Sparse LS-SVMs with L_0–norm minimization

J. López[1], K. De Brabanter[2], J.R. Dorronsoro[1] and J.A.K. Suykens[2] *

1- Universidad Autónoma de Madrid (UAM)
Departamento de Ingeniería Informática–IIC
C/ Francisco Tomás y Valiente 11, 28049 Madrid, Spain

2- Katholieke Universiteit Leuven (K.U. Leuven)
Departement Electrotechniek (ESAT–SCD–SISTA)
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

Abstract. Least-Squares Support Vector Machines (LS-SVMs) have been successfully applied in many classification and regression tasks. Their main drawback is the lack of sparseness of the final models. Thus, a procedure to sparsify LS-SVMs is a frequent desideratum. In this paper, we adapt to the LS-SVM case a recent work for sparsifying classical SVM classifiers, which is based on an iterative approximation to the L_0-norm. Experiments on real-world classification and regression datasets illustrate that this adaptation achieves very sparse models, without significant loss of accuracy compared to standard LS-SVMs or SVMs.

1 Introduction

LS-SVMs were introduced in [1] as a simplification of SVMs with several advantages: 1) the dual formulations for classification and regression are identical (after a transformation of variables), 2) this common formulation can be rewritten as a linear system of equations, 3) the subsequent models show a similar performance to the SVM ones in real-world problems.

The most serious drawback of LS-SVM models is the lack of sparseness, as nearly all patterns become support vectors (SVs). For this reason, there are numerous works addressing the sparseness of LS-SVM models, mostly based on two categories: 1) pruning after training and then retraining, and 2) enforcing sparseness from the beginning.

The first category started with [2], where the patterns with smallest $|\alpha_i|$ are eliminated. We retrain and repeat the procedure until the performance degrades too much. Later, [3] suggested to eliminate those closest to the decision boundary. A problem in both works is that it is not guaranteed that the number of SVs will be greatly reduced.

*J. López is a doctoral researcher kindly supported by the FPU grant AP2007-00142. K. De Brabanter is a doctoral researcher at K.U. Leuven. J.A.K. Suykens is a professor at K.U. Leuven. J.R. Dorronsoro is a professor at UAM. Research supported by Spain's TIN2010-21575–C02–01 project, Cátedra IIC en Modelado y Predicción, and Research Council K.U. Leuven: GOA AMBioRICS, GOA-MaNet, CoE EF/05/006, OT/03/12, PhD/postdoc & fellow grants; Flemish Government: FWO PhD/postdoc grants, FWO projects G.0499.04, G.0211.05, G.0226.06, G.0302.07; Research communities (ICCoS, ANMMM, MLDM); AWI: BIL/05/43, IWT: PhD Grants; Belgian Federal Science Policy Office: IUAP DYSCO. E-mail: {j.lopez, jose.dorronsoro}@uam.es, {kris.debrabanter, johan.suykens}@esat.kuleuven.be.

If, instead of solving the dual as a KKT system, we use the Sequential Minimal Optimization (SMO) algorithm [4], a pruning scheme is described in [5], but it only covers the case of LS-SVMs with no bias terms. In the extreme case, we can prune just one pattern per iteration, like in [6], but it suffers from the high cost of matrix inversions.

Switching to the second category, we can fix in advance the number of SVs of the final model [7]. Its main problem is the loss of performance when the number selected is too small. A somewhat similar method is the one in [8], where a set of linearly independent patterns is sought to reduce the size of the kernel matrix. However, some datasets are reported for which almost no reduction is possible.

The L_0-norm has been receiving increasing attention in the Machine Learning community. It counts the number of non-zero elements of a vector. Therefore, minimizing it leads to very sparse models. Unfortunately, this cannot be directly done, since the resulting problem is NP-hard, so some approximations to it are discussed in [9].

An iterative scheme that converges to the L_0-norm is described in [10], used for achieving sparseness in SVM classifiers. In this paper, we will adapt this scheme to the LS-SVM case, not only for classification but also for regression. Another difference is that we do not try to sparsify the bias term, since that is not always convenient.

The rest of the paper is organized as follows. Section 2 describes the LS-SVM primal and dual formulations. Next, the sparsifying procedure is explained in Section 3. Its performance is reported in Section 4 for four real datasets. Finally, Section 5 gives a brief discussion and pointers to further work.

