

# Optimized Fixed-Size Kernel Models for Large Data Sets

K. De Brabanter<sup>a,\*</sup>, J. De Brabanter<sup>a,b</sup>, J.A.K. Suykens<sup>a</sup>, B. De Moor<sup>a</sup>

<sup>a</sup>*Department of Electrical Engineering ESAT-SCD, Katholieke Universiteit Leuven,  
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium*

<sup>b</sup>*Hogeschool KaHo Sint-Lieven (Associatie K.U.Leuven), Departement Industrieel  
Ingenieur, B-9000 Gent*

---

## 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 is compared to standard support vector machines and  $\nu$ -support vector machines resulting in sparser models with lower computational cost and comparable accuracy.

*Keywords:* kernel methods, least squares support vector machines, classification, regression, plug-in estimate, entropy, cross-validation

---

## 1. Introduction

Support vector machines (SVM) (Vapnik, 1995, 1999) and least-squares support vector machines (LS-SVM) (Suykens & Vandewalle, 1999; Suykens et al., 2002) are state-of-the-art learning algorithms for pattern recognition

---

\*Corresponding author

*Email 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)

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 design 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 reason SMO is considered as the current state-of-the-art QP solver for solving medium as well as large-scale SVM's.

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  $L_2$  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 & 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's was proposed by Keerthi & Shevade (2003). This method is based on the solution of the dual problem using a similar idea to that of the SMO algorithm, i.e. using Wolfe duality theory, for SVM's.

The vast majority of textbooks and articles discussing SVM's and LS-SVM's first state the primal optimization problem and then go directly to the dual formulation (Vapnik, 1995; Suykens & Vandewalle, 1999). A successful attempt for solving LS-SVM's in primal weight space resulting in a parametric model and sparse representation, introduced by Suykens et al. (2002), is called *Fixed-Size Least Squares Support Vector Machines* (FSLSSVM) and was also applied in Espinoza et al. (2007). In this method an explicit expression of the feature map or an approximation to it is required. A procedure to find this approximated feature map is based on the Nyström method (Nyström, 1930; Baker, 1977). Williams & Seeger (2001)

used the Nyström method to speed up Gaussian processes (GP) (Williams & 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 literature (Suykens et al., 2002; Achlioptas et al., 2002; Drineas & Mahoney, 2005). Smola & Schölkopf (2000) presented a sparse greedy approximation technique to construct 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 upperbound 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 to search 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 remaining 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 especially for 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 & Terrel, 1987), smoothed bootstrap (SB) (Taylor, 1989; Faraway & Jhun, 1990), plug-ins (Hall, 1980; Sheather, 1986; Sheather & Jones, 1991), reference rules (Deheuvels, 1977; Silverman, 1986). In this paper we use the *solve-the-plug-in equation* (Sheather & Jones, 1991) which is related to the plug-in family. The rationale for using this method is based on the fact that it can be calculated 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 to estimate 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 & Chapelle, 2000; Wahba et al., 2000) for SVM, (Ying & Keong, 2004; An et al., 2007) for LS-SVM and (Cawley & Talbot,

2004) for sparse LS-SVM. Using the fact that 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 to estimate 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 produced solution. 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 to find 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, 2009). 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 class of methods consider the work of Suykens et al. (2001). One of the largest differences with SA is that CSA features a new form of acceptance probabilities 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 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's 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 to determine the number of PVs (prototype vectors). Section 6 discusses the proposed  $v$ -fold CV algorithm for FS-LSSVM. In Sections 7 to 8 the different algorithms are successfully demonstrated on real-life data sets. Section 9 states the conclusions of the paper. Finally, Appendix A 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  $\mathcal{D}_n = \{(X_k, Y_k) : X_k \in \mathbb{R}^d, Y_k \in \{-1, +1\}; k = 1, \dots, n\}$ , 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 & Vandewalle, 1999)

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

The classifier in the primal weight space takes the form

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

where  $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^{n_h}$  is the feature map to the high dimensional feature space, where  $n_h$  denotes the dimension of the feature space (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 & 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  $K(x, X_k) = \varphi(x)^T \varphi(X_k)$ ,  $\alpha$  and  $b$  are the solution of the linear system

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

with  $1_n = (1, \dots, 1)^T$ ,  $\Omega_{kl} = Y_k Y_l \varphi(X_k)^T \varphi(X_l)$  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  $\varphi$  but  $\varphi$  is often not explicitly known.

