Expert Systems with Applications 39 (2012) 11029–11035

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Sparse multikernel support vector regression machines trained by active learning

V. Ceperic [a,b,*], G. Gielen [a], A. Baric [b]

[a] Department of Electrotechnical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, Leuven, Belgium
[b] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia

ARTICLE INFO

Keywords:
Support vector machines
Support vector regression
Multikernel
Sparse models
Active learning

ABSTRACT

A method for the sparse multikernel support vector regression machines is presented. The proposed method achieves a high accuracy versus complexity ratio and allows the user to adjust the complexity of the resulting models. The sparse representation is guaranteed by limiting the number of training data points for the support vector regression method. Each training data point is selected based on its influence on the accuracy of the model using the active learning principle. A different kernel function is attributed to each training data point, yielding multikernel regressor. The advantages of the proposed method are illustrated on several examples and the experiments show the advantages of the proposed method.

1. Introduction

Cortes and Vapnik (1995) presented support vector machines (SVM), a set of related supervised learning methods used for classification and regression. Since then, support vector machines have generated a lot of interest in the machine learning community due to their excellent performance in a variety of learning problems, e.g. text categorisation (Kuo & Yajima, 2010; Hao, Chiang, & Tu, 2007), image analysis (Guo, Gunn, Damper, & Nelson, 2008; Chang, Wang, & Li, 2009), solving problems in bioinformatics (Ben-Hur, Ong, Sonnenburg, Schölkopf, & Rätsch, 2008; Zhong, He, Harrison, Tai, & Pan, 2007), bankruptcy prediction (Härdle, Moro, & Hoffmann, 2010; Shin, Lee, & Kim, 2005). Some of the additional reasons for the wide use of SVMs are: theoretical guarantees about their performance, lower susceptibility to local minima and higher immunity to the increase of model complexity that is usually associated with adding extra dimensions to the model input space.

SVMs were initially applied for classification problems (Cortes & Vapnik, 1995), but soon after the introduction, the formulation was extended to regression problems (Drucker, Burges, Kaufman, Smola, & Vapnik, 1996; Vapnik, Golowich, & Smola, 1996). Support vector machines seem to offer excellent generalisation properties on real-life regression and classification problems while they are still capable of producing sparse models. The common formulation of SVM regression is Vapnik's ε-tube support vector regression (ε-SVR) (Drucker et al., 1996; Vapnik et al., 1996). The model produced by ε-SVR depends only on a subset of the training data while ignoring any training data within the threshold ε to the model prediction. This step unveils the potential problem: if the threshold ε is small, then the model will depend on a larger number of the overall training data, thus making the solution non-sparse, as shown in Guo, Zhang, and Zhang (2010).

In general, sparse regression models are simplified models. They can exhibit high accuracy to complexity ratio. There are several benefits to sparse regression models:

- tendency to avoid over-fitting: if the model is less complex, it is less likely that the model will over-fit the data;
- decrease of computational burden in the phase of active use, e.g. the evaluation time for the SVM model is directly proportional to the number of support vectors and, if that number is decreased, the execution speed is increased;
- ability to generalise well: generalisation and over-fitting are two closely related terms in such a way that if the chance for over-fitting is decreased, the generalisation quality of the model is increased. It should be noted that the sparsity of the model, by itself, is not a guarantee that the model will generalise well. One of the common techniques for improving generalisation and overcoming over-fitting is regularisation (Neumaier, 1998). Support vector machines in their training formulation have a regularisation term which helps to reduce over-fitting.

Several approaches for sparse modelling with (single) kernel based methods have already been presented, e.g. relevance vector machines (RVM) (Tipping, 2001), adaptive sparse supervised learning (ASSL) (Figueiredo, 2003), kernelised least absolute shrinkage and selection operator regression (KLASSO) (Roth, 2004), fixed-size least squares support vector machines (FS-LSSVM) (Brabanter, Brabanter, Suykens, & Moor, 2010; Suykens, 2002).

If a data set exhibits different data trends in different regions, the single kernel to construct the model usually does not achieve the satisfying accuracy or the model is over-complicated. Recently

* Corresponding author at: Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia.
E-mail address: vladimir.ceperic@ieee.org (V. Ceperic).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.021

developed multikernel SVM learning algorithms, e.g. (Bi, Zhang, & Bennett, 2004; Lanckriet, Cristianini, Bartlett, Ghaoui, & Jordan, 2004; Nguyen & Tay, 2008; Ong, Smola, & Williamson, 2005; Zhao & Sun, 2011; Zhao, Sun, & Zou, 2010) have demonstrated better accuracy of the models when using multiple kernels or the combination of kernels rather than a single fixed kernel. Therefore, we adapt the multikernel SVM principle.

In this paper, the focus will be on increasing the accuracy while limiting the complexity of the multikernel SVM regression model, when compared with other state-of-the-art multikernel SVM methods. To achieve this goal, the active learning principle is adapted to be used for multikernel ε-tube support vector regression. The sparsity is guaranteed by limiting the number of training data points for ε-SVR. To this end, the training data set is iteratively constructed, based on the influence of each training data point on the accuracy of the model. In this way, the active learning principle can be used to reduce the complexity of the SVM if the iterative procedure is stopped earlier, which enables the development of sparse solutions based on the least number of support vectors. On the other hand, this incremental construction of the model enables robust optimisation of the kernel parameters for each support vector.

