A New Learning Paradigm for Stochastic Configuration Network: SCN+

Yanshuang Ao *, Xinyu Zhou *, Wei Dai *b,∗

* School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China;
∗b State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China

ABSTRACT

Learning using privileged information (LUPI) paradigm, which pioneered teacher-student interaction mechanism, makes the learning models use additional information in training stage. This paper is the first to propose an incremental learning algorithm with LUPI paradigm for stochastic configuration network (SCN), named SCN+. This novel algorithm can leverage privileged information into SCN in the training stage, which provides a new method to train SCN. Moreover, the convergences have been studied in this paper. Finally, experimental results indicate that SCN+ indeed performs favorably.

Keywords:
Learning using privileged information
Stochastic configuration networks
Randomized learner model
Incremental learning
Constructive method

1. Introduction

Gradient algorithms [1] are currently the most widely used in neural network learnings. Although an increasing number of works are aiming at further improving the performance of gradient algorithms, there are still some problems, such as slow convergence, easy to fall into local minimum, strong dependence on the setting of initial parameters, etc. [2]. Thus, it is meaningful to establish a strategy that could make learning process more quickly and more efficient.

Random weight neural networks (RWNNs) are representative examples and have received extensive attention due to its fast learning speed, simplicity of implementation, and excellent generalization performance [3] [4]. In 1994, Pao et al. [5] presented random vector functional link (RVFL) networks, which add a direct link between the input and the output layers in the single layer feedforward neural networks. It should be noted here that RVFL networks calculate the network weights connected to the output layer nodes by solving simple linear regression problems, while other network weights and biases are randomly generated in a specific interval based on a given probability distribution [6] [7]. This type of algorithms does not need to iterate repeatedly, which overcomes the bottleneck problem encountered by traditional gradient algorithms.

Wang et al. [8] proposed a novel random incremental learning technology named stochastic configuration networks (SCNs). By randomly assigning hidden layer parameters within an adjustable interval and innovatively introducing supervisory mechanism to constrain them, SCNs ensure the universal approximation property. Besides, as an incremental learning model, SCNs establish a candidate node pool in each incremental learning process to select the best nodes, which speeds up the convergence rate. At the same time, the network structure could also be determined in the incremental learning process. Aiming at the selection of key parameters in the supervisory mechanism, Zhu et al. [9] gave two kinds of inequality constraints instructively to improve the efficiency of random parameter configuration and help for establishing the candidate node pools. In [10], an ensemble learning method for quickly disassociating heterogeneous neurons was proposed. It used SCNs as base models and adopted a negative correlation learning strategy to evaluate output weights. The TSVD-SCN model was proposed in [11] and aimed at solving ill-posed conditions such as multicollinearity that may exist in the output matrix of the hidden layer. Besides, the truncated singular value decomposition was used to perform generalized inverse operations on the reconstructed output matrix. Wang et al. [12] improved the generalization performance of SCNs by introducing regularization techniques. Wang and Li [13] extended SCNs to deep structure (DeepSCNs) and reported the theoretical analysis and algorithm realization. The results showed that DeepSCNs could be generated quickly and efficiently in a self-organization way, and the generalization performance is similar with the other state-of-the-art deep neural networks. The use of one-dimensional SCNs might destroy the spatial information of the image and lead to poor performance in the tasks of image data modeling. Thus, Li and Wang [14] expanded the original SCNs into a two-dimensional version called 2D-SCNs, which shows good potential for image data analytics. To solve the uncertain

*Corresponding author.
Email address: weidai@cumt.edu.cn

This work was supported by the Nature Science Foundation of Jiangsu Province [grant number BK20160275]; the Postdoctoral Science Foundation of China [grant numbers 2015M581885, 2018T110571]; and the Open Project Foundation of State Key Laboratory of Synthetical Automation for Process Industries [grant number PAL-N201706].
data regression problem, two different robust SCNs with kernel density estimation and maximum correntropy criterion were proposed in [15] and [16], respectively. For solving the prediction interval estimation problems that are commonly presented in industrial processes, Lu and Ding et al. [17] developed the corresponding deep ensemble, and robust versions of SCNs.

The existing algorithms on SCNs have not considered cases that some data are available in the training stage but unavailable in the test stage. Suppose our goal is to determine whether a patient has cancer based on the biopsy image. The problem here is to find the classification rules in the pixel space given the image described in the pixel space. However, in addition to the pictures, the doctor also has a report written by a pathologist, which uses high-level language to describe the pictures. The problem is that if you train the pictures together with the pathologist’s report, you will not be able to get the pathologist’s report in time when predicting new pictures. In fact, the goal here is to make an accurate diagnosis without consulting a pathologist. However, the pathologist’s diagnosis has very important guiding significance for judging the classification of these pictures. Unfortunately, original SCNs is hard to utilize this instructive additional information that is different from the original feature space into the training phase for more accurately predicting.

