Information Sciences 417 (2017) 55–71

Contents lists available at ScienceDirect

Information Sciences

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

Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics

Dianhui Wang*, Caihao Cui

Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia

ARTICLE INFO

Article history:
Received 8 December 2016
Revised 2 July 2017
Accepted 3 July 2017
Available online 4 July 2017

Keywords:
Stochastic configuration networks
Large-scale data analytics
Heterogeneous features
Ensemble learning
Negative correlation learning

ABSTRACT

This paper presents a fast decorrelated neuro-ensemble with heterogeneous features for large-scale data analytics, where stochastic configuration networks (SCNs) are employed as base learner models and the well-known negative correlation learning (NCL) strategy is adopted to evaluate the output weights. By feeding a large number of samples into the SCN base models, we obtain a huge sized linear equation system which is difficult to be solved by means of computing a pseudo-inverse used in the least squares method. Based on the group of heterogeneous features, the block Jacobi and Gauss–Seidel methods are employed to iteratively evaluate the output weights, and a convergence analysis is given with a demonstration on the uniqueness of these iterative solutions. Experiments with comparisons on two large-scale datasets are carried out, and the system robustness with respect to the regularizing factor used in NCL is given. Results indicate that the proposed ensemble learning techniques have good potential for resolving large-scale data modelling problems.

1. Introduction

Machine learning has received considerable attention over the past years due to its significant role for data analytics [13]. Under big data setting with decentralized information structure, advanced machine learning algorithms with robust and parallel implementations are needed along with the growth of data [24,25]. Various ensemble learning frameworks, aiming to improve the generalization performance of a learning system, have been developed over the last two decades, and many interesting ideas and theoretical works, including bagging, boosting, AdaBoost and random forests can be found in [1–7,9,12,15,16,21,23,26]. Generally speaking, learning-based ensembles share some common nature in system design, such as data sampling and the output integration. The basis of ensemble learning theory lies in a rational sampling implementation for building each base learner model, which may provide a sound predictability though learning a subset of the whole data set.

For neural network ensembles [5,12,16,23], the base models are trained by the error back-propagation (BP) algorithm and the regularizing factor used in the negative correlated cost function can be determined by the cross-validation method. Unfortunately, BP algorithm suffers from the sensitive setting of the learning rate, local minima and very slow convergence. Therefore, it is challenging to apply the existing ensemble methods for large-scale data sets. To overcome this problem, we employed random vector functional-link (RVFL) networks [11,19] to develop a fast decorrelated neuro-ensemble (termed

* Corresponding author.
E-mail address: dh.wang@latrobe.edu.au (D. Wang).

http://dx.doi.org/10.1016/j.ins.2017.07.003
0020-0255/

DNNE) in [1]. From our experience, DNNE can perform well on smaller data sets [1,15]. However, it is quite limited for dealing with large scale data because of its high computational complexity, the scalability of numerical algorithms for the least squares solution, and hardware constraint (here mainly referring to the PC memory). Recall that physical data may come from different types of sensors, localized information source or potential features extracted from multiple runs of some certain feature selection algorithms [6,10,17,18,20,22]. Thus, for large-scale data analytics, it is useful and significant to develop a generalized neuro-ensemble framework with heterogeneous features.

This paper is built on our previous work reported in [1], which is a specific implementation of the well-known NCL learning scheme using RVFL networks with a default scope setting of the random weights and biases. From theoretical statements on the universal approximation property in [11] and our empirical results on RVFL networks in [14], the default scope setting (i.e., $[-1, 1]$) for the random weights and biases cannot ensure the modelling performance at all. Therefore, readers should be aware of this pitfall and must be careful in making use of our code.[1] Limits of DNNE mainly come from the following aspects: (i) the system inputs are centralized or combined with different types of features; and (ii) the analysed method of computing the output weights becomes infeasible for large-scale data sets, which is related to the nature of the base learner model (i.e., the number of nodes at the hidden layer must be sufficiently large to achieve sound performance). To relax these constraints and emphasize on the fast building of neuro-ensembles with heterogeneous features, we generalize the classical NCL-based ensemble framework into a more general form, where a set of input features are feed into the SCN base models separately. This work also provides a feasible solution by using two iterative methods for evaluating the output weights of the SCN ensemble (SCNE). In addition, some analyses and discussions on the convergence of these iterative schemes are given through a demonstration on the correlations among the iterative solutions and the pseudo-inverse solution.

The remainder of the paper is organized as follows: Section 2 provides some technical supports, including the basics of the SCN model, a generalized version of the ensemble generalization error and the negative correlation learning scheme. Section 3 describes the proposed SCNE with heterogeneous features, details two iterative learning algorithms and discusses their convergence. Section 4 reports some experimental results on two large-scale data sets, including a robustness analysis on the system performance with respect to the regularizing factor used in NCL. Section 5 concludes this paper with some remarks on further studies.

2. Technical supports

This section briefly reviews the stochastic configuration networks, extends the ensemble generalization error with heterogeneous features, followed by the negative correlation learning scheme for building ensemble models.

2.1. Revisit of stochastic configuration networks

SCNs are a class of randomized learner models which are recently developed in [28]. The unique characteristics of the SCN model, different from the classical randomized learner model (i.e., RVFL networks), is the way of generating the random input weights and biases. In contrast to RVFL networks, SCNs are built incrementally according to a supervisory mechanism, which constrains the random input weights and biases to take values in a data-dependent territory, namely stochastic configuration support (SCS). This constructive approach for building SCNs guarantees the universal approximation property of the resulting SCN model for a given nonlinear map. For the sake of completeness, we revisit the main theoretical result in Theorem 1 below.

Given a target function $f : \mathcal{R}^d \rightarrow \mathcal{R}^m$. Suppose that an SCN model has already been built with $L-1$ hidden nodes, i.e., $f_{L-1} = \sum_{l=1}^{L-1} \beta_l \phi_l(\boldsymbol{w}_l^T \boldsymbol{x} + b_l)$ $(L = 1, 2, \ldots;\ f_0 = 0)$, where $\beta_l = [\beta_{l,1}, \beta_{l,2}, \ldots, \beta_{l,m}]^T$, and $\phi_l(\boldsymbol{w}_l^T \boldsymbol{x} + b_l)$ is an activation function of the lth hidden node with random input weights $\boldsymbol{w}_l$ and bias b_l. Denoted the residual error by $e_{L-1}^* = f - f_{L-1} = [e_{L-1,1}^*, \ldots, e_{L-1,m}^*]$, where $[\beta_1^*, \beta_2^*, \ldots, \beta_{L-1}^*] = \arg\min_\beta \|f - \sum_{l=1}^{L-1} \beta_l \phi_l\|$.

Let $\Gamma = \{\phi_1, \phi_2, \phi_3, \ldots\}$ be a set of real-valued functions, and span(Γ) denote a function space spanned by Γ; $L_2(D)$ denote the space of all Lebesgue measurable functions $f = [f_1, f_2, \ldots, f_m] : \mathcal{R}^d \rightarrow \mathcal{R}^m$ defined on $D \subset \mathcal{R}^d$, with the L_2 norm defined as

$$\|f\| = \left(\sum_{q=1}^{m} \int_D |f_q(x)|^2 dx \right)^{1/2} < \infty. \tag{1}$$

The inner product of $\theta = [\theta_1, \theta_2, \ldots, \theta_m] : \mathcal{R}^d \rightarrow \mathcal{R}^m$ and f is defined as

$$\langle f, \theta \rangle = \sum_{q=1}^{m} \langle f_q, \theta_q \rangle = \sum_{q=1}^{m} \int_D f_q(x) \theta_q(x) dx. \tag{2}$$

Theorem 1 (Wang and Li [28]). *Suppose that span(Γ) is dense in L_2 space and for any $\phi \in \Gamma$, $0 < \|\phi\| < b_\phi$ for some $b_\phi \in \mathcal{R}^+$. Given $0 < r < 1$ and a nonnegative real number sequence $\{\mu_L\}$ with $\lim_{L \rightarrow +\infty} \mu_L = 0$ subjected to $\mu_L \leq (1 - r)$. For $L = 1, 2, \ldots,$*

[1] http://homepage.cs.latrobe.edu.au/dwang/html/DNNEweb/index.html.

denoted by

$$\delta_L^* = \sum_{q=1}^{m} \delta_{L,q}^*, \quad \delta_{L,q}^* = (1 - r - \mu_L)\|e_{L-1,q}^*\|^2, \quad q = 1, 2, \ldots, m. \tag{3}$$

If the random basis function ϕ_L is generated to satisfy the following inequalities:

$$\langle e_{L-1,q}^*, \phi_L \rangle^2 \geq b_\phi^2 \delta_{L,q}^*, \quad q = 1, 2, \ldots, m, \tag{4}$$

and the output weights are evaluated by

$$[\beta_1^*, \beta_2^*, \ldots, \beta_L^*] = \arg\min_{\beta} \|f - \sum_{l=1}^{L} \beta_l \phi_l\|. \tag{5}$$

Then, we have $\lim_{L\to+\infty} \|f - f_L^*\| = 0$, where $f_L^* = \sum_{l=1}^{L} \beta_l^* \phi_l$, $\beta_l^* = [\beta_{l,1}^*, \beta_{l,2}^*, \ldots, \beta_{l,m}^*]^T$.

Given a training data set with N sample pairs $\{(\boldsymbol{x}_n, \boldsymbol{y}_n), n = 1, 2, \ldots, N\}$, where $\boldsymbol{x}_n \in \mathcal{R}^d$ and $\boldsymbol{y}_n \in \mathcal{R}^m$. Let $X \in \mathcal{R}^{N\times d}$ and $Y \in \mathcal{R}^{N\times m}$ represent the input and output data matrix, respectively; $e_{L-1}(X) \in \mathcal{R}^{N\times m}$ be the residual error matrix, where each column $e_{L-1,q}(X) = [e_{L-1,q}(\boldsymbol{x}_1), \ldots, e_{L-1,q}(\boldsymbol{x}_N)]^T \in \mathcal{R}^N$, $q = 1, 2, \ldots, m$. Denote the output vector of the Lth hidden node ϕ_L for the input X by

$$h_L(X) = [\phi_L(\boldsymbol{w}_L^T x_1 + b_L), \ldots, \phi_L(\boldsymbol{w}_L^T x_N + b_L)]^T. \tag{6}$$

Thus, the hidden layer output matrix of f_L can be expressed as $H_L = [h_1, h_2, \ldots, h_L]$. Denoted by

$$\xi_{L,q} = \frac{\left(e_{L-1,q}^T(X) \cdot h_L(X)\right)^2}{h_L^T(X) \cdot h_L(X)} - (1 - r - \mu_L) e_{L-1,q}^T(X) e_{L-1,q}(X), \quad q = 1, 2, \ldots, m. \tag{7}$$

The SC-III algorithm reported in [28] firstly generates a large pool of $T_{\max}$ candidate nodes, namely $\{\phi_L^{(1)}, \phi_L^{(2)}, \ldots, \phi_L^{(T_{\max})}\}$, in varying intervals. Then, it picks up those candidate nodes whose minimal value of the set $\{\xi_{L,1}, \ldots, \xi_{L,m}\}$ is positive. Then, the candidate node ϕ_L^* with the largest value of $\xi_L = \sum_{q=1}^{m} \xi_{L,q}$ will be assigned as the Lth hidden node for f_L. Thus, the output weight matrix of the SCN model, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]^T \in \mathcal{R}^{L\times m}$, could be computed by the standard least squares method, that is,

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \|H_L\boldsymbol{\beta} - Y\|_F^2 = H_L^\dagger Y, \tag{8}$$

where $H_L^\dagger$ is the Moore–Penrose generalized inverse of the matrix H_L, and $\|\cdot\|_F$ represents the Frobenius norm [8].

