Knowledge-Based Systems 65 (2014) 21–30

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Bagging-like metric learning for support vector regression

Peng-Cheng Zou *, Jiandong Wang, Songcan Chen *, Haiyan Chen

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

ARTICLE INFO

Article history:
Received 26 December 2013
Received in revised form 28 February 2014
Accepted 2 April 2014
Available online 19 April 2014

Keywords:
Distance metric learning
Support vector regression
Ensemble learning
Bagging
Distance-based kernel

ABSTRACT

Metric plays an important role in machine learning and pattern recognition. Though many available off-the-shelf metrics can be selected to achieve some learning tasks at hand such as for k-nearest neighbor classification and k-means clustering, such a selection is not necessarily always appropriate due to its independence on data itself. It has been proved that a task-dependent metric learned from the given data can yield more beneficial learning performance. Inspired by such success, we focus on learning an embedded metric specially for support vector regression and present a corresponding learning algorithm termed as SVRML, which both minimizes the error on the validation dataset and simultaneously enforces the sparsity on the learned metric matrix. Further taking the learned metric (positive semi-definite matrix) as a base learner, we develop a bagging-like effective ensemble metric learning framework in which the resampling mechanism of original bagging is specially modified for SVRML. Experiments on various datasets demonstrate that our method outperforms the single and bagging-based ensemble metric learnings for support vector regression.

1. Introduction

Metric learning plays an important role in many learning tasks including k-nearest neighbor classification, k-means clustering and kernel-based algorithms such as support vector machines [1–5]. In recent years, many studies have demonstrated empirically and theoretically that it is often beneficial for a learning task to learn a metric from the given data, instead of using an off-the-shelf one such as Euclidean distance metric.

Depending on the availability of the given data, these methods roughly fall into two main categories: unsupervised metric learning and supervised metric learning. Each unsupervised metric learning method is essentially to learn a distance metric without supervised information [6,7]. While in supervised metric learning, more information about data such as label information is used to learn the metric and it is better to capture the idiosyncrasies of the data of interest [8,9]. We pay particular attention to the supervised methods in this paper.

Supervised distance metric learning can be further divided into task-independent and task-dependent metric learnings. The task-independent methods usually include two separated learning steps: in the first step, a metric is learned by solving an optimization problem with the supervised information. Then the second step uses the learned metric to solve a subsequent task. The classical Linear Discriminant Analysis (LDA) though as a dimensionality reduction method can also be viewed as a pseudo-metric learning method [10]. The metric learned by LDA can be used in many subsequent tasks such as k-nearest neighbor classification. In addition, MMC by Xing et al. learns a metric by minimizing the distances in equivalence constraints and maximizing the distances in inequivalence constraints. Then the metric learned by MMC is used in different clustering tasks [1].

Though the task-independent methods have used the supervised information when learning the metrics, such a two step method cannot guarantee the learned metric is optimal for the subsequent task. Therefore, a more desirable method is to learn the metric directly via incorporating the specific subsequent task, just as the task-dependent distance metric learning. It is similar to the feature selection problem that embedding methods can usually achieve better performance than filter methods [11]. The task-independent metric learning is corresponding to the filter method and the task-dependent metric learning is corresponding to the embedding method. One of the most representative works is Large Margin Nearest Neighbor (LMNN) [2], in which the learned metric is tailored specially for k-nearest neighbor classification and leads to significant improvement compared to k-nn with task-independent metrics. Several related researches have also been proposed, such as Neighborhood components analysis (NCA) [4], multi-task LMNN [12] and Non-linear LMNN [13], etc.

It should be noted that most of the existing task-dependent metric learning methods are designed for classification tasks especially k-nn. Similar to classification, regression is another important task in machine learning and its performance is also

* Corresponding authors. Tel.: +86 15850685790.
E-mail addresses: zou_pc@163.com (P.-C. Zou), s.chen@nuaa.edu.cn (S. Chen).

http://dx.doi.org/10.1016/j.knosys.2014.04.002
0950-7051/

highly dependent on the chosen metric. However, these methods designed for classification tasks cannot be used directly for regression tasks. Only few of the metric learning methods have been proposed specially for regression tasks so far. A typical one is MLKR [14] which learns a metric specially for kernel regression. Unfortunately, the improvement of regression performance achieved by MLKR is limited that it is still difficult to achieve a comparable performance with some sophisticated methods such as support vector regression [15] on many datasets.

To explore further the metric learning method for regression tasks, we consider learning a metric via incorporation of support vector regression (SVR) which is one of the most popular regression algorithms. Metric is also important for SVR especially with kernels. Typical kernels for SVR have no prior knowledge about the meaning of the features and are assumed to be isotropic. Therefore, we focus on learning an embedded metric in SVR to improve the regression performance. We propose a corresponding learning algorithm termed as SVRML, which minimizes the error on the validation set and enforces the sparsity on the learned metric matrix simultaneously. The learning process combines the Mahalanobis [16] metric learning with the training of SVR. More importantly, to make the metric learned by SVRML more effective, we propose a bagging-like ensemble metric learning framework. It extends the original bagging algorithm [17] in which a positive semi-definite matrix is taken as a base-learner rather than either classifier or regressor.

The proposed SVRML algorithm has the following desirable properties: (1) SVRML learns a sparse Mahalanobis metric which is capable of removing potential redundancy or noise in data. (2) SVRML can parallelly learn multiple base metrics by using a bagging-like ensemble metric learning framework and obtain an aggregated metric to achieve better generalization performance for SVR. (3) It is easy to implement and can be treated as an alternative feature selection method to provide a convenient way to pre-process the data automatically. The primary contributions of this work are therefore as follows: (1) We propose a task-dependent metric learning algorithm for SVR. (2) We develop an effective bagging-like ensemble metric learning framework in which the resampling mechanism of original bagging is specially modified for SVRML.

The rest of this paper is organized as follows: we provide an overview of the related work in Section 2. Section 3 explains how to learn an embedded metric for SVR. The bagging-like ensemble metric learning framework is discussed detailedly in Section 4. Experimental studies are shown in Section 5. Finally, we draw the conclusions and list our future works in Section 6.

2. Related works

Over the last decade, several task-dependent metric learning algorithms have been proposed [2,4,18,14]. However, only few of them are designed specially for regression tasks. Support vector regression which is very popular for regression tasks also depend heavily on the metric. As far as we know, our work is the first to combine metric learning with support vector regression. Our proposed method SVRML is also in the family of task-dependent distance metric learning.

Weinberger and Tesauro constructed a metric learning algorithm for kernel regression termed as MLKR [14] which learns a task-specific (pseudo-)metric over the input space where small distances between two vectors imply similar target values. This metric in MLKR is learned by directly minimizing the leave-one-out regression error. Similarly, Xu et al. [19] proposed a metric learning algorithm for support vector classification by minimizing the 0–1 classification error. Inspired by these work, we consider learning a metric for SVR by minimizing the regression error on a validation set. But one drawback of them is that they incline to overfit the validation data [8].

As a remedy, ensemble learning is an alternate method we can use to combine with the metric learning process, as ensemble learning is able to improve the generalization performance of learning systems [20]. Some ensemble learning methods such as boosting [21] have already been introduced into metric learning. For example, Shen et al. [22] proposed a boosting-based technique BoostMetric to learn a metric using trace-one rank-one matrices as weak learners. Chang [23] developed a metric base-learner specific to the boosting framework by improving a loss function iteratively. Mu et al. [24] proposed a local discriminative metrics ensemble learning algorithm. But none of them focus on regression tasks. To fill the gap, we propose a bagging-like ensemble framework designed specially for SVRML to improve the regression performance. Different from the existing methods such as BoostMetric which iteratively learns the base metrics, our framework retains the parallelism like bagging. In our framework, the resampling mechanism of original bagging is specially modified for SVRML to achieve better performance.

