Fast Exact Leave-One-Out Cross-Validation of Sparse Least-Squares Support Vector Machines

Gavin C. Cawley [a,*] Nicola L. C. Talbot [a]

[a] *School of Computing Sciences, University of East Anglia, Norwich, United Kingdom, NR4 7TJ*

---

* Corresponding author, tel: +44 (0)1603 593258, fax: +44 (0)1603 592245, email: gcc@cmp.uea.ac.uk

Abstract

Leave-one-out cross-validation has been shown to give an almost unbiased estimator of the generalisation properties of statistical models, and therefore provides a sensible criterion for model selection and comparison. In this paper we show that exact leave-one-out cross-validation of sparse Least-Squares Support Vector Machines (LS-SVMs) can be implemented with a computational complexity of only $\mathcal{O}(\ell n^2)$ floating point operations, rather than the $\mathcal{O}(\ell^2 n^2)$ operations of a naïve implementation, where ℓ is the number of training patterns and n is the number of basis vectors. As a result, leave-one-out cross-validation becomes a practical proposition for model selection in large scale applications. For clarity the exposition concentrates on sparse least-squares support vector machines in the context of non-linear regression, but is equally applicable in a pattern recognition setting.

Key words:

model selection, cross-validation, least-squares support vector machine

1 Introduction

The generalisation properties of statistical models are typically governed by a small set of parameters, for instance regularisation parameters controlling the bias-variance trade-off; finding the optimal values for these parameters is an activity known as "model selection". Model selection frequently seeks to optimise a cross-validation estimate of an appropriate statistic measuring generalisation ability on unseen data, for instance misclassification rate in the case of pattern recognition or the sum-of-squares error in a regression setting. A k-fold cross-validation procedure partitions the available data into k disjoint subsets. k models are then trained, each model being trained on a different combination of $k-1$ of the k subsets and the test statistic evaluated over the remaining partition. The mean of the test statistic for each of the k models is known as the cross-validation estimate of the test statistic. The most extreme form of cross-validation, such that k is given by the number of training patterns, ℓ, is known as leave-one-out cross-validation. Leave-one-out cross-validation provides an sensible model selection criterion as it has been shown to provide an almost unbiased estimate of the true generalisation ability of the model (Lemma 1). Empirical studies have shown that in some cases model selection based on k-fold cross-validation out performs selection procedures based on the leave-one-out estimator as the latter is known to exhibit a comparatively high variance, e.g. [1]. However, bounds on, or approximations to the leave-one-out estimator have proved effective criteria for model selection for conventional support vector machines, e.g. [2–4].

Lemma 1 (Luntz and Brailovsky [5])

Given a sequence of observations $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^{\ell}$*, forming an independent and*

identically distributed (i.i.d.) sample from an underlying distribution $P(\boldsymbol{z})$*, and a hypothesis class* $\mathcal{H}$*, the empirical risk can be written as*

$$R_{\text{emp}}(\mathcal{H}, \mathcal{Z}) = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(\boldsymbol{z}_i, h \mid \mathcal{Z}), \quad h \in \mathcal{H}, \quad \boldsymbol{z}_i \in \mathcal{Z},$$

where $Q(\boldsymbol{z}, h \mid \mathcal{Z})$ *measures the loss for pattern* $\boldsymbol{z}$ *of the optimal hypothesis* $h \in \mathcal{H}$ *formed on the basis on the set of observations* $\mathcal{Z}$*, denoted by* $h \mid \mathcal{Z}$*. Let* $\mathcal{L}(\mathcal{H}, \mathcal{Z})$ *be the corresponding leave-one-out estimator,*

$$\mathcal{L}(\mathcal{H}, \mathcal{Z}) = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(\boldsymbol{z}_i, h_i \mid \mathcal{Z} \setminus \{\boldsymbol{z}_i\}), \quad h_i \in \mathcal{H}, \quad \boldsymbol{z}_i \in \mathcal{Z},$$

then the leave-one-out cross-validation estimate of the risk of the hypothesis class $\mathcal{H}$ *on data* $\mathcal{Z}$*, with respect to* $P(\boldsymbol{z})$*,*

$$R(\mathcal{H}, \mathcal{Z}) = \int Q(\boldsymbol{z}, h \mid \mathcal{Z}) dP(\boldsymbol{z}), \quad h \in \mathcal{H},$$

is almost unbiased in the sense that

$$E\left\{\mathcal{L}(\mathcal{H}, \mathcal{Z})\right\} = E\{R(\mathcal{H}, \mathcal{Z}')\},$$

where the expectations, $E\{\cdot\}$*, are taken over i.i.d. samples* $\mathcal{Z}, \mathcal{Z}' \sim P(\boldsymbol{z})$ *of size* ℓ *on the left hand side and* $\ell - 1$ *on the right* [5, 6].

Regularisation Machines [7], Kernel Ridge-Regression [8] and the Least-Squares Support Vector Machine (LS-SVM) [9, 10] comprise a family of closely related techniques constructing a linear model in a kernel-induced feature space minimising a regularised sum-of-squares loss functional. These methods provide an attractive approach for non-linear pattern recognition and regression problems as the optimal model parameters are given by the solution of a simple system of linear equations. The use of formal regularisation [11] also provides convenient control of the bias-variance trade-off [12] necessary for adequate

generalisation in non-trivial applications. In this paper, we demonstrate that the computational complexity of leave-one-out cross-validation of a sparse formulation of the least squares support vector machine is of the same order as that of the basic training algorithm, i.e. $\mathcal{O}(\ell^3)$. Exact leave-one-out cross-validation therefore provides a feasible criterion for model selection in large scale applications.

The remainder of this paper is structured as follows: Section 2 introduces the sparse least-squares support vector machine and describes a computationally efficient method for performing leave-one-out cross-validation for this class of kernel machines. Section 3 presents results obtained using the proposed method on two real-world non-linear regression tasks, based on well known, public domain benchmark datasets. Section 4 discusses the origins of the proposed procedure and suggests some possible extensions. Finally, the work is summarised in Section 5.

2 Method

In this section, we first describe the least-squares support vector machine as a kernel form of ridge regression [13], before introducing a sparse formulation with an efficient training algorithm for large scale applications. A computationally efficient algorithm for leave-one-out cross-validation is then proposed, based on a corollary of the familiar Sherman-Woodbury-Morrison formula.

