Applied Intelligence (2024) 54:9622–9640
https://doi.org/10.1007/s10489-024-05686-6

Semi-supervised regression via embedding space mapping and pseudo-label smearing

Liyan Liu[1,2] · **Jin Zhang**[1,2] · **Kun Qian**[1,3] · **Fan Min**[1,2,4]

Accepted: 14 July 2024 / Published online: 19 July 2024

Abstract
Co-training is a semi-supervised algorithm that aims to improve prediction effects by exchanging confident instances and pseudo-labels among multiple learners. One central issue is how to mitigate the negative impact of low-quality pseudo-labels during training. In this paper, we propose semi-supervised regression via embedding space mapping and pseudo-label smearing (S2RMS) to ensure that unlabeled data contribute positively to the prediction process. First, a Triplet neural network is trained using pairwise data generated from labeled data. This network maps the training data to the embedding space to better separate dissimilar instances. Second, the embedded data are randomly partitioned into different subsets to train corresponding regression models (a.k.a. regressors). These regressors are integrated into the prediction process. Third, these subsets are augmented using unlabeled data with high similarity to the labeled data and high-confidence pseudo-labels. Here, the similarity and confidence are calculated using the network and the smearing technique, respectively. Experiments are conducted on fourteen datasets, and the results are compared to those of three excellent algorithms. The results show that S2RMS outperforms other co-training and metric semi-supervised regression algorithms. The source code is available at: https://github.com/BetaCatPro/S2RMS.

Keywords Co-training · Deep metric learning · Embedding space · Semi-supervised regression

1 Introduction

Semi-supervised learning can enhance the performance of a learning algorithm by utilizing an extensive amount of unlabeled data in real-world tasks [1, 2]. Several semi-supervised methods have been developed, including co-training [3, 4], self-training [5, 6], consistency regularization [7, 8], and holistic methods [9, 10]. The co-training approach can be described as a disagreement-based technique for semi-supervised learning [11]. During the co-training process, two separate learners are trained on different views, and they select the unlabeled instances about which they are most confident. The unlabeled instances of one learner are then given pseudo-labels and added to the training set of the other learner, which uses them to update its model parameters. Unlike graph and kernel methods [12, 13], co-training does not require additional complex computations or large amounts of memory.

Two main problems are encountered when conducting co-training. First, some studies have indicated that casually using unlabeled data may result in a decrease in model effectiveness [14]. This is because not all unlabeled data can be used for training purposes. It is difficult to assign accurate pseudo-label to some of these unlabeled instances [15, 16]. If unlabeled instances carrying incorrect pseudo-labels continue to be added to the training set, the predictive performance of the utilized learner can be seriously disturbed [17].

✉ Fan Min
minfan@swpu.edu.cn

Liyan Liu
liuliyan@swpu.edu.cn

Jin Zhang
zj20162325@163.com

Kun Qian
kqian2020@163.com

[1] School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, China

[2] Lab of Machine Learning, Southwest Petroleum University, Chengdu 610500, China

[3] School of Sciences, Southwest Petroleum University, Chengdu 610500, China

[4] Institute for Artificial Intelligence, Southwest Petroleum University, Chengdu 610500, China

The situation described above is comparable to the introduction of noisy data to the training set, which is beyond our control. Second, weaker learners struggle to assign accurate pseudo-labels to unlabeled data during the initial stages of training, resulting in a failure to satisfy the expected confidence level [18]. These unsuitable instances are also added to the training set, which leads to the accumulation of errors until the end of the training process.

In this paper, we present semi-supervised regression via embedding space mapping and pseudo-label smearing (S2RMS). This approach combines co-training with a deep metric neural network to handle the above issues. To address the first problem, we use a metric network to learn a decision function that can effectively utilize unlabeled data. To address the second problem, we employ a pseudo-label smearing technique to train robust regressors. Figure 1 shows the framework of S2RMS.

Figure 1 shows that S2MRS selects multiple regressors to build a co-training model. In this study, a particular set of regressors is chosen to test the validity of S2RMS. To better cope with different environments, S2RMS uses random forests [19] possessing varying numbers of evaluators to form the base regressors. Each regressor also has different parameters to enhance the variability between different models.

Under the assumption of smoothing [1], instances located in high-density areas that are close to each other typically have similar labels, while those that are further apart are more likely to be distinct. Based on this hypothesis, a regression model trained with a small amount of labeled data will have a higher prediction accuracy when confronted with unlabeled data that behave similarly to the labeled data.

First, a pairwise dataset is generated for all anchor instances in the labeled dataset, and this dataset comprises anchor-positive and anchor-negative instance pairs. Subsequently, a deep metric Triplet neural network is trained using this pairwise dataset to learn embedding mappings. Similar instances are located close to each other in this embedding space. The Triplet neural network is used to map labeled and unlabeled data into an embedding space. Second, k subsets are created by randomly sampling the transformed labeled data. Each subset is divided into two parts, with one part used for training the regressors and the other used to calculate the confidence level. The regressors are initialized with

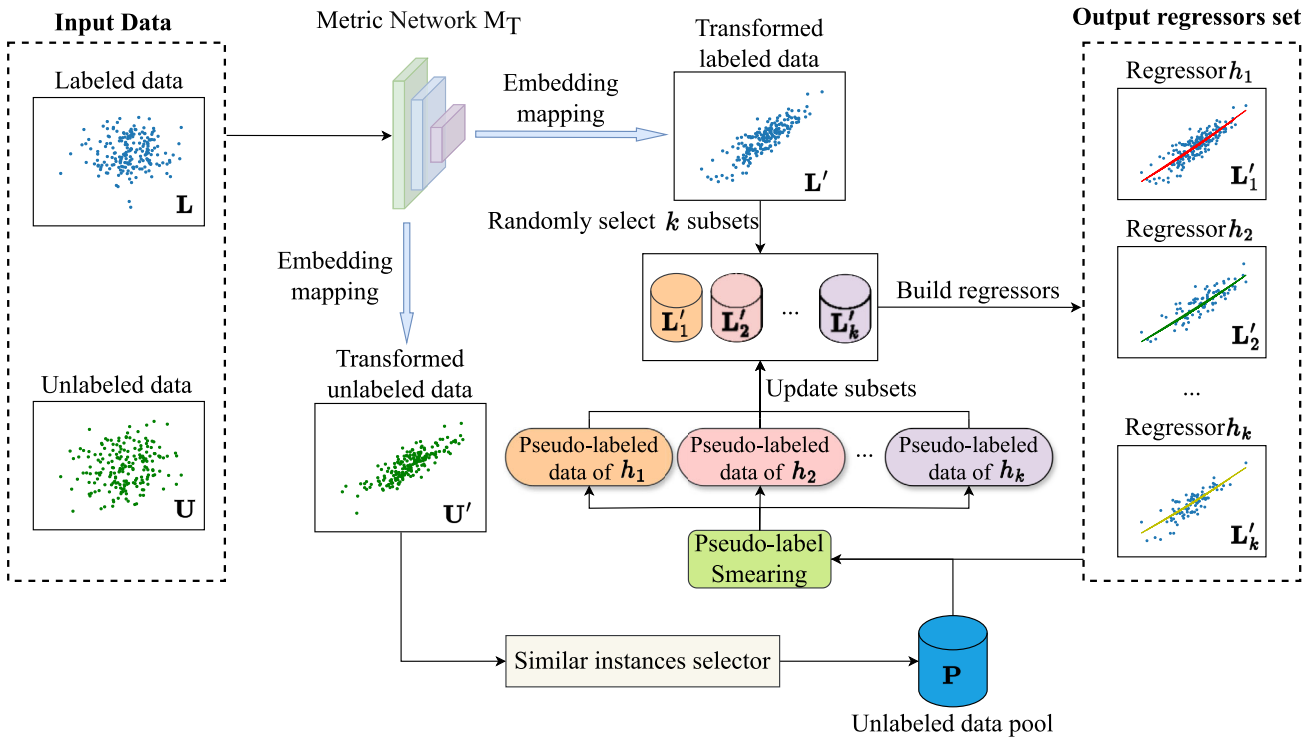

Fig. 1 The framework of S2RMS. The deep metric network M_T used in S2RMS maps the labeled data $\mathbf{L}$ and unlabeled data $\mathbf{U}$ into a d-dimensional embedding space, which facilitates subsequent operations. To establish the co-training framework, k subsets $\{\mathbf{L}'_i\}_{i=1}^{k}$ are randomly and repeatedly selected from the transformed $\mathbf{L}'$ and used to initialize k regressors $\{h_j\}_{j=1}^{k}$. During the co-training iterations, instances are identified from the transformed $\mathbf{U}'$ using a similar instances selector and added to the data pool $\mathbf{P}$. The co-regressor then employs a label smearing technique within $\mathbf{P}$ to ensure representation stability and assigns appropriate pseudo-labels for the unlabeled instances. Instances with pseudo-labels are added to the training set of the corresponding regressor, which is then retrained

different parameters to enhance the diversity and ensure the efficacy of S2RMS. In each iteration, the Triple neural network functions as an instance selector, choosing two similar unlabeled instances for each labeled instance. These similar instances are then added to the pool. The other regressors use pseudo-label smearing to select stable instances from the pool for another iteration. Afterward, these stable instances are assigned pseudo-labels with confidence. Finally, all k training subsets are augmented with these stable instances. All regressors are trained again with the updated data before moving on to the next iteration. After completing all training processes, the unknown data are predicted through averaging.

The effectiveness of the S2RMS algorithm is evaluated through experiments conducted on 14 benchmark datasets. In addition, these datasets are normalized to better visualize the experimental results. We compare the S2RMS algorithm with three excellent semi-supervised regression algorithms. These algorithms also use co-training or metric networks. The results indicate that the S2RMS algorithm outperforms these algorithms.

The remainder of this paper is organized as follows. In Section 2, popular metric learning and semi-supervised learning approaches are reviewed. Section 3 presents the proposed S2RMS algorithm. The spatial mapping method and the smearing technique are described in this section. Section 4 provides the experimental results and analysis. In Section 5, we conclude the paper and outline further work directions.

2 Related works

This section reviews two fundamental techniques that are used in S2RMS: deep metric learning and co-training.

2.1 Deep metric learning

Metric learning aims to measure the similarity between instances using an optimal distance metric for performing the learning task. The main objective is to learn a new metric that reduces the distances between instances belonging to the same class and increases the distances between instances belonging to different classes [20]. Traditional metric learning methods typically use a linear projection strategy, which may not be suitable for real-world problems with nonlinear structures. In deep metric learning, this problem is solved by activation functions with nonlinear structures [21].

The development of deep metric learning is primarily reflected in the available loss functions. The two most basic loss functions in deep metric learning are Contrastive Loss [22] and Triplet Loss [23]. Many variants have been derived from these two functions, such as the Quadruple Loss [24], ArcFace Loss [25], and Ranked List Loss [26]. Furthermore, deep metric learning has been primarily applied in three areas: 1) image retrieval, 2) person re-identification (ReID), and 3) face recognition. In the field of image retrieval, some studies [27, 28] have examined tasks such as visual sentiment analysis and massive image retrieval by designing different similarity losses. Constrained by the limitations of manual annotation, TSSML [29] improves the hard instance mining process through curriculum learning for semi-supervised ReID tasks. In the facial recognition task, DDFMs [30] use a novel deep metric network approach based on the triplet loss for facial recognition/verification and visual image classification purposes.

2.2 Co-training

Co-training is a divergence-based approach for semi-supervised learning that is designed for multi-view data [3]. This method assumes that data can be effectively represented through various perspectives and that distinct learners can be trained on each view. These learners then label unlabeled instances and incorporate their most confident unlabeled instances into the training sets of other learners. Following each iteration, the learners undergo retraining with the augmented training set.

In the field of co-training, research has focused primarily on three areas: 1) classification, 2) regression, and 3) image segmentation and recognition. In classification tasks, Tri-training [31] utilizes three classifiers with a single view and assigns pseudo-labels to a classifier only if the other two classifiers agree on the label of the example. SIVLC [32] addresses the multi-view challenges encountered in co-training by utilizing sufficient irrelevant views and label consistency. Deep Co-Training [33] successfully constructs deep collaborative classification networks on two views by imposing view difference constraints. Tri-net [34] enhances disparities by generating three other distinct classification models from a shared model with a single view.

