Knowledge and Information Systems
https://doi.org/10.1007/s10115-023-02044-9

REGULAR PAPER

BSRU: boosting semi-supervised regressor through ramp-up unsupervised loss

Liyan Liu[1,2] · Haimin Zuo[1,2] · Fan Min[1,2,3]

Received: 5 May 2023 / Revised: 28 November 2023 / Accepted: 7 December 2023

Abstract
Semi-supervised regression aims to improve the performance of the learner with the help of unlabeled data. Popular approaches select some unlabeled data with high-quality pseudo labels to enrich the training set. In this paper, we propose a new approach with a semi-supervised regressor, a learner, and a respective loss function. First, an off-the-shelf semi-supervised regressor is trained to provide pseudo labels for all unlabeled data. These labels are often reliable enough to guide the learning process. Second, we design a neural network with dropout to train data with Gaussian noise added. In this way, the robustness of our learners is enhanced. Third, we design a weighted sum combining the supervised and unsupervised loss. The weight for pseudo-labels ramp-up over time, indicating more attention to the pseudo-labels. Six state-of-the-art algorithms are employed as the base model of our framework. Results on 15 real-world data sets show that our model has a significant improvement over the respective base regressor on most data sets.

Keywords Semi-supervised regression · Unsupervised loss · Ramp-up

1 Introduction

Semi-supervised regression aims to improve learning performance using unlabeled data without interactions, where predictions are real numbers [1, 2]. One of the critical issues is how

H. Zuo and F. Min contributed equally to this work.

✉ Fan Min
minfan@swpu.edu.cn

Liyan Liu
liuliyan@swpu.edu.cn

Haimin Zuo
zhm202121000473@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

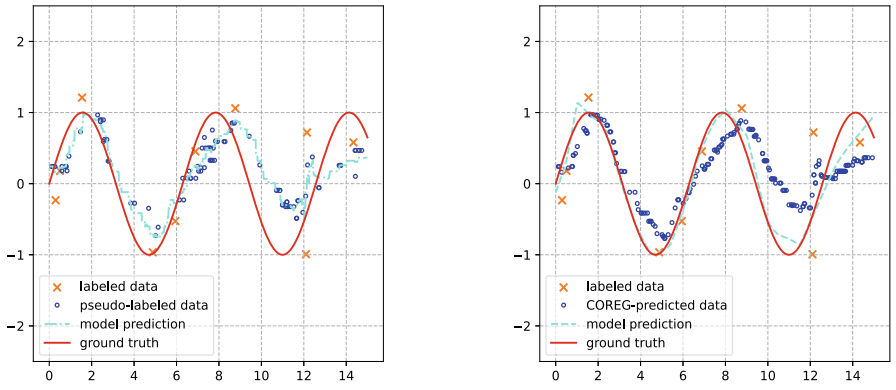
[3] Institute for Artificial Intelligence, Southwest Petroleum University, Chengdu 610500, China

Published online: 18 January 2024

to assign pseudo-labels for unlabeled data. The original co-training method, as described in [3], involves training two separate classifiers on two views that are both sufficient and redundant. In this method, the two attribute sets for the class label are conditionally independent of each other. One classifier assigns pseudo-labels to the other classifier on unlabeled data. However, it is hard to meet the requirement of both sufficient and redundant views in most scenarios. Therefore, many variants of the co-training algorithm [4–8] attempt to relax such a requirement. They deliberately build disagreement among diverse learners to assign pseudo-labels to one another in multiple iterations. Compared to graph [9–11] and kernel methods [12], co-training does not necessitate additional complex computations or a large memory requirement.

Despite the advantages of co-training in semi-supervised regression, there is still room for performance improvement. Specifically, they only analyze a small subset of the unlabeled examples to rapidly assign high-confidence pseudo-labels [5]. Most co-training style algorithms select high-confidence data from a pool that includes only a tiny portion of the unlabeled data since computing the confidence scores is time-consuming [4, 6]. In other words, the information contained in the unlabeled data is not fully leveraged. Indeed, COREG sets the pool size and the maximum iteration to 100, which means that at most 300 unlabeled examples will be analyzed (see Fig. 1a). Notably, this is only a tiny part of the unlabeled data. Moreover, the random selection of the pool from the unlabeled samples leads to unstable outcomes in repeated experiments.

In this paper, we design a neural network training scheme that utilizes both the labeled and the unlabeled data. This approach guarantees that all the labeled and unlabeled data are leveraged in the training process of the neural network. To ensure the quality of the pseudo-labels, an off-the-shelf semi-supervised regressor is employed to assign labels to all unlabeled samples. In this way, our algorithm can effectively learn from all unlabeled samples (see Fig. 1b), as opposed to co-training algorithms that can only learn from one sample at a



(a) COREG (b) BSRU based on COREG

Fig. 1 Illustrating the difference between COREG and BSRU for semi-supervised regression on the toy data set. The left (**a**) depicts a COREG with a maximum of 100 iterations and a fixed pool size of 100. This indicates that a maximum of 300 unlabeled examples are used to assign pseudo-labels. To simplify the presentation, only one set of pseudo-labels assigned by one of the regressors is shown in the (**a**). The right (**b**) depicts the BSRU approach, which uses COREG to assign pseudo-labels to all unlabeled examples. The incorporation of unlabeled data with pseudo-labels results in smoother fitted curves that closely approximate the true prediction results

time. In detail, our approach consists of three parts: a semi-supervised regression, a learner, and a respective loss function.

Firstly, the approach needs an off-the-shelf semi-supervised model to assign pseudo-labels to unlabeled data. This model is trained on both the labeled data set D_L and the unlabeled data set D_U, as it is a semi-supervised regressor. It is worth noting that the off-the-shelf regressor used in our work can be any type of semi-supervised regressor such as COREG. Our main focus is to enhance the performance of the base model, rather than designing the base model from scratch. In detail, it does not participate in updates during the entire training process. It only provides its predictions for unlabeled data with Gaussian noise added. In other words, it provides pseudo-labels of unlabeled data for training the neural network. These labels are usually reliable enough to guide the learning process even though they are noisy [13–15].

Secondly, we design a neural network with the dropout technique as the learner. Since there are limited labeled data in semi-supervised regression, overfitting is often observed when training neural networks. The adopted dropout technique can effectively prevent the neural network from overfitting. Furthermore, we add Gaussian noise to the features of both labeled and unlabeled data as a random perturbation during training. The addition of Gaussian noise enhances the training data set and makes the fitted model more robust.

Thirdly, we design a respective loss function, which consists of two terms: supervised loss and unsupervised loss. The supervised loss is computed between the neural network outputs and the true labels in the labeled data set. In contrast, the unsupervised loss is calculated between the neural network activations and the predictions made by the semi-supervised regressor. It is worth noting that the regressor has been trained using both labeled and unlabeled data. The weighted sum of the supervised loss and unsupervised loss gives the final total loss. It is worth noting that the weights of the unsupervised loss ramp-up from 0 to the maximum during training. In this way, we decrease the leverage of the unlabeled data with noisy pseudo-labels at the beginning of training. The loss gradient is calculated to update the neural network model. The predictions on test data are generated by the fitted network.

The experiments are conducted on 15 data sets commonly used in the regression task. Six state-of-the-art algorithms are used as the base model of our framework. The results show that our model has a significant improvement over the respective base regressor on most of the data sets. In addition, the ablation experiments on Gaussian noise and unsupervised loss show their effectiveness. The algorithm is available at https://github.com/zuohaimin/BSRU-Algorithm.

The rest of this paper is organized as follows: Section 2 briefly reviews related work of semi-supervised learning. Section 3 proposes the BSRU algorithm. Section 4 reports on the experiments. Finally, Sect. 5 concludes.

2 Related work

Semi-supervised learning is a subfield of machine learning that utilizes both labeled and unlabeled data to achieve specific learning objectives [2]. The primary challenge in semi-supervised learning lies in effectively leveraging the limited amount of labeled data in conjunction with the abundant unlabeled data. In this section, we will provide an overview of common approaches used in semi-supervised learning, as well as state-of-the-art methods employed in semi-supervised regression.

2.1 Semi-supervised learning

Both labeled and unlabeled data are used for training in semi-supervised learning [11, 16]. Semi-supervised learning is in strong real-world demand in areas such as medical image processing and analysis, where raw data are readily available but annotation is expensive [17–19].

Our approach is based on the concept of consistency regularization, which posits that the model's output should remain consistent even when the input undergoes reasonable perturbations [20–22]. The Π model [20] uses dropout [23] as the perturbation process and penalizes the difference between the two outputs of the network with different dropout and augmentation. Compared with Π model, virtual adversarial training [21] uses adversarial perturbations instead of independent noise. Mean teacher [22] speeds up network convergence by considering a moving average of connection weights instead of network outputs. On the basis of Mean teacher, dual student [24] replaces the teacher with another student to improve performance further. Both FixMatch [25] and FlexMatch [26] utilize class probability thresholding to assign pseudo-labels, which has led to state-of-the-art performance in semi-supervised classification. Similar techniques have been applied to video action recognition [27], image generation [28], medical image segmentation [29, 30], and other tasks [31, 32].

2.2 Semi-supervised regression

The classification problem involves predicting the probabilities of different classes, whereas the regression problem involves predicting continuous real numbers in $\mathbb{R}$. They are fundamentally different from each other. A simple way to generate pseudo-labels in semi-supervised classification is to use thresholding functions to select high-probability class pseudo-labels [33]. However, since there is no equivalent to probability thresholding for real number predictions, it cannot be adapted to regression tasks [12]. Different formulations are used to evaluate the quality of pseudo-labels for regression tasks. COREG [4] evaluates the quality of the pseudo-label by calculating the mean squared error changes of predictions on labeled instances caused by adding the unlabeled instance. MSSRA [34] employs the range of predictions from three models on unlabeled data for evaluation.

Co-training is a semi-supervised approach to multi-view data prediction that was proposed by Blum and Mitchell [3]. It is based on three assumptions: the multi-view assumption, the compatibility assumption, and the independence assumption. The multi-view assumption is that each example can be described by two different types of information. The compatibility assumption suggests that each view of the data can be effectively used for classification purposes, while the independence assumption assumes that each view is conditionally independent given the class label. In this context, two classifiers trained separately on each view of labeled data augment each other's training set with the most confident predictions on unlabeled data. Semi-supervised regression algorithms in the co-training style can be broadly categorized into two types: multi-view Co-Regression and single-view Co-Regression.

In multi-view Co-Regression, examples in the data set naturally satisfy the multi-view assumption in co-training [3, 35, 36]. The coRLSR, a kernel regression algorithm designed a semi-parametric variant that greatly reduces the runtime for handling numerous unlabeled data [37]. The SS-SVR (Semi-Supervised Support Vector Regression) co-trains two SVRs in a semi-supervised setting [38]. It has been demonstrated to outperform the multiple statistical regression method. The co-training PLS (partial least squares) algorithm combines PLS regression with the co-training approach to iteratively augment the labeled data by

Table 1 Notations

Notation	Meaning
D_L	Labeled data set
D_U	Unlabeled data set
$\mathcal{L}_s$	Supervised loss
$\mathcal{L}_u$	Unsupervised loss
f_θ	Neural network with parameters θ
ξ	Perturbation on the network, e.g., dropout
$\hat{\mathbf{x}}_i$	Example $\mathbf{x}_i$ with Gaussian noise added
ϵ_j	Gaussian noise added to j-th dimension feature of example $\mathbf{x}_i$
d	Distance function
g	An off-the-shell semi-supervised regressor
$w(t)$	Ramp-up weighted function

incorporating the most confident examples from the unlabeled data [39]. The Online S^3VM (OS^3VM) extends support vector machine (SVM) to handle streaming data [40]. The algorithm proposed in [41] trains two SVR variants to assign confident pseudo-labels. Diversity between two learners is attained through the utilization of distinct subsets of labeled data.

