Engineering 18 (2022) 186–196

Contents lists available at ScienceDirect

Engineering

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

Research
Smart Process Manufacturing—Article

# A Local Quadratic Embedding Learning Algorithm and Applications for Soft Sensing

Yaoyao Bao, Yuanming Zhu *, Feng Qian *

*Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China*

ARTICLE INFO

*Article history:*
Received 30 December 2019
Revised 25 January 2021
Accepted 29 April 2022
Available online 28 September 2022

*Keywords:*
Local quadratic embedding
Metric learning
Regression machine
Soft sensor

ABSTRACT

Inspired by the tremendous achievements of meta-learning in various fields, this paper proposes the local quadratic embedding learning (LQEL) algorithm for regression problems based on metric learning and neural networks (NNs). First, Mahalanobis metric learning is improved by optimizing the global consistency of the metrics between instances in the input and output space. Then, we further prove that the improved metric learning problem is equivalent to a convex programming problem by relaxing the constraints. Based on the hypothesis of local quadratic interpolation, the algorithm introduces two lightweight NNs; one is used to learn the coefficient matrix in the local quadratic model, and the other is implemented for weight assignment for the prediction results obtained from different local neighbors. Finally, the two sub-models are embedded in a unified regression framework, and the parameters are learned by means of a stochastic gradient descent (SGD) algorithm. The proposed algorithm can make full use of the information implied in target labels to find more reliable reference instances. Moreover, it prevents the model degradation caused by sensor drift and unmeasurable variables by modeling variable differences with the LQEL algorithm. Simulation results on multiple benchmark datasets and two practical industrial applications show that the proposed method outperforms several popular regression methods.



## 1. Introduction

In the cement production process, it is essential to monitor the quality of products, such as the fineness of raw meal, the free calcium oxide content of clinkers, and so forth. However, online instrumentations for these indicators are costly and require frequent regular maintenance. In industrial practice, off-line analysis in the lab is often implemented for these indexes every 2 h or more, which results in untimely feedback for real-time control systems. These problems can be solved by soft-sensing techniques [1,2].

Soft sensing is essentially a regression machine that evaluates quality indexes in real time using other instrumental variables that are available online. That is, given the $D$-dimensional input variables $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(s)}\}$ (where each element represents an instance) and their corresponding output variables $Y = \{y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(s)}\}$, the objective of the regression machine is to construct an optimal mapping function using the knowledge implied in the training data, which achieves remarkable prediction accuracy on the test set. Successful soft-sensing applications can be found in diversified industries such as petroleum refining [3], metallurgical processes [4], and energy management [5,6].

Soft-sensing models originate from multivariate statistical regression models, including linear regression (LR), principal component regression (PCR), partial least squares (PLS), and some variants with regularization strategies to balance the empirical error and complexity of the model, such as least absolute shrinkage and selection operator (LASSO) and ridge [7]. Kernel strategies have been extensively studied and combined with the aforementioned algorithms to solve the regression problem for nonlinear problems [8,9]. After that, machine learning methods such as $k$-nearest neighbor regression ($k$-NNR) [10], classification and regression trees (CARTs) [11,12], and support vector regression (SVR) [13,14] have been proposed for knowledge mining in massive data. To improve the performance of a single tree model, bagging strategies are implemented in random forest (RF) algorithms [15,16]. Similarly, the prediction accuracy of boosting algorithms can be increased by combining a series of iteratively learned weak machines [17,18], such as gradient boosting machines (GBMs) and

* Corresponding authors.
*E-mail addresses:* yuanmingzhu@ecust.edu.cn (Y. Zhu), fqian@ecust.edu.cn (F. Qian).

https://doi.org/10.1016/j.eng.2022.04.025

extreme gradient boosting (XGBoost). Furthermore, breakthroughs in deep learning in image and speech recognition have caused neural networks (NNs) [19,20] to become one of the most popular methods in the field of machine learning, especially when the data samples are sufficient. This popularity can be attributed to NNs' powerful feature extraction capabilities with specially designed structures [21].

Among these algorithms, *k*-NNR is the simplest and one of the most prevailing regression methods. It is widely used in machine learning problems because it does not require an explicit model structure or any prior knowledge for data distribution. However, the strategy to use the average output of its *k*-nearest neighbors (*k*-NNs) as the prediction result also leads to this method's greatest disadvantages. Initially, the *k*-NNR algorithm employed the Euclidean distance metric for the measurement of sample similarities. However, the magnitudes of the input features can vary greatly; redundancies and correlations between variables can also be misleading, resulting in an unpractical distance metric. To cope with this problem, a generalization of the Mahalanobis distance [22] was proposed, which is equivalent to a weighted Euclidean distance between two linear projected images. However, in practical applications, the input features tend to have distinct contributions to the output variables. The key is to develop a reliable feature extraction model and apply the classical metrics, such as the Euclid distance and cosine similarity, to the mapped features. Locally linear embedding (LLE) reconstructs the samples in a low-dimensional space using the locally linear weighting method and achieves dimension reduction by minimizing the reconstruction error [23]. Nevertheless, the adjacency relation constructed by the classical Euclidean metric in a high-dimensional space cannot meet the needs of all classification tasks. Thus, researchers usually try to transform the input features into a scaled space [24,25] and to get the weight coefficients to predict the label by means of local reconstruction in the space. However, this method is very dependent on elegant design of the transformation model. For example, in a fuzzy transformation, the basic function and the division of fuzzy intervals may have a great influence on the prediction result, because the meaningful information contained in the output labels is not made full use of. To address this issue, Weinberger and Saul [26] introduced the concept of Mahalanobis distance metric learning, which allows the inverse covariance matrix in the Mahalanobis distance to denote any positive semidefinite matrix. Similar to the idea of linear discriminant analysis (LDA) [27], the Mahalanobis distance metric is learned by maximizing the ratio of the average internal class distance to the average between-class distance. Xing et al. [28] constructed a convex optimization problem for metric learning by taking the average between-class distance as the optimization target and the average within-class distance as the constraint. This method has been applied to semi-supervised data clustering problems.

The above methods are mainly designed for classification problems. For regression problems, Nguyen et al. [27] established a convex optimization problem by maximizing the consistency of the input and output distances over a set of constraint triplets in the neighborhood of each instance. However, the researchers did not elaborate the solution for a transformation matrix $\boldsymbol{A}$ in metric learning; the weight matrix $\boldsymbol{W}$ is optimized only under the condition of a given transformation matrix $\boldsymbol{A}$. Moreover, the tradeoff parameter $C$ tends to have a significant impact on the performance of the algorithm. Linear metric learning (LML) has limited power in feature representation, especially for high-dimensional samples such as image and text data. Deep metric learning (DML) uses deep neural network (DNN) models instead of linear transformations to extract features in order to achieve metric learning [29–31]. One of the greatest differences between LML and DML lies in the form of the loss function. For example, Song et al. [30] minimized the distances between samples from the same class and maximized the distances with a margin from different classes. In general, these methods involve the construction of triplet sets, which consist of an anchor, a positive point, and a negative point. This implies that the methods cannot be directly applied to regression problems.

In addition, using the average of *k*-NNs as the output prediction often results in conservative result. Take the wine quality assessment dataset on University of California Irvine (UCI) machine learning repository as an example. The *k*-NNR algorithm does not distinguish well between particularly high-grade or inferior wines. So, how does an operator predict the label? First, the operator will identify the most similar cases to the current sample in the historical data as references and then modify the label according to the change of the input features. We summarize this process and propose the local quadratic embedding learning (LQEL) algorithm. However, the coefficient matrix of the quadratic embedding function is difficult to obtain. Fortunately, the matrix is dependent on the location of the expansion point—that is, the current sample mentioned above. Thus, the coefficient matrix can be estimated by NNs, taking the current sample as the input. However, an appropriate network scale must be determined; otherwise, the model becomes over-fitted. To this end, ensemble methods to integrate multiple NNs are utilized to improve the generalization ability of NNs model [20,32]. The literature shows that standardizing the output of the hidden layer in the network by batch normalization (BN) can prevent distribution changes during the training process [33], which accelerates the convergence of networks. It has been pointed out that the dropout strategy can improve the generalization ability of the NN [34]. Moreover, superimposing a certain intensity of Gaussian noise on sample data can increase the number of training samples and thus improve the robustness of the model [35]. In general, these approaches improve the generalization of NNs in two ways. First, they increase the number of training samples; second, they add constraints to the network structure, reduce the complexity, and thus improve the network's predictive ability. This paper follows the latter route.

In this paper, metric learning is first accomplished to determine the neighborhood of a certain instance by maximizing the consistency of the distances between the input and output spaces. This makes full use of the information contained in the target labels and achieves the first step of the operators' strategy. Then, a local quadratic coefficient matrix is generated by a well-trained NN to make predictions based on neighboring references; this prevents the model degradation caused by sensor drift and unmeasured variables by means of the differential compensation method. Furthermore, the other NN assigns weights to the predictions provided by different neighbors according to their confidence, which achieves a balance between the prediction errors and measurement noises, thereby minimizing the prediction errors. The parameters of these two networks can be optimized by end-to-end training with stochastic gradient descent (SGD) algorithms. Empirical studies on several regression datasets, including two practical industrial datasets from the cement production process and hydrocracking process, show that, in most cases, the proposed method outperforms the popular regression methods.