In regression tasks, COREG [35] uses two K-nearest neighbor (K-NN) [36] models with different distance measures and K values for regression analysis. CoBCReg [37] uses three RBFNN [38] regressors based on COREG to perform confident instance selection through a committee mechanism. In addition, by integrating a safer and self-paced learning process [39], the collaborative regression model can learn security pseudo-labels with higher quality, resulting in improved model performance. In real-world scenarios, E-CoGRF [40] combines co-training and a random forest model with information entropy grouping to assess the severity levels of depressive symptoms.

In image processing tasks, particularly in the field of medical images, Co-DeepSVS [41] applies support vector regression to multi-view deep co-training networks for survival time estimation. UCMT [42] uses the mean teacher model as a base co-learner while incorporating uncertainty-

guided region mixing for medical image segmentation. Some studies [43] have improved the semantic segmentation performance achieved for medical images by fusing collaborative training with adversarial training. In addition, in the field of remote sensing images, RSCoTr [44] combines multi-task learning and co-training to construct a transformer that can simultaneously handle three tasks: classification, segmentation and detection.

In addition to the previously mentioned research areas, co-training is also commonly employed in subfields such as multi-label learning [45], time series learning [46], and class-imbalanced learning [47]. Furthermore, the amalgamation of co-training with other semi-supervised techniques can significantly enhance the performance achieved in various tasks [48, 49].

3 The proposed algorithm

In this section, we present the implementation details of the S2RMS algorithm. The core techniques of the S2RMS model include embedding space mapping, a similarity-based instance selection strategy and pseudo-label smearing. Finally, the process details of the S2RMS algorithm are given. Table 1 details the notations used in the paper.

Table 1 Notations

Notation	Meaning
$\mathbf{L}$	The labeled dataset
$\mathbf{U}$	The unlabeled dataset
n_l	The number of labeled instances
n_u	The number of unlabeled instances
h_i	The i-th regressor
k	The number of regressors
$\mathbf{L}'$	The transformed labeled dataset
$\mathbf{U}'$	The transformed unlabeled dataset
x	The instance
d	The dimensionality of the embedding space
f^d	The Triplet neural network
$\mathcal{L}$	The loss function
$\mathbf{P}$	The unlabeled data pool
τ	The similarity threshold
$\mathbf{L}'_i, \mathbf{C}'_i$	The subsets of the transformed labeled dataset
μ	The standard deviation threshold
δ	The validation error
Δ_x	The confidence of x
Ω_i	The i-th set of instances with highest confidence

3.1 The model

Let $\mathbf{L} = \{(x_i, y_i)\}_{i=1}^{n_l}$ denote the labeled dataset, and $\mathbf{U} = \{x_j\}_{j=n_l+1}^{n_l+n_u}$ the unlabeled dataset, where n_l and n_u represent the numbers of labeled and unlabeled instances, respectively. Let $\mathbf{H} = \{h_j\}_{j=1}^{k}$ denote a set of k regressors and $\mathcal{L}$ be the loss function using mean square error (MSE). A general approach using co-training is to train k learners using the original labeled data $\mathbf{L}$. Then, the unlabeled data $\mathbf{U}$ are selected to facilitate mutual learning among the learners and improve the performance of the algorithm. In our algorithm, $\mathbf{L}$ and $\mathbf{U}$ are mapped to $\mathbf{L}'$ and $\mathbf{U}'$, respectively, by f^d to better select unlabeled data for training. f^d represents a metric-based neural network that estimates similarity and maps data to the embedding space, with d denoting the dimensionality of the embedding space. Furthermore, the label smearing technique is proposed as a means of obtaining stable instances. The regressors then passe through these stable instances to smooth the predicted output. Then, the objective function can be expressed as follows:

$$\min_{\mathbf{H}} E = \sum_{j\in\{1,\ldots,k\}} \left(\sum_{x_l \in \mathbf{L}'_j} \mathcal{L}(h_j(x_l), y_l) + \sum_{x_s \in \Omega_j} \mathcal{L}(h_j(x_s), y_s) \right), \tag{1}$$

where $\mathbf{L}'_j$ is the j-th labeled subset and Ω_j is the j-th set of stable instances with high confidence. To optimize the training process, a pool of unlabeled instances is constructed using a similarity metric strategy. Each optimization iteration applied to the regressors is performed using the data in this pool.

3.2 Embedding space mapping

To address the first problem of co-training, an effective approach must employ an improved metric for instance selection. The typical approach involves randomly selecting a specific number of unlabeled instances and assigning them pseudo-labels. This approach overlooks the distribution of the given data, potentially resulting in the selection of noisy data that can contaminate the training set. Considering the smoothing assumption, the selected unlabeled data should conform to the same distribution as that of the labeled data. Next, a metric method is employed to find unlabeled data that are close to the labeled data. One potential approach is to use the K-NN algorithm, which enables the direct calculation of such metrics. However, constrained by the manifold assumption, we use f^d to replace the simple linear metric methods. For f^d, we utilize a triplet neural network. This

network consists of three identical subnetworks that receive an input pair and extract instance features. These features are then connected to form features that are associated with the input pair. The final layer of the network estimates the similarity with the input pairs.

A pairwise dataset $\{(x_i, \mathbf{P}_{x_i}, \mathbf{N}_{x_i})\}_{i=1}^{n_l}$ is generated from $\mathbf{L}$ to train the Triplet neural network. Specifically, three objects derived from the pairs are the anchor x_i, positive instance set $\mathbf{P}_{x_i}$ and negative instance set $\mathbf{N}_{x_i}$. A positive instance has high similarity to the anchor, and a negative instance has the opposite relationship. For the anchor $x_a \in \mathbf{L}$, the positive instance set $\mathbf{P}_{x_a}$ and the negative instance set $\mathbf{N}_{x_a}$ are constructed by d_{ij}. The initial similarity d_{ij} is calculated between instances x_i and x_j, while $i \neq j$ is calculated by the Euclidean distance measures. The numbers of positive and negative instances are set to 2.

After generating the pairwise dataset for all anchor instances, the Triplet neural network exploits $n_l(n_l-1)$ pairs of instances. Then, the Ranked List Loss function $\mathcal{L}_{\mathrm{RLL}}$ [26] is employed as follows:

$$\begin{aligned}
\mathcal{L}_{\mathrm{P}}(x_a, \mathbf{P}_{x_a}) &= \sum_{x_j \in \mathbf{P}_{x_a}} \frac{\exp(\varepsilon(d_{aj} - (\alpha - M)))}{\sum_{x_j \in \mathbf{P}_{x_a}} \exp(\varepsilon(d_{aj} - (\alpha - M)))} (d_{aj} - (\alpha - M)), \\
\mathcal{L}_{\mathrm{N}}(x_a, \mathbf{N}_{x_a}) &= \sum_{x_j \in \mathbf{N}_{x_a}} \frac{\exp(\varepsilon(\alpha - d_{aj}))}{\sum_{x_j \in \mathbf{N}_{x_a}} \exp(\varepsilon(\alpha - d_{aj}))} (\alpha - d_{aj}), \\
\mathcal{L}_{\mathrm{RLL}} &= (1-\gamma)\mathcal{L}_{\mathrm{P}}(x_a, \mathbf{P}_{x_a}) + \gamma \mathcal{L}_{\mathrm{N}}(x_a, \mathbf{N}_{x_a}),
\end{aligned} \tag{2}$$

where α is the boundary parameter, ε is the temperature parameter, and M is the margin value, $\mathcal{L}_{\mathrm{P}}(x_a, \mathbf{P}_{x_a})$ and $\mathcal{L}_{\mathrm{N}}(x_a, \mathbf{N}_{x_a})$ represent the positive and negative instance losses, respectively. γ is the weight that controls the percentage of losses. In our experiments, we treat both targets equally and set γ to 0.5. In the feature space of the Triplet neural network, $\mathcal{L}_{\mathrm{RLL}}$ brings positive instances closer to each other than the negatives are by a margin of M.

The output of the Triplet neural network depends on the form of the input data. In the network, the last layer is designated as the similarity output layer, while the output of the penultimate hidden layer is employed as the embedding space. In the case of a single input, the network generates the deep embedding space of the input data. For pairs of inputs, the network outputs the similarity between pairs of inputs. In particular, when labeled or unlabeled data is employed as inputs, the network returns the data after the metric mapping. Furthermore, for pairs of inputs comprising labeled and unlabeled data, the network outputs the similarity between pairs of data. Consequently, the Triple neural network is capable of executing metric space mapping and similarity measuring operations.

Once the training of the Triple neural network has been completed, $\mathbf{L}$ and $\mathbf{U}$ are mapped to $\mathbf{L}'$ and $\mathbf{U}'$ through the network. For each instance in $\mathbf{L}'$, the similarity between it and any instance in $\mathbf{U}'$ is computed by the network. This similarity is then compared to a threshold value, τ, and the two highest unlabeled instances with similarity below this threshold are selected. The similarity threshold τ is used to control the bounds of the similar instance selection procedure. These instances are used to construct an unlabeled data pool. In each iteration, the Triplet neural network selects a batch consisting of new data to update the pool according to τ. The threshold τ decreases with the number of iterations as follows:

$$\tau_{t+1} = \lambda \tau_t, \tag{3}$$

where t is the number of iterations and λ is the attenuation coefficient. The selected data are then removed from $\mathbf{U}'$. Figure 2 shows the unlabeled instances selection process.

3.3 Co-training with pseudo-label smearing

To address the second problem encountered during co-training, an effective training framework is proposed. k pairs of subsets $(\mathbf{L}'_i, \mathbf{C}'_i)$, $i \in \{1, \ldots, k\}$ are randomly and repeatedly selected from $\mathbf{L}'$. $\mathbf{L}'_i$ is used to initialize regressor h_i, and

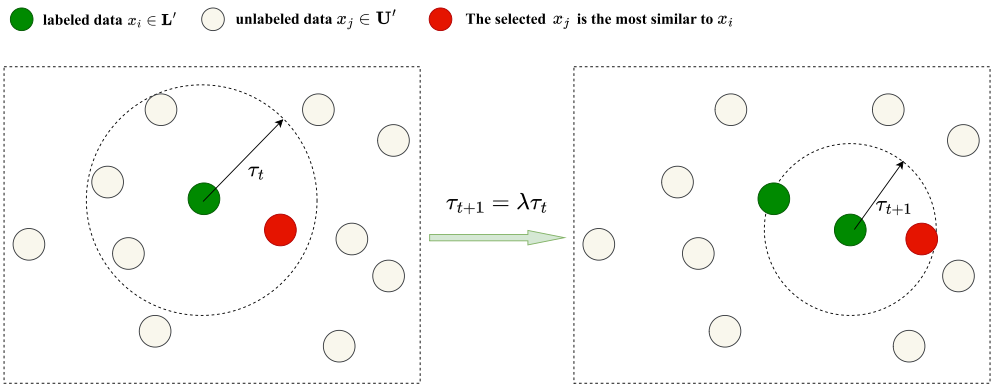

Fig. 2 The unlabeled instances selection process. For each embedded labeled instance $x_i \in \mathbf{L}'$, the Triplet neural network selects the most similar instances from the embedded set $\mathbf{U}'$ according to the threshold τ. The threshold τ decreases with the number of iterations

$\mathbf{C}_i'$ is used as part of the confidence calculation. Specifically, $\mathbf{C}_i'$ is not included in the training data.

In each iteration, other regressors assign pseudo-labels to all instance data in the unlabeled data pool. Since the pseudo-labels assigned to the unlabeled data may be inaccurate, adding them to the training set of the current regressor will degrade the performance of this regressor. Utilizing these erroneous pseudo-labels is equivalent to introducing noise to the training set. To address this issue, we employ a smearing technique to assign stable pseudo-labels to the unlabeled data. The purpose of label smearing is to select stable instances for improving the robustness of the regressors by smoothing out the differences between different collaborative regressors. At its core, label smearing calculates the variance of the predicted values of the unlabeled data, treats the unlabeled data with smaller variances as stable instances, and smooths out the differences between the predictions of the different regressors by averaging them.

