A Fast Training Algorithm For Least-Squares Support Vector Machines

Xiao-Lei Xia *, **Kang Li** * **and Min-Rui Fei** **

* *School of Electronics, Electrical Engineering and Computer Science Queen's University Belfast, Belfast BT9 5AH, UK (Tel: 44-28-90974164; e-mail: {xxia01,k.li}@qub.ac.uk)*
** *School of Mechatronics and Automation, Shanghai University, Shanghai 200072, China (e-mail: mrfei@staff.shu.edu.cn)*

Abstract: The paper addresses the issue of training acceleration for binary Least Squares Support Vector Machines (LS-SVMs). An LS-SVM is trained by solving a linear system, for which the conjugate-gradient (CG) method is applied however in a very complicated way, therefore slows down the training process. To overcome the drawback, this paper first introduces a variant formulation of LS-SVMs to achieve explicit application of the CG method. Then, an alternative to the CG method - namely Forward Linear Regression (FLR) is proposed to further speed up the training process. Different from the CG method, the FLR, derived from a two-stage algorithm for fast model selection in nonlinear system identification, is a non-iterative algorithm and is easy to implement. Experimental results on the two-spiral dataset confirm the efficacy of the proposed techniques.

1. INTRODUCTION

The last decade has seen the growing popularity of the Support Vector Machines (SVM) in machine learning, see Vapnik (1995), Cristianini and Shawe-Taylor (2000). It is founded on concrete mathematical background and has demonstrated outstanding performance on various applications, including pattern recognition and function estimation. The learning of an SVM can be formulated as a quadratic optimization problem with linear inequality constraints. Hence it can be very costly, in terms of both the resource requirement and the computational complexity, to train an SVM.

Recently, a least squares formulation of the standard SVM - Least Squares SVM (LS-SVM) (Suykens and Vandewalle (1999)), has been proposed to avoid the quadratic programming (QP) problem. An LS-SVM considers only equality constraints rather than the inequality ones. As a result, an LS-SVM is trained by solving a linear system. Empirical studies suggest that LS-SVMs have very competitive generalization abilities to standard SVMs, see Gestel et al. (2004).

The well-known conjugate-gradient (CG) (Golub and Loan (1989)) algorithm can be employed to solve the linear system associated with an LS-SVM. To apply the CG method requires that the associated coefficient matrix is positive definite. Unfortunately, the linear system associated to an LS-SVM does not necessarily satisfy the condition, which makes the direct use of the CG method inapplicable. To tackle the problem, the conventional approach is to convert the original linear regression problem to a dual systems that share the same positive definite coefficient matrix. The drawback is that the size of the linear system is almost doubled, hence leading to an increase in the computational complexity. Moreover, the CG method uses iterative searching process which can be exhaustive.

To speed up the training for LS-SVMs, a sequential minimization optimization (SMO) algorithm was proposed (see Joachims (1998)) and their experimental results showed that the SMO is faster than the CG. Despite the fact that the SMO brings back the quadratic programming technique for standard SVMs into the LS-SVM training, there is however no analytic evidence on its superiority in terms of the computational complexity.

This paper introduces a variant form of the LS-SVMs, aiming to reduce the computational complexity, thus ensuring the positive definite of the coefficient matrix associated to the variant linear regression problem. This allows direct use of the CG method without doubling the size of the linear problem, thus leading to a significant improvement in the training speed by roughly 50% compared with the original LS-SVM training. Next this paper introduces an alternative to the CG method - Forward Linear Regression (FLR) to solve the variant linear system. The FLR is derived from a two-stage algorithm (see Li et al. (2005), Li et al. (2006)) for fast method model selection in nonlinear system identification. This algorithm has been successfully applied in Xia et al. (2008) to select optimal number of support vectors, as well as in Xia and Li (2008) to train multi-class support vector machines. The original algorithm has two major parts: the forward model selection and the back model refinement. The first part of the algorithm in Li et al. (2005) and Li et al. (2006) is modified and evolved to the proposed FLR algorithm. Like

⋆ K Li would like to acknowledge the support from the U.K. EPSRC under Grant EP/E055877/1. M Fei would like to acknowledge the support from the Shanghai Leading Academic Disciplines (T0103), Shanghai Key Laboratory of Power Station Automation Technology, and the International Exchange Scheme from Queen's University Belfast.

the CG method, FLR can also be used to solve the linear system associated to the variant formulation of LS-SVM and there is no need to change the system size. Since FLR is a non-iterative process, learning of LS-SVM can be further speeded up. Experiments on the two-spiral dataset show that the variant LS-SVM, implemented by both CG method and FLR algorithm, are superior to the original one in terms of computation time.