## 2.2. Regression

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

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

where  $e_k \in \mathbb{R}$  are assumed to be i.i.d. random errors with  $E[e_k | X = X_k] = 0$ ,  $\text{Var}[e_k] = \sigma^2 < \infty$ ,  $g \in C^z(\mathbb{R})$  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)

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

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 & Van Loan, 1989) with 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  $\alpha$

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

with  $Y = (Y_1, \dots, Y_n)^T$ ,  $1_n = (1, \dots, 1)^T$ ,  $\alpha = (\alpha_1, \dots, \alpha_n)^T$  and  $\Omega_{kl} = \varphi(X_k)^T \varphi(X_l) = K(X_k, X_l)$ , with  $K(X_k, X_l)$  positive definite, for  $k, l = 1, \dots, n$ . 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 primal weight space

In the following subsection we review how to estimate an approximation to the feature map  $\varphi$  using the Nyström method (Williams & 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  $\varphi$  or an approximation to the feature map  $\hat{\varphi} : \mathbb{R}^d \rightarrow \mathbb{R}^m$ , with  $m \ll n$ . Let  $X_k \in \mathbb{R}^d$ ,  $k = 1, \dots, n$  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

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

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

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

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

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

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

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,  $U = (u_1, \dots, u_n)$  is a  $n \times n$  matrix of eigenvectors of  $\Omega$  and  $\Lambda$  is a  $n \times n$  diagonal matrix of nonnegative eigenvalues in a decreasing order. Expression (3) delivers direct approximations of the eigenvalues and eigenfunctions for the  $x_k \in \mathbb{R}^d$ ,  $k = 1, \dots, n$  points

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

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).  $\lambda_i$  denote the eigenvalues of (2). Substituting (4) in (3) results in an approximation of an eigenfunction evaluation in 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). Based on the Nyström approximation, an expression for the entries of the approximation of the feature map  $\hat{\varphi}_i : \mathbb{R}^d \rightarrow \mathbb{R}$ , with  $\hat{\varphi}(x) = (\hat{\varphi}_1(x), \dots, \hat{\varphi}_m(x))^T$ , is given by

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



In order to introduce parsimony, one chooses a fixed size  $m$  ( $m \ll n$ ) as a working subsample (see Section 4). A likewise  $m$ -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 & Seeger (2001) to speed-up Gaussian Processes (GP). However, Williams & 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 solution 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)

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

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

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

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

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

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 PVs an entropy based selection method is discussed as proposed by Suykens et al. (2002). It is especially important to select the PVs. The selected PVs 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 PVs, 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 & Wolf, 2002) and Rényi (Rényi, 1961; Girolami, 2002; Vollbrecht & Wolf, 2002; Principe et al., 2000). The second subsection reviews a method, based on the solve-the-equation plug-in method introduced by Sheather & Jones (1991), to select the smoothing parameter for entropy estimation. We investigate this for use in Fixed-Size LS-SVM's for large scale applications.

##### 4.1. Subsample based on entropy criteria

Let  $X_k \in \mathbb{R}^d$ ,  $k = 1, \dots, n$  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  $X_k \in \mathbb{R}^d$ ,  $k = 1, \dots, n$ , is contained in a subsample  $X_j \in \mathbb{R}^d$ ,  $j = 1, \dots, m$  ( $m \ll n$ ). In other words, the purpose of a subsample selection is to extract  $m$  ( $m \ll n$ ) samples from  $\{X_1, \dots, X_n\}$ ,

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 mentioned before, two entropy criteria will be used i.e. Shannon and Rényi. The Shannon or differential entropy  $H_S(X)$  is defined by

$$\begin{aligned} H_S(X) &= \mathbb{E}[-\log f(X)] \\ &= - \int \dots \int f(x) \log(f(x)) dx, \end{aligned} \quad (9)$$

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

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

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)

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

where  $h_j$  denotes the bandwidth for each dimension  $j$  and the kernel  $\kappa : \mathbb{R} \rightarrow \mathbb{R}$  satisfies  $\int_{\mathbb{R}} \kappa(u) du = 1$ . 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  $(X_1, \dots, X_n)^T \in \mathbb{R}^{n \times d}$ . Another possibility to estimate  $f(x)$  is by using the general multivariate kernel estimator (Scott, 1992) given by

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

where  $|\cdot|$  denotes the determinant,  $D$  is a non-singular matrix of the form  $D = \text{diag}(h_1, \dots, h_d)$  and the kernel  $\kappa : \mathbb{R}^d \rightarrow \mathbb{R}$  satisfies  $\int_{\mathbb{R}^d} \kappa(u) du = 1$ . 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

$H_{R2}^m(x)$ , see (10). The differential entropy can be viewed as one member of the Rényi's entropy family, because  $\lim_{q \rightarrow 1} H_{Rq}(x) = H_S(x)$ . 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's 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 (11) and (10), Rényi's quadratic entropy estimator, based on  $m$  prototype vectors and setting  $q = 2$ , becomes

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

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 (PVs). 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  $\mathcal{D}_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ , 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**  $\hat{H}_{R2}^m(X_1, \dots, X_{m-1}; X^+) > \hat{H}_{R2}^m(X_1, \dots, X_i^*, \dots, X_m)$  **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  $\hat{H}_{R2}^m(X)$  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  $X_1, \dots, X_n$  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 to select the bandwidth  $h$  in the case of kernel density estimation are available e.g. least squares cross-validation (Silverman, 1986; Li & Racine, 2006), biased cross-validation (Scott & Terrel, 1987), bootstrap bandwidth selector (Hall et al., 1999; Li & Racine, 2006), regression-based bandwidth selector (Härdle, 1991; Fan & Tong, 1996), double kernel method (Devroye & Lugosi, 1989), plug-in methods (Silverman, 1986; Li & Racine, 2006; Raykar & 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

