, resulting in

$$\begin{bmatrix} \boldsymbol{Q} & \mathbf{1_n} \\ \mathbf{1}_{\mathbf{n}}^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix} \tag{6}$$

where $\boldsymbol{Q} \in R^{n \times n}$ with $ij$th entry $\boldsymbol{Q}_{ij} = K(x_i, x_j) + (1/C)\delta_{ij}$[1], $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$ and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$. Note that $\boldsymbol{Q}$ is symmetric and positive–definite since the matrix $\boldsymbol{K} \in R^{n \times n}$ with $\boldsymbol{K}_{ij} = K(x_i, x_j)$ is semipositive–definite and the diagonal term $1/C$ is positive. Solving (6) for $\boldsymbol{\alpha}$ and $b$, the discriminant function can be obtained from $f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$.

Suykens *et al.* [1] suggested the use of the CG method for the solution of (6). In addition, they reformulated (6) so as to exploit the positive–definiteness of $\boldsymbol{Q}$ and proposed to solve two systems of linear equations for $\boldsymbol{\alpha}$. More exactly, their algorithm can be described as

[1] $\delta_{ij}$ is 1 only when $i = j$, otherwise 0.

1) Solve the intermediate variables $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ from $\boldsymbol{Q}\cdot\boldsymbol{\eta} = \boldsymbol{y}$ and $\boldsymbol{Q}\cdot\boldsymbol{\nu} = \mathbf{1_n}$ using CG methods.
2) Find solution $b = (\mathbf{1}_\mathbf{n}^T\cdot\boldsymbol{\eta})/(\mathbf{1}_\mathbf{n}^T\cdot\boldsymbol{\nu})$, and $\boldsymbol{\alpha} = \boldsymbol{\eta} - b\cdot\boldsymbol{\nu}$.

In step 1, the $n$th-order linear equations are solved twice, using CG method, for the solutions of $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$. In the following, we propose a single step approach that solves the linear system having $n-1$ order. We begin by stating some known results.

*Lemma 1:* Consider the partition of the symmetric and positive–definite matrix $\boldsymbol{Q} := \begin{bmatrix} \bar{\boldsymbol{Q}} & \boldsymbol{q} \\ \boldsymbol{q}^T & \boldsymbol{Q}_{nn} \end{bmatrix}$, where $\bar{\boldsymbol{Q}} \in R^{(n-1)\times(n-1)}$, $\boldsymbol{q} \in R^{n-1}$ and $\boldsymbol{Q}_{nn} \in R$. Then

$$\tilde{\boldsymbol{Q}} := \bar{\boldsymbol{Q}} - \mathbf{1}_{n-1}\cdot\boldsymbol{q}^T - \boldsymbol{q}\cdot\mathbf{1}_{n-1}^T + \boldsymbol{Q}_{nn}\cdot\mathbf{1}_{n-1}\cdot\mathbf{1}_{n-1}^T \tag{7}$$

is positive–definite.

*Proof:* Let $\boldsymbol{M} = \begin{bmatrix} \mathbf{I}_{n-1} & \mathbf{0}_{n-1} \\ -\mathbf{1}_{n-1}^T & 1 \end{bmatrix}$ and note that

$$\begin{aligned}\boldsymbol{M}^T\cdot\boldsymbol{Q}\cdot\boldsymbol{M} &= \begin{bmatrix} \mathbf{I}_{n-1} & -\mathbf{1}_{n-1} \\ \mathbf{0}_{n-1}^T & 1 \end{bmatrix}\begin{bmatrix} \bar{\boldsymbol{Q}} & \boldsymbol{q} \\ \boldsymbol{q}^T & \boldsymbol{Q}_{nn} \end{bmatrix}\begin{bmatrix} \mathbf{I}_{n-1} & \mathbf{0}_{n-1} \\ -\mathbf{1}_{n-1}^T & 1 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\boldsymbol{Q}} & \tilde{\boldsymbol{q}} \\ \tilde{\boldsymbol{q}}^T & \boldsymbol{Q}_{nn} \end{bmatrix}\end{aligned} \tag{8}$$

where $\tilde{\boldsymbol{Q}}$ is as given by (7). Since $\boldsymbol{Q}$ is positive–definite, so is the matrix at the right-hand side of (8). As $\tilde{\boldsymbol{Q}}$ is a submatrix of a positive–definite matrix, the result follows.