The paper is organised as follows. Section 2 gives a brief overview of support vector regression. Section 3 presents sparse multikernel support vector regression modelling by using an active learning principle. In Section 4 several modelling experiments are reported together with the comparison of the performance of various methods. Finally, conclusions are given in Section 5.

2. Support vector regression

The ε-tube support vector regression is briefly introduced, as it is the base of the models proposed in this paper.

Firstly the case of linear regression is considered. The training data is denoted as:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}\}_{i=1}^{l}, \tag{1}$$

where y_i is the real number associated to the p-dimensional input vector $\mathbf{x}_i$. Let X denote the space of input patterns, $X \subset \mathbb{R}^p$. Let l in (1) denote the number of training data samples.

The first goal is to find a linear function f, which will later be extended to the nonlinear case, that best describes Eq. (1) in the form of

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{2}$$

where $\mathbf{w} \in X$ is the weight vector, $b \in \mathbb{R}$ is a constant and $\langle \cdot, \cdot \rangle$ denotes a dot product in X.

The function f should be as flat as possible. Flatness of the function f means that one seeks a small weight vector $\mathbf{w}$. An overview of flatness definitions of such functions is given in Schölkopf and Smola (2001). A simple way to ensure required flatness is to minimise the norm $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle$. This problem can be written as a quadratic optimisation problem:

$$\begin{aligned} &\text{minimise} \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \varepsilon \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \varepsilon \end{cases}, \end{aligned} \tag{3}$$

for $i = 1, \ldots, l$.

The assumption in Eq. (3) is that the function f as in (2) exists and that it can approximate the training data pairs with accuracy ε (Vapnik et al., 1996). If this assumption is correct, the optimisation problem in (3) can be solved. Sometimes this may be impossible, or in other cases some extra error could be allowed in order to improve generalisation. In that situation, analogously to the "soft margin" loss function in Cortes and Vapnik (1995), the slack variables ξ_i and ξ_i^* can be introduced (Drucker et al., 1996; Smola & Schölkopf, 2004; Vapnik et al., 1996) (Fig. 1) and described as:

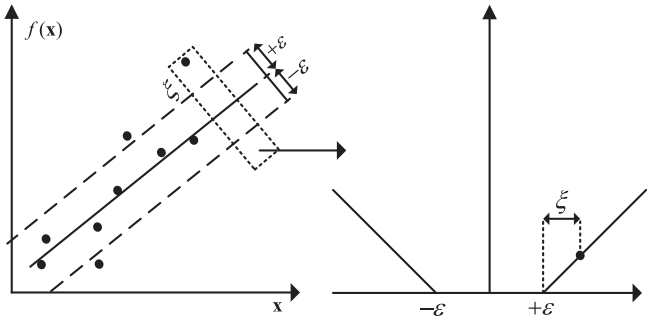

Fig. 1. The soft margin loss setting for a linear SVM.

$$\begin{aligned} &\text{minimise} \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{subject to} \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b & \varepsilon + \xi_i \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & 0 \end{cases}, \end{aligned} \tag{4}$$

for $i = 1, \ldots, l$.

The constant C in (4) determines the balance between the flatness of f and the amount of *tolerated* deviations larger than ε. In other words, the constant C can be considered as a tuning parameter between model complexity and required training data set model accuracy.

The optimisation problem in (4) can be solved more easily in its dual formulation (Smola & Schölkopf, 2004). The idea is to construct a Lagrange function L from both the primal objective function and the corresponding constraints by introducing a dual set of variables. The function L has a saddle point with respect to the primal and dual variables at the optimal solution (Smola & Schölkopf, 2004):

$$\begin{aligned} L := \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}(\eta_i \xi_i + \eta_i^* \xi_i^*) \\ - \sum_{i=1}^{l}\alpha_i(\varepsilon + \xi_i - y_i + \langle \mathbf{w}, \mathbf{x}_i \rangle + b) - \sum_{i=1}^{l}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b) \end{aligned} \tag{5}$$

where $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ are the Lagrangian multipliers with the following constraints:

$$\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geqslant 0. \tag{6}$$

The partial derivatives of L with respect to the primal variables $(\mathbf{w}, b, \xi_i, \xi_i^*)$ should be equal to zero (Smola & Schölkopf, 2004):

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \tag{7}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)\mathbf{x}_i = 0 \tag{8}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \eta_i = 0 \tag{9}$$

$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0 \tag{10}$$

Placing Eqs. (7)–(10) into (5) yields the dual optimisation problem (Smola & Schölkopf, 2004):

$$\begin{aligned} &\text{maximise} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{cases} \\ &\text{subject to} \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \quad \text{and } \alpha_i, \alpha_i^* \in [0, C], \end{aligned} \tag{11}$$

From (11) it can be seen that the training complexity of a support vector machine is dependent on the number of α_i, α_i^* coefficients l which is equal to the number of training data samples.