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

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

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

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

where  $R(\kappa) = \int_{\mathbb{R}} \kappa(x)^2 dx$ ,  $\mu_2(\kappa) = \int_{\mathbb{R}} x^2 \kappa(x)^2 dx$ ,  $\kappa(x)$  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  $R(f^{(2)})$  depends on the second derivative of the density  $f$ . An estimator of the functional  $R(f^{(r)})$  using a kernel density derivative estimate for  $f^{(r)}$  is given by

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

The optimal bandwidth  $h_{\text{AMSE}}$  to estimate the density functional is given in Theorem 2.

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

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

#### 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 & Jones (1991). The AMISE optimal bandwidth (13), see Theorem 1, can now be written as follows

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

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

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

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

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

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

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

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

This of course also reveals the problem of how to choose  $r$ , 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 & Marron (1992) favour taking  $r$  to be at least 2, with  $r = 2$  being a common choice.

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

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

The main computational bottleneck is the estimation of the kernel density derivatives  $R(f^{(r)})$  which is of  $\mathcal{O}(n^2)$ . A method for fast evaluation of these kernel density derivatives  $R(f^{(r)})$  is proposed in Raykar & 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 & Duraiswami (2007). The Matlab code for IFGT is freely available at [http://www.umiacs.umd.edu/~vikas/Software/IFGT/IFGT\\_code.htm](http://www.umiacs.umd.edu/~vikas/Software/IFGT/IFGT_code.htm).

**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 both performances on test we use the UCI Boston Housing

---

**Algorithm 2** solve-the-equation plug-in method

---

- 1: Compute an estimate  $\hat{\sigma}$  of the standard deviation.
  - 2: Estimate density functionals  $\hat{R}(f^{(6)})$  and  $\hat{R}(f^{(8)})$  using (17).
  - 3: Estimate density functionals  $\hat{R}(f^{(4)})$  and  $\hat{R}(f^{(6)})$  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. Newton-Raphson method.
- 

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 next section) for both selection criteria. Figure 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.

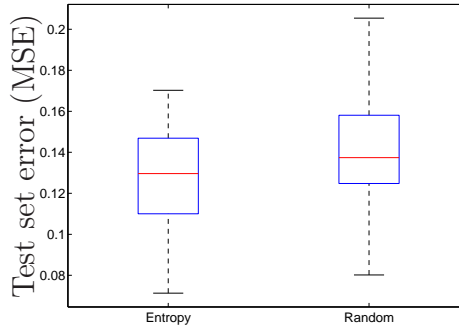


Figure 1: Boxplot of the MSE on test (30 runs) for models estimated with entropy based and random selection of prototype vectors.

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



## 5. Selecting the number of prototype vectors

An important task in this framework is to determine the number of PVs  $m \in \mathbb{N}_0$  used in the FS-LSSVM model. Existing methods (Smola & Schölkopf, 2000; Keerthi et al., 2006; Jiao et al., 2007; Zhao & Sun, 2008) select the number of PVs based on a greedy approach. One disadvantage of these type of methods is that they are time consuming, hence making them infeasible for large scale data sets. In Smola & Schölkopf (2000) and Keerthi et al. (2006) an additional parameter is introduced in the subset selection process which requires tuning. This parameter determines the number of random PVs to try outside the already selected subset. Keerthi et al. (2006) pointed out that there is no universal answer to set this parameter because the answer would depend on the cost associated with the computation of the kernel function, on the number of selected PVs and on the number of training points.

A second class of methods to select the PVs is by constructing a basis in feature space (Baudat & Anouar, 2001). This method was also used by Cawley & 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 (PVs). This method is computationally heavy since it involves numerous matrix-matrix products and matrix inverses.

A third class of method is based on matrix algebra to obtain the number of PVs. Valyon & Horváth (2004) obtain PVs by bringing the kernel matrix to the reduced row echelon form (Golub & 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 be become intractable for large data sets. Abe (2007) uses an incomplete Cholesky factorization to select the PVs. The number of PVs 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 into the memory, which can be problematic when data sets are large, but the factorization can be done incrementally.

Our proposed method selects the PVs by maximizing the quadratic Rényi entropy (12). This algorithm requires the entropy kernel bandwidth(s), see Subsection 4.2. The computational complexity associated with determining  $d$  bandwidths is roughly  $nd$  (Raykar & Duraiswami, 2006), where  $d$  is the number of dimensions. One only has to calculate the kernel matrix associated with the PVs 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 when the set of PVs 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 PVs has to be calculated only once and can then be removed from the memory.

In theory we could gradually increase the number of PVs till  $m = n$ . Naturally it would be to computationally expensive, but it gives an insight of 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 test set) and the Boston Housing data set (168 data points are randomly selected as test set). In this example the number of PVs  $m$  are gradually increased and the performance on test data is determined for each of the chosen prototype vector sizes. Figure 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.