In addition to the above, our work is also inspired by the kernel-parameter selection methods for SVR. For example, Chang and Lin [25] derived various leave-one-out bounds for SVR parameter selection to improve the generalization performance. The kernel-parameter selection for SVR can be analyzed on the metric learning perspective that the adjusting of the inner product leads to different distance metrics. Different from choosing a single or a few kernel-parameters, our method optimizes the entire metric matrix and learns a nonlinear metric.

3. Metric learning for support vector regression (SVRML)

3.1. Support vector regression

Our method is based on L2-SVR [15], one of the most commonly used varieties of SVR. Given a set of training examples $\{x_i, y_i\}_{i=1}^{\ell}$ of size ℓ, where the input vector $x_i \in \mathbb{R}^d$, and the target value $y_i \in \mathbb{R}$, L2-SVR solves the primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\xi^*} \quad & \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + \tfrac{C}{2}\sum_{i=1}^{\ell}\xi_i^2 + \tfrac{C}{2}\sum_{i=1}^{\ell}(\xi_i^*)^2 \\ \text{s.t.} \quad & -\varepsilon - \xi_i^* \leqslant \mathbf{w}^T\phi(\mathbf{x}_i) + b - y_i \leqslant \varepsilon + \xi_i, i = 1,\ldots,\ell. \end{aligned} \tag{1}$$

In order to solve the above problem effectively, practically we solve the dual problem of (1) instead:

$$\begin{aligned} \min_{\alpha,\alpha*} \quad & \tfrac{1}{2}(\alpha^* - \alpha)^T\tilde{K}(\alpha^* - \alpha) + \varepsilon\sum_{i=1}^{l}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{l} y_i(\alpha_i^* - \alpha_i) \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0, i = 1,\ldots,\ell, \\ & \alpha_i, \alpha_i^* \geqslant 0, i = 1,\ldots,\ell, \end{aligned} \tag{2}$$

where $k(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the kernel function. $\tilde{K} = K + I/C$ and $I^{d\times d}$ is an identity matrix. The final prediction function is

$$g(x) = \mathbf{w}^T\phi(x) + b = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)k(x_i, x) + b. \tag{3}$$

As the convenience of narrative, we do not distinguish L2-SVR from SVR in the following sections any longer. Many kernel functions are used for SVR. In fact, any function $k(\cdot,\cdot)$ can be used as a well-defined kernel if only it is positive semi-definite. In this paper, we use the popular kernel function RBF kernel uniformly due to its popularity and particularity that it depends on the distance function directly. The RBF kernel is defined as follows:

$$k(x_i, x_j) = \exp\left\{-d^2(x_i, x_j)\right\}, \tag{4}$$

where $d(\cdot,\cdot)$ is the distance metric of data. In the RBF kernel, it is commonly the squared Euclidean distance with a kernel width parameter $\sigma(\sigma > 0)$. When training the SVR, the prediction performance can be improved by choosing an effective parameter σ.

In order to incorporate the Mahalanobis distance into RBF kernel, we transform the original RBF kernel in the following form:

$$k_A(x_i,x_j)=\exp\left\{-d_A^2(x_i,x_j)\right\}=\exp\left\{-(x_i-x_j)^TA(x_i-x_j)\right\} \tag{5}$$

The matrix A is the target metric matrix we would like to optimize. When we set the matrix as $A=\frac{1}{\sigma^2}I^{d\times d}$, the kernel function will resume to the original form of RBF kernel.

3.2. Cost function

In order to learn an effective metric parameterized by A, our cost function is designed by single validation estimate. This estimate is unbiased and its variance gets smaller as the size of the validation set increases [26]. Supposing the validation set is $\{x_i',y_i'\}_{i=1}^p$, our cost function is given by:

$$\begin{aligned}&\varepsilon(A)=\frac{1}{p}\sum_{i=1}^{p}\left(g(x_i')-y_i'\right)^2+\lambda\|A\|_{(2,1)},\\ &\text{where}\quad g(x_i')=\sum_{j=1}^{\ell}(\alpha_j^*-\alpha_j)\exp\left\{-(x_i'-x_j)^TA(x_i'-x_j)\right\}+b\\ &\qquad\qquad=K_A(x_i',X)(\boldsymbol{\alpha}^*-\boldsymbol{\alpha})^T+b.\end{aligned} \tag{6}$$

There are two competing terms in the cost function. The first term penalizes the mean square error between each target value and its estimated value. The prediction function $g(x)$ is obtained by solving the dual problem of SVR in (2) on the training set. With the updating of A, $g(x)$ will also be re-trained. The second term is a mixed (2,1)-norm regularization [27] over A:

$$\|A\|_{(2,1)}=\sum_{i=1}^{d}\|A_i\|_2 \tag{7}$$

where A_i denote the ith row vector of A. Due to $A=L^TL$, by enforcing the L1-norm regularization across the vector $(\|A_1\|_2,\|A_2\|_2,\ldots,\|A_d\|_2)$, it will enforce several columns of L to be zeros and eliminate the impact of the some features to yield feature sparsity.

By incorporating the regularization term into our cost function, it learns a sparse metric, which reduces the risk of overfitting caused by the noise in data and improves the prediction performance of SVR. We would like to find an effective distance matrix A by both minimizing the error on validation set and enforcing the sparsity on the metric matrix simultaneously. The proposed sparse metric learning formulation is:

$$\begin{aligned}&\min_A\quad \frac{1}{p}\sum_{i=1}^{p}\left(g(x_i')-y_i'\right)^2+\lambda\|A\|_{(2,1)}\\ &\text{s.t.}\quad A\succeq 0.\end{aligned} \tag{8}$$

3.3. Optimization algorithm

In order to compute the distance matrix, we use the common gradient-based optimization method in this paper. Considering the continuity and differentiability of the cost function, we make an assumption about the kernel function first:

Assumption 1. The kernel function is differentiable respect to distance matrix A.

However, the mixed (2,1)-norm regularization over A is non-convex and non-differentiable. Actually, the minimum of the mixed (2,1)-norm regularization is equivalent to the trace of A, which is presented as Theorem 1. The theorem is firstly proved by Huang and Sun [28].

Theorem 1. (Huang and Sun [28]) $\min\|A\|_{(2,1)}=tr(A)$.

According to the theorem, the optimization problem (8) can be reformulated as

$$\begin{aligned}&\min_A\quad \frac{1}{p}\sum_{i=1}^{p}\left(g(x_i')-y_i'\right)^2+\lambda tr(A)\\ &\text{s.t.}\quad A\succeq 0.\end{aligned} \tag{9}$$

Then, we can obtain the gradient of $\varepsilon(A)$ by computing the derivative with respect to A of each term in $\varepsilon(A)$ respectively. Making use of $\frac{\partial tr(A)}{\partial A}=I$, the derivative of the second term in (9) is

$$\frac{\partial\lambda tr(A)}{\partial A}=\lambda I. \tag{10}$$

For the simplicity of notation, we use $\mathcal{L}$ to denote the first term of (9). To obtain the derivative of with respect to A, we complete the chain-rule as:

$$\begin{aligned}&\tfrac{\partial\mathcal{L}}{\partial A}=\tfrac{2}{p}\sum_{i=1}^{p}\tfrac{\partial\mathcal{L}}{\partial g(x_i')}\tfrac{\partial g(x_i')}{\partial A},\\ &\tfrac{\partial\mathcal{L}}{\partial g(x_i')}=g(x_i')-y_i'.\end{aligned} \tag{11}$$

The SVR prediction function g, defined in (3), can be further rewritten as:

$$g(x_i')=\bar{K}_A(x_i',X)(\hat{\boldsymbol{\alpha}},b), \tag{12}$$

where $\bar{K}_A(x_i',X)=(K_A(x_i',X),1)$ and $\hat{\boldsymbol{\alpha}}=\boldsymbol{\alpha}^*-\boldsymbol{\alpha}$. So, $g(x)$ depends on indirectly through $(\hat{\boldsymbol{\alpha}},b)$ and K_A. Applying the chain-rule to the derivative of $g(x)$ results in

$$\frac{\partial g(x_i')}{\partial A}=\frac{\partial g(x_i')}{\partial(\hat{\boldsymbol{\alpha}},b)}\frac{\partial(\hat{\boldsymbol{\alpha}},b)}{\partial A}+\frac{\partial g(x_i')}{\partial K_A(x_i',X)}\frac{\partial K_A(x_i',X)}{\partial A} \tag{13}$$

The derivatives $\frac{\partial g(x_i')}{\partial(\hat{\boldsymbol{\alpha}},b)}$, $\frac{\partial g(x_i')}{\partial K_A(x_i',X)}$, $\frac{\partial K_A(x_i',X)}{\partial A}$, are straight-forward and follow from definition (13) [29].