2.1 Ridge Regression

Ridge regression [13] is a method from classical statistics that implements a regularised form of least-squares regression. Given training data,

$$\mathcal{D} = \{\boldsymbol{x}_i, t_i\}_{i=1}^{\ell}, \quad \boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d, \quad t_i \in \mathcal{T} \subset \mathbb{R},$$

ridge regression determines the weight vector, $\boldsymbol{w} \in \mathbb{R}^d$, and bias, b, of a linear model, $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$, via minimisation of the following objective function:

$$L(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \gamma \sum_{i=1}^{\ell} (t_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b)^2. \tag{1}$$

The objective function used in ridge regression (1) implements form of Tikhonov regularisation [11] of a sum-of-squares error metric, where γ is a regularisation parameter controlling the bias-variance trade-off [12, 13]. This corresponds to penalised maximum likelihood estimation of $\boldsymbol{w}$, assuming the targets have been corrupted by an independent and identically distributed (i.i.d.) sample drawn from a Gaussian noise process, with zero mean and fixed variance σ^2,

$$t_i = \boldsymbol{w} \cdot \boldsymbol{x}_i + b + \epsilon_i, \quad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

2.2 The Least-Squares Support Vector Machine

A non-linear form of ridge regression [8], known as the Least-Squares Support Vector Machine (LS-SVM) [9, 10], can be obtained via the so-called "kernel trick", whereby a linear ridge regression model is constructed in a feature space, $\mathcal{F}$ ($\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$), induced by a Mercer kernel defining the inner product $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')$. The kernel function, $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ may be any positive definite kernel. For an introduction to kernel methods in ma-

chine learning see Cristianini and Shawe-Taylor [14]. The objective function minimised in constructing a least-squares support vector machine is given by

$$L(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \gamma \sum_{i=1}^{\ell} (t_i - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x_i}) - b)^2.$$

The representer theorem [15] indicates that the solution of an optimisation problem of this nature can be written in the form of an expansion over training patterns, i.e. $\boldsymbol{w} = \sum_{i=1}^{\ell} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)$. The output of the least-squares support vector machine is then given by the kernel expansion

$$f(\boldsymbol{x}) = \sum_{i=1}^{\ell} \alpha_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

It can easily be shown [8, 9] that the optimal coefficients of this expansion are given by the solution of a set of linear equations

$$\begin{bmatrix} \boldsymbol{K} + \gamma^{-1}\boldsymbol{I} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{t} \\ 0 \end{bmatrix},$$

where $\boldsymbol{K} = [k_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^{\ell}$, $\boldsymbol{t} = (t_1, t_2, \ldots, t_\ell)^T$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_\ell)^T$ and $\mathbf{1} = (1, 1, \ldots, 1)^T$.

2.3 The Sparse Least-Squares Support Vector Machine

Unfortunately, unlike the original support vector machine [16, 17], the kernel expansions defining least-squares support vector machines are typically fully dense, i.e. $\alpha_i \neq 0, \ \forall \ i \in \{1, 2, \ldots, \ell\}$. This severely limits the utility of the LS-SVM in large scale applications as the complexity of the basic training algorithm is of order $\mathcal{O}(\ell^3)$ (although more efficient methods have been proposed, e.g. [18]). Fortunately a range of methods for obtaining sparse least-

squares support vector machines are available, see e.g. [19–21]. The sparsity of the original support vector machine is perhaps best viewed as a convenient consequence of the use of the hinge loss, instead of the least-squares loss, in defining the primal optimisation problem, rather than as a deliberate design aim of the algorithm. As a result, the kernel expansions for support vector machine in general are not maximally sparse either and so several algorithms have been proposed for simplifying existing support vector expansions [22–24], or for training support vector machines using a reduced set of candidate support vectors [25]. The latter is similar in spirit to the sparse least-squares support vector machine used here.

In order to construct a sparse least-squares support vector machine, it is assumed that the weight vector, $\boldsymbol{w}$, can be represented as a weighted sum of a limited number of basis vectors $\{\boldsymbol{\phi}(\boldsymbol{z}_i)\}_{i=1}^{n},\ \boldsymbol{z} \in \mathcal{X} \subset \mathbb{R}^d,\ n \ll \ell$, such that

$$\boldsymbol{w} = \sum_{i=1}^{n} \beta_i \boldsymbol{\phi}(\boldsymbol{z}_i).$$

The objective function minimised by the sparse least-squares support vector machine then becomes

$$L(\boldsymbol{\beta}, b) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j k_{ij} + \gamma \sum_{i=1}^{\ell} (t_i - \sum_{j=1}^{n} \beta_j \hat{k}_{ij} - b)^2, \tag{2}$$

where $k_{ij} = \mathcal{K}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ and $\hat{k}_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{z}_j)$. It is straightforward to show that this optimisation problem can again be expressed in the form of a system of linear equations (see appendix A.3),

$$(\boldsymbol{R} + \boldsymbol{Z}^T \boldsymbol{Z})\boldsymbol{p} = \boldsymbol{Z}^T \boldsymbol{t} \tag{3}$$

where $\boldsymbol{p} = (\boldsymbol{\beta}, b)^T$ is the vector of model parameters, $\boldsymbol{Z} = [\hat{\boldsymbol{K}}\ \boldsymbol{1}]$, $\hat{\boldsymbol{K}} = [\hat{k}_{ij}]_{i=1,j=1}^{i=\ell,j=n}$, $\boldsymbol{K} = [k_{ij}]_{i,j=1}^{n}$, $\boldsymbol{1} = [1]_{i=1}^{\ell}$ and $\boldsymbol{R}$ corresponds to the regularisation

term in (2)

$$\boldsymbol{R} = \begin{bmatrix} \frac{1}{2}\gamma^{-1}\boldsymbol{K} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}.$$

Note that the computational complexity of the resulting training algorithm is of order $\mathcal{O}(\ell n^2)$, being dominated by construction of the $(n+1) \times (n+1)$ matrix $(\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Z})$, and therefore is far better suited to large scale application.

2.4 Selection of Reference Vectors

There are a number of viable approaches to the problem of selecting a set of reference vectors: Baudat and Anouar [26] propose selection of a minimal set of training vectors forming a basis for the subspace of $\mathcal{F}$ populated by the data (for alternative approaches, see [19–21, 27]). Suykens *et al.* [9, 10] iteratively prune patterns having a small error from the kernel expansion (note however that this involves starting with a fully dense least-squares support vector machine, which is computationally demanding for large ℓ). Alternatively, one could simply select a random subsample of the training data, or even the entire training set. For this study, we adopt the method of Baudat and Anouar, as summarised in the following section.