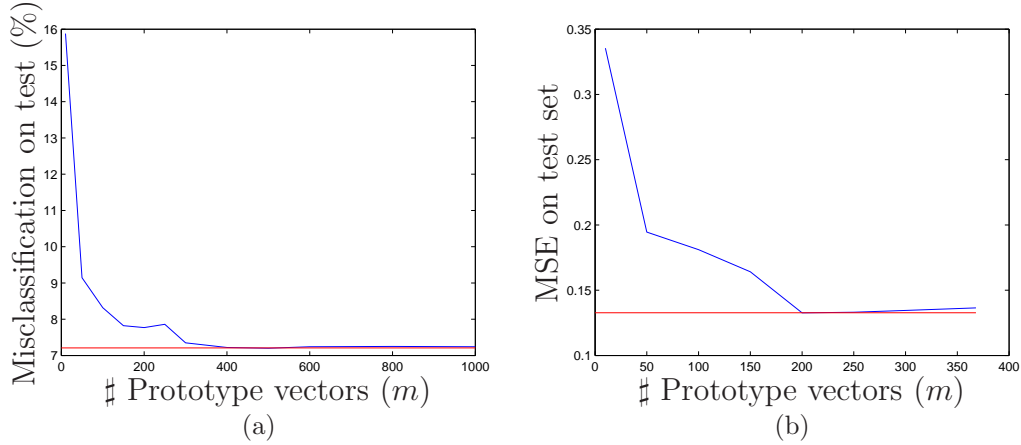


Figure 2: Number of prototype vectors in function of the performance on a test set. The straight line is the LS-SVM estimate and serves as a baseline comparison. (a) Spam data (binary classification); (b) Boston Housing data (regression).

In practice, however, due to time constraints and computational burden, it is impossible to gradually increase the number of prototype vectors till 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 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 final  $m$ , the number of prototype vector of the model which has the best performance on test data. Also note that in this way not always the sparsest model is selected, but it reduces computation time.

In the FS-LSSVM framework the number of PVs  $m$  is a tuning parameter, but the choice is not very crucial as can be seen in Figure 2. This is however an inherent drawback of the method in contrast to SVM. In the SVM framework the number of SVs follow 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 & Hinkley, 2003).
- 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), Bayes Information Criterion (Schwartz, 1979) and 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's and sparse LS-SVM's a fast leave-one-out cross-validation algorithm based on one complete matrix inversion is found in Ying & Keong (2004) and Cawley & 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 An et al. (2007) and Ying & Keong (2004), the algorithm is not based on one full matrix inverse because of the complexity  $\mathcal{O}((m+1)^3)$  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 remaining is for training. The extended feature matrix  $\hat{\Phi}_e \in \mathbb{R}^{n \times (m+1)}$  is given by

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

where  $1_{tr} = (1, \dots, 1)^T$ ,  $1_{val} = (1, \dots, 1)^T$ ,  $X_{tr,j}$  and  $X_{val,j}$  denote the  $j$ th element of the training data and validation data respectively.  $n_{tr}$  and  $n_{val}$  are the number of data points in the training data set and validation set respectively such that  $n_{tr} + n_{val} = n$ . Also set  $A = \hat{\Phi}_e^T \hat{\Phi}_e + \frac{I_{m+1}}{\gamma}$  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 remaining is for training. So in the  $v$ -th iteration

$$A_v \begin{pmatrix} \tilde{w} \\ b \end{pmatrix} = c_v, \quad (19)$$

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

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

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

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

This results in

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

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 result holds for  $c_v$

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

thus

$$c_v = c - \left( \begin{array}{c|c} \hat{\Phi}_{val}^T & \\ \hline 1_{val}^T & \end{array} \right) Y_{val}. \quad (21)$$

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  $\hat{e}_{v,i} = Y_{val,i} - (\hat{w}^T \hat{\varphi}(X_{val,i}) + \hat{b})$  in each fold.
  - 6: Choose an appropriate loss function to assess 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 & Talbot (2004) and Ying & 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 which method can be best used with different type of folds.

implementation	# folds
Cawley & Talbot (2004)	$n$
An et al. (2007)	$> 20$
proposed	3-20

Table 2: Summary of different implementations of CV.

### 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 into the memory. Let  $l_s$ , with  $s = 1, \dots, S$ , 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

$$\hat{\Phi}_e = \begin{pmatrix} \frac{\hat{\Phi}_{e,[1]}}{\hat{\Phi}_{e,[S]}} \end{pmatrix},$$

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

$$Y = \begin{pmatrix} \frac{Y_{[1]}}{Y_{[S]}} \end{pmatrix},$$

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 into the memory since the size 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  $\mathcal{D}_{tr} = \{(X_1, Y_1), \dots, (X_t, Y_t)\}$  and a fixed validation sequence  $\mathcal{D}_{val} = \{(X_{t+1}, Y_{t+1}), \dots, (X_{t+l}, Y_{t+l})\}$ , 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.