In (11) the dual variables η_i and η_i^* are eliminated through the use of (9) and (10).

The support Vector (SV) expansion (Smola & Schölkopf, 2004) can be written as

$$\mathbf{w} = \sum_{i=1}^{l} (\alpha_i + \alpha_i^*)\mathbf{x}_i \tag{12}$$

and therefore the regression function

$$f(\mathbf{x}) = \sum_{i=1}^{l} (\alpha_i + \alpha_i^*)\langle \mathbf{x}_i, \mathbf{x}\rangle + b, \tag{13}$$

where $\mathbf{w}$ is the weight vector that can completely be described as a linear combination of the training patterns $\mathbf{x}_i$. Note that even when evaluating $f(\mathbf{x})$, no explicit computation of $\mathbf{w}$ is necessary. The $\mathbf{x}_i$ vectors corresponding to non-zero $(\alpha_i + \alpha_i^*)$ are called the support vectors. From (13) it can be seen that the evaluation complexity of a support vector machine is dependent on the number of support vectors.

The computation of b in (13) can be accomplished by exploiting the Karush–Kuhn–Tucker (KKT) conditions (Karush, 1939; Kuhn & Tucker, 1951). Further considerations about choosing the optimal parameter b can be found in Keerthi, Shevade, Bhattacharyya, and Murthy (2001) and Smola and Schölkopf (2004).

Up to now, we have discussed the linear regression case. The next step towards general function approximation is to make the SVM nonlinear. This can be achieved by simply preprocessing the training patterns $\mathbf{x}_i$ using the map function Φ: $X \to F$. The function Φ transforms the input space into some feature space F where it is possible to apply the standard SV regression algorithm (Smola & Schölkopf, 2004). Unfortunately, the evaluation of the function Φ can be computationally expensive.

Boser, Guyon, and Vapnik (1992) proved that if we can find a new function that satisfies $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x})\rangle$ in the feature space F, it is possible to avoid the computationally expensive calculation of the function Φ. This can be done by using so called kernel functions.

A kernel function on two vectors $\mathbf{v}$ and $\mathbf{z}$ is the function $K : X \to \mathbb{R}$ that satisfies:

$$K(\mathbf{v}, \mathbf{z}) = \langle \Phi(\mathbf{v}), \Phi(\mathbf{z})\rangle. \tag{14}$$

In this paper, the radial basis function (RBF) is used as a kernel function:

$$\begin{aligned} K(\mathbf{v}, \mathbf{z}) &= \exp(-\gamma\|\mathbf{v} - \mathbf{z}\|^2) \\ &= \exp(-\gamma(\langle \mathbf{v}, \mathbf{v}\rangle + \langle \mathbf{z}, \mathbf{z}\rangle - 2\langle \mathbf{v}, \mathbf{z}\rangle)), \quad \text{for } \gamma > 0. \end{aligned} \tag{15}$$

Other kernel functions can be used with the proposed method too.

The ε-SVR implementation used in this paper is based on the LIBSVM tool (Chang & Lin, 2001).

3. Sparse multikernel support vector regression by using active learning principle

This paper presents the algorithm named sparse multikernel active learning support vector regression (MK-ALSVR). The formulation of ε-SVR, as described in the previous section, is extended to facilitate various kernels in the same model and to reduce the model complexity and improve the accuracy by limiting the maximum number of support vectors and using one kernel per support vector strategy. The support vectors in the SVM function approximation are chosen from the set of training data points that is iteratively constructed by using the active learning principle. Unlike the commonly used multikernel SVM approach to construct the kernel that is actually a combination of several kernels, e.g. as described in Zhao and Sun (2011), an active learning principle is used to gain additional model sparsity, by applying different kernel (or the same type of kernel but with different parameters) for each support vector.

The basic idea behind MK-ALSVR is the same as in all kernel based methods, i.e. first preprocess the data by some non-linear mapping and then apply the linear algorithm. Therefore, the MK-ALSVR model consists of two layers:

- nonlinear preprocessing layer. In each iteration of the algorithm, one kernel function is added to the model. The kernel function hyper-parameters are optimised using the PSO algorithm;
- linear, ε-tube SVR layer, as implemented in LIBSVM tool. The ε-tube SVR hyper-parameters are optimised using the PSO algorithm.

The MK-ALSVR algorithm is summarised below and presented in the form of a flow chart in Fig. 2. The numbers in the flow chart correspond to the steps in the Algorithm 1.

Algorithm1: Training algorithm for sparse multikernel active learning support vector regression (MK-ALSVR)

Step 1: Start with one data point in the current training data set (CTDS) (i.e. use the procedure for the selection of initial CTDS as described in this Section).
Step 2: Set the variable *iteration* to 1.
Step 3: Add one kernel function to the model.
Step 4: Optimise the hyper-parameters to fit the training data set with the current model (e.g. with PSO as described in Section 2).
Step 5: Build the model using the best parameter set determined in Step 4.
Step 6: Calculate the error vector (e.g. the difference between the model output and the full training data set).
Step 7: Identify the next point to include in the CTDS using the error vector.
Step 8: Increase the value of the variable *iteration* by 1.
Step 9: If the stopping criterion is not met, go to Step 3.

