Simultaneous input variable and basis function selection for RBF networks

Jarkko Tikka *

Department of Information and Computer Science, Helsinki University of Technology, P.O. Box 5400, FIN-02015 HUT, Finland

Abstract

Input selection is advantageous in regression problems. For example, it might decrease the training time of models, reduce measurement costs, and circumvent problems of high dimensionality. Inclusion of useless inputs into the model increases also the likelihood of overfitting. Neural networks provide good generalization in many cases, but their interpretability is usually limited. However, selecting a subset of variables and estimating their relative importances would be valuable in many real world applications. In the present work, a simultaneous input and basis function selection method for the radial basis function (RBF) network is proposed. The selection is performed by minimizing a constrained cost function, in which sparsity of the network is controlled by two continuous valued shrinkage parameters. Each input dimension is weighted and the constraints are imposed on these weights and the output layer coefficients. Direct and alternating optimization procedures are presented to solve the problem. The proposed method is applied to simulated and benchmark data. In the comparison with existing methods, the resulting RBF networks have similar prediction accuracies with the smaller numbers of inputs and basis functions.

Key words: Regression, Function approximation, Feedforward neural network, Radial basis function network, RBF, Input selection, Basis function selection, Feature selection

1 Introduction

Nowadays, amount of collected data increases rapidly. This means that lots of input variables are available for the model construction in many cases. For

* Corresponding author. Tel.: +358 9 451 3543; fax: +358 9 451 3277. E-mail address: tikka@cis.hut.fi. Web page: http://www.cis.hut.fi/tikka

instance, in the analysis of microarray data the number of input variables may be several thousands. It is impossible to evaluate all the possible combinations of input variables in any reasonable time, even with the simplest models. Thus, effective and automatic input selection methods are required in many applications.

Relevance of an input variable depends on the problem to be solved. In the present work, the purpose is to construct as accurate regression models as possible. An input is considered to be useful or informative if its inclusion decreases the approximation error considerably. It is thereby useless or noninformative if its rejection has a minor effect on the approximation error. Several other definitions of relevance of the input variables are given in [9,24,18].

There are at least three advantages in input selection [18]. First, generalization capability or prediction performance of the model may be increased. Second, the cost-effective models are obtained. It is faster to train the models and economic savings are achieved if measuring of the variables is expensive. Third, understanding of the underlying process that has generated the data is improved since the irrelevant variables are dropped from the model. Input selection methods are typically divided into three classes, which are filter approaches [9], wrapper approaches [24], and embedded methods [18].

In the filter approach, the input selection procedure is independent from the final prediction model that is used to estimate the output values [9]. In the first phase, input selection is carried out using a simple model, for example, linear models [7,48] or linear-in-the-parameter models, e.g. polynomials [26]. A computationally more demanding choice is to evaluate mutual information between the inputs and the output [16]. In the second phase, a nonlinear model based on the selected inputs is trained. Filter approaches are usually computationally fast, since the final prediction model is constructed only once. On the other hand, the obtained subset of inputs may not be optimal for the nonlinear model.

In the wrapper approach, the prediction model of interest is used to rank the selected subsets of inputs [24]. The ranking is based on generalization error estimation techniques, e.g. the cross validation. The optimal but computationally infeasible wrapper method, or the input selection method in general, is the exhaustive search, in which prediction performances of all the possible input combinations are evaluated. The common strategies to search a feasible number of candidate subsets are forward selection and backward elimination [49,50]. Naturally, they can also be used in the input selection phase of the filter approaches, especially in the case of a large number of input variables. The wrapper approach is typically computationally more demanding than the filter approach, since the training of the prediction model is carried out several times.

In the embedded methods, the input selection procedure is incorporated into the training of the model [18]. The basic principle is to formalize an objective function consisting of two terms, which compete with each other. The first term measures the goodness-of-fit and it should be maximized. The second term penalizes for the number of inputs and it should be minimized. The lack of algorithms to directly minimize the number of variables for nonlinear predictors is stated by the authors of [18]. However, the second term can be replaced by a regularization term, which shrinks the values of parameters toward zero or set them exactly to zero. Algorithms, that minimize regularized or constrained cost functions, have been proposed for the multilayer perceptron [12] and the radial basis function (RBF) networks [47]. Computational complexities of embedded methods are typically between complexities of the filter and wrapper approaches.

In the present article, an embedded method for input variable and basis function selection for the RBF network is proposed. The novelty of the present work can be summarized as follows. The problem is formulated as a constrained optimization problem. The objective is to minimize the errors between the network outputs and the observed values with subject to sparsity constraints. The first constraint is imposed on the weights of input dimensions and the second constraint restricts the values of parameters of the output layer. Both constraints are implemented by bounding the sum of absolute values of the parameters. The problem is furthermore written as an unconstrained problem using a logarithmic barrier function. A Quasi-Newton type optimization algorithm is proposed to solve the unconstrained formulation.

The rest of the article is organized as follows. In Section 2, a regression problem is defined and the RBF networks are shortly introduced. After that, the closely related least squares support vector machine and reduced rank kernel ridge regression methods are reviewed. Section 3 contains relevant issues on the model selection. The constrained cost function for the simultaneous input and basis function selection and its optimization is proposed in Section 4. The results of experiments on two illustrative examples and three benchmark data sets are shown in Section 5. Finally, summary and conclusions are given in Section 6.

2 Regression problem

In a regression problem, the purpose is to learn an input-output relationship based on data. The underlying functional form is typically unknown. However, dependencies between the inputs and the output are usually nonlinear. An additional objective is to find the most informative combination of input variables.

The neural networks are an appropriate choice to model data, since they are capable to approximate a wide class of functions very well [33]. No restrictive assumptions about the actual underlying dependency structure have to be made. The quality of estimation depends on the complexity of the network, which can be controlled by varying the number of neurons and using different subsets of the input variables [1].

2.1 RBF networks

A radial basis function (RBF) network consists of input, hidden, and output nodes [32]. Each node in the single hidden layer corresponds to a basis function, whose activation is evaluated by the distance between an input vector and the center of the basis function. The output of the network is a linear combination of the basis functions.

Let us consider a set of N measurements from d inputs $\boldsymbol{x}_n = [x_{n1}, \ldots, x_{nd}]$ and an output $y_n,\ n = 1, \ldots, N$. The objective is to estimate the output y_n as accurately as possible using the inputs $\boldsymbol{x}_n$. The output of RBF network with the Gaussian basis functions is

$$\begin{aligned} &\hat{y}_n = f(\boldsymbol{x}_n) = \sum_{m=1}^{M} \alpha_m K(\boldsymbol{c}_m, \boldsymbol{x}_n) + \alpha_0 \ , \\ &\text{where} \ \ K(\boldsymbol{c}_m, \boldsymbol{x}_n) = \exp\left(-\frac{\|\boldsymbol{c}_m - \boldsymbol{x}_n\|^2}{\sigma_m^2} \right) \ , \end{aligned} \tag{1}$$

and M, $\boldsymbol{c}_m$, and σ_m are the number, the centers, and the widths of the basis functions, respectively. The model can be written in the matrix form as

$$\hat{\boldsymbol{y}} = \boldsymbol{K}\boldsymbol{\alpha}, \tag{2}$$

where the elements of matrix $\boldsymbol{K}$ are defined as $\boldsymbol{K}_{nm} = \{K(\boldsymbol{c}_m, \boldsymbol{x}_n)\}$, $m = 1, \ldots, M$, $n = 1, \ldots, N$ and the $(M+1)^{\text{th}}$ column is the vector of ones corresponding to the bias term.

Usually, the training of the RBF network includes three steps. First, the centers of the basis functions $\boldsymbol{c}_m$ are placed using some unsupervised clustering algorithm, such as k-means [28] or the Gaussian mixture models trained with the Expectation-Maximization algorithm [8]. Second, the widths of the basis functions are evaluated, for instance, by weighting the standard deviation of the data in each cluster [6] or determining the Euclidean distance from each center to its p^{th} nearest neighbor [28]. These heuristics vary the widths such that there are adequate overlap between the basis functions. Third, the output layer parameters $\boldsymbol{\alpha}$ are estimated by minimizing the mean squared error (MSE) between the network outputs $\hat{y}_n$ and the observations y_n.

The observed values y_n are not taken into account in the placement of centers in [28] and [8]. The approximation accuracy may be increased if the samples y_n are considered as it is done in [31]. The centers of the basis functions are selected from a large set of possible locations, i.e from all the samples in the training set, using the forward selection by minimizing the estimate of the generalization error. The widths of the basis functions have the same predefined value in [31].