### 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 perform 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 & Mead, 1965; Lagarias et al., 1998) using the previous result as start value.

To determine good initial starting values we use the method of Coupled Simulated Annealing with variance control (CSA) (Xavier de Souza et al.,

---

**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  $s = 1, \dots, S$ , calculated by (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:    $A_v \leftarrow A_v + \hat{\Phi}_{e,[s]}^T \hat{\Phi}_{e,[s]}$
  - 7:    $c_v \leftarrow c_v + \hat{\Phi}_{e,[s]}^T Y_{[s]}$
  - 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.
- 

2006, 2009), see also Appendix A, whose working principle was inspired by the effect of coupling in Coupled Local Minimizers (CLM) (Suykens et al., 2001) when 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.

## 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



---

**Algorithm 5** Summary of the optimized FS-LSSVM method

---

- 1: Given a training set defined as  $\mathcal{D}_n = \{(X_k, Y_k) : X_k \in \mathbb{R}^d, Y_k \in \mathbb{R}; k = 1, \dots, n\}$ .
  - 2: Determine the kernel bandwidth or bandwidth matrix for entropy selection according to Algorithm 2.
  - 3: Given the number of prototype vectors (PVs), begin the active PVs 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 into 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  $\hat{w}$  and  $\hat{b}$  by solving the linear system (7).
- 

of  $v$ -fold CV on a collection of data sets from 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). Noting that the complexity of 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  $m^3 + m^2n$ . 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
- Matrix 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 smaller than the simple method for  $v \geq 2$ . Keeping  $m$  fixed, it is clear that the number of folds have 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 where the computational complexity is increasing heavily with increasing folds. On the other hand consider the number of folds  $v$  fixed and  $m$  variable, a larger time increase is expected with the simple 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 efficiency of the proposed algorithm, the 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* (this data set is publicly available at <http://kdd.ics.uci.edu/> and has 1030 number of instances and 8 attributes.) data set (regression) for various number of prototype vectors (the number 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 at an 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 Figure 3 showing the number of folds as a function of computational time for various number of prototype vectors in the regression case.

Table 4 verifies the complexity of the algorithm on the *Magic Gamma Telescope* (this data set is publicly available at <http://kdd.ics.uci.edu/> and has 19020 number of instances and 10 attributes.) data set (binary classification) for various number 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 Figure 4 showing the number of folds as a function of computational time for various number of prototype vectors in the classification case.

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 number of prototype vectors was arbitrarily chosen.

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

## 8. Classification and regression results

In this Section, we report the application of FS-LSSVM on benchmark data sets (Blake & Merz, 1998) of which a brief description is included in Subsection 8.1. The performance of the FS-LSSVM is compared with standard SVM and  $\nu$ -SVM (LIBSVM software (Chang & Lin, 2001)). Although this paper focusses on large-scale data sets, also smaller data sets will be included for completeness. The randomized test set results are discussed in Subsections 8.3 - 8.4 (classification) and Subsection 8.5 (regression).

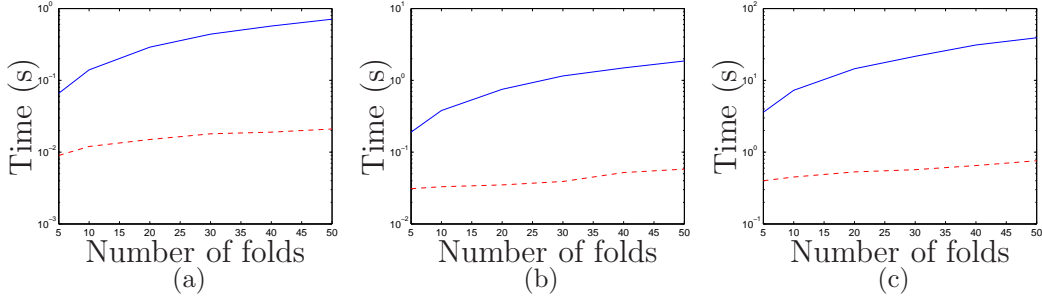


Figure 3: Number of folds as a function of computation time (in seconds) for various number of prototype vectors (50, 100, 400) on the Concrete Compressive Strength data set. The full line represents the simple implementation and the dashed line the proposed method.

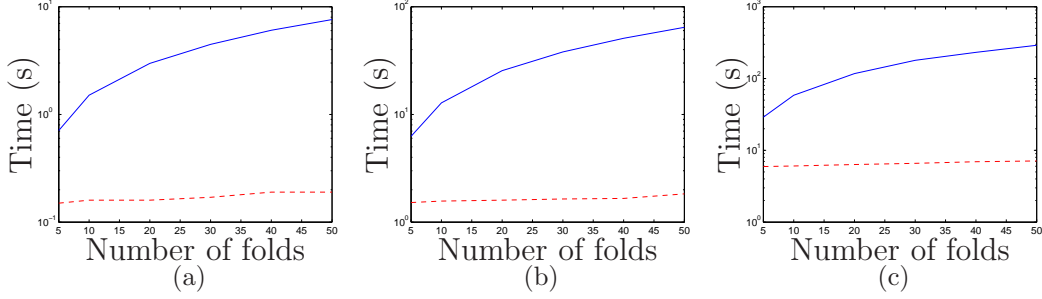


Figure 4: Number of folds as a function of computation time (in seconds) for various number of prototype vectors (50, 300, 700) on the Magic Gamma Telescope data set. The full line represents the simple implementation and the dashed line the proposed method.