If there is no candidate node that satisfies these conditions, the SC algorithm will automatically increase the value of r to relax the constraints and further adjust the scoping interval to generate a new pool of candidate nodes. This process continues until the residual error decreases to a predefined tolerance τ. To speed up the procedure of building SCN models, we could add the top n_B ranked (according to the values of ξ_L) candidate nodes as a batch in each incremental loop of the SC algorithm. With a proper setting on n_B, the batch version of SC-III algorithm could greatly shorten the training time without weakening the prediction performance of the SCN model. Readers could find more details about the proof of the theorem and the SC algorithms in [28].

2.2. Generalization error of ensembles with heterogeneous features

For data regression, an ensemble model $\bar{f}$ that approximates an unknown target function g could be accomplished by fitting a collection of training samples $D_t = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$, where $\boldsymbol{x}_n \in \mathcal{R}^d$, $y_n \in \mathcal{R}$, and the sample pair $(\boldsymbol{x}_n, y_n)$ is an independent and identically distributed (i.i.d) sample from an unknown joint probability distribution $p(\boldsymbol{x}, y)$. Note that the training set D_t is a realization of a random sequence D that shares the same distribution $p(\boldsymbol{x}, y)$.

Suppose that the input $\boldsymbol{x}$ is concatenated by M parts: $\boldsymbol{x} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)})$, where $\boldsymbol{x}^{(m)} \in \mathcal{R}^{d_m}$, $\sum_{m=1}^{M} d_m = d$, and there exist functional relationships between the sub-features $\boldsymbol{x}^{(m)}$ and the measured output y, that is, $y = g_m(\boldsymbol{x}^{(m)}) + \epsilon^{(m)}$, where $\epsilon^{(m)}$ is the additive noise with zero mean ($E\{\epsilon^{(m)}\} = 0$) and the finite variance ($Var\{\epsilon^{(m)}\} = \sigma_{(m)}^2 < \infty$), $m = 1, 2, \ldots, M$. Thus, we have $y = g(\boldsymbol{x}) + \epsilon$, where $g(\boldsymbol{x}) = \frac{1}{M}\sum_{m=1}^{M} g_m(\boldsymbol{x}^{(m)})$, $E\{\epsilon\} = 0$ and $Var\{\epsilon\} = \frac{1}{M^2}\sum_{m=1}^{M} \sigma_{(m)}^2 = \sigma^2$ (under assumption that $\epsilon^{(1)}, \epsilon^{(2)}, \ldots, \epsilon^{(M)}$ are mutually independent random variables).

Let the $\{f_1, f_2, \ldots, f_M\}$ denote M base models, where the mth model f_m is separately trained on D_m, where $D_m = \{(\boldsymbol{x}_n^{(m)}, y_n), n = 1, 2, \ldots, N\}$. Usually, the parameters θ_m of the model f_m is estimated by

$$\theta_m^*(D_m) = \arg\min_{\theta_m} \frac{1}{N}\sum_{n=1}^{N} (f_m(\boldsymbol{x}_n^{(m)}; \theta_m) - y_n)^2. \tag{9}$$

Since the estimated θ_m^* depends on the given D_m, we write $\theta_m^*(D_m)$ to clarify the dependency of D_m. Therefore, an output value of the f_m for an input $\boldsymbol{x}^{(m)}$ should be written as $f_m(\boldsymbol{x}^{(m)}, \theta_m^*(D_m))$; for simplicity, denoted by $f_m(\boldsymbol{x}^{(m)}; D_m)$.

The ensemble output for an input $\boldsymbol{x}$ is defined as

$$\bar{f}(\boldsymbol{x}) = \sum_{m=1}^{M} a_m f_m(\boldsymbol{x}^{(m)}; D_m), \tag{10}$$

where $\sum_{m=1}^{M} a_m = 1$ and $0 < a_m < 1$, which is the weight of the mth base model. For the case of $a_m = 1/M$, the base models independently and equally contribute to the ensemble output, termed as the **Naive** method in this paper.

Let (X_0, Y_0) be a random sample, taken from D but independent of D_t, where $X_0 = [X_0^{(1)}, X_0^{(2)}, \ldots, X_0^{(M)}]$. In the light of the bias/variance decomposition by Geman and Bienenstock et al. [7], the ensemble generalization error expression by Ueda and Nakano [27], and the diversity discussion by Brown and Wyatt [4] for the ensemble modelling, we extend the result on the generalization error of the ensemble estimator as follows.

Theorem 2. *Let $G_E(\bar{f})$ denote the generalization error of an ensemble $\bar{f}$ in Eq. (10). Then, we have*

$$G_E(\bar{f}) = E_{X_0}\left\{\frac{1}{M}\overline{Var}(X_0) + \left(1 - \frac{1}{M}\right)\overline{Cov}(X_0) + \overline{Bias}(X_0)^2\right\} + \sigma^2, \tag{11}$$

where $\overline{Var}(X_0)$, $\overline{Cov}(X_0)$ and $\overline{Bias}(X_0)$ are the average conditional variance, conditional covariance and conditional bias of the M models with heterogeneous features for X_0, respectively, that is

$$\overline{Var}(X_0) = \frac{1}{M}\sum_{m=1}^{M} Var\left\{(f_m|X_0^{(m)})\right\}, \tag{12}$$

$$\overline{Bias}(X_0) = \frac{1}{M}\sum_{m=1}^{M} Bias\left\{(f_m|X_0^{(m)})\right\}, \tag{13}$$

$$\overline{Cov}(X_0) = \frac{1}{M(M-1)}\sum_{m=1}^{M}\sum_{q\neq m} Cov\left\{(f_m|X_0^{(m)}), (f_q|X_0^{(q)})\right\}. \tag{14}$$

where

$$Var\left\{(f_m|X_0^{(m)})\right\} = E_{D_t}\left\{\left(f_m(X_0^{(m)}; D_t) - E_{D_t}\left\{f_m(X_0^{(m)}; D_t)\right\}\right)^2\right\}, \tag{15}$$

$$Bias\left\{(f_m|X_0^{(m)})\right\} = E_{D_t}\left\{f_m(X_0^{(m)}; D_t)\right\} - g_m(X_0^{(m)}), \tag{16}$$

$$Cov\left\{(f_m, X_0^{(m)}), (f_q|X_0^{(q)})\right\} = E_{D_t}\left\{\left[f_m(X_0^{(m)}; D_t) - E_{D_t}\left\{f_m(X_0^{(m)}; D_t)\right\}\right]\left[f_q(X_0^{(q)}; D_t) - E_{D_t}\left\{f_q(X_0^{(q)}; D_t)\right\}\right]\right\}. \tag{17}$$

Therefore, managing the covariance term $\overline{Cov}(X_0)$ explicitly helps in controlling the divergence of the base models, leading to a better generalized ensemble. In practice, samples from a validation data set D_v, which is i.i.d to D_t, could represent the random sample pair (X_0, Y_0) in Theorem 2, and the validation error could be regarded as a realization of the generalization error of the ensemble.

2.3. Negative correlation learning

The negative correlation learning (NCL) is a typical training scheme to build neural network ensembles [7,9]. The key idea behind this learning algorithm lies in reducing the covariance among the base models while keeping the variance and bias terms of the ensemble not to be increased. Mathematically, the cost function of the mth base model over the training data D_m in NCL is given by

$$e_m = \sum_{n=1}^{N}\frac{1}{2}\left[(f_m(\boldsymbol{x}_n^{(m)}) - y_n)^2 + \lambda p_m(\boldsymbol{x}_n)\right], \tag{18}$$

where

$$p_m(\boldsymbol{x}_n) = (f_m(\boldsymbol{x}_n^{(m)}) - \bar{f}(\boldsymbol{x}_n))\sum_{q\neq m}^{M}(f_q(\boldsymbol{x}_n^{(q)}) - \bar{f}(\boldsymbol{x}_n)), \tag{19}$$

and $0 < \lambda < 1$ is the regularizing factor. Note that

$$\sum_{q\neq m}^{M}(f_q(\boldsymbol{x}_n^{(q)}) - \bar{f}(\boldsymbol{x}_n)) = -(f_m(\boldsymbol{x}_n^{(m)}) - \bar{f}(\boldsymbol{x}_n)). \tag{20}$$

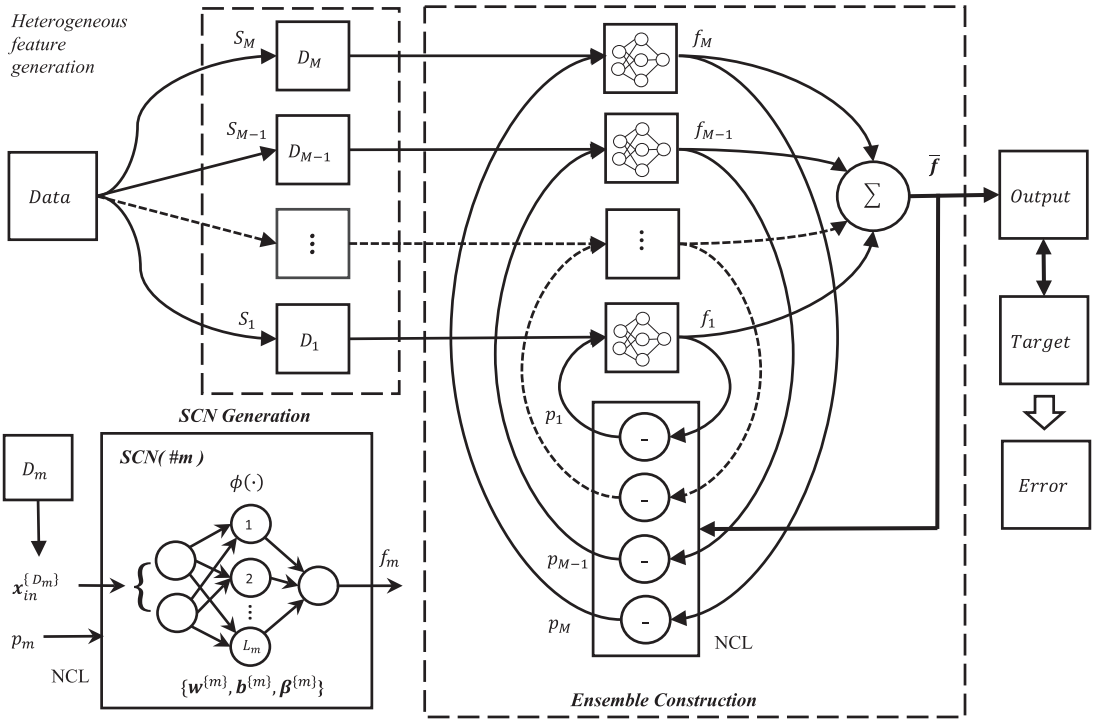


Fig. 1. Framework of the proposed SCNE with heterogeneous features.