2.4.1 Feature Vector Selection

The normalised Euclidean distance between the image of a datum, $\boldsymbol{x}$, in feature space, $\boldsymbol{\phi}(\boldsymbol{x})$, and $\hat{\boldsymbol{\phi}}_{\mathcal{S}}(\boldsymbol{x})$, it's optimal reconstruction using a sub-set of the training vectors $\{\boldsymbol{\phi}(\boldsymbol{x}_i)\}_{i\in\mathcal{S}}$, is given by

$$\delta_i(\mathcal{S}) = \frac{\|\boldsymbol{\phi}(\boldsymbol{x_i}) - \hat{\boldsymbol{\phi}}_{\mathcal{S}}(\boldsymbol{x}_i)\|^2}{\|\boldsymbol{\phi}(\boldsymbol{x}_i)\|^2}. \tag{4}$$

This distance can be expressed in terms of inner products, and so via the "kernel trick",

$$\delta_i(\mathcal{S}) = 1 - \frac{\boldsymbol{K}_{\mathcal{S}i}^T \boldsymbol{K}_{\mathcal{S}\mathcal{S}}^{-1} \boldsymbol{K}_{\mathcal{S}i}}{k_{ii}}.$$

where $\boldsymbol{K}_{\mathcal{S}\mathcal{S}}$ is a square sub-matrix of the kernel matrix $\boldsymbol{K} = [k_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^{\ell}$, such that $\boldsymbol{K}_{\mathcal{S}\mathcal{S}} = \{k_{ij}\}_{i,j\in\mathcal{S}}$ and $\boldsymbol{K}_{\mathcal{S}i} = (k_{ji})_{j\in\mathcal{S}}^T$ is a column vector of inner products in $\mathcal{F}$. To form a basis for the subspace in $\mathcal{F}$ populated by the data, it is sufficient to minimise the mean reconstruction error δ_i over all patterns in the training set [26], i.e. maximise

$$J(\mathcal{S}) = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{\boldsymbol{K}_{\mathcal{S}i}^T \boldsymbol{K}_{\mathcal{S}\mathcal{S}}^{-1} \boldsymbol{K}_{\mathcal{S}i}}{k_{ii}}. \tag{5}$$

Starting with $\mathcal{S} = \emptyset$, a basis is constructed in a greedy manner, at each step adding the training vector maximising $J(\mathcal{S})$. Note that provided the feature vectors currently selected are linearly independent $\boldsymbol{K}_{\mathcal{S}\mathcal{S}}$ will be invertible. This will always be the case; clearly one would not select a linearly dependent feature vector as this would not result in a reduction in the mean reconstruction error. The algorithm therefore terminates when $\boldsymbol{K}_{\mathcal{S}\mathcal{S}}$ is no longer invertible, indicating that a complete basis has been identified, or when an upper bound on the number of basis vectors is reached, or when the mean relative reconstruction error falls below a given threshold. An efficient implementation of the feature vector selection algorithm is given in [28]. The computational complexity of the standard feature vector selection is $\mathcal{O}(\ell^2|\mathcal{S}|^2)$, where $|\mathcal{S}|$ is the number of feature vectors extracted. The computational complexity of the improved algorithm is only $\mathcal{O}(|\mathcal{S}|^3)$, however this is achieved at the expense of introducing a stochastic element, such that there is no guarantee that the same set of feature vectors will be extracted each time the algorithm is executed. However, provided the feature vectors extracted (approximately) span the subspace in $\mathcal{F}$ occupied by the training data, the mapping implemented

by the sparse LS-SVM will be (approximately) the same as that of the full LS-SVM. The storage complexity of both algorithms is $\mathcal{O}(|\mathcal{S}|^2)$, being dominated by storage of $\boldsymbol{K}_{\mathcal{SS}}$.

2.5 Fast Exact Leave-One-Out Cross-Validation

At each step of the leave-one-out cross-validation procedure, a sparse least-squares support vector machine is constructed excluding a single training pattern from the data. The vector of model parameters, $\boldsymbol{p}_{(i)}$ at the i^{th} iteration is then given by the solution of the following system of linear equations,

$$\boldsymbol{p}_{(i)} = \begin{bmatrix} \boldsymbol{\beta}_{(i)} \\ \\ b_{(i)} \end{bmatrix} = \left[\boldsymbol{R} + \boldsymbol{Z}_{(i)}^T \boldsymbol{Z}_{(i)}\right]^{-1} \boldsymbol{Z}_{(i)}^T \boldsymbol{t}$$

where $\boldsymbol{Z}_{(i)}$ is the sub-matrix formed by omitting the i^{th} row of $\boldsymbol{Z}$. Normally the most computationally expensive step is the inversion of the matrix $\boldsymbol{C}_{(i)} = \left[\boldsymbol{R} + \boldsymbol{Z}_{(i)}^T \boldsymbol{Z}_{(i)}\right]$, with a complexity of $\mathcal{O}(n^3)$ operations. Fortunately $\boldsymbol{C}_{(i)}$ can be written as a rank one modification of a matrix $\boldsymbol{C}$,

$$\boldsymbol{C}_{(i)} = \left[\boldsymbol{R} + \boldsymbol{Z}^T \boldsymbol{Z} - \boldsymbol{z}_i \boldsymbol{z}_i^T\right] = \left[\boldsymbol{C} - \boldsymbol{z}_i \boldsymbol{z}_i^T\right],$$

where $\boldsymbol{z}_i$ is the i^{th} row of $\boldsymbol{Z}$ and $\boldsymbol{C} = \boldsymbol{R} + \boldsymbol{Z}^T \boldsymbol{Z}$. The following matrix inversion lemma, a corollary of the familiar Sherman-Woodbury-Morrison formula, can then be used to find $\boldsymbol{C}_{(i)}^{-1}$ in only $\mathcal{O}(n^2)$ operations, given that $\boldsymbol{C}^{-1}$ is already known.

Lemma 2 (Bartlett [29]) *Given an invertible matrix $\boldsymbol{A}$ and column vector*

$\boldsymbol{v}$, then assuming $1 + \boldsymbol{v}^T \boldsymbol{A}^{-1} \boldsymbol{v} \neq 0$,

$$\left(\boldsymbol{A} + \boldsymbol{v}\boldsymbol{v}^T\right)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{v}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{v}}.$$

This can be obtained straight-forwardly as a special case of the familiar Sherman-Woodbury-Morrison formula (see appendix A.2).