2 Least Squares Support Vector Machines

Given a sample of N patterns $\{X_i, y_i\}, i = 1, \ldots, N$, where $X_i \in \mathbb{R}^m$ and $y_i \in \{+1, -1\}$ for classification and $y_i \in \mathbb{R}$ for regression, the LS-SVM primal problem is formulated as follows:

$$\begin{array}{ll} \min\limits_{W,b,\xi} & \frac{1}{2}\|W\|^2 + \frac{C}{2}\sum_i \xi_i^2 \\ s.t. & W \cdot \Phi\left(X_i\right) + b = y_i - \xi_i, \quad \forall i, \end{array} \tag{1}$$

where $\cdot$ denotes the vector inner product (dot product), and $\Phi(X_i)$ is the image of pattern X_i in the feature space with feature map $\Phi(\cdot)$. Using coefficients α_i for the Lagrangian $\mathcal{L}$, we get

$$\partial\mathcal{L}/\partial W = 0 \;\Rightarrow\; W = \sum_i \alpha_i \Phi(X_i), \qquad \partial\mathcal{L}/\partial b = 0 \;\Rightarrow\; \sum_i \alpha_i = 0,$$

$$\partial\mathcal{L}/\partial \xi_i = 0 \;\Rightarrow\; \alpha_i = C\xi_i, \qquad \partial\mathcal{L}/\partial \alpha_i = 0 \;\Rightarrow\; W \cdot \Phi\left(X_i\right) + b = y_i - \xi_i \;\; \forall i.$$

Condition $\alpha_i = C\xi_i$ is the reason why LS-SVMs are not sparse: whenever $\xi_i \neq 0$ we have $\alpha_i \neq 0$, so the point X_i will have an influence in the W vector. The only way for ξ_i to be 0 is that $W \cdot \Phi(X_i) + b = y_i$, which is very unlikely.

Combining $\partial\mathcal{L}/\partial W = 0$, $\partial\mathcal{L}/\partial\xi_i = 0$ and $\partial\mathcal{L}/\partial\alpha_i = 0$ we get $y_i(\sum_j \alpha_j K_{ij} + b) = y_i - \alpha_i/C \ \forall i$, where $K_{ij} = k(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j)$, with k a Mercer kernel function. This, together with $\partial\mathcal{L}/\partial b = 0$, form the system:

$$\left[\begin{array}{cc} 0 & 1^T \\ 1 & \tilde{K} \end{array} \right] \left[\begin{array}{c} b \\ \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ y \end{array} \right], \tag{2}$$

with $\tilde{K}$ the modified kernel matrix with elements $\tilde{K}_{ij} = K_{ij} + \delta_{ij}/C$, 1^T a row vector of N ones, and $y = (y_1, \ldots, y_N)^T$. Thus, the LS-SVM model can be found by solving this linear system of equations.

3 Sparsifying procedure

For the description of the procedure we follow the lines of [10]. As a starting point, let us consider the following primal problem:

$$\begin{array}{ll} \min\limits_{\alpha,b,\xi} & \frac{1}{2}\sum_i \lambda_i \alpha_i^2 + \frac{C}{2}\sum_i \xi_i^2 \\ s.t. & \sum_j \alpha_j K_{ij} + b = y_i - \xi_i, \quad \forall i. \end{array} \tag{3}$$

Comparing (3) to (1), notice that now we do not have an explicit W vector, but it still underlies under an implicit feature map $W = \sum_j \alpha_j \Phi(X_j)$. The regularization term now is not on $\|W\|^2$, but on the $\|\alpha\|^2$ vector, via the prefixed λ_i coefficients. Introducing coefficients β for the Lagrangian $\mathcal{L}$, one obtains:

$$\partial\mathcal{L}/\partial\alpha_i = 0 \ \Rightarrow \ \alpha_i = \sum_j \beta_j K_{ij}/\lambda_i, \qquad \partial\mathcal{L}/\partial b = 0 \ \Rightarrow \ \sum_i \beta_i = 0,$$

$$\partial\mathcal{L}/\partial\xi_i = 0 \ \Rightarrow \ \beta_i = C\xi_i, \qquad \partial\mathcal{L}/\partial\beta_i = 0 \ \Rightarrow \ \sum_j \alpha_j K_{ij} + b = y_i - \xi_i \ \forall i.$$

As in the previous section, combining $\partial\mathcal{L}/\partial\alpha_i = 0$, $\partial\mathcal{L}/\partial\xi_i = 0$ and $\partial\mathcal{L}/\partial\beta_i = 0$ yields $\sum_j K_{ij}(\sum_m \beta_m K_{jm})/\lambda_j + b = y_i - \beta_i/C \Rightarrow \sum_m \beta_m H_{im} + b = y_i$, with $H = K\,\mathrm{diag}(\lambda)^{-1}K + I_N/C$ and K the kernel matrix. This, together with $\partial\mathcal{L}/\partial b = 0$, form the system

$$\left[\begin{array}{cc} 0 & 1^T \\ 1 & H \end{array} \right] \left[\begin{array}{c} b \\ \beta \end{array} \right] = \left[\begin{array}{c} 0 \\ y \end{array} \right], \tag{4}$$

which is identical to (2) after we switch from $\tilde{K}$ to H, and from α to β.