Hence, Eq. (18) can be rewritten as

$$e_m = \sum_{n=1}^{N} \frac{1}{2}\left[(f_m(\boldsymbol{x}_n^{(m)}) - \boldsymbol{y}_n)^2 - \lambda (f_m(\boldsymbol{x}_n^{(m)}) - \bar{f}(\boldsymbol{x}_n)^2 \right], m = 1, 2, \ldots, M. \tag{21}$$

The NCL scheme aims to find all the parameters (including the weights, biases and regularizing factor) of the M models through minimizing every e_m.

3. Stochastic configuration networks ensemble

In [1], RVFL networks are employed as the base models to build a neural network ensemble where all the base models share the same input and identical architecture. In the construction of the DNNE, we used the pseudo-inverse method to evaluate the output weights of RVFL models. However, RVFL networks cannot perform at all if the scope of the random weights and biases is not chosen properly [14,28,29]. Also, it is difficult to estimate the number of hidden node of each RVFL base model in the ensemble. Due to the merits of the SCN model for overcoming these issues associated with RVFL networks, we employ SCNs as the base learner models with heterogeneous features to build SCNE.

For large-scale data modelling, the pseudo-inverse method used in the DNNE for evaluating the output weights of the ensemble is limited. Therefore, the **block Jacobi method** and **block Gauss–Seidel method** [30], are used to calculate the output weights of SCNE model. The proposed SCNE framework is shown in Fig. 1. The original data set D is partitioned into M subsets according to the heterogeneous feature sets, denoted by $\{S_m, m = 1, 2, \ldots, M\}$. Then, the corresponding sample data set for the mth model is denoted as $D_m = \{(\boldsymbol{x}_n^{(m)}, y_n), n = 1, 2, \ldots, N\}$, where $\boldsymbol{x}_n^{(m)} \in \mathcal{R}^{d_m}$ and $y_n \in \mathcal{R}$. Each set will be used to build the SCN base model in the ensemble. The partition of the full feature set could be done by a prior knowledge or various feature selection schemes.

The procedure of building the SCNE model could be divided into two steps: (i) the SCN base model generation, that is, each SCN model will be built independently according to the SC-III algorithm in [28]; and (ii) the ensemble construction by means of the NCL algorithm, where the random weights and biases of the SCN models generated in the first step are fixed.

3.1. SCN Base model generation

The SCN base models can be constructed individually with L_m nodes in the hidden layer on the data set D_m, and the output weights $\boldsymbol{\beta}_m$ are also calculated according the stochastic configuration algorithm (SC-III) in [28]. One of the important issues in building SCNs is associated with the overfitting problem when too many hidden nodes are added incrementally.

Fortunately, this problem can be solved by monitoring the modelling performance over a validation set with a quick pruning method. Alternatively, we can apply the trial-and-error method to determine an appropriate number of the hidden nodes of the SCN base model.

3.2. SCNE Construction

In the ensemble construction, the input weights and biases of the base models are fixed. The output weights $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_M$ of the SCNE model can be obtained by using the NCL scheme, that is,

$$\{\boldsymbol{\beta}_1^*, \boldsymbol{\beta}_2^* \ldots, \boldsymbol{\beta}_M^*\} = \arg\min_{\boldsymbol{\beta}_m^*}\{e_m\}, m = 1, 2, \ldots, M. \tag{22}$$

The following sections detail the pseudo-inverse method and two iterative methods for computing the output weights. To understand some properties of the solutions obtained by these three methods, a demonstration over a 2-D function approximation is given to see the time cost and the impact of the regularized model on the correlationship among the solutions.

3.2.1. Analytical solution

By Eq. (21), the cost function e_m of the SCN model f_m with the NCL penalty term could be rewritten in the following matrix form:

$$e_m = \frac{1}{2}\left(\|H_m\boldsymbol{\beta}_m - \boldsymbol{y}\|^2 - \lambda\|H_m\boldsymbol{\beta}_m - \frac{1}{M}\boldsymbol{H}\boldsymbol{B}\|^2\right), \tag{23}$$

where

$$H_m = \begin{bmatrix} \phi_1(\boldsymbol{x}_1^{(m)}) & \ldots & \phi_{L_m}(\boldsymbol{x}_1^{(m)}) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_N^{(m)}) & \ldots & \phi_{L_m}(\boldsymbol{x}_N^{(m)}) \end{bmatrix}_{N\times L_m}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M \end{bmatrix}_{\boldsymbol{L}\times 1}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}_{N\times 1}. \tag{24}$$

The H_m is the output matrix at the hidden layer of the mth base model; $\boldsymbol{B}$ is the output weighs of the whole ensemble (all the output weights of the base models); $\boldsymbol{y}$ is the target; $\boldsymbol{H} = [H_1, H_2, \ldots, H_M]_{N\times\boldsymbol{L}}$; $\boldsymbol{L}$ is the total number of the hidden nodes the SCNE model. Simple computations give that

$$\frac{\partial e_m}{\partial \beta_m} = c_1 H_m^T H_m \boldsymbol{\beta}_m + c_2 H_m^T \tilde{\boldsymbol{H}}_m \boldsymbol{B} - H_m^T \boldsymbol{y} = 0, \quad m = 1, 2, \ldots, M, \tag{25}$$

where $\tilde{\boldsymbol{H}}_m = \left[H_1, \ldots, H_{m-1}, \boldsymbol{0}_{N\times L_m}, H_{m+1}, \ldots, H_M\right]_{N\times\boldsymbol{L}}$, and

$$c_1 = 1 - \frac{\lambda(M-1)^2}{M^2}, \quad c_2 = \frac{\lambda(M-1)}{M^2}. \tag{26}$$

From Eq. (25), a huge sized linear equation system can be obtained, that is,

$$\mathbb{H}\boldsymbol{B} = \boldsymbol{H}^T\boldsymbol{y}, \tag{27}$$

where

$$\mathbb{H} = \begin{bmatrix} c_1 H_1^T H_1 & c_2 H_1^T H_2 & \ldots & c_2 H_1^T H_M \\ c_2 H_2^T H_1 & c_1 H_2^T H_2 & \ldots & c_2 H_2^T H_M \\ \vdots & \vdots & \ddots & \vdots \\ c_2 H_M^T H_1 & c_2 H_M^T H_2 & \ldots & c_1 H_M^T H_M \end{bmatrix}_{\boldsymbol{L}\times\boldsymbol{L}}. \tag{28}$$

If all the $H_m^T H_m$s are invertible, which means all the SCN models are built with a full column rank of H_m, then $\mathbb{H}$ is invertible if the regularizing factor λ takes a sufficiently smaller value. In this case, $\boldsymbol{B} = \mathbb{H}^{-1}\boldsymbol{H}^T\boldsymbol{y}$. However, we cannot ensure full rank condition for every matrix $H_m^T H_m$ in system implementation. In such a case, the output weights of the SCNE model can be evaluated by

$$\boldsymbol{B} = \mathbb{H}^{\dagger}\boldsymbol{H}^T\boldsymbol{y}, \tag{29}$$

where the $\mathbb{H}^{\dagger}$ represents the pseudo-inverse of the matrix $\mathbb{H}$ [8].

3.2.2. Iterative solutions

Due to some constraints on either the numerical algorithms for computing the pseudo inverse of a huge sized matrix or computing device with limited RAM memory, the analytical solution described in Algorithm 1 becomes impractical and/or time-consuming for large-scale datasets. Therefore, it is necessary and important to develop iterative algorithms for problem solving. Instead of evaluating the whole output weights $\boldsymbol{B}$ at once, we update gradually the $\boldsymbol{\beta}$ of each SCN base model to

Algorithm 1: Pseudo-inverse method for building SCNE.

Input : M, $\{L_m\}$, $\{D_m\} = \{(\boldsymbol{x}_n^{(m)}, \boldsymbol{y}), n = 1, 2, \ldots, N\}$, $\boldsymbol{x}_n^{(m)} \in \mathcal{R}^{d_m}$, $\boldsymbol{y} \in \mathcal{R}$; λ;
Output: SCNE.

1 **SCN generation by SC-III**
2 **Calculate all** $\{H_m\}$ **and** $\boldsymbol{H}$ (Eq. (24))
3 **for** $m \leftarrow 1$ *to* M **do**
4 | **for** $n \leftarrow 1$ *to* N, $l \leftarrow 1$ *to* L_m **do**
5 | | $H_m(n, l) = \phi_l(\boldsymbol{x}_n^{(m)})$
6 | **end**
7 | $\boldsymbol{H} := [\boldsymbol{H}, H_m]$
8 **end**
9 **Calculate the coefficients** c_1 **and** c_2 (Eq. (26))
10 $c_1 = 1 - \lambda(M-1)^2/M^2$, $\quad c_2 = \lambda(M-1)/M^2$
11 **Calculate the large matrix** $\mathbb{H}$ (Eq. (28))
12 **for** $m, q \leftarrow 1$ *to* M **do**
13 | **if** $m \neq q$ **then**
14 | | $\mathbb{H}(m, q) = c_2 H_m^T H_q$
15 | **else**
16 | | $\mathbb{H}(m, q) = c_1 H_m^T H_q$
17 | **end**
18 **end**
19 **return** $\boldsymbol{B} = \mathbb{H}^{\dagger}\boldsymbol{H}^T\boldsymbol{y}$ (Eq. (29)).

reduce the computing time and memory cost. The minimum of the cost function Eq. (23) with respect to the local output weights $\boldsymbol{\beta_m}$ at kth iteration can be obtained by solving the following equation:

$$\frac{\partial e_m^{(k)}}{\partial \boldsymbol{\beta}_m} = \frac{\partial\left(\|H_m\boldsymbol{\beta}_m - \boldsymbol{y}\|^2 - \lambda\|H_m\boldsymbol{\beta}_m - \frac{1}{M}\boldsymbol{H}\boldsymbol{B}_{m,X}^{(k)}\|^2\right)}{2\partial\boldsymbol{\beta}_m} = 0, \tag{30}$$

where $X \in \{J, G\}$, indexing the **Jacobi** and **Gauss–Seidel** iteration schemes, respectively [30].