For regressor h_i, the other regressors $h_{j,j\neq i}$ compute the predicted values for all instances in the unlabeled data pool. Then, the stable instances are filtered from these instances through a stability assessment. Stability is expressed as the variance of the values predicted by these regressors for the instances, as follows:

$$STD_i(x_u) = \sqrt{\frac{1}{k-1}\sum_{j=1,j\neq i}^{k}(h_j(x_u) - \frac{1}{k-1}\sum_{j=1,j\neq i}^{k}h_j(x_u))^2}, \quad (4)$$

where x_u is an instance in the unlabeled data pool. In the context of the S2RMS algorithm, the smearing strategy involves calculating the stability of the unlabeled instances based on the variance of the predictions of the regressors. The instances with less variance are then selected as stable instances, and their predicted values are replaced with the mean prediction value, which serves as their corresponding pseudo-labels. Once stable instances are obtained and pseudo-labeling is completed, confidence calculation can be performed with h_i as follows:

$$\Delta_{x_u} = \frac{\delta - \delta'}{\delta + \delta'}, \quad (5)$$

where δ and δ' represent the validation errors induced on $\mathbf{C}_i'$ prior to and following the update of $\left(x_u, \frac{1}{k-1}\sum_{j=1,j\neq i}^{k}h_j(x_u)\right)$, respectively. In particular, the validation error in the algorithm is defined as the root mean square (RMSE). Instances with high confidence expand the training set of h_i, and the corresponding regressor is retrained.

Algorithm 1 S2RMS algorithm.

input : Labeled dataset $\mathbf{L} = \{(x_i, y_i)\}_{i=1}^{n_l}$, unlabeled dataset $\mathbf{U} = \{x_j\}_{j=n_l+1}^{n_l+n_u}$, the initial value of τ, the maximum number of iterations T, the value of μ, the number of regressors k, the number of confident instances R.
output: Prediction regressor set $\mathbf{H} = \{h_j\}_{j=1}^{k}$.

1 Generate pairwise data from $\mathbf{L}$;
2 Use the pairwise data to learn a Triplet neural network f^d according to (2);
3 $\mathbf{L}' \leftarrow f^d(\mathbf{L}), \mathbf{U}' \leftarrow f^d(\mathbf{U})$;
4 **for** $i \leftarrow 1$ **to** k **do**
5 $(\mathbf{L}_i', \mathbf{C}_i') \leftarrow RandomSelect(\mathbf{L}'), h_i \leftarrow Learn(\mathbf{L}_i')$;
6 **for** $t \leftarrow 1$ **to** T **do**
7 $\mathbf{P} \leftarrow \emptyset$;
8 **for** *each* $x_l \in \mathbf{L}'$ **do**
9 Calculate the similarity between x_l and each $x_u \in \mathbf{U}'$ using f^d;
10 Move the top-2 similar instances from $\mathbf{U}'$ to $\mathbf{P}$;
11 Decrease the value of τ according to (3);
12 **while** $\mathbf{P} \neq \emptyset$ **do**
13 **for** $i \leftarrow 1$ **to** k **do**
14 Calculate the validation error δ_i of h_i on $\mathbf{C}_i'$;
15 $\Pi_i \leftarrow \emptyset, \Omega_i \leftarrow \emptyset$;
16 **for** *each* $x_u \in \mathbf{P}$ **do**
17 Calculate $STD_i(x_u)$ according to (4);
18 **if** $STD_i(x_u) \leq \mu$ **then**
19 $\tilde{y}_u \leftarrow \frac{1}{k-1}\sum_{j=1,j\neq i}^{k}h_j(x_u)$, $\Pi_i \leftarrow \Pi_i \cup \{(x_u, \tilde{y}_u)\}$;
20 $h_i' \leftarrow Learn(\{(x_u, \tilde{y}_u)\} \cup \mathbf{L}_i')$;
21 Calculate the validation error δ_i' of h_i' on $\mathbf{C}_i'$;
22 $\Delta_{x_u} \leftarrow \frac{\delta - \delta'}{\delta + \delta'}$;
23 **for** $r \leftarrow 1$ **to** R **do**
24 $\Omega_i \leftarrow \Omega_i \cup \{\arg\max_{x_u \in \Pi_i} \Delta_{x_u} | \Delta_{x_u} > 0\}$;
25 $\mathbf{P} \leftarrow \mathbf{P} \backslash \Omega_i, \mathbf{L}' \leftarrow \mathbf{L}' \cup \Omega_i$;
26 **if** $\cup_{i=1}^{k}\Omega_i = \emptyset$ **then**
27 break; // No more confident instances.
28 **for** $i \leftarrow 1$ **to** k **do**
29 $\mathbf{L}_i' \leftarrow \mathbf{L}_i' \cup \Omega_i, h_i \leftarrow Learn(\mathbf{L}_i')$;
30 **return** $\mathbf{H} = \{h_j\}_{j=1}^{k}$;

3.4 Algorithm description

Algorithm 1 shows the complete training process of S2RMS. The inputs of S2RMS include the labeled dataset $\mathbf{L} = \{(x_i, y_i)\}_{i=1}^{n_l}$, the unlabeled dataset $\mathbf{U} = \{x_j\}_{j=n_l+1}^{n_l+n_u}$, the initial value of τ, the maximum number of iterations T, the value of μ, the number of regressors k, and the number of confident instances R.

A Triplet neural network f^d is trained to generate pairwise data from $\mathbf{L}$ according to (2). In line 3, the labeled dataset $\mathbf{L}$ and unlabeled dataset $\mathbf{U}$ are mapped by f^d to $\mathbf{L}'$ and $\mathbf{U}'$, respectively. The transformed data $\mathbf{L}'$ are then randomly split into 2 subsets $\mathbf{L}_i'$ and $\mathbf{C}_i'$, as shown in line 5. k different

subsets $\mathbf{L}_i', i \in \{1, \ldots, k\}$ are obtained after repeating the random selection procedure k times. These subsets are used to initialize the k regressors $\{h_j\}_{j=1}^{k}$.

The next step is to train these regressors. At the beginning of each iteration, in lines 7 to 10, an unlabeled data pool $\mathbf{P}$ is created. This pool is generated by selecting unlabeled data using the metric method and the value of the threshold τ with the help of f^d. The threshold τ is then decreased according to (3). For each regressor h_i, the other regressors $h_{j,j\neq i}$ select the instances with the highest confidence value. First, in line 14, the validation error δ_i of h_i on $\mathbf{C}_i'$ is calculated. Next, for each x_u in $\mathbf{P}$, its stability value is calculated according to (4). In lines 17 to 20, the instances x_u that satisfy the stability threshold μ receive pseudo-labels and are temporarily merged with the original training set $\mathbf{L}_i'$ to train the regressor h_i'. Then, in line 21, the validation error δ_i of h_i' on $\mathbf{C}_i'$ is calculated. In line 22, the confidence Δ_{x_u} of x_u is calculated according to (5). As shown in line 24, the instances x_u with the highest confidence values are selected and added to Ω_i. Eventually, at most R instances are contained in Ω_i. In line 29, the instances of Ω_i are integrated into the training set of h_i. Regressor h_i is subsequently retrained on the updated training set.

A set of regression models $\mathbf{H} = \{h_j\}_{j=1}^{k}$ is obtained for data prediction purposes at the end of the S2RMS training process. The predicted label of x_u is then given by

$$\hat{y}_{x_u} = \frac{1}{k}\sum_{j=1}^{k} h_j(x_u). \tag{6}$$

S2RMS controls the differences among the predictions of learners by using label smearing during the training process. This ensures that the prediction results of the learners are stable without the need for weighted averaging.

3.5 Complexity analysis

The co-training process is where the complexity of the entire algorithm lies. This stage involves processing each x_u contained in the unlabeled data pool individually. In the S2RMS algorithm, we use random forests as the underlying co-regressors. These random forest regressors are executed in parallel during the iterative process. Therefore, we multiply the dominant term of the corresponding time complexity by the number of iterations to form the final complexity representation.

The random forest model typically employs a classification and regression tree (CART) as its base model. The time complexity of constructing a single CART is denoted by $O(I_{\text{ins}} \times A_{\text{attr}} \times \log I_{\text{ins}})$, where I_{ins} denotes the number of instances and A_{attr} denotes the number of attributes. Therefore, the time complexity of constructing a random forest model is $O(T_{\text{trees}} \times I_{\text{ins}} \times A_{\text{attr}} \times \log I_{\text{ins}})$, where T_{trees} represents the number of trees. As the computation process involves random feature selection, additional time is required to handle this process. Consequently, the average time complexity of constructing a random forest model is $O(T_{\text{trees}} \times I_{\text{ins}} \times A_{\text{attr}} \times \log I_{\text{ins}})$.

Co-training involves calculating confidence values and selecting instances based on these confidence levels. During the confidence calculation, additional random forest regressors are constructed for each stable instance x_u in the unlabeled data pool $\mathbf{P}$. When selecting confident instances, the time cost is only $O(R)$, where R is the number of confident instances. This cost can be ignored compared to the cost of the confidence calculation. Therefore, the total time complexity is

$$O(k \times N \times S \times T_{\text{trees}} \times I_{\text{ins}} \times A_{\text{attr}} \times \log I_{\text{ins}}),$$

where N is the maximum number of iterations and S is the number of stable instances. In the worst-case, all unlabeled instances are stable instances, hence the total time complexity is

$$O(k \times N \times |\mathbf{P}| \times T_{\text{trees}} \times I_{\text{ins}} \times A_{\text{attr}} \times \log I_{\text{ins}}),$$

where $|\mathbf{P}|$ is the cardinality of $\mathbf{P}$ and $|\mathbf{P}| \geq S$.

4 Experiments

To validate the effectiveness of our approach, we employ S2RMS to conduct a diverse range of regression tasks involving inductive semi-supervised learning. First, we test it on a total of fourteen datasets obtained from machine learning repositories (e.g., UCI, Delve, and Statlib). We calculate the RMSE values of S2RMS and the comparison methods. The effectiveness of similarity-based instance selection strategy and pseudo-label smearing techniques are evaluated through ablation experiments. All the experiments are performed on PyTorch [50].

4.1 Experimental settings

The detailed configuration of the experimental setup is as follows.

Datasets Table 2 lists the fourteen real-world datasets used in our experiments. For each dataset, we randomly select 2000 instances as the training set, while the remaining instances are used for testing. The training set is divided into labeled and unlabeled portions based on a specific ratio. In our experiments, 0.5%, 2.5% and 5% of the training set are used as labeled data in the separate scenarios. In most cases (except Section 4.3.4), 20% of the training set is used for validation.

Table 2 Datasets information

Name	Instances	Features	Source
Abalone	4 177	8	UCI
Bank32nh	8 192	32	Delve
Cal_housing	20 460	8	StatLib
Cpu_act	8 192	21	Delve
Cpu_small	8 192	12	Delve
Elevators	9 517	7	UCI
Folds5x2_pp	9 565	4	UCI
Kin8nm	8 192	8	Delve
Parkinsons	5 875	21	UCI
Pollen	3 848	5	StatLib
Puma8NH	8 192	8	UCI
Space_ga	3 107	6	StatLib
Wind	6 574	14	StatLib
Wine_quality	6 497	11	UCI

As an illustration, let us consider the Abalone dataset. We randomly select 2000 instances as the training set, while the remaining 2177 instances are used for testing. With a label rate of 2.5%, we include 50 instances in the labeled dataset, whereas the remaining 1950 instances are included in the unlabeled dataset. Among them, 10 instances serve as the validation set.

Parameters In this study, we choose a neural network with three hidden layers as Triplet metric model. In addition, we select specific regressors and parameters to suit our purposes. The three networks in the Triplet neural network share the same architecture, and each hidden layer uses the LeakyReLU [51] function to perform nonlinear transformations. The numbers of neurons in the three hidden layers are 16, 32 and 64. In particular, the dimensionality of the input layer is the number of features contained in the input data, while the output of the second hidden layer is taken as the embedding space. For the parameter settings of the Ranked List Loss function, α, ϵ and M are set to 40, -0.5 and 35, respectively.

To increase the diversity among the learners, three random forest regressors are initialized with different parameters. Specifically, the three regressors are based on 100 estimators, 120 estimators and 150 estimators. The minimum instance splits are set to 1, 2 and 3. In the experiments, the initial similarity threshold τ and the number of confident instances R are set to 0.017 and 2, respectively. The attenuation coefficient λ is set to 0.98. Instances with μ less than 0.02 are used for the selection of stable instances. The number of total experiments is set to 30.

Metrics The RMSE and R^2 metrics are chosen to evaluate the effectiveness of the tested algorithms. The RMSE is the root mean square error, which measures the square root of the average of the squared differences between the predicted and ground-truth values divided by the number of observations n_{test}. This metric calculates the difference between the predicted and ground-truth values and is particularly impacted by extreme values in the utilized dataset. The RMSE is calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2}, \tag{7}$$

where y_i represents the ground-truth value, $\hat{y}_i$ represents the predicted value and n_{test} represents the number of instances.