In fact, the multi-view assumption can hardly be met in real-life applications [42]. Co-EM creates two views by manufacturing a feature split and operates on all unlabeled samples at each iteration [43]. It is further improved by replacing naive Bayes classifier with SVM [44]. The disagreement-based algorithm has become a vital learning paradigm in single-view Co-Regression [45]. Specifically, the disagreement between multiple learners trained on labeled data is used to assign pseudo-labels to unlabeled examples [4, 6]. COREG employs two k-nearest neighbor (kNN) regressors with different distance metrics and k values to assign the most confident prediction to the other [4]. It is further enhanced with interactive genetic algorithms (IGAs) [46]. Democratic co-training is an extension of the co-training approach, which involves multiple models with varying architectures and learning algorithms rather than multiple views of the input data [47]. To be more precise, the models that have been trained on labeled data are utilized to assign labels to a specific unlabeled example by relying on the collective agreement of the majority of models, provided that their predictions are confident enough. CoBCReg [6] employs n RBF neural network regressors with different hyperparameters to extend single-view co-regression. MSSRA [34] constructs the disagreement between three supervised regression algorithms to guide the meta-learner in producing final predictions. SPOR [48] uses two neural networks with different structures to build disagreement to self-paced assign safe pseudo-labels for unlabeled examples.

3 The proposed model

This section describes the details of the BSRU algorithm. Table 1 lists the main notations used in the paper.

In the semi-supervised regression, we are presented with labeled data $D_L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and unlabeled data $D_U = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$. Each data point $(\mathbf{x}_i, y_i)$ in D_L consists of an object $\mathbf{x}_i \in \mathbb{R}^m$ from a given m-dimensional input space. To further clarify, let $\mathbf{x}_i = [x_{ij}]_{j=1}^{m}$, where

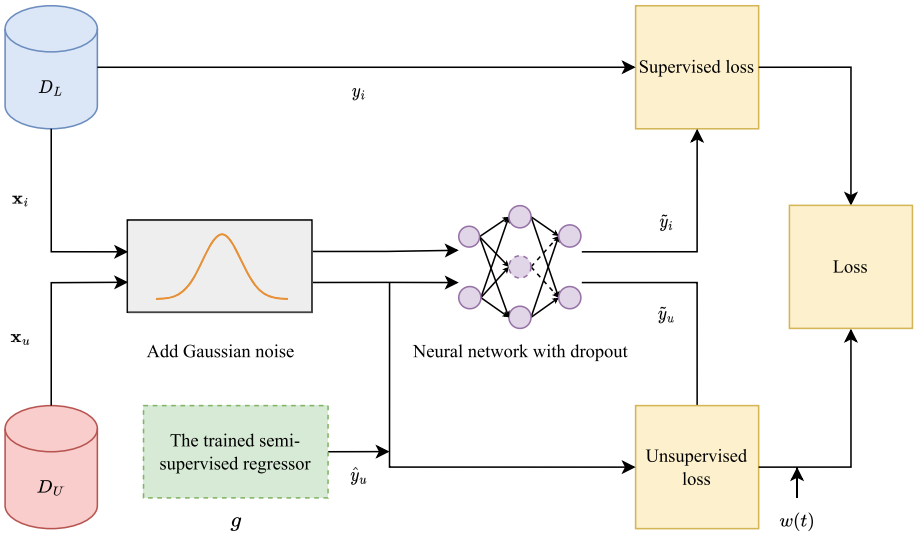


Fig. 2 y_i is the label associated with the example $\mathbf{x}_i$ in labeled data set D_L, while $\mathbf{x}_u$ is an unlabeled example in unlabeled data set D_U. $\widetilde{y}_i$ and $\widetilde{y}_u$ represent the predictions of the network for instances $\mathbf{x}_i$ and $\mathbf{x}_u$, respectively. $\hat{y}_u$ is the pseudo-label given to instance $\mathbf{x}_u$ by the semi-supervised regressor g trained on both D_L and D_U. The unsupervised loss measures the distance between $\hat{y}_u$ and $\widetilde{y}_u$, whereas the supervised loss quantifies the difference between y_i and $\widetilde{y}_i$. The final total loss is computed as a weighted sum of the supervised loss and the unsupervised loss. As the training epoch progresses, the weight ($w(t)$) assigned to the unsupervised loss gradually ramps up. The final total loss gradient updates the network in reverse until the network converges

x_{ij} represents the value of the j-th feature for the given data point $\mathbf{x}_i$. Each $\mathbf{x}_i$ is associated with a label y_i which is real-valued in regression problems. Additionally, we have access to a set of unlabeled data D_U, whose size is u. It is worth noting that the amount of unlabeled data far surpasses the size of labeled data, i.e., $u \gg l$.

Figure 2 demonstrates the main structure of the BSRU algorithm. The final loss consists of two terms. The first term is the supervised loss, which is only computed on labeled data. Compared to the classical supervised loss, it incorporates Gaussian noise into the training examples to smooth the output and prevent overfitting of the network. The second term refers to the unsupervised loss, which serves as a regularization term specifically designed for unlabeled data. It encourages consistency between the output of the network on unlabeled examples and the predictions made by well-trained semi-supervised regressors.

As the predictions of semi-supervised regressors on unlabeled data are prone to noise, the consistency may not be reliable during the early stages of training. To reduce unsupervised loss during the initial stages of training, we design a ramp-up weighting function to reweight the attention on the unsupervised loss. It begins at 0 and gradually ramps up to its maximum value, after which it remains constant for the rest of the training time. This method directs the attention of the network toward trustworthy labeled data during the initial training stages. As training progresses, the emphasis shifts toward unsupervised loss.

3.1 Loss function

This section describes the loss function of BSRU, which employs existing semi-supervised regression models to assign pseudo-labels to unlabeled data for training. In detail, it consists

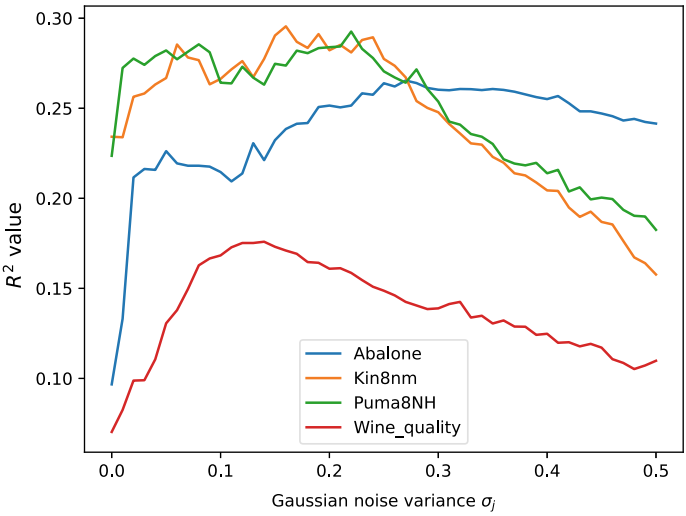


Fig. 3 Impact of Gaussian noise variance σ_j on R^2 values across different data sets. The R^2 value represents the performance of our algorithm on the validation set, with higher values indicating better performance

of two loss terms: a supervised loss $\mathcal{L}_s$ applied to labeled data and an unsupervised loss $\mathcal{L}_u$ applied to unlabeled data.

Specifically, the loss on labeled examples is given by:

$$\mathcal{L}_s = \frac{1}{l}\sum_{i=1}^{l} d(f_\theta(\hat{\mathbf{x}}_i;\xi), y_i), \tag{1}$$

where

- the example $\hat{\mathbf{x}}_i = [x_{ij} + \epsilon_j]_{j=1}^m$ involves adding Gaussian noise to each dimension of the feature vector $\mathbf{x}_i$. Specifically, Gaussian noise $\epsilon_j \sim \mathcal{N}(0, \sigma_j^2)$ added to each dimension feature is independent. Figure 3 shows that the best results are consistently found around $\sigma_j = 0.2$ in each data set. Consequently, σ_j is set to a constant value 0.2 in the experiment.
- $f_\theta(\hat{\mathbf{x}}_i;\xi)$ represents the output of a child mode spawned from the parent model f_θ via ξ perturbation, i.e., the dropout technique [23]. It is worth noting that, as a result of utilizing the dropout technique, the output of the network during training is a stochastic variable. Therefore, two evaluations of the same input $\hat{\mathbf{x}}_i$ under the same network weights θ can produce different results.
- $d(\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a nonnegative function that measures the distance between the activations of the network and the true labels. It will be further elaborated in Sect. 3.3.

According to smooth assumption [2], if two examples are close in the input space, their labels should be also close. Given that $\hat{\mathbf{x}}$ is the example with slight Gaussian noise added, and thus, minimizing this distance is a reasonable goal.

BSRU computes a pseudo-label for each unlabeled example, which is subsequently utilized to calculate the unsupervised loss. To acquire pseudo-labels, we initially train an off-the-shelf semi-supervised regressor g on $D_L \cup D_U$. Then, we employ the trained regressor to assign pseudo-labels to all instances with Gaussian noise in D_U. The unsupervised loss $\mathcal{L}_u$ is defined

as

$$\mathcal{L}_u = \frac{1}{u} \sum_{i=l+1}^{l+u} d(f_\theta(\hat{\mathbf{x}}_i; \xi), g(\hat{\mathbf{x}}_i)), \tag{2}$$

where $g(\hat{\mathbf{x}}_i)$ indicates the prediction of trained semi-supervised regressor to the example $\hat{\mathbf{x}}_i$. This distance can be regarded as a loss of unlabeled data, which is referred to as unsupervised loss. In other words, the training of the network is directed by the trained regressor that utilizes all the available unlabeled data. In addition, both Gaussian noise and dropout technique enhance the robustness of the network.

The loss minimized by BSRU is the weighted sum of both the supervised loss $\mathcal{L}_s$ and unsupervised loss $\mathcal{L}_u$. It is given by:

$$\mathcal{L} = \mathcal{L}_s + w(t)\mathcal{L}_u, \tag{3}$$

where $w(t)$ is a time-dependent weighting function. In detail, it is defined as:

$$w(t) = \begin{cases} \lambda_u \mathrm{e}^{-5\left(1-\frac{t}{t_m}\right)^2}, & t < t_m; \\ \lambda_u, & \text{otherwise}, \end{cases} \tag{4}$$

where $w(t)$ ramps up, starting from zero, along a Gaussian curve until it reaches the peak value of λ_u within the first t_m training epochs. Notably, the choice of the value -5 has almost no impact on the performance of our algorithm. It is set as a constant value, inspired by [20, 22]. Subsequently, it maintains a constant value of λ_u throughout the remaining epochs. It is worth noting that the cost coefficient of the unsupervised loss is always smaller than that of the supervised loss, which is due to the fact that the pseudo-labels in the unsupervised loss item are noisy.

There are at least two compelling reasons for opting for the ramp-up unsupervised loss approach:

(1) **Mitigating the Risk of Overfitting**: Incorporating unsupervised loss can serve as a form of regularization when working with noisy pseudo-labels. This effectively extends the effective size of the training set, helping to counteract the potential for overfitting.
(2) **Expediting Network Convergence**: The application of the ramp-up technique directs the network's focus toward accurately fitting the labeled data initially. As training progresses, the network gradually adjusts to accommodate the noisy pseudo-labels. This progressive transition prevents the network from being misled by the noisy labels during the initial stages of training. Therefore, it makes network convergence faster.

