Information Sciences 699 (2025) 121793

Contents lists available at ScienceDirect

Information Sciences

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

A novel robust data synthesis method based on feature subspace interpolation to optimize samples with unknown noise

Yukun Du [a,1], Yitao Cai [b,1], Xiao Jin [c,1], Haiyue Yu [a,*], Zhilong Lou [b], Yao Li [b], Jiang Jiang [a], Yongxiong Wang [c,*]

[a] College of System Engineering, National University of Defense Technology, 410073, Changsha, China
[b] School of Statistics and Data Science, Nanjing Audit University, 211815, Nanjing, China
[c] School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, 200093, Shanghai, China

ARTICLE INFO

Keywords:
Data synthesis
Feature subspace
Interpolation
Optimize samples
Robust

ABSTRACT

Sample noise, characterized by complex and elusive distribution patterns, poses a significant challenge to the efficacy of data analysis and machine learning models, especially in scenarios of data scarcity and high dimensionality. Traditional methods for managing datasets with unknown noise often result in the loss of original information and lead to overgeneralization. Most existing data synthesis methods do not adequately address noisy dataset conditions because they were not specifically designed to handle sample noise. This paper proposes a novel data synthesis approach, the Robust Subspace Interpolation Synthesis (RSIS), which is based on feature subspace linear and equidistant interpolation. This technique synthesizes samples with reduced noise without sacrificing the original sample information and adaptively interpolates between feature subspaces to enhance data diversity and representativeness. Contrary to the common assumption that sample noise is identically distributed, we theoretically prove that the RSIS method maintains optimization capabilities across varying noise distributions, which is further validated in simulation experiments. Benchmark dataset experiments reveal that RSIS is applicable across various models, significantly enhancing their predictive performance. RSIS exhibits excellent robustness, even when processing samples with high-variance noise and under practical conditions that deviate from RSIS assumptions. Additionally, by integrating RSIS with convolutional neural networks, our experiments demonstrate its effectiveness in improving image prediction accuracy.

1. Introduction

Data noise, obscuring true information, severely undermines the reliability and accuracy of analytical and predictive models. It refers to any inaccurate, irrelevant, or misleading information within the data, which may stem from various reasons such as instrument errors, data entry mistakes, transmission issues, or even systematic biases in the data collection process [1,2]. In many practical scenarios, the distribution of noise is intricate and not well-defined, leading to substantial reductions in the efficacy of data analysis and machine learning models. This problem is particularly critical in fields that demand highly accurate predictions and

* Corresponding authors.
E-mail addresses: haiyue_nudt@163.com (H. Yu), xjin6868@163.com (Y. Wang).
[1] Equal contributions.

https://doi.org/10.1016/j.ins.2024.121793
Received 2 April 2024; Received in revised form 11 November 2024; Accepted 18 December 2024

Available online 24 December 2024

analyses, such as medical image analysis, quantitative finance, and environmental surveillance [3,4]. In these domains, even minor data errors can lead to serious consequences. Therefore, effectively identifying and handling noise in the data has become a crucial technique in improving the accuracy and reliability of predictive models.

To address the problem of datasets with unknown noise, researchers have developed a variety of methods, including data cleaning [5,6], data smoothing [7,8], and outlier detection [9]. These methods primarily focus on mitigating or removing noise from the data, but their effectiveness is limited and can result in information loss [10]. For instance, outliers may in some cases be an important information in data analysis, while data cleaning might mistakenly identify them as noise and remove them [11]. Similarly, data smoothing techniques, such as moving averages or Gaussian smoothing, can reduce random fluctuations in the data but may also lead to overgeneralization. This overgeneralization can obscure important features and details within the data, thereby affecting the accuracy of data analysis. Moreover, traditional noise handling methods often perform poorly when dealing with complex and high-dimensional datasets. In high-dimensional data, the boundary between noise and useful information can become more blurred, potentially leading to the erroneous deletion of valuable data or the retention of excessive noise during the noise reduction process [12].

While traditional methods offer some effectiveness under specific conditions, they often fall short in real-world applications characterized by complex noise patterns, particularly in datasets with a limited number of samples. In addition, existing data synthesis methods, which aim to perform data augmentation, sample expansion, and simulation by generating non-real yet statistically representative data, do not exhibit robust performance on noisy datasets, primarily because most were not designed to address sample noise issues [13,14]. Against this backdrop, we introduce a novel data synthesis method for datasets with unknown noise, based on the concept of feature subspace interpolation, named Robust Subspace Interpolation Synthesis (RSIS). This method can optimize the original dataset in two ways. Firstly, the RSIS method can synthesize samples with less noise without losing the information of the original samples, not only reducing the average error of the samples but also decreasing the proportion of high-noise samples. Secondly, the RSIS method can adaptively interpolate samples between feature subspaces to increase data diversity and representativeness, better highlighting the true functional relationships between variables. Compared to traditional methods, RSIS is more suitable for sample noise processing in the field of machine learning, effectively enhancing the generalization ability of models, even when dealing with high-dimensional data containing unknown complex noise.

