Computational Statistics and Data Analysis 54 (2010) 1484–1504

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

Optimized fixed-size kernel models for large data sets

K. De Brabanter [a,*], J. De Brabanter [a,b], J.A.K. Suykens [a], B. De Moor [a]

[a] Department of Electrical Engineering ESAT-SCD, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
[b] Hogeschool KaHo Sint-Lieven, (Associatie K.U. Leuven), Departement Industrieel Ingenieur, B-9000, Gent, Belgium

ARTICLE INFO

Article history:
Received 13 October 2009
Received in revised form 7 January 2010
Accepted 19 January 2010
Available online 28 January 2010

Keywords:
Kernel methods
Least squares support vector machines
Classification
Regression
Plug-in estimate
Entropy
Cross-validation

ABSTRACT

A modified active subset selection method based on quadratic Rényi entropy and a fast cross-validation for fixed-size least squares support vector machines is proposed for classification and regression with optimized tuning process. The kernel bandwidth of the entropy based selection criterion is optimally determined according to the solve-the-equation plug-in method. Also a fast cross-validation method based on a simple updating scheme is developed. The combination of these two techniques is suitable for handling large scale data sets on standard personal computers. Finally, the performance on test data and computational time of this fixed-size method are compared to those for standard support vector machines and ν-support vector machines resulting in sparser models with lower computational cost and comparable accuracy.

1. Introduction

Support vector machines (SVM) (Vapnik, 1995, 1999) and least squares support vector machines (LS-SVM) (Suykens and Vandewalle, 1999; Suykens et al., 2002) are state of the art learning algorithms for pattern recognition and function estimation. Typically a quadratic programming (QP) problem has to be solved in dual space in order to determine the SVM model. The formulation of the optimization problem in the primal space associated with this QP problem involves inequality constraints in the form of box constraints and an additional equality constraint.

Unfortunately, the designs of QP solvers, e.g. MINOS and LOQO, assume that the full kernel matrix is readily available. To overcome this difficulty, decomposition methods (Osuna et al., 1997a,b; Saunders et al., 1998; Joachims, 1999) were designed. A particular case of the decomposition method is iterative chunking where the full scale problem is restricted to a small subset of training examples called the working set. An extreme form of chunking is sequential minimal optimization (SMO) proposed by Platt (1999). SMO uses the smallest possible working set size, i.e. two elements. This choice greatly simplifies the method. Due to this, SMO is considered as the current state of the art QP solver for solving medium scale as well as large scale SVM.

In the LS-SVM formulation the inequality constraints are replaced by equality constraints and a sum of squared errors (SSE) cost function is used. Due to the use of equality constraints and the L_2 cost function in LS-SVM the solution is found by solving a linear system instead of quadratic programming. To tackle large scale problems with LS-SVM, Suykens et al. (1999) and Van Gestel et al. (2004) effectively employed the Hestenes–Stiefel conjugate gradient algorithm (Golub and Van Loan, 1989; Suykens et al., 1999). This method is well suited for problems with a larger number of data (up to about 10,000 data points). As an alternative, an iterative algorithm for solving large scale LS-SVM was proposed by Keerthi and Shevade (2003). This method is based on the solution of the dual problem using an idea similar to that of the SMO algorithm, i.e. using Wolfe duality theory, for SVM.

* Corresponding author. Tel.: +32 16 328658.
E-mail addresses: kris.debrabanter@esat.kuleuven.be (K. De Brabanter), jos.debrabanter@kahosl.be (J. De Brabanter), johan.suykens@esat.kuleuven.be (J.A.K. Suykens), bart.demoor@esat.kuleuven.be (B. De Moor).

0167-9473/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2010.01.024

The vast majority of textbooks and articles discussing SVM and LS-SVM first state the primal optimization problem and then go directly to the dual formulation (Vapnik, 1995; Suykens and Vandewalle, 1999). A successful attempt at solving LS-SVM in primal weight space resulting in a parametric model and sparse representation, introduced by Suykens et al. (2002), is referred to as using *fixed-size least squares support vector machines* (FS-LSSVM) and was also applied in Espinoza et al. (2007). In this method an explicit expression for the feature map or an approximation to it is required. A procedure for finding this approximated feature map is based on the Nyström method (Nyström, 1930; Baker, 1977). Williams and Seeger (2001) used the Nyström method to speed up Gaussian processes (GP) (Williams and Barber, 1998). The Nyström method is related to finding a low rank approximation to the given kernel matrix by choosing m rows or columns of the kernel matrix. Many ways of selecting those m rows or columns of the kernel matrix can be found in the literature (Suykens et al., 2002; Achlioptas et al., 2002; Drineas and Mahoney, 2005). Smola and Schölkopf (2000) presented a sparse greedy approximation technique for constructing a compressed representation of the kernel matrix. This technique approximates the kernel matrix by the subspace spanned by a subset of its columns. The basis vectors are chosen incrementally to minimize an upper bound of the approximation error. A comparison of some of the above mentioned techniques can be found in Hoegaerts et al. (2004). Suykens et al. (2002) proposed searching for m rows or columns while maximizing the quadratic Rényi entropy criterion and estimate in the primal space leading to a sparse representation. This criterion will be used in the remainder of the paper.

The kernel representation of the quadratic Rényi entropy, established by Girolami (2002) and related to density estimation and principal component analysis, requires a bandwidth specific to the entropy criterion. Numerous bandwidth selection methods for density estimation exist, e.g. least squares cross-validation (LSCV) (Rudemo, 1982; Bowman, 1984), biased cross-validation (BCV) (Scott and Terrel, 1987), smoothed bootstrap (SB) (Taylor, 1989; Faraway and Jhun, 1990), plug-ins (Hall, unpublished manuscript; Sheather, 1986; Sheather and Jones, 1991), reference rules (Deheuvels, 1977; Silverman, 1986). In this paper we use the *solve-the-equation plug-in method* (Sheather and Jones, 1991) which is related to the plug-in family. The rationale for using this method is based on the fact that the calculation can be done efficiently using the improved fast Gauss transform (IFGT) (Yang et al., 2003) and hence it is computationally more efficient than LSCV, BCV and SB. Also it has better convergence rates than the above mentioned methods (Sheather, 2004).

Kernel based methods require the determination of tuning parameters including a regularization constant and kernel bandwidth. A widely used technique for estimating these parameters is cross-validation (CV) (Burman, 1989). A simple implementation of v-fold cross-validation trains a classifier/regression model for each split of the data and is thus computationally expensive when v is large, e.g. in leave-one-out (LOO) CV. An extensive literature exists on reducing the computational complexity of v-fold CV and LOO-CV; see e.g. (Vapnik and Chapelle, 2000; Wahba et al., 2000) for SVM, (Ying and Keong, 2004; An et al., 2007) for LS-SVM and (Cawley and Talbot, 2004) for sparse LS-SVM. Using the fact that the FS-LSSVM training problem has a closed form, we apply a simple updating scheme to develop a fast v-fold CV suitable for large data sets. For typical 10-fold CV, the proposed algorithm is 10 to 15 times faster than the simple implementation. Experiments also show that the complexity of the proposed algorithm is not very sensitive to the number of folds.

A typical method for estimating the tuning parameters would define a grid (grid-search) over these parameters of interest and perform v-fold CV for each of these grid values. However, three disadvantages come up with this approach (Bennett et al., 2006). A first disadvantage of such a grid-search CV approach is the limitation of the desirable number of tuning parameters in a model, due to the combinatorial explosion of grid points. A second disadvantage of this approach is their practical inefficiency; namely, they are incapable of assuring the overall quality of the solution produced. A third disadvantage in grid-search is that the discretization fails to take into account the fact that the tuning parameters are continuous. Therefore we propose an alternative for finding better tuning parameters. Our strategy is based on the recently developed coupled simulating annealing (CSA) method with variance control proposed by Xavier de Souza et al. (2006, in press). Global optimization methods are typically very slow. For many difficult problems, ensuring convergence to a global optimum might mean impractical running times. For such problems, a reasonable solution might be enough in exchange for a faster convergence. Precisely for this reason, many simulated annealing (SA) algorithms (Ingber, 1989; Rajasekaran, 2000) and other heuristic based techniques have been developed. However, due to speed-up procedures, these methods often get trapped in poor optima. The CSA method used in this paper is designed to easily escape from local optima and thus improves the quality of solution without compromising too much the speed of convergence. To better understand the underlying principles of these classes of methods consider the work of Suykens et al. (2001). One of the largest differences from SA is that CSA features a new form of acceptance probability functions that can be applied to an ensemble of optimizers. This approach considers several current states which are coupled together by their energies in their acceptance function. Also, in contrast with the case for classical SA techniques, parallelism is an inherent characteristic of this class of methods.

In this paper we propose a fast cross-validation technique suitable for large scale data sets. We modify and apply the solve-the-equation plug-in method for entropy bandwidth selection. Finally, we combine a fast global optimization technique with a simplex search in order to estimate the tuning parameters (regularization parameter and kernel bandwidth).

This paper is organized as follows. In Section 2 we give a short introduction concerning LS-SVM for classification and regression. In Section 3 we discuss the estimation in the primal weight space. Section 4 explains the active selection of a subsample based on the quadratic Rényi entropy together with a fast optimal bandwidth selection method using the *solve-the-equation plug-in* method. Section 5 describes a simple heuristic for determining the number of PV (prototype vectors). Section 6 discusses the proposed v-fold CV algorithm for FS-LSSVM. In Sections 7 and 8 the different algorithms are successfully demonstrated on real-life data sets. Section 9 states the conclusions of the paper. Finally, Appendix gives a detailed discussion of CSA.

2. Least squares support vector machines

In this section we give a brief summary on basic principles of least squares support vector machines (LS-SVM) for classification and regression.

2.1. Classification

Given a training set defined as $\mathcal{D}_n = \{(X_k, Y_k) : X_k \in \mathbb{R}^d, Y_k \in \{-1, +1\}; k = 1, \ldots, n\}$, where X_k is the k-th input pattern and Y_k is the k-th output pattern, in the primal weight space the optimization problem for classification becomes (Suykens and Vandewalle, 1999)

$$\min_{w,b,e} \mathcal{J}(w, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{n} e_k^2$$
$$\text{s.t. } Y_k[w^T \varphi(X_k) + b] = 1 - e_k, \quad k = 1, \ldots, n.$$

The classifier in the primal weight space takes the form

$$y(x) = \text{sign}[w^T \varphi(x) + b],$$

where $\varphi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is the feature map to the high dimensional feature space, where n_h denotes the dimension of the feature space (which can be infinite dimensional), as in the standard support vector machine (SVM) case (Vapnik, 1999) and $w \in \mathbb{R}^{n_h}, b \in \mathbb{R}$.

Using Lagrange multipliers, the classifier can be computed in the dual space and is given by Suykens and Vandewalle (1999)

$$\hat{y}(x) = \text{sign}\left(\sum_{k=1}^{n} \hat{\alpha}_k Y_k K(x, X_k) + \hat{b} \right),$$

with $K(x, X_k) = \varphi(x)^T \varphi(X_k)$, where α and b are the solutions of the linear system

$$\left(\begin{array}{c|c} 0 & Y^T \\ \hline Y & \Omega + \frac{1}{\gamma} I_n \end{array} \right) \left(\begin{array}{c} b \\ \hline \alpha \end{array} \right) = \left(\begin{array}{c} 0 \\ \hline 1_n \end{array} \right),$$

with $1_n = (1, \ldots, 1)^T$, $\Omega_{kl} = Y_k Y_l \varphi(X_k)^T \varphi(X_l)$ where $K(X_k, X_l) = \varphi(X_k)^T \varphi(X_l)$ is a positive definite kernel. An extension of this binary classification problem to multi-class problems is formulated in Suykens et al. (2002). In SVM and LS-SVM one chooses the kernel K. A positive definite K guarantees the existence of the feature map φ but φ is often not explicitly known.

2.2. Regression

Given a training set defined as $\mathcal{D}_n = \{(X_k, Y_k) : X_k \in \mathbb{R}^d, Y_k \in \mathbb{R}; k = 1, \ldots, n\}$ of size n drawn i.i.d. from an unknown distribution F_{XY} according to

$$Y_k = g(X_k) + e_k, \quad k = 1, \ldots, n,$$

where $e_k \in \mathbb{R}$ are assumed to be i.i.d. random errors with $E[e_k | X = X_k] = 0, Var[e_k] = \sigma^2 < \infty, g \in C^z(\mathbb{R})$ with $z \geq 2$, is an unknown real-valued smooth function and $E[Y_k | X = X_k] = g(X_k)$, the optimization problem of finding the vector $w \in \mathbb{R}^{n_h}$ and $b \in \mathbb{R}$ for regression can be formulated as follows (Suykens et al., 2002):

$$\min_{w,b,e} \mathcal{J}(w, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{n} e_k^2 \tag{1}$$
$$\text{s.t. } Y_k = w^T \varphi(X_k) + b + e_k, \quad k = 1, \ldots, n.$$

The cost function $\mathcal{J}$ consists of a residual sum of squares (RSS) fitting error and a regularization term corresponding to ridge regression in the feature space (Golub and Van Loan, 1989) with an additional bias term.