*Lemma 2:* Let $\tilde{\boldsymbol{\alpha}}^*$ be the solution of $\tilde{\boldsymbol{Q}}\cdot\tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{y}} - y_n\cdot\mathbf{1}_{n-1}$ with $\tilde{\boldsymbol{y}} = [y_1, y_2, \ldots, y_{n-1}]^T$ and $\tilde{\boldsymbol{Q}}$ as given by (7). Then the vector $\boldsymbol{\alpha}^* = \begin{bmatrix} \tilde{\boldsymbol{\alpha}}^* \\ -\mathbf{1}_{n-1}^T\cdot\tilde{\boldsymbol{\alpha}}^* \end{bmatrix}$ and $b^* = y_n + \boldsymbol{Q}_{nn}\cdot(\mathbf{1}_{n-1}^T\cdot\tilde{\boldsymbol{\alpha}}^*) - \boldsymbol{q}^T\cdot\tilde{\boldsymbol{\alpha}}^*$ are the solution of the optimization problem (2)–(3).

*Proof:* Since $\tilde{\boldsymbol{Q}}$ is positive–definite, $\tilde{\boldsymbol{\alpha}}^*$ is unique. Using $\tilde{\boldsymbol{Q}}$ from (7) and $\tilde{\boldsymbol{Q}}\cdot\tilde{\boldsymbol{\alpha}}^* = \tilde{\boldsymbol{y}} - y_n\cdot\mathbf{1}_{n-1}$, we have

$$\begin{aligned}&\bar{\boldsymbol{Q}}\cdot\tilde{\boldsymbol{\alpha}}^* - \boldsymbol{q}\cdot\mathbf{1}_{n-1}^T\cdot\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{y}} \\ &\quad= (\boldsymbol{q}^T\cdot\tilde{\boldsymbol{\alpha}}^* - \boldsymbol{Q}_{nn}\cdot(\mathbf{1}_{n-1}^T\cdot\tilde{\boldsymbol{\alpha}}^*) - y_n)\cdot\mathbf{1}_{n-1} \\ &\quad= -b^*\cdot\mathbf{1}_{n-1}\end{aligned} \tag{9}$$

where we have used

$$b^* = y_n + \boldsymbol{Q}_{nn}\cdot(\mathbf{1}_{n-1}^T\cdot\tilde{\boldsymbol{\alpha}}^*) - \boldsymbol{q}^T\cdot\tilde{\boldsymbol{\alpha}}^*. \tag{10}$$

Rewriting (9) and (10) into matrix form, we have

$$\boldsymbol{Q}\cdot\boldsymbol{\alpha}^* + b^*\cdot\mathbf{1_n} = \boldsymbol{y}. \tag{11}$$

From (11) and the fact that $\mathbf{1}_n^T\cdot\boldsymbol{\alpha}^* = 0$, it follows that $\boldsymbol{\alpha}^*$ and $b^*$ are the solution of (6) and, hence, satisfy the optimization problem (2)–(3).

Following Lemma 2, we can use the standard CG algorithm [8] for the solution of the reduced linear system $\tilde{\boldsymbol{Q}}\cdot\tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{y}} - y_n\cdot\mathbf{1}_{n-1}$. Clearly, compared with the scheme proposed by Suykens *et al.* [1], our algorithm can save at least 50% of the computational effort. In addition, $\tilde{\boldsymbol{Q}}$ is positive–definite and the numerical stability of our approach is the similar to that proposed by Suykens *et al.* [1].

## III. Numerical Experiments

For comparison purpose, we implemented our proposed algorithm with standard CG methods, the algorithm proposed by Suykens *et al.* [1], and the SMO algorithm given by Keerthi and Shevade [6]. The stopping conditions used in all three algorithms are the same, and is based on the value of the duality gap, i.e., $P(\boldsymbol{w}, b, \boldsymbol{\xi}) - D(\boldsymbol{\alpha}) \le \epsilon D(\boldsymbol{\alpha})$, where $P(\boldsymbol{w}, b, \boldsymbol{\xi})$ is defined as in (2), $D(\boldsymbol{\alpha})$ is the dual functional given by $D(\boldsymbol{\alpha}) = (1/2)\cdot\boldsymbol{\alpha}^T\cdot\boldsymbol{Q}\cdot\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\cdot\boldsymbol{y}$, and $\epsilon = 10^{-6}$. Note that this is not the traditional stopping condition for CG algorithm. We have discounted the extra cost caused by computing the stopping condition in CG for a fair comparison. In the implementations, the diagonal entries of $\boldsymbol{Q}$ were cached for efficiency, and we also cached the vector $\boldsymbol{q}$ for our improved CG scheme. The programs used in the experiments were written in ANSI C and executed on a Pentium III 866 PC running on Windows 2000 platform.[2] Six benchmark data sets were used in these experiments: Banana, Waveform, Image, Splice, MNIST, and Computer Activity.[3] The Gaussian kernel $K(x, x') = \exp\left(-(\|x - x'\|^2/2\sigma^2)\right)$ was used as the kernel function. The values of $\sigma^2$ used are based on the suggested values given in Duan *et al.* [9].