The rest of this paper is organized as follows. In Section 2, a metric learning model is introduced and the optimization problem is proved to be equivalent to a convex optimization problem. In Section 3, the framework of the proposed LQEL is presented. In Section 4, several empirical studies, including a validation using actual industrial cases, are reported. The conclusions and contributions of this paper are summarized in Section 5.

## 2. Metric learning

A metric distance is a function $d: X \times X \to R_0^+$ that satisfies the following, for any $\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}, \boldsymbol{x}^{(k)} \in X$:

(1) **Non-negativity:** $d(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) \geq 0$, the equality holds if and only if $\boldsymbol{x}^{(i)} \equiv \boldsymbol{x}^{(j)}$

(2) **Symmetry:** $d(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) = d(\boldsymbol{x}^{(j)},\boldsymbol{x}^{(i)})$

(3) **Triangle inequality:** $d(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}) + d(\boldsymbol{x}^{(j)},\boldsymbol{x}^{(k)}) \geq d(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(k)})$

Given a set of $D$-dimensional input variables $X = \{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\boldsymbol{x}^{(3)},\ldots,\boldsymbol{x}^{(s)}\}$ and corresponding output labels $Y = \{y^{(1)},y^{(2)},y^{(3)},\ldots,y^{(s)}\}$, metric learning has to find an implied metric function with these training data. In this metric space, instances with similar output labels are gathered together, and dissimilar samples are pushed far away. Studies in this field focus a great deal of attention on Mahalanobis metric learning (MML) [26] due to its simplicity and clarity. In addition, this problem can usually be transformed into simple convex optimization, making it extremely convenient to find the global optimum. The model structure of MML is defined as follows:

$$d^2(\boldsymbol{u}-\boldsymbol{v}) = (\boldsymbol{u}-\boldsymbol{v})^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{u}-\boldsymbol{v}) \tag{1}$$

where $\boldsymbol{M}$ is a positive definite metric matrix to be learned, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are two different instances. The objective of MML is to obtain the optimal matrix $\boldsymbol{M}$ that meets the purpose of metric learning.

We hope to use the information implied in the output labels to guide the direction of metric learning. The basic principle is that similar input samples lead to similar target labels. The consistency of the distances between the input and output spaces, from a statistical point of view, can be described with the Pearson correlation coefficient. Therefore, the optimization problem is formed as follows:

$$\underset{M}{\operatorname{argmin}}\, J(\boldsymbol{M}) = \frac{-\sum_{i>j} d_{ij}^{(y)} \cdot \left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}}{\sqrt{\sum_{i>j}\left[\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2 - \frac{2}{N(N-1)}\left[\sum_{i>j}\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2}} \quad \text{s.t. } \boldsymbol{M} \geq 0 \tag{2}$$

where $d_{ij}^{(y)}$ is the square distance between the $i$th and $j$th instances in the target space, $\boldsymbol{\delta}_{ij}^{(x)}$ is the difference in input space denoted as $\boldsymbol{\delta}_{ij}^{(x)} \equiv \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}$, and $N$ is the sample number. Since the numerator and denominator of the objective function are homogeneous to $\boldsymbol{M}$, Eq. (2) can be equivalently converted to the optimization problem shown in Eq. (3):

$$\underset{M}{\operatorname{argmin}}\, J'(\boldsymbol{M}) = -\sum_{i>j} d_{ij}^{(y)} \cdot \left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}$$

$$\text{s.t. } \boldsymbol{M} \geq 0 \tag{3}$$

$$\sum_{i>j}\left[\left(\boldsymbol{\delta}_{ij}^{(\mathbf{x})}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2 - \frac{2}{N(N-1)}\left[\sum_{i>j}\left(\boldsymbol{\delta}_{ij}^{(\mathbf{x})}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2 = 1$$

Here, we prove that the above problem has a unique global optimal solution and that the solution can be obtained by relaxing the constraints. The reconstructed problem after constraint relaxation is shown in Eq. (4):

$$\underset{M}{\operatorname{argmin}}\, J'(\boldsymbol{M}) = -\sum_{i>j} d_{ij}^{(y)} \cdot \left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}$$

$$\text{s.t. } \boldsymbol{M} \geq 0 \tag{4}$$

$$\sum_{i>j}\left[\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2 - \frac{2}{N(N-1)}\left[\sum_{i>j}\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2 \leq 1$$

Denote $g(\boldsymbol{M}) = \sum_{i>j}\left[\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2 - \frac{2}{N(N-1)}\left[\sum_{i>j}\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]^2$; then, the first-order and second-order partial derivatives of $g(\boldsymbol{M})$ to $\boldsymbol{M}$ are as follows:

$$\frac{\partial g}{\partial \mathrm{vec}(\boldsymbol{M})} = 2\sum_{i>j}\left(\boldsymbol{\delta}_{ij}^{(x)}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)} \cdot \left[\boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right] - \frac{4}{N(N-1)}\left[\sum_{i>j}\left(\boldsymbol{\delta}_{ij}^{(\mathbf{x})}\right)^{\mathrm{T}} \boldsymbol{M} \boldsymbol{\delta}_{ij}^{(x)}\right]\left[\sum_{i>j} \boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right] \tag{5}$$

$$\frac{\partial^2 g}{\partial \mathrm{vec}(\boldsymbol{M})^2} = 2\sum_{i>j}\left[\boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right] \cdot \left[\boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right]^{\mathrm{T}} - \frac{4}{N(N-1)}\left[\sum_{i>j} \boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right]\left[\sum_{i>j} \boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right]^{\mathrm{T}} \tag{6}$$

where $\otimes$ is the Kronecker product, and $\mathrm{vec}(\boldsymbol{M})$ is the column expansion of $\boldsymbol{M}$. For $\forall \boldsymbol{u} \in R^{D^2}$, the inequality in Eq. (7) shows that the function $g(\boldsymbol{M})$ is a convex function.

$$\boldsymbol{u}^{\mathrm{T}} \frac{\partial^2 g}{\mathrm{vec}(\boldsymbol{M})^2} \boldsymbol{u} = 2\sum_{i>j} m_{ij}^2 - \frac{4}{N(N-1)}\left(\sum_{i>j} m_{ij}\right)^2 \geq 0 \tag{7}$$

where $m_{ij} \equiv \boldsymbol{u}^{\mathrm{T}} \cdot \left[\boldsymbol{\delta}_{ij}^{(x)} \otimes \boldsymbol{\delta}_{ij}^{(x)}\right]$. This implies that the constraints in Eq. (4) cause the feasible domain to be a convex set. Meanwhile, the second-order partial derivatives of the objective function $J(\boldsymbol{M})$ to $\boldsymbol{M}$ are calculated as follows:

$$\frac{\partial^2 J}{\mathrm{vec}(\boldsymbol{M})^2} = 0 \tag{8}$$

In summary, the problem in Eq. (4) is demonstrated to be a convex optimization problem—that is, it has a unique global optimal solution [36]. Denote the optimal solution as $\boldsymbol{M}^*$; it then follows that $g(\boldsymbol{M}^*) = 1$. Otherwise, $0 < g(\boldsymbol{M}^*) < 1$. Denote $\boldsymbol{M}' = \boldsymbol{M}^*/g(\boldsymbol{M}^*)$ and substitute this into Eq. (4). It is not difficult to verify that $\boldsymbol{M}'$ is within the feasible domain. In addition, the objective follows $J(\boldsymbol{M}') = \frac{J(\boldsymbol{M}^*)}{g(\boldsymbol{M}^*)} < J(\boldsymbol{M}^*)$, which is contradictory. The conclusion indicates that the problem in Eq. (3) has a unique global optimal solution, which can be obtained by solving the convex optimization problem in Eq. (4).

## 3. Local quadratic embedding learning

Most $k$-NNR algorithms take the average weight of neighboring outputs as the prediction result. This will lead to moderate predictions, as these neighboring outputs will not exceed the maximum and minimum intervals of the neighboring samples. Given an instance $\boldsymbol{x}$, its $k$-NNs $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(K)}$, and their corresponding outputs $y^{(1)}, y^{(2)}, \ldots, y^{(K)}$, an intuitive idea is to take the linear weighted average of neighboring labels as the prediction. However, the result determined by $\hat{y} = \sum_{i=1}^{K} w_i y^{(i)}\left(w_i \geq 0, \sum_{i=1}^{K} w_i = 1\right)$ always follows $\min_i\{y^{(i)}\} \leq \hat{y} \leq \max_i\{y^{(i)}\}$, which leads to conservative prediction results. To address this issue, we intend to establish a local linear mapping model between the differences in the two spaces, along with an independent model to distinguish the reliability of different neighboring predictions—that is, to assign different weights to the predictions based on different neighbors.