This trick of ramp-up weighting is similar to previous research [20, 22]. The distinction lies in the maximum value of the weighting function $w(t)$. In earlier studies, the maximum value assigned to $w(t)$ is often set considerably higher than 1 (e.g., 100), whereas the peak value λ_u in the BSRU approach is intentionally maintained below 1. In previous research, $w(t)$ represents the cost coefficient utilized for consistency regularization. This type of regularization quantifies the discrepancy between the output before and after applying perturbations. It proves valuable for establishing decision boundaries with large margins [18, 49]. Based on the smoothness assumption, consistency regularization operates without bias [25, 26]. Furthermore, employing a cost coefficient significantly greater than 1 serves to hinder the network from easily falling into a degenerate solution during the concluding stages of training. Unlike previous studies, the BSRU approach employs $w(t)$ as the cost coefficient applied to $\mathcal{L}_u$ the gap between the network's predictions and the pseudo-labels. Due to the inherent bias present in the pseudo-labels, the maximum value λ_u is deliberately set to be less than

1. This adjustment signifies that the emphasis placed on the unsupervised loss $\mathcal{L}_u$ does not exceed that of the supervised loss $\mathcal{L}_s$.

3.2 Network design

Figure 4 shows the architecture of the network. A fully connected network with three hidden layers is used as the base learner. In the hidden layers, there are 32, 256, and 32 neural units contained in each layer, from front to back. The input layer comprises m neural units, with each unit representing a unique dimensional feature of the input example. The output layer contains only a single unit. It is noteworthy that the design of the network is a balance between effectiveness and efficiency. For complex networks with more parameters, the improvement is not significant while the training time increases substantially. Moreover, it is important to note that the data sets used in the experiment are tabular data, which are considerably smaller than image data. Therefore, there is no need to design an overly complex network.

In this paper, we consider only the single-output dimension regression problem. Additionally, if the multi-output dimensions are independent of each other, we can utilize a series of models to predict each dimension's output separately. In the second hidden layer, the dropout technique is additionally used. Notably, the performance of our algorithm is not sensitive to which layer the dropout technique is located. The placement of the dropout layer in the second layer is simply an empirical design. This dropout probability is optimized through grid search on the validation sets. When it is set to 0.2, which means each unit in this layer has a 20% chance of being selected for dropping, our algorithm achieves the best results in the majority of data sets.

3.3 Algorithm description

Algorithm 1 summarizes the optimization process. The first step is to train the semi-supervised regressor, as it then participates in computing the unsupervised loss in Eq. (2). Line 1 trains the off-the-shelf semi-supervised regressor g using both labeled and unlabeled data. In line 2, D_L and D_U are utilized to create training batches consisting of a predetermined quantity of labeled and unlabeled data. We define an epoch as one single iteration over all unlabeled examples. As the quantity of unlabeled examples is significantly larger than that of labeled examples, each labeled example is encompassed multiple times within one epoch. In this way, unlabeled data is fully used by the neural network in each minibatch. Lines 3–9 are the main neural network training schema. In this way, the unsupervised loss is able to leverage the unlabeled data effectively.

We trained all networks for 1000 epochs with a batch size of 256. The value of λ_u is set to 0.8, and the value of t_m is set to 800, which corresponds to 80 percent of the total number of training epochs. Specifically, for each minibatch, there are 50 labeled data and the rest are unlabeled data. Stochastic gradient descent (SGD) is utilized to train the network with an initial learning rate of 0.05 and a Nesterov momentum of 0.9. All the above hyperparameters are optimized through grid search on the validation sets. The choice of these hyperparameters is not sensitive to different data sets. Therefore, the grid search is carried out on some of the data sets, rather than all of them for efficiency. Moreover, we do not tune hyperparameters for specific data sets. All experiments on different data sets are performed using the same set of hyperparameters. We do not use any form of early stopping. In other words, the network stops training until it reaches the maximum number of iterations.

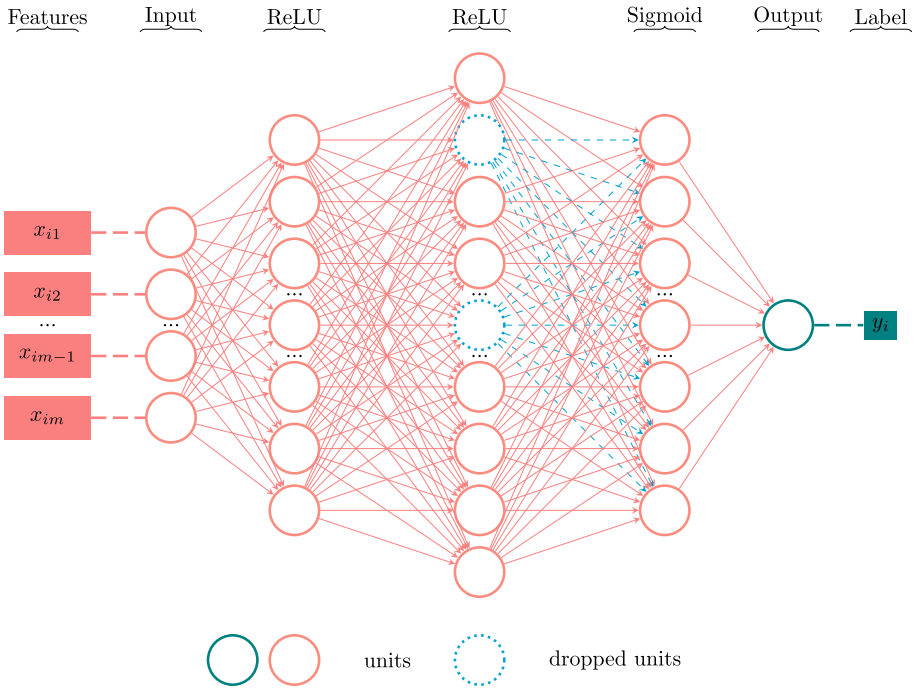


Fig. 4 The architecture of the neural network with dropout. Dashed nodes indicate dropped units. They are randomly selected from their respective layers with a fixed probability

Algorithm 1 BSRU algorithm

Input: Labeled data set $D_L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data set $D_U = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$, an off-the-shelf semi-supervised regressor g.
Output: θ.
1: Train semi-supervised regressor g with both the labeled data from D_L and the unlabeled data from D_U
2: Construct batch **B** with D_L and D_U;
3: **for** t in [1, epochs] **do**
4: **for** each minibatch B **do**
5: Calculate supervised loss on labeled data according to Eq. (1);
6: Calculate unsupervised loss on unlabeled data according to Eq. (2);
7: Calculate weighted sum loss according to Eq. (3);
8: Update θ using SGD;
9: **end for**
10: **end for**
11: **return** θ;

In the paper, d represents the Huber loss function [50], which is given by:

$$d(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{if } |y - \hat{y}| < \delta; \\ \delta\left(|y - \hat{y}| - \frac{\delta}{2}\right), & \text{otherwise.} \end{cases} \tag{5}$$

The function increases quadratically when the distance between y and $\hat{y}$ is small, but exhibits linear growth for larger distances. Figure 5 illustrates how varying the value of δ affects the function value. The parameter δ determines the point at which the transition shifts from quadratic to linear behavior. As the value of δ increases, the Huber loss gradually becomes

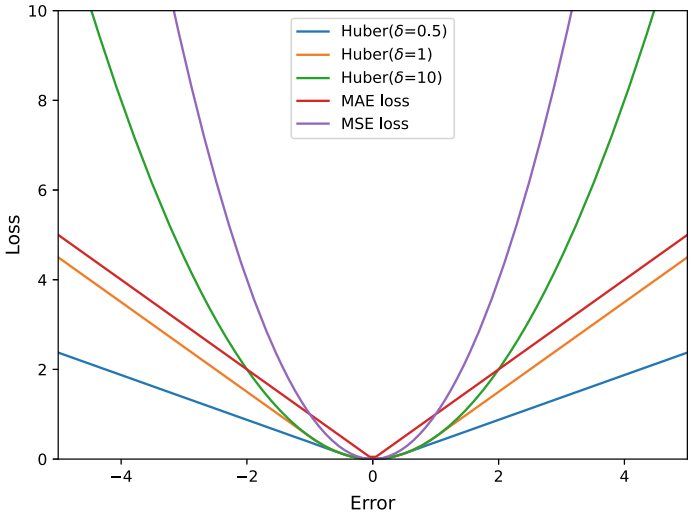


Fig. 5 The horizontal axis represents the difference between the predicted and actual results, while the vertical axis shows the calculated function value

more similar to the mean squared error (MSE) loss. On the other hand, for smaller values of δ, the Huber loss is more similar to the mean absolute error (MAE) loss.

Notably, the Huber loss is robust to outliers. In simpler terms, it is less influenced by extremely large residual values that may be identified as outliers. In greater detail, when certain sample points have outlier label values, the quadratic loss function is transformed into a linear loss function. This strategy effectively minimizes the impact of abnormal samples on the overall loss function, although it does not eliminate it completely. As a result, even when outliers are present, the curve that is fit to the data will still remain close to the majority of normal sample points. At the same time, the model will still take into account the presence of the outlier samples to some degree, ensuring that they are not entirely ignored. It is common practice to assign a fixed value to δ rather than estimate it from the data. In the experiment, the data used have been normalized to fall within the range of 0 to 1, and therefore, we have set δ equal to 0.5.

3.4 Approach comparison

Now, we present the similarities and differences between our algorithm and existing approaches. The main difference is that BSRU focuses on enhancing the performance of the existing semi-supervised regressor, rather than designing the base model from scratch. In fact, how to boost the performance of existing algorithms is a challenging issue [51–54]. Dissimilar to ensemble learning, which combines multiple weak learners to create a strong learner, BSRU enhances only a single weak learner at a time.

Furthermore, BSRU is somewhat similar to knowledge distillation [52–54]. Both methods involve leveraging predictions from a separate model to enhance the learning of the main model, but the underlying motivations and mechanisms differ. In knowledge distillation, knowledge is transferred from a complex model (teacher) to a simpler model (student) [54], while in BSRU, the predictions come from an independent regressor rather than a teacher–student relationship.

The idea of unsupervised loss is somewhat inspired by consistency regularization, which has blossomed in the field of semi-supervised learning [32, 55, 56]. Consistency regularization aims to ensure stability and robustness by maintaining output consistency before and after perturbations [31, 57, 58], while BSRU's unsupervised loss leverages the predictions of another regressor to enhance the model's performance. It is essential to emphasize that in consistency regularization, the predictions are all derived from the same model, whereas in BSRU's unsupervised loss, the predictions come from two distinct and independent models. This distinction is crucial as it highlights the fundamental difference in how the two techniques utilize predictions to enhance the learning process in semi-supervised scenarios. In addition, BSRU also combines two valuable perturbation techniques, Gaussian noise and dropout, which are widely used in consistency regularization.

Indeed, the utilization of pseudo-labels can either enhance or decrease the performance of an algorithm. Hence the design of the algorithm is essential. Among these aspects, a central challenge lies in the trade-off between the quality and quantity of pseudo-labels. In more detail, cotraining-style algorithms [4, 6, 8] in semi-supervised regression concentrate on guaranteeing the quality of the pseudo-labels. These algorithms typically opt for their most confident pseudo-labels, which correspond to a small subset of unlabeled examples. In other words, they sacrifice the quantity of pseudo-labels for quality. In contrast to the approach described above, all pseudo-labels in our algorithm actively contribute to the training process. The pseudo-labels assigned by a trained off-the-shelf model are frequently dependable enough to provide guidance during the learning process, despite their inherent noise. This signifies that our algorithm places a greater emphasis on the quantity of pseudo-labels rather than solely prioritizing their quality.

4 Experiments

In this section, we report experimental results on 15 real-world data sets to analyze the effectiveness of the BSRU algorithm. Through experiments, we aim to answer:

(1) Can BSRU enhance its base regressor?
(2) How do examples with Gaussian noise improve the robustness of our model?
(3) Can the unsupervised loss in Eq. (2) leverage the unlabeled data to improve learning performance?