The solution to the matrix inversion problem can then be written as

$$\boldsymbol{C}_{(i)}^{-1} = [\boldsymbol{C} - \boldsymbol{z}_i \boldsymbol{z}_i^T]^{-1} = \boldsymbol{C}^{-1} + \frac{\boldsymbol{C}^{-1}\boldsymbol{z}_i\boldsymbol{z}_i^T\boldsymbol{C}^{-1}}{1 - \boldsymbol{z}_i^T\boldsymbol{C}^{-1}\boldsymbol{z}_i}.$$

The computational complexity of the leave-one-out cross-validation process is thus reduced to only $\mathcal{O}(\ell n^2)$ operations, which is the same as that of the basic training algorithm for the sparse least-squares support vector machine.

2.6 A Further Refinement

For the purposes of model selection based on leave-one-out cross-validation, we are not generally interested in the values of the model parameters themselves, but only in the predicted output of the model for the training pattern excluded during each trial. For instance we might adopt Allen's PRESS (predicted residual sum of squares) statistic [30],

$$\text{PRESS} = \sum_{i=1}^{\ell} \left\{\boldsymbol{e}_{(i)}\right\}_i^2,$$

where $\left\{\boldsymbol{e}_{(i)}\right\}_i = t_i - \left\{\boldsymbol{y}_{(i)}\right\}_i$ is the residual for the i^{th} training pattern during the i^{th} iteration of the leave-one-out cross-validation procedure. Fortunately it is possible to compute these residuals without explicitly evaluating the model parameters in each trial. It is relatively straight-forward to show that

$$\left\{\boldsymbol{e}_{(i)}\right\}_i = \frac{e_i}{1 - h_{ii}},$$

(see appendix A.4) where $e_i = t_i - y_i$ is the residual for the i^{th} training pattern for a sparse least-squares support vector machine trained on the entire dataset, $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{C}^{-1}\boldsymbol{Z}^T$ is the *hat*, or *smoother*, matrix [31, 32] of which h_{ii} the i^{th} element of the leading diagonal [33]. Allen's PRESS statistic can therefore be evaluated in closed form without explicit inversion of $\boldsymbol{C}_{(i)}$, again with a computational complexity of only $\mathcal{O}(\ell n^2)$.

3 Results

This section presents results obtained using conventional and fast leave-one-out cross-validation procedures for two well-studied non-linear regression benchmark tasks: the Motorcycle and Boston Housing data sets. The Motorcycle data set, being a univariate regression task is easily visualised, whereas the Boston Housing data set is used to illustrate the scaling properties of the proposed procedure. All experiments were performed using the MATLAB programming environment on a personal computer with dual 1.7GHz Xeon processors and 4Gb of RDRAM (400MHz bus) running under the Linux operating system.

3.1 The Motorcycle Dataset

The Motorcycle benchmark consists of a sequence of accelerometer readings through time following a simulated motor-cycle crash during an experiment to determine the efficacy of crash-helmets [34]. Figure 1 shows conventional and sparse support vector machine models of the motorcycle dataset, using a

Gaussian radial basis function (RBF) kernel,

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left\{-\sigma^{-2}\|\boldsymbol{x} - \boldsymbol{x}'\ |^2\right\}.$$

The hyper-parameters were selected to minimise a 10-fold cross-validation estimate of the test sum-of-squares error using a straight-forward Nelder-Mead simplex algorithm [35] ($\gamma = 1.2055 \times 10^{-3}$ and $\sigma^{-2} = 13.1$). Note that the sparse model is functionally identical to the full least-squares support vector machine model with only 15 basis vectors comprising the sparse kernel expansion. The Gram matrix for an RBF kernel is of full-rank [36], and so it is not generally possible to form a complete sparse basis for the subspace of $\mathcal{F}$ inhabited by the training data $\mathcal{D}$. In this case however, only 15 basis vectors can be extracted before the condition of $\boldsymbol{K}_{\mathcal{SS}}$ deteriorates to the point at which it is no longer invertible, indicating that these vectors form an extremely close approximation to a complete basis.

Figure 2 illustrates the leave-one-out cross-validation behaviour of the sparse least-squares support vector machine, i.e. the predicted value for each datapoint comprising the training data using a model trained only on the remaining datapoints. Note that the behaviour computed using the proposed fast algorithm (depicted by the diamonds) is essentially identical to that obtained by explicit leave-one-out cross-validation (depicted by crosses). The fast leave-one-out cross-validation algorithm is however more than two orders of magnitude faster than the conventional approach for this dataset, as shown in table 1. Let the relative sum-of-squares error be defined as

$$E_r = \frac{\|\boldsymbol{e} - \hat{\boldsymbol{e}}\|}{\|\boldsymbol{e}\|}, \tag{6}$$

where $\boldsymbol{e}$ is the vector of predicted (leave-one-out) residuals computed explic-

itly and $\hat{\boldsymbol{e}}$ is the vector of predicted residuals computed via the proposed efficient method. The relative sum-of-squares error for the Motorcycle data set is negligible ($E_r = 7.6 \times 10^{-6}$).

3.2 The Boston Housing Dataset

The Boston housing dataset describes the relationship between the median value of owner occupied homes in the suburbs of Boston and thirteen attributes representing environmental and social factors believed to be relevant [37]. The input patterns, $\{\boldsymbol{x}_i\}_{i=1}^{\ell}$, were first standardised, such that each element of the input vector has a zero mean and unit variance, allowing an isotropic Gaussian radial basis kernel (6) to be used. Again the hyper-parameters were chosen so as to minimise a 10-fold cross-validation estimate of the test sum-of-squares error ($\gamma = 2.89 \times 10^4$ and $\sigma^{-2} = 4.36$). Figure 3 shows the mean relative reconstruction error (4) as a function of the number of basis vectors identified by the feature vector selection algorithm. The mean relative reconstruction error is negligible if more than ≈ 200 basis vectors are used.

Explicit and fast leave-one-out cross-validation procedures were then evaluated for sparse least-squares regression networks based on the first 200 basis vectors identified by the feature vector selection algorithm. Figure 4 shows the mean run-time in seconds for both algorithms as a function of the number of training patterns, ℓ, over 20 trials. The fast leave-one-out cross-validation procedure is in all cases considerably faster than explicit evaluation, and has better scaling properties.