The RSIS method includes several key steps. First, using the unsupervised clustering algorithm we proposed, the original dataset is divided into multiple subsets with approximately equal sample sizes, each corresponding to a feature subspace. Then, utilizing the clustering results, the RSIS method introduces the concept of the Traveling Salesman Problem (TSP) to order the feature subspaces. Subsequently, we integrate the mechanism of soft parameter sharing, conducting a linear fitting between adjacent subspaces considering global information based on the ordering results. Finally, we introduce the minimum weight matching problem based on our designed interpolation matching rules. To address this scenario, we propose an innovative multi-stage minimum weight matching method. Through this solution, a suboptimal interpolation matching strategy is obtained, allowing for the generation of samples with smaller errors through linear and equidistant interpolation of adjacent subspaces.

Currently, most studies assume that sample noise is identically distributed, which is rare in practical applications. We theoretically prove that the RSIS method can still exhibit good optimization effects even when faced with noise of different distributions. We also simulated samples with unknown and differently distributed complex noise, where the RSIS method demonstrated good robustness and significantly optimized the samples. In addition, experimental results of benchmark datasets demonstrate exceptional adaptability with different models, generally improving their prediction accuracy after optimization by RSIS, and even if there are violations of the RSIS assumptions in practical applications, this method may still achieve good optimization results. Furthermore, we apply the RSIS method in conjunction with convolutional neural networks (CNN) to the field of computer vision, where experimental results show that the RSIS method can also effectively improve the generalization ability of image classification models.

2. Related work

Existing data synthesis methods are widely used in scenarios of data scarcity, privacy protection, or model training. For instance, in the healthcare field, which demands stringent data privacy, researchers utilize synthetic data to improve the accuracy and efficiency of disease diagnosis through model training, while ensuring compliance and diversity, thus advancing medical science research and innovation [15,16]. Currently, the main methods of data synthesis are as follows.

Rule-based methods depend on predefined rules or algorithms to modify existing data or generate new data instances. Classic techniques such as linear interpolation and polynomial interpolation utilize known data points to estimate unknown points [17,18]. Data Augmentation is a technique used to increase the diversity of training data, thereby improving the generalization capability of machine learning models. It is widely used in fields such as image processing, natural language processing, and speech recognition. For image data, data augmentation enhances dataset diversity through predefined transformations such as rotation, scaling, flipping, and cropping. Techniques like MixUp and CutMix further enhance model generalization by synthesizing new samples through proportional mixing of random image samples, following predefined rules [13,19]. Furthermore, simulation methods such as Monte Carlo simulations and surrogate models employ random sampling or simplified representations to accurately emulate data distributions. This approach facilitates the analysis of complex system behaviors, enables rapid assessment of system responses, and significantly enhances the quality and accessibility of data [20].

Statistical-based methods can be subdivided into parametric and non-parametric, utilizing statistical models to estimate data distributions and thereby generate data. Within parametric approaches, the synthesis of data is contingent upon predefined assumptions about distributions, with typical methods including Gaussian mixture models and Bayesian networks [21,22]. These models assume

that the data follows certain recognized probability distributions, such as normal, Poisson, or other parametric distributions, and then utilize these distributions to generate new data points, maintaining correlations within the data. Note that the performance of parametric methods is significantly contingent upon the underlying assumptions and selections inherent to the model, necessitating a degree of prior knowledge concerning the distributions of data [23,24]. Non-parametric methods, such as histogram approaches, kernel density estimation, and SMOTE methods along with their variants (SVM-SMOTE, SMOTE-ENC), do not rely on fixed distribution models but infer the shape of distributions directly from the data, thereby affording greater flexibility in accommodating various types and configurations of data distributions [25,26].

Data synthesis methods based on machine learning have seen rapid development in recent years, particularly with the emergence of technologies like Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Conditional Generative Adversarial Networks (cGANs). GANs introduced an innovative adversarial training mechanism, where the generator and discriminator engage in competition, enabling the generator to produce increasingly realistic data [27]. VAEs model high-dimensional data distributions by optimizing the variational lower bound of the latent space, providing a robust mathematical framework for the analysis of complex data distributions [28]. cGANs, by incorporating conditional variables, allow for the addition of specific conditions during the data generation process, thus providing enhanced flexibility and control in data synthesis [29,30]. Moreover, autoregressive models such as PixelRNN and PixelCNN have demonstrated strong capabilities in sequentially generating image and text data, by learning the sequential dependency relationships within the data [31,32].