To answer the first question, we will compare the root-mean-squared error (RMSE) and coefficient of determination (R^2) values between BSRU and its related algorithms on benchmark datasets. For the second question, a comparison between running loss with and without Gaussian noise is plotted. For the third question, we design an ablation experiment for the unsupervised loss. Briefly, we compare the RMSE between the model with and without unsupervised loss to evaluate its effectiveness.

4.1 Experimental setting

Table 2 lists 15 publicly available regression data sets from different domains used for our experiments. Among them, 6 data sets (Abalone, Elevators, Parkinsons, Puma8NH, Wine_quality, Folds5x2_pp) come from the UCI machine learning data repository, 5 data sets (Bank8FM, Bank32NH, Cpu_act, Cpu_small, Kin8nm) come from the Delve data repository, and 4 data sets (Cal_housing, Pollen, Space_ga, Wind) come from the StatLib data repository. In detail, the data sets are mainly from three domains: life (e.g., Abalone), physics (e.g.,

Table 2 Datasets

Dataset	Instances	Attributes	Source	Area
Abalone	4177	8	UCI	Physical
Bank8FM	8192	9	Delve	Business
Bank32NH	8192	32	Delve	Business
Cal_housing	20, 460	8	StatLib	Life
Cpu_act	8192	21	Delve	Business
Cpu_small	8192	12	Delve	Business
Elevators	9517	7	UCI	Physical
Folds5x2_pp	9565	4	UCI	Engineering
Kin8nm	8192	8	Delve	Physical
Parkinsons	5875	21	UCI	Biomedical
Pollen	3848	5	StatLib	biography
Puma8NH	8192	8	UCI	Physical
Space_ga	3107	6	StatLib	Business
Wind	6574	14	StatLib	Life
Wine_quality	6497	11	UCI	Business

Elevators), and business (e.g., Cal_housing). The sample size ranges from around 3,107 (Space_ga) to 20,460 (Cal_housing). The feature size ranges from 4 (e.g., Folds5x2_pp) to 32 (e.g., Bank32NH).

For each data set, we split the train and test set with fivefold cross-validation. In some applications, there might be a scarcity of labeled data, whereas in others, there could be an abundance of labeled data accessible. To better simulate real-life scenarios, a part of the training set is designated as labeled data set D_L, while the remaining part is treated as unlabeled data set D_U. We aim to test BSRU under various scenarios. In different experimental setups, we considered different sizes for $|D_L|$, with candidates being 50, 100, 200, and 400, indicating the selection of 50, 100, 200, and 400 samples from the training set as labeled data. To minimize statistical variability, each algorithm underwent five runs using fivefold cross-validation. The ultimate evaluation result is the average of these five sets of fivefold cross-validation outcomes.

We choose RMSE and R^2 to evaluate the effectiveness of the algorithm. RMSE is a quadratic scoring rule that measures the average magnitude of the error. It can be described as the square root of the mean of the squared difference between the predicted and actual values. The RMSE score is given by:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{6}$$

where y_i is the real value, while $\hat{y}_i$ is the prediction.

R^2 is a statistical measure that indicates how well the regression model fits the observed data. Specifically, it represents the proportion of the explained variance in the regression relationship to the total variance. The formula for calculating R^2 is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}, \tag{7}$$

where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the mean value of the label. Its value ranges from 0 to 1, with higher values indicating a better fit between the model and the data.

BSRU is compared with the following methods. The parameter settings are also provided as follows.

- **Self-kNN**: Semi-supervised extension of the supervised kNN method based on self-training [59]. It trains a kNN [60] regressor using only labeled data and then predicts pseudo-labels for unlabeled instances. After that, it trains on both labeled data and unlabeled data labeled with its own pseudo-labels. This process is repeated iteratively until either the predictions on the unlabeled data no longer change or the maximum number of iterations is reached. Here we set $k = 3$, and therefore, it is also referred to as Self-3NN.
- **COREG**[1] [4]: The well-known algorithm in co-training style for regression. Two 3NN regressors are used to assign pseudo-labels to each other. Here we set the distance metrics as Euclidean and Mahalanobis, respectively. The final predictions are the average of the two regressors' outputs.
- **SAFER**[2] [61]: A safe regression technique with multiple regressors. It employs the supervised 1NN as the baseline for the safety label. Furthermore, two Self-3NN regressions with different distance metrics are integrated.
- **MSSRA**[3] [34]: A multi-scheme semi-supervised regression approach. Three regressors are used to feed their decisions to the meta-learner to produce final predictions. Here, we set three regressors as SMOReg [62], Random Forest [63] and M5 [64], respectively. One notable aspect of this approach is its utilization of the M5 regressor, which relies on model trees as a meta-learner.
- **BHD**[4] [65]: A graph-based semi-supervised regression algorithm. Its main idea is based on a physical heat diffusion system with boundary conditions.
- **S3VR** [66]: A semi-supervised regression method based on SVR. It expands SVR to accommodate a semi-supervised setting.

Notably, **Self-kNN**, **COREG**, **SAFER** all use kNN as their base models. There are at least two reasons for choosing kNN as the base model:

(1) In the context of semi-supervised learning, regressors undergo refinement through multiple learning iterations. The kNN method is classified as a lazy learning approach without a separate training phase. It enables more efficient refinement in contrast to regressors such as neural networks, which necessitate a separate training phase.
(2) Estimating label confidence is essential when selecting appropriate unlabeled examples for labeling in semi-supervised learning. Based on the manifold assumption of local smoothness in regression problems, kNN can easily obtain label confidence by calculating the variance of its k-th nearest labeled example.

The choice of setting $k = 3$ results from a hyperparameter optimization process. During this optimization, **COREG** systematically tested different values of k and found that $k = 3$ yielded the best performance in the experiment [4].

For **COREG**, **SAFER**, **MSSRA**, **BHD**, we use the best experimental settings in their papers to ensure the best performance. All experiments were performed in Python 3.9.

[1] http://www.lamda.nju.edu.cn/code_COREG.ashx.

[2] http://www.lamda.nju.edu.cn/code_SAFER.ashx.

[3] http://ml.upatras.gr/mssregression.

[4] https://github.com/timilsinamohan/SSR.

4.2 Comparison with corresponding semi-supervised methods

To answer the first question, we compare BSRU with their respective base models. Six state-of-the-art algorithms are employed as the base model of our framework. Each data set underwent evaluation under various label sizes through five times of fivefold cross-validation. Table 3 showcases the RMSE outcomes for various algorithms under four different label sizes, while Table 4 illustrates the R^2 results of these algorithms for the same four label sizes. We employ an incremental labeling strategy, i.e., the larger training sets contain all the data from the smaller training sets. It can be observed that BSRU has superior performance compared to their corresponding algorithms, as follows:

(1) BSRU has a significant improvement over its base model. Across most data sets, BSRU outperforms its base model in terms of RMSE under four distinct label sizes: 50, 100, 200, and 400. Additionally, while the R^2 value of BSRU may be negative, it is generally greater than that of its base model in most data sets. Moreover, the average RMSE and R^2 of BSRU are always better than its base model. In detail, when Self-3NN serves as the base model, BSRU achieved better performance on 13 of 15 data sets under 200 and 400 label sizes.
(2) As the label size increases, BSRU achieves more performance gains than other semi-supervised regression algorithms. In Abalone, BSRU based on BHD significantly outperforms the other six semi-supervised algorithms at the same label size. Moreover, the RMSE average of BSRU based on SAFER under 100 label size approached or exceeded Self-3NN, MSSRA, BHD, and S3VR under 400 label size. It can be seen from this that BSRU can perform well when the label size is small.
(3) BSRU has a better performance when served with a better base model. SAFER is ahead of the other five existing semi-supervised regression algorithms on RMSE average under the same label size. BSRU based on SAFER has the best performance on all the listed algorithms under four label sizes. However, the enhancement of BSRU remains constrained in some cases. It is notable that BSRU based on MSSRA performs nearly equivalently under the 200 label size compared to the 400 label size, despite serving more label information.

Here we observe that BSRU appears to be slightly less effective than the corresponding methods on certain data sets. For example, in the Folds5x2_pp data set with a 4-dimensional attribute space, BSRU failed to improve the performance of Self-3NN, COREG, SAFER, and MSSRA under four different label sizes. Meanwhile, in the Pollen data set with a 5-dimensional attribute space, BSRU shows slightly higher error rates compared to its base algorithm under four different label sizes. There may be at least the following three reasons.

(1) The limited attribute dimension may constrain the learning ability of the neural network. In other words, the neural network excels in handling high-dimensional attribute data.
(2) To ensure generalization performance, the hyperparameters of BSRU are not individually optimized for each data set. In simple terms, the hyperparameters used in the experiments may not be the optimal settings for each particular data set.
(3) BSRU does not employ an early stopping strategy, which can potentially result in underfitting or overfitting on certain data sets.

4.3 Ablation study

Two ablation experiments are designed to answer the second and third questions.

Table 3 The RMSE values for the compared methods and BSRU under 50, 100, 200, and 400 label sizes using five times fivefold cross-validation

$\|D_L\|$	Dataset	Self-3NN		COREG		SAFER		MSSRA		BHD		S3VR	
		Self-3NN	BSRU	COREG	BSRU	SAFER	BSRU	MSSRA	BSRU	BHD	BSRU	S3VR	BSRU
50	Abalone	0.1066	0.0996•	0.0983	0.0973•	0.1077	0.1025•	0.1128	0.1093•	0.1166	0.0923•	0.1469	0.1207•
	Bank32NH	0.1676	0.1645•	0.1647	0.1560•	0.1661	0.1596•	0.1363	0.1270•	0.1546	0.1507•	0.1728	0.1708•
	Bank8FM	0.1969	0.1761•	0.1712	0.1356•	0.1955	0.1598•	0.0568	0.0492•	0.1894	0.1468•	0.1138	0.0888•
	Cal_housing	0.2034•	0.2035	0.2276	0.2126•	0.1973	0.1907•	1.2480	0.3832•	0.2573	0.2475•	0.2144•	0.2261
	Cpu_act	0.1292	0.1253•	0.1382•	0.1426	0.1275•	0.1449	0.1659	0.1499•	0.1854	0.1763•	0.1483	0.1468•
	Cpu_small	0.1303	0.1264•	0.1350•	0.1403	0.1275•	0.1342	0.1451	0.1133•	0.1855	0.1757•	0.1473	0.1464•
	Elevators	0.0626•	0.0745	0.0636•	0.0642	0.0628	0.0595•	0.0579•	0.0646	0.0885	0.0697•	0.0722	0.0635•
	Folds5x2_pp	0.0771•	0.1016	0.0815•	0.0997	0.0769•	0.0777	0.0622•	0.1113	0.2278	0.1473•	0.0914•	0.1025
	Kin8nm	0.1535	0.1518•	0.1494	0.1489•	0.1522	0.1471•	0.1603	0.1517•	0.1867	0.1709•	0.2358	0.2019•
	Parkinsons	0.1100	0.1031•	0.1122•	0.1126	0.1104	0.1035•	0.0932	0.0841•	0.1308	0.1262•	0.1206	0.1173•
	Pollen	0.5707•	0.5715	0.5707•	0.5715	0.5707•	0.5716	0.5705•	0.5714	0.5754	0.5715•	0.5704•	0.5714
	Puma8NH	0.2117	0.2022•	0.2148	0.2038•	0.2122	0.2065•	0.1903	0.1836•	0.2321	0.2202•	0.3148	0.2706•
	Space_ga	0.3113	0.3066•	0.3102	0.3081•	0.3114	0.3066•	0.3131	0.3075•	0.3085	0.3045•	0.3076	0.3063•
	Wind	0.1367•	0.1448	0.1519•	0.1595	0.1370•	0.1374	0.1344•	0.1472	0.1609	0.1582•	0.1350•	0.1487
	Wine_quality	0.1660	0.1595•	0.1666	0.1604•	0.1665	0.1556•	0.1513	0.1432•	0.1735	0.1692•	0.1926	0.1896•
	Avg	0.1822	0.1807•	0.1837	0.1809•	0.1814	0.1772•	0.2399	0.1798•	0.2115	0.1951•	0.1989	0.1914•
100	Abalone	0.1072	0.1015•	0.0989	0.0960•	0.1065	0.1029•	0.1088	0.1044•	0.1181	0.0947•	0.1520	0.1237•
	Bank32NH	0.1541	0.1466•	0.1508	0.1424•	0.1533	0.1399•	0.1261	0.1209•	0.1485	0.1435•	0.1477	0.1451•
	Bank8FM	0.1958	0.1439•	0.1584	0.1133•	0.1935	0.1147•	0.0572	0.0483•	0.1887	0.1316•	0.1043	0.0730•
	Cal_housing	0.1845	0.1843•	0.2191	0.2151•	0.1815	0.1757•	1.4480	0.3218•	0.2470	0.2369•	0.2008•	0.2020
	Cpu_act	0.0800	0.0691•	0.1067	0.0949•	0.0790	0.0759•	0.1482	0.0995•	0.1858	0.1743•	0.1425	0.1389•
	Cpu_small	0.0767	0.0676•	0.1043	0.0943•	0.0755	0.0669•	0.1448	0.0922•	0.1858	0.1740•	0.1432	0.1396•
	Elevators	0.0615•	0.0618	0.0622	0.0618•	0.0613	0.0582•	0.0577•	0.0621	0.0881	0.0663•	0.0640•	0.0648