Taking the initial values of the output weights as $\boldsymbol{B}^{(0)} = \left[H_1^{\dagger}\boldsymbol{y}, H_2^{\dagger}\boldsymbol{y}, \ldots, H_M^{\dagger}\boldsymbol{y}\right]^T$. From Eq. (30), we can derive the following block iterative algorithms:

$$\boldsymbol{\beta}_{m,J}^{(k)} = \frac{1}{c_1}H_m^{\dagger}\left(\boldsymbol{y} - c_2\tilde{\boldsymbol{H}}_m\boldsymbol{B}_{m,J}^{(k)}\right), \quad \boldsymbol{B}_{m,J}^{(k)} = \left[\boldsymbol{\beta}_1^{(k-1)}, \ldots, \boldsymbol{\beta}_{m-1}^{(k-1)}, \boldsymbol{\beta}_m, \boldsymbol{\beta}_{m+1}^{(k-1)}, \ldots, \boldsymbol{\beta}_M^{(k-1)}\right]^T, \tag{31}$$

$$\boldsymbol{\beta}_{m,G}^{(k)} = \frac{1}{c_1}H_m^{\dagger}\left(\boldsymbol{y} - c_2\tilde{\boldsymbol{H}}_m\boldsymbol{B}_{m,G}^{(k)}\right), \quad \boldsymbol{B}_{m,G}^{(k)} = \left[\boldsymbol{\beta}_1^{(k)}, \ldots, \boldsymbol{\beta}_{m-1}^{(k)}, \boldsymbol{\beta}_m, \boldsymbol{\beta}_{m+1}^{(k-1)}, \ldots, \boldsymbol{\beta}_M^{(k-1)}\right]^T, \tag{32}$$

The pseudo codes of these two algorithms are given in Algorithms 2 and 3 below. The main difference between these two schemes occurs at line 13. In practice, we provide a **tolerance** τ and a **maximum iteration number** $k_{\max}$ experimentally to guarantee that the algorithms will be terminated if either the SCNE error $E_{ens} < \tau$ or the iteration time reaches the $k_{\max}$. Note that Eqs. (31) and (32) are rewritten in discrete summation forms in the algorithms for the convenience of programming.

Remark 1. Suppose that every H_m has full column rank. Then, under some certain conditions, we can prove that the block Jacobi and block Gauss–Seidel methods [30] converge to the same solution as obtained by the pseudo-inverse method. To do so, let us rewrite the block Jacobi iterative scheme in the following matrix form:

$$\boldsymbol{B}_J^{(k)} = \mathbb{D}^{-1}(\boldsymbol{H}^T\boldsymbol{y} - \mathbb{R}\boldsymbol{B}_J^{(k-1)}), \tag{33}$$

where

$$\mathbb{D} = \begin{bmatrix} c_1H_1^TH_1 & 0 & \ldots & 0 \\ 0 & c_1H_2^TH_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & c_1H_M^TH_M \end{bmatrix}, \quad \mathbb{R} = \begin{bmatrix} 0 & c_2H_1^TH_2 & \ldots & c_2H_1^TH_M \\ c_2H_2^TH_1 & 0 & \ldots & c_2H_2^TH_M \\ \vdots & \vdots & \ddots & \vdots \\ c_2H_M^TH_1 & c_2H_M^TH_2 & \ldots & 0 \end{bmatrix}. \tag{34}$$

Similarly, the block Gauss–Seidel iterative scheme can be rewritten as

$$\boldsymbol{B}_G^{(k)} = \mathbb{L}^{-1}(\boldsymbol{H}^T\boldsymbol{y} - \mathbb{U}\boldsymbol{B}_G^{(k-1)}), \tag{35}$$

Algorithm 2: Block Jacobi method for building SCNE.

Input : M, $\{L_m\}$, $\{D_m\} = \{(\boldsymbol{x}_n^{(m)}, \boldsymbol{y}), n = 1, 2, \ldots, N\}$, $\boldsymbol{x}_n^{(m)} \in \mathcal{R}^{d_m}$, $\boldsymbol{y} \in \mathcal{R}$; τ; k_{max};
Output: SCNE.

SCN generation by SC-III
Calculate all $\{H_m\}$ and $\{\boldsymbol{\beta}_m^{(0)}\}$
for $m \leftarrow 1$ *to* M **do**
  for $n \leftarrow 1$ *to* N, $l \leftarrow 1$ *to* L_m **do**
    $H_m(n, l) = \phi_l(\boldsymbol{x}_n^{(m)})$
  end
  $\boldsymbol{\beta}_m^{(0)} = H_m^{\dagger}\boldsymbol{y}$
end
Calculate the coefficients c_1 **and** c_2
$c_1 = 1 - \lambda(M-1)^2/M^2, \quad c_2 = \lambda(M-1)/M^2$
for $k \leftarrow 1$ *to* k_{max} **do**
  for $m \leftarrow 1$ *to* M **do**
    $\boldsymbol{\beta}_{m,J}^{(k)} = \frac{1}{c_1} H_m^{\dagger}\left(\boldsymbol{y} - c_2 \sum_{q \neq m} H_q \boldsymbol{\beta}_q^{(k-1)}\right)$ (Eq. (31))
    Update $\boldsymbol{B}_J^k$ with $\boldsymbol{\beta}_{m,J}^k$
  end
  if $E_{ens}^{(k)} < \tau$ **then**
    break
  end
end
return $\boldsymbol{B}_J \leftarrow \boldsymbol{B}_J^k$.

Algorithm 3: Block Gauss–Seidel method for building SCNE.

Input : M, $\{L_m\}$, $\{D_m\} = \{(\boldsymbol{x}_n^{(m)}, \boldsymbol{y}), n = 1, 2, \ldots, N\}$, $\boldsymbol{x}_n^{(m)} \in \mathcal{R}^{d_m}$, $\boldsymbol{y} \in \mathcal{R}$; ϕ; α; λ; τ; k_{max};
Output: SCNE.

SCN generation by SC-III
Calculate all $\{H_m\}$ and $\{\boldsymbol{\beta}_m^{(0)}\}$
for $m \leftarrow 1$ *to* M **do**
  for $n \leftarrow 1$ *to* N, $l \leftarrow 1$ *to* L_m **do**
    $H_m(n, l) \leftarrow \phi_l(\boldsymbol{x}_n^{(m)})$
  end
  $\boldsymbol{\beta}_m^{(0)} = H_m^{\dagger}\boldsymbol{y}$
end
Calculate the coefficients c_1 **and** c_2
$c_1 = 1 - \lambda(M-1)^2/M^2, \quad c_2 = \lambda(M-1)/M^2$
for $k \leftarrow 1$ *to* k_{max} **do**
  for $m \leftarrow 1$ *to* M **do**
    $\boldsymbol{\beta}_{m,G}^{(k)} = \frac{1}{c_1} H_m^{\dagger}\left[\boldsymbol{y} - c_2\left(\sum_{q<m} H_q \boldsymbol{\beta}_q^{(k)} + \sum_{q>m} H_q \boldsymbol{\beta}_q^{(k-1)}\right)\right]$(Eq. (32))
    Update $\boldsymbol{B}_G^k$ with $\boldsymbol{\beta}_{m,G}^k$
  end
  if $E_{ens}^{(k)} < \tau$ **then**
    break
  end
end
return $\boldsymbol{B}_G \leftarrow \boldsymbol{B}_G^k$.

where

$$\mathbb{L} = \begin{bmatrix} c_1 H_1^T H_1 & 0 & \dots & 0 \\ c_2 H_2^T H_1 & c_1 H_2^T H_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_2 H_M^T H_1 & c_2 H_M^T H_2 & \dots & c_1 H_M^T H_M \end{bmatrix}, \quad \mathbb{U} = \begin{bmatrix} 0 & c_2 H_1^T H_2 & \dots & c_2 H_1^T H_M \\ 0 & 0 & \dots & c_2 H_2^T H_M \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}. \tag{36}$$

Suppose that $\boldsymbol{B}_J^* := \lim_{k\to\infty} \boldsymbol{B}_J^{(k)}$ and $\boldsymbol{B}_G^* := \lim_{k\to\infty} \boldsymbol{B}_G^{(k)}$ exist. Then, from Eq. (33), the following holds

$$\lim_{k\to\infty} \boldsymbol{B}_J^{(k)} = \lim_{k\to\infty} \mathbb{D}^{-1}(\boldsymbol{H}^T\boldsymbol{y} - \mathbb{R}\boldsymbol{B}_J^{(k-1)}). \tag{37}$$

Therefore, $(\mathbb{D}+\mathbb{R})\boldsymbol{B}_J^* = \boldsymbol{H}^T\boldsymbol{y}$. Similarly, we have $(\mathbb{L}+\mathbb{U})\boldsymbol{B}_G^* = \boldsymbol{H}^T\boldsymbol{y}$.

Note that $\mathbb{H} = \mathbb{D}+\mathbb{R} = \mathbb{L}+\mathbb{U}$, where $\mathbb{L}$ is lower triangular component of $\mathbb{H}$, $\mathbb{U}$ is the strictly upper triangular component, $\mathbb{D}$ is a diagonal component of $\mathbb{H}$, and $\mathbb{R}$ is the remainder. Thus, we have

$$(\mathbb{D}+\mathbb{R})\boldsymbol{B}_J^* = (\mathbb{L}+\mathbb{U})\boldsymbol{B}_G^* = \mathbb{H}\boldsymbol{B} = \boldsymbol{H}^T\boldsymbol{y}. \tag{38}$$

From Eq. (38), we get $\mathbb{H}(\boldsymbol{B}_J^* - \boldsymbol{B}) = \mathbb{H}(\boldsymbol{B}_G^* - \boldsymbol{B}) = \boldsymbol{0}$. If $\mathbb{H}$ is nonsingular, it is easy to see that $\boldsymbol{B}_J^* = \boldsymbol{B}_G^* = \boldsymbol{B}$. Thus, we conclude that the two iterative schemes converge to the same solution from the pseudo-inverse method provided that (i) every H_m is full column rank; (ii) the regularizing factor λ is chosen so that the spectral radius of the iteration matrix is less than 1, that is,

$$\rho(\mathbb{D}^{-1}\mathbb{R}) < 1 \quad or \quad \rho(\mathbb{L}^{-1}\mathbb{U}) < 1. \tag{39}$$