Hybrid methods aim to enhance the quality, diversity, and practicality of synthetic data by integrating various data synthesis techniques, such as Bayesian Gaussian Mixture Models (Bayesian GMM), Dirichlet Process Mixture Models (DPMM), and Hierarchical Mixture Models (HMM). The Bayesian GMM utilizes Bayesian approaches to estimate the parameters of Gaussian Mixture Models, providing a posterior distribution of parameters that more accurately reflects their uncertainty [33,34]. The DPMM, a non-parametric Bayesian approach, adapts automatically to data, enabling a flexible number of infinite latent categories or components for accurately modeling data distributions, thereby producing synthetic data that mirrors the complex structure of the original data [35]. The HMM adds a hierarchical structure to mixture models, allowing for variability at different levels and facilitating the creation of synthetic data that more closely resembles the inherent hierarchical organization of the data [36].

Existing data synthesis methods do not exhibit robust performance on noisy datasets, primarily because most were not designed to address sample noise issues. For instance, the SMOTE algorithm aims to balance sample data in classification tasks by generating synthetic samples for the minority class, but it does not account for the presence of noise, potentially amplifying noisy data. GANs and VAEs strive to learn the underlying distribution of data to generate new samples indistinguishable from the original dataset. However, these methods often struggle with noisy datasets as GANs are sensitive to noise, affecting training stability, and VAEs may not effectively capture complex noise patterns, leading to less accurate representations of the true data distribution. Additionally, GMMs assume that data can be represented as a mixture of several Gaussian distributions, which are symmetric and have light tails. If the noise in the data does not follow a Gaussian distribution, such as having heavy tails or being asymmetric, GMMs may fail to effectively capture and handle this noise. RSIS, a novel data synthesis approach derived from hybrid methods, effectively addresses the challenges of noisy datasets. It showcases the potential of advanced synthesis techniques to optimize samples, thereby improving the performance and generalization capacity of machine learning models when dealing with complex data.

3. Proposed method

The RSIS method is divided into four steps: The first step involves dividing the original feature space into multiple subspaces by unsupervised clustering method, with the RSIS method requiring that these subspaces have nearly equal sample sizes.

Subsequently, the subspaces are ranked in order, with this step of the process transforming the subspace interpolation sorting issue into TSP to be solved with the goal of minimizing the total distance.

We integrate the principle of soft parameter sharing mechanisms, executing linear fitting on samples across adjacent subspaces. In this framework, while each task possesses its distinct model with unique weights, the divergence between the parameters of these task-specific models is incorporated into the combined objective function, which ensures that the fitting functions consider global information.

Lastly, we introduce a multi-stage minimum weight matching algorithm that can quickly obtain effective linear interpolation matching strategies for samples between adjacent subspaces. Based on the obtained interpolation matching strategies, multiple interpolations are performed on samples between adjacent subspaces. The samples synthesized through linear and equidistant interpolation between feature subspaces exhibit smaller noise errors compared to the original samples, achieving the objectives of sample optimization. The overall flowchart of the RSIS method is shown in Fig. 1.

3.1. Dividing of feature spaces

We propose an unsupervised clustering algorithm that can partition the feature space into multiple subspaces with similar sample sizes. This algorithm consists of two stages: hierarchical clustering and the K-Nearest Neighbors optimization.

3.1.1. Hierarchical clustering

The design of the hierarchical clustering process is primarily based on the concept of a greedy strategy, implemented through an unsupervised algorithm that aggregates clusters via multiple iterative cycles. An initial hyperparameter k is required, where k intuitively represents the number of subspaces into which the original feature space is partitioned.