4 Discussion

The application of the Bartlett correction formula to the efficient evaluation of the leave-one-out cross-validation error of linear models minimising a least-squares criterion has been known for some time in the field of statistics [33]. The "kernel trick" allows this method to be extended to non-linear models formed via construction of a linear model in a feature space defined by a Mercer kernel. In principal it should be possible to apply a similar procedure to related kernel models, such as least-squares support vector classification networks [38], Kernel Fisher Discriminant (KFD) analysis [39, 40] and 2-norm support vector machines [6, 16, 17], where the Lagrange multipliers associated with the support vectors are given by the solution of a system of linear equations corresponding to the Karush-Kuhn-Tucker (KKT) conditions.

5 Summary

In this paper we have shown that the Bartlett correction formula can be used to implement exact leave-one-out cross-validation of sparse least-squares support vector regression networks with a computational complexity of only $\mathcal{O}(\ell n^2)$, rather than the $\mathcal{O}(\ell^2 n^2)$ of a naïve direct implementation. Since the computational complexity of the basic training algorithm is also $\mathcal{O}(\ell n^2)$, leave-one-out cross-validation becomes a practical criterion for model selection for this class of kernel machines, as demonstrated by experiments on the Motorcycle and Boston Housing benchmark data sets.

Acknowledgements

The authors would like to thank the anonymous reviewers, Danilo Mandic, Tony Bagnall and Rob Foxall for their helpful comments on previous drafts. This work was supported by Royal Society Research Grant RSRG-22270.

References

[1] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence (IJCAI)*, pages 1137–1143, San Mateo, CA, 1995. Morgan Kaufmann.

[2] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, September 2000.

[3] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[4] S. S. Keerthi. Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms. *IEEE Transactions on Neural Networks*, 13(5):1225–1229, September 2002.

[5] A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition (in russian). *Techicheskaya Kibernetica*, 3, 1969.

[6] V. N. Vapnik. *Statistical Learning Theory.* Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications and Control. Wiley, New York, 1998.

[7] T. Poggio and Girosi F. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.

[8] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings, 15th International Conference on Machine Learning*, pages 515–521, Madison, WI, July 24–27 1998.

[9] J. Suykens, L. Lukas, and J. Vandewalle. Sparse approximation using least-squares support vector machines. In *Proceedings, IEEE International*

Symposium on Circuits and Systems, pages 11757–11760, Geneva, Switzerland, May 2000.

[10] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines : robustness and sparse approximation. *Neurocomputing*, 48(1–4):85–105, October 2002.

[11] A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems*. John Wiley, New York, 1977.

[12] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilema. *Neural Computation*, 4(1):1–58, 1992.

[13] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[14] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, Cambridge, U.K., 2000.

[15] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

[16] B. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on computational learning theory*, pages 144–152, Pittsburgh, 1992. ACM.

[17] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[18] J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle. Least squares support vector machine classifiers : a large scale algorithm. In *Proceedings of the European Conference on Circuit Theory and Design (ECCTD-99)*, pages 839–842, Stresa, Italy, September 1999.

[19] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[20] G. C. Cawley and N. L. C. Talbot. A greedy training algorithm for sparse least-squares support vector machines. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-2002)*, volume 2415 of *Lecture Notes in Computer Science (LNCS)*, pages 681–686, Madrid, Spain, August 27–30 2002. Springer.

[21] G. C. Cawley and N. L. C. Talbot. Improved sparse least-squares support vector machines. *Neurocomputing*, 48:1025–1031, October 2002.

[22] C. J. C. Burges. Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, Bari, Italy, 1996.

[23] C. J. C. Burges and B. Schölkopf. Improving speed and accuracy of support vector learning machines. In *Advances in Neural Information Processing Systems*, volume 9, pages 375–381. MIT Press, 1997.

[24] T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, December 2001.

[25] Y.-J. Lee and O. L. Mangasarian. RSVM: reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining*, Chicago, IL, USA, April 5–7 2001.

[26] G. Baudat and F. Anouar. Kernel-based methods and function approximation. In *Proc. IJCNN*, pages 1244–1249, Washington, DC, July 2001.

[27] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine*

Learning, pages 911–918. Morgan Kaufmann, San Fransisco, CA, June 29 - July 2 2000.

[28] G. C. Cawley and N. L. C. Talbot. Efficient formation of a basis in a kernel induced feature space. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN-2002)*, pages 1–6, Bruges, Belgium, April 24–26 2002.

[29] M. S. Bartlett. An inverse matrix adjustment arising in disciminant analysis. *Annals of Mathematical Statistics*, 22(1):107–111, 1951.

[30] D. M. Allen. The relationship between variable selection and prediction. *Technometrics*, 16:125–127, 1974.

[31] S. Weisberg. *Applied Linear Regression*. Wiley, New York, 1985.

[32] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Texts in Statistics. Springer, 2001.

[33] R. D. Cook and S. Weisberg. *Residuals and Influence in Regression*. Monographs on Statistics and Applied Probability. Chapman and Hall, New York, 1982.

[34] B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society, B*, 47(1):1–52, 1985.

[35] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[36] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[37] D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal Environmental Economics and Management*, 5:81–102, 1978.

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

[39] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing*, volume IX, pages 41–48. IEEE Press, New York, 1999.

[40] J. Xu, X. Zhang, and Y. Li. Kernel MSE algorithm: A unified framework for KFD, LS-SVM and KRR. In *Proc. IJCNN*, pages 1486–1491, Washington, DC, July 2001.

[41] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics*, 20(4):621, 1949.

[42] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[43] M. Woodbury. Inverting modified matrices. Memorandum report 42, Princeton University, Princeton, U.S.A., 1950.

[44] W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, June 1989.

[45] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition edition, 1996.

A Proofs and Derivations

A.1 Proof of Lemma 1 (Luntz and Brailovsky [5])

Following the approach of Vapnik [6], the expectation of the leave-one-out estimator is simply the integral over all datasets, $\mathcal{Z}$, constituting an i.i.d sample from $P(\boldsymbol{z})$ of size ℓ:

$$E\{\mathcal{L}(\mathcal{H},\mathcal{Z})\} = \int \frac{1}{\ell}\sum_{i=1}^{\ell} Q(z_i, h_i|\mathcal{Z}\setminus\{z_i\})dP(z_1)\cdots dP(z_\ell).$$

Since the dataset is formed by an i.i.d. sample from $P(\boldsymbol{z})$, the i^{th} integral may be moved inside the summation:

$$= \int \frac{1}{\ell}\sum_{i=1}^{\ell}\left(\int Q(z_i, h_i|\mathcal{Z}\setminus\{z_i\})dP(z_i)\right)dP(z_1)\cdots dP(z_{i-1}dP(z_{i+1})\cdots dP(z_\ell).$$

Let $\mathcal{Z}'$ represent an i.i.d. sample from $P(\boldsymbol{z})$ of size $\ell-1$:

$$= E\left\{\frac{1}{\ell}\sum_{i=1}^{\ell} R(\mathcal{H},\mathcal{Z}')\right\},$$

and so

$$E\{\mathcal{L}(\mathcal{H},\mathcal{Z})\} = E\{R(\mathcal{H},\mathcal{Z}')\},$$

and the lemma is proven.