However, one does not need to evaluate w and $\varphi(\cdot)$ explicitly. By using Lagrange multipliers, the solution of (1) can be obtained by taking the Karush–Kuhn–Tucker (KKT) (Fletcher, 1987) conditions for optimality. The result is given by the following linear system in the dual variables α:

$$\left(\begin{array}{c|c} 0 & 1_n^T \\ \hline 1_n & \Omega + \frac{1}{\gamma} I_n \end{array} \right) \left(\begin{array}{c} b \\ \hline \alpha \end{array} \right) = \left(\begin{array}{c} 0 \\ \hline Y \end{array} \right),$$

with $Y = (Y_1, \ldots, Y_n)^T$, $1_n = (1, \ldots, 1)^T$, $\alpha = (\alpha_1, \ldots, \alpha_n)^T$ and $\Omega_{kl} = \varphi(X_k)^T \varphi(X_l) = K(X_k, X_l)$, with $K(X_k, X_l)$ positive definite, for $k, l = 1, \ldots, n$. According to Mercer's theorem, the resulting LS-SVM model for function estimation becomes

$$\hat{g}(x) = \sum_{k=1}^{n} \hat{\alpha}_k K(x, X_k) + \hat{b},$$

where $K(\cdot, \cdot)$ is an appropriately chosen positive definite kernel.

3. Estimation in the primal weight space

In the following subsection we review how to estimate an approximation to the feature map φ using the Nyström method (Williams and Seeger, 2001). The computed eigenfunctions will then be used to solve the problem in primal space.

3.1. Approximation to the feature map

For large data sets it is often advantageous to solve the problem in the primal space (Suykens et al., 2002). However, one needs to have an explicit expression for φ or an approximation to the feature map $\hat{\varphi} : \mathbb{R}^d \to \mathbb{R}^m$, with $m \ll n$. Let $X_k \in \mathbb{R}^d$, $k = 1, \ldots, n$, be a random sample from an unknown distribution F_X. Let C be a compact subset of $\mathbb{R}^d$, $V = L^2(C)$ and $M(V, V)$ be a class of linear operators from V into V. Consider the eigenfunction expansion of a kernel function

$$K(x, t) = \sum_i \lambda_i \phi_i(x)\phi_i(t),$$

where $K(x, t) \in V$, $\lambda_i \in \mathbb{C}$ and $\phi_i \in V$ are respectively the eigenvalues and the eigenfunctions, defined by the Fredholm integral equation of the first kind

$$\begin{aligned}(T\phi_i)(t) &= \int_C K(x, t)\phi_i(x)\mathrm{d}F_X(x) \\ &= \lambda_i \phi_i(t),\end{aligned} \tag{2}$$

where $T \in M(V, V)$.

One can discretize (2) on a finite set of evaluation points $\{X_1, \ldots, X_n\} \in C \subseteq \mathbb{R}^d$ with associated weights $v_k \in \mathbb{R}$, $k = 1, \ldots, n$. Define a quadrature method Q_n, $n \in \mathbb{N}$

$$Q_n = \sum_{k=1}^{n} v_k \psi(X_k).$$

Let $v_k = \frac{1}{n}$, $k = 1, \ldots, n$; the Nyström method (Williams and Seeger, 2001) approximates the integral by means of Q_n and determines an approximation ϕ_i by

$$\lambda_i \phi_i(t) \approx \frac{1}{n} \sum_{k=1}^{n} K(X_k, t)\phi_i(X_k), \quad \forall\, t \in C \subseteq \mathbb{R}^d. \tag{3}$$

Let $t = X_j$; in matrix notation one obtains

$$\Omega U = U\Lambda,$$

where $\Omega_{kj} = K(X_k, X_j)$ are the elements of the kernel matrix, $U = (u_1, \ldots, u_n)$ is an $n \times n$ matrix of eigenvectors of Ω and Λ is an $n \times n$ diagonal matrix of nonnegative eigenvalues in decreasing order. Expression (3) delivers direct approximations of the eigenvalues and eigenfunctions for the $x_k \in \mathbb{R}^d$, $k = 1, \ldots, n$, points

$$\phi_i(x_j) \approx \sqrt{n} u_{i,n} \quad \text{and} \quad \lambda_i \approx \frac{1}{n}\lambda_{i,n}, \tag{4}$$

where $\lambda_{i,n}$ and $u_{i,n}$ denote the i-th eigenvalue and the i-th eigenvector of (3) respectively (the subscript n denotes the eigenvalues and eigenvectors of (3) based on the complete data set). The λ_i denote the eigenvalues of (2). Substituting (4) in (3) results in an approximation of an eigenfunction evaluation at point $t \in C \subseteq \mathbb{R}^d$:

$$\hat{\phi}_i(t) \approx \frac{\sqrt{n}}{\lambda_{i,n}} \sum_{k=1}^{n} K(X_k, t) u_{ki,n},$$

with $u_{ki,n}$ the k-th element of the i-th eigenvector of (3). On the basis of the Nyström approximation, an expression for the entries of the approximation of the feature map $\hat{\varphi}_i : \mathbb{R}^d \to \mathbb{R}$, with $\hat{\varphi}(x) = \left(\hat{\varphi}_1(x), \ldots, \hat{\varphi}_m(x)\right)^T$, is given by

$$\begin{aligned}\hat{\varphi}_i(x) &= \sqrt{\lambda_i}\hat{\phi}_i(X) \\ &= \frac{1}{\sqrt{\lambda_{i,n}}} \sum_{k=1}^{n} u_{ki,n} K(X_k, x).\end{aligned} \tag{5}$$

In order to introduce parsimony, one chooses a fixed size m ($m \ll n$) as a working subsample (see Section 4). Likewise an m-approximation can be made and the model takes the form

$$\begin{aligned}\hat{y}(x) &= \tilde{w}^T \hat{\varphi}(X) + b \\ &= \sum_{i=1}^{m} \tilde{w}_i \frac{1}{\sqrt{\lambda_{i,m}}} \sum_{k=1}^{m} u_{ki,m} K(X_k, x) + b,\end{aligned}$$

with $\tilde{w} \in \mathbb{R}^m$. The subscript m denotes the eigenvalues and eigenvectors based on the kernel matrix of the working subsample (prototype vectors).

This method was used in Williams and Seeger (2001) to speed up Gaussian processes (GP). However, Williams and Seeger (2001) used the Nyström method to approximate the eigenvalues and eigenvectors of the complete kernel matrix by using a random subset of size $m \ll n$. A major difference is also that we estimate here in the primal space which leads to a sparse representation. In this paper we use an entropy based criterion to select the prototype vectors (working sample); see Section 4. In this paper we denote vectors chosen by the entropy criterion as prototype vectors (PV) as the vectors serve as prototypes for the underlying density f. The solutions of the SVM QP problem we denote as support vectors (SV).

3.2. Solving the problem in primal space

Given the finite dimensional approximation to the feature map $\hat{\varphi}$, the following ridge regression problem can be solved in the primal weight space with unknowns $\tilde{w} \in \mathbb{R}^m$, $b \in \mathbb{R}$ and m the number of prototype vectors (PV):

$$\min_{\tilde{w},b} \frac{1}{2}\sum_{i=1}^{m} \tilde{w}_i^2 + \gamma \frac{1}{2}\sum_{k=1}^{n} L(Y_k - \tilde{w}^T\hat{\varphi}(X_k) - b), \tag{6}$$

where L denotes the loss function. Taking an L_2 loss function, the optimization problem (6) corresponds to ridge regression with additional intercept term b and corresponds to fixed-size LS-SVM (FS-LSSVM). The solution is given by

$$\begin{pmatrix} \hat{\tilde{w}} \\ \hat{b} \end{pmatrix} = \left(\hat{\Phi}_e^T\hat{\Phi}_e + \frac{I_{m+1}}{\gamma}\right)^{-1}\hat{\Phi}_e^T Y, \tag{7}$$

where $\hat{\Phi}_e$ is the $n \times (m+1)$ extended feature matrix

$$\hat{\Phi}_e = \begin{pmatrix} \hat{\varphi}_1(X_1) & \cdots & \hat{\varphi}_m(X_1) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \hat{\varphi}_1(X_n) & \cdots & \hat{\varphi}_m(X_n) & 1 \end{pmatrix}, \tag{8}$$

and I_{m+1} the $(m+1) \times (m+1)$ identity matrix.

On the other hand by taking an L_1 loss function, one obtains an L_1 regularization problem resulting in more robust solutions than (7); see e.g. Liu et al. (2007). The solution corresponding to the L_1 problem is given by solving a QP problem. It is clear that both approaches directly lead to a sparse representation of the model.

4. Active selection of a subsample

Instead of using a purely random selection of PV, an entropy based selection method is discussed, as proposed by Suykens et al. (2002). It is especially important for selecting the PV. The selected PV should represent the main characteristics of the whole training data set, i.e. they take a crucial role in constructing the FS-LSSVM model. Further, in this section it will be illustrated, by means of a toy example, why this criterion is preferred over a purely random selection. This active selection of PV, based on entropy, refers to finding a subset of size m, with $m \ll n$, of the columns in the kernel matrix that best approximate the kernel matrix.

In the first subsection two entropy criteria will be discussed: Shannon (Shannon, 1948; Vollbrecht and Wolf, 2002) and Rényi (Rényi, 1961; Girolami, 2002; Vollbrecht and Wolf, 2002; Principe et al., 2000) ones. The second subsection reviews a method, based on the solve-the-equation plug-in method introduced by Sheather and Jones (1991), for selecting the smoothing parameter for entropy estimation. We investigate this for use in fixed-size LS-SVM for large scale applications.

4.1. A subsample based on entropy criteria

Let $X_k \in \mathbb{R}^d$, $k = 1, \ldots, n$, be a set of input samples from a random variable $X \in \mathbb{R}^d$. The success of a selection method depends on how much information about the original input sample $X_k \in \mathbb{R}^d$, $k = 1, \ldots, n$, is contained in a subsample $X_j \in \mathbb{R}^d$, $j = 1, \ldots, m$ ($m \ll n$). In other words, the purpose of a subsample selection is to extract m ($m \ll n$) samples from $\{X_1, \ldots, X_n\}$, such that $H_m(x)$, the information or entropy of the subsample, becomes as close to $H_n(x)$ (the entropy of the original sample) as possible. As mentioned before, two entropy criteria will be used, i.e. Shannon and Rényi entropy. The Shannon or differential entropy $H_S(X)$ is defined by

$$\begin{aligned} H_S(X) &= \mathrm{E}[-\log f(X)] \\ &= -\int \ldots \int f(x)\log(f(x))\mathrm{d}x, \end{aligned} \tag{9}$$

and the Rényi entropy $H_{Rq}^m(x)$ of order q is defined as

$$H_{Rq}^m(x) = \frac{1}{1-q}\log\int f(x)^q \mathrm{d}x, \tag{10}$$

with $q > 0, q \neq 1$. In order to compute both entropy criteria it can be seen that an estimate of the density f is required.

The (multivariate) density function f can be estimated by the (multivariate) kernel density estimator (Silverman, 1986; Scott, 1992)

$$\hat{f}(x_1, \ldots, x_d) = \frac{1}{n \prod_{j=1}^{d} h_j} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{d} \kappa \left(\frac{x_i - X_{ij}}{h_j} \right) \right\},$$

where h_j denotes the bandwidth for each dimension j and the kernel $\kappa : \mathbb{R} \to \mathbb{R}$ satisfies $\int_{\mathbb{R}} \kappa(u)\mathrm{d}u = 1$. For $d > 1$, the same (univariate) kernel is used in each dimension but with a different smoothing parameter (bandwidth) for each dimension. The point X_{ij} is the ij-th entry of the given data matrix $(X_1, \ldots, X_n)^T \in \mathbb{R}^{n \times d}$. Another possibility for estimating $f(x)$ is by using the general multivariate kernel estimator (Scott, 1992) given by

$$\hat{f}(x) = \frac{1}{n|D|} \sum_{i=1}^{n} \kappa \left\{ D^{-1}(x - X_i) \right\}, \tag{11}$$

where $|\cdot|$ denotes the determinant, D is a non-singular matrix of the form $D = \mathrm{diag}(h_1, \ldots, h_d)$ and the kernel $\kappa : \mathbb{R}^d \to \mathbb{R}$ satisfies $\int_{\mathbb{R}^d} \kappa(u)\mathrm{d}u = 1$. In what follows the general multivariate kernel estimator (11) will be used.