In order to solve the above problem, learning using privileged information (LUPI) paradigm [18] was first proposed by Vapnik and Vashist. They provided a support vector machine algorithm with privileged information termed SVM+. LUPI paradigm fasten the learning attention on some elements of human teaching. It considers that a teacher who can provide students with explanations plays a significant role in human learning. In contrast to the traditional learning paradigm, LUPI paradigm can provide a set of additional information for the training data during the training process. By employing privileged information, LUPI paradigm solves an open problem of how the data modalities being only available during training time help the learning model achieve better prediction. The new learning paradigm is incorporated into weighted SVM [19], metric learning [20] [21], learning to rank [22], RVFL networks [23] and so on. However, the existing algorithms with LUPI paradigm are all based on batch algorithms, which have certain limitations. There are two essential defects make the application of batch algorithms restricted to some extent: 1) The scale of hidden nodes cannot be determined before training, which means it is difficult to set a suitable network structure; 2) The hidden layer parameters are always generated in a constant interval, which affects its actual approximation property [24] [25].

Moreover, comparing with pruning algorithms, the incremental learning algorithms in SCNs have more advantages. First, the incremental learning algorithms directly define a small initial network, but the pruning algorithms do not know how large the initial network is. Second, incremental learning algorithms always search for small solutions, while pruning algorithms spend most of their training time on networks larger than the optimal solution, which is more time-consuming. Third, there are many different size networks may be acceptable, incremental learning algorithms are likely to find smaller network solutions than pruning algorithms. Finally, the supervisory mechanism adopted by SCNs ensures the universal approximation characteristics of the networks.

In order to enable SCN to use privileged information to build a more generalized model, an incremental learning algorithm and a supervisory mechanism with LUPI paradigm for RVFL is proposed, termed SCN+. The convergence of SCN+ has been proved in this paper, which provides a strong theoretical support. Finally, experiments on the datasets in the machine learning database verify the effectiveness of SCN+ for classification and regression tasks. The work of this paper is mainly carried out from the perspective of algorithm improvement without considering the practical application.

The rest of this paper is as follows: Section 2 briefly describes SCNs and LUPI. Section 3 introduces SCNs+ in detail, including theoretical analysis and algorithm description. Section 4 reports performance evaluation results. Section 5 summarizes this paper and makes some comments on the follow-up work.

2. Related work

Given $N$ samples $(x_i, t_i)$, input $X = \{x_1, x_2, \ldots, x_N\}$, $x_i = \{x_{i,1}, \ldots, x_{i,d}\} \in \mathbb{R}^d$, corresponding output $T = \{t_1, t_2, \ldots, t_N\}$, $t_i = \{t_{i,1}, \ldots, t_{i,m}\} \in \mathbb{R}^m$, with $i=1, 2, \ldots, N$. SCNs with $L$ hidden nodes can be expressed as:

$$f_l(X) = \sum_{j=1}^{L} H_j(\omega_j, b_j, X)\beta_j,$$

(1)

where $H_j(\cdot)$ represents activation function of the $j$-th hidden node; the hidden-node parameters $(\omega_j$ and $b_j$) are randomly assigned from $[-\lambda, \lambda]^d$ and $[-\lambda, \lambda]$ respectively; $\beta_j = [\beta_{j,1}, \ldots, \beta_{j,q}, \ldots, \beta_{j,m}]^T$ expresses the output weights between the $j$-th hidden node and the output nodes; $f_l$ denotes the output function of current network.

SCNs adopt incremental learning method to build model.

2.1. SCNs

In SCNs, the hidden nodes are increased individually. That is to say, starting from randomly generating the first node $g_1 = (\omega_1, b_1, x)$, SCNs will gradually add nodes to the network [8]. The output $f_l$ can be expressed as a specific combination of the previous network $f_{l-1}$ and the newly added node $g_l$ ($v_l$ and $b_l$):

$$f_l(x) = f_{l-1}(x) + H_l\beta_l,$$

(2)

where the output weight of new node can be obtained based on the following formula:
\[ \beta_i = \frac{[e_{L,i}H_i]}{[H_i]} \]  

And the residual of previous network is
\[ e_{L,i} = f - f_{L,i} = [e_{L-1,i}, \ldots, e_{L-1,n}] \]  

The supervisory mechanism used in SCNs is presented in Theorem 1.

**Theorem 1** [11]: Let \( \Gamma := \{H_i, H_0, H_b, \ldots \} \) be a set of real-valued function, and span(\( \Gamma \)) denotes a function space spanned by \( \Gamma \). Suppose that span(\( \Gamma \)) is dense in \( L^2 \) space and \( \forall g \in \Gamma, 0 < \|\| H \|\| < b_g \) for some \( b_g \in R^+ \). Given \( 0 < r < 1 \) and a non-negative real number sequence \( \{\mu_L\} \), with \( \lim_{L \to +\infty} \mu_L = 0 \) and \( \mu_L \leq (1 - r) \). For \( L=1, 2, \ldots \), denoted by:
\[ \delta_L = (1 - r - \mu_L)\|e_{L-1}\| \]  

If the random basis function \( g_L \) is generated to satisfy the following inequalities:
\[ \langle e_{L-1,i}, H_i \rangle \geq b^2_L \delta_L \]  

Then, we have \( \lim_{L \to +\infty} \|f - f_0\| = 0 \).

The network structure and parameter optimization problems can be solved simultaneously in the above learning process.

**Remark 1**: According to the different solution \( \beta_L \) of output weight, three algorithmic implementations of SCNs, namely SC-I, SC-II, and SC-III, are developed. (3) is used in SC-I, and other details can be referred to [11].