In order to compute the derivative $\frac{\partial(\hat{\boldsymbol{\alpha}},b)}{\partial A}$, we need to recall the Karash–Kunh–Tucker (KKT) [25] optimality conditions of the dual problem of L2-SVR in (2): A vector $\boldsymbol{\alpha}$ is optimal for (2) if and only if it satisfied constraints of (2) and there is a scalar b such that

$$\begin{aligned}&(\tilde{K}(\boldsymbol{\alpha}^*-\boldsymbol{\alpha}))_i+b=y_i+\varepsilon, && \text{if } \alpha_i>0,\\ &(\tilde{K}(\boldsymbol{\alpha}^*-\boldsymbol{\alpha}))_i+b=y_i-\varepsilon, && \text{if } \alpha_i^*>0,\\ &y_i-\varepsilon\leqslant(\tilde{K}(\boldsymbol{\alpha}^*-\boldsymbol{\alpha}))_i+b\leqslant y_i+\varepsilon, && \text{if } \alpha_i=\alpha_i^*=0,\end{aligned} \tag{14}$$

where$(\tilde{K}(\boldsymbol{\alpha}^*-\boldsymbol{\alpha}))_i$ is the ith element of $(\tilde{K}(\boldsymbol{\alpha}^*-\boldsymbol{\alpha}))$. From (14), $\alpha_i\alpha_i^*=0$ when the KKT condition holds. So, for all support vectors, KKT optimality conditions (14) and (2) imply that

$$\underbrace{\begin{pmatrix}\tilde{K}_{SV} & \mathbf{e}_{SV}^T\\ \mathbf{e}_{SV} & 0\end{pmatrix}}_{M}\begin{pmatrix}\hat{\boldsymbol{\alpha}}_{SV}\\ b\end{pmatrix}=\begin{pmatrix}\mathbf{p}\\ 0\end{pmatrix},\text{where } p_i=\begin{cases}y_i-\varepsilon \text{ if } \hat{\alpha}_i>0,\\ y_i+\varepsilon \text{ if } \hat{\alpha}_i<0,\end{cases} \tag{15}$$

and $\mathbf{e}_{SV}=(1,1,\ldots,1)^T$. $\hat{\boldsymbol{\alpha}}_{SV}=\boldsymbol{\alpha}_{SV}^*-\boldsymbol{\alpha}_{SV}$ is the support vectors' part of $\hat{\boldsymbol{\alpha}}$. Since the parameters $\alpha_i^{(*)}$ of non-support vectors are zero, the derivative of these $\alpha_i^{(*)}$ with respect to A are also zero, we can only consider the derivative $\frac{\partial(\hat{\boldsymbol{\alpha}}_{SV},b)}{\partial A}$ instead of $\frac{\partial(\hat{\boldsymbol{\alpha}},b)}{\partial A}$. According to (15), we obtain

$$\begin{pmatrix}\hat{\boldsymbol{\alpha}}_{SV}\\ b\end{pmatrix}=M^{-1}\begin{pmatrix}\mathbf{p}\\ 0\end{pmatrix}. \tag{16}$$

It follows that

$$\begin{aligned}\tfrac{\partial(\hat{\boldsymbol{\alpha}}_{SV},b)}{\partial A}&=-M^{-1}\tfrac{\partial M}{\partial A}M^{-1}\begin{pmatrix}\mathbf{p}\\ 0\end{pmatrix}=-M^{-1}\tfrac{\partial M}{\partial A}\begin{pmatrix}\hat{\boldsymbol{\alpha}}_{SV}\\ b\end{pmatrix}\\ &=-M^{-1}\begin{pmatrix}\frac{\partial\tilde{K}_{SV}}{\partial A}\hat{\boldsymbol{\alpha}}_{SV}\\ \mathbf{0}\end{pmatrix}=-M^{-1}\begin{pmatrix}\frac{\partial\tilde{K}_{SV}}{\partial A} & \mathbf{0}_{SV}\\ \mathbf{0}_{SV}^T & 0\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{\alpha}}_{SV}^T\\ b\end{pmatrix}\\ \tfrac{\partial(\tilde{K}_{SV})_{ij}}{\partial A}&=\tfrac{\partial(K_{SV}+I/C)_{ij}}{\partial A}=\tfrac{\partial(K_{SV})_{ij}}{\partial A}\\ &=-\exp\left\{-(x_i-x_j)^TA(x_i-x_j)\right\}\left((x_i-x_j)(x_i-x_j)^T\right).\end{aligned} \tag{17}$$

Completing the gradient computation $\frac{\partial \varepsilon(A)}{\partial A}$, we implement an iterative sub-gradient projection method to optimize (9) in terms of the positive semi-definite matrix A. We refer to the Mahalanobis distance matrix at the tth iteration as A_t. At each iteration, the metric matrix is updated by

$$A_t = A_{t-1} - \eta \frac{\partial \varepsilon}{\partial A} \tag{18}$$

where η is a small positive step-size constant, denoting the learning rate. The optimization of (9) must enforce the constraint that the matrix A_t remains positive semi-definite. To enforce this constraint, our optimization takes a step along the sub-gradient to reduce the cost function and then projects A_t onto the feasible set, the cone of all positive semi-definite matrixes S_+. Let $A_t = P\Lambda P^T$ denote the eigendecomposition of A_t, where P is the orthonormal matrix of eigenvectors and Λ is the diagonal matrix of corresponding eigenvalues. The projection of A_t onto the cone of positive semi-definite matrixes is given by (19). The projection effectively truncates any negative eigenvalues from the gradient step, setting them equal to zero. The sub-gradient methods using the alternating projection technique provably converge [30] to a correct solution, provided that the gradient step-size is sufficiently small [31].

$$P_S(A_t) = P\Lambda^+ P^T \tag{19}$$

According to the above details, the proposed algorithm is illustrated in Algorithm 1. A validation set independent from the training set is used to monitor the performance. We control the steps of sub-gradient descent by early-stopping with maximum iterations and stop the sub-gradient descent when the cost function converges to the minimum. The local minima is a concern due to the non-convex of the optimization problem (9). To overcome this shortage, we use several runs with different learning rates η, and choose the outcome with minimum cost function $\varepsilon(A)$. Additionally, our bagging-like ensemble metric learning framework discussed in the next section is also conducive to make up this shortage.

Algorithm 1. Metric learning for support vector regression (SVRML)

Input: Training examples, *maxiter* (maximum iterations), convergence threshold δ.

1: Initialize distance matrix $A_0 \leftarrow I^{d\times d}$, cost $\varepsilon_{old} \leftarrow Inf$, integer $t \leftarrow 0$;
2: solve the dual problem of L2-SVR in (2) with A_0 to obtain $\alpha^{(0)}$ and $b^{(0)}$;
3: compute the cost function ε_t;
4: **while** $|\varepsilon_{old} - \varepsilon_t| > \delta$ and $t \leqslant maxiter$ **do**
5: $t \leftarrow t+1, \varepsilon_{old} \leftarrow \varepsilon_{t-1}$;
6: compute the gradient of cost function $\frac{\partial \varepsilon}{\partial A_{t-1}}$ at the tth iteration;
7: $A_t \leftarrow P_S\left(A_{t-1} - \eta \frac{\partial \varepsilon}{\partial A_{t-1}}\right)$;
8: solve the dual problem of L2-SVR in (2) with A_t to obtain $\alpha^{(t)}$ and $b^{(t)}$;
9: compute the cost function ε_t at the tth iteration;
10: **end while**

Output: A_t.