When the Shannon (differential) entropy (Shannon, 1948), given by (9), is used along with the kernel density estimate $\hat{f}(x)$, the estimation of the entropy $H_S(x)$ becomes very complicated. However, Rényi's entropy of order $q = 2$ (also called quadratic entropy) leads to a simpler estimate of entropy $H^m_{R2}(x)$; see (10). The differential entropy can be viewed as one member of the Rényi entropy family, because $\lim_{q \to 1} H_{Rq}(x) = H_S(x)$. Although Shannon's entropy is the only one which possesses properties such as continuity, symmetry, extremal property, recursivity and additivity for an information measure, the Rényi entropy family is equivalent with respect to entropy maximization (Rényi, 1961, 2007). In real problems, the choice of information measure depends on other requirements such as ease of implementation. Combining (10) and (11), Rényi's quadratic entropy estimator, based on m prototype vectors and setting $q = 2$, becomes

$$\hat{H}^m_{R2}(X) = -\log \frac{1}{m^2 |D|^2} \sum_{k=1}^{m} \sum_{l=1}^{m} \kappa \left\{ \left(D\sqrt{2} \right)^{-1} (X_k - X_l) \right\}. \tag{12}$$

We choose a fixed size m ($m \ll n$) for a working set of data points (prototype vectors) and actively select points from the pool of training input samples as a candidate for the working set (PV). In the working set, a point is randomly selected and replaced by a randomly selected point from the training input sample if the new point improves Rényi's quadratic entropy criterion. This leads to the following active selection algorithm as introduced in Suykens et al. (2002). For classification, Algorithm 1 can be used in a stratified sampling scheme.

Algorithm 1 Active prototype vector selection (Suykens et al., 2002)

1: Given a training set $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, choose a working set of prototype vectors $\mathcal{W}_m \subset \mathcal{D}_n$ of size m at random
2: Randomly select a sample point $X^* \in \mathcal{W}_m$, and $X^+ \in \mathcal{D}_n$, swap (X^*, X^+)
3: **if** $\hat{H}^m_{R2}(X_1, \ldots, X_{m-1}; X^+) > \hat{H}^m_{R2}(X_1, \ldots, X^*_i, \ldots, X_m)$ **then**
4: $X^+ \in \mathcal{W}_m$ and $X^* \notin \mathcal{W}_m$, $X^* \in \mathcal{D}_n$
5: **else**
6: $X^+ \notin \mathcal{W}_m$ and $X^* \in \mathcal{W}_m$, $X^* \in \mathcal{D}_n$
7: **end if**
8: Calculate $\hat{H}^m_{R2}(X)$ for the present $\mathcal{W}_m$
9: Stop if the change in entropy value (12) is small

4.2. Bandwidth selection for entropy estimation

The most popular non-parametric density estimate $\hat{f}$ of a univariate density f based on a random sample $X_1, \ldots, X_n$ is given by

$$\hat{f}(x) = \frac{1}{nh} \sum_{k=1}^{n} \kappa \left(\frac{x - X_k}{h} \right).$$

Efficient use of kernel density estimation requires the optimal selection of the bandwidth of the kernel. A wide variety of methods for selecting the bandwidth h in the case of kernel density estimation are available, e.g. least squares cross-validation (Silverman, 1986; Li and Racine, 2006), biased cross-validation (Scott and Terrel, 1987), the bootstrap bandwidth selector method (Hall et al., 1999; Li and Racine, 2006), the regression based bandwidth selector method (Härdle, 1991; Fan et al., 1996), the double-kernel method (Devroye and Lugosi, 1989), plug-in methods (Silverman, 1986; Li and Racine, 2006; Raykar and Duraiswami, 2006), normal reference rule of thumb (Silverman, 1986; Scott, 1992) and the test graph method (Silverman, 1978, 1986).

However, since large data sets are considered, computational aspects of the selection of the bandwidth should not be neglected. Therefore, only the normal reference rule of thumb and plug-in methods can be considered. In what follows only the plug-in method will be discussed. The selection of the smoothing parameter is based on choosing h to minimize a kernel based estimate of the mean integrated squared error

$$\mathrm{MISE}(\hat{f}, f) = E\left[\int \left(\hat{f}(x) - f(x)\right)^2\right].$$

The optimal bandwidth h_{AMISE}, minimizing $\mathrm{MISE}(\hat{f}, f)$, is given by Theorem 1.

Theorem 1 (*Silverman (1986)*). *The asymptotic mean integrated squared error (AMISE) optimal bandwidth h_{AMISE} is the solution to the equation*

$$\hat{h}_{\mathrm{AMISE}} = \left(\frac{R(\kappa)}{\mu_2(\kappa)^2 R\left(f^{(2)}\right)}\right)^{1/5} n^{-1/5}, \tag{13}$$

where $R(\kappa) = \int_{\mathbb{R}} \kappa(x)^2 \mathrm{d}x$, $\mu_2(\kappa) = \int_{\mathbb{R}} x^2 \kappa(x)^2 \mathrm{d}x$, $\kappa(x)$ is the kernel function, $f^{(2)}$ is the second derivative of the density f and n is the number of data points.

However, expression (13) cannot be used directly since $R\left(f^{(2)}\right)$ depends on the second derivative of the density f. An estimator of the functional $R\left(f^{(r)}\right)$ using a kernel density derivative estimate for $f^{(r)}$ is given by

$$\hat{R}\left(f^{(r)}\right) = \frac{1}{n^2 h^{r+1}} \sum_{i=1}^{n} \sum_{j=1}^{n} \kappa^{(r)}\left(\frac{X_i - X_j}{h_{\mathrm{AMSE}}}\right). \tag{14}$$

The optimal bandwidth h_{AMSE} for estimating the density functional is given in Theorem 2, see also Marron and Wand (1992).

Theorem 2 (*Wand and Jones (1995)*). *The asymptotic mean squared error (AMSE) optimal bandwidth h_{AMSE} for (14) is given by*

$$\hat{h}_{\mathrm{AMSE}} = \left(\frac{-2\kappa^{(r)}(0)}{\mu_2(\kappa)\hat{R}\left(f^{(r+2)}\right)}\right)^{1/(r+3)} n^{-1/(r+3)}.$$

4.3. Solve-the-equation plug-in method

One of the most successful methods for bandwidth selection for kernel density estimation is the *solve-the-equation plug-in method* (Jones et al., 1996). We use the version as described by Sheather and Jones (1991). The AMISE optimal bandwidth (13) (see Theorem 1) can now be written as follows:

$$\hat{h}_{\mathrm{AMISE}} = \left(\frac{R(\kappa)}{\mu_2(\kappa)^2 \hat{R}\left(f^{(4)}, \rho(\hat{h}_{\mathrm{AMSE}})\right)}\right)^{1/5} n^{-1/5}, \tag{15}$$

where $\hat{R}\left(f^{(4)}, \rho(\hat{h}_{\mathrm{AMSE}})\right)$ is an estimate of $R(f^{(4)})$ using the pilot bandwidth $\rho(h_{\mathrm{AMSE}})$. Notice that this bandwidth for estimating the density functional (14) is different from the bandwidth h_{AMISE} used for kernel density estimation. On the basis of Theorem 2 the bandwidth h_{AMSE} for $R\left(f^{(4)}\right)$ is given by

$$\hat{h}_{\mathrm{AMSE}} = \left(\frac{-2\kappa^{(4)}(0)}{\mu_2(\kappa)\hat{R}\left(f^{(6)}, \rho(\hat{h}_{\mathrm{AMSE}})\right)}\right)^{1/7} n^{-1/7}.$$

Using (13) and substituting for n, h_{AMSE} can be written as a function of the bandwidth h_{AMISE} for kernel density estimation:

$$\hat{h}_{\mathrm{AMSE}} = \left(\frac{-2\kappa^{(4)}(0)\mu_2(\kappa)\hat{R}\left(f^{(4)}, \hat{h}_1\right)}{R(\kappa)\hat{R}\left(f^{(6)}, \hat{h}_2\right)}\right)^{1/7} \hat{h}_{\mathrm{AMISE}}^{5/7},$$

where $\hat{R}\left(f^{(4)}, \hat{h}_1\right)$ and $\hat{R}\left(f^{(6)}, \hat{h}_2\right)$ are estimates of $R\left(f^{(4)}, h_1\right)$ and $R\left(f^{(6)}, h_2\right)$ using bandwidths h_1 and h_2 respectively. The bandwidths are chosen such that they minimize the AMSE and are given by Theorem 2

$$\hat{h}_1 = \left(\frac{-2\kappa^{(4)}(0)}{\mu_2(\kappa)\hat{R}\left(f^{(6)}\right)}\right)^{1/7} n^{-1/7} \quad \text{and} \quad \hat{h}_2 = \left(\frac{-2\kappa^{(6)}(0)}{\mu_2(\kappa)\hat{R}\left(f^{(8)}\right)}\right)^{1/9} n^{-1/9}, \tag{16}$$

where $\hat{R}\left(f^{(6)}\right)$ and $\hat{R}\left(f^{(8)}\right)$ are estimates of $R\left(f^{(6)}\right)$ and $R\left(f^{(8)}\right)$.

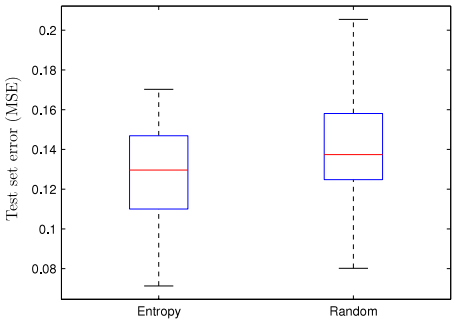

Fig. 1. Boxplot of the MSE on test (30 runs) for models estimated with entropy based and random selection of prototype vectors.

Table 1
Comparison of the mean and standard deviation of the MSE on test for the Boston Housing data set using $m = 200$ over 30 randomizations.

Selection method	Average MSE	Standard deviation MSE
Entropy	0.1293	0.0246
Random	0.1433	0.0323

This of course also reveals the problem for choosing r, the number of stages. As r increases the variability of this bandwidth selector will increase, but it becomes less biased since the dependence on the normal reference rule diminishes. Theoretical considerations by Aldershof (1991) and Park and Marron (1999) favor taking r to be at least 2, with $r = 2$ being a common choice.

If f is a normal density with variance σ^2 then, according to Wand and Jones (1995), $R\left(f^{(6)}\right)$ and $R\left(f^{(8)}\right)$ can be calculated exactly. An estimator of $R\left(f^{(r)}\right)$ will use an estimate $\hat{\sigma}^2$ of the variance. An estimator for $R\left(f^{(6)}\right)$ and $R\left(f^{(8)}\right)$ is given by

$$\hat{R}\left(f^{(6)}\right) = \frac{-15}{16\sqrt{\pi}}\hat{\sigma}^{-7} \quad \text{and} \quad \hat{R}\left(f^{(8)}\right) = \frac{105}{32\sqrt{\pi}}\hat{\sigma}^{-9}. \tag{17}$$

The main computational bottleneck is the estimation of the kernel density derivatives $R\left(f^{(r)}\right)$ which is of $\mathcal{O}\left(n^2\right)$. A method for fast evaluation of these kernel density derivatives $R\left(f^{(r)}\right)$ is proposed in Raykar and Duraiswami (2006). This method is based on Taylor expansion of the Gaussian and hence adopts the main idea of the improved fast Gauss transform (IFGT) (Yang et al., 2003).

The two-stage solve-the-equation plug-in method using a Gaussian kernel is given in Algorithm 2. A general overview of IFGT with applications to machine learning can be found in Raykar and Duraiswami (2007). The Matlab code for IFGT is freely available at http://www.umiacs.umd.edu/~vikas/Software/IFGT/IFGT_code.htm.

Algorithm 2 Solve-the-equation plug-in method

1: Compute an estimate $\hat{\sigma}$ of the standard deviation.
2: Estimate density functionals $\hat{R}\left(f^{(6)}\right)$ and $\hat{R}\left(f^{(8)}\right)$ using (17).
3: Estimate density functionals $\hat{R}\left(f^{(4)}\right)$ and $\hat{R}\left(f^{(6)}\right)$ with bandwidths $\hat{h}_1$ and $\hat{h}_2$ using (16) and (14).
4: The optimal bandwidth is the solution to the nonlinear equation (15). This equation can be solved by using e.g. the Newton–Raphson method.