2.2 LS-SVM and RRKRR

Support vector machine (SVM) is the widely used and well known kernel machine, which is also applicable in the regression problems [43]. SVM uses inequality constraints, which results in a quadratic programming problem. If the inequalities are replaced by the equality constraints the solution is obtained by solving a system of linear equations, and that formulation is known as kernel ridge regression (KRR) [40] or least squares support vector machine (LS-SVM) [44]. If particular choices are made in the construction of the RBF network, it is also closely related to KRR and LS-SVM.

The derivation of LS-SVM starts from the model

$$\hat{y} = \boldsymbol{z}^T \phi(\boldsymbol{x}) + b \ , \tag{3}$$

where $\phi(\cdot)$ is a mapping from the input space $\boldsymbol{x}$ to a high dimensional feature space. Given a data set $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, the optimization problem in the primal weight space is

$$\begin{array}{ll} \underset{\boldsymbol{z},b,\boldsymbol{e}}{\text{minimize}} & J_p = \dfrac{1}{2}\boldsymbol{z}^T\boldsymbol{z} + \dfrac{\gamma}{2}\displaystyle\sum_{n=1}^{N} e_n^2 \\ \text{such that} & y_n = \boldsymbol{z}^T\phi(\boldsymbol{x}_n) + b + e_n, \quad n = 1, \ldots, N \ , \end{array} \tag{4}$$

where γ is the regularization parameter and $\boldsymbol{e}^T = [e_1, \ldots, e_N]$. The cost function in Eq. (4) is the usual ridge regression [20] formulation in the feature space. However, the primal problem cannot be solved in the case of an infinite dimensional feature space. The optimum is therefore calculated via the Lagrange function. The resulting model is

$$\hat{y}(\boldsymbol{x}) = \sum_{n=1}^{N} \beta_n K(\boldsymbol{x}_n, \boldsymbol{x}) + b \ , \tag{5}$$

where the kernel trick $K(\boldsymbol{x}_k, \boldsymbol{x}_l) = K_{kl} = \phi(\boldsymbol{x}_k)^T\phi(\boldsymbol{x}_l)$ is applied and the parameters β_n, $n = 1, \ldots, N$ and b are obtained by solving a system of linear equations. The kernel function $K(\boldsymbol{x}_k, \boldsymbol{x}_l)$ must satisfy Mercer's condi-

tion [44]. The most typical choice is perhaps the Gaussian kernel $K(\boldsymbol{x}_n, \boldsymbol{x}) = \exp(-\|\boldsymbol{x}_n - \boldsymbol{x}\|^2/\sigma^2)$, when the feature space is infinite dimensional.

The LS-SVM model in Eq.(5) is generally fully dense, i.e $\beta_n \neq 0$, $n = 1, \ldots, N$. Sparse approximation strategies of LS-SVM, in which the pruning is based on the sorted absolute values of the parameters β_n, are presented in [45]. Reduced rank kernel ridge regression (RRKRR) is an alternative approach to built a sparse model [10]. The aim is to identify a subset of the training samples, $\{\boldsymbol{x}_j\}_{j\in\mathcal{S}} \subset \{\boldsymbol{x}_n\}_{n=1}^N$, such that the mappings of all the training samples are approximated by a linear combination of the mappings of the samples in $\mathcal{S}$.

The training of the RRKRR model consists of two parts. First, the mean reconstruction error

$$\mathrm{MRE}(\mathcal{S}) = \sum_{n=1}^{N} \left(1 - \frac{\boldsymbol{K}_{\mathcal{S}n}^T \boldsymbol{K}_{\mathcal{S}\mathcal{S}}^{-1} \boldsymbol{K}_{\mathcal{S}n}}{K_{nn}}\right) \ , \tag{6}$$

where $\boldsymbol{K}_{\mathcal{S}\mathcal{S}}$ is a square matrix and $\boldsymbol{K}_{\mathcal{S}n}$ is a column vector such that $\boldsymbol{K}_{\mathcal{S}\mathcal{S}} = \{K_{j_1 j_2}\}_{j_1, j_2 \in \mathcal{S}}$ and $\boldsymbol{K}_{\mathcal{S}n} = \{K_{jn}\}_{j\in\mathcal{S}}$, is used to identify a subset of feature vectors [2]. The selection is a forward iterative process. Starting from the empty set $\mathcal{S} = \emptyset$, a training sample that minimize MRE is found at each iteration.

Second, the following problem

$$\underset{\tilde{\boldsymbol{\beta}}, b}{\text{minimize}} \quad J_{\mathrm{RRKRR}} = \frac{1}{2} \sum_{j_1, j_2 \in \mathcal{S}} \tilde{\beta}_{j_1} \tilde{\beta}_{j_2} K_{j_1, j_2} + \frac{\gamma}{2} \sum_{n=1}^{N} (y_n - \sum_{j\in\mathcal{S}} \tilde{\beta}_j K_{jn} - b)^2 \tag{7}$$

is solved [10]. The parameters $\tilde{\beta}_j$, $j \in \mathcal{S}$ and b are found by solving a system of linear equations and the resulting model is

$$\hat{y}(\boldsymbol{x}) = \sum_{j\in\mathcal{S}} \tilde{\beta}_j K(\boldsymbol{x}_j, \boldsymbol{x}) + b \ . \tag{8}$$

A computationally fast strategy to estimate the generalization error of the RRKRR model in Eq. (8) is presented in [11].

3 Model complexity determination

It is easy to train a RBF network or a LS-SVM model, which interpolates exactly the training data. However, they would produce very poor generalization errors. In order to achieve accurate predictions for novel data, the effective complexity of the model has to be controlled. The generalization error is commonly estimated using resampling techniques [14].

The complexity of the model can be controlled by the number of basis functions and the regularization parameter in the cases of the RBF network and LS-SVM, respectively. Alternatively, as in [47], the RBF network can be trained by solving a regularized cost function

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \quad J_{\text{RBF}} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 + C\sum_{m=1}^{M}\alpha_m^2 \ . \tag{9}$$

The regularization by the squared values of the parameters is known as weight decay [8] or ridge regression [20]. If the basis function is placed on each training data point in the RBF network ($M = N$) and the widths of the basis functions have a common value $\sigma_m = \sigma$, the difference between the RBF network in Eq. (1) and the LS-SVM model with the Gaussian kernel in Eq. (5) is the evaluation of the parameters α_m and β_n. It is unlikely that the solution of the problem in Eq. (9) is sparse in terms of the basis functions, i.e. $\alpha_m \neq 0$, $m = 1, \ldots, M$. The values of the parameters are only shrunk toward zero and the effective number of parameters is decreased.

Several sequential learning algorithms to select the basis functions for the RBF network exist. Resource allocating (RA) networks add new hidden units based on the novelty of observations and optimize the output layer weights using the least mean squares algorithm [34] or the extended Kalman filter [23]. The drawback of the RA networks is that they cannot remove the already added basis functions from the network. The minimal RA network [52] and the general growing and pruning algorithm [21] allow also the pruning of units which are found to be useless in later stages of the sequential process. A totally opposite approach is an extreme learning machine (ELM) [22]. The input layer parameters, such as the centers and the widths of the basis functions, are randomly selected and only the output layer parameters are optimized. Although ELM produces competitive prediction accuracies with extremely fast training, sometimes a practical interpretation of the input-output mapping may be difficult to give due to the random input layer parameters. ELM might potentially need a large number of basis functions, whereas the interest of the present paper is in sparse networks. Furthermore, any of the previous approaches do not consider the input selection at all.

However, the rejection of uninformative input variables from the model may also improve the generalization capability [18]. It is furthermore mentioned in [8], that the number of basis functions that is required to produce a smooth mapping is significantly decreased by discarding the irrelevant inputs. Forward selection (FS) is a well known strategy to select input variables and it is used as a baseline method in the experiments, since it is robust against overfitting [36].

In the FS algorithm [27], the input variables are added one at a time into the initially empty set of inputs. If d is the number of available inputs, $(d+1)d/2$ models have to be trained. In addition, at each step of the FS procedure the

complexity of the model should be validated using the cross validation or another generalization error estimation technique, which increases the computational burden. Finally, the selected subset of inputs has obviously the smallest generalization error of all the evaluated combinations.

In the next section, a novel learning strategy for the RBF network is proposed. It produces the networks that are sparse in terms of both input variables and basis functions. The training is carried out by minimizing a constrained cost function, that is, the fitting of the model and selection of the most useful inputs and basis functions are done simultaneously.