### 8.1. Description of the data sets

All the data sets have been obtained from the publicly accessible UCI benchmark repository (Blake & 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.

#### 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**),

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 number of prototype vectors was arbitrarily chosen.

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

Adult (**adu**), Spam Database (**spa**) and Forest Covertype (**ftc**) data set. 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 7-class classification problem was transformed into a binary classification problem where the goal is to separate class 2 from the other 6 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 set. The main characteristics of these data sets are summarized in Table 6.

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

	pid	spa	mgt	adu	ftc
$n_{CV}$	512	3068	13000	33000	531012
$n_{test}$	256	1533	6020	12222	50000
$n$	768	4601	19020	45222	581012
$d_{num}$	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. The  $M$  row 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	7328	13667	43500
$n_{test}$	1870	2000	3664	6333	14500
$n$	5620	6435	10992	20000	58000
$d_{num}$	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

### 8.1.2. Regression

The following data sets for regression were also downloaded from the UCI benchmark data set: 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 performance of SVM and  $\nu$ -SVM (Schölkopf et al., 2000),

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

both implemented in the LIBSVM software. In case of  $\nu$ -SVM the parameter  $\nu \in ]0, 0.8]$  is also considered as a tuning parameter. The three methods use a cache size of 1GB 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 percent) 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 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 standard deviations on 10 randomizations are reported. Also the mean total time and corresponding standard deviation are mentioned. The total time of the algorithms consists of (i) 10-fold CV using the optimization strategy described in Subsection 6.3. The total number of function evaluations is set to 160 (90 for CSA and 70 for simplex search). In case of  $\nu$ -SVM the parameter  $\nu$  is also considered as a tuning parameter. We have used 5 multiple starters for the CSA algorithm; (ii) training with optimal tuning parameters and (iii) evaluation on 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 Subsection 6.3 for selecting the tuning parameters on the 7 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  $\nu$ -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 seconds) and standard deviation for both algorithms.

Table 8: Comparison of the 10 times randomized **test set** performances (in percentage) and standard deviations (within parentheses) of FS-LSSVM (linear and RBF kernel) with the performance of  $C$ -SVC,  $\nu$ -SVC and Majority Rule classifier on 5 binary domains.  $n_{\text{test}}$  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 performance of the  $C$ -SVC and  $\nu$ -SVC for fixed tuning parameter(s). No cross-validation was performed because of the computational burden.

	pid	spa	mgt	adu	ftc
$n_{\text{test}}$	256	1533	6020	12222	50000
$d$	8	57	11	14	54
‡ PV FS-LSSVM	150	200	1000	500	500
‡ SV $C$ -SVC	290	800	7000	11085	185000
‡ SV $\nu$ -SVC	331	1525	7252	12205	165205
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 $\nu$ -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)

The FS-LSSVM classifier with RBF kernel (RBF FS-LSSVM) achieves the best average test performance on 3 of the 5 benchmark domains, while its accuracy is comparable to RBF SVM ( $C$ -SVC). On all binary classification data sets  $\nu$ -SVC has a slightly lower 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 time for SVM ( $C$ -SVC and  $\nu$ -SVC) in case of the Forest Covertype data set, it is practically impossible to perform 10-fold CV. Therefore, the values in Table



Table 9: Comparison of the average computation time in seconds for the FS-LSSVM,  $C$ -SVC and  $\nu$ -SVC on 5 binary classification problems. The standard deviation is shown within parentheses. The values with an asterisk only denotes the training time for a fixed pair of tuning parameters for  $C$ -SVC and  $\nu$ -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 $\nu$ -SVC	30.3(2.3)	1372(43)	35299(357)	139927(3578)	55178(*)

8 and Table 9 with an asterisk only denote the training time 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  $\nu$ -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 in 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  $\nu$ -SVC). The same kernel types as in the binary classification problem are considered. Performances of FS-LSSVM and SVM ( $C$ -SVC and  $\nu$ -SVC) are reported. We used the same experimental setup as for binary classification. The test set performances 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. Performance on test as well as the accuracy of the multi-class FS-LSSVM and multi-class SVM ( $C$ -SVC and  $\nu$ -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 types of SVM models.