The scheme of the LQEL algorithm is shown in Fig. 1. To obtain the output label corresponding to sample $\boldsymbol{x}$, the $k$-NNs are first determined using the conclusion of the metric learning in Section 2 (the ellipse in the left of the figure). Suppose a function $\mathscr{F}: \boldsymbol{\delta x} \rightarrow \delta y$ is learned to describe the mapping from the difference of input to the difference of output in two spaces. Then, for each

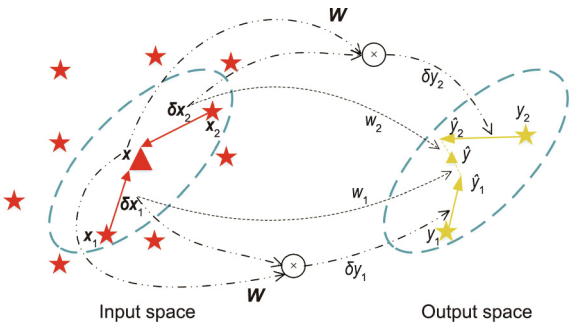


**Fig. 1.** The scheme of LQEL.

$\boldsymbol{x}_j$ $(j = 1, 2, 3, ..., K)$ adjacent to the instance $\boldsymbol{x}$, the $j$th estimation can be performed as follows: $\hat{y}_j = y_j + \mathscr{F}(\boldsymbol{x} - \boldsymbol{x}_j)$ $(j = 1, 2, 3, ..., K)$. Finally, the above prediction results are linearly combined with an appropriate set of weights to obtain the final output:

$$\hat{y} = \sum_{j=1}^{K} w_j \left[ y_j + \mathscr{F}(\boldsymbol{x} - \boldsymbol{x}_j) \right] \ (j = 1, 2, 3, ..., K) \tag{9}$$

Denote the real mapping function from input to output as $g_0 \in C^2 : X \to R$, and define $\tilde{g}_0(\boldsymbol{x}|\boldsymbol{x}_0) = 0.5\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{\boldsymbol{x}_0}\boldsymbol{x} + \boldsymbol{B}_{\boldsymbol{x}_0}\boldsymbol{x} + C_{\boldsymbol{x}_0}$ as the second-order Taylor expansion expanded at the point $\boldsymbol{x}_0$ in the $\delta$ neighborhood, where

$$\boldsymbol{A}_{\boldsymbol{x}_0} = \nabla^2 g_0(\boldsymbol{x}_0)$$

$$\boldsymbol{B}_{\boldsymbol{x}_0} = \nabla g_0(\boldsymbol{x}_0)^{\mathrm{T}} - (\boldsymbol{x}_0)^{\mathrm{T}} \nabla^2 g_0(\boldsymbol{x}_0)$$

$$C_{\boldsymbol{x}_0} = g_0(\boldsymbol{x}_0) - \nabla g_0(\boldsymbol{x}_0)^{\mathrm{T}} \boldsymbol{x}_0 + 0.5(\boldsymbol{x}_0)^{\mathrm{T}} \nabla^2 g_0(\boldsymbol{x}_0) \boldsymbol{x}_0$$

Then, for $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in U_\delta(\boldsymbol{x}_0)$, the difference in the output space can be calculated as follows:

$$\begin{aligned} y_1 - y_2 &\approx \tilde{g}_0(\boldsymbol{x}_1|\boldsymbol{x}_0) - \tilde{g}_0(\boldsymbol{x}_2|\boldsymbol{x}_0) \\ &= \left(0.5(\boldsymbol{x}_1)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} \boldsymbol{x}_1 + \boldsymbol{B}_{\boldsymbol{x}_0} \boldsymbol{x}_1 + C_{\boldsymbol{x}_0}\right) \\ &\quad - \left(0.5(\boldsymbol{x}_2)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} \boldsymbol{x}_2 + \boldsymbol{B}_{\boldsymbol{x}_0} \boldsymbol{x}_2 + C_{\boldsymbol{x}_0}\right) \\ &= 0.5(\boldsymbol{x}_1 + \boldsymbol{x}_2)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} (\boldsymbol{x}_1 - \boldsymbol{x}_2) + \boldsymbol{B}_{\boldsymbol{x}_0} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \\ &\approx \left((\boldsymbol{x}_0)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} + \boldsymbol{B}_{\boldsymbol{x}_0}\right)(\boldsymbol{x}_1 - \boldsymbol{x}_2) \\ &\equiv \boldsymbol{W}(\boldsymbol{x}_1 - \boldsymbol{x}_2) \end{aligned} \tag{10}$$

where $U_\delta(\boldsymbol{x}_0)$ represents the $\delta$ neighborhood of $\boldsymbol{x}_0$ in the metric space defined in Section 2, $\boldsymbol{W} \equiv (\boldsymbol{x}_0)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} + \boldsymbol{B}_{\boldsymbol{x}_0}$ is the weight coefficient matrix of the linear mapping function.

The result of Eq. (10) implies that a linear model could be designed for prediction in $\boldsymbol{x}_0$'s $\delta$ neighborhood. The matrix $\boldsymbol{W}$ expanded on different reference points can be estimated by an independent NN—for example, using an NN $\mathscr{N}: X \to X$ to approximate the matrix as $\mathscr{N}(\boldsymbol{x}_0) = \boldsymbol{W}$. Considering that the parameter matrices $\nabla^2 g(\boldsymbol{x}_0)$ and $\nabla g(\boldsymbol{x}_0)$ tend to be more stable than $g(\boldsymbol{x}_0)$ in most practical circumstances, the NN required here should be much simpler than the one used to estimate the output label directly. In particular, when $g_0$ is a quadratic function, the matrices $\boldsymbol{A}_{\boldsymbol{x}_0}$ and $\boldsymbol{B}_{\boldsymbol{x}_0}$ do not change with the reference point. In this case, a simple linear NN could work well. In general, these procedures can effectively reduce the complexity of the model and improve the generalization.

This strategy provides $k$ estimation results for each instance, one from each nearest neighbor, but the reliabilities can vary considerably. From an intuitive perspective, the predictions given by distant neighbors tend to have high uncertainty. This implies that different weights should be assigned to each of the predictions. Prediction uncertainties caused by the presence of measuring noise can be restrained by the averaging method. Inspired by this idea, we intend to design a machine that generates different weights according to the relative location of the instance, which minimizes the expectation of mean square error (MSE).

Denote the measuring noise superimposed on the output label $y^{(i)}$ as $v^{(i)}$, which is subject to a normal distribution $v^{(i)} \sim N(0, \sigma^2)$. The error of the $i$th prediction is calculated as follows:

$$\begin{aligned} \hat{y}_i - y_0 &= y^{(i)} + \left((\boldsymbol{x}_0)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} + \boldsymbol{B}_{\boldsymbol{x}_0}\right)(\boldsymbol{x}_0 - \boldsymbol{x}^{(i)}) - g(\boldsymbol{x}_0) \\ &\approx 0.5(\boldsymbol{x}^{(i)})^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} \boldsymbol{x}^{(i)} + \boldsymbol{B}_{\boldsymbol{x}_0} \boldsymbol{x}^{(i)} + C_{\boldsymbol{x}_0} + v^{(i)} \\ &\quad + \left((\boldsymbol{x}_0)^{\mathrm{T}} \boldsymbol{A}_{\boldsymbol{x}_0} + \boldsymbol{B}_{\boldsymbol{x}_0}\right)(\boldsymbol{x}_0 - \boldsymbol{x}^{(i)}) - g(\boldsymbol{x}_0) \\ &= 0.5(\boldsymbol{x}^{(i)} - \boldsymbol{x}_0)^{\mathrm{T}} A_{\boldsymbol{x}_0} (\boldsymbol{x}^{(i)} - \boldsymbol{x}_0) + v^{(i)} \\ &\equiv e^{(i)} + v^{(i)} \end{aligned} \tag{11}$$

where $\hat{y}_i$ is the estimation given by the $i$th neighbor, $e^{(i)}$ is the estimation error, and $v^{(i)}$ represents the uncertainty. The target is to obtain a set of weights that minimize the objective $H(\boldsymbol{w})$:

$$\begin{aligned} \min H(\boldsymbol{w}) &= E\left[\left(\sum_{i=1}^{k} w_i \hat{y}_i - y_0\right)^2\right] \\ &= E\left\{\left[\sum_{i=1}^{k} w_i (e^{(i)} + v^{(i)})\right]^2\right\} \\ &= \left(\sum_{i=1}^{k} w_i e^{(i)}\right)^2 + \sigma^2 \sum_{i=1}^{k} {w_i}^2 \end{aligned} \tag{12}$$

$$\text{s.t.} \ \sum_{i=1}^{k} w_i = 1$$

$$w_i \geq 0 \ (i = 1, 2, 3, ..., K)$$

This problem can be solved with the Lagrange multiplier method and the Karush–Kuhn–Tucker conditions. In this problem, all the variables involved in Eq. (12) are $e^{(i)}$ and $\sigma^2$, and the optimal weight groups for $\boldsymbol{x}_0$'s neighbors are determined by $\boldsymbol{\delta}_{\boldsymbol{x}_0}^{(i)} = \boldsymbol{x}^{(i)} - \boldsymbol{x}_0$. Instead of solving the optimization problem, an NN is introduced here to generate different weights, whose inputs are the differences between the instance and its neighbors, that is, $\boldsymbol{\delta}_{\boldsymbol{x}_0}^{(i)} (i = 1, 2, 3, \cdots, K)$.

In summary, the specific framework of the proposed method is shown in Fig. 2. For a certain instance $\boldsymbol{x}^{(i)}$, the $k$-NNs are first determined with the results acquired by metric learning in Section 2. Denote these samples as $\boldsymbol{x}_j^{(i)} (j = 1, 2, 3, ..., k)$ and the corresponding target labels as $y_j^{(i)} (j = 1, 2, 3, ..., k)$.