The coefficient of determination R^2 is an indicator used to measure the degree of model fitness. It indicates the proportion of dependent variable variance that the model can explain. R^2 is calculated as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n_{\text{test}}} \left(y_i - \hat{y}_i\right)^2 / n_{\text{test}}}{\sum_{i=1}^{n_{\text{test}}} \left(y_i - \bar{y}_i\right)^2 / n_{\text{test}}}, \tag{8}$$

where $\bar{y}_i$ represents the predicted average value. R^2 measures the extent to which the model explains the total variance, with a normal range of values between 0 and 1. A value closer to 1 indicates a better level of fit, while a value closer to 0 indicates a worse level of fit. A value less than 0 indicates that the model does not fit well and cannot make predictions.

4.2 Comparison with semi-supervised methods

S2RMS is compared with the following methods to further evaluate its performance.

- **COREG**: COREG uses a fixed K value and different distance metric algorithms to obtain the best experimental results. Two 3-NN regressors based on the Euclidean and Mahalanobis distance metrics are utilized. A new method using the mean squared error (MSE) is employed to calculate the confidence values.
- **SAFER** [17]: The SAFER semi-supervised regression method transforms the task of interest into a geometric projection problem. This approach utilizes multiple regressors to learn security predictions. The problem that unlabeled data may degrade the performance of the utilized algorithm is effectively circumvented.

- **DML-S2R** [52]: DML-S2R aims to learn the metric space of similar instances by efficiently using both unlabeled and labeled data. This method consists of two main steps: 1) pairwise similarity modeling using labeled data and 2) metric learning based on a Triplet network with a large amount of unlabeled data.

Table 3 presents the comparison results produced by S2RMS and other algorithms under different label rates.

The best performance entries in Table 3 are identified with bullets. First, S2RMS is compared with COREG and SAFER, which are based on divergence. At all label rates, S2RMS achieves the lowest average RMSE, while COREG and SAFER exhibit significantly greater average RMSEs than does S2RMS. S2RMS outperforms both COREG and SAFER on average on more than 10 datasets under different label rates. With a 2.5% label rate, S2RMS outperforms COREG on 12 datasets and SAFER on 11 datasets. It is evident that S2RMS outperforms both COREG and SAFER by significant margins.

Second, S2RMS is compared with DML-S2R using the metric method. Table 3 shows that S2RMS outperforms DML-S2R on all datasets under all label rates. The average performance improvement is greater than 13%. With a lower label rate, S2RMS is more effective. The above results show that S2RMS outperforms DML-S2R. In addition, S2RMS performs better in cases with extremely small rates.

Regardless of whether the label rate is 2.5% or 5%, S2RMS has poor performance on the Folds5x2_pp and Bank32NH datasets. The reasons for this result are as follows. The first reason is the inaccurate embedding space mapping process. Due to the number of features and the structural characteristics of the datasets, a fixed embedding space dimensionality may lead to data misalignment. This issue affects the training of the learner. The second reason is the special distribution of the embedding space. Because of the first reason, algorithms such as K-NN perform well in this space, but the performance of random forests decreases significantly. In contrast to COREG and SAFER, S2RMS employs a decision tree for regression purposes. As a result, its use of random forests leads to a decrease in performance compared to that of the other two methods using K-NN.

As S2RMS and COREG share co-training similarities, we adapt the co-learner of S2RMS to adopt the same K-NN regressor as that of COREG. This enables a clearer and more precise comparison between the performance results of the algorithms, as they all use the same regression model. Figure 3 shows the results of the experimental comparisons.

The experimental results indicate that when the base regressor is replaced with the K-NN algorithm, the prediction performance of S2RMS tends to decrease on some datasets relative to that attained with random forests. However, S2RMS outperforms COREG on most datasets. On the other hand, S2RMS falls behind COREG in all aspects on the Folds5x2_pp and Puma8NH datasets. The primary reason for this outcome is that the Euclidean distance measure, which is used by K-NN, is not suitable for the metric embedding spaces of these two datasets. Additionally, higher RMSEs are observed for the Cal_housing, Space_ga, and Wine_quality datasets, where the labeled data rate is set to 5%. S2RMS has an overall advantage over COREG. The experimental results indicate that the performance of the proposed algorithm is affected by the differences among the base regressors. The random forest-based co-learner outperforms K-NN under the influence of the metric mapping strategy proposed in this paper.

4.3 Ablation study

4.3.1 Similarity-based instance selection strategy

First, the regression model is initialized by labeled data. According to the smoothing assumption in semi-supervised learning, two instances that are close in terms of distance in high-density regions should have similar outputs. Therefore, for an unlabeled instance that is close to a labeled instance, the regression model can predict its target value more accurately. Thus, we construct an unlabeled data pool by selecting instances that are similar or close to the labeled data from the unlabeled data. This unlabeled data pool not only accelerates the iteration speed of the algorithm but also provides available unlabeled data for the subsequent co-training process. In conclusion, we choose a deep metric network to simultaneously complete metric space mapping and similarity-based instance selection. In the metric space, similar instances are close to each other, while instances with large differences are far from each other. Based on this characteristic of the metric space, the selection of similar instances can be better realized, and the effectiveness of the unlabeled data pool is also guaranteed.

We compile a list of the performance improvements (in percentages) resulting from the similarity-based instance selection strategy. This improvement is calculated by subtracting the RMSE of the similarity-based instance selection strategy from the RMSE obtained without using the similarity-based instance selection strategy. Then, the result is divided by the RMSE obtained without similarity-based instance selection. Table 4 shows that the performances achieved by the model on most datasets improve.

In the absence of a similarity-based instance selection strategy, an unlabeled data pool is constructed in accordance with the pool construction strategy outlined in COREG. In particular, the size of the pool is fixed and instances are randomly selected from the unlabeled dataset to be added to the pool until it is filled. After one round of iteration, the unla-

Table 3 The values of RMSE produced by the comparison algorithms under different label rates

Ratio	Dataset	COREG	SAFER	DML-S2R	S2RMS
	Abalone	$0.1090_{\pm 0.0106}$	$0.1158_{\pm 0.0226}$	$0.1153_{\pm 0.0146}$	$\bullet\ 0.1072_{\pm 0.0089}$
	Bank32nh	$0.1598_{\pm 0.0221}$	$0.1579_{\pm 0.0124}$	$0.1642_{\pm 0.0086}$	$\bullet\ 0.1545_{\pm 0.0021}$
	Cal_housing	$0.2345_{\pm 0.0369}$	$0.2496_{\pm 0.0408}$	$0.2674_{\pm 0.0286}$	$\bullet\ 0.1987_{\pm 0.0010}$
	Cpu_act	$0.1813_{\pm 0.0175}$	$0.1625_{\pm 0.0230}$	$0.2033_{\pm 0.0067}$	$\bullet\ 0.1357_{\pm 0.0009}$
	Cpu_small	$0.1767_{\pm 0.0095}$	$0.1598_{\pm 0.0205}$	$0.1789_{\pm 0.0067}$	$\bullet\ 0.1243_{\pm 0.0008}$
	Elevators	$\bullet\ 0.0732_{\pm 0.0070}$	$0.0746_{\pm 0.0077}$	$0.0755_{\pm 0.0067}$	$0.0749_{\pm 0.0048}$
0.5%	Folds5x2_pp	$0.1224_{\pm 0.0129}$	$\bullet\ 0.1127_{\pm 0.0069}$	$0.1489_{\pm 0.0039}$	$0.1683_{\pm 0.0196}$
	Kin8nm	$0.1862_{\pm 0.0116}$	$0.1857_{\pm 0.0142}$	$0.1855_{\pm 0.0152}$	$\bullet\ 0.1641_{\pm 0.0023}$
	Parkinsons	$0.1269_{\pm 0.0038}$	$\bullet\ 0.1265_{\pm 0.0048}$	$0.1374_{\pm 0.0027}$	$0.1344_{\pm 0.0020}$
	Pollen	$0.3227_{\pm 0.0078}$	$0.3287_{\pm 0.0078}$	$0.3334_{\pm 0.0077}$	$\bullet\ 0.3179_{\pm 0.0096}$
	Puma8NH	$0.2373_{\pm 0.0034}$	$\bullet\ 0.2351_{\pm 0.0078}$	$0.2741_{\pm 0.0012}$	$0.2419_{\pm 0.0192}$
	Space_ga	$0.1985_{\pm 0.0139}$	$0.2210_{\pm 0.0111}$	$0.1999_{\pm 0.0121}$	$\bullet\ 0.1696_{\pm 0.0008}$
	Wind	$0.1427_{\pm 0.0094}$	$0.1328_{\pm 0.0091}$	$0.1462_{\pm 0.0117}$	$\bullet\ 0.1187_{\pm 0.0024}$
	Wine_quality	$0.1463_{\pm 0.0141}$	$0.1473_{\pm 0.0182}$	$0.1721_{\pm 0.0044}$	$\bullet\ 0.1358_{\pm 0.0024}$
	Abalone	$0.0920_{\pm 0.0075}$	$0.0964_{\pm 0.0052}$	$0.1066_{\pm 0.0052}$	$\bullet\ 0.0854_{\pm 0.0011}$
	Bank32nh	$\bullet\ 0.1503_{\pm 0.0053}$	$0.1527_{\pm 0.0054}$	$0.1609_{\pm 0.0039}$	$0.1575_{\pm 0.0055}$
	Cal_housing	$0.1844_{\pm 0.0219}$	$0.1975_{\pm 0.0218}$	$0.2245_{\pm 0.0071}$	$\bullet\ 0.1837_{\pm 0.0029}$
	Cpu_act	$0.1317_{\pm 0.0217}$	$\bullet\ 0.1079_{\pm 0.0231}$	$0.1188_{\pm 0.0054}$	$0.1202_{\pm 0.0030}$
	Cpu_small	$0.1218_{\pm 0.0293}$	$0.0991_{\pm 0.0242}$	$0.1749_{\pm 0.0055}$	$\bullet\ 0.0952_{\pm 0.0096}$
	Elevators	$0.0645_{\pm 0.0014}$	$0.0643_{\pm 0.0016}$	$0.0689_{\pm 0.0016}$	$\bullet\ 0.0590_{\pm 0.0056}$
2.5%	Folds5x2_pp	$0.0835_{\pm 0.0044}$	$\bullet\ 0.0795_{\pm 0.0024}$	$0.1109_{\pm 0.0017}$	$0.0848_{\pm 0.0029}$
	Kin8nm	$0.1612_{\pm 0.0035}$	$0.1602_{\pm 0.0053}$	$0.1621_{\pm 0.0023}$	$\bullet\ 0.1530_{\pm 0.0082}$
	Parkinsons	$0.1047_{\pm 0.0040}$	$0.1079_{\pm 0.0072}$	$0.1195_{\pm 0.0036}$	$\bullet\ 0.0999_{\pm 0.0048}$
	Pollen	$0.3186_{\pm 0.0076}$	$0.3207_{\pm 0.0075}$	$0.3369_{\pm 0.0076}$	$\bullet\ 0.3026_{\pm 0.0054}$
	Puma8NH	$0.2092_{\pm 0.0060}$	$0.2081_{\pm 0.0042}$	$0.2470_{\pm 0.0017}$	$\bullet\ 0.1946_{\pm 0.0082}$
	Space_ga	$0.1725_{\pm 0.0117}$	$0.1952_{\pm 0.0089}$	$0.1844_{\pm 0.0098}$	$\bullet\ 0.1682_{\pm 0.0065}$
	Wind	$0.1185_{\pm 0.0175}$	$0.1130_{\pm 0.0049}$	$0.1203_{\pm 0.0046}$	$\bullet\ 0.0984_{\pm 0.0030}$
	Wine_quality	$0.1381_{\pm 0.0121}$	$0.1374_{\pm 0.0083}$	$0.1669_{\pm 0.0028}$	$\bullet\ 0.1332_{\pm 0.0016}$
	Abalone	$0.0877_{\pm 0.0037}$	$0.0949_{\pm 0.0047}$	$0.1016_{\pm 0.0032}$	$\bullet\ 0.0874_{\pm 0.0071}$
	Bank32nh	$\bullet\ 0.1494_{\pm 0.0024}$	$0.1518_{\pm 0.0036}$	$0.1591_{\pm 0.0030}$	$0.1567_{\pm 0.0014}$
	Cal_housing	$\bullet\ 0.1713_{\pm 0.0128}$	$0.1817_{\pm 0.0069}$	$0.2093_{\pm 0.0075}$	$0.1846_{\pm 0.0047}$
	Cpu_act	$0.1065_{\pm 0.0153}$	$0.0863_{\pm 0.0281}$	$0.0879_{\pm 0.0054}$	$\bullet\ 0.0856_{\pm 0.0113}$
	Cpu_small	$0.0911_{\pm 0.0203}$	$\bullet\ 0.0664_{\pm 0.0277}$	$0.0902_{\pm 0.0054}$	$0.0917_{\pm 0.0177}$
	Elevators	$0.0634_{\pm 0.0014}$	$0.0628_{\pm 0.0012}$	$0.0645_{\pm 0.0017}$	$\bullet\ 0.0599_{\pm 0.0021}$
5%	Folds5x2_pp	$0.0754_{\pm 0.0020}$	$\bullet\ 0.0738_{\pm 0.0013}$	$0.0848_{\pm 0.0022}$	$0.0834_{\pm 0.0032}$
	Kin8nm	$0.1477_{\pm 0.0021}$	$0.1510_{\pm 0.0030}$	$0.1623_{\pm 0.0028}$	$\bullet\ 0.1474_{\pm 0.0016}$
	Parkinsons	$0.0962_{\pm 0.0049}$	$0.0995_{\pm 0.0072}$	$0.0956_{\pm 0.0034}$	$\bullet\ 0.0928_{\pm 0.0012}$
	Pollen	$0.3177_{\pm 0.0071}$	$0.3185_{\pm 0.0073}$	$0.3293_{\pm 0.0074}$	$\bullet\ 0.3045_{\pm 0.0028}$
	Puma8NH	$0.2026_{\pm 0.0029}$	$0.2019_{\pm 0.0015}$	$0.2029_{\pm 0.0018}$	$\bullet\ 0.1927_{\pm 0.0012}$
	Space_ga	$0.1659_{\pm 0.0166}$	$0.1852_{\pm 0.0071}$	$0.1782_{\pm 0.0097}$	$\bullet\ 0.1658_{\pm 0.0221}$
	Wind	$0.1111_{\pm 0.0323}$	$0.1079_{\pm 0.0051}$	$0.1001_{\pm 0.0060}$	$\bullet\ 0.0996_{\pm 0.0031}$
	Wine_quality	$0.1353_{\pm 0.0086}$	$0.1347_{\pm 0.0071}$	$0.1685_{\pm 0.0038}$	$\bullet\ 0.1330_{\pm 0.0005}$