Table 10: Comparison of the 10 times randomized **test set** performances (in percentage) and standard deviations (within parentheses) of FS-LSSVM (RBF kernel) with the performance of  $C$ -SVC,  $\nu$ -SVC and Majority Rule classifier on 5 multi-class domains using MOC and 1vs1 output coding.  $n_{\text{test}}$  is the number of observations in the test set and  $d$  is the total number of attributes. Also the number of prototype vectors (PV) 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.

	opt	lan	pen	let	shu
$n_{\text{test}}$	1870	2000	3664	6333	14500
$d$	64	36	16	16	9
# PV FS-LSSVM	420	330	250	1500	175
# SV $C$ -SVC	3750	1876	1178	8830	559
# SV $\nu$ -SVC	2810	2518	4051	11359	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 $\nu$ -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 the FS-LSSVM,  $C$ -SVM and  $\nu$ -SVC on 5 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 $\nu$ -SVC (1vs1)	7963(178)	8229(304)	16589(453)	79040(2354)	50478(2879)

### 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 to assess 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  $\varepsilon$ -SVR and  $\nu$ -SVR. FS-LSSVM results into a sparser model for both data sets compared to  $\varepsilon$ -SVR. Also notice that both type of SVR methods have similar performances although  $\nu$ -SVR results into sparser models.

Table 12: Comparison of the 10 times randomized **test set** performances ( $L_2$ ,  $L_1$ ,  $L_\infty$ ) and standard deviations (within parentheses) of FS-LSSVM (RBF kernel) with the performance of  $\varepsilon$ -SVR and  $\nu$ -SVR on 2 regression domains.  $n_{\text{test}}$  is the number of observations in the test set and  $d$  is the total number of attributes. Also the number of prototype vectors (PV) 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.

		bho	ccs
$n_{\text{test}}$		168	343
$d$		14	9
# PV FS-LSSVM		200	120
# SV $\varepsilon$ -SVR		226	670
# SV $\nu$ -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)
	$L_\infty$	1.90(0.50)	1.22(0.42)
RBF $\varepsilon$ -SVR	$L_2$	0.16(0.05)	0.23(0.02)
	$L_1$	0.24(0.03)	0.33(0.02)
	$L_\infty$	2.20(0.54)	1.63(0.58)
RBF $\nu$ -SVR	$L_2$	0.16(0.04)	0.22(0.02)
	$L_1$	0.26(0.03)	0.34(0.03)
	$L_\infty$	1.97(0.58)	1.72(0.52)

Table 13: Comparison of the average computation time in seconds for the FS-LSSVM,  $\varepsilon$ -SVR and  $\nu$ -SVR on 2 regression problems. The standard deviation is shown within parentheses.

Av. Time (s)	bho	ccs
RBF FS-LSSVM	74(2)	94(3)
RBF $\varepsilon$ -SVR	63(1)	168(3)
RBF $\nu$ -SVR	61(1)	131(2)

## 9. Conclusions

In this paper an optimized version of Fixed-Size Least Squares Support Vector Machines (FS-LSSVM) 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  $\nu$ -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's were experimentally compared with two types of SVM's (LIBSVM software) on a number of classification 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 less 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.

## Acknowledgements

Research supported by: Research Council KUL: GOA AMBioRICS, CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/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), research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, McKnow-E, Eureka-Flite+; Helmholtz: viCERP. Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); EU: ERNSI; Contract Research: AMINAL

## Appendix A. Coupled Simulated Annealing

We shortly review the method of Coupled Simulated Annealing (CSA) (Xavier de Souza et al., 2009).

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 probability function. Also, as distinct from 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  $\rho$

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

with  $A_{\theta}(\rho, x_i \rightarrow y_i)$  the acceptance probability for every  $x_i \in \Theta$ ,  $y_i \in \Upsilon$  and  $\Upsilon \subset \Theta$ .  $\Upsilon$  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  $\sigma^2$  of  $A_{\theta}$  equals

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

The coupling term  $\rho$  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  $\Theta$ , which is a subset of all possible solutions  $\Upsilon$ , and accepts a probing state  $y_i$  based not only on the corresponding current state  $x_i$  but by considering also the coupling term, which depends on the energy of all other elements of  $\Upsilon$ . Algorithm 6 summarizes the complete CSA procedure:

---

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

---

```
1: Initialization: assign  $q$  random initial solutions to  $\Theta$ ; assess the costs  $E(x_i), \forall x_i \in \Theta$ 
   and evaluate coupling term  $\rho$ ; 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  $\alpha = 0.05$ 
2: for  $g = 1$  to  $G$  inner iterations do
3:   Generate a probing solution  $y_{ig}$  for each element of  $\Theta$  according to  $y_{ig} = x_{ig} +$ 
      $\varepsilon_{ig}, \forall x_{ig} \in \Theta$  and  $\varepsilon_i$  is a random variable sampled from a given distribution; assess
     the costs for all probing solutions  $E(y_{ig}), \forall i = 1, \dots, q$ ,
4:   For each  $i \in 1, \dots, q$ 
5:   if  $E(y_{ig}) \leq E(x_{ig})$  then
6:     accept solution  $y_{ig}$  with probability 1
7:   else
8:     accept solution with probability  $A_\theta(\rho, x_{ig} \rightarrow y_{ig})$ 
9:     if  $A_\theta > r$ , with  $r$  sampled from  $\mathcal{U}[0, 1]$  then
10:      set  $x_{ig} = y_{ig}$ 
11:    end if
12:  end if
13:  evaluate  $\rho_g$ 
14: end for
15: Adjust acceptance temperature  $T_k^{ac}$ 
16: if  $\sigma^2 < \sigma_D^2$  then
17:    $T_k^{ac} = T_{k-1}^{ac} (1 - \alpha)$ 
18: else
19:    $T_k^{ac} = T_{k-1}^{ac} (1 + \alpha)$ 
20: end if
21: Decrease generation temperature e.g.  $T_k = \frac{T_0}{k+1}$ 
22: if stopping criterion is met then
23:   Stop
24: else
25:   Go to Step 2
26: end if
```