Second, the coefficient matrix $\boldsymbol{W} \in R^D$ is calculated for the sample $\boldsymbol{x}^{(i)}$ with NN I, which is used for the estimation of $\hat{\delta}_{y^{(i)}j} = \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\delta}_{\boldsymbol{x}^{(i)}j} = \boldsymbol{W}^{\mathrm{T}} \left(\boldsymbol{x}^{(i)} - \boldsymbol{x}_j^{(i)}\right)$, and the $j$th prediction for $\boldsymbol{x}^{(i)}$ is calculated as $\hat{y}_j^{(i)} = y_j^{(i)} + \hat{\delta}_{y^{(i)}j}$. Finally, the weight $w_j^{(i)}$ for each estimation is provided by NN II, taking $\boldsymbol{\delta}_{\boldsymbol{x}^{(i)}j}$ as input. The final prediction $\hat{y}_i$ can be achieved with a linear combination of $\hat{y}_j^{(i)} (j = 1, 2, 3, \cdots, K)$.

In this paper, we introduce state-of-the-art strategies for NNs, such as BN and dropout. The MSE is employed as the loss function. The parameters of the proposed model, including the weights and biases in the two NNs, are optimized by the SGD algorithm.

## 4. Empirical learning

In order to know how well the proposed algorithm works, we use real-world benchmark regression datasets along with two

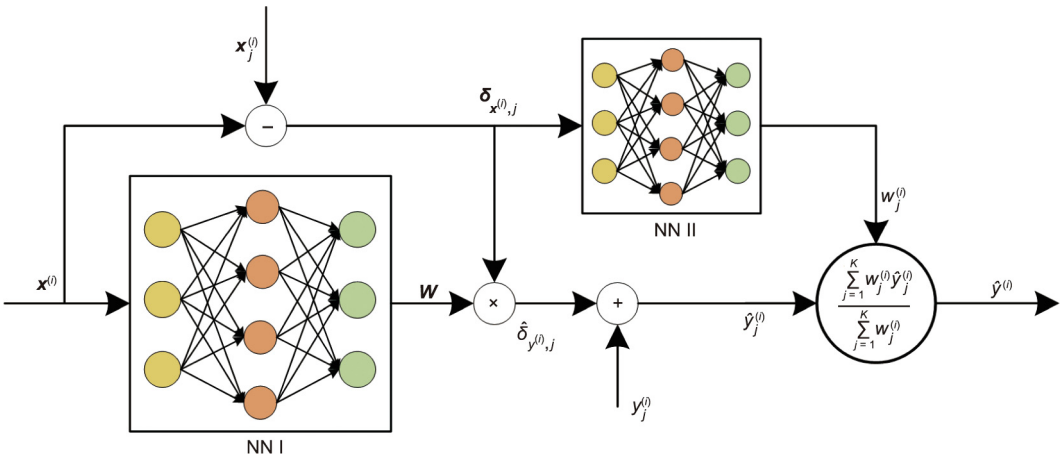


**Fig. 2.** The model structure of the proposed method.

practical industrial datasets for verification. A series of classical approaches are briefly introduced for the purpose of comparison with the proposed method. Finally, the experimental results are reported with tables and figures.

### 4.1. Descriptions of datasets

#### 4.1.1. Benchmark datasets

The details of the datasets [37–39] are shown in Table 1. For example, the red wine dataset shown in the first line contains 1599 samples. Each record contains 12 feature variables and a target label to be predicted. The objective is to establish a mathematical model to evaluate red wine quality through color, composition, and so forth. In this case, the quality of red wine is divided into nine grades from high to low, and only samples between the third and eighth grades are included in the dataset.

#### 4.1.2. Powder fineness dataset

The aim of the first practical industrial application is to make online prediction of powder fineness in the raw meal preparation process. The details of this technological process are presented in Fig. 3. In the raw meal preparation process, raw materials that consist of three or four minerals are transported onto the center of the grinding table. The materials are continuously pushed outward across the rotating grinding table due to centrifugal force. Rocks are crushed into small particles by the squeezing of the grinding rollers and the grinding table before leaving the grinding disk. When high-speed hot wind enters the mill from the bottom, finer particles are blown into the chamber, while larger particles fall to the bottom and are transported back to the entrance of the mill by a bucket elevator. High-speed airflow driven by an induced draft fan brings those finer particles into a high-efficiency dynamic classifier, where unqualified particles fall back to the mill table along the cone and get reground. Fine products gathered from cyclones and the electric dust collector are finally transported into a homogenization silo for storage.

**Table 1**
Details of the datasets used in this paper.

| Name | Prediction label | Attribute number | Sample number |
|---|---|---|---|
| Wine quality | Wine quality | 12 | 1 599 |
| Forest fire | Burned area | 12 | 517 |
| CASP | RMSD-size of residue | 10 | 45 730 |
| Air quality | CO concentration | 14 | 9 356 |
| Air quality | $NO_x$ concentration | 14 | 9 356 |
| Air quality | $NO_2$ concentration | 14 | 9 356 |
| Boston house price | House price | 13 | 506 |

CASP: critical assessment of protein structure prediction; RMSD: root mean square deviation.

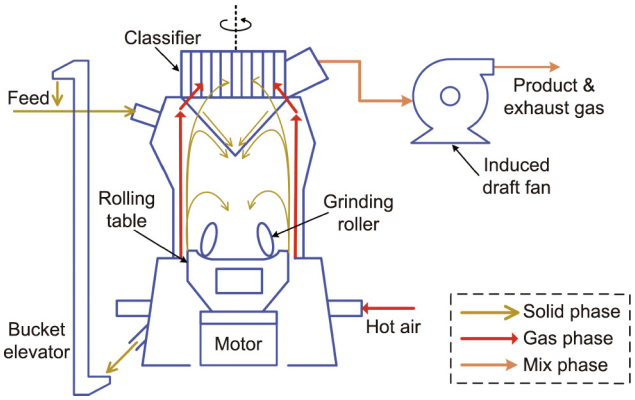


**Fig. 3.** Process flow chart of the raw meal preparation process.

The most important indicator of this process is the fineness of the product, which further influences the product quality and energy consumption of the subsequent calcination process. However, samples are collected and analyzed every 2 h due to the limited capacity for manual analysis in the lab, resulting in time lags for real-time process control and further resulting in fluctuations in raw meal fineness. Therefore, the aim is to estimate the powder fineness in real time with other available and relevant online variables—that is, to achieve soft sensing for raw meal fineness.

All of the variables that may affect or represent the fineness are considered to be auxiliary variables. These include the current of the draft fan, the current of the classifier, the current of the driven motor, the current of the bucket elevator to transport the product, the current of the bucket elevator to transport the rejected slags, the differential pressure, the inlet temperature, the outlet temperature, the feed quantity, and so forth. In general, an 80 μm sieve residue and a 200 μm sieve residue are considered to be the indicators of raw meal fineness, with the former being more sensitive. Therefore, the dataset is constructed with 14 auxiliary variables and one output label, with a total of 959 instances (about 4 months).

#### 4.1.3. Hydrocracking process dataset

The simplified flow diagram of a typical hydrocracking process is shown in Fig. 4. The feedstock is mixed with externally supplied

hydrogen, which is heated to a specified temperature and then enters the two cascade reactors. The first reactor is loaded with a hydrotreating catalyst to remove most of the sulfur and nitrogen, as well as some heavy metal compounds. The second reactor, where the cracking reaction is completed, is loaded with hydrocracking catalyst. In these reactors, low-temperature hydrogen is directly added to absorb the heat released by the exothermic reaction to maintain a stable temperature. The reaction product passes through a high-pressure separator to recycle unreacted hydrogen and then passes through a low-pressure separator to separate some light gases. Finally, the separation of different components is achieved by a fractionation tower. Six kinds of products are collected: light end (LE), light naphtha (LN), heavy naphtha (HN), kerosene (KE), diesel (DI), and bottom oil (BO).

Due to the fluctuation in product prices and changes in the market's supply and demand, the yield of different products must be relocated accordingly in order to maximize the total profit. Therefore, it is essential to accurately predict the yield of each product in time to guide the operation optimization. In this paper, we take the yield of DI as an example to establish a prediction model. In this problem, the sampling period is 4 h and the dataset covers a total of 15 months. Finally, 2052 samples with 55 related input variables, including the feed mass flow rate, volume flow of the fresh hydrogen gas, and so forth, are collected.

### 4.2. User-specified parameters

Seven typical regression algorithms are involved in this work:

(1) MML-based $k$-NNR first adopts the MML approach proposed in Ref. [27]. The model first defines the constraints based on triplets, and then formulates the optimization problem as a convex quadratic programming problem. In this algorithm, the number of nearest neighbors $K^k$ is to be determined.

(2) SVR achieves a tradeoff between structural risks and empirical risks by means of the regularization coefficient $C$ and achieves nonlinear mapping by introducing kernel methods. In this paper, different kernels such as the linear kernel, the Gaussian kernel, and the polynomial kernel are compared with each other, and the Gaussian kernel is demonstrated to be better for these regression problems. Thus, the regularization coefficient $C$ and the kernel parameter $\gamma$ are to be optimized.