The paper is organized as follows. Section 2 gives a brief review of the LS-SVMs. The training of an LS-SVM using the CG method is discussed in Section 3. Section 4 introduces the variant formulation of the LS-SVM and discusses the direct application of CG to the variant problem. Section 5 gives the derivation of the fast FLR algorithm. Experimental results are given in section 6. Section 7 concludes the paper.

2. LEAST SQUARES SUPPORT VECTOR MACHINES

For the classification of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x} \in R^N$ and $y \in \{-1, 1\}$, a Least Squares SVM (LS-SVM) seeks the optimal separating hyperplane which are parameterized by the solution of the following optimization problem:

$$\min_{\mathbf{w},b} J_0(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{1}{2}\gamma \sum_{i=1}^{n} e_i^2 \tag{1}$$

$$\text{s.t.} \quad \mathbf{w}^\top \phi(x_i) + b = y_i - e_i \quad i = 1, \ldots, n$$

where γ is a penalty factor for the tradeoff between training errors and generalization ability. ϕ is a function which maps training data into a high dimensional feature space where they become separable.

Introducing Lagrange multipliers $\alpha_i (i = 1, \ldots, n)$ for each of the equality constraints to give the following Lagrangian:

$$L(\mathbf{w}, b, \mathbf{e}, \alpha) = J_0 - \sum_{i=1}^{n} \alpha_i [\mathbf{w}^\top \phi(\mathbf{x}_i) + b - y_i + e_i] \tag{2}$$

Due to the equality constraints, α_i can either be positive or negative according to the Karush-Kuhn-Tucker (KKT) conditions (see Fletcher (1987)). The optimality of (2) requires:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \tag{3}$$

$$\frac{\partial L}{\partial b} = 0 \longrightarrow \sum_{i=1}^{n} \alpha_i = 0 \tag{4}$$

$$\frac{\partial L}{\partial e_i} = 0 \longrightarrow \alpha_i = \gamma e_i, \quad i = 1, \ldots, n \tag{5}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \longrightarrow \mathbf{w}^\top \phi(\mathbf{x}_i) + b = y_i - e_i \tag{6}$$

These condition leads to a linear system which is formulated as:

$$\begin{bmatrix} \mathbf{E} & 0 & 0 & -\Phi \\ 0 & 0 & 0 & \mathbf{I}^\top \\ 0 & 0 & -1 & \mathbf{I}^\top \\ \Phi^\top & \mathbf{I} & \mathbf{E} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \\ \mathbf{e} \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{y} \end{bmatrix} \tag{7}$$

where $\Phi = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$, $\mathbf{y} = [y_1, \ldots, y_n]^\top$, $\mathbf{e} = [e_1, \ldots, e_n]^\top$, $\alpha = [\alpha_1, \ldots, \alpha_n]^\top$, $\mathbf{I}$ is a n-length vector of ones and $\mathbf{E}$ is an unity matrix of arbitrary rank.

The linear equations can be further simplified as:

$$\left[\begin{array}{c|c} \mathbf{H} + \gamma^{-1}\mathbf{E} & \mathbf{I} \\ \hline \mathbf{I}^\top & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right] \tag{8}$$

where $H = \Phi^\top \Phi$.

The kernel technique is introduced to enables the implicit mapping of training data from the input space to the feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \tag{9}$$

where $K(\cdot, \cdot)$ is a function which satisfies the Mercer's Conditions (see Vapnik (1995), Burges (1998)). Typical kernel functions include:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2 / 2\sigma^2} \tag{10}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^p \tag{11}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i^\top \mathbf{x}_j + \delta) \tag{12}$$

where (10) gives a Gaussian radial basis function (RBF), (11) is a polynomial function of degree p and (12) implements a two-layer sigmoidal neural network. This lead to the Hessian matrix:

$$\mathbf{H}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \tag{13}$$

where H is a symmetric positive definite matrix.

Hence training an optimal sparse LS-SVM is formulated by the linear system of (8), which is traditionally solved by applying the Conjugate-Gradient(CG) method.