A.2 Proof of Lemma 2 (Bartlett [29])

The inverse of a block matrix,

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix},$$

can be found through block Gauss-Jordan elimination; beginning by manipulating the matrix into upper echelon form, we get

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1} \\ -\boldsymbol{S}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{S}^{-1} \end{bmatrix},$$

where $\boldsymbol{S} = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$. Equivalently, one could begin by manipulating the matrix into lower echelon form, giving

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{T}^{-1} & -\boldsymbol{T}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{T}^{-1} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{T}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \end{bmatrix},$$

where $\boldsymbol{T} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$. Equating the upper left element of the matrices on the right hand sides of the two preceding expressions gives,

$$\boldsymbol{T}^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1},$$

i.e.,

$$(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}. \tag{A.1}$$

Equation A.1 is known as the Sherman-Woodbury-Morrison formula [41–45]. An important special case results from the assumption that $\boldsymbol{D}$ is the identity matrix, $\boldsymbol{I}$, and $\boldsymbol{B}$ and $\boldsymbol{C}$ are column vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively [29]:

$$(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} + \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}.$$

A.3 Derivation of Training Procedure for the Sparse LS-SVM

The objective function minimised by the sparse least-squares support vector machine is given by

$$L(\boldsymbol{\beta}, b) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j k_{ij} + \gamma\sum_{i=1}^{\ell}(t_i - \sum_{j=1}^{n}\beta_j\hat{k}_{ij} - b)^2,$$

where $k_{ij} = \mathcal{K}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ and $\hat{k}_{ij} = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{z}_j)$. Setting the partial derivatives of the objective function with respect to $\boldsymbol{\beta}$ and b to zero, and dividing through by 2γ, yields:

$$\sum_{i=1}^{n}\beta_i\sum_{j=1}^{\ell}\hat{k}_{ji} + b = \sum_{j=1}^{\ell}t_j$$

and $\forall\ r \in \{1, 2, \ldots, n\}$,

$$\sum_{i=1}^{n}\beta_i\left(\frac{1}{2}\gamma^{-1}k_{ir} + \sum_{j=1}^{\ell}\hat{k}_{jr}\hat{k}_{ji}\right) + b\sum_{i=1}^{\ell}\hat{k}_{ir} = \sum_{i=1}^{\ell}t_i\hat{k}_{ir},$$

These equations can be expressed as a system of $n+1$ linear equations in $n+1$ unknowns,

$$\begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^T & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{\beta} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \sum_{k=1}^{\ell}t_k \end{bmatrix}, \tag{A.2}$$

where the (i, j)th element of the sub-matrix $\boldsymbol{\Omega}$ is:

$$\omega_{ij} = \frac{1}{2}\gamma^{-1}k_{ij} + \sum_{l=1}^{\ell}\hat{k}_{lj}\hat{k}_{li} \qquad 1 \le i, j \le n,$$

the jth element of the vector $\boldsymbol{\Phi}$ is:

$$\phi_j = \sum_{i=1}^{\ell}\hat{k}_{ij} \qquad 1 \le j \le n$$

and the jth element of the vector $\boldsymbol{c}$ is:

$$c_j = \sum_{i=1}^{\ell}t_i\hat{k}_{ij} \qquad 1 \le j \le n.$$

This matrix equation can be simplified by referring to the vector consisting of all the model parameters as $\boldsymbol{p} = (\boldsymbol{\beta}, b)^T$, and by defining the matrices $\boldsymbol{R}$ and $\boldsymbol{Z}$ as:

$$\boldsymbol{R} = \begin{bmatrix} \frac{1}{2\gamma}\boldsymbol{K} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$$

$$\boldsymbol{Z} = [\hat{\boldsymbol{K}} \quad \mathbf{1}]$$

where the (i,j)th element of $\boldsymbol{K}$ is k_{ij}, and the (i,j)th element of $\hat{\boldsymbol{K}}$ is $\hat{k}_{ij}$. Equation (A.2) can then be written as:

$$(\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Z})\boldsymbol{p} = \boldsymbol{Z}^T\boldsymbol{t}. \tag{A.3}$$

A.4 Derivation of Closed Form Expression for Predicted Residuals

From e.g. Equation (A.3) we know that the vector of model parameters $p = (\boldsymbol{\beta}, b)$ is given by

$$\boldsymbol{p} = (\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{t}$$

where $\boldsymbol{Z} = [\hat{\boldsymbol{K}} \ \mathbf{1}]$. For convenience, let $\boldsymbol{C} = \boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Z}$ and $\boldsymbol{d} = \boldsymbol{Z}^T\boldsymbol{t}$, such that $\boldsymbol{p} = \boldsymbol{C}^{-1}\boldsymbol{d}$. Furthermore, let $\boldsymbol{Z}_{(i)}$ and $\boldsymbol{t}_{(i)}$ represent the data with the i^{th} observation deleted, then

$$\boldsymbol{C}_{(i)} = \boldsymbol{C} - \boldsymbol{z}_i\boldsymbol{z}_i^T,$$

and

$$\boldsymbol{d}_{(i)} = \boldsymbol{d} - t_i\boldsymbol{z}_i.$$

The Bartlett matrix inversion formula then gives

$$\boldsymbol{C}_{(i)}^{-1} = \boldsymbol{C} + \frac{\boldsymbol{C}^{-1}\boldsymbol{z}_i\boldsymbol{z}_i^T\boldsymbol{C}^{-1}}{1 - \boldsymbol{z}_i^T\boldsymbol{C}^{-1}\boldsymbol{z}_i},$$

such that the vector of model parameters during the i^{th} iteration of the leave-one-out cross-validation procedure becomes

$$\boldsymbol{p}_{(i)} = \left(\boldsymbol{C} + \frac{\boldsymbol{C}^{-1}\boldsymbol{z}_i\boldsymbol{z}_i^T\boldsymbol{C}^{-1}}{1 - \boldsymbol{z}_i^T\boldsymbol{C}^{-1}\boldsymbol{z}_i} \right) (\boldsymbol{d} - t_i\boldsymbol{z}_i).$$