Table 3 continued

$\|D_L\|$	Dataset	Self-3NN		COREG		SAFER		MSSRA		BHD		S3VR	
		Self-3NN	BSRU	COREG	BSRU	SAFER	BSRU	MSSRA	BSRU	BHD	BSRU	S3VR	BSRU
	Folds5x2_pp	0.0751•	0.0908	0.0768•	0.0909	0.0748•	0.0763	0.0600•	0.0981	0.2288	0.1346•	0.0747•	0.1078
	Kin8nm	0.1431	0.1369•	0.1413	0.1401•	0.1421	0.1335•	0.1514	0.1409•	0.1863	0.1558•	0.1938	0.1715•
	Parkinsons	0.1090	0.1028•	0.1092•	0.1124	0.1098	0.1036•	0.0859•	0.0877	0.1317	0.1275•	0.1154	0.1128•
	Pollen	0.5632•	0.5638	0.5632•	0.5639	0.5632•	0.5641	0.5633•	0.5638	0.5653	0.5640•	0.5634•	0.5639
	Puma8NH	0.2048	0.1884•	0.2073	0.1912•	0.2051	0.1830•	0.1802	0.1732•	0.2323	0.2090•	0.2899	0.2421•
	Space_ga	0.2848	0.2786•	0.2783	0.2772•	0.2849	0.2781•	0.2862	0.2771•	0.2818	0.2793•	0.3043	0.2838•
	Wind	0.1410•	0.1437	0.1648•	0.1736	0.1404	0.1353•	0.1275•	0.1343	0.1643	0.1598•	0.1755	0.1690•
	Wine_quality	0.1605	0.1521•	0.1619	0.1457•	0.1601	0.1533•	0.1442	0.1388•	0.1729	0.1661•	0.1953•	0.1976
	Avg	0.1694	0.1621•	0.1736	0.1675•	0.1687	0.1574•	0.2460	0.1642•	0.2084	0.1878•	0.1911	0.1824•
200	Abalone	0.1095	0.1005•	0.0977	0.0956•	0.1080	0.1012•	0.1026	0.0975•	0.1184	0.0918•	0.1181	0.1092•
	Bank32NH	0.1508	0.1396•	0.1469	0.1341•	0.1503	0.1320•	0.1190	0.1118•	0.1484	0.1382•	0.1377	0.1302•
	Bank8FM	0.1908	0.1069•	0.1380	0.0799•	0.1870	0.0931•	0.0509	0.0452•	0.1892	0.1036•	0.0959	0.0586•
	Cal_housing	0.1806	0.1737•	0.1988•	0.2134	0.1816	0.1678•	0.5618	0.2970•	0.2429	0.2305•	0.2878•	0.2933
	Cpu_act	0.0524	0.0400•	0.0776	0.0627•	0.0525	0.0474•	0.0848	0.0514•	0.1861	0.1708•	0.1461	0.1334•
	Cpu_small	0.0498	0.0399•	0.0642	0.0497•	0.0496	0.0445•	0.0869	0.0563•	0.1862	0.1716•	0.1402	0.1263•
	Elevators	0.0605	0.0586•	0.0613	0.0592•	0.0604	0.0569•	0.0562•	0.0595	0.0882	0.0654•	0.0688	0.0642•
	Folds5x2_pp	0.0695•	0.0921	0.0705•	0.0928	0.0694•	0.0735	0.0588•	0.1001	0.2258	0.1209•	0.0697•	0.0997
	Kin8nm	0.1381	0.1185•	0.1326	0.1196•	0.1372	0.1201•	0.1471	0.1243•	0.1863	0.1325•	0.1520	0.1342•
	Parkinsons	0.1031	0.0947•	0.1047•	0.1110	0.1026	0.0962•	0.0777	0.0765•	0.1281	0.1154•	0.1092	0.0903•
	Pollen	0.5499	0.5493•	0.5499	0.5494•	0.5498	0.5495•	0.5493	0.5490•	0.5496	0.5494•	0.5495	0.5491•
	Puma8NH	0.1979	0.1710•	0.1991	0.1707•	0.1978	0.1605•	0.1650	0.1593•	0.2319	0.1712•	0.2520	0.2035•
	Space_ga	0.2529	0.2450•	0.2438	0.2393•	0.2521	0.2463•	0.2610	0.2536•	0.2342	0.2312•	0.2820	0.2735•
	Wind	0.1238•	0.1317	0.1471	0.1399•	0.1239	0.1230•	0.1186•	0.1303	0.1657	0.1590•	0.1490	0.1437•

Table 3 continued

$\|D_L\|$	Dataset	Self-3NN		COREG		SAFER		MSSRA		BHD		S3VR	
		Self-3NN	BSRU	COREG	BSRU	SAFER	BSRU	MSSRA	BSRU	BHD	BSRU	S3VR	BSRU
	Wine_quality	0.1585	0.1535•	0.1511•	0.1556	0.1585	0.1560•	0.1460	0.1440•	0.1656	0.1600•	0.2018	0.1992•
	Avg	0.1592	0.1477•	0.1589	0.1515•	0.1587	0.1445•	0.1724	0.1504•	0.2031	0.1741•	0.1840	0.1739•
400	Abalone	0.1204	0.1080•	0.1091	0.1039•	0.1186	0.1105•	0.1129	0.1054•	0.1301	0.1039•	0.1225	0.1104•
	Bank32NH	0.1505	0.1304•	0.1430	0.1242•	0.1501	0.1191•	0.1155	0.1078•	0.1484	0.1183•	0.1235	0.1146•
	Bank8FM	0.1885	0.0948•	0.1263	0.0651•	0.1842	0.0889•	0.0456	0.0413•	0.1884	0.0953•	0.0786	0.0469•
	Cal_housing	0.1928	0.1895•	0.2081•	0.2085	0.1902	0.1856•	0.6836	0.4108•	0.2408	0.2101•	0.8515	0.4092•
	Cpu_act	0.0466	0.0340•	0.0679	0.0504•	0.0467	0.0466•	0.0614	0.0550•	0.1860	0.1706•	0.1396	0.1251•
	Cpu_small	0.0444	0.0391•	0.0538	0.0409•	0.0441•	0.0442	0.0651	0.0534•	0.1860	0.1711•	0.1326	0.1250•
	Elevators	0.0602	0.0584•	0.0607	0.0587•	0.0602	0.0565•	0.0559•	0.0599	0.0879	0.0652•	0.0703	0.0643•
	Folds5x2_pp	0.0642•	0.0856	0.0645•	0.0844	0.0640•	0.0699	0.0573•	0.0960	0.2252	0.1193•	0.0635•	0.0938
	Kin8nm	0.1317	0.1040•	0.1233	0.1019•	0.1309	0.1112•	0.1427	0.1074•	0.1864	0.1304•	0.1237	0.1053•
	Parkinsons	0.1064	0.0985•	0.1078•	0.1090	0.1060	0.0981•	0.0699•	0.0719	0.1319	0.1187•	0.1004	0.0978•
	Pollen	0.5230	0.5209•	0.5229	0.5209•	0.5230	0.5213•	0.5223	0.5208•	0.5221	0.5208•	0.5221	0.5206•
	Puma8NH	0.1915	0.1542•	0.1900	0.1527•	0.1914	0.1482•	0.1497	0.1456•	0.2321	0.1626•	0.1970	0.1614•
	Space_ga	0.2713	0.2570•	0.2631	0.2536•	0.2709	0.2565•	0.2451•	0.2479	0.2673	0.2581•	0.2347•	0.2417
	Wind	0.1218•	0.1299	0.1368	0.1334•	0.1220	0.1191•	0.1162•	0.1287	0.1640	0.1566•	0.2878	0.2098•
	Wine_quality	0.1522	0.1470•	0.1393•	0.1438	0.1508	0.1464•	0.1441	0.1403•	0.1505	0.1435•	0.1922	0.1853•
	Avg	0.1577	0.1434•	0.1545	0.1434•	0.1569	0.1415•	0.1725	0.1528•	0.2032	0.1696•	0.2160	0.1741•

The best results for each data set are highlighted with bullet

Table 4 The R^2 values for the compared methods and BSRU under 50, 100, 200, and 400 label sizes using five times fivefold cross-validation