TABLE II
Computational Costs for SMO and CG Algorithms ($\boldsymbol{\alpha} = 0$ Initialization) on Large-Size Data Sets. Computer Activity is a Regression Problem. Kernel Denotes the Number of Kernel Evaluations, in Which Each Unit Denotes $10^6$ Evaluations. CPU Denotes the CPU Time in Seconds Consumed by the Optimization. $D(\boldsymbol{\alpha})$ Denotes the Dual Functional at the Optimal Solution. $\sigma^2$ is the Parameter in Gaussian Kernel, Which is Set to an Appropriate Value. $C$ is the Regularization Factor in (2)

| | MNIST Dataset, 11739 samples with 400-dimensional inputs, $\sigma^2 = 0.0025$ | | | | | |
|---|---|---|---|---|---|---|
| | Our CG Approach | | | SMO | | |
| $\log_{10} C$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ |
| -2 | 413.302 | 5611.010 | 56.136 | 401.397 | 3515.685 | 56.136 |
| -1 | 757.752 | 10284.239 | 493.689 | 403.956 | 3540.721 | 493.689 |
| 0 | 1722.135 | 23495.622 | 2685.667 | 420.814 | 3688.304 | 2685.668 |
| +1 | 4064.206 | 56682.010 | 4965.833 | 669.879 | 5872.334 | 4965.836 |
| +2 | 9643.847 | 134222.101 | 5558.749 | 1257.794 | 11027.597 | 5558.752 |
| | Computer Activity, 8192 samples with 21-dimensional inputs, $\sigma^2 = 20$ | | | | | |
| | Our CG Approach | | | SMO | | |
| $\log_{10} C$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ | Kernel | CPU | $D(\boldsymbol{\alpha})$ |
| -2 | 335.438 | 423.450 | 19.510 | 158.590 | 149.785 | 19.510 |
| -1 | 805.028 | 1021.007 | 80.608 | 159.148 | 150.226 | 80.608 |
| 0 | 1710.666 | 2185.105 | 275.971 | 220.706 | 207.939 | 275.971 |
| +1 | 4662.375 | 5880.373 | 1002.054 | 845.302 | 798.177 | 1002.054 |
| +2 | 14221.886 | 17926.644 | 5453.505 | 6382.203 | 6028.509 | 5453.501 |

We carried out the numerical experiments on the six data sets with several different regularization factor $C$, and recorded their results in Tables I and II, respectively. All the algorithms are stable and closely reach the same dual functional $D(\boldsymbol{\alpha})$. The computational cost of our approach is about half of that used by the algorithm in [1]. The increase in computational cost of the SMO algorithm at large $C$ values (greater than $10^3$) is sharp as seen from the results on Banana and Image data sets.[4] For small to medium data sets, the CG algorithm is more efficient than SMO. Experimentally, SMO scales better than the CG methods based on the two large data sets that we have solved. Consequently, there is no clear overall superiority in the performance for either of the methods. We suggest that CG algorithm is suitable for small to moderate data sets, i.e., the number of samples is less than two thousands, while SMO is suitable for large data sets.

[2]The programs and their source code can be accessed at http://guppy.mpe.nus.edu.sg/~chuwei/code/lssvm.zip.

[3]Image and Splice datasets can be accessed at http://ida.first.gmd.de/~raetsch/data/benchmarks.htm. We used the first partition in the twenty partitions. MNIST is available at http://yann.lecun.com/exdb/mnist/, and we selected the samples of the digit 0 and 8 only to set up the binary classification problem. Computer Activity dataset is available in DELVE at http://www.cs.toronto.edu/~delve/, and it corresponds to a regression problem.