### 2.2. Learning using privileged information

The LUPI paradigm can be described as follows: given a set of triplets \((x_i, \hat{x}_i, y_i)\), \(i = 1, 2, \ldots, N\), \(x_i \in X\), \(\hat{x}_i \in \hat{X}\), \(y_i \in Y\) generated according to a fixed but unknown probability measure \( P(x, \hat{x}, y) \) find among a given set of functions \( f(x, a) \), \(a \in A\), the function \( y = f(x, \hat{a}) \) that guarantees the smallest probability of incorrect classification or regression. Generally speaking, the additional information \( \hat{x} \in \hat{X} \) belongs to the space \( \hat{X} \) which is different from the space \( X \). In other words, this is a new learning paradigm that the additional information is available at the training stage but it is not available for the test set.

### 3. Stochastic configuration networks with privileged information

This section details SCN+. First, we describe the algorithm process. Afterwards, the convergence analysis and its implementation are shown.

#### 3.1. Algorithm description

![SCN+](image)

The structure of SCN+ is shown in Fig. 1. In SCN+, the definitions of \( x, \hat{x} \) and \( y \) are input, additional privileged information and output respectively. Notice that the additional privileged information \( \hat{x} \) is only used in the training stage. Usually, the SCN+ gradually increase the hidden nodes until an acceptable performance is met. The initial residual error
\[ e_0 = f - f_0 \]  

where \( f_0 = H_0\beta_0 \) and the residual error
\[ e_L = f - f_L = f - (f_{L-1} + H_i\beta_L + \hat{H}_i\hat{\beta}_L) = e_{L-1} - (H_i\beta_L + \hat{H}_i\hat{\beta}_L) \]  

in the \( L \)th \((L = 1, 2, \ldots, L_{max})\) grown (where
\[ \beta_L = [\beta_{L,1}, \ldots, \beta_{L,1}^T] \text{ and } \tilde{\beta}_L = [\tilde{\beta}_{L,1}, \ldots, \tilde{\beta}_{L,1}^T] \] are output weight vectors of newly added hidden nodes corresponding to the original feature \( x \) and privileged feature \( \tilde{x} \) respectively in the \( L \)-th grown. The hidden layer output vectors

\[
H_L = \begin{bmatrix}
G((\alpha_{L,1}, x_1) + b_1) \\
\vdots \\
G((\alpha_{L,L}, x_L) + b_L)
\end{bmatrix}
\quad \text{and} \quad
\tilde{H}_L = \begin{bmatrix}
G((\tilde{\alpha}_{L,1}, \tilde{x}_1) + \tilde{b}_1) \\
\vdots \\
G((\tilde{\alpha}_{L,L}, \tilde{x}_L) + \tilde{b}_L)
\end{bmatrix}
\]

are corresponding to the original feature and privileged feature after the \( L \)-th grown.

The target function in the constructive process can be formulated as

\[
\min_{\beta_L} f = \frac{1}{2}[\beta_L^T \beta_L] + \frac{\gamma}{2}[\tilde{\beta}_L^T \tilde{\beta}_L] + C[H_L \tilde{H}_L] + \frac{1}{2}\|e_L\|^2 \\
= \frac{1}{2}\|\beta_L\|^2 + \frac{1}{2}\|\tilde{\beta}_L\|^2 + \frac{\gamma}{2}\|\beta_L\| + \frac{\gamma}{2}\|\tilde{\beta}_L\| + C[H_L \tilde{H}_L] + \frac{1}{2}\|e_L\|^2
\]

\( s.t. \ H_L + H_L \tilde{\beta}_L = Y - (\tilde{H}_{L} \tilde{\beta}_L) \)

where

\[
\beta = [\beta_1, \beta_2, \ldots, \beta_{L-1}]^T, \quad \tilde{\beta} = [\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_{L-1}]^T,
\]

\[
H = \begin{bmatrix}
G((\alpha_{1,1} + b_1) \\
\vdots \\
G((\alpha_{L,1} + b_L)
\end{bmatrix}
\quad \text{and} \quad
\tilde{H}_L = \begin{bmatrix}
G((\tilde{\alpha}_{1,1} + \tilde{b}_1) \\
\vdots \\
G((\tilde{\alpha}_{L,1} + \tilde{b}_L)
\end{bmatrix}
\]

are constructed and fixed before the \( L \)-th grown. \( C[H_L \tilde{H}_L] \) and \( C[H_L \tilde{H}_L] + \gamma \) are slack functions and \( \gamma \) is a regularization coefficient.

Take the partial derivative of the target function with respect to \( \beta_L \) and \( \tilde{\beta}_L \), we have

\[ \frac{\partial f}{\partial \beta_L} = \beta_L + H_L^T (H_L \beta_L + \tilde{H}_L \tilde{\beta}_L - e_{L-1}) = 0 \] (8)

\[ \frac{\partial f}{\partial \tilde{\beta}_L} = \gamma \tilde{\beta}_L + \tilde{H}_L^T (H_L \beta_L + \tilde{H}_L \tilde{\beta}_L - e_{L-1}) + C[H_L \tilde{H}_L]^T = 0 \] (9)

Combining (1) and (1), the output weights can be calculated as

\[ \beta_L = \frac{(\gamma + \tilde{H}_L^T \tilde{H}_L)H_L^T \tilde{H}_L - H_L^T \tilde{H}_L}{(1 + \tilde{H}_L^T \tilde{H}_L)H_L^T \tilde{H}_L} \] (10)

\[ \tilde{\beta}_L = \frac{\tilde{H}_L^T \tilde{H}_L}{(1 + \tilde{H}_L^T \tilde{H}_L)H_L^T \tilde{H}_L} + \frac{(1 + H_L^T H_L)H_L^T (\gamma + \tilde{H}_L^T \tilde{H}_L)}{(1 + H_L^T H_L)H_L^T \tilde{H}_L} \] (11)

where \( I \) is an identity matrix. Furthermore, we can obtain the output function \( f_{\text{IRVFL+}}(z) = H(z) \beta \) in the test stage when using the test data \( z \). We call the above algorithm without supervision mechanism IRVFL+.