Example 1. It is possible to compare the performance on test between models estimated with a random prototype vector selection versus the same models estimated with quadratic Rényi entropy based prototype vector selection. In order to compare the two performances on test we use the UCI Boston Housing data set (this data set is publicly available at http://kdd.ics.uci.edu/). We use 168 test data points and set the number of prototype vectors $m = 200$. The test is randomly selected in each run. Each model is tuned via 10-fold CV (see Section 5) for both selection criteria. Fig. 1 shows the comparison for the results based on 30 runs. Table 1 shows the average MSE and the standard deviation of the MSE. These results show that using the entropy based criterion yields a lower mean and dispersion value on test.

5. Selecting the number of prototype vectors

An important task in this framework is determining the number of PV $m \in \mathbb{N}_0$ used in the FS-LSSVM model. Existing methods (Smola and Schölkopf, 2000; Keerthi et al., 2008; Jiao et al., 2007; Zhao and Sun, 2008) select the number of PV on the basis of a greedy approach. One disadvantage of such methods is that they are time-consuming, hence making them

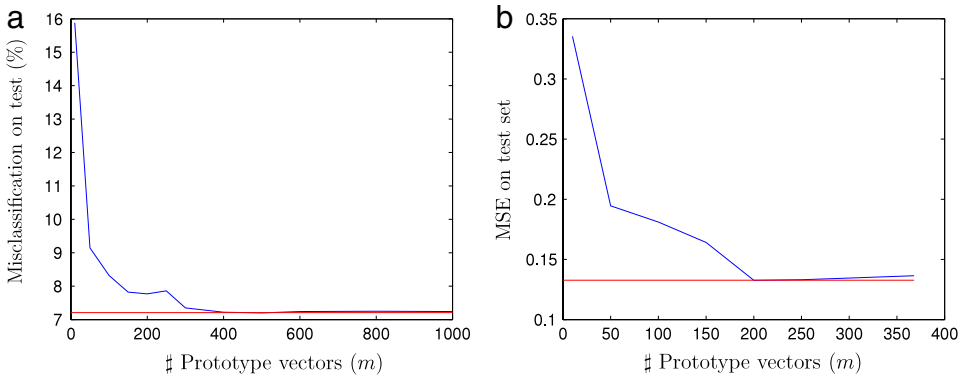

Fig. 2. Number of prototype vectors as a function of the performance on a test set. The straight line is the LS-SVM estimate and serves as a baseline comparison. (a) Spam data (binary classification); (b) Boston Housing data (regression).

infeasible for large scale data set use. In Smola and Schölkopf (2000) and Keerthi et al. (2008) an additional parameter is introduced in the subset selection process which requires tuning. This parameter determines the number of random PV to try outside the already selected subset. Keerthi et al. (2008) pointed out that there is no universal answer for setting this parameter because the answer would depend on the cost associated with the computation of the kernel function, on the number of selected PV and on the number of training points.

A second class of methods for selecting the PV is by constructing a basis in feature space (Baudat and Anouar, 2001). This method was also used by Cawley and Talbot (2002). The main idea of the method is to minimize the normalized Euclidean distance between the position of a data item in feature space and its optimal reconstruction using the set basis vectors (PV). This method is computationally heavy since it involves numerous matrix–matrix products and matrix inverses.

A third class of methods is based on matrix algebra for obtaining the number of PV. Valyon and Horváth (2004) obtain PV by bringing the kernel matrix to the reduced row echelon form (Golub and Van Loan, 1989). The total complexity of the method is given by $\frac{1}{3}m^3 + m^2(n+1) + n^2$. This method can become intractable for large data sets. Abe (2007) uses an incomplete Cholesky factorization to select the PV. The numbers of PV are controlled by an extra tuning parameter that can be determined by CV. A benefit of this method is that it does not require storage of the complete kernel matrix in the memory, which can be problematic when data sets are large, but the factorization can be done incrementally.

Our proposed method selects the PV by maximizing the quadratic Rényi entropy (12). This algorithm requires the entropy kernel bandwidth(s); see Section 4.2. The computational complexity associated with determining d bandwidths is roughly nd (Raykar and Duraiswami, 2006), where d is the number of dimensions. One only has to calculate the kernel matrix associated with the PV, which has a computational cost of m^2. The entropy value is then almost given by the sum of all elements of this kernel matrix. Each time the set of PV is altered (by using the active selection strategy) the entropy value of the new set can be simply obtained by updating the previous value. Hence, in this strategy the kernel matrix associated with the PV has to be calculated only once and can then be removed from the memory.

In theory we could gradually increase the number of PV until $m = n$. Naturally it would be too computationally expensive, but it gives an insight into how an increasing amount of prototype vectors will influence the performance (on test) of a classifier or regression estimate. Consider the Spam data set (1533 data points are randomly selected as a test set) and the Boston Housing data set (168 data points are randomly selected as a test set). In this example the number of PV m is gradually increased and the performance on test data is determined for each of the chosen prototype vector sizes. Fig. 2 shows the number of prototype vectors of the FS-LSSVM as a function of the performance on test data for both data sets. The straight line is the LS-SVM estimate on the same data sets and serves as a baseline comparison. These results indicate that only a percentage of the total number of data points is required to obtain a performance on test which is equal to the LS-SVM estimate.

In practice, however, due to time constraints and computational burden, it is impossible to gradually increase the number of prototype vectors until e.g. $m = n$. Therefore we propose a simple heuristic in order to obtain a rough estimate of the number of prototype vectors to be used in the FS-LSSVM model. Choose k different values for the number of prototype vectors, say $k = 5$. Determine k FS-LSSVM models and calculate the performance on test of each model. Choose as the final m the number of prototype vectors of the model which has the best performance on test data. Also note that in this approach it is not always the sparsest model that is selected, but it reduces computation time.

In the FS-LSSVM framework the number of PV m is a tuning parameter, but the choice is not very crucial as can be seen in Fig. 2. This is however an inherent drawback of the method in comparison to SVM. In the SVM framework the number of SV follows from solving a convex QP problem.

6. Tuning parameter selection

Tuning parameter selection methods can be divided into three broad classes:

- Cross-validation (Burman, 1989) and bootstrap (Davison and Hinkley, 2003) methods.

- Plug-in methods (Härdle, 1989). The bias of an estimate of an unknown real-valued smooth function is approximated by a Taylor expansion. A pilot estimate of the unknown function is plugged in to derive an estimate of the bias and hence an estimate of the MISE.
- Complexity criteria: Mallows' C_p (Mallows, 1973), Akaike's information criterion (Akaike, 1973), the Bayes information criterion (Schwartz, 1979) and the Vapnik–Chervonenkis dimension (Vapnik, 1999). These criteria estimate the generalization error via an estimate of the complexity term and then add it to the training error.

In the context of LS-SVM and sparse LS-SVM, fast leave-one-out cross-validation algorithms based on one complete matrix inversion are found in Ying and Keong (2004) and Cawley and Talbot (2004) respectively. An algorithm for v-fold cross-validation was proposed by An et al. (2007). In the next paragraph we propose a fast algorithm for v-fold CV based on a simple updating scheme for FS-LSSVM.

6.1. Fast v-fold cross-validation

v-fold cross-validation would require computation of the extended feature matrix $\hat{\Phi}_e$ (see (8)) in each fold. The solution can then be calculated from (7). This process has to be repeated v times. If large data sets are considered, this can be computationally demanding.

In what follows a faster v-fold cross-validation algorithm for fixed-size LS-SVM (FS-LSSVM) is presented. The algorithm is based on the fact that the extended feature matrix $\hat{\Phi}_e$ has to be calculated only once instead of v times. Unlike those of An et al. (2007) and Ying and Keong (2004), the algorithm here is not based on one full matrix inverse because of the complexity $\mathcal{O}\left((m+1)^3\right)$ of this operation.

Given a data set $\mathcal{D}_n$, at each fold of the cross-validation, the v-th group is left out for validation and the remainder is for training. The extended feature matrix $\hat{\Phi}_e \in \mathbb{R}^{n\times(m+1)}$ is given by

$$\hat{\Phi}_e = \left(\begin{array}{ccc|c} \hat{\varphi}_1(X_{tr,1}) & \cdots & \hat{\varphi}_m(X_{tr,1}) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \hat{\varphi}_1(X_{tr,n_{tr}}) & \cdots & \hat{\varphi}_m(X_{tr,n_{tr}}) & 1 \\ \hline \hat{\varphi}_1(X_{val,1}) & \cdots & \hat{\varphi}_m(X_{val,1}) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \hat{\varphi}_1(X_{val,n_{val}}) & \cdots & \hat{\varphi}_m(X_{val,n_{val}}) & 1 \end{array}\right) = \left(\begin{array}{c|c} \hat{\Phi}_{tr} & 1_{tr} \\ \hline \hat{\Phi}_{val} & 1_{val} \end{array}\right), \tag{18}$$

where $1_{tr} = (1,\ldots,1)^T$, $1_{val} = (1,\ldots,1)^T$, $X_{tr,j}$ and $X_{val,j}$ denote the jth elements of the training data and the validation data respectively. n_{tr} and n_{val} are the numbers of data points in the training data set and the validation set respectively such that $n_{tr} + n_{val} = n$. Also we set $A = \hat{\Phi}_e^T\hat{\Phi}_e + \dfrac{I_{m+1}}{\gamma}$ and $c = \hat{\Phi}_e^T Y$. At each fold of the cross-validation, the v-th group is left out for validation and the remainder is for training. So in the v-th iteration

$$A_v \begin{pmatrix} \tilde{w} \\ b \end{pmatrix} = c_v, \tag{19}$$

where A_v is a square matrix with the same dimension as A but modified from A by taking only the training data to build it. The motivation is to get A_v from A with a few simple steps instead of computing A_v in each fold from scratch. Using (7) and (18), the following holds:

$$\begin{aligned} \hat{\Phi}_e^T\hat{\Phi}_e + \frac{I_{m+1}}{\gamma} &= \left(\begin{array}{c|c} \hat{\Phi}_{tr}^T & \hat{\Phi}_{val}^T \\ \hline 1_{tr}^T & 1_{val}^T \end{array}\right)\left(\begin{array}{c|c} \hat{\Phi}_{tr} & 1_{tr} \\ \hline \hat{\Phi}_{val} & 1_{val} \end{array}\right) + \frac{I_{m+1}}{\gamma} \\ &= \left(\begin{array}{c|c} \hat{\Phi}_{tr}^T\hat{\Phi}_{tr} + \frac{I_m}{\gamma} & \hat{\Phi}_{tr}^T 1_{tr} \\ \hline 1_{tr}^T\hat{\Phi}_{tr} & 1_{tr}^T 1_{tr} + \frac{1}{\gamma} \end{array}\right) + \left(\begin{array}{c|c} \hat{\Phi}_{val}^T\hat{\Phi}_{val} & \hat{\Phi}_{val}^T 1_{val} \\ \hline 1_{val}^T\hat{\Phi}_{val} & 1_{val}^T 1_{val} \end{array}\right) \\ &= \left(\hat{\Phi}_{e,tr}^T\hat{\Phi}_{e,tr} + \frac{I_{m+1}}{\gamma}\right) + \left(\begin{array}{c} \hat{\Phi}_{val}^T \\ \hline 1_{val}^T \end{array}\right)\left(\hat{\Phi}_{val} \;\middle|\; 1_{val}\right), \end{aligned}$$

where $\hat{\Phi}_{e,tr} = \left(\hat{\Phi}_{tr} \;\middle|\; 1_{tr}\right)$ is the extended feature matrix of the training data and hence

$$\hat{\Phi}_{e,tr}^T\hat{\Phi}_{e,tr} + \frac{I_{m+1}}{\gamma} = \left(\hat{\Phi}_e^T\hat{\Phi}_e + \frac{I_{m+1}}{\gamma}\right) - \left(\begin{array}{c} \hat{\Phi}_{val}^T \\ \hline 1_{val}^T \end{array}\right)\left(\hat{\Phi}_{val} \;\middle|\; 1_{val}\right).$$

This results in

$$A_v = A - \left(\begin{array}{c} \hat{\Phi}_{val}^T \\ \hline 1_{val}^T \end{array}\right)\left(\hat{\Phi}_{val} \;\middle|\; 1_{val}\right). \tag{20}$$

The most time-consuming step in (20) is the calculation of matrix A. However, this calculation needs to be performed only once. The second term in (20) does not require complete recalculation since $\hat{\Phi}_{val}$ can be extracted from $\hat{\Phi}_e$; see (18). A similar

Table 2
Summary of different implementations of CV.