4 Simultaneous input and basis function selection for RBF network

A disadvantage of neural networks, including the model in Eq. (1), is that networks contain all the available input variables. Moreover, it is practically almost impossible to distinguish informative inputs from noninformative ones. After the training of the model, partial derivatives of the model output with respect to the inputs can be applied to determine relevance of the inputs in the MLP and RBF networks [39,13,38]. A derivative criterion measures the local sensitivity of the network output $\hat{y}$ to the input x_i while the other inputs are kept fixed. Several criteria to measure importance of the inputs are reviewed in [25]. A relevance criterion gives typically a ranking of the inputs according to their importance and a proper subset is selected, for instance, using a statistical test [37]. A backward input selection algorithm for the RBF network, that is based on the partial derivatives and an efficient evaluation of the leave-one-out cross validation error, is presented in [49].

Alternatively, the learning of relative importances of the inputs can be incorporated into the training of the model. Several metric adaptation and relevance learning approaches are applied to various classification tasks in [51], whereas the function approximation problem is discussed in [35,50,47]. Results of experiments in [51,35,50,47] show that the usage of adaptive metrics improves prediction accuracy of the model and enables input variable selection. Following [35,50,47], the relevance learning is adopted by applying the weighted Euclidean distance

$$\begin{aligned} & d_{\boldsymbol{w}}(\boldsymbol{c}_m, \boldsymbol{x}_n) = \sqrt{\sum_{i=1}^{d} w_i (c_{m,i} - x_{n,i})^2} \ , \\ & w_i \geq 0 \ , \quad i = 1, \ldots, d \end{aligned} \tag{10}$$

in the Gaussian basis functions. The constraints $w_i \geq 0$ guarantee that the distance $d_{\boldsymbol{w}}(\boldsymbol{c}_m, \boldsymbol{x}_n)$ is real-valued and nonnegative. The adaptive weights w_i

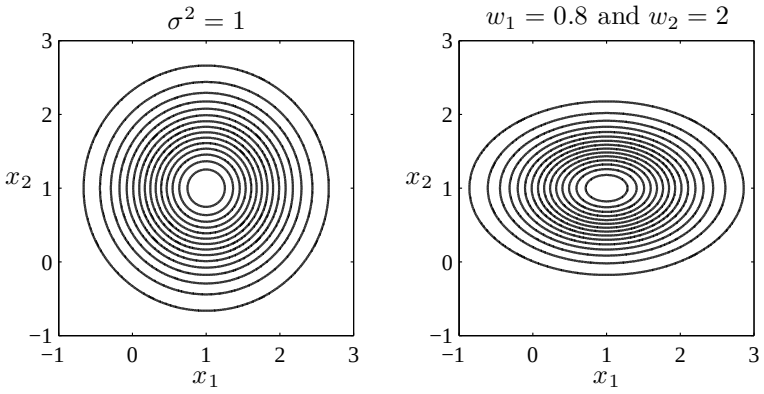

Fig. 1. The level curves of the Gaussian basis function with the Euclidean distance (*left*) and the weighted Euclidean distance (*right*).

reflect the relevance of the input variables, such that the most relevant input variable has the largest weight.

The output of the RBF network with the adaptively weighted distance in Eq. (10) is

$$\begin{aligned} \hat{y}_n(\boldsymbol{\alpha}, \boldsymbol{w}) &= \sum_{m=1}^{M} \alpha_m K_{\boldsymbol{w}}(\boldsymbol{c}_m, \boldsymbol{x}_n) + \alpha_0 \\ \text{where } K_{\boldsymbol{w}}(\boldsymbol{c}_m, \boldsymbol{x}_n) &= \exp\left(-d_{\boldsymbol{w}}(\boldsymbol{c}_m, \boldsymbol{x}_n)^2\right) \quad . \end{aligned} \tag{11}$$

The same weights w_i are used in all the basis functions, which are initially placed on each training data point. The basis functions are ellipsoidal, whose principal axes are parallel to the coordinate axes. In Fig. 1, the level curves of the Gaussian basis function with the Euclidean distance (*left*) and with the weighted Euclidean distance (*right*) are illustrated. It is observed, that the activation of the basis function gets more and more localized in the direction x_i with increasing value of the weight w_i. In the case of $w_i = 0$, the activations of basis functions are constant with respect to the corresponding input x_i, which makes the input x_i useless in the network.

The goal is to estimate the weights w_i by minimizing the MSE between the observations y_n and the outputs of the network $\hat{y}_n(\boldsymbol{\alpha}, \boldsymbol{w})$. Let us consider the assumption, that the basis function is placed on each training data point, i.e $M = N$. If $w_i > 0$, $i = 1, \ldots, d$ and the values of the parameters α_m, $m = 1, \ldots, M$ are not regularized, the network interpolates exactly the observations y_n provided that the resulting design matrix is invertible. Particularly, the design matrix approaches the identity matrix with increasing values of the weights w_i. On the other hand, if $w_i = 0$, $i = 1, \ldots, d$ the output of the network is constant, which equals to the mean of the observations y_n.

In order to achieve a smooth practical mapping and sparsity in terms of the inputs and the basis functions, there has to be constraints on the values of weights $\boldsymbol{w}$ and the output layer parameters $\boldsymbol{\alpha}$. This leads to consider the

following optimization problem

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\alpha}}{\text{minimize}} \quad & E(\boldsymbol{\alpha},\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \hat{y}_n(\boldsymbol{\alpha},\boldsymbol{w})\right)^2 \\ \text{such that} \quad & \sum_{m=0}^{M}|\alpha_m| \le r \ , \\ & \sum_{i=1}^{d} w_i \le t, \quad \text{and} \quad w_i \ge 0, \quad i = 1,\ldots,d \ . \end{aligned} \tag{12}$$

It is known, that the absolute values in the constraints are effective in the model selection [46]. With an appropriate choice of the shrinkage parameters r and t, some of the coefficients α_m and w_i will be shrunk toward zero and some of them will be set exactly to zero. The basis functions and input variables having zero coefficients do not contribute to the model output at all. The number of zero coefficients can be increased by decreasing the values of shrinkage parameters, whereas large values produce more flexible models. Generalization capability of the model depends strongly on the shrinkage parameters, that can be selected by training the model with fixed predefined monotonically increasing values and estimating the generalization error in each case. The parameters that produce the smallest estimate for the generalization error are selected and utilized in the training of the final model. For example, a grid of shrinkage parameters is evaluated using the 10-fold cross validation in the experiments of the present work.

Following [41], the first constraint $\sum|\alpha_m| \le r$ can be written equivalently as

$$\sum_{m=0}^{M} s_m \le r \quad \text{and} \quad |\alpha_m| \le s_m, \ m = 0,\ldots,M \ , \tag{13}$$

where s_m are slack variables. Nondifferentiability of absolute values makes the optimization difficult. Fortunately, the constraints including absolute values can be further written as $|\alpha_m|^2 \le s_m^2$, which eases the problem computationally. However, the above transformation increases the number of parameters to be optimized by $M+1$ parameters.

4.1 Algorithm 1 (Direct optimization)

The constrained problem in Eq. (12) can be approximated using a barrier penalty function method [3]. The idea is to construct the barrier term from the constraints and add it to the objective function. Using the logarithmic barrier function and the reformulation of the first constraint in Eq. (13), the

Algorithm 1 Direct optimization of the cost function in Eq. (14)

1: Initialization: $k = 1$, choose a strictly feasible point $\boldsymbol{\Theta}_k^T = [\boldsymbol{\alpha}_k^T \ \boldsymbol{s}_k^T \ \boldsymbol{w}_k^T]$, set $\mu_k = \mu_{\max}$, and select $\mu_{\min}$ and a factor $c \in (0, 1)$

2: Use $\boldsymbol{\Theta}_k$ as the starting point and apply Algorithm 3 (App. A) to

$$\underset{\boldsymbol{\Theta}_k}{\text{minimize}} \quad J(\boldsymbol{\alpha}_k, \boldsymbol{s}_k, \boldsymbol{w}_k) = E(\boldsymbol{\alpha}_k, \boldsymbol{w}_k) + \mu_k \left(B_1(\boldsymbol{\alpha}_k, \boldsymbol{s}_k) + B_2(\boldsymbol{w}_k)\right),$$

such that the logarithms in Eq. (14) are defined.

3: Let $\boldsymbol{\Theta}_{k+1}$ be an optimal solution in the previous step, set $\mu_{k+1} = \max\{c\mu_k, \mu_{\min}\}$, and $k = k + 1$

4: Go to step 2 or terminate if the stopping criteria are fulfilled

problem in Eq. (12) can be written as

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{s},\boldsymbol{w}}{\text{minimize}} \quad & J(\boldsymbol{\alpha}, \boldsymbol{s}, \boldsymbol{w}) = E(\boldsymbol{\alpha}, \boldsymbol{w}) + \mu \left(B_1(\boldsymbol{\alpha}, \boldsymbol{s}) + B_2(\boldsymbol{w})\right) \ , \\ & B_1 = -\log\left(r - \sum_{m=0}^{M} s_m\right) - \frac{1}{M}\sum_{m=0}^{M} \log(s_m^2 - \alpha_m^2) \ , \\ & B_2 = -\log\left(t - \sum_{i=1}^{d} w_i\right) - \frac{1}{d}\sum_{i=1}^{d} \log(w_i) \ , \end{aligned} \tag{14}$$

where the optimization parameter $\mu > 0$ controls the accuracy of the approximation. The objective function $J(\boldsymbol{\alpha}, \boldsymbol{s}, \boldsymbol{w})$ is unconstrained and differentiable with respect to all its arguments. Several methods to solve problems with the absolute value constraints are compared in [42]. Methods using a barrier function are found to be competitive in terms of convergence and the number of iterations.