3.2. Definition of SCN+

**Theorem 2.** Suppose that \( \text{span}(\Gamma) \) is dense in \( L_2 \) space. Given \( 0 < r < 1 \) and an nonnegative real number sequence \( \\{\mu_k\} \), with \( \lim_{k \to \infty} \mu_k = 0 \) and \( \mu_k \leq (1-r)^{k-1} \). For \( L = 1, 2, \ldots \), denoted by

\[ \delta_L = (1 - r - \mu_L)^{\|e_{L-1}\|^2} \] (12)

If the hidden output block vectors \( H_L \) and \( \tilde{H}_L \) are generated with the following inequalities:

\[ \langle e_{L-1}, H_L \beta_L + \tilde{H}_L \tilde{\beta}_L \rangle \geq \delta_L \] (13)

and the output weights \( \beta_L \) and \( \tilde{\beta}_L \) are evaluated by (10) and (11) respectively, then we have \( \lim_{L \to \infty} \|f - f_L\| = 0 \).
3.3. Convergence analysis

For calculating easily, we reformulate the target function of SCN+ as

\[
\min_{\Delta \beta} f = \frac{1}{2} \Delta \beta^T A \Delta \beta + \frac{1}{2} \Delta \beta_l^T A \Delta \beta_l + \Delta H \beta_l ,
\]

where

\[
\Delta \beta = \begin{bmatrix} \beta \end{bmatrix}, \quad \Delta \beta_l = \begin{bmatrix} \beta_l \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}.
\]

(14)

Thus, the solution formula for the output weight could be obtained:

\[
\Delta \beta_l = \left( A + \Delta H_l^T \Delta H_l \right)^{-1} \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right).
\]

(17)

Then, we can prove

\[
\| e_l \| - (r + \mu_k) \| e_{l-1} \| = (e_{l-1} - \Delta H_l \Delta \beta_l - \Delta H_l e_{l-1} - \Delta H_l \Delta \beta_l) - (r + \mu_k) (e_{l-1}, e_{l-1})
\]

\[
= (1 - r - \mu_k) (e_{l-1}, e_{l-1}) - 2 (e_{l-1}, \Delta H_l \Delta \beta_l)
\]

\[
+ \Delta H_l \Delta \beta_l, \Delta H_l \Delta \beta_l
\]

\[
= \delta_l - \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right)^T \left( A + \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right)^T \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right)
\]

\[
+ 2 e_{l-1}^T \Delta H_l \left( A + \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right) \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right)
\]

\[
\leq \delta_l - e_{l-1}^T \Delta H_l \left( A + \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right) \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right)
\]

\[
+ \left( \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right) \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right)
\]

\[
= \delta_l - e_{l-1}^T \Delta H_l \left( A + \Delta H_l^T \Delta H_l \right) \left( A + \Delta H_l^T \Delta H_l \right) \left( \Delta H_l^T e_{l-1} - B \Delta H_l^T \right)
\]

\[
= \delta_l - e_{l-1} \Delta H_l \Delta \beta_l
\]

\[
\leq 0
\]

(18)

so that we can easily get \( \| e_l \| \leq (r + \mu_k) \| e_{l-1} \| \).

Then, suppose \( \lim_{L \to \infty} \| e_{l-1} \| = \lim_{L \to \infty} \| e_l \| = a \). Notice that \( r + \mu_k < 1 \), so the formula \( a < (1 + \mu_k) a \) does not hold when \( L \to \infty \). Therefore, we have \( \lim_{L \to \infty} \| e_{l-1} \| = \lim_{L \to \infty} \| e_l \| = \lim_{L \to \infty} \| f - f_L \| = 0 \).

4.4. Algorithm implementation

The detailed implementation procedures of SCN+ are summarized as follows.

**Algorithm SCN+**

**Input:** A set of training data \( \{ (x_i, y_i) \mid x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^n, \forall 1 \leq i \leq N \} \); Set a nonlinear activation function \( G(\cdot) \), the maximum number of hidden nodes \( L_{\text{max}} \), tolerance error \( \varepsilon \), the user-specified coefficients \( C \) and \( \gamma \), the maximum times of random configuration \( T_{\text{max}} \); Choose a set of positive scalars \( \mathcal{Y} = \{ \lambda_{\text{min}} : \Delta A ; \lambda_{\text{max}} \} \);

1. Initialize \( e_0 = Y \), \( 0 < r < 1 \), \( \Omega = [ ] \); \( W = [ ] \); \( \tilde{W} = [ ] \);
2. While $L \leq L_{\text{max}}$ AND $\|e_0\| > \varepsilon$