The best results obtained on each dataset are denoted with bullets

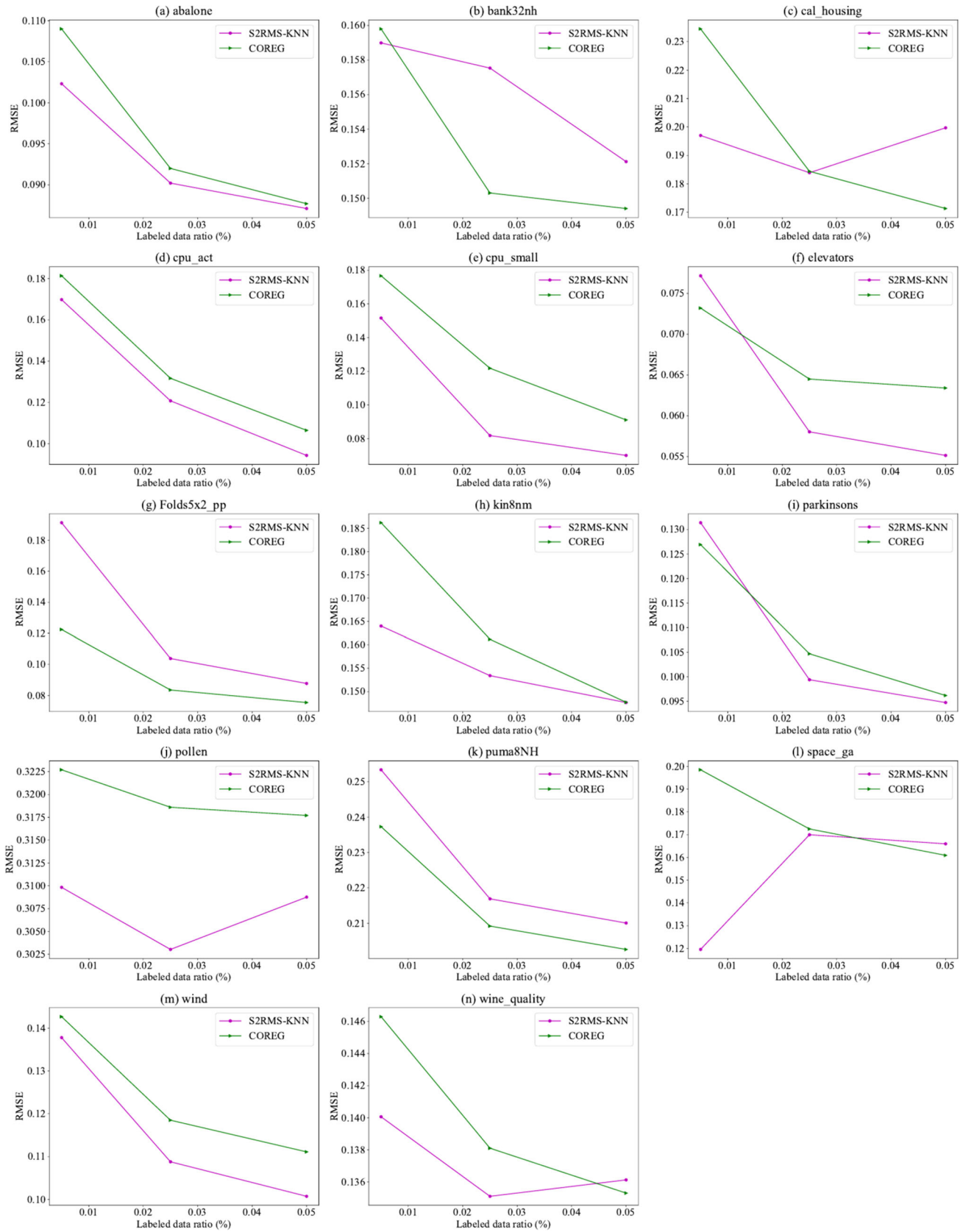

Fig. 3 RMSE comparisons between S2RMS-KNN and COREG conducted on fourteen datasets under different label rates. S2RMS-KNN denotes the S2RMS algorithm constructed after applying K-NN as the base regressor

Table 4 The RMSE values of the two methods were compared, one employing similarity-based instance selection and the other random selection

Dataset	Randomization-based strategy	Similarity-based strategy	Improvements
Abalone	$0.0901_{\pm 0.0062}$	• $0.0854_{\pm 0.0011}$	5.22%
Bank32nh	$0.1642_{\pm 0.0084}$	• $0.1575_{\pm 0.0055}$	4.14%
Cal_housing	$0.1894_{\pm 0.0191}$	• $0.1837_{\pm 0.0029}$	3.01%
Cpu_act	$0.1409_{\pm 0.0181}$	• $0.1202_{\pm 0.0030}$	14.69%
Cpu_small	$0.0958_{\pm 0.0102}$	• $0.0952_{\pm 0.0096}$	0.62%
Elevators	$0.0615_{\pm 0.0035}$	• $0.0590_{\pm 0.0056}$	4.07%
Folds5x2_pp	• $0.0842_{\pm 0.0033}$	$0.0848_{\pm 0.0029}$	−0.71%
Kin8nm	$0.1601_{\pm 0.0118}$	• $0.1530_{\pm 0.0082}$	4.43%
Parkinsons	$0.1009_{\pm 0.0024}$	• $0.0999_{\pm 0.0048}$	0.99%
Pollen	$0.3061_{\pm 0.0003}$	• $0.3026_{\pm 0.0054}$	1.14%
Puma8NH	$0.2020_{\pm 0.0041}$	• $0.1946_{\pm 0.0082}$	3.66%
Space_ga	$0.1750_{\pm 0.0162}$	• $0.1682_{\pm 0.0065}$	3.89%
Wind	$0.0986_{\pm 0.0041}$	• $0.0984_{\pm 0.0030}$	0.20%
Wine_quality	$0.1335_{\pm 0.0064}$	• $0.1332_{\pm 0.0016}$	0.22%

The label rate of the experiment is set to 2.5%

beled data pool is replenished by randomly selecting again. In this way, the construction process of the unlabeled data pool, as outlined in lines 7 to 11 of Algorithm 1, is replaced by the aforementioned method.

Experiments are conducted under a 2.5% label rate. Each dataset is used for 30 tests implemented under different label rates. All the data from the training process are mapped to the metric space. Table 4 presents the experimental RMSE results obtained before and after using similar selection. The pool size of the random selection process is set to 50. The results show that the similarity-based instance selection strategy always performs better.

The data pool of S2RMS is constructed by selecting unlabeled data that are similar to the labeled data. In the embedding space, similar instances are close to one another. More appropriate unlabeled data can be selected based on the distribution of the labeled data. According to the smoothing and manifold assumptions, these unlabeled data can be effectively used to expand the training set. Thus, the proposed method significantly improves upon the performance of random selection.

However, for the Folds5x2_pp dataset, the performance degrades. One possible reason for this finding is that this dataset has fewer features. After mapping the data to a higher-dimensional embedding space, some ambiguous features cannot correctly represent the data. Therefore, it is not possible to effectively select instances. This, in turn, affects the training process of the regressor.

4.3.2 Embedding space mapping

The COREG and SAFER comparison algorithms use regression and pseudo-labeling without embedding learning. In this case, we compare the S2RMS algorithm with the COREG and SAFER algorithms after applying the embedding space mapping strategy. Specifically, we additionally apply steps 1 to 3 of Algorithm 1 to the compared algorithms.

We then compare the performances achieved by the COREG and SAFER algorithms before and after adopting the mapping strategy. This approach aims to accurately evaluate the impact of embedding space mapping on the resulting algorithmic performance.

For an example with a 2.5% label rate, Table 5 shows the results obtained in the comparison experiments by the tested methods when using the mapping strategy. The experimental results indicate that S2RMS outperforms the other methods on the 11 datasets, even though COREG and SAFER also implement the mapping strategy.

Table 6 displays the results of comparison between the R^2 values produced before and after applying embedding mapping in the COREG and SAFER algorithms.

Table 6 shows that the two algorithms produce divergent results after applying the mapping strategy. Compared with that of COREG, the performance of COREG-ESP is significantly improved on only 6 datasets and is basically equal on 3 datasets. However, performance degradations are observed on 5 datasets: Bank32nh, Folds5x2_pp, Pollen, Space_ga, and Wine_quality. As discussed in the previous subsections, the metrics used in COREG may not be suitable for metric embedding mappings, resulting in negative shifts on some datasets. In contrast, the embedding mapping strategy works relatively well in the SAFER algorithm. The performance of SAFER-ESP is effectively improved on 12 datasets, with only slight decreases on the Bank32nh and Pollen datasets. Overall, the metric mapping strategy can improve the performance of these algorithms on most datasets.