4. Bagging-like ensemble metric learning

According to the Algorithm 1, the metric matrix can be computed. To make the metric learned by SVRML more effective, we further propose a bagging-like ensemble metric learning framework. The basic idea of bagging is to generate multiple versions of training sets with same size of the original one by bootstrap resampling. Then it trains different base-learners from these training sets and uses these base-learners to get an aggregated learner. For regression tasks, the predicting outcome of the ensemble is computed according to Eq. (20):

$$\hat{f}(x) = \sum_{i=1}^{K} w_i f_i(x), \tag{20}$$

where $f_i(x)$ is the output of sample x by the base-learner f_i, and a weight $w_i (i = 1, 2, \ldots, K)$ is assigned to each base-learner $f_i (i = 1, 2, \ldots, K)$. Then the predicting outcome $\hat{f}(x)$ is combined through weighted averaging.

4.1. Metric ensemble strategy

Similarly, we can easily train K versions of metric matrixes $A_1, A_2, \ldots, A_K$ by multiple sampling from the original train set. Then the distance of any pair of examples can be computed through weighted averaging of the distances computed by all the metrics:

$$\begin{aligned} \hat{d}_A(x_i, x_j) &= \sum_{k=1}^{K} w_k \left((x_i - x_j)^T A_k (x_i - x_j) \right) \\ &= \sum_{k=1}^{K} \left((x_i - x_j)^T w_k A_k (x_i - x_j) \right) \\ &= (x_i - x_j)^T \left(\sum_{k=1}^{K} w_k A_k \right) (x_i - x_j). \end{aligned} \tag{21}$$

Set

$$\hat{A} = \sum_{k=1}^{K} w_k A_k, \tag{22}$$

then,

$$\hat{d}_A(x_i, x_j) = (x_i - x_j)^T \hat{A} (x_i - x_j) = \hat{d_A}(x_i, x_j). \tag{23}$$

By observing the similarity between Eqs. (20) and (22), we may view the learned metric A_k as a base-learner and the matrix $\hat{A}$ computed through weighted averaging as the strong learner we would like to learn. It is obvious that the distance between examples derived by matrix $\hat{A}$ still satisfy the conditions of metric. So, the bagging-like ensemble metric learning extends the original bagging algorithm in which a positive semi-definite matrix is taken as a base-learner rather than classifier or regressor.

It should be noted that there is something subtly different between the weighted averaging of learned metrics (metric ensemble) and the weighted averaging of predicting outcomes of base regressors (regression ensemble). Especially, when the weights are equal, the outcome of regression ensemble is the mean of predicting values. In contrast, metric ensemble does not produce the final predicting outcome directly but an aggregated metric. Metric ensemble implies a principle which is similar to majority voting in classification that the aggregated metric tends to catch the common elements in matrixes recognized by the most base metrics. Then the aggregated metric is used to train the final SVR model and obtain the predicting outcome. Therefore, it is reasonable that metric ensemble can achieve better generalization performance for SVR than regression ensemble.

The bagging-like ensemble metric learning framework is shown in Fig. 1. The resampling mechanism is described as follow. Given an original set of learning examples S, a set S_T with half size[1] of S is generated from S by random sub-sampling as the training set of SVR,

[1] For the simplicity of explanation, we only consider the situation that the total number of set S is an even number, and the other situation is almost the same.

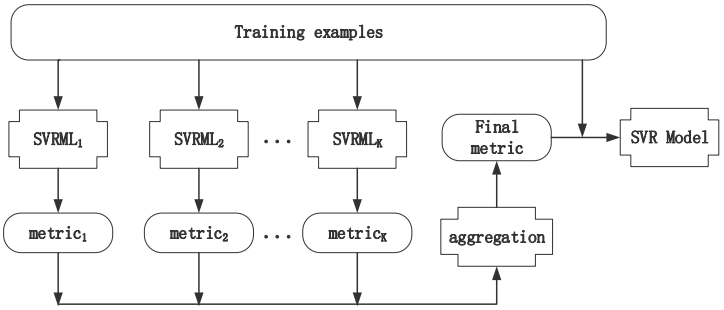


Fig. 1. Bagging-like ensemble metric learning framework.

and a set S_V with the same size of S_T is also generated from S as the validation set of SVRML to learn the distance metric.

The resampling mechanism in bagging cannot be used directly in this work. Only a single learning set is required for each base-learner in bagging. But in our framework, two learning sets both a training set and a validation set are required for each base metric. Therefore, we generate two subsets from the original learning set by resampling respectively. In addition, we use the random sub-sampling resampling method rather than bootstrap used in bagging. The main difference between them is that bootstrap is sampling with replacement and the same instance can appear several times in a validation set which easily leads to the risk of overfitting, whereas sub-sampling is sampling without replacement and can be taken as a special case of bootstrap, and it will cover more instances than bootstrap.

Then, there are two theorems about our resampling mechanism.

Theorem 2. *During the generative process of learning set for each base-learner, the training set is S_T, and the validation set is S_V. If set $S_\cap = S_T \cap S_V$, then the average size of $S_\cap$ is 1/4 of original learning set S with size N.*

Proof. Random sample $N/2$ instances from S to generate S_T, the probability of each sample being selected is $1/2$. Similarly, when random sampling $N/2$ instances from S to generate S_V, the probability of each instance being selected is also $1/2$. The twice sampling process is independent from each other. So, the probability of an instance being selected twice is $1/2 \times 1/2 = 1/4$. Then the probability the size of set $S_\cap$ being i is $P(i) = \left(C^i_{N/2} \cdot C^{(N/2)-i}_{N/2}\right) \Big/ C^{N/2}_N (i = 1, 2, \ldots, N/2)$, and the expectation of the size of $S_\cap$ is:

$$E(|\,S_\cap\,|) = \sum_{i=1}^{N/2} i \cdot P(i) = \sum_{i=1}^{N/2} i \cdot \left(\left(C^i_{N/2} \cdot C^{(N/2)-i}_{N/2}\right) \Big/ C^{N/2}_N\right) = N/4. \quad (24)$$

□

Theorem 3. *During the generative process of learning set for each base-learner, the training set is S_T, and the validation set is S_V. If set $S_\cup = S_T \cup S_V$, then the average size of $S_\cup$ is 3/4 of original learning set S with size N.*

Proof. Set $\bar{S}_\cup = S - S_\cup$. Random sample $N/2$ instances from learning set S to generate S_T, the probability of each sample being selected is $1/2$. Similarly, when random sampling $N/2$ instances from learning set S to generate S_V, the probability of each instance being selected is also $1/2$. The twice sampling process is independent from each other. So, the probability an instance not being selected each time is $(1 - 1/2) \times (1 - 1/2) = 1/4$. Then the probability that the size of set $\bar{S}_\cup$ being i is $P'(i) = \left(C^{(N/2)-i}_{N/2} \cdot C^i_{N/2}\right) \Big/ C^{N/2}_N (i = 1, 2, \ldots, N/2)$, and the expectation of the size of $\bar{S}_\cup$ is:

$$E(|\,\bar{S}_\cup\,|) = \sum_{i=1}^{N/2} i \cdot P'(i) = \sum_{i=1}^{N/2} i \cdot \left(\left(C^{(N/2)-i}_{N/2} \cdot C^i_{N/2}\right) / C^{N/2}_N\right) = N/4. \quad (25)$$

The expectation of the size of $S_\cup$ is $E(|\,S_\cup\,|) = |\,S\,| - E(|\,\bar{S}_\cup\,|) = (3/4)N$. □

According to Theorem 2, for each base-learner, the average of the same number in S_T and S_V is $1/4$ of S. Half of the instances in S_V is different from those in S_T, which guarantees the diversity between S_V and S_T. So, S_V can be used as the validation set of SVRML to learn the distance metric. By applying SVRML algorithm on the kth S_T and S_V, we obtain the metric A_k.