1) Hidden node Parameters Configuration

3. For $\lambda \in \tilde{\gamma}$, Do

4. For $i = 1, 2, \ldots, T_{\text{max}}$, Do

5. Randomly assign $\omega_i$, $\tilde{b}_i$ and $\tilde{c}_i$ from $[-\lambda, \lambda]^T$, $[-\lambda, \lambda]^T$, $[-\lambda, \lambda]$ and $[-\lambda, \lambda]$;

6. Calculate $h_i$, $\tilde{h}_i$, $\xi_{L,q}$ based on () and ()

   Set $\mu_L = (1 - r)/(L + 1)$;

7. If $\min \{\xi_{L,1}, \xi_{L,2}, \ldots, \xi_{L,m}\} \geq 0$

8. Save $\omega_i$ and $b_i$ in $W$, $\tilde{\omega}_i$ and $\tilde{b}_i$ in $\tilde{W}$, $\xi_L = \sum_{q=1}^{n} \xi_{L,q}$ in $\Omega$;

9. Else go back to Step 4

10. End If

11. End For

12. If $W$ is not empty

13. Find $\omega^*_i$, $\tilde{\omega}^*_i$, $b^*_i$ and $\tilde{b}^*_i$ that maximize $\xi_L$ in $\Omega$;

   Set $H_L = [h^*_i, \tilde{h}^*_i, \ldots, h^*_i]$, $\tilde{H}_L = [\tilde{h}^*_i, \tilde{h}^*_i, \ldots, \tilde{h}^*_i]$;

14. Break

15. Else randomly take $r \in (0, 1 - r)$, renew $r = r + \tau$, return to Step 4;

16. End If

17. End For

2) Output Weights Determination

18. Calculate $\beta_L$ and $\tilde{\beta}_L$ based on (10) and (11);

   $e_L = e_{L-1} - \left( H_L \beta_L + \tilde{H}_L \tilde{\beta}_L \right)$;

19. Renew $e_0 = e_L$; $L = L + 1$;

20. End While

21. Return $\beta$, $\omega$ and $b$.

4. Performance evaluation

4.1. Evaluation on real-world datasets

In this section, several experiments are conducted on ten real-world datasets from KEEL. Fifty trials for each method are carried out, and the average results are reported. The experiments are conducted in MATLAB 2019b environment running on a PC that equips with an Intel(R) Core(TM) i7-9700, 3.0 GHz CPU, 8 GB RAM.

| Problems     | Training data | Testing data | Input | Normal features size | Privileged features size | Output |
|--------------|---------------|--------------|-------|----------------------|--------------------------|--------|
| Mortgage     | 700           | 346          | 15    | 8                    | 7                        | 1      |
There is no exact theoretical basis to guide the selection of hyperparameters $C$ and $\gamma$, which can be only set empirically or repeat the experiment. Taking SCN+ as an example, the experiment is repeated on the Wine dataset, and each experiment $C$ and $\gamma$ are taken in the increasing sequences $[10^2, 10^3, 1, 2, 5, 10]$ and $[10^2, 10^3, \ldots, 10^9]$. The optimal parameters are selected according to the training and testing RMSE. Seeing from Fig. 2, we can intuitively see how to set $C$ and $\gamma$ can get the best solution. From (a), we can obtain that when $C$ is set to 1 and $\gamma$ is set to $10^4$, the training accuracy is the biggest, but (b) the test accuracy at this time is not, indicating that the generalization performance is not at its best. Considering (a) and (b), $C$ set as $10^4$ and $\gamma$ as $10^9$ finally. In the following experiments, we will use this method to select the appropriate parameters $C$ and $\gamma$. Due to space limitations, the description of this process will be omitted later.

We compare with SCN, SCN+, IRVFL and IRVFL+ on five classification datasets and five regression datasets. The statistics of the datasets are illustrated in Table 1 and Table 2, including the number of training and test data, attributes, the number of the normal features, the number of the privileged features and classes or output. We randomly split attributes of each dataset mentioned above in half. All samples will be preprocessed with normalization.

The hyperparameters $C$ and $\gamma$ should be pre-defined by users. The settings of other parameters will be described in detail in Table 3. The user-defined parameters $C$ and $\gamma$ are chosen through random searches within $[10^2, 10^3]$ and $[10^2, 10^9]$.

### Table 1

| Problems     | Training data | Testing data | Input | Normal features size | Privileged features size | Classes |
|--------------|---------------|--------------|-------|----------------------|--------------------------|---------|
| Treasury     | 100           | 78           | 13    | 7                    | 6                        | 3       |
| Contraceptive| 1000          | 473          | 9     | 5                    | 4                        | 3       |
| Pima         | 500           | 268          | 8     | 4                    | 4                        | 2       |
| Flare        | 700           | 366          | 11    | 6                    | 5                        | 6       |
| ACA          | 400           | 290          | 14    | 7                    | 7                        | 2       |

### Table 2

Table of regression datasets.

### Table 3

We compare with SCN, SCN+, IRVFL and IRVFL+ on five classification datasets and five regression datasets. The statistics of the datasets are illustrated in Table 1 and Table 2, including the number of training and test data, attributes, the number of the normal features, the number of the privileged features and classes or output. We randomly split attributes of each dataset mentioned above in half. All samples will be preprocessed with normalization.

The hyperparameters $C$ and $\gamma$ should be pre-defined by users. The settings of other parameters will be described in detail in Table 3. The user-defined parameters $C$ and $\gamma$ are chosen through random searches within $[10^2, 10^3]$ and $[10^2, 10^9]$.