Implementation	♯ folds
Cawley and Talbot (2004)	n
An et al. (2007)	>20
Proposed	3–20

result holds for c_v:

$$\begin{aligned} c &= \hat{\Phi}_e^T Y \\ &= \left(\begin{array}{c|c} \hat{\Phi}_{tr}^T & \hat{\Phi}_{val}^T \\ \hline 1_{tr}^T & 1_{val}^T \end{array}\right)\left(\begin{array}{c} Y_{tr} \\ \hline Y_{val} \end{array}\right) \\ &= \left(\frac{\hat{\Phi}_{tr}^T}{1_{tr}^T}\right) Y_{tr} + \left(\frac{\hat{\Phi}_{val}^T}{1_{val}^T}\right) Y_{val} \\ &= c_v + \left(\frac{\hat{\Phi}_{val}^T}{1_{val}^T}\right) Y_{val}, \end{aligned}$$

and thus

$$c_v = c - \left(\frac{\hat{\Phi}_{val}^T}{1_{val}^T}\right) Y_{val}. \tag{21}$$

In each fold one has to solve the linear system (19). This leads to Algorithm 3 for fast v-fold cross-validation.

Algorithm 3 Fast v-fold cross-validation for FS-LSSVM

1: Calculate the matrix $\hat{\Phi}_e$ (for all data using the Nyström approximation (5)) and $A = \hat{\Phi}_e^T \hat{\Phi}_e + \frac{I_{m+1}}{\gamma}$.
2: Split data randomly into v disjoint sets of nearly equal size.
3: Compute in each fold A_v and c_v using (20) and (21) respectively.
4: Solve the linear system (19).
5: Compute the residuals in each fold: $\hat{e}_{v,i} = Y_{val,i} - (\hat{\hat{w}}^T \hat{\varphi}(X_{val,i}) + \hat{b})$ in each fold.
6: Choose an appropriate loss function for assessing performance (e.g. MSE).

We conclude with a final remark regarding the proposed CV method. The literature describes other fast implementations for CV; see e.g. the methods of Cawley and Talbot (2004) and Ying and Keong (2004) for leave-one-out CV (LOO-CV) and An et al. (2007) for v-fold CV. The method of choice greatly depends on the number of folds used for CV. Table 2 gives an overview of which method can best be used with different kinds of folds.

6.2. Tuning parameter selection for very large data sets

The fast v-fold CV algorithm for FS-LSSVM (Algorithm 3) is based on the fact that the extended feature matrix (8) can fit into the memory. In order to overcome this problem we propose to calculate the extended feature matrix $\hat{\Phi}_e$ in a number of S blocks. In this way, the extended feature matrix $\hat{\Phi}_e$ does not need to be stored completely in the memory. Let l_s, with $s = 1, \ldots, S$, denote the length of the s-th block and also $\sum_{s=1}^{S} l_s = n$. The matrix $\hat{\Phi}_e$ can be written as follows:

$$\hat{\Phi}_e = \left(\begin{array}{c} \hat{\Phi}_{e,[1]} \\ \hline \vdots \\ \hline \hat{\Phi}_{e,[S]} \end{array}\right),$$

with $\hat{\Phi}_{e,[s]} \in \mathbb{R}^{l_s \times (m+1)}$ and the vector Y given by

$$Y = \left(\begin{array}{c} Y_{[1]} \\ \hline \vdots \\ \hline Y_{[S]} \end{array}\right),$$

with $Y_{[s]} \in \mathbb{R}^{l_s}$. The matrix $\hat{\Phi}_{e,[s]}^T \hat{\Phi}_{e,[s]}$ and vector $\hat{\Phi}_{e,[s]}^T Y_{[s]}$ can be calculated in an updating scheme and stored in the memory since the sizes of these are $(m+1) \times (m+1)$ and $(m+1) \times 1$ respectively.

Also because of the high computational burden, we can validate using a holdout estimate (Devroye et al., 1996). The data sequence $\mathcal{D}_n$ is split into a training sequence $\mathcal{D}_{tr} = \{(X_1, Y_1), \ldots, (X_t, Y_t)\}$ and a fixed validation sequence

$\mathcal{D}_{val} = \{(X_{t+1}, Y_{t+1}), \ldots, (X_{t+l}, Y_{t+l})\}$, where $t + l = n$. Algorithm 4 summarizes the above idea. This idea of calculating $\hat{\Phi}_e$ in blocks can also be extended to v-fold CV.

Algorithm 4 Holdout estimate for very large scale FS-LSSVM

1: Choose a fixed validation sequence $\mathcal{D}_{val}$.
2: Divide the remaining data set $\mathcal{D}_{tr}$ into approximately S equal blocks such that $\hat{\Phi}_{e,[s]}$ with $s = 1, \ldots, S$, calculated using (5), can fit into the memory.
3: Initialize matrix $A_v \in \mathbb{R}^{(m+1)\times(m+1)}$ and vector $c_v \in \mathbb{R}^{m+1}$.
4: **for** $s = 1$ to S **do**
5: Calculate matrix $\hat{\Phi}_{e,[s]}$ for the s-th block using the Nyström approximation (5)
6: $A_v \leftarrow A_v + \hat{\Phi}^T_{e,[s]}\hat{\Phi}_{e,[s]}$
7: $c_v \leftarrow c_v + \hat{\Phi}^T_{e,[s]}Y_{[s]}$
8: **end for**
9: Set $A_v \leftarrow A_v + \frac{I_{m+1}}{\gamma}$.
10: Solve the linear system (19).
11: Compute the residuals $\hat{e}$ on the fixed validation sequence $\mathcal{D}_{val}$.
12: Compute the holdout estimate.

6.3. Optimization strategy

Good tuning parameters are found by minimizing the cross-validation (CV) score function. This is a difficult task since the CV surface is non-smooth. A typical method for selecting these tuning parameters is defining a grid over the parameters of interest and performing a v-fold CV for each of the grid values. Although this method is quite common, there are some major disadvantages (Bennett et al., 2006; Huang et al., 2007) (these disadvantages are outlined in the introduction of this paper). In order to overcome these drawbacks we use a methodology consisting of two steps: first, determine good initial start values by means of a state of the art global optimization technique and, secondly, perform a fine-tuning derivative-free simplex search (Nelder and Mead, 1965; Lagarias et al., 1998) using the previous result as the start value.

To determine good initial starting values we use the method of coupled simulated annealing (CSA) with variance control ((Xavier de Souza et al., 2006, in press); see also the Appendix), whose working principle was inspired by the effect of coupling in coupled local minimizers (CLM) (Suykens et al., 2001) as compared to the uncoupled case, i.e. multi-start based methods. CLM and CSA have already proven to be more effective than multi-start gradient descent optimization (Suykens et al., 2001, 2003). Another advantage of CSA is that it uses the acceptance temperature to control the variance of the acceptance probabilities with a control scheme. This leads to an improved optimization efficiency because it reduces the sensitivity of the algorithm to the initialization parameters while guiding the optimization process to quasi-optimal runs. This initial result is then used as a starting value for a derivative-free simplex search. This extra step is a fine-tuning procedure resulting in more optimal tuning parameters and hence better performance.

To conclude this section, we summarize the complete FS-LSSVM method in Algorithm 5.

Algorithm 5 Summary of the optimized FS-LSSVM method

1: Given a training set defined as $\mathcal{D}_n = \{(X_k, Y_k) : X_k \in \mathbb{R}^d, Y_k \in \mathbb{R}; k = 1, \ldots, n\}$.
2: Determine the kernel bandwidth or bandwidth matrix for entropy selection according to Algorithm 2.
3: Given the number of prototype vectors (PV), begin the active PV selection according to Algorithm 1.
4: Determine the learning parameters by means of minimizing the CV cost function.
5: **if** the extended feature matrix (8) can be stored completely in the memory **then**
6: the value of the cost function is evaluated using Algorithm 3.
7: **else**
8: use Algorithm 4 to evaluate the cost function.
9: **end if**
10: Given the optimal learning parameters, obtain the FS-LSSVM parameters $\hat{\vec{w}}$ and $\hat{b}$ by solving the linear system (7).

7. Computational complexity analysis and numerical experiments on fast v-fold CV for FS-LSSVM

In this section, we discuss the complexity of the proposed fast v-fold CV and present some experimental results compared to a simple implementation of v-fold CV on a collection of data sets from the UCI benchmark repository.

7.1. Computational complexity analysis

The simple implementation of v-fold CV computes the extended feature matrix (8) for each split of data and uses no updating scheme. This is computationally expensive when v is large (e.g. leave-one-out CV). Note that the complexity of

Table 3
(Regression.) The average run time (seconds) over 100 runs of the proposed algorithm compared with the simple implementation for various folds and prototype vectors on the *Concrete Compressive Strength* data set for one pair of fixed tuning parameters. The standard deviation is given within parentheses. The numbers of prototype vectors were arbitrarily chosen.

Number of folds	5	10	20	30	40	50
(a) 50 prototype vectors						
Simple [s]	0.066 (0.0080)	0.14 (0.0038)	0.29 (0.0072)	0.44 (0.0094)	0.57 (0.0085)	0.71 (0.0079)
Optimized [s]	0.009 (0.0013)	0.012 (0.0013)	0.015 (0.0003)	0.018 (0.0004)	0.019 (0.0004)	0.021 (0.0004)
(b) 100 prototype vectors						
Simple [s]	0.19 (0.006)	0.38 (0.01)	0.75 (0.014)	1.15 (0.016)	1.49 (0.016)	1.86 (0.016)
Optimized [s]	0.031 (0.006)	0.033 (0.0006)	0.035 (0.0007)	0.039 (0.0001)	0.052 (0.0006)	0.058 (0.0007)
(c) 400 prototype vectors						
Simple [s]	3.60 (0.04)	7.27 (0.11)	14.51 (0.10)	21.62 (0.11)	31.07 (0.12)	39.12 (0.12)
Optimized [s]	0.40 (0.003)	0.45 (0.002)	0.53 (0.002)	0.57 (0.002)	0.65 (0.005)	0.76 (0.005)

solving a linear system with dimension $m+1$ is $\frac{1}{3}(m+1)^3$ (Press et al., 1993) and the complexity of calculating the Nyström approximation (with eigenvalue decomposition of the kernel matrix of size m) is $m^3 + m^2 n$. The total complexity of the proposed method is then given by the sum of the complexities of:

- v times solving a linear system of dimensions $m + 1$;
- calculating the Nyström approximation + eigenvalue decomposition of the kernel matrix of size m once;
- forming the matrix product $\hat{\Phi}_e^T \hat{\Phi}_e$ once.

Hence, the resulting complexity of the proposed method, neglecting lower order terms, is given by $(\frac{v}{3}+1)m^3+2nm^2$. In a similar way, the resulting complexity of the simple method is given by $\frac{4}{3}vm^3+(2v-2)nm^2$. The computational complexity of the proposed method is lower than that of the simple method for $v \geq 2$. Keeping m fixed, it is clear that the number of folds has a small influence on the proposed method, resulting in a small time increase with increasing number of folds v. This is in contrast to the simple method case where the computational complexity increases heavily with increasing number of folds. On the other hand, consider the number of folds v fixed and m variable; a larger time increase is expected with the simple method than with the proposed method, i.e. the determining factor is m^3.

7.2. Numerical experiments

All the experiments that follow are performed on a PC machine with Intel Core 2 Quad (Q6600) CPU and 3.2 GB RAM under Matlab R2008a for Windows. During the simulations the RBF kernel is used unless mentioned otherwise. To compare the efficiency of the proposed algorithm, an experimental procedure, adopted from Mika et al. (1999) and Rätsch et al. (2001), is used where 100 different random training and test splits are defined.

Table 3 verifies the computational complexity of the algorithm on the *Concrete Compressive Strength* data set (regression; the data set is publicly available at http://kdd.ics.uci.edu/ and has 1030 instances and 8 attributes) for various numbers of prototype vectors (the numbers of prototype vectors are chosen arbitrarily). From the experiments it can be seen that the computation time is not very sensitive to the number of folds, while this influence is larger in the simple implementation. Both algorithms experience an increasing complexity with increasing number of prototype vectors. This increase is stronger with the simple implementation. The latter has also a larger standard deviation. The results of Table 3 are visualized in Fig. 3 showing the number of folds as a function of computational time for various numbers of prototype vectors in the regression case.

Table 4 verifies the complexity of the algorithm on the *Magic Gamma Telescope* data set (binary classification; this data set is publicly available at http://kdd.ics.uci.edu/ and has 19,020 instances and 10 attributes) for various numbers of prototype vectors (also chosen arbitrarily) and folds. The conclusions are the same as in the regression case. The results of Table 4 are visualized in Fig. 4 showing the number of folds as a function of computational time for various numbers of prototype vectors in the classification case.