According to Theorem 3, for each base-learner, the average number of instances in $S_\cup$ is about $3/4$ of the original set S. It is larger than the one generated by bootstrap, which is about 63% of the original set [17].

The base metric learned remains different bias, as it trends to concern more about the errors on different validation sets. Though a single learning set cannot cover the full set S, with the increase of the number of base-learners, they cover more number of instances in S. We have a deduction according to Theorem 3 that when the number of base-learners is K, the coverage rate of the learning sets is $\alpha = 1 - (1 - 3/4)^K$. For example, if $K = 3$, then $\alpha \approx 98.4\%$, and if $K = 5$, then $\alpha \approx 99.9\%$.

4.2. Metric ensemble algorithm

We aggregate all learned metrics by Eq. (22). To simplify this process, we give all learned matrixes equal weights by setting $w_i = 1/K$, and obtain the final metric matrix $\hat{A}$. Then, we train SVR on the original learning set S with the learned metric. The proposed bagging-like ensemble metric learning algorithm for support vector regression is sketched in Algorithm 2.

Algorithm 2. Bagging-like metric learning for support vector regression (Bag-SVRML)

Input: Training set S, weak metric learner M(SVRML), and integer K (the number of generated metrics).
1: **for** $i = 1$ to K **do**
2: $S_T^{(i)}$ = random sample from training set S, with $S_T^{(i)} = N/2$;
3: $S_V^{(i)}$ = random sample from training set S, with $S_V^{(i)} = N/2$;
4: $A_i \leftarrow M\left(S_T^{(i)}, S_V^{(i)}\right)$;
5: **end for**
Output: $\hat{A} = \sum_{i=1}^K w_i A_i$.

Our method is very convenient to generate multiple versions of base-learners parallelly. The learning process of each metric has no communication with others. It is helpful to improve the efficiency of our method. In addition, though the size of learning set for base-learners is fixed above, we can adjust the size of learning set flexibly according to the scale of tasks. On the premise of adequate sampling, the smaller the learning set is, the more the base metrics are needed.

Table 1
Benchmark datasets used in the experiments.

Dataset	#Examples	#Features	#Training examples	#Testing examples
Housing	506	13	455	51
Auto mpg	392	7	353	39
Triazines	186	60	167	19
Abalone	4177	8	3759	418
mg	1385	6	1247	139
Pyrim	74	27	67	7
Space_ga	3107	6	2796	311
Energy efficiency	768	8	691	77
Compressive strength	1030	8	927	103
Slump test	103	7	93	10
Cpusmall	8192	12	4096	4096
Bodyfat	252	15	227	25
Eunite2001	336	16	302	34
Yacht hydrodynamics	308	6	277	31
Winequality	4898	11	2449	2449

4.3. Time complexity analysis

The time complexity analysis of our algorithm is given in this subsection. Given the data $X \in \mathbb{R}^{d\times n}$, where n is the number of samples and d is the number of features, the training time of the original SVR is $O(n^3)$[32,33]. In each iteration of updating the metric of SVRML, the time for calculating the gradient of metric matrix is $O(1/2n^2d^2)$, which is in the same complexity order with the training of MLKR [14]. Then, the total training time for each base metric become $O(\ell(n^3 + 1/2n^2d^2))$, where ℓ is the number of iterations and usually small.

Although intuitively ought to have much longer training time than the original SVR because of large number of calls to the SVR solver, our algorithm has actually contained an additional yet convenient and automatical pre-processing process for data. Moreover, the implementation of our algorithm can be much faster. On the one hand, we use a simple warm-start technique that the previous SVM solution is often a good guess for the current problem. The warm-start call to the SVM solver results in much less computation than a call from scratch [34]. On the other hand, the computational intensive parts of the gradient outside of the SVR calls are trivially parallelizable and could be computed on multiple cores. As it is not the focal point of this paper, we do not focus on further scalability here.

5. Experiments

In this paper, we propose a bagging-like metric learning for support vector regression, which learns a task-dependent distance metric. To verify the effectiveness of the proposed method, we

Table 2
The prediction error on each dataset.

Dataset	RMSE				MAPE (%)			
	SVR-RBF (5-fold)	SVR-RBF (10-fold)	MLKR	Bag-SVRML	SVR-RBF (5-fold)	SVR-RBF (10-fold)	MLKR	Bag-SVRML
Housing	3.267 (0.6250)	3.080 (0.6024)	4.126 (0.7731)	**2.423** (0.2338)	10.01 (1.676)	9.668 (1.823)	15.97 (1.547)	**8.964** (1.329)
Auto mpg	2.599 (0.5796)	2.564 (0.5186)	3.799 (0.4272)	**2.468** (0.5379)	8.122 (1.192)	8.265 (1.095)	12.08 (1.012)	**7.989** (1.156)
Triazines	0.1016 (0.0113)	0.1035 (0.0115)	0.1144 (0.0103)	**0.0989** (0.0097)	11.61 (1.473)	12.39 (1.766)	14.45 (2.371)	**11.45** (1.382)
Abalone	2.095 (0.0576)	2.090 (0.0627)	2.1703 (0.0766)	**2.051** (0.0554)	15.03 (0.479)	14.99 (0.472)	19.95 (1.104)	**14.39** (0.532)
mg	0.1284 (0.0052)	0.1274 (0.0054)	0.1336 (0.0074)	**0.1221** (0.0062)	12.45 (1.187)	12.43 (1.264)	11.47 (1.532)	**11.58** (1.187)
Pyrim	0.0600 (0.0133)	0.0583 (0.0150)	0.0839 (0.0132)	**0.0502** (0.0115)	7.205 (1.650)	6.951 (1.987)	10.90 (2.039)	**6.048** (1.555)
Space_ga	0.1145 (0.0123)	0.1158 (0.0131)	0.1580 (0.0237)	**0.1102** (0.0114)	17.33 (1.383)	16.97 (1.200)	30.20 (4.426)	**16.34** (1.290)
Energy efficiency	2.056 (0.1518)	1.624 (0.200)	2.3326 (0.3345)	**0.9696** (0.1788)	6.576 (0.395)	4.605 (0.533)	9.518 (1.233)	**2.710** (0.3740)
Compressive strength	6.048 (0.4516)	6.056 (0.5801)	8.6359 (0.7323)	**4.762** (0.4764)	16.52 (5.796)	16.68 (2.267)	21.21 (4.083)	**12.04** (2.315)
Slump test	0.8234 (0.2339)	0.6840 (0.2152)	3.0122 (0.3144)	**0.4782** (0.1256)	1.946 (0.5909)	1.627 (0.6255)	6.812 (1.148)	**1.087** (0.3461)
Cpusmall	3.398 (0.0523)	3.314 (0.0495)	4.494 (0.0530)	**3.027** (0.0439)	8.923 (1.723)	6.782 (1.655)	5.382 (0.2922)	**5.246** (0.3041)
Bodyfat	0.016 (0.0024)	**0.015** (0.0015)	0.019 (0.0053)	0.016 (0.0019)	0.1113 (0.0331)	0.1083 (0.0352)	1.538 (0.0292)	**0.1063** (0.0307)
Eunite2001	18.64 (1.939)	18.47 (1.945)	18.49 (2.515)	**17.95** (1.701)	1.977 (0.0344)	1.850 (0.0421)	2.159 (0.1826)	**1.799** (0.1317)
Yacht hydrodynamics	0.7360 (0.4204)	**0.7342** (0.4304)	2.230 (1.352)	0.7373 (0.499)	44.79 (4.049)	41.30 (3.001)	**21.06** (4.557)	37.61 (4.594)
Winequality	0.7393 (0.0097)	0.7356 (0.0094)	0.7376 (0.0060)	**0.7142** (0.0143)	10.11 (0.1083)	10.09 (0.1068)	9.980 (0.1237)	**9.724** (0.1220)

empirically evaluate it on a number of regression datasets. Firstly, we compare our algorithm with other related method on several benchmark datasets. Then, a set of experiments are given to explore the connection between the number of the base metrics and the effectiveness of the learned metric. Meanwhile, we compare the performances of SVRML with different ensemble strategies. In addition, to verify the robustness of the learned metric, we augment benchmark datasets with synthetic noise of varying dimensions to investigate the prediction performance as noisy features are introduced.