In practice, the solution of the problem in Eq. (14) is evaluated using a sequence of decreasing values of the parameter μ. The gradual decrement of μ eases the convergence [3]. An algorithm to solve the problem in Eq. (14) for the fixed values of the shrinkage parameters r and t is presented in Algorithm 1.

First, the values of parameters $\boldsymbol{\Theta}_k^T = [\alpha_0, \ldots, \alpha_M, s_0, \ldots, s_M, w_1, \ldots, w_d]$ are initialized such that the constraints in Eqs. (12) and (13) are satisfied with the inequalities, i.e. a strictly feasible starting point is used. The second step is to find the parameters $\boldsymbol{\Theta}_{k+1}$, that minimize the objective function in Eq. (14). The problem cannot be solved in the closed form, thus the solutions are evaluated using Algorithm 3, that is presented in detail in Appendix A. It is a Quasi-Newton (QN) type optimization algorithm [3], that is particularly designed to solve the proposed problem in Eq.(14). In the construction of the Hessian matrix, approximations are used for the term $E(\boldsymbol{\alpha}, \boldsymbol{w})$, whereas the exact second partial derivatives are calculated for $B_1(\boldsymbol{\alpha}, \boldsymbol{s})$ and $B_2(\boldsymbol{w})$. In the course of optimization, it is important to check explicitly that the logarithms in Eq. (14) are defined. Otherwise, the problem is unconstrained. The value of optimization parameter μ_k is decreased in the third step by a factor $c \in (0, 1)$

Algorithm 2 Alternating optimization of the cost function in Eq. (14)

1: Set $k = 1$, $\mu_k = \mu_{\max}$, initialize the weights $\boldsymbol{w}$, evaluate the basis functions $K_{\boldsymbol{w}_k}(\boldsymbol{c}_m, \boldsymbol{x}_n)$, $m = 1, \ldots, M$ and $n = 1, \ldots, N$, and select $\mu_{\min}$ and a factor $c \in (0, 1)$

2: Keep the weights $\boldsymbol{w}_k$ fixed and solve the problem

$$\underset{\boldsymbol{\alpha}, \boldsymbol{s}}{\text{minimize}} \quad J_1(\boldsymbol{\alpha}, \boldsymbol{s}) = E(\boldsymbol{\alpha}, \boldsymbol{w}_k) + \mu_k B_1(\boldsymbol{\alpha}, \boldsymbol{s})$$

using Algorithm 3 (App. A). Let $\boldsymbol{\alpha}_{k+1}$ and $\boldsymbol{s}_{k+1}$ be optimal solutions.

3: Keep the obtained values $\boldsymbol{\alpha}_{k+1}$ fixed and solve the problem

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad J_2(\boldsymbol{w}) = E(\boldsymbol{\alpha}_{k+1}, \boldsymbol{w}) + \mu_k B_2(\boldsymbol{w})$$

using Algorithm 3. Let $\boldsymbol{w}_{k+1}$ be an optimal solution.

4: Update the basis functions $K_{\boldsymbol{w}_{k+1}}(\boldsymbol{c}_m, \boldsymbol{x}_n)$, evaluate the value of cost function in Eq. (14), set $\mu_{k+1} = \max\{c\mu_k, \mu_{\min}\}$ and $k = k + 1$

5: Go to step 2 or terminate if the stopping criteria are fulfilled

until it reaches the minimum value $\mu_{\min}$. Steps 2 and 3 are repeated until the stopping criteria are fulfilled, that is, $\mu_k = \mu_{\min}$ and the sequence of values of the cost function is converged or the maximum number of iterations is performed.

4.2 Algorithm 2 (Alternating optimization)

Instead of the direct optimization method, alternating optimization (AO) can be used to solve the problem in Eq. (12). AO is an iterative procedure for optimizing an objective function by alternating the restricted optimizations over nonoverlapping subsets of variables [5]. In the present work, the obvious subsets of variables are $\boldsymbol{\alpha}$ and $\boldsymbol{w}$. The motivation to use AO is two-fold. First, the restricted minimizations of the problem in Eq. (12) with respect to the variables $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ in turn are computationally easier than the direct optimization of the problem. Second, the restricted optimization with respect to $\boldsymbol{\alpha}$ keeping $\boldsymbol{w}$ fixed is the convex optimization problem. Thus, the global minimizer is found for this subproblem. The drawback of AO is, that it may converge to a solution, which is a local optimizer or a saddle point for the original problem [5].

In Algorithm 2, the AO strategy is proposed to solve the problem in Eq. (14). Again, the values of shrinkage parameters r and t are assumed to be given and the value of optimization parameter μ is gradually decreased. First, strictly feasible initial values for the weights $\boldsymbol{w}_k$ are selected and the activations of basis functions $K_{\boldsymbol{w}_k}$ are evaluated.

The second step is to optimize the parameters $\boldsymbol{\alpha}$ using the fixed values of

$\boldsymbol{w}_k$. It is essentially an estimation of a linear model by minimizing the sum of squares subject to the absolute value constraints, which is known as the lasso problem [46]. Efficient algorithms to solve the lasso problem are presented in [30,15]. The computational complexity is reduced by restricting the optimization on an active set of variables. The algorithms in [30,15] calculate solutions for a range of values of shrinkage parameter. However, the QN algorithm (Algorithm 3) that is also applied in Algorithm 1 is used for the optimization in the present work, so that the results of Algorithms 1 and 2 are comparable with each other. Now, no approximations are needed in the construction of the Hessian matrix, since the weights $\boldsymbol{w}$ are kept fixed and, thus the second partial derivatives presented in Eq. (A.2) are not needed, see Appendix A.

In step 3, the obtained values $\boldsymbol{\alpha}_{k+1}$ are kept fixed and the optimization is carried out with respect only to the weights $\boldsymbol{w}$ using Algorithm 3. The approximations are once again used in the evaluation of the second partial derivatives $\nabla^2 J_2(\boldsymbol{w})$. The similar optimization is used in [47].

In step 4, the activations of basis functions $K_{\boldsymbol{w}_{k+1}}$ are updated and the value of objective function in Eq. (14) is evaluated using the new values of parameters $\boldsymbol{\alpha}_{k+1}$, $\boldsymbol{w}_{k+1}$. Also, the value of optimization parameter μ_k is decreased. Steps 2-4 are repeated until the stopping criteria are achieved. The same criteria as in Algorithm 1 are used.

The convergence properties of the AO technique are well known [4,19]. The sequence of parameter values converge to a local or global solution if the subproblems are solved exactly. In [19], it is shown that an inexact solution, for example a single iteration of Newton method, with respect to one subset of variables is enough to retain the convergence properties. This could be utilized in Algorithm 2 if the computational cost is too high due to the large number of basis functions or input variables.

5 Experiments

This section presents experiments on simulated and benchmark data sets. Simulated data sets are used to illustrate characteristics of the proposed method. In the case of simulated data, the assessment of quality of results is straightforward, since the underlying phenomenon is completely known.

Furthermore, Algorithms 1 and 2 are compared with two LS-SVM and two RRKRR models. The first LS-SVM is trained using all the available input variables [44]. The second LS-SVM is trained with the FS algorithm [27], and it is referred by LS-SVM with FS. The RRKRR models [10] are constructed

using all the inputs (RRKRR) and the same subset of inputs (sparse RRKRR) that is found by the FS algorithm in the training of the second LS-SVM model. In both RRKRR models, the selection of basis functions is terminated, when $\boldsymbol{K}_{\mathcal{SS}}$ in Eq. (6) is no longer invertible. All the models are built for three publicly available benchmark data sets.

In all the experiments, the values of parameters α_m, s_m, $m = 0, \ldots, M$ and w_i, $i = 1, \ldots, d$ are initialized as follows $w_i = 0.9t/d$ and $s_m = 0.9r/(M+1)$. The output layer parameters are set such that $|\alpha_m| = 0.99 s_m$ and the sign (positive or negative) is selected randomly with equal probabilities. The basis function is initially placed on each training data point, that is $M = N$. Equal initial values for the weights w_i are also used in [50]. The initialization of $\boldsymbol{w}$ corresponds to the ordinary RBF network in Eq. (1) with the width $\sigma_m^2 = d/0.9t$. The previous initializations guarantee that the logarithms in Eq. (14) are defined. All the input and output variables are scaled to have zero mean and unit variance before the training of the models. The reported errors are mean squared errors for the normalized data if it is not mentioned otherwise.