(3) RF is one of the most famous bagging algorithms. It ensembles multiple weak models to reduce the variance of model predictions. Random row sampling and column sampling strategies further improve the generalization ability of the model. In this algorithm, the maximum depth $d_{\max}^{\mathrm{rf}}$ and the number of estimators $N_{\mathrm{e}}^{\mathrm{rf}}$ are optimized by fivefold cross-validation.

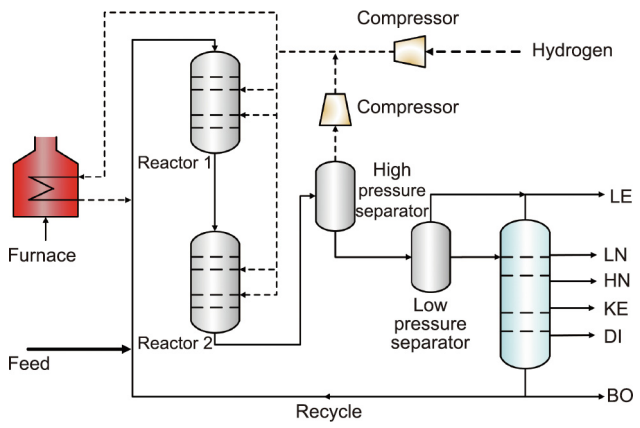


**Fig. 4.** Process flowchart of the hydrocracking process.

(4) XGBoost generates a series of estimators by fitting the residual information with weak models, which are estimated by the second-order Taylor expansion of the loss function. The algorithm has been demonstrated to have a stable and accurate prediction ability in many practical application scenarios. LightGBM [40] is one of the improved branches of XGBoost, which uses the leave-wise splitting method and applies the histogram method for preprocessing to accelerate computing. It has been demonstrated that LightGBM can greatly improve computing performance while ensuring the prediction performance. Therefore, we chose the LightGBM as one of the comparison methods. The parameters to be tuned include the maximum number of leaf nodes $N_{\mathrm{l}}^{\mathrm{lgb}}$, learning rate $\mathrm{lr}^{\mathrm{lgb}}$, and $l_2$-norm regularization coefficient $l_2^{\mathrm{lgb}}$.

(5) NNs are effective tools to solve regression problems. We implemented strategies including BN and dropout, which have been demonstrated to be the state of the art in various fields [35]. To be specific, the batch size is chosen to be 30, the proportion of dropout is 0.3, and the number of hidden neurons $N_{\mathrm{h}}^{\mathrm{n}}$ is chosen by fivefold cross-validation.

(6) Deep factorization machines (DeepFMs) [41] have made great progress in click-through rate (CTR) prediction [42] and stock market prediction [43] tasks. A DeepFM aims to learn both low- and high-order feature interactions by combining the factorization machine (FM) and a DNN. The embedding vector obtained by the FM is used as the initial embedding state of the algorithm. We apply it in regression problems where there is only one feature attribute in each field. A two-layer NN is used, which includes BN layers and dropout layers. In this algorithm, the embedding dimension $d_{\mathrm{e}}^{\mathrm{df}}$ and the number of hidden layers $N_{\mathrm{h}}^{\mathrm{df}}$ in the NN need to be optimized by cross-validation.

(7) DML-based $k$-NNR algorithms aim to find essential features by using DML algorithms and to find the most similar samples based on these features. Since this paper solves regression problems, it is impossible to construct triplet sets [29–31]. Following the principle that similar inputs lead to similar outputs, we employ the loss function, as shown in Eq. (13):

$$L = \sum_{i \neq j} \left( d_{i,j}^{y} - \left\| f\left(x^{(i)}\right) - f\left(x^{(j)}\right) \right\|_2 \right)^2 \tag{13}$$

where $d_{i,j}^{y} = \left| y^{(i)} - y^{(j)} \right|$, $f(\cdot)$ represents the embedding operator obtained by metric learning. On this basis, similar points in the feature space have similar labels; then, the $k$-NNR algorithm can be used to make a prediction. The parameters in this predictor include the embedding dimension $d_{\mathrm{e}}^{\mathrm{dml}}$ and the number of nearest neighbors $k^{\mathrm{dml}}$.

For the proposed LQEL algorithm, the parameters that need to be determined include the dimension of metric embedding $d_{\mathrm{e}}^{\mathrm{lqel}}$, the number of hidden neurons in the two employed NNs $N_{\mathrm{hs}}^{\mathrm{lqel}}$ and $N_{\mathrm{hw}}^{\mathrm{lqel}}$, and the number of neighbors $k^{\mathrm{lqel}}$. In this paper, fivefold cross-validation tests are carried out for each of these parameters to obtain the best selection scheme. The user-specified parameters utilized in our experiments are provided in Table 2.

The results in the table show that the number of nearest neighbors in the LQEL varies with different datasets. First, it depends on the scale of the dataset, which determines the density of the samples in the space. For example, in the critical assessment of protein structure prediction (CASP) dataset, the instances are sufficient for the neighbors to be better referenced for prediction. This implies that a large number of nearest neighbors can effectively improve the prediction ability of the model. However, for the industrial fineness dataset, limited samples are available for modeling. In addition, it is difficult to use the values of the instrumental variables for state representation. For example, the quantity of slag rejection in a vertical roller mill (VRM) is often evaluated by the

**Table 2**
Hyper-parameters employed in case study.

| Datasets | MML-based $k$-NNR, $K^k$ | SVR | | RF | | LightGBM | | | NN, $l_2^{lgb}$ | DeepFM | | DML-based $k$-NNR | | LQEL | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $C$ | $\gamma$ | $d_{max}^{rf}$ | $N_h^n$ | $N_e^{rf}$ | $N_l^{lgb}$ | $lr^{lgb}$ | | $d_e^{df}$ | $N_h^{df}$ | $d_e^{dml}$ | $k^{dml}$ | $d_e^{lqel}$ | $N_{hs}^{lqel}$ | $N_{hw}^{lqel}$ | $k^{lqel}$ |
| Wine quality | 50 | 1 | 0.01 | 7 | 15 | 500 | 64 | 0.1 | 0 | 3 | 256 | 16 | 50 | 10 | 2 | 3 | 80 |
| Forest fire | 40 | 100 | 10 000 | 3 | 22 | 200 | 32 | 0.001 | 1 | 3 | 128 | 8 | 10 | 5 | 1 | 4 | 100 |
| CASP | 80 | 1 000 | 100 | 4 | 18 | 600 | 128 | 0.01 | 0.1 | 8 | 256 | 8 | 80 | 14 | 2 | 4 | 140 |
| CO | 60 | 100 | 0.01 | 3 | 26 | 200 | 32 | 0.1 | 0 | 12 | 128 | 8 | 20 | 5 | 1 | 5 | 20 |
| $NO_2$ | 5 | 100 | 0.01 | 9 | 22 | 200 | 32 | 0.1 | 0.5 | 9 | 128 | 4 | 10 | 10 | 1 | 2 | 50 |
| $NO_x$ | 10 | 100 | 1 | 9 | 26 | 200 | 64 | 0.1 | 0 | 12 | 128 | 4 | 10 | 10 | 2 | 3 | 20 |
| House price | 5 | 10 000 | 10 | 7 | 22 | 50 | 32 | 0.1 | 0.5 | 9 | 128 | 16 | 5 | 20 | 4 | 3 | 50 |
| Fineness | 10 | 100 | 0.01 | 9 | 28 | 100 | 32 | 0.1 | 0 | 12 | 256 | 16 | 50 | 10 | 1 | 4 | 20 |
| Hydrocracking | 50 | 100 | 0.0001 | 7 | 110 | 50 | 64 | 0.1 | 1 | 12 | 256 | 4 | 50 | 9 | 2 | 4 | 50 |

current of the bucket elevator, but current drift occurs when regular maintenance is carried out (approximately once every 2 days), especially when lubricating oil is added. Therefore, it is necessary to pay more attention to the changes in the current. Under these circumstances, the nearest neighbors in the space may not be as instructive as those of the CASP dataset. Therefore, the model chooses a small number of neighboring samples for prediction. The table also implies that the proposed LQEL model with simple forward NNs can perform well in regression problems. Compared with the forward NN model, there are fewer hidden neurons in the LQEL model (no more than four), and a smaller scale of parameters must be estimated. This reduces the model complexity, thereby improving the model generalization.

### 4.3. Performance comparison of different datasets

To compare the performance of the proposed method with the abovementioned classical methods, a total of nine regression problems on seven datasets were used. Each experiment was repeated 30 times, and the MSE and mean absolute error (MAE) on the test sets were recorded. Then, statistical analyses were carried out on these indexes to validate the robustness of the algorithm.

Table 3 shows the average indexes of each algorithm on different datasets. The best performance for each line is marked in bold. It can be seen that, for the nine verification tests listed below, the LQEL algorithm proposed in this paper achieves the best performance on most of the datasets. Moreover, the LQEL algorithm achieves a performance comparable to those of the best-performing LightGBM and RF algorithms, and it has clear advantages when compared with other algorithms.