5.1. Comparison with representative algorithms

To evaluate the performance of our algorithm, we compare it with the related regression algorithms as follows:

(1) SVR-RBF: SVR with RBF kernel, in which the parameters C and σ^2 are selected by cross validation.

(2) MLKR: metric learning for kernel regression [14], in which a new distance metric is learned on a distance based kernel function.

For SVR-RBF, we search the penalty parameter C from the set $\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$, and the width parameter σ^2 of the RBF kernel from the set $\{10^{-2}, 10^{-1}, \ldots, 10^{4}, 10^{5}\}$ respectively through 5-fold and 10-fold cross validation. For MLKR, we initialize the distance metric with a diagonal matrix $1/\sigma^2 \times I^{d\times d}$, where the parameter σ^2 is searched from the same set as SVR-RBF. For our algorithm Bag-SVRML, the penalty parameter C is searched from the same set as SVR-RBF, and the distance matrix is initialized with the identity matrix $I^{d\times d}$. The regularization parameter λ in our algorithm is used to control the trade-off between the error on the validation and the regularization item. We empirically set λ to $1/d^2$ and uniformly set the number of base metrics to 5. The influence of the number of base metrics to the prediction results will be discussed in the next subsection. We test these regression algorithms

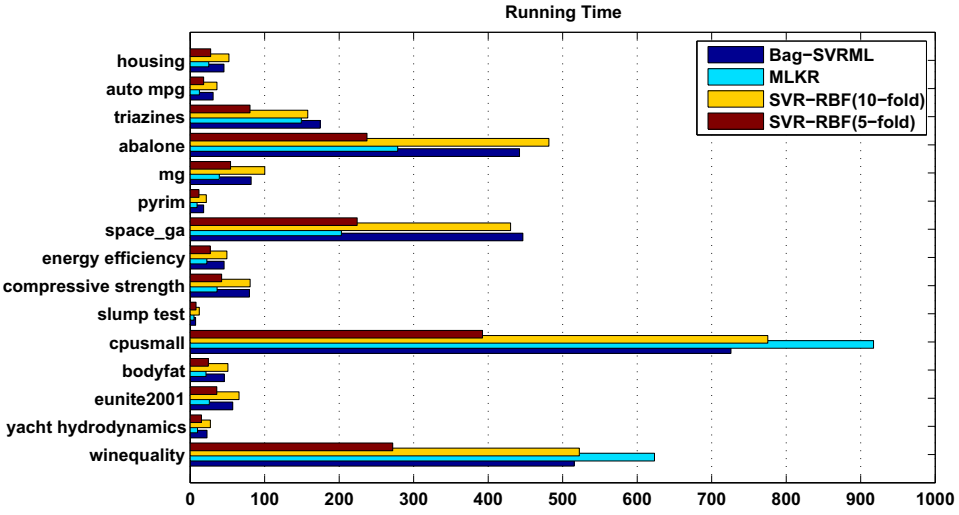


Fig. 2. The running time of each algorithm on each dataset.

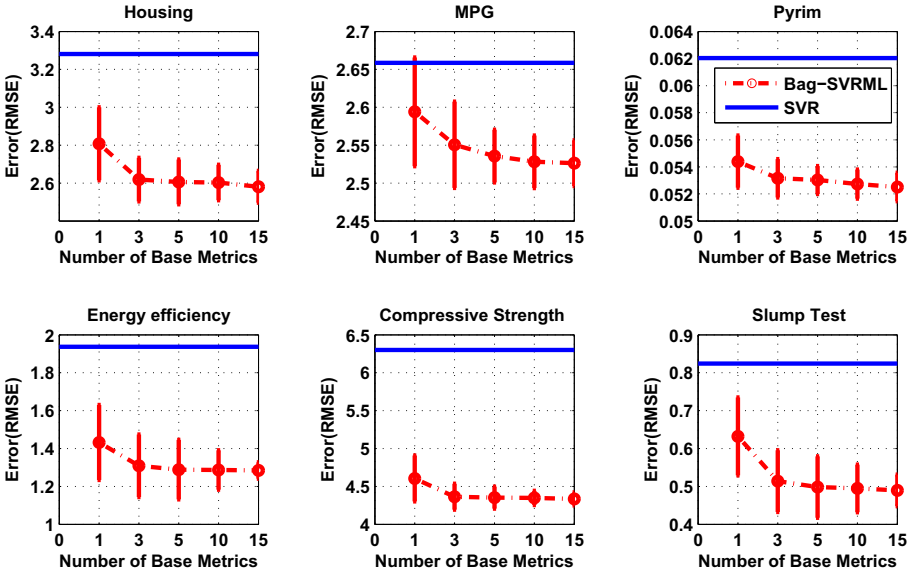


Fig. 3. The prediction errors on each dataset with the increase of the number of base metrics.

on 15 benchmark datasets. The descriptions of these datasets are summarized in Table 1.

We use two widely-used standard measures to evaluate the prediction performance of these algorithms. The first measure is root mean squared error (RMSE), which estimates the absolute error between predictive value and the target value of data. Another measure is mean absolute percentage error (MAPE), which takes into account the percentage of absolute error in target value.

For the fairness of comparison, we repeat experiments for 20 trials on each dataset. All the features of data are normalized. The final predictive values are the average results of the 20 trials. The prediction errors and the standard deviations on 15 datasets are listed in Table 2. The bold ones represent the best performance on each dataset. The predictive results provided by SVR-RBF are under the best parameters C and σ^2 chosen by 5-fold and 10-fold cross validation respectively. It is obvious that the predictive results of MLKR are worse than the SVR based methods here, while Bag-SVRML produces lower prediction error than the compared methods on the most datasets.

The running time of each algorithm is reported in Fig. 2. The time includes training, cross validation and testing. On the one hand, the results show that Bag-SVRML consumes the training resources roughly comparable with SVR-RBF by 5–10-fold cross validation. It exhibits insensitiveness, to great extent, to the involved parameters (thus we all set them to default values) and more interestingly, without the need of conducting model selection by cross validation. In contrast, the SVR-RBF needs to select its parameters by cross validation, leading to a large number of calls to the SVR solver. On the other hand, the training time of MLKR is less than that of the SVR based algorithms on most datasets. But it should be noted that, contrary to those SVR based algorithms, the test process of MLKR is much more time-consuming as

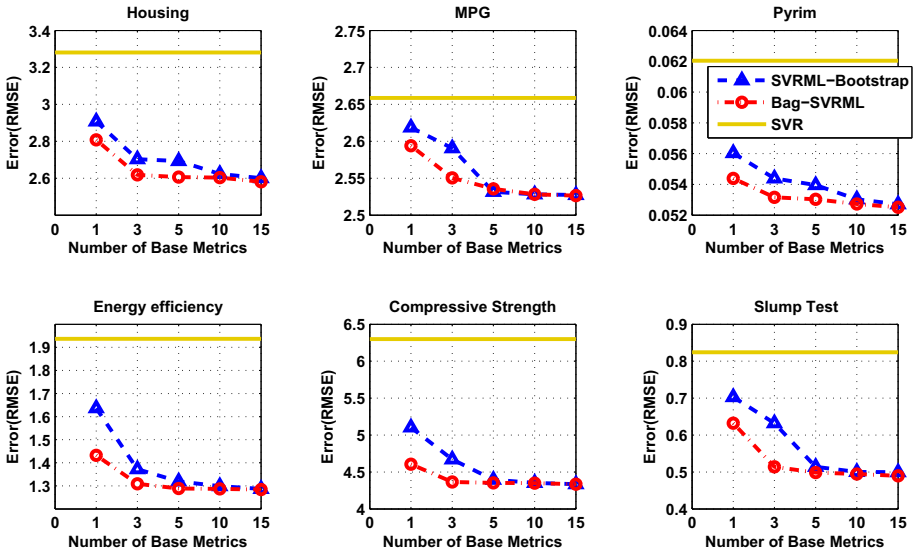


Fig. 4. The prediction errors with different resampling method on each dataset.