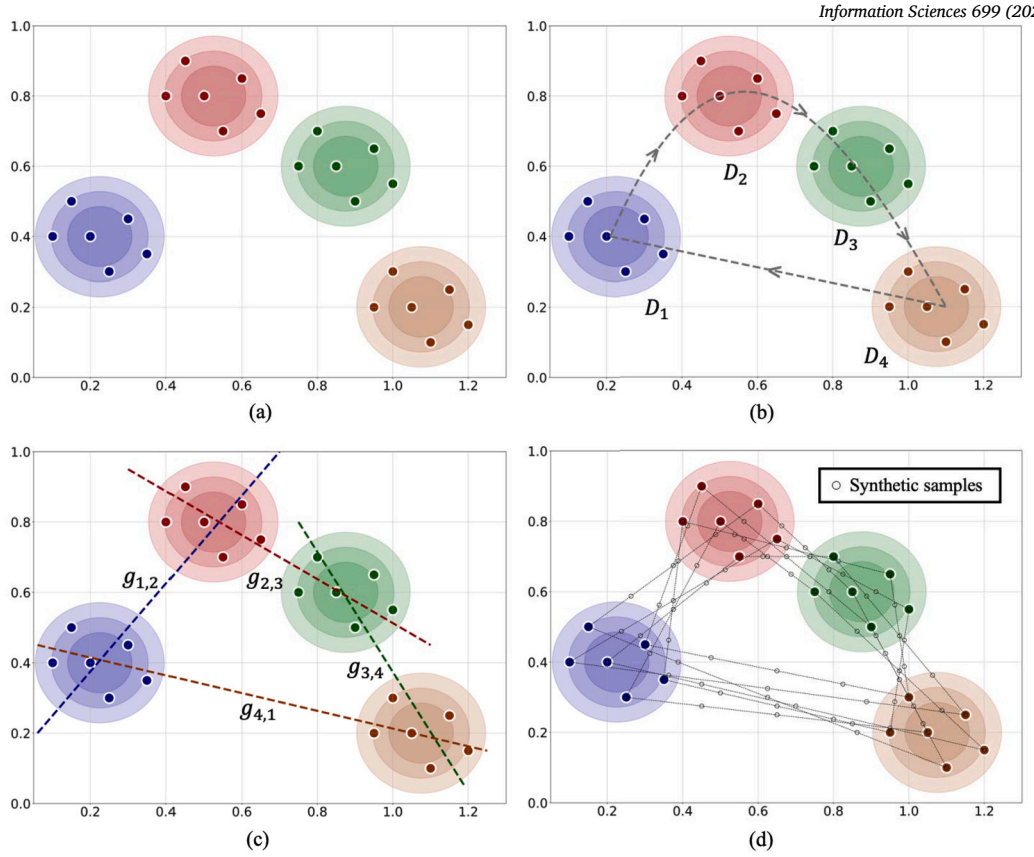

Fig. 1. The overall flowchart of the RSIS method in a 2D scenario. (a) The original feature space is divided into several feature subspaces with approximately equal sample sizes through an unsupervised clustering algorithm. (b) The TSP problem is solved for the case of a two-dimensional feature space, resulting in a subspace interpolation sorting scheme. (c) Performing linear fitting between subspaces with adjacent sorting. (d) Based on the results of linear fitting, four sample interpolation matching strategies between adjacent subspaces are obtained through the multi-stage minimum weight matching method, and interpolation is performed to synthesize samples with smaller errors.

In the initial iteration process, an ordered set of sample features is given as $C_1 = \{\boldsymbol{c}_i\}_{i=1}^{n} = \{\boldsymbol{x}_i\}_{i=1}^{n}$, where the $\boldsymbol{c}_i$ is the i-th element in set C_1. Subsequently, the closest pair of elements $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ within C_1 is identified, where $p, q = \text{argmin}_{p,q;p\neq q}\ \text{dist}(\boldsymbol{c}_p, \boldsymbol{c}_q)$. Based on this, the set $D_1 = \{\boldsymbol{c}_p, \boldsymbol{c}_q\}$, encompassing the two nearest elements, is defined as the first cluster, and calculate the centroid according to Formula (1):

$$\bar{\boldsymbol{x}}^s = \frac{\sum_{\boldsymbol{x}^s \in D_s} \boldsymbol{x}^s}{\text{num}(D_s)}, \tag{1}$$

where $\text{num}(D_s)$ is the number of samples in set D_s. At the end of the first iteration, $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ are removed from the set C_1, and add the centroid of D_1 to C_1. i.e., $C_2 = \{\boldsymbol{c}_i \in C_1 | i \neq p, q\} \cup \{\bar{\boldsymbol{x}}^1\}$.

In subsequent iterations, it is required to determine the two elements $\boldsymbol{c}_p, \boldsymbol{c}_q$ with the closest distance in the set C_t. Three scenarios are considered for $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$: both are original samples; one is a sample and the other is a cluster centroid; both are cluster centroids.

In the scenario where both $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ are original samples, the set $D_s = \{\boldsymbol{c}_p, \boldsymbol{c}_q\}$ is defined as a new cluster and its centroid $\bar{\boldsymbol{x}}^s$ is computed. $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ are then removed from set C_t and $\bar{\boldsymbol{x}}^s$ is added to C_t, i.e., $C_{t+1} = \{\boldsymbol{c}_i \in C_t | i \neq p, q\} \cup \{\bar{\boldsymbol{x}}^s\}$.

In the scenario where $\boldsymbol{c}_p$ is an original sample and $\boldsymbol{c}_q : \bar{\boldsymbol{x}}^s$ is a cluster centroid, add $\boldsymbol{c}_p$ to D_s. Then, $\boldsymbol{c}_p$ is removed from set C_t, and the centroid of D_s within C_t is recalculated and updated according to Formula (1), resulting in $C_{t+1} = \{\boldsymbol{c}_i \in C_t | i \neq p, q\} \cup \{\bar{\boldsymbol{x}}^{s'}\}$.