3. CONJUGATE-GRADIENT METHOD FOR TRAINING LS-SVMS

For a linear system of

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{14}$$

where the square matrix A is symmetric positive definite, the conjugate gradient method can find its numerical solutions, denoted by $\mathbf{x}_\star$ iteratively. The iteration starts with the input vector $\mathbf{x}_0$ which is an initial approximation of $\mathbf{x}_\star$. Without loss of generality, assume that $\mathbf{x}_0 = 0$.

Conjugate-Gradient (CG) Algorithm

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$$
$$\mathbf{p}_0 = \mathbf{r}_0$$
$$k = 0;$$

repeat

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$$

if $\mathbf{r}_{k+1} <$ a predefined stopping tolerance

$$\beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$$
$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$
$$k = k + 1$$

end

end **repeat**

Although $\mathbf{Z} = \mathbf{H} + \gamma^{-1}\mathbf{E}$ in (8) is positive definite, its associated $(n+1)$-by-$(n+1)$ square matrix is not. Meanwhile an equivalent linear system of (8) is :

$$\left[\begin{array}{c|c} \mathbf{Z} & 0 \\ \hline 0 & \mathbf{I}^\top \mathbf{Z}^{-1}\mathbf{I} \end{array}\right] \left[\begin{array}{c} \alpha + b\mathbf{Z}^{-1}\mathbf{I} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{I}^\top \mathbf{Z}^{-1}\mathbf{y} \\ \hline 0 \end{array}\right] \tag{15}$$

and $s = \mathbf{I}^\top \mathbf{Z}^{-1}\mathbf{I} > 0$

From (15), the traditional training algorithm of an LS-SVM is derived.

Training LS-SVMs Using Conjugate-Gradient

1. Solve the following two linear systems
$$\mathbf{Z}\eta = \mathbf{I} \tag{16}$$
$$\mathbf{Z}\upsilon = \mathbf{y} \tag{17}$$
using CG method.
2. Solutions of (8) are:
$$b = \eta^\top \mathbf{y}/s$$
$$\alpha = \upsilon - b\eta$$

Since (8) is one linear system of size $n+1$ while both (16) and (17) are of size n, solving (16) and (17) obviously require more system resources and computation time than solving only (8). Also, the computation complexity of training LS-SVMs is greatly influenced by that of CG method. However, CG method can be exhaustive due to its iterative procedure.

4. A VARIANT OF LS-SVM

In this section, it will be shown that after minor modification on LS-SVM formulation, a straightforward application of CG method can be realized. The terms b^2 is appended to (1) as motivated by Mangasarian and Musicant (1999). The variant of LS-SVMs optimizes both orientation w and the location b of the separating hyperplane. Hence the optimization formulation becomes:

$$\begin{aligned} &\min_{\mathbf{w},b} J_1(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}(\mathbf{w}^\top \mathbf{w} + b^2) + \gamma \sum_{i=1}^{n} e_i^2 \\ &\text{s.t.} \quad \mathbf{w}^\top \phi(\mathbf{x}_i) + b = y_i - e_i \quad i = 1, \ldots, n \end{aligned} \tag{18}$$

where γ is the penalty constant.

The Lagrangian of (18) is:

$$L(\mathbf{w}, b, \mathbf{e}, \alpha) = J_1 - \sum_{i=1}^{n} \alpha_i [\mathbf{w}^\top \phi(\mathbf{x}_i) + b - y_i + e_i] \tag{19}$$

The optimal point of (19) satisfies the following conditions:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \tag{20}$$

$$\sum_{i=1}^{n} \alpha_i = b \tag{21}$$

$$\alpha_i = \gamma e_i \tag{22}$$

$$\mathbf{w}^\top \phi(\mathbf{x}_i) + b = y_i - e_i \tag{23}$$

The linear equations can be further simplified as:

$$(\mathbf{H} + \gamma^{-1}\mathbf{E} + \mathbf{I})\alpha = \mathbf{y} \tag{24}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^\top$, $\alpha = [\alpha_1, \ldots, \alpha_n]^\top$, $\mathbf{I}$ here is an n-by-n matrix of ones, $\mathbf{E}$ is an identity matrix of size n and $\mathbf{H}$ is the same kernel matrix as in (8).

Coefficient matrix $(\mathbf{H} + \gamma^{-1}\mathbf{E} + \mathbf{I})$ in the linear system of (24) is positive definite, whose proof is given in Appendix A. This enables the direct application of CG method to solve (24). So training an LS-SVM is reduced to a single, rather than two, linear system of size n. Compared with the original formulation of (8), the time cost of training an LS-SVM can be reduced by over 50%.