![Fig. 2. Performance of SCN+ with different user-defined parameters $C$ and $\gamma$ on Wine, where (a) and (b) represent training accuracy and testing accuracy respectively.](image)

**Remark 2.** In order to make the graph easy to observe, all the data in Fig. 2 adopt the log function to normalize the data.

In theory, there is no exact theoretical basis to guide the selection of hyperparameters $C$ and $\gamma$, which can be only set empirically or repeat the experiment. Taking SCN+ as an example, the experiment is repeated on the Wine dataset, and each experiment $C$ and $\gamma$ are taken in the increasing sequences $[10^2, 10^3, 1, 2, 5, 10]$ and $[10^2, 10^3, \ldots, 10^9]$. The optimal parameters are selected according to the training and testing RMSE. Seeing from Fig. 2, we can intuitively see how to set $C$ and $\gamma$ can get the best solution. From (a), we can obtain that when $C$ is set to 1 and $\gamma$ is set to $10^4$, the training accuracy is the biggest, but (b) the test accuracy at this time is not, indicating that the generalization performance is not at its best. Considering (a) and (b), $C$ set as $10^4$ and $\gamma$ as $10^9$ finally. In the following experiments, we will use this method to select the appropriate parameters $C$ and $\gamma$. Due to space limitations, the description of this process will be omitted later.
For classification datasets, all experiments will fix the maximum hidden layer nodes $L_{\text{max}}$ to compare the performance of each algorithm. As shown in Table 4, the training accuracy and test accuracy of SCN, SCN+, IRVFL, and IRVFL+ are given. Besides, experiments on datasets Wine, Contraceptive, Pima, Flare and ACA fixed the maximum hidden layer nodes as 50, 50, 25, 50 and 25 respectively. The authors can obtain that the performance of SCN and SCN+ is better remarkably than that of IRVFL and IRVFL+, which means that the supervision mechanism of SCN can effectively improve the generalization of the model. More importantly, SCN+ can always achieve the best performance on all above classification datasets. Fig. 3 shows the training and testing accuracy curves of SCN, SCN+, IRVFL and IRVFL+ on different classification datasets. It should be noted that each node of each graph is the average of fifty times of training or testing. Like the results in Table 4, the training and testing accuracy of SCN and SCN+ is always better than IRVFL and IRVFL+. Although the training accuracy of SCN+ is not significantly better than that of SCN, the test accuracy has been improved relatively well. It should be noted that whether it is in Table 4 or Fig. 3, SCN+ and IRVFL+ can always perform better than SCN and IRVFL respectively. The above experimental results indicate that privileged information is effective in improving model performance on classification cases.

Table 4 Comparisons with three approaches on classification datasets in terms of training accuracy and testing accuracy.

| Datasets | Algorithms | Training Accuracy (%) | Testing Accuracy (%) |
|----------|------------|-----------------------|----------------------|
|          |            | AVE       | DEV      | AVE       | DEV      |
| Wine     | SCN        | 98.36     | 0.66     | 82.54     | 1.46     |
| $L_{\text{max}}=50$ | SCN+       | 98.38     | 0.45     | 82.74     | 1.16     |
|          | IRVFL      | 94.72     | 2.46     | 78.87     | 4.16     |
|          | IRVFL+     | 95.04     | 1.94     | 80.08     | 4.92     |
| Contraceptive | SCN      | 51.31     | 0.89     | 49.34     | 1.06     |
| $L_{\text{max}}=50$ | SCN+       | 51.63     | 0.61     | 49.77     | 0.98     |
|          | IRVFL      | 50.09     | 1.62     | 47.28     | 2.23     |
|          | IRVFL+     | 50.34     | 1.68     | 47.91     | 1.97     |
| Pima     | SCN        | 75.46     | 0.95     | 75.52     | 0.93     |
| $L_{\text{max}}=25$ | SCN+       | 75.62     | 0.86     | 75.81     | 0.77     |
|          | IRVFL      | 66.66     | 1.27     | 70.51     | 1.55     |
|          | IRVFL+     | 66.99     | 1.33     | 70.65     | 1.72     |
| Flare    | SCN        | 66.87     | 3.01     | 61.17     | 1.61     |
| $L_{\text{max}}=50$ | SCN+       | 66.98     | 2.74     | 61.25     | 1.43     |
|          | IRVFL      | 61.85     | 4.53     | 59.05     | 2.90     |
|          | IRVFL+     | 61.91     | 4.24     | 59.10     | 2.41     |
| ACA      | SCN        | 67.13     | 0.80     | 66.61     | 1.22     |
| $L_{\text{max}}=25$ | SCN+       | 67.19     | 0.63     | 66.92     | 1.46     |
|          | IRVFL      | 66.15     | 1.86     | 64.28     | 2.39     |
|          | IRVFL+     | 66.33     | 1.74     | 64.51     | 1.81     |
Fig. 3. Performance of SCN, SCN+ and IRVFL on classification datasets, where (a) and (b) represent the performance on Wine, (c) and (d) represent the performance on Contraceptive, (e) and (f) represent the performance on Pima, (g) and (h) represent the performance on Flare, (i) and (j) represent the performance on ACA.
For regression datasets, all experiments will fix the tolerance error $\varepsilon$ to compare the performance of each algorithm. Table 5 shows the training RMSE, the test RMSE and the number of hidden layer nodes of each algorithm on different regression data sets under the condition of achieving the same expected error. The tolerance errors of datasets Mortgage, Treasury, ENB, Laser and W-Izmir are set to 0.1, 0.1, 0.2, 0.225 and 0.1 respectively. Expect on Mortgage the training RMSE of SCN perform better than SCN+, SCN+ can achieve best performance. More importantly, under the premise of achieving the same tolerance error, the hidden layer nodes required by SCN+ on Mortgage is 11.4% less than SCN, while IRVFL+ is 17.5% less than IRVFL; on Treasury, the hidden nodes number of SCN+ are 14.2% less than SCN, and IRVFL+ is 18.1% less than IRVFL; on ENB, the hidden nodes number of SCN+ are 4.0% less than SCN, and IRVFL+ is 4.6% less than IRVFL; on Laser, the hidden nodes number of SCN+ are 3.8% less than SCN, and IRVFL+ is 10.4% less than IRVFL; on W-Izmir, the hidden nodes number of SCN+ are 8.2% less than SCN, and IRVFL+ is 4.6% less than IRVFL. It can be seen that the SC algorithms can always get much smaller network structures than IRVFL and IRVFL+ under the condition of achieving the same tolerance error, which reduce the size of the models. Besides, SCN+ and IRVFL+ can get smaller model size than SCN and IRVFL respectively for adopting LUPI paradigm. Fig. 5 shows the curves of the training RMSE and testing RMSE of the four algorithms on different datasets. The authors can obtain that the SC algorithms always converges faster than IRVFL+ and IRVFL, except that SCN and IRVFL+ perform similarly on the ENB test set. Besides, under the premise of achieving the same desired solution, SCN+ requires fewer hidden nodes than SCN, while IRVFL+ requires more than SC algorithms and IRVFL requires the most. The above illustrates that SCN+ performs effectively on regression cases.