If both $\boldsymbol{c}_p : \bar{\boldsymbol{x}}^s$ and $\boldsymbol{c}_q : \bar{\boldsymbol{x}}^l$ are centroids, then the two clusters are merged by $D_s \leftarrow D_s \cup D_l$, and the centroid of D_s is recalculated. We can obtain $C_{t+1} = \{\boldsymbol{c}_i \in C_t | i \neq p, q\} \cup \{\bar{\boldsymbol{x}}^{s'}\}$.

At the end of each iteration, if $\text{num}(D_s) \geq \lceil n/k \rceil$, remove $\bar{\boldsymbol{x}}^s$ from C_{t+1}. Subsequently, the Local Outlier Factor (LOF) algorithm [37] is employed to detect the outliers in D_s, and eliminate the superfluous samples $\{\boldsymbol{x}_i^s\}_{i=\lceil n/k \rceil+1}^{\text{num}(D_s)}$ from D_s. D_s is then considered a complete cluster and excluded from further iterations. Lastly, the redundant samples are reinserted into C_{t+1}. The iterative process ends when $C_{t+1} = \emptyset$. The LOF method involves a hyperparameter, representing the number of neighbors used to calculate the local density of each data point. In practical applications, setting this hyperparameter is not particularly critical and does not significantly affect the effectiveness of the RSIS method. In our experiments, we set the LOF algorithm's hyperparameter to the integer value of the

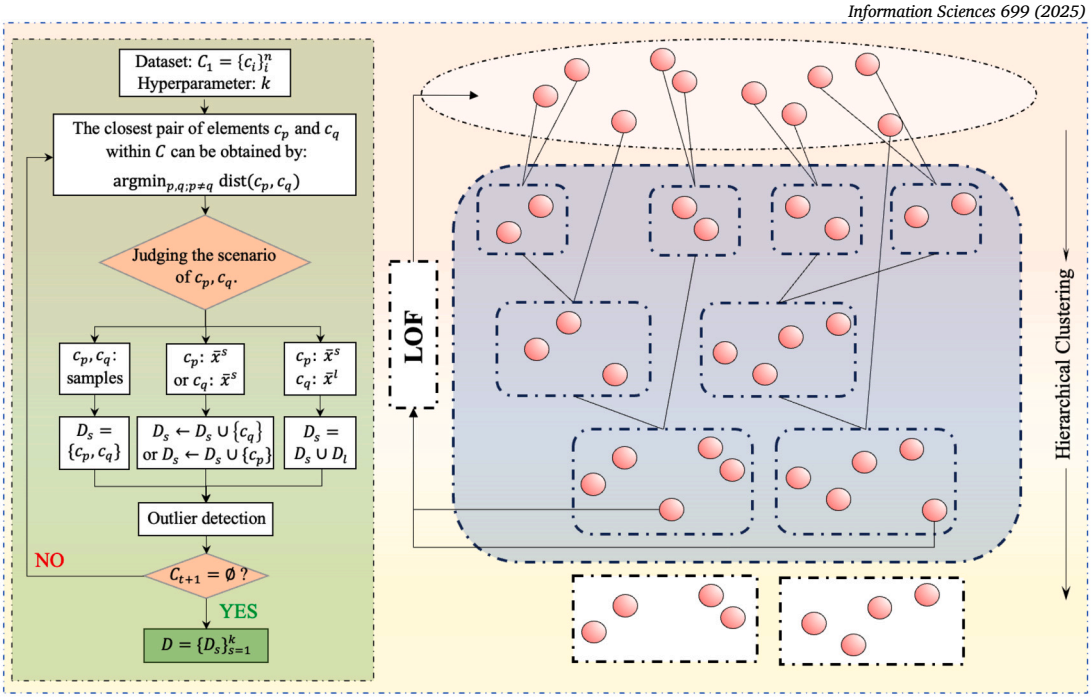

Fig. 2. The hierarchical clustering process of RSIS.

ratio between the sample size and the number of clusters. The detailed procedural flow for this segment can be referenced in Fig. 2, and Algorithm 1.

Algorithm 1 Hierarchical clustering.

Input: the set of sample features C, hyperparameter k
Output: $D = \{D_s\}_{s=1}^k$

While: $C \neq \emptyset$:
  Obtain $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ by $\text{argmin}_{p,q;p\neq q}\ \text{dist}(\boldsymbol{c}_p, \boldsymbol{c}_q)$
  Judging the situation of $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$:
    If both $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ are samples:
      Define the new cluster $D_s = \{\boldsymbol{c}_p, \boldsymbol{c}_q\}$
      Calculate the centroid of D_s and add it to C
      Remove $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ from C
    Else if $\boldsymbol{c}_p : \bar{\boldsymbol{x}}^s$ or $\boldsymbol{c}_q : \bar{\boldsymbol{x}}^s$:
      $D_s \leftarrow D_s \cup \{\boldsymbol{c}_q\}$ or $D_s \leftarrow D_s \cup \{\boldsymbol{c}_p\}$
      Recalculate the centroid of D_s and renew it in C
      Remove $\boldsymbol{c}_q$ or $\boldsymbol{c}_p$ from C
    Else if both $\boldsymbol{c}_p : \bar{\boldsymbol{x}}^s$ and $\boldsymbol{c}_q : \bar{\boldsymbol{x}}^l$ are centroids:
      $D_s \leftarrow D_s \cup D_l$
      Recalculate the centroid of D_s and add it to C
      Remove $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ from C
  If $\text{num}(D_s) \geq \lceil n/k \rceil$:
    Remove $\bar{\boldsymbol{x}}^s$ from C
    Detect the redundant samples in D_s by LOF algorithm
    Remove $\{\boldsymbol{x}_i^s\}_{i=\lceil n/k \rceil+1}^{\text{num}(D_s)}$ from D_s and add them to C
    $D \leftarrow D \cup D_s$