In Algorithms 1 and 2, the optimization parameter is decreased from the initial value $\mu_1 = \mu_{\max} = 10^{-3}$ by the factor $c = 0.85$ after each iteration. Algorithms 1 and 2 are terminated when $\mu_k = \mu_{\min} = 10^{-6}$ and the relative decrement in the value of cost function is less than 10^{-4} during the last five iterations or the maximum number of iterations $k_{\max} = 400$ is reached.

5.1 Illustrative examples on simulated data

The first task is to estimate a one-dimensional target function using five input variables. Each input variable $\boldsymbol{x} = [x_1, \ldots, x_5]$ is independently uniformly distributed in the range $x_i \in [-2, 2]$ and the target function is

$$t = (1 - x_1 + 2x_1^2) \exp(-x_1^4) \ . \tag{15}$$

The noisy samples $y = t + \varepsilon$ are obtained by adding normally distributed noise $\varepsilon \sim \mathrm{N}(0, 0.15^2)$ to the values of target function. Due to the construction, the output y depends only on the first input x_1 and the dependency is clearly nonlinear.

Algorithm 1 is evaluated in twenty points of the parameters r and t, which are logarithmically equally spaced in the ranges $r \in [2, 20]$ and $t \in [2, 15]$. In all, four hundred RBF networks are trained using $N_{tr} = 100$ samples and evaluated by separate validation data ($N_v = 1000$). The minimum validation error is achieved by the parameter values $r = 5.95$ and $t = 4.67$. In the original scale, the standard deviation of the validation errors $e_n = y_n - \hat{y}_n(\boldsymbol{\alpha}, \boldsymbol{w})$ is 0.17, which corresponds very well to the standard deviation of the noise.

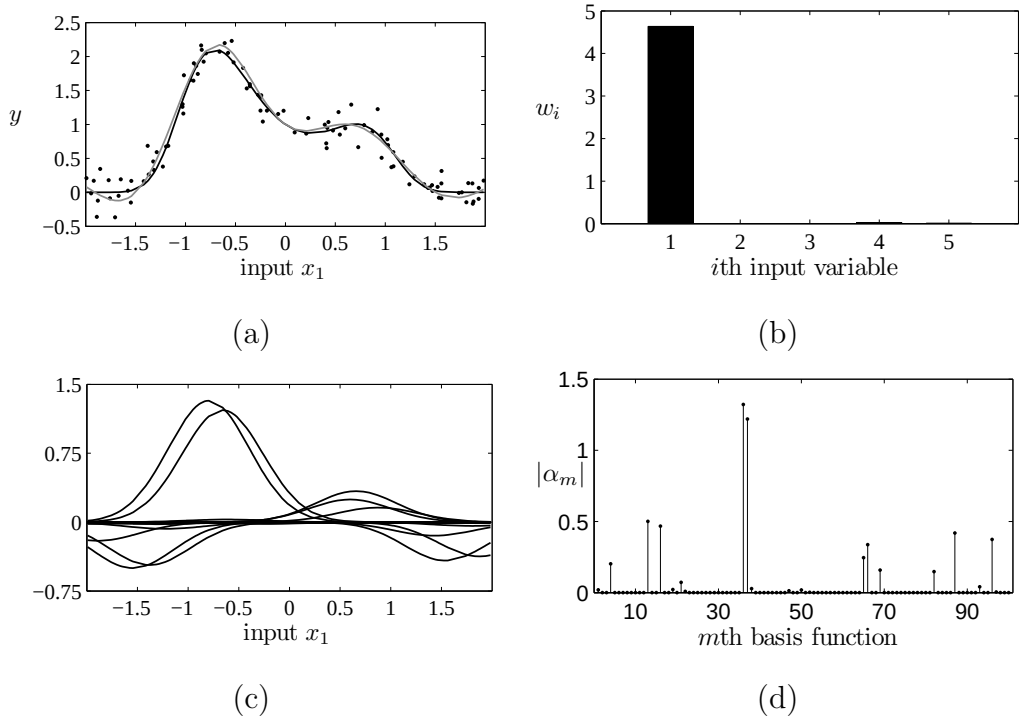

Fig. 2. (a) The first dimension of the training samples (black dots), the target function (black line), and the estimated output (gray line). (b) The weights w_i, $i = 1, \ldots, 5$. (c) The weighted basis functions $\alpha_m K(\boldsymbol{c}_m, \boldsymbol{x})$, $m = 1, \ldots, M$. (d) The absolute values of the output layer parameters $|\alpha_m|$.

The model producing the minimum validation error out of the 400 trials is summarized in Fig. 2. The maximum values of the target function are slightly overestimated and there are also slight underestimation in the beginning and in the end of the range of x_1. Otherwise, the estimated function approximates the target function nearly perfectly. Only the weight w_1 is clearly nonzero, which means that Algorithm 1 detected the correct input and discarded useless ones from the model. The activations of the most basis functions are zero through the whole range of the input variable x_1. Twenty one of the parameters α_m are nonzero, that is, solely the fifth of the available basis functions were selected to the final network.

The second simulated example is similar as in [17]. The objective is to estimate the following target function

$$t = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5. \tag{16}$$

The available input variables x_i, $i = 1 \ldots, 10$ are sampled independently from a uniform distribution $\mathrm{U}(0, 1)$. The target does not depend on the last five input variables, which are thereby useless. The noisy output observations y are generated by adding the Gaussian noise with zero mean and unit variance to the target values.

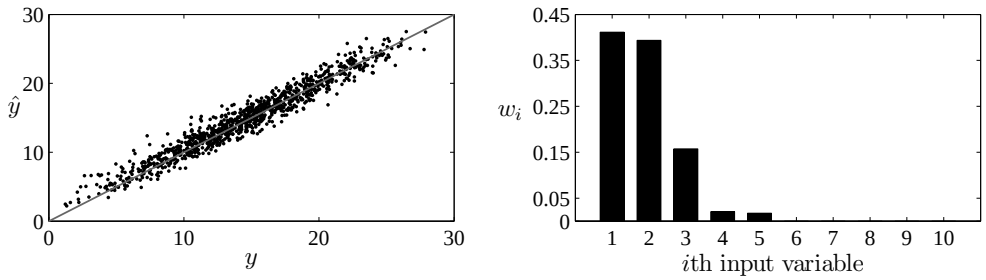

Fig. 3. The scatter plot of the estimated values $\hat{y}$ versus the output y for the validation data (*left*). The scaled weights w_i, $i = 1, \ldots, 10$ (*right*).

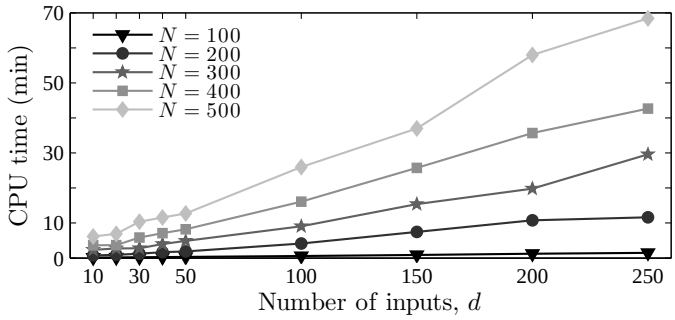

Fig. 4. The experimental running times of Algorithm 1 in minutes for several choices of the numbers of inputs d and samples N.

Algorithm 1 is again evaluated using twenty logarithmically equally spaced points of the shrinkage parameters $r \in [10, 10^3]$ and $t \in [0.05, 5]$. The sizes of the training and validation sets are $N_{tr} = 250$ and $N_v = 1000$, respectively. Out of the 400 possibilities, the combination $r = 380$ and $t = 0.13$ results in the minimum validation error. The standard deviation of the errors e_n is 1.15 in the original scale, which nearly coincides with the known standard deviation of the noise. The minimum validation error model is illustrated in Fig. 3. It shows that the model estimates the output very well and Algorithm 1 selects again the correct input variables. In the right panel of Fig. 3, the weights are scaled such that $\sum_i w_i = 1$ to make the comparison of the relative importances of the inputs easier. The most relevant inputs are x_1 and x_2. The model uses approximately 67% of the available basis functions.

For both simulated data sets, virtually identical results are obtained by Algorithm 2.