8. Classification and regression results

In this section, we report the application of FS-LSSVM on benchmark data sets (Blake and Merz, 1998) of which a brief description is included in Section 8.1. The performance of the FS-LSSVM is compared with those of standard SVM and

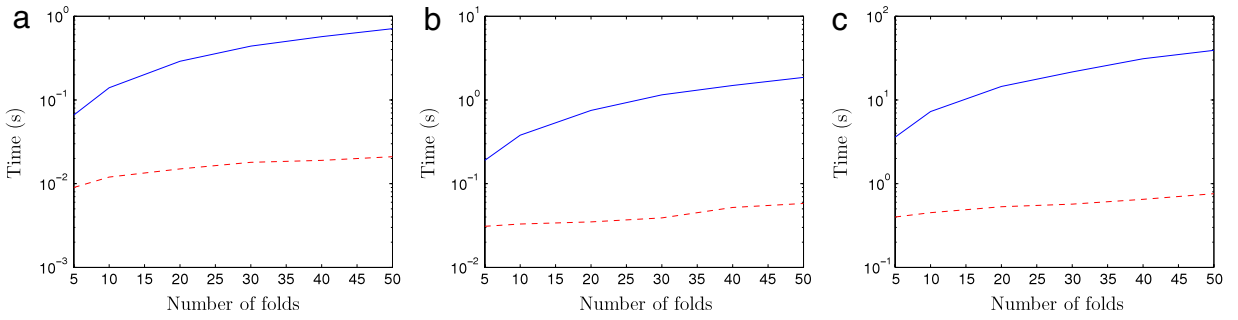

Fig. 3. Number of folds as a function of computation time (in seconds) for various numbers of prototype vectors (50, 100, 400) on the Concrete Compressive Strength data set. The full line represents the simple implementation and the dashed line the proposed method.

Table 4
(Binary classification.) The average run time (seconds) over 100 runs of the proposed algorithm compared with the simple implementation for various folds and prototype vectors on the *Magic Gamma Telescope* data set for one pair of fixed tuning parameters. The standard deviation is given within parentheses. The numbers of prototype vectors were arbitrarily chosen.

Number of folds	5	10	20	30	40	50
(a) 50 prototype vectors						
Simple [s]	0.71 (0.04)	1.51 (0.02)	2.98 (0.026)	4.47 (0.03)	6.05 (0.04)	7.60 (0.039)
Optimized [s]	0.15 (0.01)	0.16 (0.006)	0.16 (0.004)	0.17 (0.005)	0.19 (0.005)	0.19 (0.005)
(b) 300 prototype vectors						
Simple [s]	6.27 (0.08)	12.80 (0.24)	25.49 (0.27)	38.08 (0.43)	50.94 (0.26)	64.43 (0.42)
Optimized [s]	1.52 (0.05)	1.57 (0.02)	1.60 (0.05)	1.64 (0.05)	1.66 (0.04)	1.83 (0.03)
(c) 700 prototype vectors						
Simple [s]	29.01 (0.20)	58.56 (0.12)	117.12 (0.16)	179.44 (0.17)	231.59 (0.18)	290.36 (0.71)
Optimized [s]	5.95 (0.046)	6.06 (0.09)	6.34 (0.025)	6.60 (0.024)	6.94 (0.032)	7.11 (0.026)

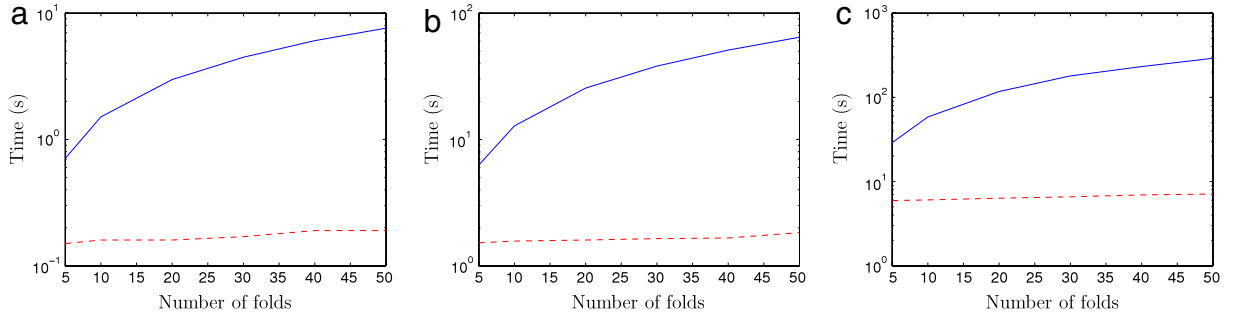

Fig. 4. Number of folds as a function of computation time (in seconds) for various numbers of prototype vectors (50, 300, 700) on the Magic Gamma Telescope data set. The full line represents the simple implementation and the dashed line the proposed method.

ν-SVM (LIBSVM software (Chang and Lin, 2001)). Although this paper focuses on large scale data sets, smaller data sets will also be included for completeness. The randomized test set results are discussed in Sections 8.3 and 8.4 (classification) and Section 8.5 (regression).

8.1. Description of the data sets

All the data sets have been obtained from the publicly accessible UCI benchmark repository (Blake and Merz, 1998). As a preprocessing step, all records containing unknown values are removed from consideration. All given inputs are normalized to zero mean and unit variance.

Table 5
Characteristics of the binary classification UCI data sets, where N_{CV} is the number of data points used in the CV based tuning procedure, n_{test} is the number of observations in the test set and n is the total data set size. The numbers of numerical and categorial attributes are denoted by d_{num} and d_{cat} respectively; d is the total number of attributes.

	pid	spa	mgt	adu	ftc
n_{CV}	512	3068	13,000	33,000	531,012
n_{test}	256	1533	6,020	12,222	50,000
n	768	4601	19,020	45,222	581,012
d_{num}	8	57	11	6	10
d_{cat}	0	0	0	8	44
d	8	57	11	14	54

Table 6
Characteristics of the multi-class classification UCI data sets. Row M denotes the number of classes for each data set encoded by L_{MOC} and L_{1vs1} bits for minimum output coding (MOC) and one-versus-one output coding (1vs1) respectively.

	opt	lan	pen	let	shu
n_{CV}	3750	4435	7,328	13,667	43,500
n_{test}	1870	2000	3,664	6,333	14,500
n	5620	6435	10,992	20,000	58,000
d_{num}	64	36	16	16	9
d_{cat}	0	0	0	0	0
d	64	36	16	16	9
M	10	7	10	26	7
L_{MOC}	4	3	4	5	3
L_{1vs1}	45	21	45	325	21

Table 7
Characteristics of the regression UCI data sets.

	bho	ccs
n_{CV}	338	687
n_{test}	168	343
n	506	1030
d_{num}	14	9
d_{cat}	0	0
d	14	9

8.1.1. Classification

The following binary data sets were downloaded from http://kdd.ics.uci.edu/: Magic Gamma Telescope (mgt), Pima Indians Diabetes (pid), Adult (adu), Spam Database (spa) and Forest Covertype (ftc) data sets. The main characteristics of these data sets are summarized in Table 5. A modification was made from this last data set (Collobert et al., 2002; Liu et al., 2004): the seven-class classification problem was transformed into a binary classification problem where the goal is to separate class 2 from the other six classes.

The following multi-class data sets were used: Letter Recognition (let), Optical Recognition (opt), Pen Based Recognition (pen), Statlog Landsat Satellite (lan) and Statlog Shuttle (shu) data sets. The main characteristics of these data sets are summarized in Table 6.

8.1.2. Regression

The following data sets for regression were also downloaded from the UCI benchmark data sets: Boston Housing (bho) and Concrete Compressive Strength (ccs). The main characteristics of these data sets are given in Table 7.

8.2. Description of the reference algorithms

The test performance of the FS-LSSVM classifier/regression model is compared with the performances of SVM and ν-SVM (Schölkopf et al., 2000), both implemented in the LIBSVM software. In the case of ν-SVM the parameter $\nu \in\ ]0, 0.8]$ is also considered as a tuning parameter. The three methods use a cache size of 1 GB and the stopping criterion is set to 10^{-3}. Shrinking is applied in the SVM case. For classification, the default classifier or majority rule (Maj. rule) is included as a baseline in the comparison tables. The majority rule (in %) is given by the largest number of data points belonging to a class divided by total number of data points (of all classes) multiplied by a hundred. All comparisons are made on the same 10 randomizations.

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 is reserved for training and/or cross-validation. For each algorithm, the average test set performances and sample

Table 8
Comparison of the 10-times-randomized **test set** performances (as percentages) and standard deviations (within parentheses) of FS-LSSVM (linear and RBF kernel) with the performances of C-SVC, ν-SVC and majority rule classifiers on five binary domains. n_{test} is the number of observations in the test set and d is the total number of attributes. Also the number of prototype vectors (PV) for FS-LSSVM and number of support vectors (SV) used by the algorithms are reported. The number of prototype vectors of FS-LSSVM are determined by the heuristic described in Section 5. The values with an asterisk only denote the performances of the C-SVC and ν-SVC for fixed tuning parameter(s). No cross-validation was performed because of the computational burden.

	pid	spa	mgt	adu	ftc
n_{test}	256	1533	6020	12,222	50,000
d	8	57	11	14	54
♯ PV FS-LSSVM	150	200	1000	500	500
♯ SV C-SVC	290	800	7000	11,085	185,000
♯ SV ν-SVC	331	1525	7252	12,205	165,205
RBF FS-LSSVM	76.7(3.43)	92.5(0.67)	86.6(0.51)	85.21(0.21)	81.8(0.52)
Lin FS-LSSVM	77.6(0.78)	90.9(0.75)	77.8(0.23)	83.9(0.17)	75.61(0.35)
RBF C-SVC	75.1(3.31)	92.6(0.76)	85.6(1.46)	84.81(0.20)	81.5(*)
Lin C-SVC	76.1(1.76)	91.9(0.82)	77.3(0.53)	83.5(0.28)	75.24(*)
RBF ν-SVC	75.8(3.34)	88.7(0.73)	84.2(1.42)	83.9(0.23)	81.6(*)
Maj. rule	64.8(1.46)	60.6(0.58)	65.8(0.28)	83.4(0.1)	51.23(0.20)

Table 9
Comparison of the average computation times in seconds for the FS-LSSVM, C-SVC and ν-SVC on five binary classification problems. The standard deviation is shown within parentheses. The values with an asterisk only denote the training time for a fixed pair of tuning parameters for C-SVC and ν-SVC. No cross-validation was performed because of the computational burden.

Av. time (s)	pid	spa	mgt	adu	ftc
RBF FS-LSSVM	30.1(1.9)	249(16)	9985(112)	7344(295)	122290(989)
Lin FS-LSSVM	19.8(0.5)	72(3.8)	1298(13)	1404(47)	5615(72)
RBF C-SVC	24.8(3.1)	1010(53)	20603(396)	139730(5556)	58962(*)
Lin C-SVC	18(0.65)	785(22)	13901(189)	130590(4771)	53478(*)
RBF ν-SVC	30.3(2.3)	1372(43)	35299(357)	139927(3578)	55178(*)

standard deviations on 10 randomizations are reported. Also the mean total time and corresponding standard deviation are mentioned. The total time of the algorithms is made up as follows. (i) 10-fold CV using the optimization strategy described in Section 6.3. The total number of function evaluations is set to 160 (90 for CSA and 70 for simplex search). In the case of ν-SVM the parameter ν is also considered as a tuning parameter. We have used five multiple starters for the CSA algorithm. (ii) Training with optimal tuning parameters. (iii) Evaluation on a test set. For FS-LSSVM we set the parameter $k = 5$.

8.3. Performance of binary FS-LSSVM classifiers

In the following subsections, the results are presented and discussed using the optimization strategy, outlined in Section 6.3 for selecting the tuning parameters on the seven UCI binary benchmark data sets described above. As kernel types, RBF and linear (Lin) kernels were used. Performances of FS-LSSVM, SVM (C-SVC) and ν-SVC are reported. The following experimental setup is used: each binary classifier is designed on 2/3 (random selection) of the data using 10-fold CV, while the remaining 1/3 are put aside for testing. The test set performances on the data sets are reported in Table 8. Table 9 gives the average computation time (in s) and standard deviation for both algorithms.

The FS-LSSVM classifier with RBF kernel (RBF FS-LSSVM) achieves the best average test performance on three of the five benchmark domains, while its accuracy is comparable to that of RBF SVM (C-SVC). On all binary classification data sets, ν-SVC has a slightly poorer performance compared to FS-LSSVM and C-SVC. Comparison of the average test set performance achieved by the RBF kernel with the average test set performance of the linear kernel illustrates that most domains are weakly nonlinear (Holte, 1993), except for the Magic Gamma Telescope data set. Due to the high training times for SVM (C-SVC and ν-SVC) in the case of the Forest Covertype data set, it is practically impossible to perform 10-fold CV. Therefore, the values in Tables 8 and 9 with an asterisk only denote the training times for a fixed pair of tuning parameters for SVM. No cross-validation was performed because of the computational burden. Notice also that the FS-LSSVM models are sparser than the RBF SVM (C-SVC and ν-SVC) models while resulting in equal or better performance on test.