Consequently, the iterative process can partition the original dataset into $k-1$ subsets, each containing an equal number of samples; and one additional subset consisting of no more than $\lceil n/k \rceil$ samples. Formally, we define $D = \cup_{s=1}^k D_s$, $D_s = \{\boldsymbol{x}_i\}_{i=1}^l$, where $1 \leq l \leq \lceil n/k \rceil$. Moreover, the iterative clustering process of RSIS results in the division of the original feature space into k feature subspaces, represented as $\mathcal{X} = \cup_{s=1}^k \mathcal{X}^s$.

During the iterative hierarchical clustering process, it is required to identify the two closest elements in C_t. This step can be optimized using the K-dimensional tree (K-d tree) method. The construction of a K-d tree typically has a time complexity of $O(n \log n)$, and the average time complexity for deleting or adding a point and finding the two nearest elements in a K-d tree is $O(\log n)$. However, as the iteration proceeds, the sample size gradually decreases, hence the actual search time will slightly decline. Excluding the LOF algorithm which may reassign redundant samples from a cluster back to the dataset, the total number of iterations is n. If the K-d tree

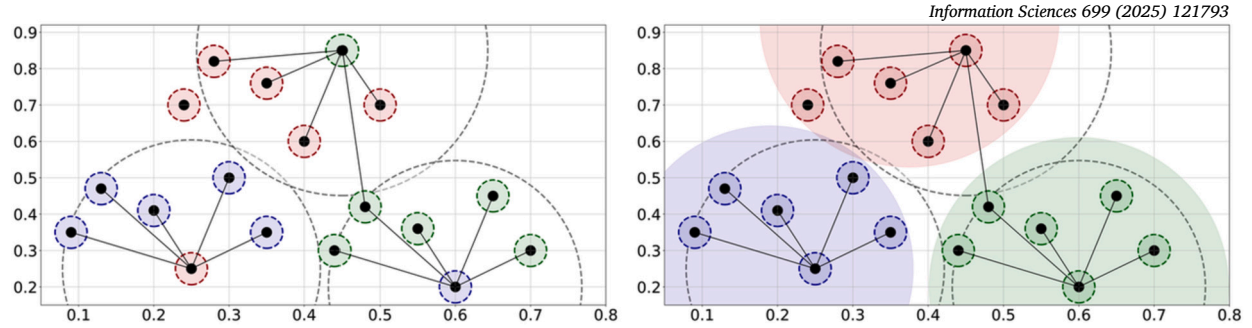

Fig. 3. K-Nearest Neighbors optimization.

is not rebuilt at every iteration, and the operations of querying and updating are efficient on an existing K-d tree, the overall time complexity might indeed be closer to $O(n \log n)$.

3.1.2. K-nearest neighbors optimization

The hierarchical clustering process, being based on the concept of a greedy strategy, may yield irrational outcomes in later iterations, which affects the overall clustering performance. Consequently, an optimization process utilizing K-Nearest Neighbors (KNN) has been developed to fine-tune the clustering results. This approach may affect the balance of the number of samples in each cluster, but it will significantly improve the performance of the clustering.

Upon completing the hierarchical clustering phase, each sample is endowed with a label reflecting its cluster membership. During the KNN optimization phase, the algorithm identifies the $\lceil n/k \rceil$ nearest neighbors for each sample in the feature space. Each sample's cluster label is then updated to reflect the most frequently occurring cluster label among its neighbors. It is important to note that this process may induce changes in the cluster affiliation of some samples, thereby causing slight fluctuations in the number of samples within clusters. Such alterations could result in the dissolution of clusters with a more dispersed sample distribution. By integrating the hierarchical clustering process with KNN optimization, we arrive at the final clustering results $\{D_s\}_{s=1}^{k'}$, where k' denotes the number of clusters post-optimization. This algorithmic approach, depicted in Fig. 3, ensures the uniform distribution of sample sizes across clusters, significantly enhancing the overall performance of clustering.