5.1.1 Computational time

Theoretical complexities of the proposed algorithms are difficult to derive. However, experimental running times of Algorithm 1 are reported to illustrate the computational complexity. Several data sets are generated by varying the number of input variables d and the number of samples N. The inputs are

Table 1
Properties of the data sets, and the ranges of the hyperparameters (σ^2 and γ) and the shrinkage parameters (r and t).

	Bank	Boston housing	Wine
Training (N_t)	250	400	94
Test (N_{test})	7942	106	30
Inputs (d)	32	13	256
σ^2	$[10, 10^5]$	$[1, 10^4]$	$[10^3, 10^9]$
γ	$[1, 10^4]$	$[0.1, 10^4]$	$[100, 10^8]$
r	$[1, 100]$	$[20, 400]$	$[20, 300]$
t	$[0.01, 3]$	$[0.05, 5]$	$[10^{-3}, 0.05]$

always drawn from the uniform distribution $\mathcal{U}(0, 1)$ and a noisy output is generated as in the experiment on the second simulated data, see Eq. (16). Thus, the number of useful inputs is always the same and the number of useless ones varies.

The average computational times are shown in Fig. 4. The reported values are averages calculated from five replications. Each time the optimization is carried out using the shrinkage parameter values $r = 380$ and $t = 0.13$. The algorithms are implemented in MATLAB and all the experiments are carried out using a 2.2 GHz AMD Opteron processor. It can be seen that the computational time is more sensitive to the number of samples than to the number of input variables. However, all the cases including at most 50 inputs are fairly fast to evaluate. Algorithm 2 had consistently slightly faster running times.

5.2 Comparisons on benchmark data sets

The first data set is called Bank data [1]. It is synthetically generated and it simulates queues in a series of banks. It is also known that the dependencies are nonlinear and the amount of noise in the values of output is moderate. The second set is Boston Housing data [1], which is the extensively used benchmark data set. The task is to predict the median value of house prices in the area of Boston. The third set is Wine data [2]. In this case, the purpose is to estimate the level of alcohol of a wine from its observed mean infrared spectrum. The data sets are randomly divided into training and test sets. The sample sizes and the numbers of input variables are shown in Table 1.

[1] Available from: http://www.cs.toronto.edu/~delve/data/datasets.html
[2] Available from: http://www.dice.ucl.ac.be/mlg/index.php?page=DataBases

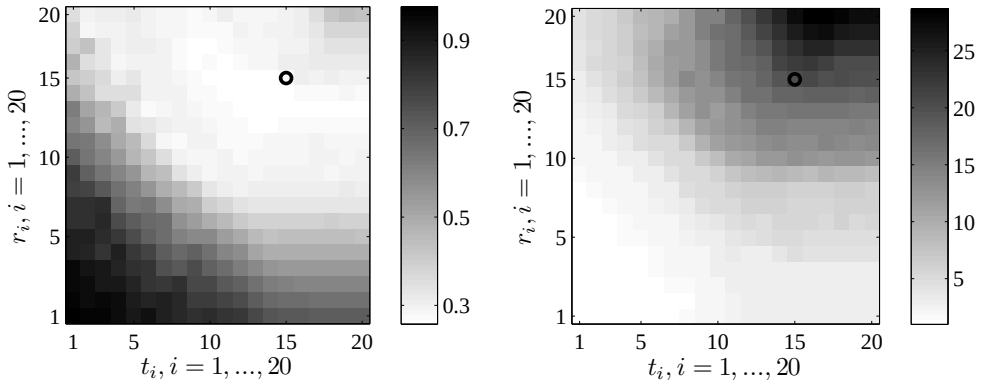

Fig. 5. The average validation errors (*left*) and the average numbers of selected inputs (*right*) in the CV repetitions for the different values of parameters r and t in Algorithm 1 in the case of Bank data. The black circle refers to the minimum validation error model.

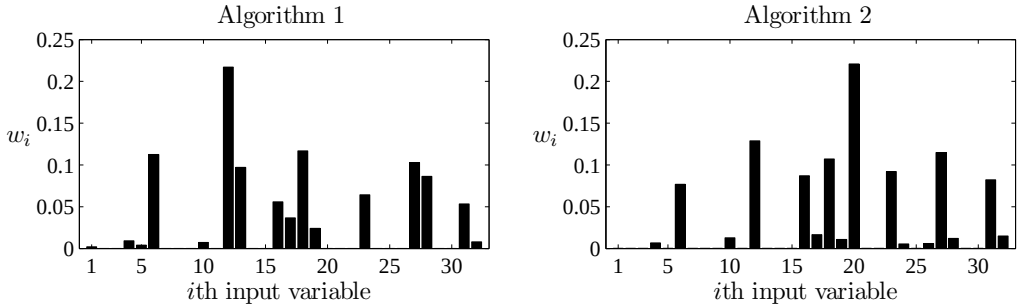

Fig. 6. The scaled values of weights w_i for the Bank data evaluated by Algorithm 1 (*left*) and Algorithm 2 (*right*).

The shrinkage parameters r and t of Algorithm 1 and 2 and the hyperparameters γ and σ^2 of the LS-SVM and RRKRR models are selected using the 10-fold cross validation (CV). The models are evaluated using twenty values of each parameter, which are logarithmically equally spaced in the ranges that are presented in Table 1. For instance, Algorithm 1 is evaluated using 400 different combinations of the parameters r and t. The test errors are reported for the models that produced the minimum validation errors.

The validation errors of the Bank data for different values of r and t in Algorithm 1 are illustrated in the left panel of Fig. 5. In the right panel of the same figure, the average numbers of selected inputs in CV repetitions are shown. It can be seen, that the network is notably affected by both shrinkage parameters. With small values of r and t (lower left corner), the network is overly restricted, since almost all the inputs are rejected from the model. The number of selected inputs increases when the values of r and t increase. Eventually, nearly all the inputs have nonzero weight w_i, which causes an upward trend in the validation error (upper right corner).

Table 2
The minimum validation errors (the first value) and the corresponding test errors (the second value) for the benchmark data sets.

	Bank	Boston housing	Wine
LS-SVM	0.31 | 0.27	0.12 | 0.14	0.0023 | 0.0039
LS-SVM with FS	0.21 | 0.30	0.10 | 0.12	0.0006 | 0.0055
RRKRR	0.31 | 0.28	0.12 | 0.14	0.0024 | 0.0030
sparse RRKRR	0.21 | 0.30	0.10 | 0.12	0.0019 | 0.0020
Algorithm 1	0.26 | 0.29	0.13 | 0.15	0.0026 | 0.0047
Algorithm 2	0.25 | 0.28	0.12 | 0.18	0.0041 | 0.0039

Table 3
The numbers of inputs (the first value) and basis functions (the second value) in the networks evaluated using the parameters, which produce the minimum validation errors.

	Bank	Boston housing	Wine
LS-SVM	32 | 250	13 | 400	256 | 94
LS-SVM with FS	11 | 250	11 | 400	74 | 94
RRKRR	32 | 149	13 | 343	256 | 41
sparse RRKRR	11 | 222	11 | 319	74 | 63
Algorithm 1	16 | 40	12 | 152	13 | 94
Algorithm 2	16 | 27	13 | 227	19 | 94

In Fig. 6, the values of weights $\boldsymbol{w}$ for the Bank data are presented. The weights are scaled such that $\sum_i w_i = 1$ to make the comparison of the relative importances of the inputs between the algorithms easier. Both algorithms selected 16 input variables. According to Algorithm 2, the most important input is x_{20}, i.e the weight w_{20} is the largest. Algorithm 1 dropped that input from the final network. Otherwise, both algorithms rank nearly the same input variables to be the most important in the prediction. In all, the algorithms selected 13 same input variables.

The minimum validation errors and the corresponding test errors are presented in Table 2. In terms of the test MSEs, there are not considerable differences between the methods in the cases of Bank and Wine data. LS-SVM with FS and sparse RRKRR perform better than Algorithm 2 (33% smaller MSE in the test set) in the Boston Housing data. The relative differences seem to be large also in the Wine data, however, all the methods are able to predict the output practically perfectly.

In Table 3, the numbers of selected inputs and basis functions are reported. The number of support vectors or basis functions equals always to the number of training samples in the LS-SVM models. In the cases of Bank and Boston Housing data, Algorithms 1 and 2 decrease remarkably the initial number of basis functions. They use also evidently smaller numbers of basis functions than the RRKRR models. In the case of Wine data, all the parameters α_m are nonzero in the models trained by Algorithms 1 and 2. However, the absolute values of three parameters are clearly larger than the rest are. Perhaps, the basis functions with the small absolute values would be discarded if there were more noise in the values of output. The RRKRR models manage to select basis functions also from the Wine data, since the selection is solely based on the input data.