$\|D_L\|$	Dataset	Self-3NN		COREG		SAFER		MSSRA		BHD		S3VR	
		Self-3NN	BSRU	COREG	BSRU	SAFER	BSRU	MSSRA	BSRU	BHD	BSRU	S3VR	BSRU
50	Abalone	0.1365	0.2460•	0.2642	0.2818•	0.1177	0.2025•	0.0251	0.0827•	−0.0293	0.3542•	−0.6838	−0.1207•
	Bank32NH	−0.2758	−0.2300•	−0.2311	−0.1058•	−0.2526	−0.1575•	0.1537	0.2678•	−0.0845	−0.0315•	−0.3574	−0.3251•
	Bank8FM	−0.0820	0.1338•	0.1810	0.4863•	−0.0666	0.2871•	0.9094	0.9321•	−0.0015	0.3980•	0.6379	0.7794•
	Cal_housing	0.2650•	0.2645	0.0788	0.1982•	0.3080	0.3516•	−140.4628	−1.6992•	−0.1702	−0.0837•	0.1835•	0.0917
	Cpu_act	0.5032	0.5316•	0.4353•	0.4029	0.5166•	0.3865	0.1066	0.3445•	0.0019	0.1002•	0.3607	0.3740•
	Cpu_small	0.4953	0.5237•	0.4595•	0.4214	0.5161•	0.4674	0.3687	0.6203•	0.0016	0.1058•	0.3694	0.3777•
	Elevators	0.4933•	0.2760	0.4756•	0.4642	0.4901	0.5418•	0.5667•	0.4538	−0.0145	0.3704•	0.3230	0.4765•
	Folds5x2_pp	0.8834•	0.7930	0.8698•	0.7987	0.8842•	0.8798	0.9239•	0.7431	−0.0163	0.5729•	0.8360•	0.7818
	Kin8nm	0.3168	0.3313•	0.3527	0.3562•	0.3282	0.3715•	0.2542	0.3326•	−0.0103	0.1539•	−0.6195	−0.1885•
	Parkinsons	0.2689	0.3565•	0.2384•	0.2345	0.2632	0.3531•	0.4553	0.5713•	−0.0322	0.0389•	0.1171	0.1674•
	Pollen	−2.9130•	−2.9247	−2.9128•	−2.9243	−2.9130•	−2.9258	−2.9107•	−2.9236	−2.9782	−2.9244•	−2.9089•	−2.9232
	Puma8NH	0.1619	0.2356•	0.1372	0.2230•	0.1584	0.2027•	0.3205	0.3697•	−0.0061	0.0941•	−0.8682	−0.3742•
	Space_ga	−1.3353	−1.2736•	−1.3204	−1.2971•	−1.3372	−1.2742•	−1.3662	−1.2860•	−1.2941	−1.2441•	−1.2799	−1.2704•
	Wind	0.2681•	0.1752	0.0873•	0.0048	0.2654•	0.2577	0.2797•	0.1464	−0.0133	0.0205•	0.2790•	0.1268
	Wine_quality	−0.3051	−0.2066•	−0.3259	−0.2188•	−0.3151	−0.1496•	−0.0926	0.0304•	−0.4248	−0.3543•	−0.7843	−0.7203•
	Avg	−0.0746	−0.0512•	−0.0807	−0.0449•	−0.0691	−0.0137•	−9.3646	−0.0676•	−0.3381	−0.1619•	−0.2930	−0.1831•
100	Abalone	0.1248	0.2160•	0.2557	0.2993•	0.1360	0.1937•	0.0946	0.1659•	−0.0581	0.3198•	−0.8144	−0.1963•
	Bank32NH	−0.0769	0.0245•	−0.0308	0.0798•	−0.0651	0.1134•	0.2783	0.3369•	0.0007	0.0659•	0.0097	0.0443•
	Bank8FM	−0.0700	0.4219•	0.2993	0.6413•	−0.0457	0.6323•	0.9084	0.9346•	0.0055	0.5163•	0.6958	0.8509•
	Cal_housing	0.3982	0.3985•	0.1474	0.1819•	0.4173	0.4536•	−160.6705	−0.9604•	−0.0780	0.0075•	0.2871•	0.2769
	Cpu_act	0.7972	0.8393•	0.6649	0.7355•	0.8031	0.8171•	0.3122	0.6877•	−0.0021	0.1204•	0.4111	0.4406•
	Cpu_small	0.8087	0.8442•	0.6755	0.7406•	0.8163	0.8563•	0.3416	0.7502•	−0.0020	0.1234•	0.4046	0.4358•
	Elevators	0.5107•	0.5049	0.4984	0.5031•	0.5127	0.5612•	0.5685•	0.4992	−0.0050	0.4300•	0.4703•	0.4543

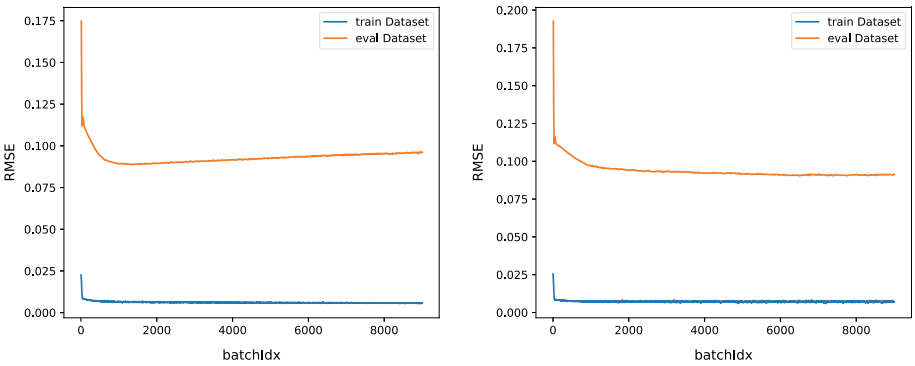
Table 4 continued

$\|D_L\|$	Dataset	Self-3NN		COREG		SAFER		MSSRA		BHD		S3VR	
		Self-3NN	BSRU	COREG	BSRU	SAFER	BSRU	MSSRA	BSRU	BHD	BSRU	S3VR	BSRU
	Folds5x2_pp	0.8896•	0.8362	0.8844•	0.8361	0.8905•	0.8833	0.9293•	0.8060	−0.0254	0.6449•	0.8904•	0.7597
	Kin8nm	0.4070	0.4559•	0.4209	0.4286•	0.4152	0.4826•	0.3358	0.4251•	−0.0056	0.2966•	−0.0937	0.1440•
	Parkinsons	0.2813	0.3618•	0.2801•	0.2387	0.2717	0.3517•	0.5526•	0.5351	−0.0461	0.0194•	0.1928	0.2300•
	Pollen	−2.8110•	−2.8196	−2.8116•	−2.8201	−2.8107•	−2.8228	−2.8122•	−2.8198	−2.8398	−2.8225•	−2.8133•	−2.8207
	Puma8NH	0.2162	0.3373•	0.1969	0.3173•	0.2138	0.3743•	0.3928	0.4386•	−0.0077	0.1841•	−0.5712	−0.0961•
	Space_ga	−0.9547	−0.8772•	−0.8687	−0.8585•	−0.9559	−0.8704•	−0.9747	−0.8582•	−0.9143	−0.8883•	−1.2347	−0.9496•
	Wind	0.2205•	0.1898	−0.0782•	−0.1932	0.2272	0.2816•	0.3639•	0.2914	−0.0559	0.0012•	−0.2476	−0.1482•
	Wine_quality	−0.2187	−0.0950•	−0.2535	−0.0050•	−0.2131	−0.1130•	0.0122	0.0890•	−0.4139	−0.3045•	−0.8501•	−0.8675
	Avg	0.0349	0.1092•	0.0187	0.0750•	0.0409	0.1463•	−10.6245	0.0881•	−0.2965	−0.0857•	−0.2175	−0.0961•
200	Abalone	0.0873	0.2329•	0.2741	0.3059•	0.1130	0.2228•	0.1996	0.2767•	−0.0637	0.3604•	−0.0686	0.0911•
	Bank32NH	−0.0313	0.1170•	0.0221	0.1843•	−0.0236	0.2095•	0.3568	0.4322•	0.0013	0.1338•	0.1397	0.2296•
	Bank8FM	−0.0163	0.6807•	0.4681	0.8219•	0.0235	0.7577•	0.9276	0.9430•	0.0005	0.7005•	0.7425	0.9035•
	Cal_housing	0.4239	0.4671•	0.2949•	0.1548	0.4170	0.5023•	−22.2745	−0.5850•	−0.0422	0.0612•	−0.4734•	−0.5232
	Cpu_act	0.9197	0.9529•	0.8248	0.8851•	0.9193	0.9344•	0.7841	0.9221•	−0.0057	0.1549•	0.3780	0.4801•
	Cpu_small	0.9276	0.9534•	0.8798	0.9279•	0.9281	0.9421•	0.7753	0.9067•	−0.0060	0.1471•	0.4273	0.5364•
	Elevators	0.5261	0.5541•	0.5139	0.5457•	0.5280	0.5816•	0.5905•	0.5412	−0.0070	0.4462•	0.3836	0.4634•
	Folds5x2_pp	0.9054•	0.8287	0.9027•	0.8265	0.9057•	0.8901	0.9321•	0.7959	0.0018	0.7129•	0.9048•	0.7966
	Kin8nm	0.4470	0.5895•	0.4906	0.5814•	0.4543	0.5806•	0.3728	0.5499•	−0.0051	0.4905•	0.3284	0.4767•
	Parkinsons	0.3589	0.4584•	0.3352•	0.2505	0.3652	0.4423•	0.6342	0.6462•	0.0110	0.1966•	0.2731	0.5066•
	Pollen	−2.6338	−2.6253•	−2.6332	−2.6267•	−2.6327	−2.6282•	−2.6255	−2.6217•	−2.6289	−2.6270•	−2.6287	−2.6230•

Table 4 continued

$\|D_L\|$	Dataset	Self-3NN		COREG		SAFER		MSSRA		BHD		S3VR	
		Self-3NN	BSRU	COREG	BSRU	SAFER	BSRU	MSSRA	BSRU	BHD	BSRU	S3VR	BSRU
	Puma8NH	0.2685	0.4537•	0.2595	0.4554•	0.2691	0.5189•	0.4908	0.5261•	−0.0045	0.4525•	−0.1887	0.2247•
	Space_ga	−0.5421	−0.4514•	−0.4333	−0.3848•	−0.5318	−0.4671•	−0.6426	−0.5582•	−0.3218	−0.2938•	−0.9261	−0.8266•
	Wind	0.3996•	0.3160	0.1448	0.2263•	0.3994	0.4067•	0.4494•	0.3320	−0.0746	0.0104•	0.1242	0.1878•
	Wine_quality	−0.1889	−0.1142•	−0.0843•	−0.1454	−0.1890	−0.1510•	−0.0096	0.0112•	−0.2975	−0.2111•	−0.9509	−0.8936•
	Avg	0.1235	0.2276•	0.1506	0.2006•	0.1297	0.2495•	−1.2693	0.2079•	−0.2295	0.0490•	−0.1023	0.0020•
400	Abalone	−0.1053	0.1156•	0.0918	0.1793•	−0.0722	0.0740•	0.0285	0.1567•	−0.2885	0.1799•	−0.1449	0.0749•
	Bank32NH	−0.0272	0.2288•	0.0724	0.2998•	−0.0214	0.3569•	0.3943	0.4723•	0.0018	0.3653•	0.3084	0.4033•
	Bank8FM	0.0081	0.7488•	0.5546	0.8816•	0.0531	0.7790•	0.9418	0.9524•	0.0096	0.7461•	0.8272	0.9387•
	Cal_housing	0.3428	0.3655•	0.2303•	0.2289	0.3605	0.3904•	−7.3182	−1.9846•	−0.0247	0.2198•	−11.8757	−1.9589•
	Cpu_act	0.9364	0.9663•	0.8658	0.9259•	0.9364•	0.9364	0.8871	0.9040•	−0.0041	0.1570•	0.4302	0.5454•
	Cpu_small	0.9425	0.9552•	0.9154	0.9509•	0.9432•	0.9427	0.8740	0.9097•	−0.0043	0.1520•	0.4890	0.5467•
	Elevators	0.5311	0.5571•	0.5227	0.5519•	0.5316	0.5863•	0.5957•	0.5335	−0.0008	0.4503•	0.3587	0.4600•
	Folds5x2_pp	0.9193•	0.8536	0.9185•	0.8570	0.9196•	0.9020	0.9356•	0.8122	0.0064	0.7206•	0.9208•	0.8219
	Kin8nm	0.4977	0.6839•	0.5593	0.6975•	0.5035	0.6405•	0.4095	0.6640•	−0.0070	0.5068•	0.5562	0.6754•
	Parkinsons	0.3168	0.4151•	0.2966•	0.2817	0.3229	0.4199•	0.7044•	0.6881	−0.0483	0.1513•	0.3671	0.4111•
	Pollen	−2.2870	−2.2600•	−2.2859	−2.2602•	−2.2861	−2.2648•	−2.2783	−2.2587•	−2.2754	−2.2590•	−2.2754	−2.2566•
	Puma8NH	0.3149	0.5557•	0.3256	0.5646•	0.3159	0.5899•	0.5810	0.6041•	−0.0057	0.5061•	0.2742	0.5128•
	Space_ga	−0.7738	−0.5983•	−0.6704	−0.5563•	−0.7686	−0.5925•	−0.4500•	−0.4879	−0.7225	−0.6120•	−0.3313•	−0.4159
	Wind	0.4196•	0.3356	0.2624	0.2933•	0.4170	0.4430•	0.4642•	0.3416	−0.0531	0.0403•	−2.3557	−0.7378•
	Wine_quality	−0.0964	−0.0215•	0.0811•	0.0141	−0.0761	−0.0132•	0.0163	0.0691•	−0.0714	0.0257•	−0.7677	−0.6412•
	Avg	0.1293	0.2601•	0.1827	0.2607•	0.1386	0.2794•	−0.2143	0.1584•	−0.2325	0.0900•	−0.8812	−0.0414•