8.4. Performance of multi-class FS-LSSVM classifiers

Each multi-class problem is decomposed into a set of binary classification problems using minimum output coding (MOC) and one-versus-one (1vs1) output coding for FS-LSSVM and one-versus-one (1vs1) output coding for SVM (C-SVC and ν-SVC). The same kernel types as in the binary classification problem are considered. Performances of FS-LSSVM and SVM (C-SVC and ν-SVC) are reported. We used the same experimental setup as for binary classification. The test set performances

Table 10
Comparison of the 10-times-randomized **test set** performances (as percentages) and standard deviations (within parentheses) of FS-LSSVM (RBF kernel) with the performances of C-SVC, ν-SVC and majority rule classifier on five multi-class domains using MOC and 1vs1 output coding. n_{test} is the number of observations in the test set and d is the total number of attributes. Also the numbers of prototype vectors (PV) and numbers of support vectors (SV) used by the algorithms are reported. The numbers of prototype vectors of FS-LSSVM are determined by the heuristic described in Section 5.

	opt	lan	pen	let	shu
n_{test}	1870	2000	3664	6333	14,500
d	64	36	16	16	9
♯ PV FS-LSSVM	420	330	250	1500	175
♯ SV C-SVC	3750	1876	1178	8830	559
♯ SV ν-SVC	2810	2518	4051	11,359	521
RBF FS-LSSVM (MOC)	96.87(0.70)	91.83(0.43)	99.44(0.17)	89.14(0.22)	99.87(0.03)
RBF FS-LSSVM (1vs1)	98.14(0.10)	91.93(0.3)	99.57(0.10)	95.65(0.17)	99.84(0.03)
Lin FS-LSSVM (1vs1)	97.18(0.35)	85.71(0.77)	96.67(0.35)	84.87(0.49)	96.82(0.18)
Lin FS-LSSVM (MOC)	78.62(0.32)	74.35(0.26)	68.21(0.36)	18.20(0.46)	84.78(0.33)
RBF C-SVC (1vs1)	97.73(0.14)	92.14(0.45)	99.51(0.13)	95.67(0.19)	99.86(0.03)
Lin C-SVC (1vs1)	97.21(0.26)	86.12(0.79)	97.52(0.62)	84.96(0.56)	97.02(0.19)
RBF ν-SVC (1vs1)	95.3(0.12)	88.3(0.31)	95.96(0.16)	93.18(0.21)	99.34(0.03)
Maj. rule	10.45(0.12)	23.61(0.16)	10.53(0.07)	4.10(0.14)	78.81(0.04)

Table 11
Comparison of the average computation time in seconds for FS-LSSVM, C-SVM and ν-SVC on five multi-class classification problems. The standard deviation is shown within parentheses.

Av. time (s)	opt	lan	pen	let	shu
RBF FS-LSSVM (MOC)	4892(162)	2159(83)	2221(110)	105930(2132)	5908(272)
RBF FS-LSSVM (1vs1)	623(36)	739(17)	514(42)	10380(897)	2734(82)
Lin FS-LSSVM (1vs1)	282(19)	153(6)	156(4)	2792(11)	501(8)
Lin FS-LSSVM (MOC)	942(6)	409(13)	279(10)	44457(1503)	645(31)
RBF C-SVC (1vs1)	11371(573)	6612(347)	11215(520)	59102(2412)	52724(3619)
Lin C-SVC (1vs1)	474(1)	1739(48)	880(16)	11203(467)	50174(2954)
RBF ν-SVC (1vs1)	7963(178)	8229(304)	16589(453)	79040(2354)	50478(2879)

on the data sets are reported in Table 10. Table 11 gives the average computation time (in seconds) and standard deviation for the three algorithms. Test performances as well as the accuracies of the multi-class FS-LSSVM and multi-class SVM (C-SVC and ν-SVC) are similar. From Table 10 it is clear that there is a difference between the encoding schemes. In general, 1vs1 output coding results in better performances on test than minimum output coding (MOC). This can be especially seen from the Lin FS-LSSVM result on the Letter Recognition data set. Notice that the FS-LSSVM models are again sparser than the two kinds of SVM models.

8.5. Performance of FS-LSSVM for regression

As in the classification problems, all the inputs are normalized to zero mean and unit variance. Each regressor is designed on 2/3 (random selection) of the data using 10-fold CV, while the remaining 1/3 is kept for assessing performance on test data. The test performances of the data sets are given in Table 12. Table 13 reports the average computation time (in seconds) and standard deviations for both algorithms. In each of the regression examples the RBF kernel is used. From these results it can be seen that our algorithm has better performances and smaller standard deviations than ε-SVR and ν-SVR. FS-LSSVM results in a sparser model for both data sets compared to ε-SVR. Also notice that both kinds of SVR methods have similar performances although ν-SVR results in sparser models.

9. Conclusions

In this paper an optimized fixed-size least squares support vector machines (FS-LSSVM) version was proposed, suitable for mining large scale data sets. First, a computationally attractive method for bandwidth selection was used to determine the smoothing parameter for entropy estimation. Secondly, a fast v-fold cross-validation algorithm for FS-LSSVM was presented, combined with state of the art optimization techniques (CSA + simplex search) in order to find optimal tuning parameters. The resulting FS-LSSVM were experimentally compared with two kinds of SVM (LIBSVM software) for a number of classifications as well as regression data sets, with promising performances and timing results. The speed-up achieved by our algorithm is about 10 to 20 times compared with LIBSVM. We observed that our method requires fewer prototype vectors than support vectors in SVM (a fraction of the total number of training points which is roughly between 1% and 24%), hence resulting in a sparser model.

Table 12
Comparison of the 10-times-randomized **test set** performances (L_2, L_1, L_∞) and standard deviations (within parentheses) of FS-LSSVM (RBF kernel) with the performances of ε-SVR and ν-SVR on two regression domains. n_{test} is the number of observations in the test set and d is the total number of attributes. Also the numbers of prototype vectors (PV) and numbers of support vectors (SV) used by the algorithms are reported. The numbers of prototype vectors of FS-LSSVM are determined by the heuristic described in Section 5.

		bho	ccs
n_{test}		168	343
d		14	9
♯ PV FS-LSSVM		200	120
♯ SV ε-SVR		226	670
♯ SV ν-SVR		195	330
RBF FS-LSSVM	L_2	0.13(0.02)	0.17(0.02)
	L_1	0.24(0.02)	0.30(0.03)
	L_∞	1.90(0.50)	1.22(0.42)
RBF ε-SVR	L_2	0.16(0.05)	0.23(0.02)
	L_1	0.24(0.03)	0.33(0.02)
	L_∞	2.20(0.54)	1.63(0.58)
RBF ν-SVR	L_2	0.16(0.04)	0.22(0.02)
	L_1	0.26(0.03)	0.34(0.03)
	L_∞	1.97(0.58)	1.72(0.52)

Table 13
Comparison of the average computation time in seconds for FS-LSSVM, ε-SVR and ν-SVR on two regression problems. The standard deviation is shown within parentheses.

Av. time (s)	bho	ccs
RBF FS-LSSVM	74(2)	94(3)
RBF ε-SVR	63(1)	168(3)
RBF ν-SVR	61(1)	131(2)

Acknowledgements

BDM is a full professor at the Katholieke Universiteit Leuven, Belgium. JS is a professor at the Katholieke Universiteit Leuven, Belgium. This research was supported by Research Council KUL: GOA AMBioRICS, GOA MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several Ph.D./post-doc & fellow grants; Flemish Government: FWO: Ph.D./postdoc grants, projects G.0452.04 (new quantum algorithms), G.0499.04 (Statistics), G.0211.05 (Nonlinear), G.0226.06 (cooperative systems and optimization), G.0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), IWT: Ph.D. Grants, McKnow-E, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, POM, Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007–2011); EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, EMBOCOM, Contract Research: AMINAL, Other: Helmholtz, viCERP, ACCM, Bauknecht, Hoerbiger.

Appendix. Coupled simulated annealing

We briefly review the method of coupled simulated annealing (CSA) (Xavier de Souza et al., in press).

CSA features acceptance probability functions of a new form that can be applied to an ensemble of optimizers. This approach considers several current states which are coupled together by their energies in their acceptance probability function. Also, as distinct from the case for classical SA techniques, parallelism is an inherent characteristic of this class of methods. The objective of creating coupled acceptance probability functions that comprise the energy of many current states, or solutions, is to generate more information when deciding to accept less favorable solutions.

The following equation describes the acceptance probability function A with coupling term ρ:

$$A_\theta(\rho, x_i \to y_i) = \frac{\exp\left(\frac{-E(y_i)}{T_k^{ac}}\right)}{\exp\left(\frac{-E(y_i)}{T_k^{ac}}\right) + \rho},$$

with $A_\theta(\rho, x_i \to y_i)$ the acceptance probability for every $x_i \in \Theta$, $y_i \in \Upsilon$ and $\Upsilon \subset \Theta$. Υ denotes the set of all possible states and the set $\Theta \equiv \{x_i\}_{i=1}^q$ is presented as the set of current states of q minimizers. The variance σ^2 of A_θ equals

$$\frac{1}{q} \sum_{\forall\, x_i \in \Theta} A_\Theta^2 - \frac{1}{q^2}.$$

The coupling term ρ is given by

$$\rho = \sum_{x_j \in \Theta} \exp\left(\frac{-E(y_i)}{T_k^{ac}}\right).$$

Hence, CSA considers many current states in the set Θ, which is a subset of all possible solutions Υ, and accepts a probing state y_i based not only on the corresponding current state x_i but also on the coupling term, which depends on the energy of all other elements of Υ. Algorithm 6 summarizes the complete CSA procedure:

Algorithm 6 CSA with variance control (Xavier de Souza et al., 2009)

Initialization: assign q random initial solutions to Θ; assess the costs $E(x_i), \forall x_i \in \Theta$ and evaluate coupling term ρ; set initial temperatures $T_k = T_0$ and $T_k^{ac} = T_0^{ac}$; set time index $k = 0$, $\sigma_D^2 = 0.99\left(\frac{q-1}{q^2}\right)$ and $\alpha = 0.05$
for $g = 1$ to G inner iterations **do**
  Generate a probing solution y_{ig} for each element of Θ according to $y_{ig} = x_{ig} + \varepsilon_{ig}, \forall x_{ig} \in \Theta$ and ε_i is a random variable sampled from a given distribution; assess the costs for all probing solutions $E(y_{ig}), \forall i = 1, \ldots, q$,
  For each $i \in 1, \ldots, q$
  if $E(y_{ig}) \leq E(x_{ig})$ **then**
    accept solution y_{ig} with probability 1
  else
    accept solution with probability $A_\theta(\rho, x_{ig} \to y_{ig})$
    if $A_\Theta > r$, with r sampled from $\mathcal{U}[0, 1]$ **then**
      set $x_{ig} = y_{ig}$
    end if
  end if
  evaluate ρ_g
end for
Adjust acceptance temperature T_k^{ac}
if $\sigma^2 < \sigma_D^2$ **then**
  $T_k^{ac} = T_{k-1}^{ac}(1 - \alpha)$
else
  $T_k^{ac} = T_{k-1}^{ac}(1 + \alpha)$
end if
Decrease generation temperature, e.g. $T_k = \frac{T_0}{k+1}$
if stopping criterion is met **then**
  Stop
else
  Go to Step 2
end if

References

Abe, S., 2007. Sparse least squares support vector training in the reduced empirical feature space. Pattern Analysis and Applications 10, 203–214.
Achlioptas, D., McSherry, F., Schölkopf, B., 2002. Sampling techniques for kernel methods. In: Annual Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference, pp. 335–342.
Akaike, H., 1973. Statistical predictor identification. Annals of the Institute of Statistical Mathematics 22, 203–217.
Aldershof, B., 1991. Estimation of Integrated Squared Density Derivatives. Ph.D. Thesis, University of North Carolina, Chapel Hill, USA.
An, S., Liu, W., Venkatesh, S., 2007. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. Pattern Recognition 40, 2154–2162.
Baker, C.T.H., 1977. The Numerical Treatment of Integral Equations. Oxford Clarendon Press.
Baudat, G., Anouar, F., 2001. Kernel based methods and function approximation. In: Proc. of the International Joint Conference on Neural Networks, vol. 3, pp. 1244–1249.
Bennett, K.P., Hu, J., Ji, X., Kunapuli, G., Pang, J.S., 2006. Model selection via bilevel optimization. In: International Joint Conference on Neural Networks, pp. 1922–1929.
Blake, C.L., Merz, C.J., 1998. UCI repository of machine learning databases. http://archive.ics.uci.edu/ml/datasets.html, Irvine, CA.
Bowman, A.W., 1984. An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71, 353–360.
Burman, P., 1989. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning–testing methods. Biometrika 76 (3), 503–514.
Cawley, G.C., Talbot, N.L.C., 2004. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. Neural Networks 17, 1467–1475.
Cawley, G.C., Talbot, N.L.C., 2002. Reduced rank kernel ridge regression. Neural Processing Letters 16, 293–302.
Chang, C.C., Lin, C.J., 2001. LIBSVM: A library for support vector machines. Software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.
Collobert, R., Bengio, S., Bengio, Y., 2002. A parallel mixture of SVMs for very large scale problems. Neural Computation 14, 1105–1114.
Davison, A.C., Hinkley, D.V., 2003. Bootstrap Methods and Their Application, reprinted with corrections. Cambridge University Press.
Deheuvels, P., 1977. Estimation Nonparamétrique de la Densité par Histogrammes Généralisés. Revue Statistique Applique 25, 5–42.
Devroye, L., Györfi, L., Lugosi, G., 1996. A Probabilistic Theory of Pattern Recognition. Springer.
Devroye, L., Lugosi, G., 1989. Combinatorial Methods in Density Estimation. Springer-Verlag.
Drineas, P., Mahoney, M.W., 2005. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. Journal of Machine Learning Research 6, 2153–2175.
Espinoza, M., Suykens, J.A.K., Belmans, R., De Moor, B., 2007. Electric load forecasting—using kernel based modeling for nonlinear system identification. IEEE Control Systems Magazine, Special Issue on Applications of System Identification 27 (5), 43–57.
Fan, J., Yao, Q., Tong, H., 1996. Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. Biometrika 83, 189–206.