At this point, the procedure consists of solving iteratively system (4) for different values of λ. Given the t-th iteration, we will have vector λ^t. From it we can build matrix $H^t = K\,\mathrm{diag}(\lambda^t)^{-1}K + I_N/C$ and solve the system, which will give us the solution β^t and b^t. From this solution we get λ^{t+1} and a new iteration starts. This is summarised in Algorithm 1 (ISLS-SVM).

It can also be shown that, as $t \to \infty$, α_t converges to a stationary point α^*. Moreover, this model is guaranteed to be sparse, since the term $(\alpha^t)^T\,\mathrm{diag}(\lambda)\alpha^t$ in (3) converges to the L_0-norm of α^* [10]. Since this α^* depends on the initial

choice of weights, we set them to the LS-SVM solution, so as to avoid ending up in different local minima solutions.

Algorithm 1 Iterative Sparse LS-SVM (ISLS-SVM)

Solve system (2) to give α and b.
$\lambda_i \leftarrow \alpha_i, i = 1, \ldots, N$.
repeat
  $H \leftarrow K \operatorname{diag}(\lambda)^{-1} K + I_N/C$.
  Solve system (4) to give β and b.
  $\alpha \leftarrow \operatorname{diag}(\lambda)^{-1} K\beta$.
  $\lambda_i \leftarrow 1/\alpha_i^2, i = 1, \ldots, N$.
until convergence
return model (α, b).

4 Experiments

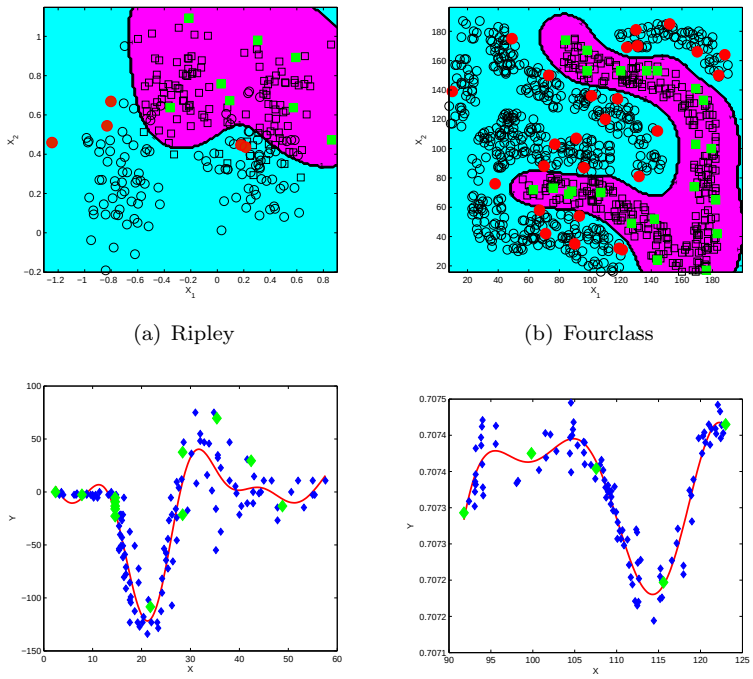
To illustrate the sparsifying ability of ISLS-SVM, we run experiments on 2 classification (*Ripley* [11] and *Fourclass* [12]) and 2 regression (*Motorcycle* [13] and *Fossil* [14]) datasets. In these datasets the input space is $\mathbb{R}^2$. We compare LS-SVMs against ISLS-SVMs and SVMs over the datasets mentioned. The RBF kernel $K_{ij} = \exp(-\|X_i - X_j\|^2/\sigma^2)$ is used for all experiments.

Dataset	N	LS–SVM		ISLS–SVM		SVM	
		Test error	SVs	Test error	SVs	Test error	SVs
Ripley	250	12.8	167.0	13.4	13.0	11.6	61.8
Fourclass	862	0.0	575.0	0.0	54.4	0.0	232.2
Motorcycle	133	503.1	89.0	533.1	8.4	604.7	73.2
Fossil	106	7.5e-10	71.0	8.1e-10	6.2	4.2e-09	48.9

Table 1: Number of patterns and averages of the error rates and SVs in both methods. Errors are given in % of misclassification for classification, and in MSE for regression.