Table 5 The values of R^2 produced by the comparison algorithms at a label rate of 2.5%

Dataset	COREG-ESP	SAFER-ESP	S2RMS
Abalone	$0.2738_{\pm 0.0458}$	$0.1132_{\pm 0.0734}$	• $0.3886_{\pm 0.0460}$
Bank32nh	$-0.2234_{\pm 0.0872}$	$-0.2447_{\pm 0.1077}$	• $-0.1723_{\pm 0.0847}$
Cal_housing	$0.1080_{\pm 0.0802}$	$0.3183_{\pm 0.1036}$	• $0.3762_{\pm 0.1359}$
Cpu_act	$0.3686_{\pm 0.1095}$	$0.4731_{\pm 0.1096}$	• $0.6866_{\pm 0.1941}$
Cpu_small	$0.4483_{\pm 0.1435}$	$0.4726_{\pm 0.1821}$	• $0.8685_{\pm 0.1979}$
Elevators	$0.4871_{\pm 0.0334}$	• $0.4960_{\pm 0.0246}$	$0.4843_{\pm 0.0256}$
Folds5x2_pp	$0.8616_{\pm 0.0217}$	• $0.8811_{\pm 0.0700}$	$0.8459_{\pm 0.0101}$
Kin8nm	• $0.3678_{\pm 0.0318}$	$0.3406_{\pm 0.0107}$	$0.3117_{\pm 0.0429}$
Parkinsons	$0.2642_{\pm 0.0530}$	$0.2628_{\pm 0.0748}$	• $0.4887_{\pm 0.0912}$
Pollen	$-2.9128_{\pm 0.1539}$	$-2.9131_{\pm 0.1504}$	• $-0.2070_{\pm 0.1513}$
Puma8NH	$0.1533_{\pm 0.0551}$	$0.1538_{\pm 0.0480}$	• $0.3411_{\pm 0.0349}$
Space_ga	$-1.3920_{\pm 0.2160}$	$-1.3540_{\pm 0.2347}$	• $0.4155_{\pm 0.2040}$
Wind	$0.0953_{\pm 0.0695}$	$0.2804_{\pm 0.0602}$	• $0.3989_{\pm 0.0551}$
Wine_quality	$-0.3879_{\pm 0.1020}$	$-0.2922_{\pm 0.1155}$	• $0.1419_{\pm 0.1555}$

The best results obtained on each dataset are denoted with bullets. COREG-ESP and SAFER-ESP denote the COREG and SAFER algorithms constructed after applying embedding space mapping (ESP), respectively

4.3.3 Pseudo-label smearing

In the pseudo-label calculation process, S2RMS uses the smearing technique to smooth out the predictions of the regressors. To evaluate the effectiveness of the proposed method, we compare the RMSE values obtained with and without pseudo-label smearing. Lines 16 to 22 of Algorithm 1 represent the process of label smearing. The core of this process is the calculation of stability, which is performed using (4) to filter the stable instances. Once the stable instances have been identified, the subsequent confidence assessment operation is performed. In the baseline experiment where label smearing is not employed, the step in line 17 of Algorithm 1 is ignored. Line 19 and subsequent operations are conducted directly without any form of judgement. The objective of this experiment is to assess the impact of stable instance filtering on the performance of the algorithm in the context of label smearing.

Figure 4 shows the results of the experimental comparisons.

Under the same configuration, the pseudo-label smearing technique can effectively improve the performance of the algorithm. At a label rate of 0.5%, the average RMSE is notably greater when employing the smearing technique than

Table 6 The values of R^2 produced by COREG and SAFER at a label rate of 2.5%

Dataset	COREG		SAFER	
	COREG	COREG-ESP	SAFER	SAFER-ESP
Abalone	$0.2661_{\pm 0.0538}$	• $0.2738_{\pm 0.0458}$	$0.1011_{\pm 0.0147}$	• $0.1132_{\pm 0.0734}$
Bank32nh	• $-0.2224_{\pm 0.0905}$	$-0.2234_{\pm 0.0872}$	• $-0.2042_{\pm 0.0903}$	$-0.2447_{\pm 0.1077}$
Cal_housing	$0.0851_{\pm 0.1737}$	• $0.1080_{\pm 0.0802}$	$0.2739_{\pm 0.1503}$	• $0.3183_{\pm 0.1036}$
Cpu_act	$0.2552_{\pm 0.1778}$	• $0.3686_{\pm 0.1095}$	$0.3381_{\pm 0.1898}$	• $0.4731_{\pm 0.1096}$
Cpu_small	$0.2685_{\pm 0.2345}$	• $0.4483_{\pm 0.1435}$	$0.3822_{\pm 0.1977}$	• $0.4726_{\pm 0.1821}$
Elevators	$0.3500_{\pm 0.0225}$	• $0.4871_{\pm 0.0334}$	$0.4475_{\pm 0.0246}$	• $0.4960_{\pm 0.0246}$
Folds5x2_pp	• $0.7135_{\pm 0.0159}$	$0.8616_{\pm 0.0217}$	$0.8155_{\pm 0.0084}$	• $0.8811_{\pm 0.0700}$
Kin8nm	$0.2288_{\pm 0.0239}$	• $0.3678_{\pm 0.0318}$	$0.2424_{\pm 0.0347}$	• $0.3406_{\pm 0.0107}$
Parkinsons	$0.1019_{\pm 0.0483}$	• $0.2642_{\pm 0.0530}$	$0.2210_{\pm 0.0900}$	• $0.2628_{\pm 0.0748}$
Pollen	• $-2.9091_{\pm 0.1492}$	$-2.9128_{\pm 0.1539}$	• $-2.9090_{\pm 0.1511}$	$-2.9131_{\pm 0.1504}$
Puma8NH	$0.1292_{\pm 0.0485}$	• $0.1533_{\pm 0.0551}$	$0.1386_{\pm 0.0405}$	• $0.1538_{\pm 0.0480}$
Space_ga	• $-1.3672_{\pm 0.2581}$	$-1.3920_{\pm 0.2160}$	$-1.4029_{\pm 0.2048}$	• $-1.3540_{\pm 0.2347}$
Wind	$0.0152_{\pm 0.2447}$	• $0.0953_{\pm 0.0695}$	$-0.1860_{\pm 0.0536}$	• $0.2804_{\pm 0.0602}$
Wine_quality	$-0.1888_{\pm 0.2463}$	• $-0.3879_{\pm 0.1020}$	$-0.3210_{\pm 0.1593}$	• $-0.2922_{\pm 0.1155}$

The best results obtained by each dataset are denoted with bullets. COREG-ESP and SAFER-ESP denote the COREG and SAFER algorithms constructed after applying embedding space mapping (ESP), respectively

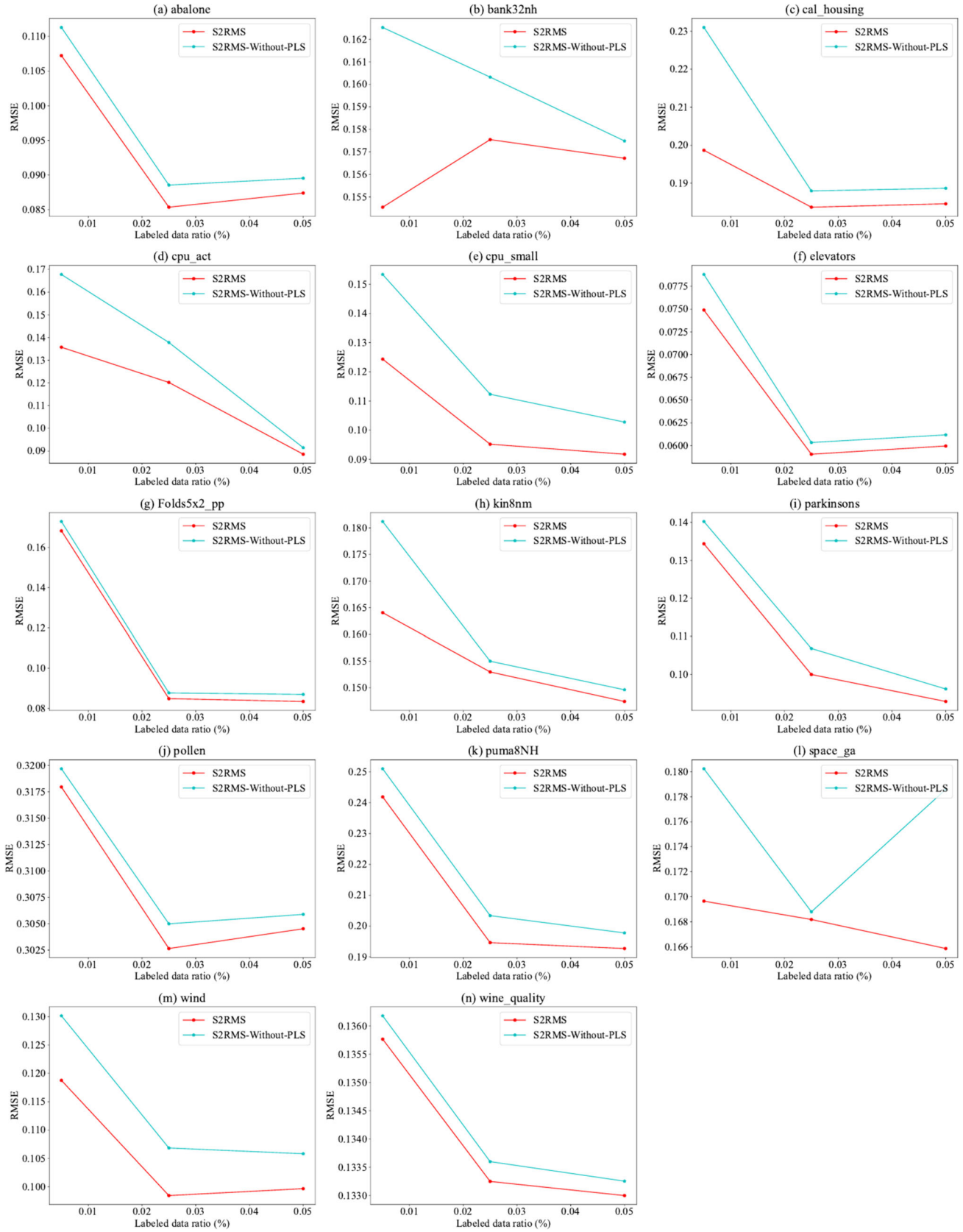

Fig. 4 Average RMSE values produced over thirty tests on fourteen datasets. S2RMS-Without-PLS denotes the S2RMS algorithm without the pseudo-label smearing (PLS) technique

that attained when not using it. With the help of smearing, the RMSE decreases smoothly on most of the datasets as the label rate increases. Especially on the Space_ga dataset, the RMSE is more volatile without the smearing technique. This indicates that the smearing technique can effectively leverage stable instances to optimize the regression performance of the algorithm.

4.3.4 The influence of validation set

Considering the relatively small size of the labeled dataset, the validation set $\mathbf{C}_i'$ is also expected to be small. To investigate potential biases in the confidence calculation executed during the pseudo-label smearing process caused by the size of the validation set, we conduct comparative experiments with different $\mathbf{C}_i'$ sizes.

The experiment of this subsection aims to study the impact of the validation set size. Therefore, we add some additional labeled data to increase the validation set. In this way, the size can be 30 or 50. We also consider the extreme case where the size of the validation set is 0. In this case, it is impossible to perform a confidence calculation, and the algorithm degenerates into a self-training mode in its ordinary form. Consequently, lines 14, 21, 22, and 24 of Algorithm 1 are disabled, and the strategy that replaces them is to randomly select R instances from Π_i.

Table 7 shows the experimental results obtained on the validation set $\mathbf{C}_i'$ under different size settings. The performance of the algorithm gradually improves as the size of the validation set increases. A larger validation set leads to more accurate calculation results for (5). Inevitably, the labeled dataset size is positively correlated with the size of the validation set. With a small labeled dataset, a small validation set can cause bias in the confidence computation. Nevertheless, the algorithm also performs better with a small validation set than it does without it.

4.3.5 Parameter sensitivity

The parameters τ and μ are key parameters that affect the performance of the S2RMS algorithm. One of the parameters, τ, is the attenuation coefficient, which is applied to the process of creating a pool of unlabeled instances. The value of τ reflects the size of the instance pool. The number of instances in the pool directly affects the performance of the co-regressors in subsequent iterations. The use of too few unlabeled instances is meaningless for the learner. An instance pool that is excessively large makes the learning process less efficient. To obtain a suitable value for τ, we observe the performance achieved by the algorithm with different parameter settings. When testing parameter τ, it is important to note that we only focus on the process related to parameter τ to clearly observe the experimental results. The co-training process removes the label smearing strategy and eliminates the experimental bias caused by the stable instances controlled by parameter μ.