Let $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{C}^{-1}\boldsymbol{Z}^T$ represent the *hat* matrix; note that the i^{th} element of the leading diagonal can be written $h_{ii} = \boldsymbol{z}_i^T\boldsymbol{C}^{-1}\boldsymbol{z}_i$, so expanding the brackets we have

$$\begin{aligned}\boldsymbol{p}_{(i)} =& \boldsymbol{C}^{-1}\boldsymbol{d} - \boldsymbol{C}^{-1}t_i\boldsymbol{z}_i + \frac{\boldsymbol{C}^{-1}\boldsymbol{z}_i\boldsymbol{z}_i^T\boldsymbol{C}^{-1}}{1 - \boldsymbol{z}_i^T\boldsymbol{C}^{-1}\boldsymbol{z}_i}\boldsymbol{d} - \frac{\boldsymbol{C}^{-1}\boldsymbol{z}_i\boldsymbol{z}_i^T\boldsymbol{C}^{-1}}{1 - \boldsymbol{z}_i^T\boldsymbol{C}^{-1}\boldsymbol{z}_i}t_i\boldsymbol{z}_i \\ =& \boldsymbol{p} + \left(\frac{\boldsymbol{z}_i^T\boldsymbol{p} - t_i}{1 - h_{ii}} \right) \boldsymbol{C}^{-1}\boldsymbol{z}_i.\end{aligned}$$

The residual error for the i^{th} training pattern is $e_i = t_i - \boldsymbol{z}_i^T\boldsymbol{p}$ and so

$$\boldsymbol{p}_{(i)} = \boldsymbol{p} - \frac{e_i}{1 - h_{ii}}\boldsymbol{C}^{-1}\boldsymbol{z}_i.$$

Noting that $\boldsymbol{y} = \boldsymbol{Z}\boldsymbol{p}$, the output of the model during the i^{th} iteration of the leave-one-out cross-validation procedure can be written as

$$\boldsymbol{y}_{(i)} = \boldsymbol{Z}\boldsymbol{p}_{(i)} = \boldsymbol{y} - \frac{e_i}{1 - h_{ii}}\boldsymbol{h}_i$$

where $\boldsymbol{h}_i$ is the i^{th} column of $\boldsymbol{H}$. The vector of residuals for the training patterns during the leave-one-out cross-validation procedure can then be written in closed form as

$$\boldsymbol{e}_{(i)} = \boldsymbol{t} - \boldsymbol{y}_{(i)} = \boldsymbol{e} + e_i\frac{1}{1 - h_{ii}}\boldsymbol{h}_i.$$

The i^{th} element of $\boldsymbol{e}_{(i)}$ is given by

$$\{\boldsymbol{e}_{(i)}\}_i = \frac{1}{1 - h_{ii}}e_i.$$

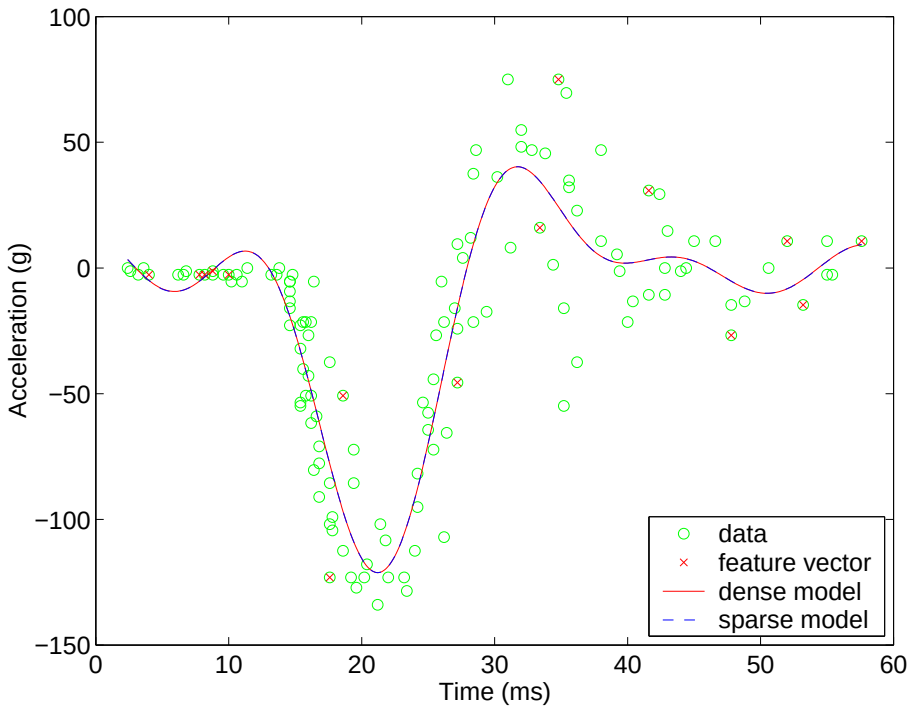
Figure Captions

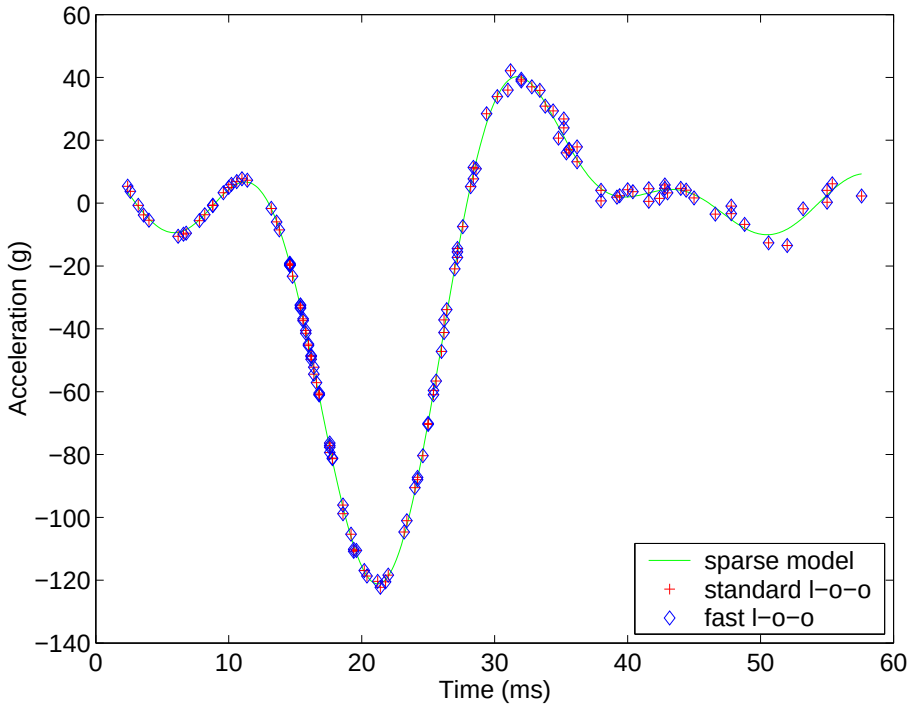
1 Dense and sparse least-squares support vector machine models of the motorcycle data set; note the standard and sparse models are essentially identical.