The FS algorithm selects the smaller number of inputs than Algorithms 1 and 2 do in the case of Bank data. Nevertheless, most of the inputs selected by the FS algorithm are the same which have the highest weights in Algorithms 1 and 2, see Fig. 6. In the prediction of Wine data, Algorithms 1 and 2 use less than 10% of the available input variables. The model obtained by the FS algorithm includes definitely more inputs.

6 Summary and conclusions

A novel method for the simultaneous input variable and basis function selection for the RBF network is proposed. The problem is formulated as a constrained optimization problem, in which the objective is to minimize the mean squared error between the network outputs and the observed values. The separate constraints are imposed on the sum of the weights of input dimensions and on the sum of the absolute values of output layer parameters. Two continuous valued shrinkage parameters control the numbers of inputs and basis functions. Thus, no sequential backward elimination or forward selection strategies through the inputs and basis functions are needed.

The networks constructed by the proposed method have two major advantages. First, sparsity in terms of the input variables and the basis functions makes the networks less prone to overfitting. Second, the rejection of useless inputs from the network highlights the dependencies in the data. In addition, the weights of input dimensions describe relative importances of the inputs, that is, the most relevant inputs have the largest weights.

Another constraint in the original problem formulation is nondifferentiable due to the absolute values. The problem is reformulated using the logarithmic barrier function. Two optional strategies to solve the barrier reformulation are presented, namely the direct and alternating optimization. There are no

remarkable differences between the two solutions. The computational burden could be potentially diminished by restricting the calculations on active sets of inputs and basis functions. In the alternating optimization approach, this would be fairly straightforward for the optimization of the output layer parameters, since wide variety of the active set algorithms for the lasso problem already exist.

In the experiments on simulated data, the correct input variable selection is accomplished in conjuction with the accurate prediction of the target function. The decent results are also achieved in the experiments on the benchmark data sets. Algorithms 1 and 2 are competitive with the LS-SVM and RRKRR models in terms of generalization capability and with the FS algorithm in terms of the number of selected input variables.

Acknowledgements

The author would like to thank the reviewers and the associate editor of the article for insightful and valuable comments that helped to improve the manuscript significantly.

A Algorithm 3 (The Quasi-Newton optimization method)

A Quasi-Newton type optimization method to solve the problem in Eq. (14) for the fixed values of r, t, μ, and the given initialization $\boldsymbol{\Theta}_l^T = [\boldsymbol{\alpha}_l^T \boldsymbol{s}_l^T \boldsymbol{w}_l^T] = [\alpha_0^l, \ldots, \alpha_M^l, s_0^l, \ldots, s_M^l, w_1^l, \ldots, w_d^l]$ is summarized in Algorithm 3. The derivation of the algorithm is inspired by the Levenberg-Marquardt (LM) optimization algorithm [8,29].

Algorithm 3 starts by initializing the trust region parameter $\lambda_l = 1$, which is followed by the evaluation of the gradient and the Hessian matrix using the current values of the parameters $\boldsymbol{\Theta}_l$ (steps 1 and 2). The components of gradient $\nabla J(\boldsymbol{\Theta})$ of the objective function in Eq. (14) are

$$\begin{aligned} \nabla_{\alpha_m} J(\boldsymbol{\Theta}) &= -\frac{2}{N}\sum_{n=1}^{N} e_n \frac{\partial \hat{y}_n(\boldsymbol{\alpha}, \boldsymbol{w})}{\partial \alpha_m} + \mu \frac{\partial B_1(\boldsymbol{\alpha}, \boldsymbol{s})}{\partial \alpha_m} \ , \\ \nabla_{s_m} J(\boldsymbol{\Theta}) &= \mu \frac{\partial B_1(\boldsymbol{\alpha}, \boldsymbol{s})}{\partial s_m} \ , \\ \nabla_{w_i} J(\boldsymbol{\Theta}) &= -\frac{2}{N}\sum_{n=1}^{N} e_n \frac{\partial \hat{y}_n(\boldsymbol{\alpha}, \boldsymbol{w})}{\partial w_i} + \mu \frac{\partial B_2(\boldsymbol{w})}{\partial w_i} \ , \end{aligned} \tag{A.1}$$

where $e_n = y_n - \hat{y}_n(\boldsymbol{\alpha}, \boldsymbol{w})$. The following second partial derivatives with respect

Algorithm 3

1: Set $l = 1$, $\lambda^l = 1$, and use the initialization $\boldsymbol{\Theta}_l^T = [\boldsymbol{\alpha}_l^T \ \boldsymbol{s}_l^T \ \boldsymbol{w}_l^T]$

2: Evaluate the exact gradient $\nabla J(\boldsymbol{\Theta}_l)$ and the approximation of Hessian matrix $\tilde{\boldsymbol{H}}(\boldsymbol{\Theta}_l)$

3: Compute the search direction $\boldsymbol{p}_l$ by solving
$\left[\tilde{\boldsymbol{H}}(\boldsymbol{\Theta}_l) + \lambda_l \boldsymbol{I}\right] \boldsymbol{p}_l = -\nabla J(\boldsymbol{\Theta}_l)$

4: Determine the step length

$$\delta = \max\{\ 0 \leq \delta \leq 1 :$$
$$\textstyle\sum_{i=1}^{d} w_i^l + \delta p_{w_i}^l < t, \quad w_i^l + \delta p_{w_i}^l > 0, \quad i = 1, \ldots, d$$
$$\textstyle\sum_{m=0}^{M} s_m^l + \delta p_{s_m}^l < r,$$
$$|\alpha_m + \delta p_{\alpha_m}^l| < s_m^l + \delta p_{s_m}^l, \ m = 0, \ldots, M\}$$

5: Calculate the ratio
$r^l = \dfrac{J(\boldsymbol{\Theta}_l) - J(\boldsymbol{\Theta}_l + \delta \boldsymbol{p}_l)}{J(\boldsymbol{\Theta}_l) - \tilde{J}(\boldsymbol{\Theta}_l + \delta \boldsymbol{p}_l)}$

6: If $r^l > 0.75$, set $\lambda^l = \lambda^l / 2$

7: If $r^l < 0.25$, set $\lambda^l = 2\lambda^l$

8: Set $\lambda^{l+1} = \lambda^l$, and if $J(\boldsymbol{\Theta}_l + \delta \boldsymbol{p}_l) < J(\boldsymbol{\Theta}_l)$ set $\boldsymbol{\Theta}_{l+1} = \boldsymbol{\Theta}_l + \delta \boldsymbol{p}_l$

9: Set $l = l + 1$, go to step 2 if the stopping criterion is not fulfilled

to the MSE part

$$\begin{aligned} \nabla^2_{\alpha_m, w_i} E(\boldsymbol{\alpha}, \boldsymbol{w}) &= \frac{2}{N} \sum_{n=1}^{N} \frac{\partial \hat{y}_n}{\partial \alpha_m} \frac{\partial \hat{y}_n}{\partial w_i} + e_n \frac{\partial^2 \hat{y}_n}{\partial \alpha_m \partial w_i} \ , \\ \nabla^2_{w_i, w_j} E(\boldsymbol{\alpha}, \boldsymbol{w}) &= \frac{2}{N} \sum_{n=1}^{N} \frac{\partial \hat{y}_n}{\partial w_i} \frac{\partial \hat{y}_n}{\partial w_j} + e_n \frac{\partial^2 \hat{y}_n}{\partial w_i \partial w_j} \ , \end{aligned} \tag{A.2}$$

are approximated by neglecting the second terms as is done in [8,29]. The approximations are used to decrease the computational complexity of the algorithm. Observe, that $\nabla^2_{\alpha_{m_1}, \alpha_{m_2}} E(\boldsymbol{\alpha}, \boldsymbol{w})$ can be evaluated exactly using only the partial derivatives $\partial \hat{y}_n / \partial \alpha_m$, since the second partial derivatives of the network output with respect to $\boldsymbol{\alpha}$ are zero, i.e. $\partial^2 \hat{y}_n / \partial \alpha_{m_1} \partial \alpha_{m_2} = 0$. Therefore, only the first derivatives of the model in Eq. (11) with respect to α_m and w_i are needed in the evaluation of the approximated second partial derivatives of $E(\boldsymbol{\alpha}, \boldsymbol{w})$. The partial derivatives with respect to the barrier functions $B_1(\boldsymbol{\alpha}, \boldsymbol{s})$ and $B_2(\boldsymbol{w})$ are evaluated exactly, and the approximated Hessian matrix of $J(\boldsymbol{\Theta})$ is denoted as $\tilde{\boldsymbol{H}}(\boldsymbol{\Theta}) \approx \nabla^2 J(\boldsymbol{\Theta})$.