Table 5

| Datasets | Algorithms | Training RMSE | Testing RMSE | $L$ |
|----------|------------|---------------|--------------|-----|
|          |            | AVE | AVE | AVE | DEV |     |
| Mortgage | SCN        | 0.0962 | 0.0039 | 0.0956 | 0.0057 | 10.92 |
| $\varepsilon=0.1$ | SCN+       | 0.0963 | 0.0034 | 0.0949 | 0.0056 | 9.68  |
|          | IRVFL      | 0.1005 | 0.0036 | 0.1020 | 0.0043 | 131.38 |
| Treasury | SCN        | 0.0989 | 0.0012 | 0.1079 | 0.0034 | 19.08 |
| $\varepsilon=0.1$ | SCN+       | 0.0987 | 0.0015 | 0.1068 | 0.0032 | 16.38 |
|          | IRVFL      | 0.1044 | 0.0137 | 0.1118 | 0.0102 | 147.80 |
|          | IRVFL+     | 0.1008 | 0.0045 | 0.1133 | 0.0088 | 210.08 |
| ENB      | SCN+       | 0.1992 | 0.0010 | 0.2676 | 0.0245 | 23.86 |
| $\varepsilon=0.2$ | IRVFL      | 0.2030 | 0.0075 | 0.3132 | 0.0554 | 73.30 |
|          | IRVFL+     | 0.2023 | 0.0072 | 0.3120 | 0.0558 | 69.92 |
| Laser    | SCN        | 0.2238 | 0.0011 | 0.2336 | 0.0020 | 20.26 |
| $\varepsilon=0.225$ | SCN+       | 0.2237 | 0.0012 | 0.2335 | 0.0017 | 19.48 |
|          | IRVFL      | 0.2280 | 0.0075 | 0.2437 | 0.0073 | 66.78 |
|          | IRVFL+     | 0.2247 | 0.0028 | 0.2434 | 0.0073 | 59.84 |
| W-Izmir  | SCN        | 0.0964 | 0.0040 | 0.0998 | 0.0066 | 14.84 |
| $\varepsilon=0.1$ | SCN+       | 0.0942 | 0.0047 | 0.0990 | 0.0061 | 13.62 |
|          | IRVFL      | 0.1048 | 0.0093 | 0.1072 | 0.0112 | 149.74 |
|          | IRVFL+     | 0.1037 | 0.0080 | 0.1105 | 0.0116 | 142.90 |
Fig. 4. Performance of SCN, SCN+ and IRVFL on regression datasets, where (a) and (b) represent the performance on Mortgage, (c) and (d) represent the performance on Treasury, (e) and (f) represent the performance on ENB, (g) and (h) represent the performance on Laser, (i) and (j) represent the performance on W-Izmir.
4.2. Discussion

Based on the above experimental results, we discuss the following points of the proposed method:

1) Obviously the hyperparameters $C$ and $\gamma$ have a great impact on the performance of SCN+. Although the hyperparameters set for different tasks are not exactly the same, they all have one thing in common: $C$ must be small enough, and $\gamma$ must be large enough. As mentioned above, $CR_1\beta_1$ and $CR_2\beta_2$ are slack functions which means the role of $C$ is a parameter to correct the error that should not be set too large. According to the ridge regression theory, as $\gamma$ increases, the variance of the model will be smaller, but $\beta$ will deviate from the true value that lead a larger model deviation. Therefore, $\gamma$ should be set large enough but not too large which needs to be determined according to the actual situation.