3.2. Subspace interpolation path sorting

The design philosophy of the interpolation path involves starting from a subspace, traversing each subspace exactly once, and ultimately returning to the original point. The aim is to minimize the total distance of the path between centroids of clusters (subspaces). This challenge can be formulated as TSP, which is then addressed by depicting the issue as a weighted graph. Bridging this concept to our ultimate goal, the objective function is defined as:

$$\min \sum_{i=1}^{k'-1} \mathrm{dist}(\bar{\boldsymbol{x}}^i, \bar{\boldsymbol{x}}^{i+1}) + \mathrm{dist}(\bar{\boldsymbol{x}}^{k'}, \bar{\boldsymbol{x}}^1). \tag{2}$$

The TSP is a classic NP-hard problem in combinatorial optimization, posing significant challenges in finding polynomial-time solutions as graph complexity increases. Widely applicable in logistics, planning, and manufacturing, the TSP's direct solution is complex, prompting the development of heuristic and approximation algorithms to seek near-optimal solutions efficiently [38]. To ensure that a sufficiently good solution can be obtained within an acceptable timeframe, we first employ a greedy algorithm to quickly find an initial solution, which is then optimized using the 3-opt method to arrive at the final solution [39], as is shown in Fig. 4. Ultimately, a sorted set $D_{\text{sorted}} = \{D_{(s)}\}_{s=1}^{k'}$ is obtained. Correspondingly, $\mathcal{X}_{\text{sorted}} = \cup_{s=1}^{k'} \mathcal{X}^{(s)}$. In the subsequent sections of this paper, the feature subspace corresponding to two subsets with adjacent indices will be defined as adjacent feature subspaces.

The above discussion applies to multidimensional feature spaces. However, in the case of a one-dimensional feature space, given the unique characteristics of one-dimensional spaces, the design of the interpolation path still requires passing through all subspaces exactly once, but it does not necessitate returning to the starting point. A greedy strategy can be directly employed to solve this, and in the one-dimensional case, the solution obtained by this method is guaranteed to be optimal. First, the starting subspace of the interpolation path is identified:

$$D_{(1)} = \underset{D_{(s)} \in D_{\text{sorted}}}{\operatorname{argmin}} \ \mathrm{dist}(\bar{x}^s, x_0), \tag{3}$$

where x_0 is the minimum value in feature set. The definition for subsequent clusters is as follows:

$$D_{(d)} = \underset{\substack{D_{(s)} \in D_{\text{sorted}}; \\ D_{(s)} \neq D_{(1)}, \ldots, D_{(d-1)}}}{\operatorname{argmin}} \ \mathrm{dist}\left(\bar{x}^s, \bar{x}^{s-1}\right). \tag{4}$$

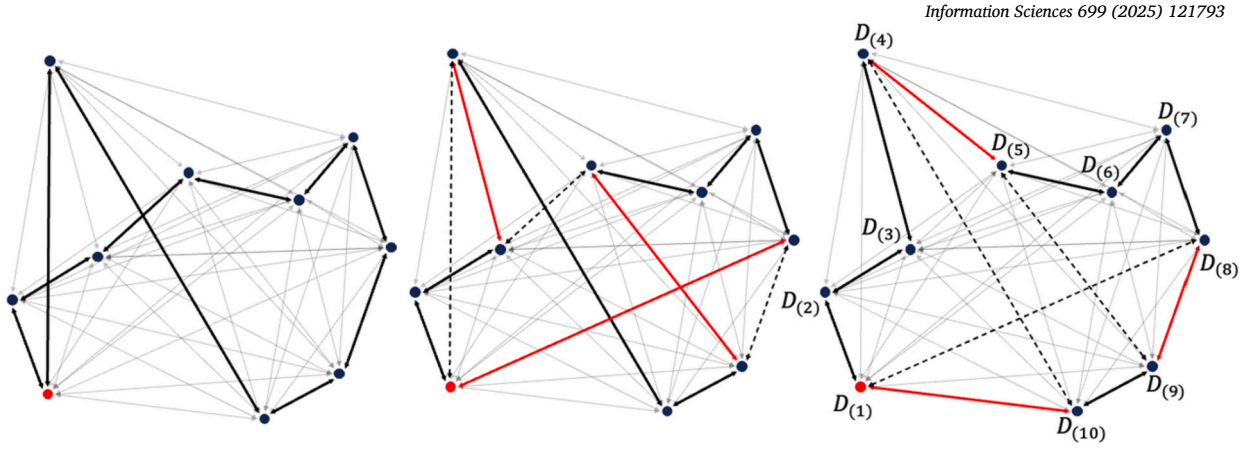

Fig. 4. Greedy algorithm and 3-opt solved for TSP in two-dimensional feature space.

Unless otherwise specified, the subsequent research in this paper will primarily focus on exploring the case of multidimensional feature spaces.