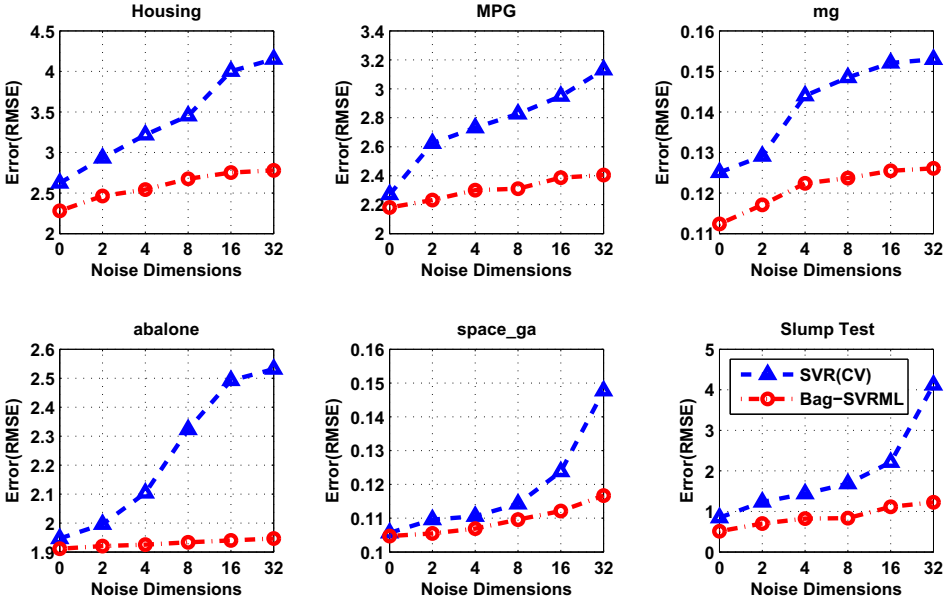


Fig. 5. The average prediction performance on each dataset with different number of noise features.

it shows on dataset cpusmall and winequality. Therefore, considering both the prediction accuracy and time consumption, Bag-SVRML outperforms the compared regression algorithms.

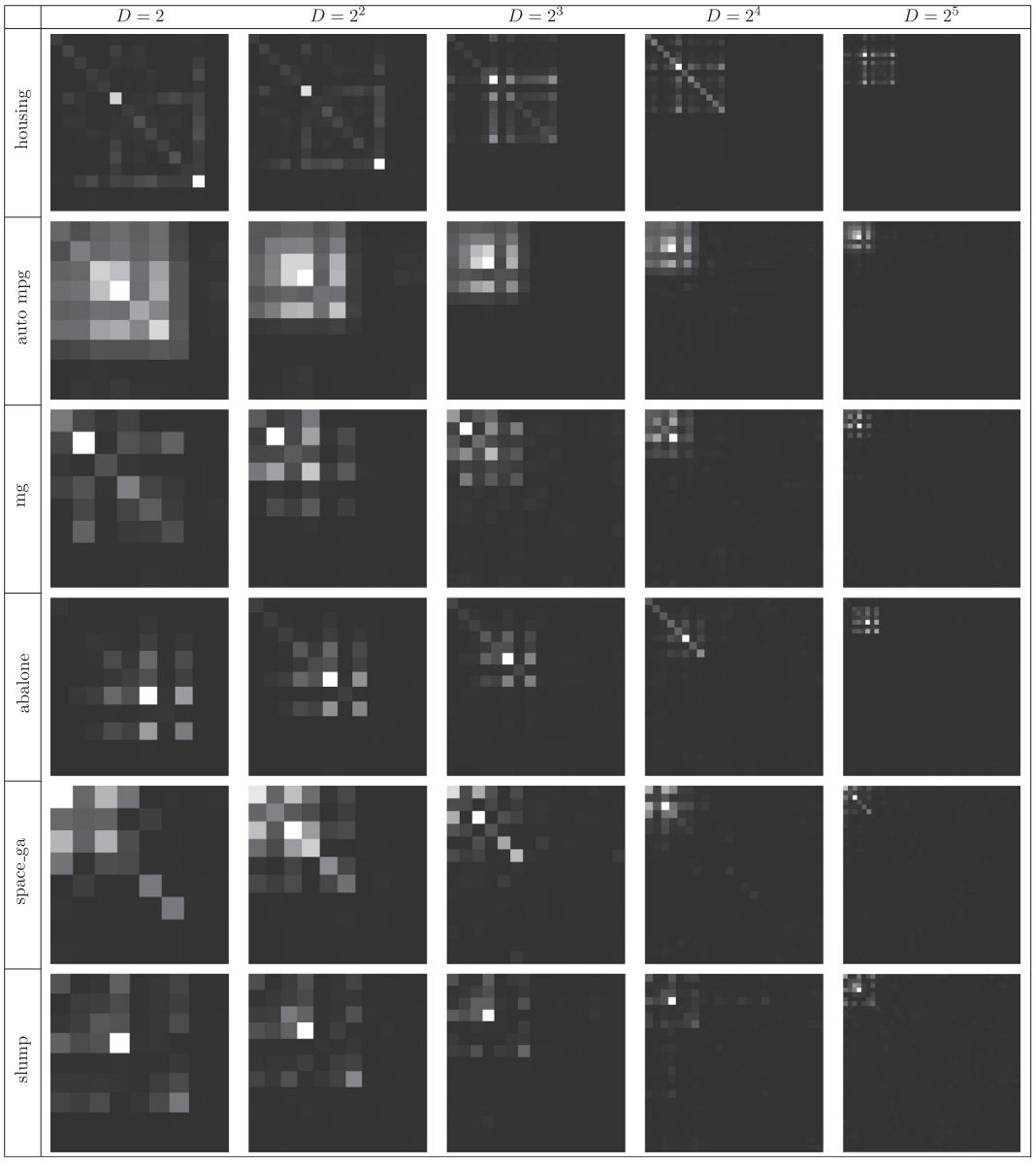
5.2. Influence of the number of base metrics and the ensemble strategies

The number of base learners is very important to the ensemble performance in bagging. Similarly, for Bag-SVRML, the number of base metrics also has influence on the effectiveness of aggregated metric, which directly displays on the prediction results. Meanwhile, the effectiveness of our ensemble strategies designed specially for SVRML need verifying against other existing ensemble strategies. Therefore, we explore the performance of Bag-SVRML with different number of base metrics and also compare the performances of SVRML with different ensemble strategies. The parameters set in the experiment is the same as Section 5.1.