[4]Keerthi and Shevade [6] argued that too large $C$ values might actually be out of our interest since the optimal $C$ is seldom greater than $10^3$ in practice.

## IV. Conclusion

In this letter, we proposed a new scheme for the numerical solution of LS-SVM using CG methods. The new scheme is simple and efficient and involves the solution of the linear system of equations of $n - 1$ order. Numerical results provided shows that the proposed scheme is at least twice as efficient when compared with the algorithm proposed by Suykens *et al.* [1]. It also has a comparable performance when compared with the SMO approach by Keerthi and Shevade [6].

## References


[1] J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle, "Least squares support vector machine classifiers: A large scale algorithm," in *Proc. Eur. Conf. Circuit Theory Design (ECCTD'99)*, pp. 839–842.

[2] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[3] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[4] T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle, "A Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel fisher discriminant analysis," *Neural Computat.*, vol. 14, pp. 1115–1147, 2002.

[5] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vanthienen, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[6] S. S. Keerthi and S. K. Shevade, "SMO algorithm for least squares SVM formulations," *Neural Computat.n*, vol. 15, no. 2, Feb. 2003.

[7] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[8] R. Fletcher, *Practical Methods of Optimization*. New York: Wiley, 1987.

[9] K. Duan, S. S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," *Neurocomputing*, vol. 51, pp. 41–59, 2003.


# Solving Systems of Linear Equations Via Gradient Systems With Discontinuous Righthand Sides: Application to LS-SVM

Leonardo V. Ferreira, Eugenius Kaszkurewicz, and Amit Bhaya

***Abstract*—A gradient system with discontinuous righthand side that solves an underdetermined system of linear equations in the $L_1$ norm is presented. An upper bound estimate for finite time convergence to a solution set of the system of linear equations is shown by means of the Persidskii form of the gradient system and the corresponding nonsmooth diagonal type Lyapunov function. This class of systems can be interpreted as a recurrent neural network and an application devoted to solving least squares support vector machines (LS-SVM) is used as an example.**

***Index Terms*—Diagonal type functions, gradient systems, least absolute deviation, neural networks, nonsmooth systems, Persidskii systems, support vector machines (SVMs), systems of linear equations.**

## I. Introduction

This letter proposes the use of a gradient dynamical system to solve underdetermined systems of linear equations in the $L_1$ norm. The system of linear equations is associated to an unconstrained convex optimization problem, which has the same solution set as the linear system. The unconstrained optimization problem, in turn is mapped into a gradient system with a discontinuous righthand side, which can be considered as a neural network with discontinuous activation functions [1]. The advantage of using this class of gradient systems is that convergence to the solution set of the system of linear equations occurs in finite time, and an upper bound for the latter is easily obtained. In addition, hardware implementation of this class of systems is simple.

Gradient systems were used to solve optimization problems for the first time in [2], where a method for solving linear programming problems on an analog computer is presented. Since then, this approach has been widely used [3]–[7].

Convergence analysis is performed by means of a Persidskii form of the gradient system in conjunction with a diagonal type Lyapunov function [8], [9]. The approach used in this letter has been already used in [10] and [11], where Persidskii systems, together with the associated diagonal type Lyapunov functions were used to derive convergence conditions of discontinuous gradient systems that solve linear programming problems and in [12], where a class of Persidskii systems with discontinuous righthand sides is analyzed.

The proposed gradient dynamical system can be solved using standard ordinary differential equation (ODE) software and this could be an advantage, when the number of unknowns is large. Furthermore, implementations of these standard ODE methods on parallel computers could be used, in order to make it possible to deal with large datasets. In addition, gradient dynamical systems can be implemented as an analog circuit using only resistors, amplifiers and switches, which is appropriate for real time processing using VLSI technology [1].

Manuscript received March 19, 2004; revised November 9, 2004. This work was supported by Conselho Nacional de Desenvolvimento Cientifico e Technologico (CNPq) under Projects 140811/2002-8, 551863/2002-1, and 471262/03-0.

The authors are with the Department of Electrical Engineering, NACAD-COPPE/Federal University of Rio de Janeiro, 21945-970 Rio de Janeiro, RJ Brazil (e-mail: lvalente@coep.ufrj.br; eugenius@nacad.ufrj.br; amit@nacad.ufrj.b).

Digital Object Identifier 10.1109/TNN.2005.844091