By using Greschgorin Circle Theorem [8] for block matrices, we know that any eigenvalue z of the matrix $\mathbb{D}^{-1}\mathbb{R}$ must be constrained by one of the following inequalities:

$$|z| \le \frac{c_2}{c_1} \sum_{q\neq m} \|(H_m^T H_m)^{-1} H_m^T H_q\|, m = 1, 2, \dots, M. \tag{40}$$

Hence, a theoretical estimate of the regularizing factor λ should be subjected to $c_2 c_1^{-1}\theta_0 < 1$, where

$$\theta_0 = \max\{\sum_{q\neq m} \|(H_m^T H_m)^{-1} H_m^T H_q\|, q = 1, 2, \dots, M\}. \tag{41}$$

Simple computations give that

$$0 < \lambda < \frac{M^2}{(M-1)(M+\theta_0-1)}. \tag{42}$$

Remark 2. In practice, the full rank condition is hard to meet for all the base models. Thus, the convergence and uniqueness of the iterative solutions cannot be ensured. To fix this problem, we can employ a regularization model to resolve the linear model in Eq. (27), resulting in a solution as $\boldsymbol{B} = \mathbb{H}_r^{-1}\boldsymbol{H}^T\boldsymbol{y}$, where

$$\mathbb{H}_r = \begin{bmatrix} c_1 H_1^T H_1 & c_2 H_1^T H_2 & \dots & c_2 H_1^T H_M \\ c_2 H_2^T H_1 & c_1 H_2^T H_2 & \dots & c_2 H_2^T H_M \\ \vdots & \vdots & \ddots & \vdots \\ c_2 H_M^T H_1 & c_2 H_M^T H_2 & \dots & c_1 H_M^T H_M \end{bmatrix} + c_1 \begin{bmatrix} r_1 I_1 & 0 & \dots & 0 \\ 0 & r_2 I_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_M I_M \end{bmatrix}, \tag{43}$$

here r_m is a positive regularization parameter of the mth SCN base model, and I_m is a $L_m \times L_m$ identity matrix, $m = 1, 2, \dots, M$. Obviously, the $\mathbb{H}_r$ is invertible. For the iterative methods, each $H_m^\dagger$ is replaced by $(H_m^T H_m + r_m I_m)^{-1} H_m^T$ in Eqs. (31) and (32), the new initial $\boldsymbol{B^{(0)}}$ and the iterative equations of $\boldsymbol{\beta}_m$ can be updated as follows:

$$\boldsymbol{B}^{(0)} = \begin{bmatrix} \boldsymbol{\beta}_1^{(0)} \\ \vdots \\ \boldsymbol{\beta}_M^{(0)} \end{bmatrix} = \begin{bmatrix} (H_1^T H_1 + r_1 I_1)^{-1} H_1^T \boldsymbol{y} \\ \vdots \\ (H_M^T H_M + r_M I_M)^{-1} H_M^T \boldsymbol{y} \end{bmatrix}, \tag{44}$$

$$\boldsymbol{\beta}_{m,J}^{(k)} = \frac{1}{c_1}(H_m^T H_m + r_m I_m)^{-1} H_m^T \left(\boldsymbol{y} - c_2 \tilde{\boldsymbol{H}}_m \boldsymbol{B}_{m,J}^{(k)}\right), \tag{45}$$

$$\boldsymbol{\beta}_{m,G}^{(k)} = \frac{1}{c_1}(H_m^T H_m + r_m I_m)^{-1} H_m^T \left(\boldsymbol{y} - c_2 \tilde{\boldsymbol{H}}_m \boldsymbol{B}_{m,G}^{(k)}\right). \tag{46}$$

To reduce the algorithm complexity, we take equal values for $r_1, r_2, \dots, r_M$, namely r, in Eq. (43). Adjusting the r will control the l_2-norm of the output weights of the ensemble model, especially when dealing a large number of hidden nodes. It is interesting to look into the impact of such a parameter on the modelling performance. It is still unclear on this point, and a further research is being expected in this direction.

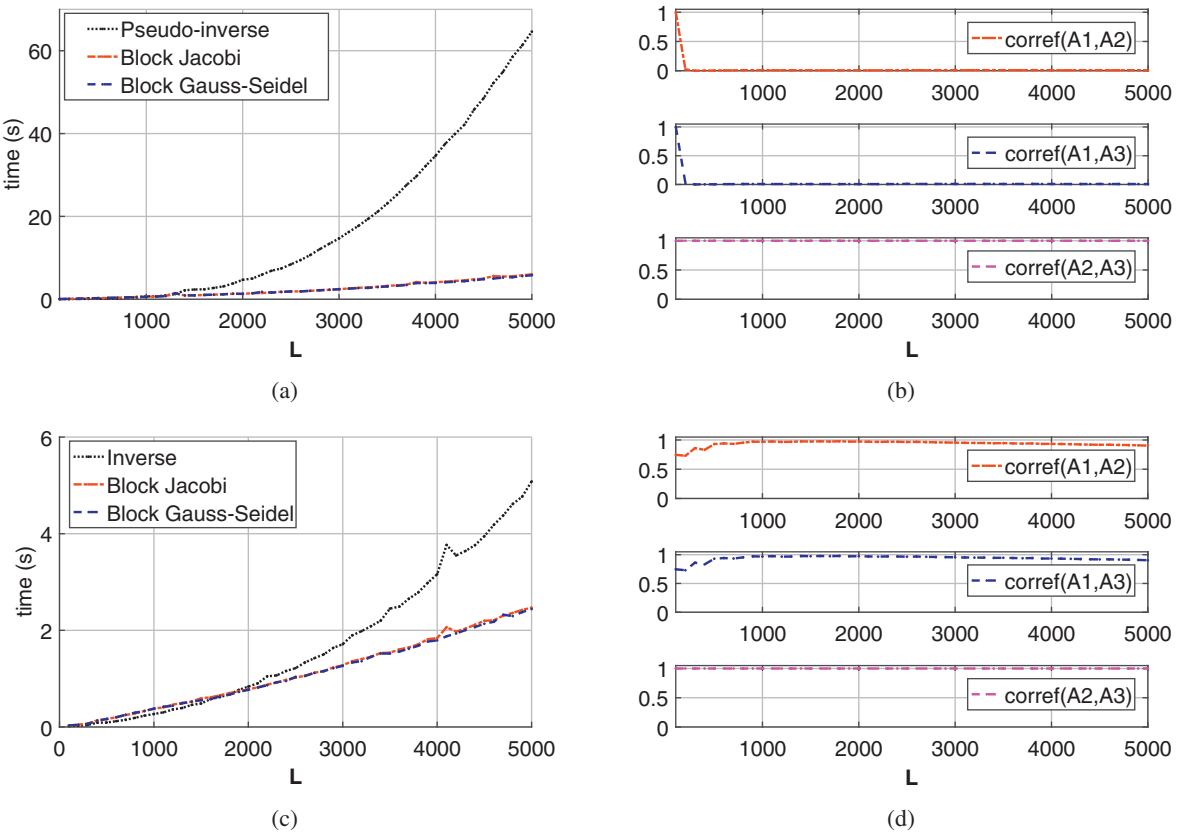


Fig. 2. Comparisons of different algorithms (a) Ensemble construction time by Algorithm 1–3; (b) Correlation coefficients of **B**s from Algorithm 1–3 (c) Ensemble construction time of Algorithm 1–3 with regularization ($r = 0.1$) (d) Correlation coefficients of the **B**s from Algorithm 1–3 with regularization.

3.3. Demonstration

A synthetic dataset is used to demonstrate the training time along with the number of the hidden nodes $\boldsymbol{L}$ in constructing the SCNEs with the pseudo-inverse, block Jacobi and block Gauss–Seidel methods, respectively. The reason behind this set-up is that a normal PC can run this task as the $\boldsymbol{L}$ grows up, and the memory cost of the computer is controllable compared to the large datasets. In this section, we report two sets of results obtained from the original SCNE and the modified SCNE with a regularization factor $r = 0.1$. From this comparison, we can infer that a similar consequence holds for large-scale datasets.

The synthetic dataset $D = \{(x_{1n}, x_{2n}, y_n), n = 1, \ldots, 5000\}$ contains 4000 training samples and 1000 test samples, all generated by a simple function $y = cos(2 \times x_2)/e^{x_1}$ where $x_2 = \sin(x_1)$, $x_1 \in [-5, 5]$. This dataset is easy to learn by the SCNE, so we omit the training and test results in the context in order to focus on the time comparisons. In this demonstration, all the SCN base models use the same training datasets. By setting the $M = 10$ (10 base models), $\lambda = 0.1$, $\tau = 10^{-6}$, $k_{\max} = 5$, and gradually increasing the hidden node of each base model L_m from 10 to 500, the computing time spent on SCNEs construction by using the three algorithms is presented in Fig. 2a (original) and Fig. 2c (regularized), respectively.

It is clear to see that the computing time of the pseudo-inverse method grows exponentially, while the computing time of the iterative methods grow linearly. The iterative methods may cost more time when the hidden node is small, but when $\boldsymbol{L}$ becomes larger ($\boldsymbol{L} > 2000$ in this demonstration), the pseudo-inverse approach is clearly inefficient and infeasible for large-scale datasets.

With regard to the convergence of the output weights $\boldsymbol{B}$ from different algorithms, Fig. 2b shows the correlations between these three algorithms. Here, A1, A2 and A3 represent the pseudo-inverse, the block Jacobi and the block Gauss–Seidel methods, respectively. The results show that with these settings, the correlations of the A1–A2 and A1–A3 decrease along with the increasing $\boldsymbol{L}$, meanwhile, the correlation coefficient of A2–A3 is consistent, implying that the iterative methods converge into the same solution. This phenomenon is caused by the use of the pseudo-inverse in evaluating the output weights. With the increasing $\boldsymbol{L}$, the rank of $\mathbb{H}$ drops sharply, leading to an uncorrelated result to the iterative methods. When all the pseudo-inverse calculations of the algorithms are replaced by a regularized form with $r = 0.1$, the correlations of $\boldsymbol{B}$ between all these algorithms are almost one, as shown in Fig. 2d. Note that these correlations are not strictly one for the smaller $\boldsymbol{L}$, this is because the H_m is of full column rank in this case.

Table 1
Datasets description.

Dataset	Samples	Features	Values	Training/testing
Buzz prediction on Twitter	583,250	77	Numeric	495,763/87,487
Year prediction MSD	515,345	90	Numeric	463,715/51,630

Table 2
Heterogeneous feature groups of Twitter dataset ($M = 11$).