The best results are highlighted with bullet



(a) Training without Gaussian noise (b) Training with Gaussian noise

Fig. 6 Training RMSE curves comparison between with and without Gaussian noise added in the Abalone data set. Both of them are trained with 50 labeled examples and validated with 200 examples. The left (**a**) is the curves training with pure examples, while the right (**b**) is the curves training with examples added with Gaussian noise $\epsilon_j \sim \mathcal{N}(0, 0.4^2)$. Specifically, the blue line shows the RMSE fluctuation of the examples in the training set, while the yellow line shows the RMSE fluctuation of the examples in the validation set

4.3.1 Gaussian noise

Training with Gaussian noise is quite essential since we did not use any form of early stopping in the experiment. The training curves in Fig. 6 help us understand the effects of adding Gaussian noise to examples. The yellow line (validation set) in Fig. 6a obviously shows the overfitting in training with pure examples. This is often observed when training neural networks with limited labeled data, which is naturally encountered in semi-supervised learning. However, this cannot be observed in training with Gaussian noise added examples in Fig. 6b. The reason for this may be that Gaussian noise augments the limited labeled data. Based on the smoothing hypothesis, it is reasonable to assume that a labeled sample shares its label with its neighbors in its immediate vicinity. Additionally, the label of a sample should be consistent with the label of its slightly perturbed version with tiny Gaussian noise. In other words, Gaussian noise can augment the limited labeled data and prevent neural networks from overfitting.

4.3.2 Unsupervised loss

We design an ablation experiment with and without unsupervised loss to answer the third question. Figure 7 shows the difference between BSRU based on MSSRA with and without unsupervised loss on 15 data sets under four label ratios: 0.025, 0.05, 0.1, and 0.2 corresponding to label sizes of 50, 100, 200, and 400. In most data sets, BSRU with unsupervised loss boost significantly compared to the learner without unsupervised loss. In detail, the improvement of Cpu_act and Cpu_small is the most significant, reaching 54.81% and 54.44%, respectively.

However, the unsupervised loss negatively impacts the performance on certain datasets under specific label ratios. For example, in the Folds5x2_pp dataset with a 4-dimensional attribute space, the unsupervised loss leads to a decline in performance under four different label ratios. There are at least two possible reasons:

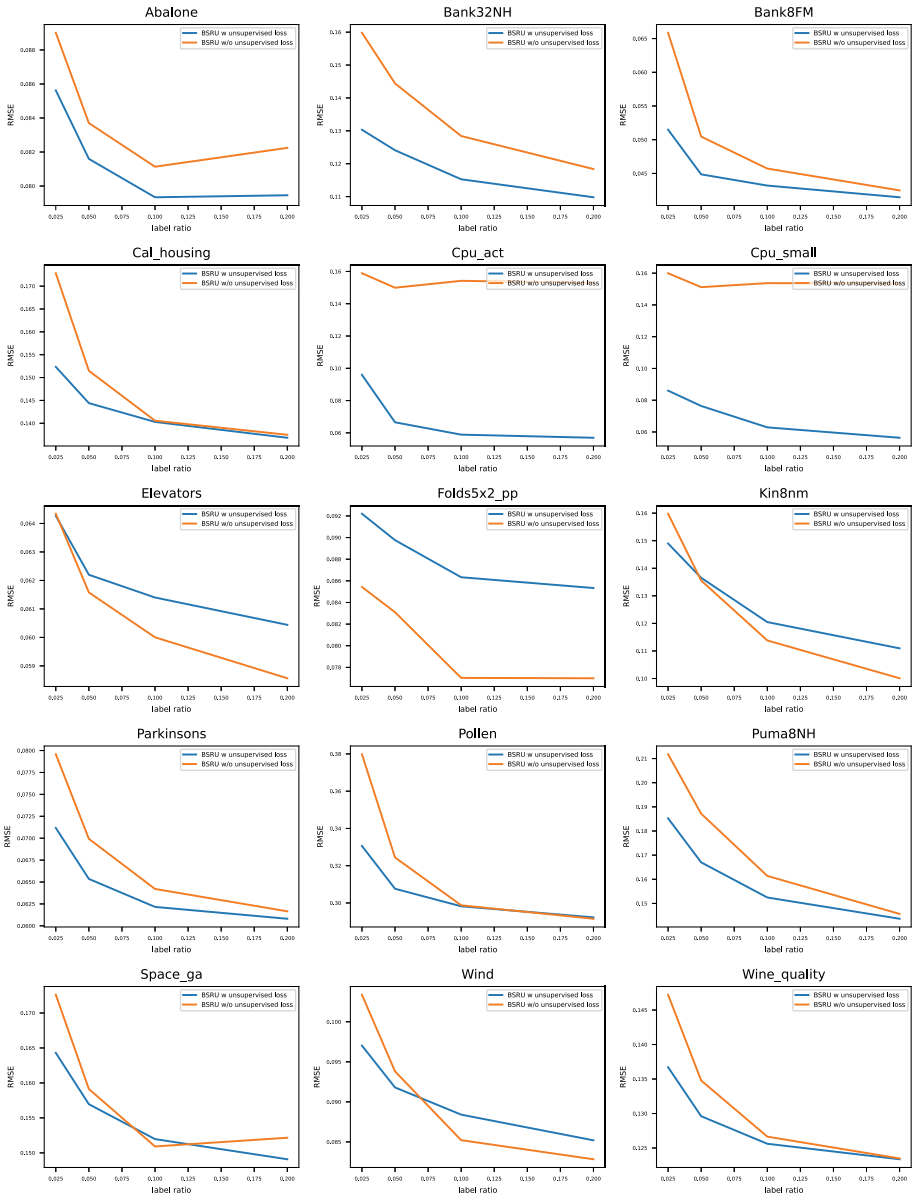


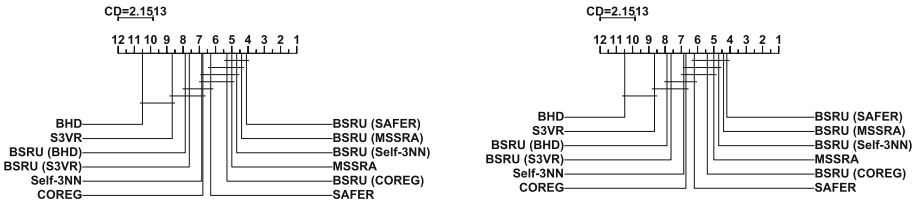
Fig. 7 The RMSE comparison between BSRU based on MSSRA with and without unsupervised loss on fifteen data sets under four label ratios: 0.025, 0.05, 0.1, and 0.2

(1) The corresponding base model exhibits substantial prediction errors on unlabeled data, which might mislead the network when optimizing unsupervised losses.
(2) The low-dimensional attribute data limits the network's optimization capability for unsupervised loss.

We can now answer the questions proposed at the beginning of this section. According to Table 3 and 4, BSRU enhances its base model in most of the data sets. Meanwhile, in certain

Table 5 Summary of the Friedman test and the critical value in terms of each evaluation metric

Metric	F_F	Critical value ($\alpha = 0.05$)
RMSE	24.0277	1.8034
R^2	23.5610	



(a) RMSE (b) R^2

Fig. 8 Comparison of the twelve algorithms with the Nemenyi test at significance level $\alpha = 0.05$

low-dimensional attribute data sets, BSRU performs similarly to or slightly less effectively than the baseline methods. According to Fig. 6, Gaussian noise enhances robustness by preventing overfitting during training. It is worth noting that adding Gaussian noise is essentially a technique to augment limited labeled data. According to Fig. 7, the unsupervised loss item is effective. Utilizing this technique, BSRU outperforms in 13 out of 15 data sets.

4.4 Statistical hypothesis test

For further assessment of algorithm performance, we employ the Friedman test [67] at a 5% significance level. This test evaluates whether the average ordering of measurements for each approach significantly differs under the null hypothesis of all approaches. Table 5 illustrates the results of the Friedman test and the corresponding critical values for each evaluation metric. The results indicate the rejection of the null hypothesis, suggesting that all the algorithms do not exhibit equal performance.

Since the Friedman test rejects the null hypothesis, the Nemenyi test [67, 68] with 5% significance level is applied to the above fifteen data sets. If the difference between the average rankings of the two algorithms on all data sets is less than or equal to the critical difference (CD), there is no significant difference between these algorithms; otherwise, a significant difference exists. Furthermore, the difference between the two algorithms is distinguished by the $\text{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$, where $q_\alpha = 3.268$, the uppermost behavior threshold $\text{CD} = 2.1513$ ($k = 12$, $N = 60$). If there is no significant difference between the two algorithms, the solid line is connected, and vice versa.

As shown in Fig. 8, the performance of the algorithm is sequentially reduced from right to left in the subfigures. We can observe that BSRU always maintains the best average rank compared with its base model. Meanwhile, in Fig. 8a, there is no evidence of a statistical difference among BSRU (SAFER), BSRU (MSSRA), BSRU (Self-3NN), and BSRU (COREG). However, they outperform their corresponding base models. Similarly, this performance also exists in the R^2 evaluation measure in Fig. 8b.

5 Conclusion

This paper designs a ramp-up unsupervised loss to learn a better learner from an existing semi-supervised regressor. The loss can be broken down into two parts: the supervised loss, which pertains to the labeled data, and the unsupervised loss, which pertains to the unlabeled data. Notably, the unsupervised loss increases with the training epoch, indicating more attention to the noisy pseudo-labels. Experimental results verify the advantage of BSRU beyond the baseline methods. Meanwhile, two ablation experiments demonstrate the effectiveness of the unsupervised loss.

There are still some topics that require further investigation.

(1) Developing a safe technique. We observed that BSRU exhibits slightly inferior performance compared with the base model in certain data sets. Therefore, the research objectives for our future work include developing a safe technique that ensures the performance is not inferior to the base regressor, even in low-dimensional data sets.
(2) Combining with consistency regularization which has flourished in semi-supervised classification. BSRU leverages the predictions of another regressor to enhance the performance of the network. Therefore, it is worthwhile to consider incorporating consistency regularization to enhance the model's resilience against perturbations.
(3) Further enhancing by self-learning. By leveraging an off-the-shelf trained regressor, BSRU can achieve a better-trained model. Exploring further enhancements through self-learning techniques, such as self-distillation [52, 53], is an intriguing direction for future research.
(4) Embedding with a label confidence strategy. BSRU places a greater emphasis on the quantity of pseudo-labels rather than solely prioritizing their quality. It tries to use all the pseudo-labels assigned by a trained off-the-shelf model. These pseudo-labels are often high-quality but may contain noise. Hence, an intriguing direction for future research is to integrate a label confidence strategy [41], maintaining a balance between the quality and quantity of pseudo-labels.

Acknowledgements This work was supported by the National Social Science Foundation of China under Grant No. 22FZXB092. We thank Yan-Xue Wu for his valuable suggestions.

Author Contributions CRediT authorship contribution statement Liyan Liu did methodology, software, writing—original draft; Haimin Zuo done formal analysis, writing—review & editing; Fan Min contributed to conceptualization, supervision, funding acquisition, writing—review & editing.