In Zhao et al. (2010), limited training data set is also considered for multikernel support vector regression. Reducing samples strategy is utilised to select training samples for admissible functions. This is in contract with the approach in this paper which incrementally constructs the training data set and the model itself.

The training data set in MK-ALSVR method is initially set to have one training data point. This training data set is referred to as the current training data set (CTDS) to avoid confusion with the overall training data set. The procedure for the initial selection of the CTDS can be based on: minimum or maximum value of the output variable, random selection, user experience, trial-and-error procedure, cross-validation (CV), etc.

The procedure used in this paper for the selection of the initial CTDS is very simple:

1. four values with highest value of the output are identified;
2. one input–output pair is selected from the pool identified in Step 1;
3. the MK-ALSVR model is constructed and the mean square error (MSE) on the full training data set is calculated;
4. Steps 2 and 3 are repeated until all combinations are tested;
5. the input–output values that constructs the model with the least MSE is selected as the initial CTDS.

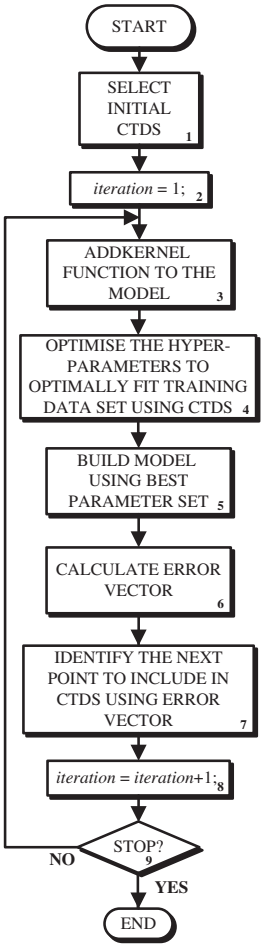

Fig. 2. MK-ALSVR training flow chart.

It should be noted that if the input–output pair that is selected as initial CTDS is not optimal, this point can still be neglected as the ε-SVR does not have to select all point in the CTDS as support vectors. The crucial step in MK-ALSVR is the addition of new points to the CTDS in each iteration (Step 7). The next point to be included in the CTDS is determined by the model performance on the full training data set, i.e. by using an active learning principle. The point (or points) where the model performance is worst, is included in the CTDS. In the experiments reported in Section 4, several points in the area where the model performance is worst are included one by one in the CTDS; it is tested how they influence the model quality on the full training data set and then the one point that contributes most to the model quality is selected. At the end of the iterative procedure, the total number of kernel functions is equal to the number of points in the CTDS.

Standard SVM tools (e.g. LIBSVM) can be used in MK-ALSVR for the linear ε-SVR layer. An additional advantage is that the user does not have to change the code of the SVM tool, i.e. the SVM tool can be used to formulate linear ε-SVR solutions, while the kernel preprocessing and iterative construction of the training data set can be done in any programming language or environment, e.g. MATLAB.

The stopping criterion to end the iterative procedure in the MK-ALSVR algorithm (Step 9) is set if:

- the model reached the user specified complexity limit, i.e. the maximum number of support vectors is reached;
- the model reached the user specified accuracy.

In this paper, Gaussian RBFs are used as kernel functions with the centre defined by its corresponding SV. Other kernel functions can be used or even mixed.

3.1. Optimisation of the model hyper-parameters

One of the key questions in the practical application of the active learning principle in MK-ALSVR is how to select the optimal hyper-parameters. The optimal choice of the hyper-parameters is critical because an active learning principle relies on the tuned regressor in each iteration. Therefore, the selection of the optimal hyper-parameters is embedded into the training algorithm; MK-ALSVR optimises kernel and ε-SVR hyper-parameters in each iteration. The optimisation of the hyper-parameters in each iteration of the algorithm ensures a near-optimal model for the current data set.

For the ε-tube support vector regression machine, as in Fig. 1, two parameters are the most critical:

- the regularisation parameter C;
- the width of the ε-tube, i.e. the parameter ε.

Kernel functions might require the selection of additional parameters, e.g. γ in Eq. (15). The choice of the SVM parameters directly influences the sparsity of the model, e.g. if ε is set too small, the set of support vectors might include the whole training data set. The SVM parameters and the kernel parameters are often referred to as hyper-parameters, also in this paper.

The standard approach to find a (near) optimal C and ε is to perform a grid search (GS), as suggested in Hsu, Chang, and Lin (2003). It should be noted that in Hsu et al. (2003) the grid search for optimal parameters is performed for classification problems. This grid search approach was adopted by many researchers and used also for SVM regression, e.g. (Bao, Lu, & Zhang, 2004; Ceperic & Baric, 2004). Hsu et al. in Hsu et al. (2003) suggested exponentially growing sequences of SVM parameters as a practical method to identify optimal parameters. The problem with the grid search strategy is that it is very time consuming (Bao et al., 2004).