---

## References

- Abe, S. (2007). Sparse Least Squares Support Vector Training in the Reduced Empirical Feature Space. *Pattern Anal. Applic.*, 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. *Ann. I. Statist. Math.*, 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*, 3, 1244–1249.
- Bennett, K. P., Hu, J., Ji, X., Kunapuli, G., & Pang, J. S. (2006). Model selection via bilevel optimization. *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. (2002). Reduced Rank Kernel Ridge Regression”. *Neural Processing Letters*, 16, 293–302.
- 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.
- 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. *J. Am. Stat. Assoc.*, 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. *J. Amer. Statist. Assoc.*, 94, 154–163.
- Härdle, W. (1989). Resampling for inference from curves. In *Proceedings of the 47th Session of International Statistical Institute* (pp. 59–69).
- Härdle, W. (1991). *Smoothing techniques with implementation in S*. Springer-Verlag, New York.



- 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)*, 3316, 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 Trans. Neural Networks*, 18(3), 685–697.
- Joachims, T. (1999). *Making large-scale SVM practical*. In B. Schölkopf, C. J. C. Burges & A. J. Smola (Eds.), *Advances in Kernel Methods: Support Vector Learning* (pp. 169–184). Cambridge: MIT Press.
- 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. & Shevade, S. K. (2003). SMO algorithm for least-squares SVM formulations. *Neural computation*, 15(2), 487–507.
- Keerthi, S. S., Chapelle, O. & DeCoste, D. (2008). Building Support Vector Machines with Reduced Classifier Complexity. *J. Mach. Learn. Res.*, 7, 1493–1515.
- 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 Y.-H. Hu, J. Larsen, E. Wilson, & S. Douglas (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. *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 B. Schölkopf, C. J. C. Burges & A. J. Smola (Eds.), *Advances in Kernel Methods: Support Vector Learning* (pp. 185–208). Cambridge: MIT Press.
- 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 S. Haykin (Eds.), *Unsupervised Adaptive Filtering*, volume 1: Blind Source Separation (pp. 265–319). New York: John Wiley & Sons.
- 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. *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 L. Bottou, O. Chapelle, D. DeCoste & J. Weston (Eds.), *Large-Scale Kernel Machines* (pp. 175–202). MIT Press.

- Rényi, A. (1961). On measures of information and entropy. *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability* (pp. 547–561).
- Rényi, A. (2007). *Probability Theory*. Dover Publications.
- Rice, J. A. (1995). *Mathematical Statistics and Data Analysis*. (2nd ed). Duxbury Press.
- Rudemo, M. (1982). Empirical Choice of Histograms and Kernel Density Estimators. *Scand. J. Stat.*, 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.
- Schwartz, G. (1979). Estimating the dimension of a model. *Ann. of Stat.*, 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. & 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.
- Sheather, S. J. (2004). Density Estimation. *Statistical Science*, 19(4), 588–597.
- Silverman, B. W. (1978). Choosing the window width when estimating a density. *Biometrika*, 65, 1–11.
- Silverman, B. W. (1986). *Density Estimation for Statisticians and Data Analysis*. (1st ed.). Chapman & Hall.
- 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).
- Schölkopf, B., Williamson, R. C. & Bartlett, P. L. (2000). New Support Vector Algorithms. *Neural Computation*, 12, 1207–1245.
- Suykens, J. A. K. & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3), 293–300.

- 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., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least Squares Support Vector Machines*. Singapore: World Scientific.
- 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., Yalcin, M. E., & Vandewalle, J. (2003). Coupled chaotic simulated annealing processes. *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*. New York: Springer-Verlag.
- Vapnik, V. & Chapelle, O. (2000). Bounds on error expectation for support vector machines. *Neural Computation*, 12(9), 2013–2036.
- Vapnik, V. N. (1999). *Statistical Learning Theory*. John Wiley & Sons, Inc.
- 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 P. J. Bartlett, B. Schölkopf, D. Schuurmans & A. J. Smola (Eds.), *Advances in Large Margin Classifiers* (pp. 297–309). Cambridge: MIT Press.
- 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. *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.
- 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)*, 3, 494–497.
- Zhao, Y. & Sun, J. (2008). Recursive Reduced Least Squares Support Vector Machines. *Pattern Recognition*, 42, 837–842.