Declarations

Conflict interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Zhu X-J, Goldberg AB (2009) Introduction to semi-supervised learning. Synthe Lect Artif Intell Mach Learn 3(1):1–130. https://doi.org/10.2200/S00196ED1V01Y200906AIM006
2. Engelen JEV, Hoos HH (2020) A survey on semi-supervised learning. Mach Learn 109(2):373–440. https://doi.org/10.1007/s10994-019-05855-6
3. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: COLT, pp 92–100
4. Zhou Z-H, Li M (2005) Semi-supervised regression with co-training. In: IJCAI, pp 908–913
5. 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

6. Hady MFA, Schwenker F, Palm G (2009) Semi-supervised learning for regression with co-training by committee. In: ICANN, pp 121–130
7. Qiao S, Shen W, Zhang Z, Wang B, Yuille A (2018) Deep co-training for semi-supervised image recognition. In: ECCV, pp 142–159
8. Chen D-D, Wang W, Gao W, Zhou Z-H (2018) Tri-Net for semi-supervised deep learning. In: IJCAI, pp 2014–2020
9. Zhang J, Li M, Gao K, Meng S, Zhou C (2021) Word and graph attention networks for semi-supervised classification. Knowl Inf Syst 63(11):2841–2859. https://doi.org/10.1007/s10115-021-01610-3
10. Zhang T, Zhu T, Han M, Chen F, Li J, Zhou W, Yu PS (2022) Fairness in graph-based semi-supervised learning. Knowl Inf Syst 65(2):543–570. https://doi.org/10.1007/s10115-022-01738-w
11. Lebichot B, Saerens M (2020) An experimental study of graph-based semi-supervised classification with additional node information. Knowl Inf Syst 62(11):4337–4371. https://doi.org/10.1007/s10115-020-01500-0
12. Jean N, Xie SM, Ermon S (2018) Semi-supervised deep kernel learning: regression with unlabeled data by minimizing predictive variance. In: NeurIPS, pp 5327–5338
13. Reed SE, Lee H, Anguelov D, Szegedy C, Erhan D, Rabinovich A (2015) Training deep neural networks on noisy labels with bootstrapping. In: ICLRW, pp 1–11
14. Wei H, Feng L, Chen X, An B (2020) Combating noisy labels by agreement: a joint training method with co-regularization. In: CVPR, pp 13726–13735
15. Tan C, Xia J, Wu L, Li SZ (2021) Co-learning: Learning from noisy labels with self-supervision. In: ACM Int. Conf. Multimedia, pp 1405–1413
16. Khan FH, Qamar U, Bashir S (2017) A semi-supervised approach to sentiment analysis using revised sentiment strength based on sentiwordnet. Knowl Inf Syst 51(3):851–872. https://doi.org/10.1007/s10115-016-0993-1
17. Dai W, Li X, Cheng K-T (2023) Semi-supervised deep regression with uncertainty consistency and variational model ensembling via bayesian neural networks. In: AAAI, pp 1–10
18. Berthelot D, Carlini N, Cubuk ED, Kurakin A, Sohn K, Zhang H, Raffel C (2020) Remixmatch: semi-supervised learning with distribution alignment and augmentation anchoring. In: ICLR, pp 1–10
19. Kuo C-W, Ma C-Y, Huang J-B, Kira Z (2020) Featmatch: Feature-based augmentation for semi-supervised learning. In: ECCV, pp 479–495
20. Laine S, Aila T (2017) Temporal ensembling for semi-supervised learning. In: ICLR, pp 1–13
21. 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
22. Tarvainen A, Valpola H (2007) Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results. In: NeurIPS, pp 1–16
23. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
24. Ke Z, Wang D, Yan Q, Ren J, Lau RWH (2019) Dual student: breaking the limits of the teacher in semi-supervised learning. In: ICCV, pp 1–12
25. Sohn K, Berthelot D, Carlini N, Zhang Z, Zhang H, Raffel CA, Cubuk ED, Kurakin A, Li C-L (2020) Fixmatch: simplifying semi-supervised learning with consistency and confidence. In: NeurIPS, pp 596–608
26. Zhang B, Wang Y, Hou W, WU H, Wang J, Okumura M, Shinozaki T (2021) Flexmatch: boosting semi-supervised learning with curriculum pseudo labeling. In: NeurIPS, pp 18408–18419
27. Xu Y, Wei F, Sun X, Yang C, Shen Y, Dai B, Zhou B, Lin S (2022) Cross-model pseudo-labeling for semi-supervised action recognition. In: CVPR, pp 2959–2968
28. Bodla N, Hua G, Chellappa R (2018) Semi-supervised fusedGAN for conditional image generation. In: ECCV, pp 689–704
29. You C, Zhao R, Staib LH, Duncan JS (2022) Momentum contrastive voxel-wise representation learning for semi-supervised volumetric medical image segmentation. In: MICCAI, pp 639–652
30. Lin Y, Yao H, Li Z, Zheng G, Li X (2022) Calibrating label distribution for class-imbalanced barely-supervised knee segmentation. In: MICCAI, pp 109–118
31. Zheng M, You S, Huang L, Wang F, Qian C, Xu C (2022) Simmatch: semi-supervised learning with similarity matching. In: CVPR, pp 14451–14461
32. Yang L, Qi L, Feng L, Zhang W, Shi Y (2023) Revisiting weak-to-strong consistency in semi-supervised semantic segmentation. In: CVPR, pp 1–13
33. Lee D-H (2013) Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks. In: ICML Workshop, pp 1–20

34. Fazakis N, Karlos S, Kotsiantis S, Sgarbas K (2019) A multi-scheme semi-supervised regression approach. Pattern Recognit Lett 125:758–765. https://doi.org/10.1016/j.patrec.2019.07.022
35. Nigam K, Ghani R (2000) Analyzing the effectiveness and applicability of co-training. In: CIKM, pp 86–93
36. Wang W, Zhou Z-H (2013) Co-training with insufficient views. In: ACML, pp 467–482
37. Brefeld U, Gärtner T, Scheffer T, Wrobel S (2006) Efficient co-regularised least squares regression. In: ICML, pp 137–144
38. Wang X, Fu L, Ma L (2011) Semi-supervised support vector regression model for remote sensing water quality retrieving. Chin Geogr Sci 21:57–64
39. Bao L, Yuan X, Ge Z (2015) Co-training partial least squares model for semi-supervised soft sensor development. Chemom Intell Lab Syst 147:75–85. https://doi.org/10.1016/j.chemolab.2015.08.002
40. Liu Y, Xu Z, Li C (2018) Online semi-supervised support vector machine. Inf Sci 439–440:125–141. https://doi.org/10.1016/j.ins.2018.01.048
41. Chen X, Cao W, Gan C, Ohyama Y, She J, Wu M (2021) Semi-supervised support vector regression based on data similarity and its application to rock-mechanics parameters estimation. Eng Appl Artif Intell 104:104317. https://doi.org/10.1016/j.engappai.2021.104317
42. 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 35:1483–1500. https://doi.org/10.3233/JIFS-169689
43. Nigam K, Ghani R (2000) Analyzing the effectiveness and applicability of co-training. In: CIKM, pp 86–93
44. Brefeld U, Scheffer T (2004) Co-EM support vector learning. In: ICML, p 16
45. Zhou Z-H, Li M (2010) Semi-supervised learning by disagreement. Knowl Inf Syst 24(3):415–439. https://doi.org/10.1007/s10115-009-0209-z
46. Sun X, Gong D, Zhang W (2012) Interactive genetic algorithms with large population and semi-supervised learning. Appl Soft Comput 12(9):3004–3013. https://doi.org/10.1016/j.asoc.2012.04.021
47. Zhou Y, Goldman S (2004) Democratic co-learning. In: ICTAI, pp 594–602
48. Min F, Li Y, Liu L (2022) Self-paced safe co-training for regression. In: PAKDD, pp 71–82
49. 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
50. Owen AB (2007) A robust hybrid of lasso and ridge regression. Contemp Math 443(7):59–72
51. Dong X, Yu Z, Cao W, Shi Y, Ma Q (2020) A survey on ensemble learning. Front Comput Sci 14:241–258. https://doi.org/10.1007/s11704-019-8208-z
52. Mobahi H, Farajtabar M, Bartlett PL (2020) Self-distillation amplifies regularization in Hilbert space. In: NeurIPS, pp 3351–3361
53. Zhang Z, Sabuncu M (2020) Self-distillation as instance-specific label smoothing. In: NeurIPS, pp 2184–2195
54. Gou J, Yu B, Maybank SJ, Tao D (2021) Knowledge distillation: a survey. Int J Comput Vision 129(6):1789–1819. https://doi.org/10.1007/s11263-021-01453-z
55. Wang Y, Chen H, Heng Q, Hou W, Fan Y, Wu Z, Wang J, Savvides M, Shinozaki T, Raj B, Schiele B, Xie X (2023) Freematch: self-adaptive thresholding for semi-supervised learning. In: ICLR, pp 1–20
56. Chen H, Tao R, Fan Y, Wang Y, Wang J, Schiele B, Xie X, Raj B, Savvides M (2023) Softmatch: addressing the quantity-quality tradeoff in semi-supervised learning. In: ICLR, pp 1–21
57. Rizve MN, Duarte K, Rawat YS, Shah M (2021) In defense of pseudo-labeling: an uncertainty-aware pseudo-label selection framework for semi-supervised learning. In: ICLR, pp 1–20
58. Xu Y, Shang L, Ye J, Qian Q, Li Y-F, Sun B, Li H, Jin R (2021) Dash: semi-supervised learning with dynamic thresholding. In: ICML, pp 11525–11536
59. Yarowsky D (1995) Unsupervised word sense disambiguation rivaling supervised methods. In: ACL, pp 189–196
60. Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27. https://doi.org/10.1109/TIT.1967.1053964
61. Li Y-F, Zha H-W, Zhou Z-H (2017) Learning safe prediction for semi-supervised regression. In: AAAI, pp 2217–2223
62. Shevade SK, Keerthi SS, Bhattacharyya C, Murthy KRK (2000) Improvements to the SMO algorithm for SVM regression. IEEE Trans Neural Netw 11(5):1188–1193. https://doi.org/10.1109/72.870050
63. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
64. Barros RC, Ruiz DD, Basgalupp MP (2011) Evolutionary model trees for handling continuous classes in machine learning. Inf Sci 181(5):954–971. https://doi.org/10.1016/j.ins.2010.11.010
65. Timilsina M, Figueroa A, d'Aquin M, Yang H (2021) Semi-supervised regression using diffusion on graphs. Appl Soft Comput 104:107188. https://doi.org/10.1016/j.asoc.2021.107188

66. Seok K (2014) Semi-supervised regression based on support vector machine. J Korean Data Inf Sci Soc 25(2):447–454
67. Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701. https://doi.org/10.1080/01621459.1937.10503522
68. Nemenyi PB (1963) Distribution-free multiple comparisons. PhD thesis, Princeton University

Liyan Liu received the M.S. degree in Software Engineering from the School of Computer Science, Southwest Petroleum University in 2010. She is currently an associate professor at Southwestern University of Petroleum. Her research interests include AI in oilfield development and semi-supervised learning.

Haimin Zuo is a graduate student with Southwest Petroleum University, Chengdu, China. His current research interests include semi-supervised regression and deep learning.

Fan Min received the M.S. and Ph.D. degrees from the School of Computer Science and Engineering, University of Electronics Science and Technology of China in 2000 and 2003, respectively. He visited the University of Vermont from 2008 to 2009. He is currently a professor with Southwest Petroleum University, China. He has published more than 200 refereed papers in various journals and conferences, including IEEE Transactions on Geoscience and Remote Sensing, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Pattern Recognition, Knowledge-Based Systems, Computers and Geosciences, and Information Sciences. His research has been supported by National Natural Science Foundation of China. His current research interests include machine learning and its applications such as seismic inversion, recommender systems, and microbiology.