5. FORWARD LINEAR REGRESSION

Although CG method can be directly used with the variant formulation of LS-SVMs, training an LS-SVM can still be time-consuming due to the iterative nature of CG method. To address this issue, a new linear regression method, named Forward Linear Regression (FLR) is introduced, which is a non-iterative process and can solve (24) directly as well.

FLR originates from an efficient nonlinear regression method - the two-stage algorithm proposed in (Li et al. (2005), Li et al. (2006)) which is for fast model selection in nonlinear identification. Given the regression matrix $\mathbf{\Phi} = [\varphi_1, \ldots, \varphi_N] \in R^{n \times N}$ where n is the number of examples, N is the total number of regressor terms, and the output vector $\mathbf{b} \in R^n$, the problem is to model the data with m column entries from $\mathbf{\Phi}$, denoted as $\mathbf{A} \in R^{n \times m}$ and an associated weight vector $\mathbf{x} \in R^m$, such that the following cost function $\mathbf{L}$ is minimized

$$\mathbf{L} = (\mathbf{A}\mathbf{x} - \mathbf{b})^\top (\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{25}$$

For arbitrary $m \leq N$, two-stage algorithm can efficiently find the optimal $\mathbf{A}$ on the condition that $\text{rank}(\mathbf{\Phi}) \geq m$. It consists of two major step: forward model selection and backward model refinement. In the first step, a $\varphi_i (i = 1, \ldots, m)$ is selected from $\mathbf{\Phi}$ at each time which leads to the steepest descent of $\mathbf{L}$. It is easily seen that this procedure of forward model selection, referred to as Forward Linear Regression(FLR) here, can be directly applied to the linear regression of:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{26}$$

if the rank of the matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ is m.

Forward Linear Regression (FLR)

$$\begin{aligned} &\mathbf{y}_1 = \mathbf{b} \\ &\mathbf{A}_1 = \mathbf{A} = \left[\mathbf{a}_1^{(1)}, \ldots, \mathbf{a}_N^{(1)}\right] \\ &\mathbf{s} = [\,] \\ &\mathbf{C}_0 = [\,] \\ &i = 1 \\ &\textbf{repeat} \\ &\quad \text{for } j = i \text{ to } N \text{ do} \\ &\quad\quad \Upsilon(j) = \frac{\left(\mathbf{a}_j^{(i)}\right)^\top \mathbf{y}}{\left(\mathbf{a}_j^{(i)}\right)^\top \mathbf{a}_j^{(i)}} \\ &\quad \text{end} \\ &\quad\quad k = \underset{j}{\text{argmax}}\, \Upsilon(j) \\ &\quad\quad \mathbf{s} = [\mathbf{s} \quad k] \end{aligned} \tag{27}$$

if $k \neq i$

swap $\mathbf{a}_i$ and $\mathbf{a}_k$

end if

$$\mathbf{C}_i = \left[\mathbf{C}_{i-1},\ (\mathbf{a}_i^{(i)})^\top \mathbf{a}_i^{(i)}\right]$$

$$t_i = (\mathbf{a}_i^{(i)})^\top \mathbf{y}_i$$

$$\mathbf{y}_{i+1} = \mathbf{y}_i - \frac{(\mathbf{a}_i^{(i)})^\top \mathbf{y}_i}{(\mathbf{a}_i^{(i)})^\top \mathbf{a}_i^{(i)}}\, \mathbf{a}_i^{(i)}$$

$$\mathbf{A}_{i+1} = \mathbf{A}_i\left(\mathbf{E} - \frac{\mathbf{a}_i^{(i)}(\mathbf{a}_i^{(i)})^\top}{(\mathbf{a}_i^{(i)})^\top \mathbf{a}_i^{(i)}}\right)$$

$i = i + 1$

until $i = m + 1$

Denote the m columns in $\mathbf{C}_m$ as $[c_1^{(1)}, \ldots, c_m^{(m)}]$ and compute

$$\theta_m = t_m / c_m^{(m)}$$

$$\theta_k = \Big(t_k - \sum_{i=k+1}^{m} c_i^{(i)} \theta_i\Big) / c_k^{(k)}, \quad k = m-1, \ldots, 1$$

Define $\theta = [\theta_1, \ldots, \theta_m]$ and $\mathbf{A}_s$ is the matrix generated from rearranging the columns of $\mathbf{A}$ according to $\mathbf{s}$ which is a permutation of the column indexes. Then $(\mathbf{A}_s, \theta)$ is the solution of (26).