Group	NCD	AI	AS(NA)	BL	NAC	AS(NAC)	CS	AT	NA	ADL	NAD
Columns	[1,7]	[8,14]	[15,21]	[22,28]	[29,35]	[36,42]	[43,49]	[50,56]	[56,63]	[64,70]	[71,77]

1. NCD: number of created discussions. This feature measures the number of discussions created at time step t and involving the instance's topic.
2. AI: author increase. This feature measures the number of new authors interacting on the instance's topic at time t (i.e. its popularity).
3. AS(NA): attention level (measured with the number of authors). This feature is a measure of the attention paid to the instance's topic on a social media.
4. BL: burstiness level. The burstiness level for a topic z at a time t is defined as the ratio of NCD and NAD.
5. NAC: number of atomic containers. This feature measures the total number of atomic containers generated through the whole social media on the instance's topic until time t.
6. AS(NAC): attention level (measured with the number of contributions). This feature is a measure of the attention paid to the instance's topic on a social media.
7. CS: contribution sparseness. This feature is a measure of spreading of contributions over discussion for the instance's topic at time t.
8. AI: author interaction. This feature measures the average number of authors interacting on the instance's topic within a discussion.
9. NA: number of authors. This feature measures the number of authors interacting on the instance's topic at time t.
10. ADL: average discussions length. This feature directly measures the average length of a discussion belonging to the instance's topic.
11. NAD: number of average discussions. This features measures the number of discussions involving the instance's topic until time t.

4. Performance evaluation

This section reports the experimental results of the SCNE on two large-scale datasets. We first introduce the datasets and explain the heterogeneous feature generation process. Then we present all the details of the experimental settings and results. The comparisons between the SCNE and the DNNE are carried out with remarks, and a robustness analysis of the SCNE is discussed at the end.

4.1. Large-scale datasets

The SCNE are used to explore two large-scale datasets, which are (i) Buzz Prediction on Twitter (Twitter)[2] and (ii) Year Prediction MSD (Year).[3] Table 1 shows the samples, features, values and training/testing split. These datasets have been analysed previously with some results.

The Twitter dataset is used for the annotation prediction of the mean number of active discussion (NAD). This target is a positive integer that describes the popularity of the sample's topic. Each sample covers weeks of observation for a specific topic and is described by 77 features. All the observations are independent and identically distributed.

Year dataset is used for the prediction of the release year of a song from audio features. Songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s. Among the 90 features, there are 12 belongs to timbre averages and 78 belongs to timbre covariance. The first value is the year (target), ranging from 1922 to 2011. All the features extracted from the timbre features from The-Echo-Nest-API.[4] The average and covariance are taken over all segments, each segment being described by a 12-dimensional timbre vector.

4.2. Heterogeneous feature generation

There are various methods to generate the heterogeneous feature sets for the base models. In our experiments, the 77 features of Twitter dataset are partitioned into 11 groups based on the introduction of the dataset. Each group has its specific meaning. The details, such as group names, feature indexes and explanations are listed in Table 2.

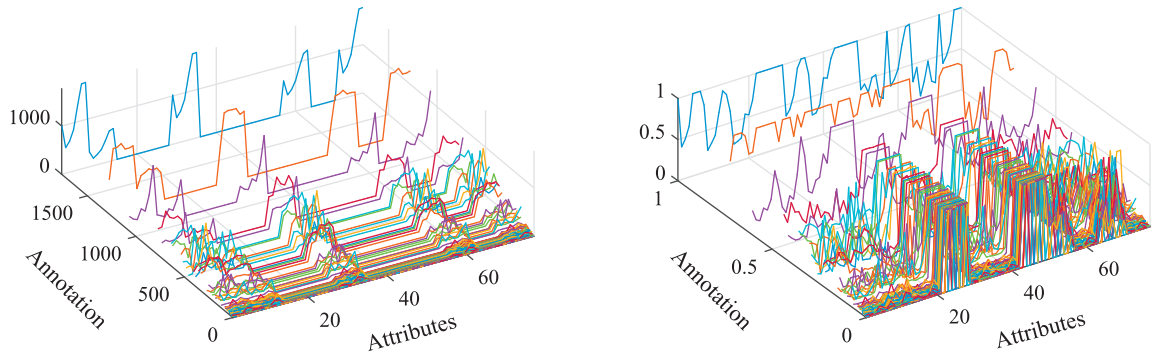
According to the data description of the Year dataset, the total 90 features could be partitioned into two groups, timbre averages group (12 features) and timbre covariance (78 features) group. We equally split the timbre covariance feature group into 6 small groups (TC-a to TC-f). Thus each small timbre covariance feature group contains 13 features, which is close to the timbre average feature group. The details could be found in Table 3.

[2] http://ama.liglab.fr/resourcestools/datasets/buzz-prediction-in-social-media/.
[3] http://archive.ics.uci.edu/ml/datasets/YearPredictionMSD.
[4] https://developer.spotify.com/spotify-echo-nest-api/.

Table 3
Heterogeneous feature groups of Year dataset ($M = 7$).

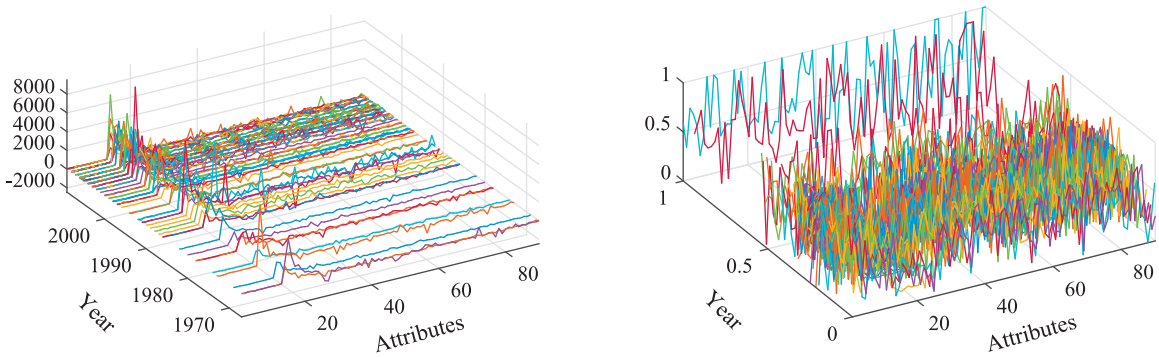
Group	TA	TC-a	TC-b	TC-c	TC-d	TC-e	TC-f
Columns	[1,12]	[13,25]	[26,38]	[39,51]	[52,64]	[65,77]	[78,90]

1. TA: timbre averages. These features are extracted from the 'timbre' features and described by a 12-dimensional timbre vector.
2. TC-[a,...,f]: timbre covariance. There features are calculated according to the original segments of the songs.



(a) Original. (b) 0-1 normalization.

Fig. 3. The Buzz Prediction on Twitter data: 80 samples from the original dataset. The x-axis represents the attributes, the y-axis shows the Annotation, and the z-axis denotes the values with respect to each attribute. Each line represents one Twitter sample.



(a) Original. (b) 0-1 normalization.

Fig. 4. The Year Prediction MSD data: 80 samples from the original dataset. The x-axis represents the attributes, the y-axis shows the Year, and the z-axis denotes the timbre values (average and covariance) with respect to each attribute. Each line represents one Year sample.

4.3. Experimental setup

As a common knowledge in machine learning, data preprocessing plays a crucial role before modelling. Normalization and standardization are two widely used methods for rescaling data. The 0–1 normalization could scale all numeric variables into the range [0, 1], one possible formula is $x_{new} = (x - x_{\min})/(x_{\max} - x_{\min})$. The z-score standardization transforms the data to have zero mean and unit variance by the formula $x_{new} = (x - \bar{x})/\sigma$, which indicates how many standard deviations an element is from the mean. However, if there were outliers in the dataset, normalization will certainly scale the "normal" data to a very small interval. Our experiments aim to demonstrate the capability of the SCNE for large-scale datasets, here we choose the 0–1 normalization method for data preprocessing and assume there is no outlier in the dataset. To show the data distribution, we randomly select a small batch of samples from each dataset and present them in Figs. 3 and 4, respectively.

All the experiments are designed, repeated and followed the same procedure. Fig. 5 presents the general experimental diagram. The arrows indicate the direction of the data feeds. The experiments are designed in two stages, training and testing, indicated by the dash-lined box in the diagram. In the training stage, the training dataset is used to build the SCNE, and the validation data is used to adjust and refine the hyper-parameters of the ensemble, such as $\{S\}$, M, λ, $L_{\max}$ and $k_{\max}$ (In our experiments, the heterogeneous feature set $\{S\}$ and the base model number M is predefined). When all these parameters are properly estimated, the training and validation data will be used together as one combined set to retrain

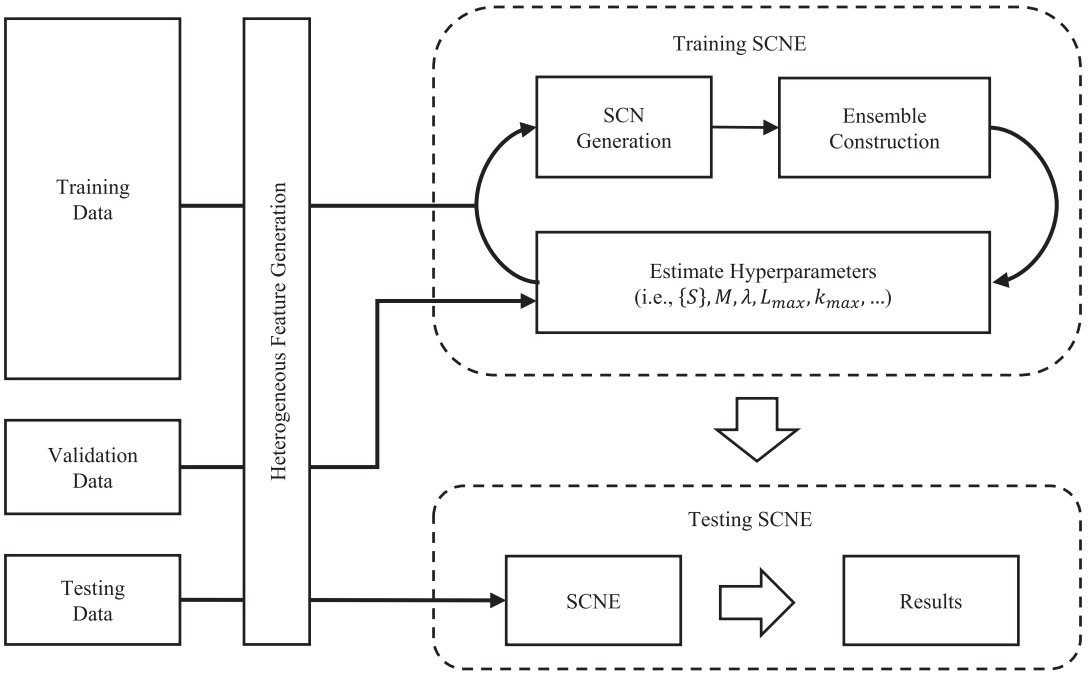


Fig. 5. Experimental diagram of the SCNE including the SCN generation and ensemble construction.

the SCNE again; then this final ensemble will be tested on the testing dataset. The testing results could be seen as a good indicator for the generalization ability of the SCNE. Unlike some benchmark dataset, there is no ready-made training or testing sets of Twitter data, which leaves us various choices to partition the data. For Twitter data, we apply 70% of the total samples for training, 15% for validation and the rest 15% for testing, denoted as "70-15-15". The Year dataset has been specified with the testing data (almost 10%), so we randomly split the rest 90% samples into two parts 70% for training and 20% for validation, denoted as "70-20-10".