Faraway, J.J., Jhun, M., 1990. Bootstrap choice of bandwidth for density estimation. Journal of American Statistical Association 85, 1119–1122.
Fletcher, R., 1987. Practical Methods of Optimization. John Wiley & Sons.
Girolami, M., 2002. Orthogonal series density estimation and the kernel eigenvalue problem. Neural Computation 14, 669–688.
Golub, G., Van Loan, C., 1989. Matrix Computations. John Hopkins University Press.
Hall, P., 1980. Objective Methods for the Estimation of Window Size in the Nonparametric Estimation of a Density (unpublished manuscript).
Hall, P., Wolf, R.C., Yao, Q., 1999. Methods for estimating a conditional distribution function. Journal of the American Statistical Association 94, 154–163.
Härdle, W., 1991. Smoothing Techniques with Implementation in S. Springer-Verlag, New York.
Härdle, W., 1989. Resampling for inference from curves. In: Proceedings of the 47th Session of International Statistical Institute, pp. 59–69.
Hoegaerts, L., Suykens, J.A.K., Vandewalle, J., De Moor, B., 2004. A comparison of pruning algorithms for sparse least squares support vector machines. In: Proceedings of the 11th International Conference on Neural Information Processing, ICONIP 2004, vol. 3316, pp. 1247–1253.
Holte, R.C., 1993. Very simple classification rules perform well on most commonly used datasets. Machine Learning 11, 63–90.
Huang, C.-M., Lee, Y.-J., Lin, D.K.J., Huang, S.-Y., 2007. Model selection for support vector machines via uniform design. Computational Statistics & Data Analysis 52 (1), 335–346.
Ingber, L., 1989. Very fast simulated re-annealing. Journal of Mathematical Computer Modelling 12, 967–973.
Jiao, L., Bo, L., Wang, L., 2007. Fast sparse approximation for least squares support vector machine. IEEE Transactions on Neural Networks 18 (3), 685–697.
Joachims, T., 1999. Making large-scale SVM practical. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge, pp. 169–184.
Jones, M.C., Marron, J.S., Sheater, S.J., 1996. A brief survey of bandwidth selection for density estimation. Journal of the American Statistical Association 91 (433), 401–407.
Keerthi, S.S., Chapelle, O., DeCoste, D., 2008. Building support vector machines with reduced classifier complexity. Journal of Machine Learning Research 7, 1493–1515.
Keerthi, S.S., Shevade, S.K., 2003. SMO algorithm for least-squares SVM formulations. Neural computation 15 (2), 487–507.
Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E., 1998. Convergence properties of the Nelder–Mead simplex method in low dimensions. SIAM Journal of Optimization 9 (1), 112–147.
Li, Q., Racine, J.S., 2006. Nonparametric Econometrics: Theory and Practice. Princeton University Press.
Liu, X., Hall, L., Bowyer, K.W., 2004. Comments on "A parallel mixture of SVMs for very large scale problems". Neural Computation 16, 1345–1351.
Liu, Y., Zhang, H.H., Park, C., Ahn, J., 2007. Support vector machines with adaptive L_q penalty. Computational Statistics & Data Analysis 51 (12), 6380–6394.
Mallows, C.L., 1973. Some comments on C_p. Technometrics 15, 661–675.
Marron, J.S., Wand, M.P., 1992. Exact mean integrated squared error. Annals of Statistics 20 (2), 712–736.
Mika, S., Rätsch, G., Weston, J., Schölkopf, B., Müller, K.-R., 1999. Fisher discriminant analysis with kernels. In: Hu, Y.-H., Larsen, J., Wilson, E., Douglas, S. (Eds.), IEEE Neural Networks for Signal Processing IX, pp. 41–48.
Nelder, J.A., Mead, R., 1965. A simplex method for function minimization. Computer Journal 7, 308–313.
Nyström, E.J., 1930. Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. Acta Mathematica 54, 185–204.
Osuna, E., Freund, R., Girosi, F., 1997a. An improved training algorithm for support vector machines. In: Proceedings of the 1997 IEEE workshop Neural Networks for Signal Processing VII, pp. 276–285.
Osuna, E., Freund, R., Girosi, F., 1997b. Support vector machines: Training and applications. Technical Report AIM-1602, MIT Artificial Intelligence Laboratory, Cambridge, MA.
Park, B.U., Marron, J.S., 1999. On the use of pilot estimators in bandwidth selection. Journal of Nonparametric Statistics 1, 231–240.
Platt, J., 1999. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge, pp. 185–208.
Press, W., Teukolsky, S., Vetterling, W., Flannery, B., 1993. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press.
Principe, J.C., Fisher, J., Xu, D., 2000. Information theoretic learning. In: Haykin, S. (Ed.), Unsupervised Adaptive Filtering. In: Blind Source Separation, vol. 1. John Wiley & Sons, New York, pp. 265–319.
Rajasekaran, S., 2000. On simulated annealing and nested annealing. Journal of Global Optimization 16, 43–56.
Rätsch, G., Onoda, T., Müller, K.-R., 2001. Soft margin for Adaboost. Machine Learning 42 (3), 287–320.
Raykar, V.C., Duraiswami, R., 2006. Fast optimal bandwidth selection for kernel density estimation. In: Proc. of the 2006 SIAM International Conference on Data Mining, Bethesda, Maryland.
Raykar, V.C., Duraiswami, R., 2007. The improved fast gauss transform with applications to machine learning. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (Eds.), Large-Scale Kernel Machines. MIT Press, pp. 175–202.
Rényi, A., 1961. On measures of information and entropy. In: Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, pp. 547–561.
Rényi, A., 2007. Probability Theory. Dover Publications.
Rudemo, M., 1982. Empirical choice of histograms and kernel density estimators. Scandinavian Journal of Statistics 9, 65–78.
Saunders, C.J., Stitson, M., Weston, J., Bottou, L., Schölkopf, B., Smola, A.J., 1998. Support vector machine reference manual. Technical Report CSD-TR-98-03, Royal Holloway University of London.
Schölkopf, B., Williamson, R.C., Bartlett, P.L., 2000. New support vector algorithms. Neural Computation 12, 1207–1245.
Schwartz, G., 1979. Estimating the dimension of a model. Annals of Statistics 6, 461–464.
Scott, D.W., 1992. Multivariate Density Estimation. John Wiley & Sons, Inc.
Scott, D.W., Terrel, G.R., 1987. Biased and unbiased cross-validation in density estimation. Journal of the American Statistical Association 82, 1131–1146.
Shannon, C.E., 1948. A mathematical theory of communication. The Bell System Technical Journal.
Sheather, S.J., 1986. An improved data-based algorithm for choosing the window width when estimating the density at a point. Computational Statistics and Data Analysis 4, 61–65.
Sheather, S.J., 2004. Density estimation. Statistical Science 19 (4), 588–597.
Sheather, S.J., Jones, M.C., 1991. A reliable data-based bandwidth selection method for kernel density estimation. Journal of the Royal Statistical Society B 53, 683–690.
Silverman, B.W., 1986. Density Estimation for Statisticians and Data Analysis, 1st ed. Chapman & Hall.
Silverman, B.W., 1978. Choosing the window width when estimating a density. Biometrika 65, 1–11.
Smola, A. J., Schölkopf, B., 2000. Sparse greedy matrix approximation for machine learning. In: Proceedings of the 17th International Conference on Machine Learning, pp. 911–918.
Suykens, J.A.K., Lukas, L., Van Dooren, P., De Moor, B., Vandewalle, J., 1999. Least squares support vector machine classifiers : A large scale algorithm. In: Proceedings of the European Conference on Circuit Theory and Design, ECCTD'99, pp. 839–842.
Suykens, J.A.K., Vandewalle, J., 1999. Least squares support vector machine classifiers. Neural Processing Letters 9 (3), 293–300.
Suykens, J.A.K., Vandewalle, J., De Moor, B., 2001. Intelligence and cooperative search by coupled local minimizers. International Journal of Bifurcation and Chaos 11 (8), 2133–2144.
Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J., 2002. Least Squares Support Vector Machines. World Scientific, Singapore.
Suykens, J.A.K., Yalcin, M.E., Vandewalle, J., 2003. Coupled chaotic simulated annealing processes. In: IEEE International Symposium on Circuits and Systems, ISCAS, pp. 582–585.
Taylor, C.C., 1989. Bootstrap choice of the smoothing parameter in kernel density estimation. Biometrika 76, 705–712.
Valyon, J., Horváth, G., 2004. A sparse least squares support vector machine classifier. In: Proc. of the International Joint Conference on Neural Networks, IJCNN 2004, pp. 543–548.

Van Gestel, T., Suykens, J.A.K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., De Moor, B., Vandewalle, J., 2004. Benchmarking least squares support vector machine classifiers. Machine Learning 54 (1), 5–32.
Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York.
Vapnik, V.N., 1999. Statistical Learning Theory. John Wiley & Sons, Inc.
Vapnik, V., Chapelle, O., 2000. Bounds on error expectation for support vector machines. Neural Computation 12 (9), 2013–2036.
Vollbrecht, K.G.H., Wolf, M.M., 2002. Conditional entropies and their relation to entanglement criteria. Journal of Mathematical Physics 43, 4299.
Wahba, G., Lin, Y., Zhang, H., 2000. Generalized approximate cross-validation for support vector machines. In: Bartlett, P.J., Schölkopf, B., Schuurmans, D., Smola, A.J. (Eds.), Advances in Large Margin Classifiers. MIT Press, Cambridge, pp. 297–309.
Wand, M.P., Jones, M.C., 1995. Kernel Smoothing. Chapman & Hall.
Williams, C.K.I., Barber, D., 1998. Bayesian classification with Gaussian processes. IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (12), 1342–1351.
Williams, C.K.I., Seeger, M., 2001. Using the Nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems.
Xavier de Souza, S., Suykens, J.A.K., Vandewalle, J., Bollé, D., 2006. Cooperative behavior in coupled simulated annealing processes with variance control. In: Proc. of the International Symposium on Nonlinear Theory and its Applications, NOLTA2006 pp. 114–119.
Xavier de Souza, S., Suykens, J.A.K., Vandewalle, J., Bollé, D., 2009. Coupled simulated annealing. IEEE Transactions on Systems, Man and Cybernetics—Part B, in press (doi:10.1109/TSMCB.2009.2020435).
Yang, C., Duraiswami, R., Gumerov, N., Davis, L., 2003. Improved fast Gauss transform and efficient kernel density estimation. IEEE International Conference on Computer Vision 1, 464–471.
Ying, Z., Keong, K.C., 2004. Fast leave-one-out evaluation and improvement on inference for LS-SVM's. In: Proceedings of the 17th International Conference on Pattern Recognition, ICPR, vol. 3, pp. 494–497.
Zhao, Y., Sun, J., 2008. Recursive reduced least squares support vector machines. Pattern Recognition 42, 837–842.