With FLR algorithm, the matrix $\mathbf{A}$ in (26) is required to be invertible rather than a stricter condition of positive definite in CG. Since the n-by-n square coefficient matrix in (24) is positive definite, it is obviously invertible. Hence FLR can be directly applied to solving (24). The superiority of FLR algorithm to CG method is that it is a non-iterative procedure. These properties make possible training an LS-SVM by FLR more efficient.

6. EXPERIMENTS AND RESULTS

The FLR was benchmarked against the CG training for LS-SVMs on a two-spiral dataset accessible from the Carnegie Mellon repository (see Fahlman (1993)). The dataset has two groups of spiral-shaped data points, with 97 points for each. The benchmark is illustrated in Fig. 1 in which class "+" and "-" marked separately by cross and circle. The CG method was implemented by the software package for LS-SVMS that is provided in (Pelckmans et al. (2002)). Both algorithms were executed using Matlab 7 on a Pentium 1.83GHz machine running Windows XP. The Gaussian RBF was chosen as the kernel function. Parameters of an LS-SVM, which included penalty constant γ of (1)and the width σ of RBF kernel, were tuned by ten-fold cross-validation.

Stopping tolerance of iterations in the CG training was set at $1e^{-10}$ and maximum number of iterations was 194. Parameter settings for the CG were: $\gamma = 11.33$, $\sigma^2 = 0.1603$, and parameter settings for FLR were: $\gamma = 0.5$, $\sigma^2 = 50$. Fig. 2 and 3 show the performance of the resultant two-Spiral LS-SVM classifiers trained by both the CG and the FLR respectively. The black solid line of these figures are the optimal separating hyperplane. Fig. 2

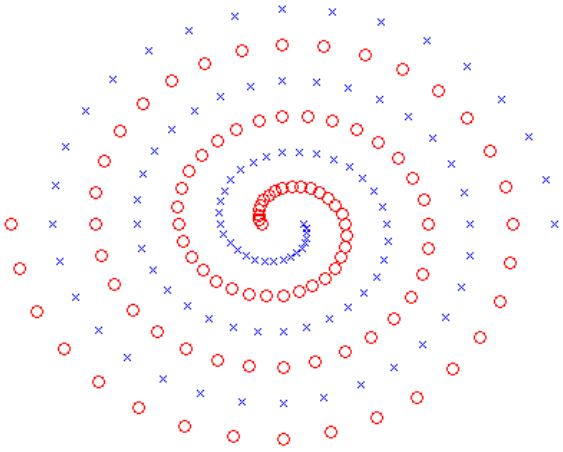

Fig. 1. Two-Spiral Dataset

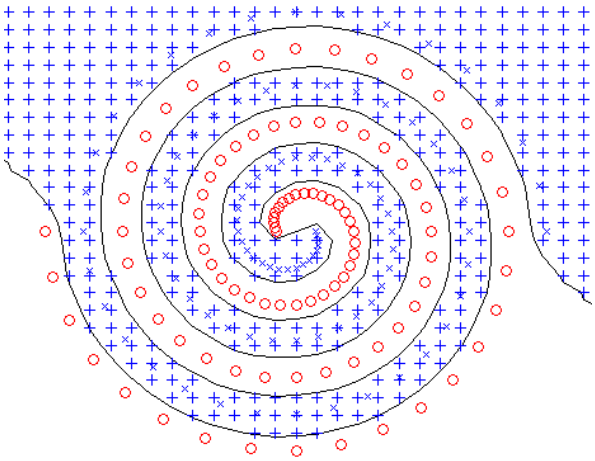

Fig. 2. Two-Spiral LS-SVM classifier trained by CG method with stopping tolerance $1e^{-10}$