4.4. Results and discussion

The following section presents all the training, validation, testing results and parameter estimations of SCNE on Twitter and Year datasets. All the experimental conditions and details are stated and specified as follows for the convenience to repeat the experiments. The comparative ensemble is the DNNE with the same structure of the SCNE.

For large datasets, such as the Twitter and Year in this paper, the general training-validation procedure for estimating hyper-parameters or preventing model overfitting usually costs a large amount of time. According to the SC-III algorithm, each base model needs to compute the results of all its training inputs, and then evaluate all the candidate nodes to add one or a few hidden nodes in one SC-search loop. This would take more time to estimate the proper hidden node number L_m of each SCN models of the ensemble. To accelerate this procedure, a down-sampling approach is applied, that is, instead of using all the samples, we randomly select two sub-datasets to estimate the parameters from the training and validation datasets, respectively, and repeat this procedure for several times to obtain statistic results. In this paper, we choose 70,000 samples from the training set and 30,000 from the validation set with heterogeneous features as the training-validation datasets for the estimation of the hidden node number of SCNs in the ensemble. Then build each SCN model with increasing hidden nodes to find the hidden node number with respect to the minimal validation error. The search range for the L of each SCN model is [1, 120].

Fig. 6 presents two examples of the training-validation error lines for the hidden nodes estimations of the base models on Twitter and Year datasets, respectively. This procedure is repeated for 10 times. The estimated hidden node numbers of the SCNs are summarised in Tables 4 and 5. The medium of each column is chosen as the hidden node number for each SCN base model.

For fair comparisons, the DNNE shares the same architecture as the SCNE, that is, each RVFL base model has the same number of hidden nodes as the corresponding SCN base model. According to [14], the range of the random weights and biases of the RVFL networks, $[-\alpha, +\alpha]$, needs to be estimated to improve the training performance. In our experiments, we compare the RVFL models with varying α from 0.5 to 1.4. The results are listed in Tables 6 and 7. The training data and validation data are sampled as the same as those for the SCN model. Each result is the average of 10 independent repeats with the same α. For each RVFL base model, the α^* with the minimal validation error is chosen for the RVFL base model used to construct the final DNNE.

Table 4
SCNE base model hidden node estimations on the Twitter dataset.

SCNE base model (Twitter)											
Index $\backslash m$		2	3	4	5	6	7	8	9	10	11
1	17	14	44	66	10	9	42	41	23	30	15
2	19	22	14	63	25	10	45	48	14	17	13
3	10	20	23	61	18	32	45	33	21	18	17
4	14	30	25	77	10	18	40	24	24	29	19
5	23	32	15	64	17	15	41	31	18	39	16
6	16	30	24	69	23	21	39	39	22	24	10
7	17	17	15	90	17	6	38	34	32	23	19
8	14	30	12	53	16	13	39	41	29	32	14
9	17	18	30	72	23	13	41	33	18	30	17
10	7	13	20	73	15	31	43	36	14	20	19
Medium	17	21	22	68	17	14	41	35	22	27	17

Table 5
SCNE base model hidden node estimations on the Year dataset.

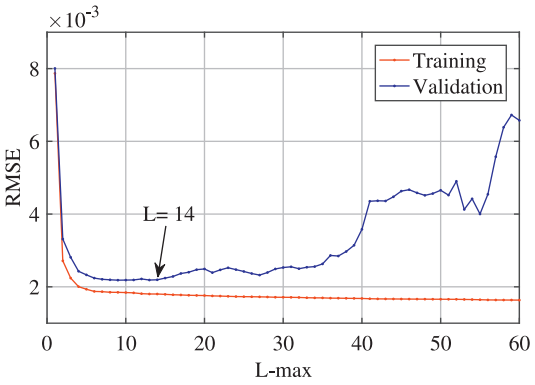
SCNE base model (Year)							
Index $\backslash m$	1	2	3	4	5	6	7
1	70	23	17	36	51	15	18
2	61	54	32	19	17	21	18
3	62	56	17	29	36	29	37
4	60	44	28	23	43	26	30
5	61	22	23	20	22	18	33
6	61	24	19	24	32	21	18
7	60	21	28	20	38	22	33
8	65	28	26	19	18	19	38
9	40	27	21	50	16	22	23
10	60	27	28	19	12	29	20
Medium	61	27	25	22	27	22	27

Table 6
RVFL base model random weight range estimations for the Twitter dataset.

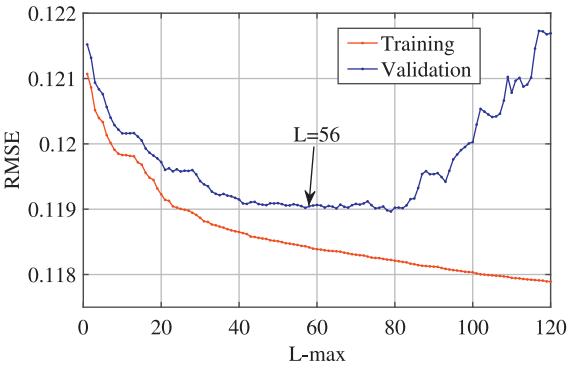
Validation RMSE on Twitter data(10^{-3})											
$\alpha \backslash m$	1	2	3	4	5	6	7	8	9	10	11
0.5	4.244	9.574	6.014	7.652	2.226	3.756	8.012	9.821	4.448	8.213	2.108
0.6	2.527	14.557	5.060	7.842	3.143	4.242	7.845	8.590	3.524	8.578	2.484
0.7	3.990	4.922	7.852	8.085	2.849	4.141	7.749	10.030	3.990	8.146	2.949
0.8	2.117	6.759	4.602	8.206	2.731	3.744	8.048	8.550	4.639	8.596	2.580
0.9	2.133	4.849	5.584	8.011	2.332	3.964	7.899	8.375	5.041	8.553	2.198
1.0	2.064	4.001	3.899	8.206	2.332	3.823	7.872	8.336	5.082	8.468	2.297
1.1	2.568	9.704	4.361	8.268	3.029	4.069	8.218	9.124	5.293	7.915	2.593
1.2	2.666	4.211	3.992	7.904	2.308	4.289	7.792	11.154	5.207	8.059	2.337
1.3	2.509	5.262	4.551	8.073	2.137	4.259	7.719	8.250	3.602	8.527	2.295
1.4	2.325	5.375	5.131	8.161	2.418	3.505	7.973	8.389	3.713	8.076	2.327
α^*	1.0	1.0	1.0	0.5	1.3	1.4	1.3	1.3	0.6	1.1	0.5

Table 7
RVFL base model random weight range estimations for the Year dataset.

Validation RMSE on Year data (10^{-1})							
$\alpha \backslash m$	1	2	3	4	5	6	7
0.5	1.101	1.202	1.206	1.202	1.195	1.204	1.208
0.6	1.103	1.209	1.207	1.201	1.197	1.204	1.209
0.7	1.103	1.204	1.207	1.200	1.196	1.207	1.209
0.8	1.105	1.202	1.206	1.202	1.198	1.202	1.210
0.9	1.105	1.203	1.206	1.200	1.197	1.207	1.211
1.0	1.103	1.204	1.213	1.201	1.200	1.207	1.211
1.1	1.105	1.204	1.213	1.200	1.194	1.202	1.215
1.2	1.102	1.204	1.207	1.203	1.199	1.202	1.213
1.3	1.100	1.207	1.206	1.202	1.193	1.208	1.211
1.4	1.101	1.204	1.207	1.197	1.198	1.206	1.212
α^*	1.3	0.8	0.8	1.4	1.3	0.8	0.5



(a) Twitter, m =1, index = 4, L_m = 14.



(b) Year, m =2, index = 3, L_m = 56.

Fig. 6. Examples of the training-validation error lines for estimating the hidden nodes of the SCN model in the ensemble on the Twitter and Year datasets. (a) SCN base model 1, index = 4, on the Twitter dataset; (b) SCN base model 2, index = 3, on the Year dataset. For better visual effect, the axis of (a) has been adjusted from [0, 120] to [0, 60].

Table 8
Twitter dataset ensemble results ($M = 11$).

Ensemble Type	Algorithm	Training RMSE(10^{-3})	Test RMSE(10^{-3})
SCNE	Naive	3.950± 0.0037	3.678± 0.0041
	Pseudo-inverse	3.811± 0.0032	3.542± 0.0031
	Block Jacobi	3.811± 0.0032	3.542± 0.0042
	Block Gauss–Seidel	3.811± 0.0031	3.542± 0.0041
DNNE	Naive	3.959± 0.0036	3.839± 0.0114
	Pseudo-inverse	3.829± 0.0050	3.696± 0.0104
	Block Jacobi	3.824± 0.0052	3.694± 0.0107
	Block Gauss–Seidel	3.824± 0.0052	3.694± 0.0101

Table 9
Year dataset ensemble results ($M = 7$).

Ensemble Type	Algorithm	Training RMSE(10^{-1})	Test RMSE(10^{-1})
SCNE	Naive	1.16351± 0.00101	1.14829± 0.00096
	Pseudo-inverse	1.16093± 0.00165	1.14720± 0.00121
	Block Jacobi	1.16042± 0.00083	1.14690± 0.00114
	Block Gauss–Seidel	1.16042± 0.00097	1.14690± 0.00109
DNNE	Naive	1.16451± 0.00097	1.16129± 0.00098
	Pseudo-inverse	1.16122± 0.00113	1.15320± 0.00115
	Block Jacobi	1.16122± 0.00114	1.15320± 0.00115
	Block Gauss–Seidel	1.16122± 0.00103	1.15320± 0.00113

The following results show the advantages of the SCNE on Twitter and Year datasets. When the hidden node numbers of the SCNE and random weights ranges of the DNNE are estimated according to the previous experiments, we retrain the SCNE and DNNE with the full training datasets with the regularizing factor $\lambda = 0.10$ and the iteration number $k_{\max} = 10$. Table 8 and 9 present the final training and testing results (i.e., the root mean square error, RMSE) of SCNE and DNNE with respect to 4 construction algorithms. All the results are the averages of 10 independent repeats. It is clear to see that the SCNE has lower training and testing RMSE than the DNNE on both datasets. The pseudo-inverse method and iterative schemes got much lower errors than the naive algorithm. Despite the training time cost, these results clearly indicate that the proposed SCNE works better than the DNNE.