Moreover, to evaluate the robustness of the algorithm, it is necessary to compare the distribution of the obtained indexes. The MSE and MAE distributions of multiple repeated tests are shown with box plots in Figs. 5 and 6, respectively. The figure implies that the LQEL has the most remarkable stability on most datasets, except for the wine quality, CASP, and fineness datasets. Although the performance fluctuates slightly more than some of the other algorithms, the overall MSE and MAE are significantly lower—that is, the algorithms with more stable performances often sacrifice precision as the cost. In particular, strategies such as dropout, batch learning, and BN are implemented in both the NN and LQEL algorithms, but the latter outperforms the former.

Figs. 7 and 8 show scatter plots of the prediction results for different algorithms on the two industrial datasets, in which the abscissa is the ground truth value and the ordinate is the prediction results. The coefficient of determination ($R^2$) is marked on the top left corner, and indicates that the LQEL algorithm shows advantages over the other algorithms on these two soft sensing applications. This can be attributed to two aspects:

(1) The absolute value of the variables in these industrial datasets cannot well describe the process state. The method proposed in this paper makes corrections to the nearest neighbors according to the change of auxiliary variables, which puts greater emphasis on the differences and thus reduces the risk of the above problem.

(2) This method employs two extremely simple NNs to achieve LQEL. One NN aims to find the coefficients of local quadratic functions, and the other realizes the weight assignment for predictions

**Table 3**
Performance comparison of different algorithms.

| Dataset | Error | MML-based $k$-NNR | SVR | RF | LightGBM | NN | DeepFM | DML-based $k$-NNR | LQEL |
|---|---|---|---|---|---|---|---|---|---|
| Wine quality | MSE | 0.627 | 0.537 | 0.517 | **0.486** | 0.528 | 0.582 | 0.530 | 0.487 |
| | MAE | 0.792 | 0.733 | 0.700 | 0.697 | 0.769 | 0.768 | 0.720 | **0.684** |
| Forest fire | MSE | 3570 | 3680 | 3730 | 3620 | 3890 | 4220 | 4010 | **3470** |
| | MAE | 62.4 | 63.4 | 65.7 | 62.9 | 72.2 | 70.0 | 68.5 | **61.8** |
| CASP | MSE | 23.0 | 23.9 | **20.9** | 21.6 | 23.8 | 24.7 | 23.6 | 21.5 |
| | MAE | 4.80 | 4.89 | 4.67 | 4.65 | 4.89 | 4.87 | 4.80 | **4.59** |
| CO | MSE | 0.0690 | 0.0610 | 0.0617 | 0.0628 | 0.0838 | 0.0766 | 0.0635 | **0.0564** |
| | MAE | 0.263 | 0.247 | 0.261 | 0.251 | 0.281 | 0.301 | 0.269 | **0.246** |
| $NO_2$ | MSE | 133.0 | 212.0 | 84.3 | 77.6 | 287.0 | 205.0 | 102.0 | **63.3** |
| | MAE | 11.50 | 14.60 | 8.81 | 8.81 | 16.20 | 14.20 | 11.10 | **8.26** |
| $NO_x$ | MSE | 502 | 546 | 440 | 434 | 527 | 490 | 458 | **403** |
| | MAE | 22.4 | 23.4 | 21.1 | **20.8** | 22.7 | 22.7 | 23.0 | 20.8 |
| House price | MSE | 7.84 | 9.56 | 6.27 | 5.48 | 8.12 | 8.86 | 5.76 | **4.71** |
| | MAE | 2.80 | 3.09 | 2.34 | **2.27** | 2.98 | 2.86 | 2.51 | 2.29 |
| Fineness | MSE | 0.292 | 0.220 | 0.262 | 0.247 | 0.283 | 0.205 | 0.208 | **0.202** |
| | MAE | 0.541 | 0.448 | 0.523 | 0.497 | 0.502 | 0.447 | 0.456 | **0.431** |
| Hydrocracking | MSE | 0.0703 | 0.0843 | 0.0749 | 0.0381 | 0.0718 | 0.0393 | 0.0567 | **0.0307** |
| | MAE | 0.265 | 0.290 | 0.286 | 0.195 | 0.264 | 0.200 | 0.234 | **0.181** |

Bold values in each line indicate the best performance among different algorithms.

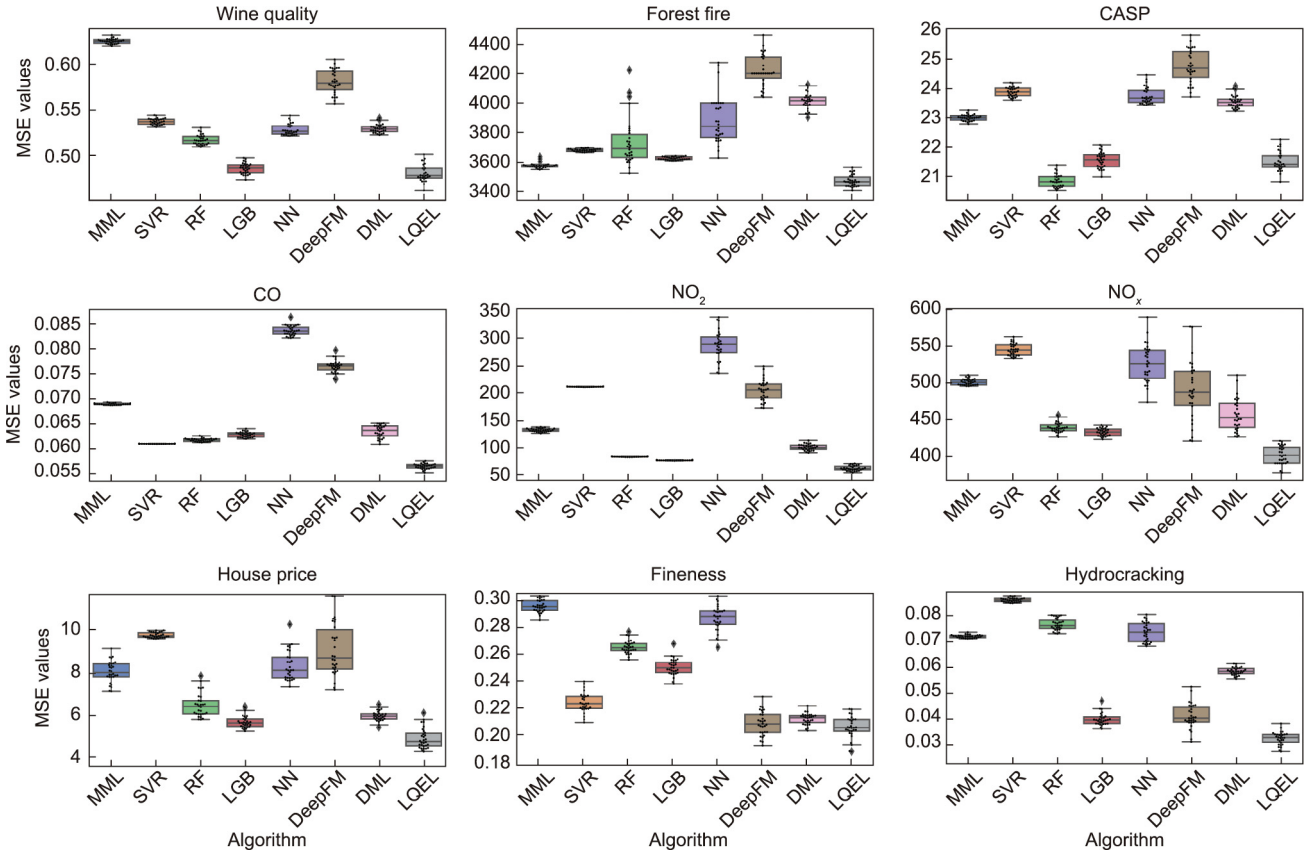


**Fig. 5.** MSE box plots of algorithms tested on different datasets. MML: MML-based *k*-NNR; LGB: LightGBM; DML: DML-based *k*-NNR.