Next, the search direction $\boldsymbol{p}_l$ is determined (step 3). In the vector $\boldsymbol{p}_l$, the elements corresponding to the parameters $\boldsymbol{\alpha}$, $\boldsymbol{s}$, and $\boldsymbol{w}$ are denoted as $p_{\alpha_m}^l$, $p_{s_m}^l$, and $p_{w_i}^l$, respectively. The step length δ is defined such that the next point $\boldsymbol{\Theta}_l + \delta \boldsymbol{p}_l$ stays strictly inside of the feasible region (step 4). The equalities are

not allowed in step 4, which guarantees that the logarithms are defined in Eq. (14).

The trust region parameter λ is adjusted according to a ratio r^l (steps 5-7). It is the ratio between the actual and the predicted decrease in the value of objective function $J(\boldsymbol{\alpha}, \boldsymbol{s}, \boldsymbol{w})$. The obtained point is accepted if it decreases the current value of objective function (step 8). Steps 2-8 are repeated until the relative decrement in the value of objective function is less than 10^{-4} during the last five iterations or $l \leq 50$,

References

[1] U. Anders and O. Korn, Model selection in neural networks, Neural Networks 12 (1999) 309–323.

[2] G. Baudat and F. Anouar, Kernel-based methods and function approximation, In: Proc. International Joint Conference on Neural Networks (IJCNN 2001), Vol. 2, IEEE, 1244–1249.

[3] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty, Nonlinear programming theory and algorithms, John Wiley & Sons, 1979.

[4] J.C. Bezdek, R.J. Hathaway, R.E. Howard, C.A. Wilson, and M.P. Windham, Local convergence analysis of grouped variable version of coordinate descent, Journal of Optimization Theory and Applications 54 (1987) 471–477.

[5] J.C. Bezdek and R.J. Hathaway, Some notes on alternating optimization, In: Proc. The 2002 AFSS International Conference on Fuzzy Systems (AFSS 2002), Lecture Notes in Computer Science, vol. 2275, Springer-Verlag, 288–300.

[6] N. Benoudjit and M. Verleysen, On the kernel widths in radial-basis function networks, Neural Processing Letters 18 (2003) 139–154.

[7] J. Bi, K.P. Bennett, M. Embrechts, C.M. Breneman, and M. Song, Dimensionality reduction via sparse support vector machines, Journal of Machine Learning Research. 3 (2003) 1229–1243.

[8] C.M. Bishop, Neural networks for pattern recognition, Oxford University Press, 1995.

[9] A.L. Blum and P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1997) 245–271.

[10] G.C. Cawley and N.L.C. Talbot, Reduced rank kernel ridge regression, Neural Processing Letters 16 (2002) 293–302.

[11] G.C. Cawley and N.L.C. Talbot, Fast exact leave-one-out cross-validation of sparse least-squares support vector machines, Neural Networks 17 (2002) 1467–1475.

[12] N. Chapados and Y. Bengio, Input decay: Simple and effective soft variable selection, In: Proc. International Joint Conference on Neural Networks (IJCNN 2001), Vol. 2, IEEE, 1233–1237.

[13] B. Dorizzi, G. Pellieux, F. Jacquet, T. Czernichow, and A. Munoz, Variable selection using generalized RBF networks: Application to the forecast of the French T-bonds, In: Proc. IEEE-IMACS Multiconference on Computational Engineering in Systems Applications (CESA 1996), 1996.

[14] B. Efron and R.J. Tibshirani, An introduction to the bootstrap, Chapman & Hall/CRC, 1993.

[15] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, Least angle regression, The Annals of Statistics 32 (2004) 407–499.

[16] D. François, F. Rossi, V. Wertz, and M. Verleysen, Resampling methods for parameter-free and robust feature selection with mutual information, Neurocomputing 70 (2007) 1276–1288.

[17] J.H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1991) 1–141.

[18] I. Guyon and A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157-1182.

[19] R.J. Hathaway and J.C. Bezdek, Grouped coordinate minimization using Newton's method for inexact minimization in one vector coordinate, Journal of Optimization Theory and Applications 71 (1991) 503–516.

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

[21] G.-B. Huang, P. Saratchandran, and N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation, IEEE Transactions on Neural Networks 16 (2005) 57–67.

[22] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (2006) 489–501.

[23] V. Kadirkamanathan and M. Niranjan, A function estimation approach to sequential learning with neural networks, Neural Computation 5 (1993) 954–975.

[24] R. Kohavi and G.H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1997) 273–324.

[25] P. Leray and P. Gallinari, Feature selection with neural networks, Behaviormetrika 26 (1999), 145–166.

[26] K. Li and J.-X. Peng, Neural input selection–A fast model-based approach, Neurocomputing 70 (2007) 762–769.

[27] A.J. Miller, Subset selection in regression, Chapman and Hall, 1990.

[28] J. Moody and C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 1 (1989) 218–294.

[29] M. Nørgaard, 0. Ravn, N.K. Poulsen, and L.K. Hansen, Neural networks for modelling and control of dynamic systems, Springer-Verlag, 2003.

[30] M.R. Osborne, B. Presnell, B.A. Turlach, A new approach to variable selection in least squares problems, IMA Journal of Numerical Analysis 20 (2000), 389–404.

[31] M.J. Orr, Regularization in the selection of radial basis function centers, Neural Computation 7 (1995) 606–623.

[32] M.J. Orr, Introduction to radial basis functions networks, Technical Report, Edinburgh University, Edinburgh, Scotland, UK, April 1996.

[33] J. Park and I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Computation 3 (1991) 246–257.

[34] J. Platt, A resource-allocating network for function interpolation, Neural Computation 3 (1991) 213–225.

[35] T. Poggio and F. Girosi, Networks for approximation and learning, Proceedings of the IEEE 78 (1990) 1481-1497.

[36] J. Reunanen, Overfitting in making comparisons between variable selection methods, Journal of Machine Learning Research 3 (2003) 1371–1382.

[37] A.N. Refenes, A. Zapranis, and J. Utans, Neural model identification, variable selection, and model adequacy, In: Proc. Conference on Neural Networks in the Capital Markets (NNCM 1996), 1996.

[38] F. Rossi, Attribute suppression with multi-layer perceptron, In: Proc. IEEE-IMACS Multiconference on Computational Engineering in Systems Applications (CESA 1996), 1996.

[39] D.W. Ruck, S.K. Rogers, M. Kabrisky, Feature selection using a multilayer perceptron, Journal of Neural Network Computing 2 (1990), 40–48.

[40] C. Saunders, A. Gammerman, and V. Vovk, Ridge regression learning algorithm in dual variables, In: Proc. The 15th International Conference on Machine Learning (ICML 1998), 515-521

[41] T. Similä and J. Tikka, Input selection and shrinkage in multiresponse linear regression, Computational Statistics & Data Analysis 52 (2007) 406–422.

[42] M. Schmidt, G. Fung, and R. Rosales, Fast optimization methods for L1 regularization: A comparative study and two new approaches, In: Proc. The 18th European Conference on Machine Learning (ECML 2007), Lecture Notes in Computer Science, vol. 4701, Springer-Verlag, 286–297.

[43] A.J. Smola and B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (2004) 199–222.

[44] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, Least squares support vector machines, World Scientifc Publishing, 2002.

[45] J.A.K. Suykens, J. De Brabanter, L. Lukas, and J. Vadewalle, Weighted least squares support vector machines: robustness and sparse approximation, Neurocomputing 48 (2002) 85–105.

[46] R. Tibshirani, Regression shrinkage and selection via the LASSO, Journal of the Royal Statistical Society, Series B (Methodological) 58 (1996) 267–288.

[47] J. Tikka, Input selection for radial basis function networks by constrained optimization, In: Proc. The 17th International Conference on Artificial Neural Networks (ICANN 2007), Lecture Notes in Computer Science, Vol. 4668, Springer-Verlag, 239–248.

[48] J. Tikka and J. Hollmén, Sequential input selection algorithm for long-term prediction of time series, Neurocomputing 71 (2008) 2604-2615.

[49] J. Tikka and J. Hollmén, Selection of important inputs for RBF network using partial derivatives, In: Proc. The 16th European Symposium on Artificial Neural Networks (ESANN 2008), 167–172.

[50] T. Van Gestel, J.A.K. Suykens, B. De Moor, and J. Vandewalle, Automatic relevance determination for least squares support vector machine regression In: Proc. International Joint Conference on Neural Networks (IJCNN 2001), Vol. 4, IEEE, 2416-2421.

[51] Th. Villman, F. Schleif, and B. Hammer, Comparison of relevance learning vector quantization with other metric adaptive classification methods, Neural Networks 19 (2006), 610–622.

[52] L. Yingwei, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks, Neural Computation 9 (1997) 461–478.