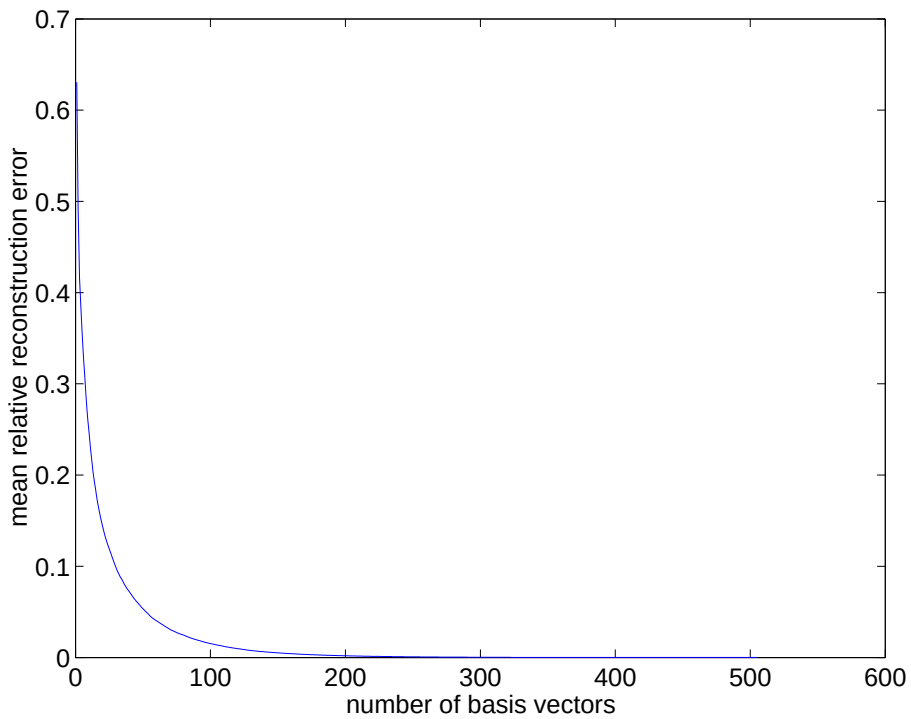
2 Leave-one-out cross-validation behaviour of a sparse least-squares support vector model of the Motorcycle benchmark dataset, using conventional and fast leave-one-out cross-validation algorithms.

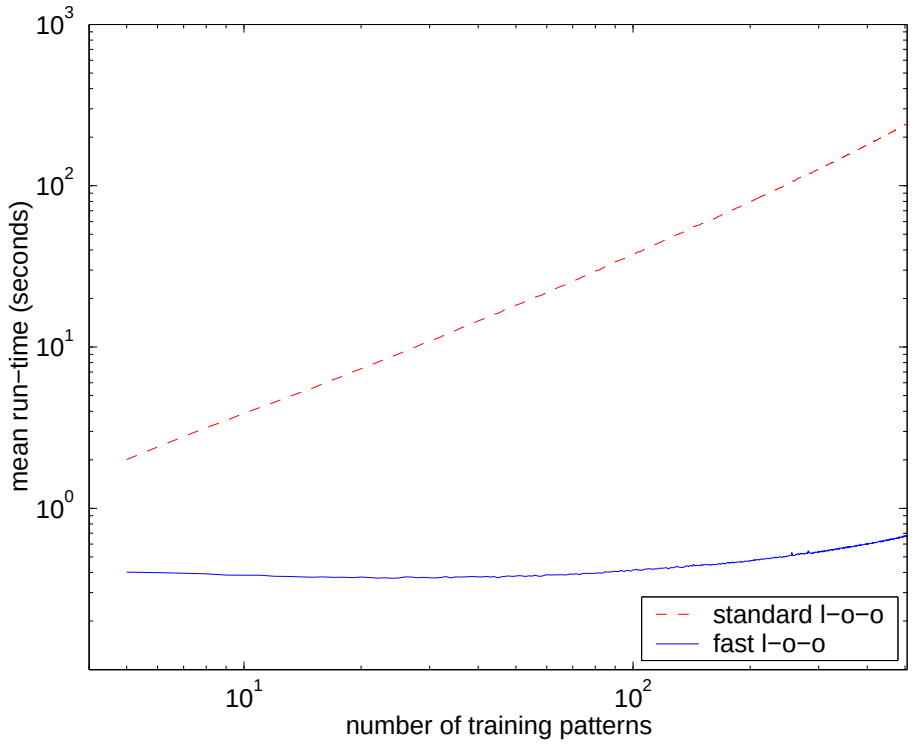
3 Mean relative reconstruction error as a function of the number of basis vectors identified by the efficient feature vector selection algorithm for the Boston Housing data set.

4 Mean run-time for explicit and fast leave-one-out cross-validation procedures as a function of the number of training patterns (averaged over 20 trials).


100
50
0
−50
−100
−150
0
10
20
30
40
50
60
Acceleration (g)
Time (ms)
data
feature vector
dense model
sparse model


60
40
20
0
−20
−40
−60
−80
−100
−120
−140
0
10
20
30
40
50
60
Acceleration (g)
Time (ms)
sparse model
standard l−o−o
fast l−o−o


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
mean relative reconstruction error
0
100
200
300
400
500
600
number of basis vectors


10^3
10^2
10^1
10^0
mean run−time (seconds)
10^1
10^2
number of training patterns
standard l−o−o
fast l−o−o

List of Tables

Algorithm	Mean Run Time (sec)	Std. Err.
Explicit	0.396950	1.212448×10^{-3}
Fast	0.003072	1.479775×10^{-5}