The comparison is performed on an out-of-sample test set consisting of 1/3 of the data. The first 2/3 of the randomized data are reserved for training and/or cross-validation (CV). For a fair comparison, we need to properly tune the hyperparameters of the different machines. For (IS)LS-SVMs, we use the routine *tunelssvm* from the LS-SVMlab Matlab toolbox [15]. For SVMs, we use the LIBSVM [12] software. The number of folds in CV is set to 10. We use 5 multiple starters for the CSA algorithm, and 10 randomizations. Convergence of Algorithm 1 is assumed when the difference $\|\alpha^t - \alpha^{t+1}\|/N$ is lower than 10^{-4} or when the number of iterations t exceeds 50.

For each method, the average errors on the test data and number of SVs are reported in Table 1. It can be seen that the performances of ISLS-SVM are comparable to the ones for LS-SVM. This is remarkable, since the sparse model



(a) Ripley (b) Fourclass

(c) Motorcycle (d) Fossil

Fig. 1: Examples of sparse models (decision surfaces in classification and red lines in regression) obtained by ISLS-SVM.

will not only be much faster in testing (about 10 times faster, considering the SVs), but also will have good generalization abilities. This can be confirmed looking at the results for SVMs, which generalize a bit better for classification, but not for regression. Regarding sparsity, SVMs are sparser than LS-SVMs, but not as sparse as ISLS-SVMs.

Next, we plot in Figure 1 some examples of the obtained ISLS-SVM models, this time over the whole training data. SVs are highlighted with bigger and differently coloured markers. A pattern X_i is considered to be a SV if, after convergence, we get $|\alpha_i| > 10^{-6}$. Notice that the resulting models suit pretty well the data distributions.

5 Discussion

In this work we applied the technique in [10] to sparsify LS-SVMs. This is of great interest, since LS-SVMs suffer from lack of sparseness. Another advantage is that LS-SVMs are formulated in the same way for classification and regression, so the sparsifying procedure is the same for both cases. Results on 2

classification and 2 regression datasets illustrate that the resulting models have a generalization ability comparable to the ones of standard LS-SVMs and SVMs.

However, this procedure is computationally expensive, since each iteration requires solving a system with complexity $\mathcal{O}(N^3)$. The number of iterations is usually quite low (around 15 or 20), but this number of iterations is precisely the factor of extra cost compared to standard LS-SVM training. Thus, further work is needed to accelerate its execution and make it applicable to larger datasets. To this aim, one option may be using the SMO algorithm to solve the associated dual formulations instead of the systems of equations [4].

References

[1] J.A.K. Suykens and J. Vandewalle. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[2] J.A.K. Suykens, L. Lukas, and J. Vandewalle. Sparse Approximation using Least Squares Support Vector Machines. In *Proceedings of ISCAS'00*, pages 757–760, 2000.

[3] Y. Li, C. Lin, and W. Zhang. Improved Sparse Least-Squares Support Vector Machine Classifiers. *Neurocomputing*, 69(13-15):1655–1658, 2006.

[4] J. López and J.A.K. Suykens. First and Second Order SMO Algorithms for LS-SVM classifiers. *Neural Processing Letters*, 33(1):31–44, 2011.

[5] X. Zeng and X.W. Chen. SMO-based Pruning Methods for Sparse Least Squares Support Vector Machines. *IEEE Transactions on Neural Networks*, 16(6):1541–1546, 2005.

[6] B.J. De Kruif and T.J.A. De Vries. Pruning Error Minimization in Least-Squares Support Vector Machines. *IEEE Transactions on Neural Networks*, 14(3):696–702, 2003.

[7] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, and B. De Moor. Optimized Fixed-Size Kernel Models for Large Data Sets. *Computational Statistics & Data Analysis*, 54(6):1484–1504, 2010.

[8] J. Valyon and G. Horvath. A Sparse Least Squares Support Vector Machine Classifier. In *Proceedings of IJCNN'04*, pages 543–548, 2004.

[9] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the Zero Norm with Linear Models and Kernel Methods. *Journal of Machine Learning Research*, 3:1439–1461, 2003.

[10] K. Huang, D. Zheng, J. Sun, Y. Hotta, K. Fujimoto, and S. Naoi. Sparse Learning for Support Vector Classification. *Pattern Recognition Letters*, 31(13):1944–1951, 2010.

[11] University of California Irvine. UCI Machine Learning Repository. Datasets available at `http://archive.ics.uci.edu/ml`.

[12] C.-C. Chang and C.-J. Lin. *LIBSVM: a Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[13] R.L. Eubank. *Nonparametric Regression and Spline Smoothing*. Marcel Dekker, New York, 1999.

[14] Y.T. Chien. *Interactive Pattern Recognition*. Marcel Dekker, New York, 1978.

[15] K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, and J.A.K. Suykens. LS-SVMlab Toolbox User's Guide version 1.7. Technical Report 10-146, Katholieke Universiteit Leuven, 2010. Software available at `http://www.esat.kuleuven.be/sista/lssvmlab`.