In the last few years, many researchers used the particle swarm optimisation (PSO) method to optimise the parameters of the SVM, e.g. (Cao & Xu, 2007; Jiang & Yuan, 2007; Fei, Liu, Zeng, & Miao, 2008; Liu, Zhuang, & Liu, 2011). Particle swarm optimisation is an optimisation algorithm based on principles of swarm intelligence that finds a solution to an optimisation problem in a search space based on mimicking the social behaviour of swarms. In this paper, a variant of PSO from the toolbox PSwarmM (Vaz & Vicente, 2007) is used for the hyper-parameters optimisation.

We assume that the optimal hyper-parameters of the model with similar number of SV on the same (overall) training data set is also similar. Therefore, in the incremental training algorithm procedure, the (near) optimal hyper-parameters determined in previous iteration are set as the initial condition (i.e. as one PSO particle) of the next iteration.

3.2. Illustrative example of the model training algorithm

The algorithm iteration steps are illustrated on a simple example, a multiple Gaussian RBF data set constructed as a combination of nine Gaussian RBF functions, as shown on Fig. 3(a). The Gaussian RBF centres, widths and amplitudes are chosen randomly and they are allowed to overlap.

Fig. 3 illustrates how kernel functions shape the model. Fig. 3(b) shows the model with 4 SV, while Fig. 3(c) and (d) show the model with 7 and 10 SV. Each additional SV with accompanying kernel function contributes to the model accuracy.

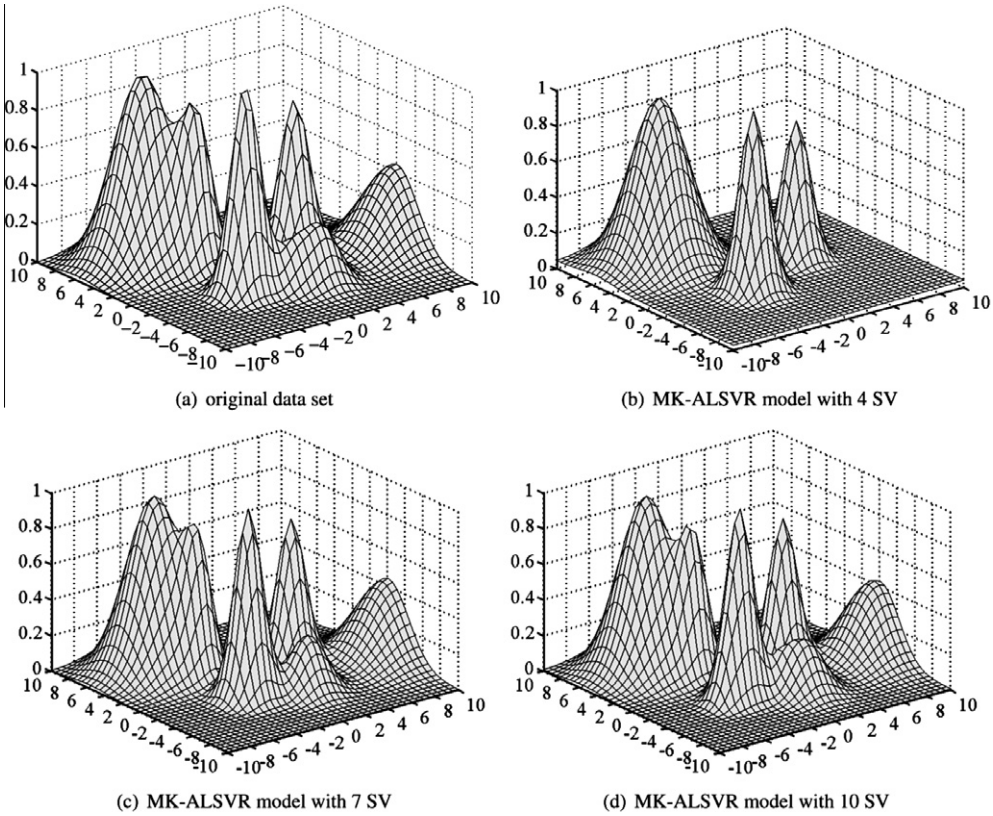

Fig. 3. Illustrative example of the model training algorithm on the multiple Gaussian RBF data set.

Fig. 4 shows the accuracy vs. complexity plot of the MK-ALSVR training algorithm on the multiple Gaussian RBF data set. Accuracy is defined as the mean square error (MSE) and the complexity is defined as the number of SVs in the model. Fig. 4 shows that the MSE value decreases rapidly for the first 6 SVs and then continues to decrease nearly linearly with each additional SV added to the model. The rapid decrease in model MSE for the first 6 SVs can be attributed to the 6 largest Gaussian RBFs in the training data set, as visible in Fig. 3(a).

3.3. Algorithm complexity

Another important factor related to all machine learning algorithms is the training time and memory space requirement. A common quadratic programming optimisation problem (as found in Eq. (11)) has a training time and memory space requirement proportional to $O(l^3)$ and $O(l^2)$ (Guo et al., 2010), respectively, where l is the total number of variables. In this case l depends on the total number of training data points. Several algorithms exist, e.g. sequential minimal optimisation (SMO) (Platt, 1999), that reduce the training time requirement of the SVM problem. SMO has a typical training time requirement between $O(l)$ and $O(l^{2.3})$. The training time requirement is further reduced in LIBSVM, as described in Fan, Chen, and Lin (2005).