Table 8 shows the experimental results produced by S2RMS with different τ values. A higher τ value implies that the conditions for the similarity threshold are relaxed to a greater extent. When τ is set to 0.008, this lower value prevents the selection of more unlabeled instances that do not satisfy the imposed similarity requirement. This results in a final unlabeled data pool that is too small or even empty after just a few iterations due to the decaying value of τ. The

Table 7 The values of R^2 obtained on the validation set $\mathbf{C}_i'$ under sizes of 0, 30 and 50 at a label rate of 2.5%

Dataset	$\lvert\mathbf{C}_i'\rvert = 0$	$\lvert\mathbf{C}_i'\rvert = 30$	$\lvert\mathbf{C}_i'\rvert = 50$
Abalone	$0.3217_{\pm 0.0046}$	$0.3617_{\pm 0.0510}$	$\bullet\ 0.3880_{\pm 0.0264}$
Bank32nh	$-0.6241_{\pm 0.1990}$	$-0.2864_{\pm 0.2465}$	$\bullet -0.2122_{\pm 0.1837}$
Cal_housing	$0.3879_{\pm 0.0074}$	$0.3722_{\pm 0.0537}$	$\bullet\ 0.3973_{\pm 0.0278}$
Cpu_act	$0.6230_{\pm 0.0009}$	$0.6673_{\pm 0.0115}$	$\bullet\ 0.6926_{\pm 0.0364}$
Cpu_small	$0.6854_{\pm 0.1011}$	$0.7104_{\pm 0.0238}$	$\bullet\ 0.7426_{\pm 0.0120}$
Elevators	$0.4582_{\pm 0.0006}$	$0.4824_{\pm 0.0101}$	$\bullet\ 0.4874_{\pm 0.0152}$
Folds5x2_pp	$0.8376_{\pm 0.0041}$	$0.8318_{\pm 0.0023}$	$\bullet\ 0.8406_{\pm 0.0070}$
Kin8nm	$0.1743_{\pm 0.0330}$	$\bullet\ 0.2872_{\pm 0.0394}$	$0.2685_{\pm 0.0128}$
Parkinsons	$0.3172_{\pm 0.1101}$	$0.4298_{\pm 0.0748}$	$\bullet\ 0.4354_{\pm 0.1142}$
Pollen	$-0.1654_{\pm 0.0779}$	$\bullet -0.1137_{\pm 0.0233}$	$-0.1289_{\pm 0.0407}$
Puma8NH	$0.1669_{\pm 0.1201}$	$\bullet\ 0.3237_{\pm 0.1863}$	$0.2115_{\pm 0.1337}$
Space_ga	$\bullet\ 0.4721_{\pm 0.1367}$	$0.4222_{\pm 0.0131}$	$0.4458_{\pm 0.0271}$
Wind	$0.3763_{\pm 0.0361}$	$0.4154_{\pm 0.0707}$	$\bullet\ 0.4289_{\pm 0.0890}$
Wine_quality	$0.1647_{\pm 0.1103}$	$0.2153_{\pm 0.1291}$	$\bullet\ 0.2283_{\pm 0.1321}$

The best results obtained on each dataset version are denoted with bullets. $\lvert\mathbf{C}_i'\rvert$ denotes the cardinality of the validation set $\mathbf{C}_i'$

Table 8 The values of R^2 produced under various settings of parameter τ at a label rate of 2.5%

Dataset	$\tau = 0.008$	$\tau = 0.011$	$\tau = 0.014$	$\tau = 0.017$	$\tau = 0.020$
Abalone	$0.3298_{\pm 0.1258}$	$0.3049_{\pm 0.1094}$	$\bullet\ 0.3378_{\pm 0.0827}$	$0.3339_{\pm 0.1371}$	$0.3245_{\pm 0.1104}$
Bank32nh	$-0.1567_{\pm 0.1844}$	$\bullet -0.1370_{\pm 0.1652}$	$-0.1380_{\pm 0.1285}$	$-0.1421_{\pm 0.1771}$	$-0.1500_{\pm 0.1801}$
Cal_housing	$0.3176_{\pm 0.0120}$	$0.3721_{\pm 0.0240}$	$0.3254_{\pm 0.0553}$	$\bullet\ 0.3975_{\pm 0.0683}$	$0.3345_{\pm 0.0334}$
Cpu_act	$0.6507_{\pm 0.0889}$	$0.6122_{\pm 0.0548}$	$0.6487_{\pm 0.0856}$	$\bullet\ 0.6859_{\pm 0.0400}$	$0.6666_{\pm 0.0798}$
Cpu_small	$0.6625_{\pm 0.0832}$	$0.6808_{\pm 0.0639}$	$0.6575_{\pm 0.1057}$	$\bullet\ 0.7198_{\pm 0.0752}$	$0.7108_{\pm 0.0560}$
Elevators	$0.4594_{\pm 0.0266}$	$0.4889_{\pm 0.0222}$	$0.4662_{\pm 0.0315}$	$\bullet\ 0.4908_{\pm 0.0260}$	$0.4532_{\pm 0.0301}$
Folds5x2_pp	$0.8373_{\pm 0.0141}$	$0.8322_{\pm 0.0064}$	$0.8435_{\pm 0.0189}$	$\bullet\ 0.8436_{\pm 0.0127}$	$0.8401_{\pm 0.0191}$
Kin8nm	$0.2237_{\pm 0.0283}$	$0.2580_{\pm 0.0094}$	$0.2510_{\pm 0.0083}$	$0.2459_{\pm 0.0242}$	$\bullet\ 0.2680_{\pm 0.0101}$
Parkinsons	$\bullet\ 0.3645_{\pm 0.0825}$	$0.3576_{\pm 0.0938}$	$0.3366_{\pm 0.0992}$	$0.3450_{\pm 0.0960}$	$0.3332_{\pm 0.0887}$
Pollen	$-0.1418_{\pm 0.0401}$	$-0.1454_{\pm 0.0572}$	$-0.1388_{\pm 0.0317}$	$\bullet -0.1193_{\pm 0.0636}$	$-0.1297_{\pm 0.0443}$
Puma8NH	$0.2916_{\pm 0.1027}$	$0.3046_{\pm 0.1188}$	$\bullet\ 0.2983_{\pm 0.1418}$	$0.2698_{\pm 0.0975}$	$0.2721_{\pm 0.1770}$
Space_ga	$0.4022_{\pm 0.0843}$	$0.3789_{\pm 0.0460}$	$0.4518_{\pm 0.0489}$	$0.4153_{\pm 0.0380}$	$\bullet\ 0.4522_{\pm 0.0599}$
Wind	$0.4121_{\pm 0.0102}$	$0.4120_{\pm 0.0110}$	$0.4117_{\pm 0.0027}$	$0.4160_{\pm 0.0229}$	$\bullet\ 0.4183_{\pm 0.0149}$
Wine_quality	$0.2000_{\pm 0.0204}$	$0.1945_{\pm 0.0334}$	$0.1830_{\pm 0.0205}$	$\bullet\ 0.2070_{\pm 0.0285}$	$0.1934_{\pm 0.0289}$

The best results obtained on each dataset are denoted with bullets

performance of the subsequent regressors remains relatively low due to the insufficient availability of unlabeled data.

As the value of τ increases to 0.017, more unlabeled instances are able to satisfy the larger threshold. Additionally, sufficient training rounds and data are available for improving the performance of the regressor until the threshold τ decays to the point where the instance pool can no longer be constructed. As the value of τ increases again, some unlabeled data that do not satisfy the imposed conditions are selected, affecting the performance of the algorithm. The experimental results indicate that a parameter τ setting of approximately 0.017 is appropriate.

The parameter μ denotes the stability coefficient, which is used to screen for stable instances that satisfy the preset conditions. The stability of a instance is indicated by the variance of the values predicted by multiple regressors. For each instance in the pool of unlabeled instances, when its stability value is lower than the parameter μ, it is regarded as a stable instance and marked with a pseudo-label. In summary, parameter μ controls the prediction variance among the co-learners by setting a stability threshold to ensure the robustness of the algorithm. Similar to when testing parameter τ, we focus solely on the process related to parameter μ and exclude the similarity-based instance selection process that is controlled by parameter τ. The pool of unlabeled

Table 9 The values of R^2 produced with various settings of μ at a label rate of 2.5%

Dataset	$\mu = 0.01$	$\mu = 0.02$	$\mu = 0.03$	$\mu = 0.04$
Abalone	$0.3316_{\pm 0.1323}$	$\bullet\ 0.3594_{\pm 0.1372}$	$0.3378_{\pm 0.1140}$	$0.3129_{\pm 0.1322}$
Bank32nh	$-0.1580_{\pm 0.1305}$	$\bullet -0.1434_{\pm 0.1250}$	$-0.1499_{\pm 0.1228}$	$-0.1522_{\pm 0.1893}$
Cal_housing	$0.3859_{\pm 0.0032}$	$\bullet\ 0.3947_{\pm 0.0083}$	$0.3505_{\pm 0.0290}$	$0.3434_{\pm 0.0360}$
Cpu_act	$0.6418_{\pm 0.0385}$	$0.6153_{\pm 0.0107}$	$\bullet\ 0.6912_{\pm 0.0435}$	$0.6286_{\pm 0.0209}$
Cpu_small	$0.6854_{\pm 0.0162}$	$\bullet\ 0.7662_{\pm 0.0392}$	$0.6338_{\pm 0.0167}$	$0.6578_{\pm 0.0085}$
Elevators	$0.4682_{\pm 0.0308}$	$\bullet\ 0.4967_{\pm 0.0111}$	$0.4645_{\pm 0.0376}$	$0.4466_{\pm 0.0094}$
Folds5x2_pp	$0.8373_{\pm 0.0075}$	$0.8357_{\pm 0.0050}$	$\bullet\ 0.8444_{\pm 0.0012}$	$0.8411_{\pm 0.0111}$
Kin8nm	$0.2396_{\pm 0.0119}$	$\bullet\ 0.2796_{\pm 0.0231}$	$0.2716_{\pm 0.0017}$	$0.2221_{\pm 0.0056}$
Parkinsons	$0.3645_{\pm 0.1136}$	$0.3274_{\pm 0.1214}$	$0.3570_{\pm 0.1133}$	$\bullet\ 0.3805_{\pm 0.0933}$
Pollen	$-0.1424_{\pm 0.0059}$	$\bullet -0.1269_{\pm 0.0233}$	$-0.1454_{\pm 0.0604}$	$-0.1373_{\pm 0.0030}$
Puma8NH	$\bullet\ 0.3033_{\pm 0.1594}$	$0.2970_{\pm 0.1821}$	$0.2826_{\pm 0.1301}$	$0.2729_{\pm 0.1929}$
Space_ga	$0.4520_{\pm 0.0166}$	$\bullet\ 0.4484_{\pm 0.0165}$	$0.4048_{\pm 0.0124}$	$0.3926_{\pm 0.0333}$
Wind	$\bullet\ 0.4117_{\pm 0.0209}$	$0.4050_{\pm 0.0028}$	$0.3971_{\pm 0.0033}$	$0.4067_{\pm 0.0099}$
Wine_quality	$0.2114_{\pm 0.0204}$	$0.2163_{\pm 0.0460}$	$\bullet\ 0.2180_{\pm 0.0010}$	$0.1955_{\pm 0.0325}$

The best results obtained on each dataset are denoted with bullets

instances is then constructed by randomly selecting instances with a fixed size.

Table 9 shows the experimental results yielded by S2RMS under different μ values. The parameter μ indicates the degree of the predictive differences among multiple regressors. When μ is set to 0.01, only unlabeled instances with prediction variances of 0.009 or less are considered stable instances. Due to variations in the regression model settings, producing identical outputs for unlabeled data is essentially impossible. Lower parameter settings may not provide sufficient unlabeled data for sufficiently augmenting the regressors. Conversely, higher settings, such as 0.04, may result in the selection of the vast majority of the instances in the instance pool, which would interfere with the performance of the regressor due to the presence of widely varying labels. The experiments show that setting the unlabeled instance rate to 0.02 provides enough, but not all, instances to satisfy the stability constraint. These unlabeled instances can effectively enhance the training set of the corresponding regressor by labeling them with pseudo-labels.

5 Conclusions and further works

This paper proposes a semi-supervised regression approach via embedding space mapping and pseudo-label smearing. The method involves constructing a deep metric network based on pairwise data, mapping the raw training data to the metric embedding space through this network, and then training co-regressors on the transformed mapping data. The parameters of the regression model are iteratively updated using the label smearing technique. Tests conducted on 14 benchmark datasets show that the proposed algorithm outperforms 3 other excellent semi-supervised regression methods. An ablation study indicates that both the metric approach and the label smearing technique are effective for improving the performance of the algorithm.