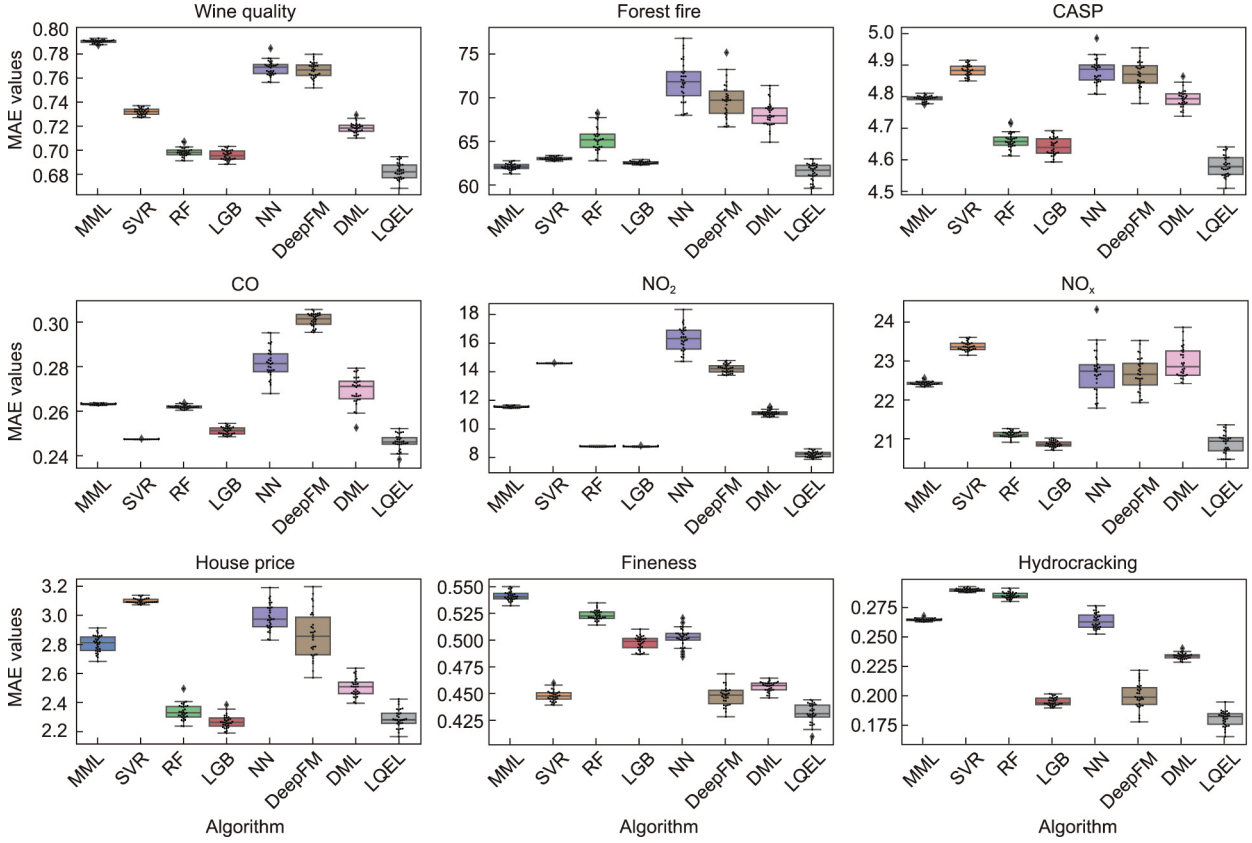


**Fig. 6.** MAE box plots of algorithms tested on different datasets.

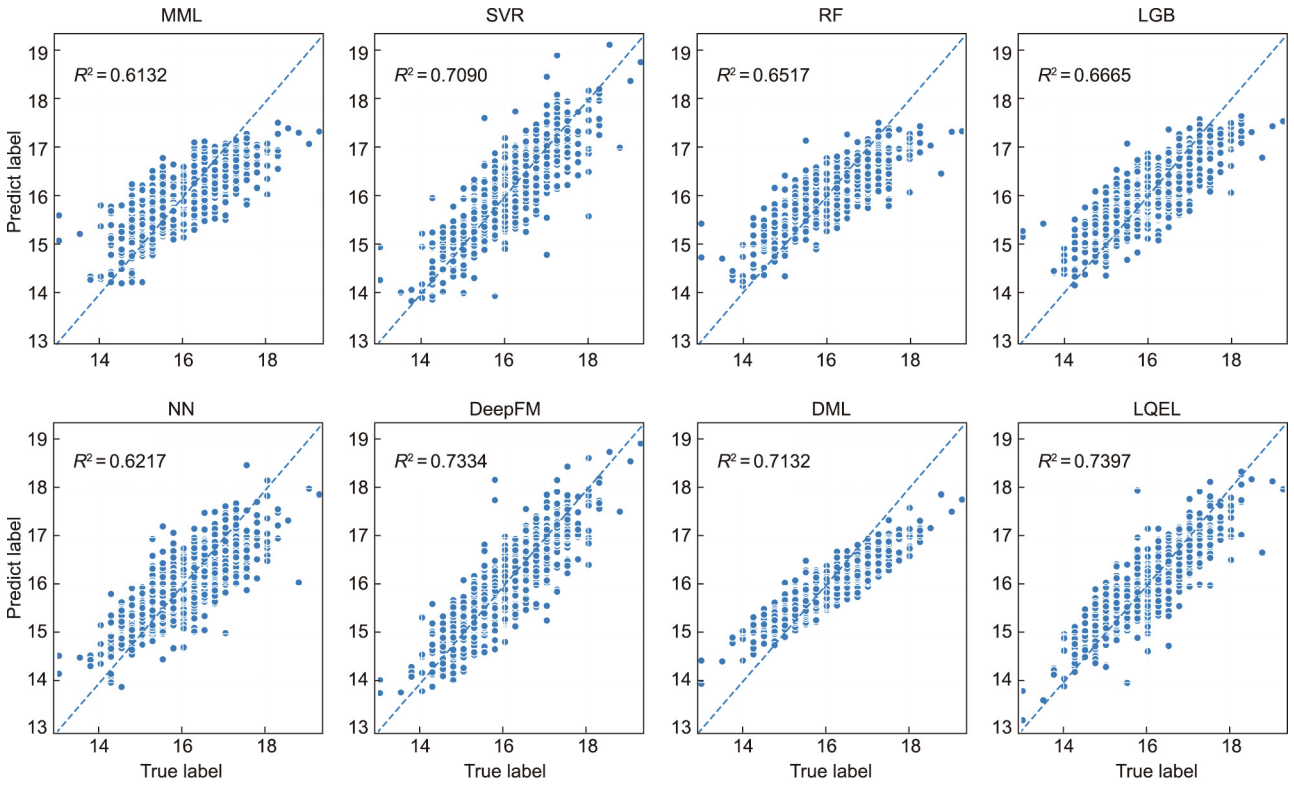


**Fig. 7.** Scatter plots of the prediction results for different algorithms on the fineness dataset.

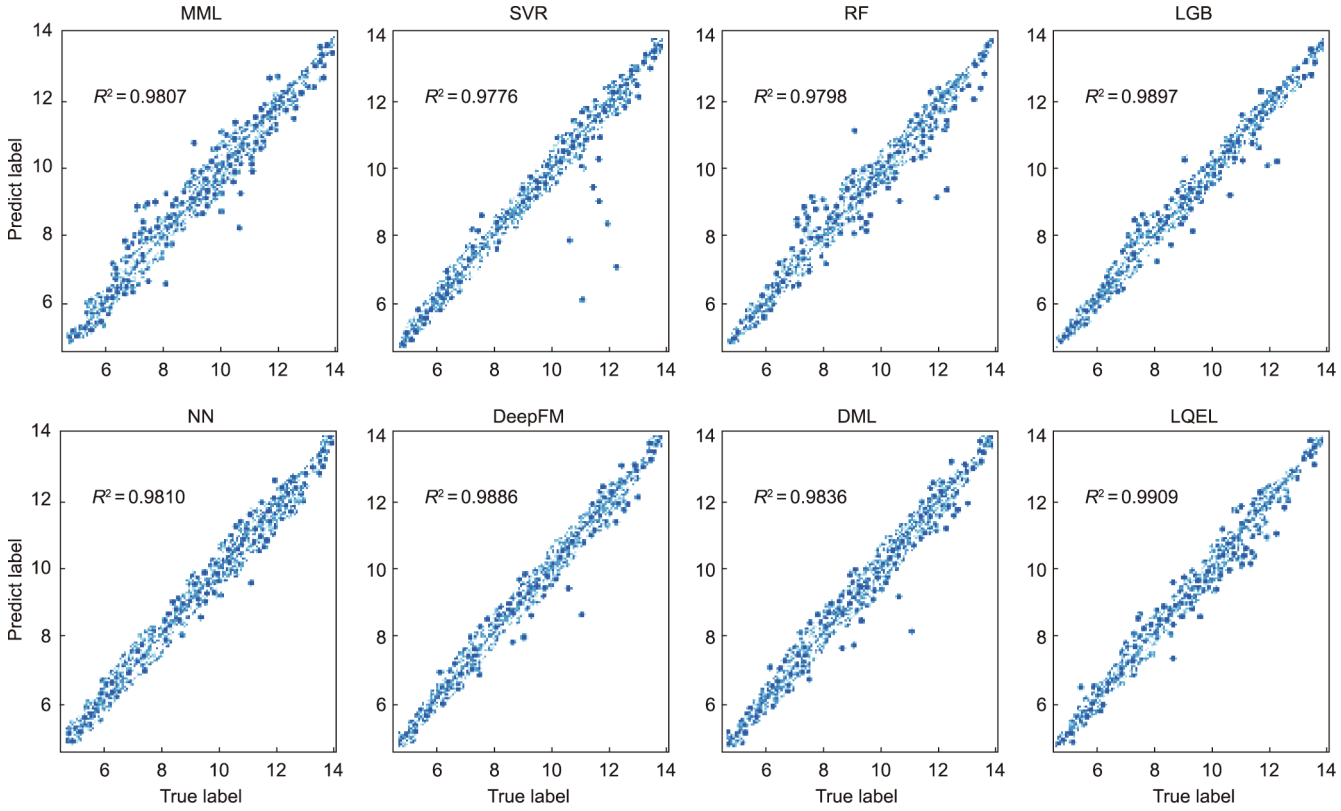


**Fig. 8.** Scatter plots of the prediction results for different algorithms on the hydrocracking dataset.

given by nearest neighbors. Based on these advantages, the generalization ability of the proposed algorithm can be effectively improved.