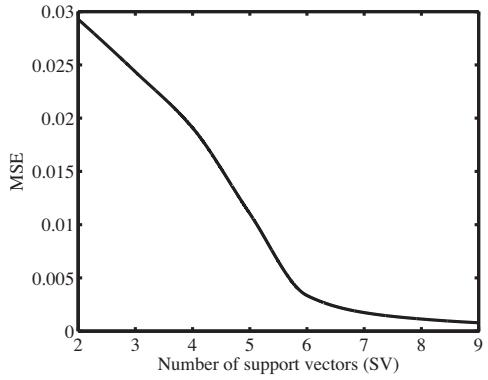

Fig. 4. Accuracy vs. complexity plot of the MK-ALSVR training algorithm for the multiple Gaussian RBF data set.

Following assumptions are made:

- the training time requirement of the algorithm used to solve the SVM problem is $O(l)$;
- p is the number of optimisation iterations needed to find the near-optimal hyper-parameters of the plain single kernel ε-SVR, i.e. total training time requirement is $p \cdot O(l)$;
- r is the number of optimisation iterations needed to find the near-optimal hyper-parameters in each iteration of the MK-ALSVR.

In the assumptions, several factors are neglected for easier comparison, e.g. the training time for the ε-SVR with (single) RBF kernel is considerably higher than the training time for the linear ε-SVR. Under those assumptions the training time requirement of the sparse multikernel active learning approach to support vector regression is:

$$r \cdot O(1) + r \cdot O(2) + \cdots + r \cdot O(T) = r \cdot O\left(\frac{T^2 + T}{2}\right), \tag{16}$$

where T is the final number of training data in the CTDS. To see when this approach is approximately equivalent to the plain ε-SVR with respect to the training time requirement, it is possible

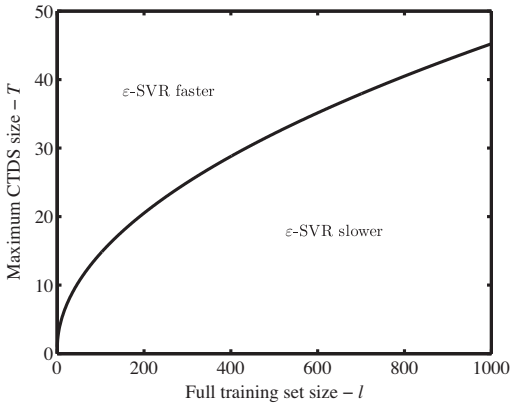

Fig. 5. The theoretical maximum number of training points in CTDS with the same training time requirement when compared to plain ε-SVR and under assumption that $p = r$.

to calculate the tipping point, i.e. the maximum number of training data in the iteratively constructed CTDS which altogether requires an equal training time requirement as the full set with l data points. From Eq. (16) and under assumption that $p = r$ it follows that the tipping point is reached at:

$$T = \frac{\sqrt{(8l+1)}}{2} - \frac{1}{2}. \tag{17}$$

Fig. 5 shows the theoretical maximum number of training points in the CTDS with the same training time requirement when compared to the plain ε-SVR algorithm. It can be noticed that if T/l is small i.e. when the final CTDS size is small compared to the full training set size, the active learning approach is time efficient. In the lower part of Fig. 5, the active learning approach is faster than the plain ε-SVR method. This is an important factor to consider when using the proposed algorithm. It should be noted that the main focus of this paper is to develop sparse solutions based on the least number of support vectors selected in a constructive manner.

4. Experiments-comparison with the state-of-the-art

In this section, experiments on the synthetic and real-world data sets are presented to examine the sparsity and overall performance of the MK-ALSVR training algorithm.

Four experiments on the following well known function approximation problems are performed:

- the mixture function (Zheng, Wang, & Zhao, 2006), defined as $f(x) = \frac{1}{\sqrt{2\pi}\cdot 0.3}\exp\left(-\frac{(x-2)^2}{2\cdot 0.3^2}\right) + \frac{1}{\sqrt{2\pi}\cdot 1.2}\exp\left(-\frac{(x-7)^2}{2\cdot 1.2^2}\right)$. The data set represents the mixture of two Gaussian distributions. Thirty training sets of size 100 are created. Normal distribution noise $(0, 0.05^2)$ is added to each training data set;
- the oscillating function (Zheng et al., 2006) defined as $r(x) = \sin\left(\frac{2\pi(0.35\times 10+1)}{0.35x+1}\right)$. As in the case of mixture function, 30 training sets of size 100 are created. Normal distribution noise $(0, 0.1^2)$ is added to each training data set;
- the auto price data set (Kibler, Aha, & Albert, 1989).[1] The training data set consists of 120 input–output pairs, while test data sets consist of 39 input–output pairs;
- the triazines data set (Hirst, King, & Sternberg, 1994). The training data set consists of 140 input–output pairs, while test data sets consist of 46 input–output pairs;

[1] The data is preprocessed as described at http://www.liaad.up.pt/ltorgo/Regression/price.html. The same preprocessing procedure is used in Zhao and Sun (2011).