Our future research will focus on designing more suitable mapping strategies and developing new confidence calculation methods. It is important to consider both the original spatial distribution of the given training data and the feature representations during the metric mapping process. Additionally, we will investigate improved confidence evaluation methods. Inspired by the evaluation metrics used in supervised learning research, novel reliability estimation strategies will be applied to increase the accuracy of confidence calculation [53]. Since the current study focuses on single-target regression, we will explore the potential to refine the proposed methods for applications involving multi-target regression tasks in our future research.

Acknowledgements This study is partially supported by the Nanchong Municipal Government-Universities Scientific Cooperation Project (Nos. 23XNSYSX0062, 23XNSYSX0013, and SXHZ051).

Author Contributions **Liyan Liu:** Methodology, Validation, Supervision, Writing - Review & Editing. **Jin Zhang:** Conceptualization, Methodology, Software, Visualization, Writing - Original Draft. **Kun Qian:** Supervision, Writing - Review & Editing. **Fan Min:** Methodology, Supervision, Writing - Review & Editing.

Declarations

Conflicts of interest The authors have no relevant financial or non-financial interests to disclose.

Ethical and informed consent for data used The data used in this study were obtained from publicly available datasets as described in Table 2.The original data collection was approved by the appropriate institutional ethical review board, and informed consent was obtained from all participants.

References

1. Chapelle O, Schlkopf B, Zien A (2009) Semi-supervised learning. IEEE Trans Neural Netw 20(3):542–542. https://doi.org/10.1109/TNN.2009.2015974
2. Goldman S, Zhou Y (2000) Enhancing supervised learning with unlabeled data. In: ICML, pp 327–334
3. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: COLT, pp 92–100. https://doi.org/10.1145/279943.279962
4. Gong M, Zhou H, Qin AK, Liu W, Zhao Z (2023) Self-paced co-training of graph neural networks for semi-supervised node classification. IEEE Trans Neural Netw Learn Syst 34(11):9234–9247. https://doi.org/10.1109/TNNLS.2022.3157688
5. Wei C, Sohn K, Mellina C, Yuille A, Yang F (2021) Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In: CVPR, pp 10852–10861. https://doi.org/10.1109/CVPR46437.2021.01071
6. Li J, Zhu Q (2020) A boosting self-training framework based on instance generation with natural neighbors for k nearest neighbor. Appl Intell 50(11):3535–3553. https://doi.org/10.1007/s10489-020-01732-1
7. Miyato T, Maeda S-I, Koyama M, Ishii S (2019) Virtual adversarial training: A regularization method for supervised and semi-supervised learning. IEEE Trans Pattern Anal Mach Intell 41(8):1979–1993. https://doi.org/10.1109/TPAMI.2018.2858821
8. Verma V, Kawaguchi K, Lamb A, Kannala J, Solin A, Bengio Y, Lopez-Paz D (2022) Interpolation consistency training for semi-supervised learning. Neural Netw 145:90–106. https://doi.org/10.1016/j.neunet.2021.10.008
9. Jiang Y, Li X, Che Y, He Y, Xu Q, Yang Z, Cao X, Huang Q (2023) Maxmatch: Semi-supervised learning with worst-case consistency. IEEE Trans Pattern Anal Mach Intell 45(5):5970–5987. https://doi.org/10.1109/TPAMI.2022.3208419
10. Duan Y, Zhao Z, Qi L, Wang L, Zhou L, Shi Y, Gao Y (2022) Mutexmatch: Semi-supervised learning with mutex-based consistency regularization. IEEE Trans Neural Netw Learn Syst 1–15. https://doi.org/10.1109/TNNLS.2022.3228380
11. Kostopoulos G, Karlos S, Kotsiantis S, Ragos O, Tiwari S, Trivedi M, Kohle ML (2018) Semi-supervised regression: A recent

review. J Intell Fuzzy Syst 35(2):1483–1500. https://doi.org/10.3233/JIFS-169689
12. Jing W, Lu L, Liu Q (2023) Graph regularized discriminative non-negative tucker decomposition for tensor data representation. Appl Intell 53(20):23864–23882. https://doi.org/10.1007/s10489-023-04738-7
13. Dong H, Yang L, Wang X (2021) Robust semi-supervised support vector machines with laplace kernel-induced correntropy loss functions. Appl Intell 51(2):819–833. https://doi.org/10.1007/s10489-020-01865-3
14. Balcan M-F, Blum A (2010) A discriminative model for semi-supervised learning. J ACM 57(3):1–46. https://doi.org/10.1145/1706591.1706599
15. Yang T, Priebe CE (2011) The effect of model misspecification on semi-supervised classification. IEEE Trans Pattern Anal Mach Intell 33(10):2093–2103. https://doi.org/10.1109/TPAMI.2011.45
16. Li Y-F, Tsang IW, Kwok JT, Zhou Z-H (2013) Convex and scalable weakly labeled SVMs. J Mach Learn Res 14(1):2151–2188
17. Li Y-F, Zha H-W, Zhou Z-H (2017) Learning safe prediction for semi-supervised regression. In: AAAI, pp 2217–2223
18. Ma F, Meng D, Xie Q, Li Z, Dong X (2017) Self-paced co-training. In: ICML, pp 2275–2284
19. Breiman L (2001) Random forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324
20. Song K, Han J, Cheng G, Lu J, Nie F (2022) Adaptive neighborhood metric learning. IEEE Trans Pattern Anal Mach Intell 44(9):4591–4604. https://doi.org/10.1109/TPAMI.2021.3073587
21. Milbich T, Roth K, Brattoli B, Ommer B (2022) Sharing matters for generalization in deep metric learning. IEEE Trans Pattern Anal Mach Intell 44(1):416–427. https://doi.org/10.1109/TPAMI.2020.3009620
22. Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R (1993) Signature verification using a "siamese" time delay neural network. Int J Pattern Recognit Artif Intell 7(4):669–688. https://doi.org/10.1142/S0218001493000339
23. Hoffer E, Ailon N (2015) Deep metric learning using triplet network. In: SIMBAD, pp 84–92. https://doi.org/10.1007/978-3-319-24261-3_7
24. Ge W, Huang W, Dong D, Scott MR (2018) Deep metric learning with hierarchical triplet loss. In: ECCV, pp 272–288. https://doi.org/10.1007/978-3-030-01231-1_17
25. Deng J, Guo J, Xue N, Zafeiriou S (2019) Arcface: Additive angular margin loss for deep face recognition. In: CVPR, pp 4685–4694. https://doi.org/10.1109/CVPR.2019.00482
26. Wang X, Hua Y, Kodirov E, Robertson NM (2022) Ranked list loss for deep metric learning. IEEE Trans Pattern Anal Mach Intell 44(9):5414–5429. https://doi.org/10.1109/TPAMI.2021.3068449
27. Yao X, She D, Zhang H, Yang J, Cheng M-M, Wang L (2021) Adaptive deep metric learning for affective image retrieval and classification. IEEE Trans Multimed 23:1640–1653. https://doi.org/10.1109/TMM.2020.3001527
28. Duan C, Feng Y, Zhou M, Xiong X, Wang Y, Qiang B, Jia W (2023) Multilevel similarity-aware deep metric learning for fine-grained image retrieval. IEEE Trans Ind Inform 19(8):9173–9182. https://doi.org/10.1109/TII.2022.3227721
29. Chang X, Ma Z, Wei X, Hong X, Gong Y (2020) Transductive semi-supervised metric learning for person re-identification. Pattern Recognit 108:107569. https://doi.org/10.1016/j.patcog.2020.107569
30. Uzun B, Cevikalp H, Saribas H (2023) Deep discriminative feature models (ddfms) for set based face recognition and distance metric learning. IEEE Trans Pattern Anal Mach Intell 45(5):5594–5608. https://doi.org/10.1109/TPAMI.2022.3205939
31. Zhou Z-H, Li M (2005) Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans Knowl Data Eng 17(11):1529–1541. https://doi.org/10.1109/TKDE.2005.186
32. Gong Y, Wu Q (2023) SIVLC: improving the performance of co-training by sufficient-irrelevant views and label consistency. Appl Intell 53:20710–20729. https://doi.org/10.1007/s10489-023-04611-7
33. Qiao S, Shen W, Zhang Z, Wang B, Alan Y (2018) Deep co-training for semi-supervised image recognition. In: ECCV, pp 142–159. https://doi.org/10.1007/978-3-030-01267-0_9
34. Chen D-D, Wang W, Gao W, Zhou Z-H (2018) Tri-net for semi-supervised deep learning. In: IJCAI, pp 2014–2020. https://doi.org/10.24963/ijcai.2018/278
35. Zhou Z-H, Li M (2007) Semisupervised regression with cotraining-style algorithms. IEEE Trans Knowl Data Eng 19(11):1479–1493. https://doi.org/10.1109/TKDE.2007.190644
36. Cover T, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27. https://doi.org/10.1109/TIT.1967.1053964
37. Hady MFA, Schwenker F, Palm G (2009) Semi-supervised learning for regression with co-training by committee. In: ICANN, pp 121–130. https://doi.org/10.1007/978-3-642-04274-4_13
38. Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1(2):281–294. https://doi.org/10.1162/neco.1989.1.2.281
39. Min F, Li Y, Liu L (2022) Self-paced safe co-training for regression. In: PAKDD, pp 71–82. https://doi.org/10.1007/978-3-031-05936-0_6
40. Lu S, Shi X, Li M, Jiao J, Feng L, Wang G (2021) Semi-supervised random forest regression model based on co-training and grouping with information entropy for evaluation of depression symptoms severity. Math Biosci Eng 18(4):4586–4602. https://doi.org/10.3934/mbe.2021233
41. Li H, Wang S, Liu B, Fang M, Cao R, He B, Liu S, Hu C, Dong D, Wang X, Wang H, Tian J (2023) A multi-view co-training network for semi-supervised medical image-based prognostic prediction. Neural Netw 164:455–463. https://doi.org/10.1016/j.neunet.2023.04.030
42. Shen Z, Cao P, Yang H, Liu X, Yang J, Zaiane OR (2023) Co-training with high-confidence pseudo labels for semi-supervised medical image segmentation. In: IJCAI, pp 4199–4207. https://doi.org/10.24963/ijcai.2023/467
43. Xie H, Fu C, Zheng X, Zheng Y, Sham C-W, Wang X (2023) Adversarial co-training for semantic segmentation over medical images. Comput Biol Med 157:106736. https://doi.org/10.1016/j.compbiomed.2023.106736
44. Li Q, Chen Y, He X, Huang L (2024) Co-training transformer for remote sensing image classification, segmentation, and detection. IEEE Trans Geosci Remote Sens 62:1–18. https://doi.org/10.1109/TGRS.2024.3354783
45. Gong Y, Wu Q, Zhou M, Wen J (2023) Self-paced multi-label co-training. Inf Sci 622:269–281. https://doi.org/10.1016/j.ins.2022.11.153
46. Liu Z, Ma Q, Ma P, Wang L (2023) Temporal-frequency co-training for time series semi-supervised learning. In: AAAI, pp 8923–8931. https://doi.org/10.1609/aaai.v37i7.26072
47. Jiang Z, Zhao L, Zhan Y (2023) A boosted co-training method for class-imbalanced learning. Expert Syst 40(9):13377. https://doi.org/10.1111/exsy.13377
48. Lu J, Gong Y (2021) A co-training method based on entropy and multi-criteria. Appl Intell 51(6):3212–3225. https://doi.org/10.1007/s10489-020-02014-6
49. Li Y, Xiong H, Wang Q, Kong L, Liu H, Li H, Bian J, Wang S, Chen G, Dou D, Yin D (2023) COLTR: Semi-supervised learning to rank with co-training and over-parameterization for web search. IEEE Trans Knowl Data Eng 35(12):12542–12555. https://doi.org/10.1109/TKDE.2023.3270750
50. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A,

Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS, pp 8026–8037
51. Khalid M, Baber J, Kasi MK, Bakhtyar M, Devi V, Sheikh N (2020) Empirical evaluation of activation functions in deep convolution neural network for facial expression recognition. In: TSP, pp 204–207. https://doi.org/10.1109/TSP49548.2020.9163446
52. Zell A, Sumbul G, Demir B (2022) Deep metric learning-based semi-supervised regression with alternate learning. In: ICIP, pp 2411–2415. https://doi.org/10.1109/ICIP46576.2022.9897939
53. Bosnić Z, Kononenko I (2008) Comparison of approaches for estimating reliability of individual regression predictions. Data Knowl Eng 67(3):504–516. https://doi.org/10.1016/j.datak.2008.08.001