## 5. Conclusions

The paper proposed an LQEL algorithm for regression problems. MML is first improved by optimizing the consistency of the distances between samples in the input and output space. By relaxing the constraints, the modified problem is proved to be a convex optimization problem, while it keeps the same solution as the original problem. Based on this, a locally quadratic embedding model is developed, and different weights are assigned to the prediction results to minimize the expectation of prediction error. In this framework, two extremely simple NNs are implemented to learn the quadratic embedding matrix and the weight assignments of the neighboring predictions. We hope to build a unified end-to-end model that prevents the independent two-layer optimization from getting stuck in a local optimal. The proposed LQEL model has the following advantages:

- A global consistency for distances in the input and output space is achieved via improved metric learning.
- The information contained in output labels is better exploited, which leads to a better determination of the neighborhood for a certain instance.
- An LQEL framework was proposed based on the local quadratic embedding hypothesis. Two specially designed networks improve generalization by simplifying the model structure from either a global or a local perspective.
- The experimental results show that the LQEL can achieve a more precise and comparable robust prediction when lightweight NNs are employed.

## Acknowledgments

This work was supported by the National Key Research and Development Program of China (2016YFB0303401), the International (Regional) Cooperation and Exchange Project (61720106008), the National Science Fund for Distinguished Young Scholars (61725301), and the Shanghai AI Lab.

## Compliance with ethics guidelines

Yaoyao Bao, Yuanming Zhu, and Feng Qian declare that they have no conflict of interest or financial conflicts to disclose.

## References

[1] Bao YY, Zhu YM, Du WL, Zhong WM, Qian F. A distributed PCA-TSS based soft sensor for raw meal fineness in VRM system. Control Eng Pract 2019;90:38–49.
[2] Zhong WM, Jiang C, Peng X, Li Z, Qian F. Online quality prediction of industrial terephthalic acid hydropurification process using modified regularized slow-feature analysis. Ind Eng Chem Res 2018;57(29):9604–14.
[3] Lu B, Chiang L. Semi-supervised online soft sensor maintenance experiences in the chemical industry. J Process Contr 2018;67:23–34.
[4] Li YG, Gui WH, Yang CH, Xie YF. Soft sensor and expert control for blending and digestion process in alumina metallurgical industry. J Process Contr 2013;23(7):1012–21.
[5] Song YC, Zhou H, Wang PF, Yang MJ. Prediction of clathrate hydrate phase equilibria using gradient boosted regression trees and deep neural networks. J Chem Thermodyn 2019;135:86–96.
[6] Touzani S, Granderson J, Fernandes S. Gradient boosting machine for modeling the energy consumption of commercial buildings. Energ Build 2018;158:1533–43.
[7] Fernández-Delgado M, Sirsat MS, Cernadas E, Alawadi S, Barro S, Febrero-Bande M. An extensive experimental survey of regression methods. Neural Netw 2019;111:11–34.
[8] Yu J. Multiway Gaussian mixture model based adaptive kernel partial least squares regression method for soft sensor estimation and reliable quality prediction of nonlinear multiphase batch processes. Ind Eng Chem Res 2012;51(40):13227–37.
[9] Yuan XF, Ge ZQ, Song ZH. Locally weighted kernel principal component regression model for soft sensing of nonlinear time-variant processes. Ind Eng Chem Res 2014;53(35):13736–49.
[10] Gou JP, Ma HX, Ou WH, Zeng SN, Rao YB, Yang HB. A generalized mean distance-based *k*-nearest neighbor classifier. Expert Syst Appl 2019;115:356–72.
[11] Juez-Gil M, Erdakov IN, Bustillo A, Pimenov DY. A regression-tree multilayer-perceptron hybrid strategy for the prediction of ore crushing-plate lifetimes. J Adv Res 2019;18:173–84.
[12] Suarez A, Lutsko JF. Globally optimal fuzzy decision trees for classification and regression. IEEE Trans Pattern Anal Mach Intell 1999;21(12):1297–311.
[13] Huang GB, Zhou HM, Ding XJ, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern 2012;42(2):513–29.
[14] Vilela LFS, Leme RC, Pinheiro CAM, Carpinteiro OAS. Forecasting financial series using clustering methods and support vector regression. Artif Intell Rev 2019;52(2):743–73.
[15] Paul A, Mukherjee DP. Reinforced quasi-random forest. Pattern Recognit 2019;94:13–24.
[16] Rodriguez-Galiano VF, Ghimire B, Rogan J, Chica-Olmo M, Rigol-Sanchez JP. An assessment of the effectiveness of a random forest classifier for land-cover classification. ISPRS J Photogramm Remote Sens 2012;67:93–104.
[17] Freund Y. Boosting a weak learning algorithm by majority. Inf Comput 1995;121(2):256–85.
[18] Chen TQ, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA; 2016. p. 785–94.
[19] Dreiseitl S, Ohno-Machado L. Logistic regression and artificial neural network classification models: a methodology review. J Biomed Inform 2002;35(5–6):352–9.
[20] Zhou ZH, Wu JX, Tang W. Ensembling neural networks: many could be better than all. Artif Intell 2002;137(1–2):239–63.
[21] Liu WB, Wang ZD, Liu XH, Zeng NY, Liu YR, Alsaadi FE. A survey of deep neural network architectures and their applications. Neurocomputing 2017;234:11–26.
[22] Martos G, Muñoz A, González J. On the generalization of the Mahalanobis distance. In: Ruiz-Shulcloper J, Sanniti di Baja G, editors. Progress in pattern recognition, image analysis, computer vision, and applications. Berlin: Springer; 2013. p. 125–32.
[23] Atkeson CG, Moore AW, Schaal S. Locally weighted learning. Artif Intell Rev 1997;11:11–73.
[24] Zhang JJ. Identification of moving loads using a local linear embedding algorithm. J Vib Control 2019;25(11):1780–90.
[25] Loia V, Tomasiello S, Vaccaro A, Gao JW. Using local learning with fuzzy transform: application to short term forecasting problems. Fuzzy Optim Decis Making 2020;19(1):13–32.
[26] Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res 2009;10:207–44.
[27] Nguyen B, Morell C, De Baets B. Large-scale distance metric learning for *k*-nearest neighbors regression. Neurocomputing 2016;214:805–14.
[28] Xing EP, Ng AY, Jordan MI, Russell S. Distance metric learning, with application to clustering with side-information. In: Becker S, Thrun S, Obermayer K, editors. Advances in neural information processing systems 15: proceedings of the 2002 conference. Cambridge: A Bradford Book; 2003. p. 521–8.
[29] Duan YQ, Lu JW, Zheng WZ, Zhou J. Deep adversarial metric learning. IEEE Trans Image Process 2020;29:2037–51.
[30] Song HO, Xiang Y, Jegelka S, Savarese S. Deep metric learning via lifted structured feature embedding. In: Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 26–Jul 1; Las Vegas, NV, USA; 2016. p. 4004–12.
[31] Cui Y, Zhou F, Lin YQ, Belongie S. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In: Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 26–Jul 1; Las Vegas, NV, USA; 2016. p. 1153–62.
[32] ALzubi JA, Bharathikannan B, Tanwar S, Manikandan R, Khanna A, Thaventhiran C. Boosted neural network ensemble classification for lung cancer disease diagnosis. Appl Soft Comput 2019;80:579–91.
[33] Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. 2015. arXiv:1502.03167.
[34] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. 2012. arXiv:1207.0580.
[35] Kay S. Can detectability be improved by adding noise? IEEE Signal Process Lett 2000;7(1):8–10.
[36] Boyd V, Vandenberghe L, Faybusovich L. Convex optimization. IEEE Trans Automat Contr 2006;51(11):1859.
[37] Cortez P, Morais A. A data mining approach to predict forest fires using meteorological data. In: Neves JM, Santos MF, Machado JM, editors. New trends in artificial intelligence: proceedings of the 13th Portuguese Conference on Artificial Intelligence; 2007 Dec 3–7; Guimarães, Portugal; 2007. p. 512–23. French.

[38] Cortez P, Cerdeira A, Almeida F, Matos T, Reis J. Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 2009;47(4):547–53.
[39] De Vito S, Massera E, Piga M, Martinotto L, Di Francia G. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sens Actuators B Chem 2008;129(2):750–7.
[40] Ke GL, Meng Q, Finley T, Wang TF, Chen W, Ma WD, et al. LightGBM: a highly efficient gradient boosting decision tree. In: Guyon I, Von Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Proceedings of the 31st Annual Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA; 2017.
[41] Guo HF, Tang RM, Ye YM, Li ZG, He XQ. DeepFM: a factorization-machine based neural network for CTR prediction. In: Sierra C, editor. Proceedings of the 26th International Joint Conference on Artificial Intelligence; 2017 Aug 19–25; Melbourne, VIC, Australia; 2017. p. 1725–31.
[42] Zhang L, Shen WC, Huang JH, Li SJ, Pan G. Field-aware neural factorization machine for click-through rate prediction. IEEE Access 2019;7:75032–40.
[43] Huang JY, Zhang X, Fang BX. CoStock: a DeepFM model for stock market prediction with attentional embeddings. In: Proceedings of 2019 IEEE International Conference on Big Data; 2019 Dec 9–12; Los Angeles, CA, USA. New York City: IEEE; 2019. p. 5522–31.