Table 1
Modelling results on the synthetic data sets.

Data set	Algorithm	RMSE	SV
Mixture	MS-SVR[a]	0.0202	15.8
	MSLP-SVR[a]	0.0181	13
	MK-ALSVR	0.016	3
Oscillating	MS-SVR[a]	0.0581	27
	MSLP-SVR[a]	0.0444	25
	MK-ALSVR	0.0424	15

[a] Reported in Zhao and Sun (2011).

Table 2
Modelling results on the real-world data sets.

Data set	Algorithm	RMSE	SV
Auto price	MSLP-SVR[a]	2.22E3	65
	MK-ALSVR	2.13E3	3
Triazines	MSLP-SVR [a]	1.55E − 1	51
	MK-ALSVR	1.47E − 1	3

[a] Reported in Zhao and Sun (2011).

The performance is measured by the model accuracy and its complexity. Accuracy is described as the root mean square error (RMSE) of the model and complexity is described as the number of support vectors in the model. For the mixture and oscillating data set, the averaged RMSE is calculated between the estimated functions and the true, non-noised function on the set of 30 experiments.

The MK-ALSVR algorithm is compared with state-of-the-art multikernel support vector regression algorithms reported in recent papers by Zhao and Sun (2011) (multikernel semiparametric linear programming support vector regression-MSLP-SVR) and Zheng et al. (2006) (multiscale support vector regression-MS-SVR). The algorithm proposed by Zhao et al. (2010) also works on limited data set as MK-ALSVR, but the comparison with this method is omitted as the improved accuracy is reported in Zhao and Sun (2011).

Table 1 shows the performance of the MK-ALSVR algorithm on the synthetic data sets (mixture and oscillating function), while Table 2 shows the performance on the real-world data sets (auto price and triazines data). MK-ALSVR shows comparable or better performance with lesser number of SV, especially in the case of real-world data sets (auto price and triazines) and the mixture function data set where only 3 SVs are enough to adequately model the data set. It should be noted that MK-ALSVR model is actually significantly simpler than MSLP-SVR and MS-SVR with the same number of SVs because there is only one kernel (Gaussian RBF) attributed to each SV.

5. Conclusion

The training algorithm MK-ALSVR has been presented for the sparse solution of multikernel support vector regression machines by adapting the active learning principle. The proposed algorithm achieves a high accuracy while keeping the complexity low, i.e. the resulting models have a low number of support vectors, which makes the models efficient in execution. The maximum number of support vectors can be defined by the user. The optimisation procedure for the algorithms's hyper-parameters is built into the training algorithm.

The advantages of the algorithm have been illustrated on several well-known benchmark examples and compared to the state-of-the-art. The performance of the proposed algorithms is shown to be better or comparable than other state-of-the-art multikernel regression methods despite the reduced complexity of the model.

Acknowledgements

The authors acknowledge the support of the IWT GoldenGates project and ON Semiconductor, Belgium.

Vladimir Ceperic acknowledges and thanks for the financial support of the National Foundation for Science, Higher Education and Technological Development of Croatia within the programme "Scholarships for Doctoral Students".

References

Bao, Y., Lu, Y., & Zhang, J. (2004). Forecasting stock price by SVMs regression. In *Artificial intelligence: Methodology, systems, and applications*. *Lecture notes in computer science* (pp. 295–303). Berlin/ Heidelberg: Springer.

Ben-Hur, A., Ong, C. S., Sonnenburg, S., Schölkopf, B., & Rätsch, G. (2008). Support vector machines and kernels for computational biology. *PLoS Computational Biology*, 4 (10).

Bi, J., Zhang, T., & Bennett, K. P. (2004). Column-generation boosting methods for mixture of kernels. In *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining KDD '04* (pp. 521–526). New York, NY, USA: ACM.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual ACM workshop on computational learning theory* (pp. 144–152). New York, NY, USA: ACM.

Brabanter, K. D., Brabanter, J. D., Suykens, J., & Moor, B. D. (2010). Optimized fixed-size kernel models for large data sets. *Computational Statistics and Data Analysis, 54*(6), 1484–1504.

Cao, C., & Xu, J. (2007). Short-term traffic flow predication based on PSO-SVM. In Q. Peng, K. C. P. Wang, Y. Qiu, Y. Pu, X. Luo, & B. Shuai (Eds.). *Proceedings of the first international conference on transportation engineering* (Vol. 246, pp. 28). Chengdu, China: American Society of Civil Engineers (ASCE).

Ceperic, V., & Baric, A. (2004). Modeling of analog circuits by using support vector regression machines. In *Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems ICECS 2004* (pp. 391–394).

Chang, C. -C., & Lin, C. -J. (2001). LIBSVM: A library for support vector machines, http://www.csie.ntu.edu.tw/cjlin/libsvm.