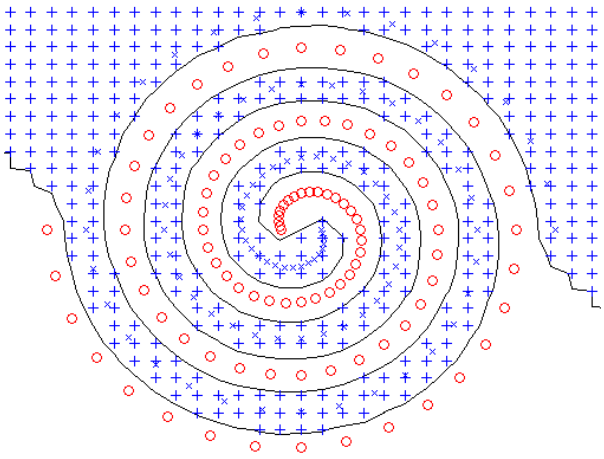

Fig. 3. Two-Spiral LS-SVM classifier trained by FLR method

implies that the FLR algorithm can train the LS-SVM to achieve an excellent generalization. Meanwhile the training time for the CG method was 0.5160 CPU seconds while only 0.281 CPU seconds for the FLR, which again confirm that the can FLR reduce the computation time by 45%.

7. CONCLUSIONS

This paper presents a new fast method - Forward Linear Regression (FLR) to speed up the LS-SVM training. The LS-SVMs are traditionally trained by the Conjugate-Gradient (CG) method which requires that the regression matrix is positive definite. This prevents a direct application of the CG method, leading to a slowdown of the training process. The FLR on the other hand sets a much more flexible condition on the regression matrix - the matrix only needs to be invertible. The size of the linear system related to the LS-SVM training by FLR is only around half of the CG method. Moreover, the CG method is an iterative algorithm which can be time consuming while FLR is not. These techniques give the FLR some advantages in terms of the training time over the CG method. Simulation results also confirm that the FLR training of LS-SVMs reduces the time complexity by almost 50%.

REFERENCES

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, New York, 2000.

S. Fahlman. Cmu benchmark collection for neural-net learning algorithms. technical report, 1993. *Carnegie Mellon Univ., School of Computer Sci., Pittsburgh, PA*, 1993.

R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, Chichester and New York, 1987.

T. V. Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore MD: Johns Hopkins University Press, 1989.

T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in kernel methods: support vector learning*, pages 169–184. MIT Press, Cambridge, MA, 1998.

K. Li, J.-X. Peng, and G. W. Irwin. A fast nonlinear model identification method. *IEEE Trans. Automat. Control.*, 50(8):1211–1216, 2005.

K. Li, J.-X. Peng, and E. Bai. A two-stage algorithm for identification of nonlinear dynamic systems. *Automatica*, 42(7):1189–1197, 2006.

O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.

K. Pelckmans, J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. Ls-svmlab: a matlab/c toolbox for least squares support vector machines. *http://www.esat.kuleuven.ac.be/sista/lssvmlab*, 2002.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9 (3):293–300, 1999.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

X. Xia and K. Li. A sparse multi-class least squares support vector machine. In *IEEE International Symposium on Industrial Electronics (accepted), Cambridge, UK*, July 2008.

X. Xia, K. Li, and G. W. Irwin. Improved training of an optimal sparse least squares support vector machine. In *17th IFAC World Congress (accepted), Seoul, Korea*, July 2008.

Appendix A

Proposition: Coefficient matrix $(\mathbf{H}+\gamma^{-1}\mathbf{E}+\mathbf{I})$ in the linear system of (24) is positive definite.

Proof: Obviously, $\mathbf{H}$ and $\gamma^{-1}\mathbf{E}$ are positive definite.

For the matrix $\mathbf{I}$, its eigenvalues λ are the solutions to the equation:

$$\det(\mathbf{I}-\lambda\mathbf{E})=0 \tag{A.1}$$

$$\begin{aligned}\det(\mathbf{I}-\lambda\mathbf{E}) &= \begin{vmatrix} 1-\lambda & 1 & \dots & 1 \\ 1 & 1-\lambda & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1-\lambda \end{vmatrix} \\ &= \begin{vmatrix} -\lambda & 0 & \dots & 0 & 0 \\ 0 & -\lambda & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -\lambda & 0 \\ 1 & 2 & \dots & n-1 & n-\lambda \end{vmatrix} \\ &= (-\lambda)^{(n-1)} \times (n-\lambda)\end{aligned}$$

Hence the eigenvalues of matrix $\mathbf{I}$ are:

$$\lambda_1=\lambda_2=\ldots=\lambda_{n-1}=0, \lambda_n=n$$

Since all the eigenvalues of matrix $\mathbf{I}$ are all greater than or equal to zero, $\mathbf{I}$ is then positive semi-definite. Consequently, matrix $(\mathbf{H}+\gamma^{-1}\mathbf{E}+\mathbf{I})$ is positive definite.