2) The authors analyzed two algorithms from the perspective of complexity [26]. The highest orders of SCN+ reaches $O(23N^4+4Nm+4)$ but SCN just $O(2N(m+1))$ for the number of data $N$, which indicates that SCN+ will consume more computing resources, especially on large-scale data. How to improve the computational efficiency of SCN+ is the focus of our future research.

5. Conclusions

By using LUPI paradigm, an incremental learner model based on SCN is proposed, named SCN+. SCN+ leverages privileged information into the IRVFL networks in the training stage. The convergence of SCN+ has been analyzed in this paper, which provides a strong theoretical guarantee. Through experiments on six real-world datasets, it is shown that the proposed SCN+ can effectively improve the generalization performance of the model. This newly-derived SCN+ can merge the incremental learning algorithm with the LUPI paradigm effectively, which provides a way of thinking for the application of LUPI paradigm in incremental learning.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature. 521(7553) (2015) 436-444.
[2] M. Alhandoosh, D. Wang, Fast decorrelated neural network ensembles with random weight, Information sciences. 264 (2014) 104-117.
[3] B. Igelnik, Y.H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320-1329.
[4] S. Scardapane, D. Wang, Randomness in neural networks: an overview, Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery. 7(2) (2017) 1200.
[5] Y.H. Pao, GH. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector Functional-link net, Neurocomputing. 6 (2) (1994) 163-180.
[6] M. Li, D.H. Wang, Insights into randomized algorithms for neural networks: practical issues and common pitfalls, Information Sciences. 382 (2017) 170-178.
[7] A.N. Gorban, I.Y. Tyukin, D.V. Prokhorov, K.I. Sofeikov, Approximation with random bases: pro et contra, Information Sciences. 364 (2016) 129-145.
[8] D.H. Wang, M. Li, Stochastic configuration networks: fundamentals and algorithms, IEEE Transactions on Cybernetics. 47(10) (2017) 3466-3479.
[9] X.L. Zhu, X.C. Peng, W.W. Wang, X.X. Jia, R.Q. He, A further study on the inequality constraints in stochastic configuration networks, Information Sciences. 487 (2019) 77-83.
[10] D.H. Wang, C.H. Cui, Stochastic configuration networks ensemble for large-scale data analytics, biochemical treatment process, Information Sciences. 417 (2017) 55-71.
[11] Z.Y. Sheng, Z.Q. Zeng, H.Q. Qu, Y. Zhang, Optical fiber intrusion signal recognition method based on TSVD-SCN, Optical Fiber Technology. 48 (2019) 270-277.
[12] Q.J. Wang, C.Y. Yang, X.P. Ma, C.F. Zhang, S.M. Peng, Underground airflow quantity modeling based on SCN, Acta Automatica Sinica. DOI:10.16383/j.aas.c190602.
[13] D.H. Wang, M. Li, Deep stochastic configuration networks with universal approximation property, In 2018 International joint conference on neural networks.2018.
[14] M. Li, D.H. Wang, Two dimensional stochastic configuration networks for image data analytics, IEEE Transactions on cybernetics. 51(1) (2019) 359-372.
[15] D.H. Wang, M. Li, Robust stochastic configuration networks with kernel density estimation for uncertain data regression, Information Sciences. 412 (2017) 210-222.
[16] M. Li, C.Q. Huang, D.H. Wang, Robust stochastic configuration networks with maximum correntropy criterion for uncertain data regression, Information Sciences. 473 (2019) 73-86.
[17] M. Li, C. Huang, D. Wang, Robust stochastic configuration networks with maximum correntropy criterion for uncertain data regression, Information Sciences. 473 (2019) 73-86.
[18] V. Vapnik, A. Vashist, A new learning paradigm: Learning using privileged information, Neural networks. 122 (2009) 544-557.
[19] M. Lapin, M. Hein, B. Schiele, Learning using privileged information: SVM+ and weighted SVM, Neural networks the official journal of the international neural network society. 53 (2014) 95-108.
[20] X. Xu, W. Li, D. Xu, Distance metric learning using privileged information for face verification and person re-identification, IEEE Transactions on neural networks and learning systems. 26 (12) (2015) 3150.
[21] S. Fouad, P. Tino, S. Raychaudhury, Incorporating privileged information through metric learning, IEEE transactions on neural networks and learning systems. 24 (7) (2013) 1086-1098.
[22] V. Sharmanska, N. Quadrianto, C.H. Lampert, Learning to rank using privileged information, Proceedings of the IEEE international conference on computer vision, (2013) 825-832.
[23] P.B. Zhang, Z.X. Yang, A new learning paradigm for random vector functional-link network: RVFL+, Neural networks. 122 (2020) 94-105.
[24] M. Li, D.H. Wang, Insights into randomized algorithms for neural networks: practical issues and common pitfalls, Information Sciences. 382 (2017) 170-178.
[25] A.N. Gorban, I.Y. Tyukin, D.V. Prokhorov, K.I. Sofeikov, Approximation with random bases: pro et contra, Information Sciences. 364 (2016) 129-145.
[26] S. Li, Z.H. You, H. Guo, Inverse-free extreme learning machine with optimal information updating, IEEE transactions on cybernetics. 46 (5) (2015) 1229-1241.