Our experiments are performed on 6 datasets in which relatively significant improvement are achieved. Fig. 3 displays prediction results of SVRML with the metrics learned by different number of base learners. When the number of base learners is zero, it denotes the performance of SVR without learned metric. It is obvious that with the increase of the number of base learners, the performance of SVRML on each dataset is improved. We get most of the improvement when there are about 5 base-learners. Meanwhile, with the increase of the number of base metrics, the variance of the prediction results decreases, leading to more stable prediction results. On the other hand, the distinction between metric ensemble and regression ensemble we discuss in Section 4.1 can also be significantly shown in Fig. 3. If the number of base learners is only one, the mean value is just equal to the result of regression ensemble. While the result of metric ensemble is shown by using multiple base metrics. From Fig. 3, we achieve better prediction performance by metric ensemble than regression ensemble.

To verify the effectiveness of the resampling mechanism in our framework, we compare the prediction performance by using our resampling method with the bootstrap in bagging. The result is

Table 3
The distance matrix learned with synthetic noise of varying dimensions.



shown in Fig. 4. It is obvious that our resampling method can achieve better prediction performance than bagging especially when the number of base metrics is small. With the increase of the number of base metrics, the improvements for SVR achieved by two different resampling method trend to be much closer.

5.3. Influence of noise features to the learned metric

In the real world regression tasks, the observed data is probably contaminated with noise or redundancy, which has deep influence on the performance of prediction results. It is an important evaluation criteria for the learned metric whether it is able to effectively remove the noise in data. In order to verify the robustness of the metric learned by Bag-SVRML, we further perform experiments in this section.

We augment benchmark datasets with synthetic noise of varying dimensions. The number of noise features D is got from the set $\{2^1, 2^2, 2^3, 2^4, 2^5\}$. Each noise feature is obtained by random generation and the value of each noise feature obeys $(10, 100)$ – Gaussian distribution. In order to avoid the influence carried by the scale of each feature for the metric, the normalization procedure is applied to each feature in our experiment. We measure the prediction performance on six benchmark datasets: housing, mpg, mg, abalone, space_ga and slump test. The chosen of the parameters in Bag-SVRML are the same as the previous subsection. SVR-RBF (10-fold) without learned metric is used as a baseline in this experiment.

Fig. 5 illustrates that the performance of original SVR get worse with a high rate when the number of noise features increasing. Meanwhile, Bag-SVRML is more robust to the noise features, and the prediction error remains very low with the increase of the number of noise. Table 3 displays the distance matrix A learned by Bag-SVRML with synthetic noise of varying dimensions on each dataset. Each grayscale image is corresponding to a learned distance matrix A. In each grayscale image, the brighter the block is, the larger the absolute value of the corresponding element in the matrix A will be. Otherwise, the absolute value of the corresponding element is smaller. It is obvious that Bag-SVRML correctly identifies and suppresses most of the noise dimensions by assigning small weights to the corresponding rows and columns of metric A. With the increase of the number of noise feature, effective dimensionality remains low and the correlation between different features remains stable.

6. Conclusion

In this work, we propose a task-dependent metric learning algorithm for support vector regression which both minimizes the error on validation set and enforces the sparsity on the learned metric matrix. Furthermore, we extend the general bagging algorithm and propose a novel bagging-like ensemble metric learning framework to enhance the effectiveness of the learned metric. Experiments on various datasets demonstrate the effectiveness of our method.

Owing to the fact that the computational complexity of learning metrics still is expensive, we will discover a method to solve it so that our method can be applied to the large-scale problems. On the other hand, we will further explore the bagging-like ensemble metric learning framework for more regression tasks.

Acknowledgements

This work is sponsored by the National Natural Science Foundation of China (No. 61139002), the Fundamental Research Funds for the Central Universities (Nos. NS2012134, NZ2013306), the National Science Foundation for Post-doctoral Scientists of Jiangsu (No. 1301013A), and Jiangsu "QingLan" Project Foundation.

References

[1] E.P. Xing, M.I. Jordan, S. Russell, A. Ng, Distance metric learning with application to clustering with side-information, in: Advances in Neural Information Processing Systems, 2002, pp. 505–512.
[2] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[3] J.V. Davis, B. Kulis, P. Jain, S. Sra, I.S. Dhillon, Information-theoretic metric learning, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 209–216.
[4] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, *****Neighbourhood components analysis (2004) 513–520.
[5] N. Shental, T. Hertz, D. Weinshall, M. Pavel, Adjustment learning and relevant component analysis, in: Computer Vision – ECCV 2002, Springer, 2006, pp. 776–790.
[6] L. Yang, R. Jin, Distance Metric Learning: A Comprehensive Survey, Michigan State University, 2006.
[7] X. Chen, J. Zhang, Maximum variance difference based embedding approach for facial feature extraction, Int. J. Pattern Recogn. Artif. Intell. 24 (2010) 1047–1060.
[8] A. Bellet, A. Habrard, M. Sebban, A survey on metric learning for feature vectors and structured data, arXiv:1306.6709, in press.
[9] X. Chen, J. Zhang, D. Li, Direct discriminant locality preserving projection with hammerstein polynomial expansion, IEEE Trans. Image Proces. 21 (2012) 4858–4867.
[10] Y. Zhang, D.-Y. Yeung, Worst-case linear discriminant analysis, in: Advances in Neural Information Processing Systems, 2010, pp. 2568–2576.
[11] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.
[12] S. Parameswaran, K.Q. Weinberger, Large margin multi-task metric learning, in: Advances in Neural Information Processing Systems, 2010, pp. 1867–1875.
[13] D. Kedem, S. Tyree, K. Weinberger, F. Sha, G. Lanckriet, Non-linear metric learning, in: Advances in Neural Information Processing Systems 25, 2012, pp. 2582–2590.
[14] K.Q. Weinberger, G. Tesauro, Metric learning for kernel regression, in: International Conference on Artificial Intelligence and Statistics, 2007, pp. 612–619.
[15] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (2004) 199–222.
[16] P.C. Mahalanobis, On the generalized distance in statistics, Proc. Nat. Inst. Sci. (Calcutta) 2 (1936) 49–55.
[17] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[18] M. Bilenko, S. Basu, R.J. Mooney, Integrating constraints and metric learning in semi-supervised clustering, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, pp. 81–88.
[19] Z. Xu, K.Q. Weinberger, O. Chapelle, Distance metric learning for kernel machines, arXiv:1208.3422, in press.
[20] N.C. Oza, Online bagging and boosting, 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, IEEE, 2005, pp. 2340–2345.
[21] R.E. Schapire, Theoretical views of boosting and applications, in: Algorithmic Learning Theory, Springer, 1999, pp. 13–25.
[22] C. Shen, J. Kim, L. Wang, A. van den Hengel, Positive semidefinite metric learning using boosting-like algorithms, J. Mach. Learn. Res. 13 (2012) 1007–1036.
[23] C.-C. Chang, A boosting approach for supervised mahalanobis distance metric learning, Pattern Recogn. 45 (2012) 844–862.
[24] Y. Mu, W. Ding, D. Tao, Local discriminative distance metrics ensemble learning, Pattern Recogn. 46 (2013) 2337–2349.
[25] M.-W. Chang, C.-J. Lin, Leave-one-out bounds for support vector regression model selection, Neural Comput. 17 (2005) 1188–1222.
[26] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (2002) 131–159.
[27] K. Huang, Y. Ying, C. Campbell, Generalized sparse metric learning with relative comparisons, Knowl. Inform. Syst. 28 (2011) 25–45.
[28] R. Huang, S. Sun, Kernel regression with sparse metric learning, J. Intell. Fuzzy Syst. 24 (2013) 775–787.
[29] K.B. Petersen, M.S. Pedersen, The Matrix Cookbook, 2006.
[30] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM Rev. 38 (1996) 49–95.
[31] S.P. Boyd, L. Vandenberghe, Convex optimization, Cambridge University Press, 2004.
[32] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Mining Knowl. Discov. 2 (1998) 121–167.
[33] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and other Kernel-based Learning Methods, Cambridge University Press, 2000.
[34] A. Rakotomamonjy, F.R. Bach, S. Canu, Y. Grandvalet, Simplemkl., J. Mach. Learn. Res. 9 (2008).