Chang, C.-Y., Wang, H.-J., & Li, C.-F. (2009). Semantic analysis of real-world images using support vector machine. *Expert Systems with Applications, 36*(7), 10560–10569.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., & Vapnik, V. (1996). Support vector regression machines. In M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9* (pp. 155–161).

Fan, R.-E., Chen, P.-H., & Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research, 6*, 1889–1918.

Fei, S., Liu, C., Zeng, Q., & Miao, Y. (2008). Application of particle swarm optimization-based support vector machine in fault diagnosis of turbo-generator. In *IITA '08: Proceedings of the 2008 second international symposium on intelligent information technology application* (pp. 1040–1044). Washington, DC, USA: IEEE Computer Society.

Figueiredo, M. A. T. (2003). Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis Machine Intelligence, 25*(9), 1150–1159.

Guo, B., Gunn, S., Damper, R., & Nelson, J. (2008). Customizing kernel functions for SVM-based hyperspectral image classification. *IEEE Transactions on Image Processing, 17*(4), 622–629.

Guo, G., Zhang, J.-S., & Zhang, G.-Y. (2010). A method to sparsify the solution of support vector regression. *Neural Computing and Applications, 19*(1), 115–122.

Hao, P.-Y., Chiang, J.-H., & Tu, Y.-K. (2007). Hierarchically SVM classification based on support vector clustering method and its application to document categorization. *Expert Systems with Applications, 33*(3), 627–635.

Härdle, W. K., Moro, R., & Hoffmann, L. (2010). Learning Machines Supporting Bankruptcy Prediction, SFB 649, Discussion Paper 2010-032, Sonderforschungsbereich 649, Humboldt Universität zu Berlin, Germany.

Hirst, J. D., King, R. D., & Sternberg, M. J. E. (1994). Quantitative structure-activity relationships by neural networks and inductive logic programming. ii. the inhibition of dihydrofolate reductase by triazines. *Journal of Computer-Aided Molecular Design, 8*(4), 421–432.

Hsu, C. -W., Chang, C. -C., & Lin, C. -J. (2003). A practical guide to support vector classification, technical report. Department of Computer Science and and Information Engineering, National Taiwan University.

Jiang, M.-H., & Yuan, X.-C. (2007). Construction and application of PSO-SVM model for personal credit scoring. In *ICCS '07: Proceedings of the 7th international conference on computational science Part IV* (pp. 158–161). Berlin, Heidelberg: Springer-Verlag.

Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints, Master's thesis, University of Chicago.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation, 13*(3), 637–649.

Kibler, D. F., Aha, D. W., & Albert, M. K. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence, 5*, 51.

Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear programming. In *Proceedings of 2nd Berkeley symposium on mathematical statistics and probabilistics*. Berkeley: University of California Press.

Kuo, T.-F., & Yajima, Y. (2010). Ranking and selecting terms for text categorization via SVM discriminate boundary. *International Journal of Intelligence System, 25*(2), 137–154.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research, 5*, 27–72.

Liu, L.-X., Zhuang, Y.-Q., & Liu, X.-Y. (2011). Tax forecasting theory and model based on SVM optimized by PSO. *Expert Systems with Applications, 38*(1), 116–120.

Neumaier, A. (1998). Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review, 40*(3), 636–666.

Nguyen, C., & Tay, D. B. H. (2008). Regression using multikernel and semiparametric support vector algorithms. *IEEE Signal Processing Letters, 15*, 481–484.

Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research, 6*, 1043–1071.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: Support vector learning* (pp. 185–208). Cambridge, MA, USA: MIT Press.

Roth, V. (2004). The generalized LASSO. *IEEE Transactions on Neural Networks, 15*(1), 16–28.

Schölkopf, B., & Smola, A. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond (adaptive computation and machine learning)*. Cambridge, MA, USA: The MIT Press.

Shin, K.-S., Lee, T. S., & Kim, H.-J. (2005). An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications, 28*(1), 127–135.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*(3), 199–222.

Suykens, J. A. K. (2002). *Least squares support vector machines*. River Edge, NJ: World Scientific.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research, 1*, 211–244.

Vapnik, V., Golowich, S. E., & Smola, A. (1996). Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems* (Vol. 9, pp. 281–287). MIT Press.

Vaz, A., & Vicente, L. (2007). A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization, 39*(2), 197–219.

Zhao, Y.-P., & Sun, J.-G. (2011). Multikernel semiparametric linear programming support vector regression. *Expert System with Applications, 38*, 1611–1618.

Zhao, Y.-P., Sun, J.-G., & Zou, X.-Q. (2010). Reducing samples for accelerating multikernel semiparametric support vector regression. *Expert System with Applications, 37*, 4519–4525.

Zheng, D., Wang, J., & Zhao, Y. (2006). Non-flat function estimation with a multi-scale support vector regression. *Neurocomputing, 70*(1–3), 420–429. neural Networks - Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04), 7th Brazilian Symposium on Neural Networks.

Zhong, W., He, J., Harrison, R., Tai, P. C., & Pan, Y. (2007). Clustering support vector machines for protein local structure prediction. *Expert Systems with Applications, 32*(2), 518–526.