Remark 3. In SCNE construction, it should be emphasized that the pseudo-inverse method costs much more computing time and memory than the iterative methods with a large $\boldsymbol{L}$. Ideally, applying the iterative methods, block Jacobi and block Gauss–Seidel methods, could lower the complexity to approximate $O(\sum_{m=1}^{M} L_m^3)$. In the Twitter experiments, the peak of the memory cost of the pseudo-inverse method is about 7.9GB, which is the limit of our computer (8GB RAM) and it tends to take more. However, for iterative methods, the peaks are at almost the same level as the naive method, never beyond 5.4GB, which means on the same machine, the iterative methods have more potentials than the pseudo-inverse method for large-scale data modelling.

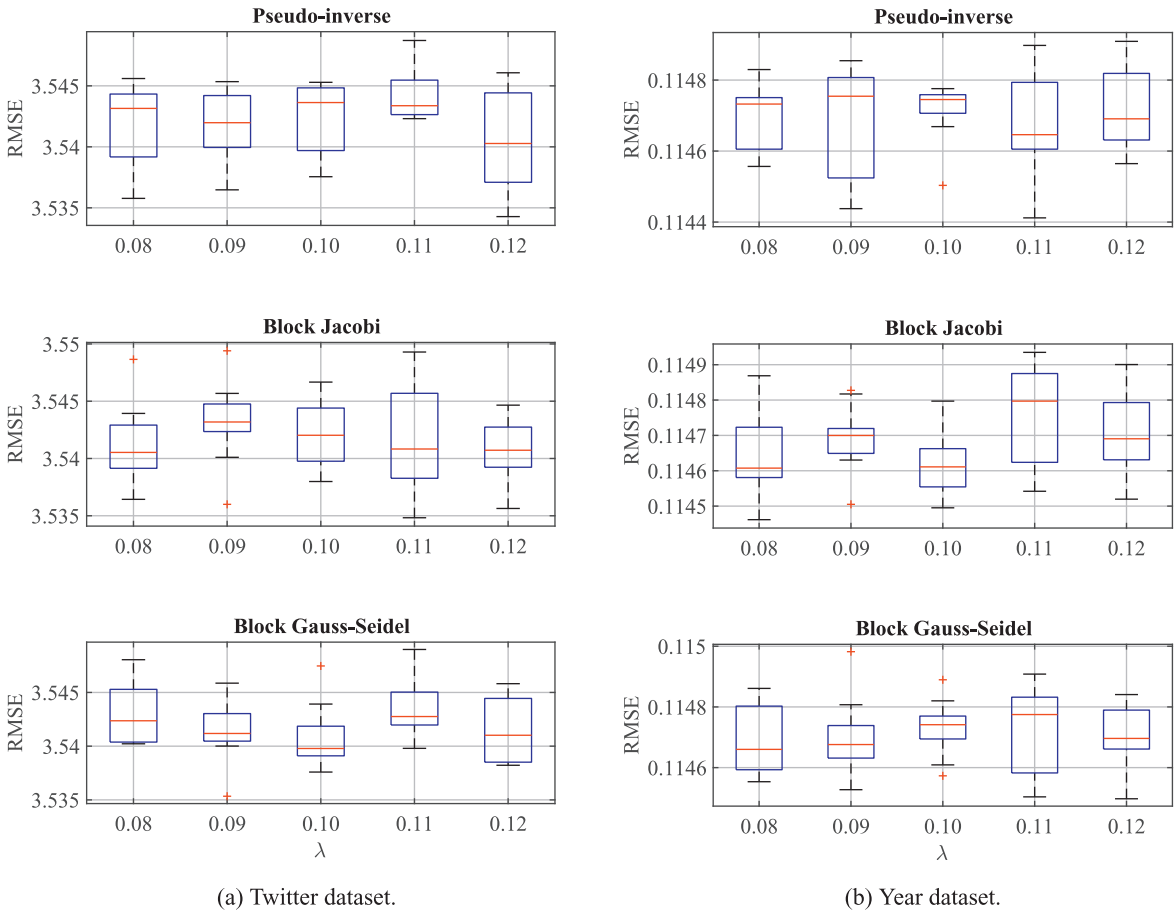


Fig. 7. Distributions of the SCNE testing results for the robustness analysis on Twitter and Year datasets with λ from 0.08 to 0.12. The figures should be scaled by 10^{-3} and 10^{-1} for the Twitter and Year datasets, respectively. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article).

4.5. Robustness analysis

The regularizing factor $\lambda = 0.10$ is used for the construction of SCNE in previous experiments. To investigate the reliability of the proposed SCNE system, an analysis on the robustness of the modelling performance with respect to some key learning parameters should be carried out. In this work, we focus on the robustness analysis for the regularizing factor λ used in the NCL. The hidden nodes and data partitions for SCN base models are the same as used in the previous experiments. Only the value of the λ is changed here. Fig. 7 depicts the distribution of 10 testing results of the SCNE in box-plots with respect to $\lambda = [0.08, 0.09, 0.10, 0.11, 0.12]$. On each box, the central red mark indicates the median, and the bottom and top edges of the box indicate the first and third percentiles ($Q1$ and $Q3$), respectively. The top and bottom bars indicate minimum and maximum within the inner fence $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$ where $IQR = Q3 - Q1$. The values beyond the inner fence are suspected outliers plotted individually using the red '+' symbol. For both datasets, the results show that the λ affects the SCNE testing performance slightly but not significantly within a certain range.

5. Conclusion

Analysing more data quickly with higher accuracy turns to be significant nowadays because of a vast number of real-world applications from various domains. Traditional machine learning techniques, such as neural networks with optimization-based learning algorithms, support vector machines and decision trees, are hardly applied for large-scale datasets. Ensemble learning with its theoretical framework helps in improving the generalization performance of the base learner models, but it still has some limitations on the efficiency and scalability for dealing with large-scale data modelling problems.

This paper contributes to the development of randomized neuro-ensemble with heterogeneous features, where the stochastic configuration networks are employed as base learners and the well-known negative correlation learning strategy

is adopted to evaluate the output weights of the SCNE model. To overcome the challenge in computing a pseudo-inverse of a huge sized linear equation system used in the least squares method, we suggest to utilize the block Jacobi and block Gauss–Seidel iterative schemes for problem solving. Some analyses and discussions on these solutions for evaluating the output weights are given by a demonstration. Simulation results clearly indicate that it is necessary to apply the ridge regression method for building the SCN base models, so that the resulting SCNE models with the iterative schemes can be consistent with the one built by using the non-iterative method in terms of the correlationship of the output weights.

The reported results in the demonstration implies that the statement on the speediness of the pseudo-inverse-based solution for building randomized learner models (either for single or ensemble models) is valid only for smaller datasets. Indeed, its computational complexity and time cost are very high, even infeasible, for large-scale datasets. Experimental results with comparisons over two large-scale benchmark datasets show that the proposed SCNE always outperforms DNNE. Robustness analysis on the modelling performance with respect to the regularizing factor used in NCL reveals that our proposed ensemble system performs robustly with good potential for large-scale data analytics. Further researches on improved feature grouping methodology, robust large-scale data regression [29] and enhancement of the generalization performance of the ensemble system are being expected.

References

[1] M. Alhamdoosh, D. Wang, Fast decorrelated neural network ensembles with random weights, Inf. Sci. 264 (2014) 104–117.
[2] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[3] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[4] G. Brown, J.L. Wyatt, P. Tiňo, Managing diversity in regression ensembles, J. Mach. Learn. Res. 6 (2005) 1621–1650.
[5] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Trans. Neural Netw. 20 (12) (2009) 1962–1979.
[6] C. Cui, D. Wang, High dimensional data regression using Lasso model and neural networks with random weights, Inf. Sci. 372 (2016) 505–517.
[7] S. Geman, B. Elie, D. René, Neural networks and the bias/variance dilemma, Neural Comput. 4 (1) (1992) 1–58.
[8] G.H. Golub, C.F.V. Loan, Matrix Computations, third, The Johns Hopkins University Press, Baltimore, 1996.
[9] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 993–1001.
[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.
[11] 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.
[12] B. Igelnik, Y.H. Pao, S.R. LeClair, C.Y. Shen, The ensemble approach to neural-network learning and generalization, IEEE Trans. Neural Netw. 10 (1) (1999) 19–30.
[13] M. Jordan, T. Mitchell, Machine learning: trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260.
[14] M. Li, D. Wang, Insights into randomized algorithms for neural networks: practical issues and common pitfalls, Inf. Sci. 382–383 (2017) 170–178.
[15] W.T. Li, D. Wang, T.Y. Chai, Multi-source data ensemble modeling for clinker free lime content in rotary kiln sintering processes, IEEE Trans. Syst. Man Cybern.: Syst. 45 (2) (2015) 303–314.
[16] Y. Liu, X. Yao, Ensemble learning via negative correlation, IEEE Trans. Neural Netw. 12 (10) (1999) 1399–1404.
[17] D. Lopez-Paz, P. Hennig, B. Schölkopf, The randomized dependence coefficient, in: Advances in Neural Information Processing Systems, 2013, pp. 1–9.
[18] H.V. Nguyen, E. Müller, J. Vreeken, P. Efros, K. Böhm., Multivariate maximal correlation analysis, in: Proceeding of International Conference on Machine Learning, 2014, pp. 775–783.
[19] Y.H. Pao, Y. Takefji, Functional-link net computing, IEEE Comput. J. 25 (5) (1992) 76–79.
[20] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238.
[21] C. Zhang, Y. Ma, Ensemble Machine Learning, Springer, US, 2012.
[22] D.N. Reshef, Y.A. Reshef, H.K. Finucane, S.R. Grossman, G. McVean, P.J. Turnbaugh, E.S. Lander, M. Mitzenmacher, P.C. Sabeti, Detecting novel associations in large data sets, Science 334 (6062) (2011) 1518–1524.
[23] B.E. Rosen, Ensemble learning using decorrelated neural networks, Connect. Sci. 8 (3–4) (1996) 373–384.
[24] S. Scardapane, D. Wang, P. Massimo, U. Aurelio, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015) 271–284.
[25] S. Scardapane, D. Wang, P. Massimo, A decentralized training algorithm for echo state networks in distributed big data applications, Neural Netw. 78 (2016) 65–74.
[26] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227.
[27] N. Ueda, R. Nakano, Generalization error of ensemble estimators, in: Proceedings of IEEE Conference on Neural Networks, 1992, pp. 90–95. 1
[28] D. Wang, M. Li, Stochastic configuration networks: Fundamentals and algorithms, 2017a, ArXiv:1702.03180[cs.NE].
[29] D. Wang, M. Li, Robust stochastic configuration networks with kernel density estimation for uncertain data regression, Inf. Sci. 412–413 (2017b) 210–222.
[30] D.M. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.