3.3. Adjacent subspace linear regression fitting

Given a dataset with noise $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$, where $\boldsymbol{x}_i \in \mathcal{X} = \mathbb{R}^d$ and $y_i \in \mathcal{Y} = \mathbb{R}$. Assume that $\dot{y}_i = f(\dot{\boldsymbol{x}}_i)$, where $\dot{\boldsymbol{x}}_i$ represents the denoised true value of $\boldsymbol{x}_i$, and $\dot{y}_i$ is the true value of y_i. The RIRS method presupposes that the function $f(\cdot)$ is continuous. And we have:

$$\dot{y}_i + \varepsilon_{i,y} = f(\dot{\boldsymbol{x}}_i + \boldsymbol{\varepsilon}_{i,x}) + \varepsilon_i, \tag{5}$$

where $\boldsymbol{\varepsilon}_{ix}$ and $\varepsilon_{i,y}$ respectively represent the noise contained in $\dot{\boldsymbol{x}}_i$ and $\dot{y}_i$, ε_i is the error term of the equation. Equation (5) can be transformed into:

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i. \tag{6}$$

Through the aforementioned clustering and subspace sorting algorithms, the dataset D can be partitioned into multiple ordered subsets $D_{\text{sorted}} = \{D_{(s)}\}_{s=1}^{k'}$, where $D_{(s)} = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{\text{num}(D_{(s)})}$, and the feature space can be divided into multiple subspaces $\mathcal{X}_{\text{sorted}} = \cup_{s=1}^{k'} \mathcal{X}^{(s)}$. For two adjacent feature subspaces, given that the RSIS method assumes the relation $f(\cdot)$ to be a continuous function, it allows $f(\cdot)$ to be approximately fitted as a linear function $g(\cdot)$, hence Equation (6) can be transformed into:

$$y_i^{s,s+1} = g_{s,s+1}(\boldsymbol{x}_i^{s,s+1}) + \varepsilon_i + \varepsilon_i', \tag{7}$$

where $g_{s,s+1}(\cdot)$ and ε_i' respectively denote the linear fitting function between adjacent feature subspaces $\mathcal{X}^{(s)}$ and $\mathcal{X}^{(s+1)}$, and the linear fitting error, $\{\boldsymbol{x}_i^{s,s+1}, y_i^{s,s+1}\} \in D_{(s)} \cup D_{(s+1)}$. If the distance between two subspaces is sufficiently close and the measure of the subspaces tends towards 0, then the linear fitting error $\varepsilon_i' \to 0$.

One may consider employing methods such as Ordinary Least Squares (OLS), Lasso regression, or locally weighted linear regression to estimate $g_{s,s+1}(\cdot)$. However, given the assumption that $f(\cdot)$ is a continuous function, merely conducting linear fitting based on samples from two subspaces does not adequately take into account global information. Therefore, we integrate the principle of soft parameter sharing mechanisms [40], estimating linear functions among multiple subspaces simultaneously, and incorporates the number of k' L_1 regularization term in the design of the loss function. This approach ensures that the estimation of each linear function $\hat{g}_{s,s+1}(\cdot)$ incorporates global information to the greatest extent possible:

$$\begin{aligned} L(\boldsymbol{\beta}) = &\sum_{s=2}^{k'-1} \left(L_2(\boldsymbol{\beta}^{s,s+1}) + \frac{\lambda}{\text{dist}(\overline{\boldsymbol{x}}^s, \overline{\boldsymbol{x}}^{s-1}) + 1} L_1(\boldsymbol{\beta}^{s,s+1}, \boldsymbol{\beta}^{s-1,s}) \right) + L_2(\boldsymbol{\beta}^{1,2}) + L_2(\boldsymbol{\beta}^{k',1}) \\ &+ \frac{\lambda}{\text{dist}(\overline{\boldsymbol{x}}^{k'}, \overline{\boldsymbol{x}}^{k'-1}) + 1} L_1(\boldsymbol{\beta}^{k',1}, \boldsymbol{\beta}^{k'-1,k'}) + \frac{\lambda}{\text{dist}(\overline{\boldsymbol{x}}^{k'}, \overline{\boldsymbol{x}}^{1}) + 1} L_1(\boldsymbol{\beta}^{1,2}, \boldsymbol{\beta}^{k',1}), \end{aligned} \tag{8}$$

where $\boldsymbol{\beta} = \left(\boldsymbol{\beta}^{1,2}, \ldots, \boldsymbol{\beta}^{k'-1,k'}, \boldsymbol{\beta}^{k',1}\right)'$, $L_2(\boldsymbol{\beta}^{p,q}) = \|\boldsymbol{y}^{p,q} - \boldsymbol{X}^{p,q}\boldsymbol{\beta}^{p,q}\|_2^2$, $L_1(\boldsymbol{\beta}^{q,r}, \boldsymbol{\beta}^{p,q}) = \|\boldsymbol{\beta}^{q,r} - \boldsymbol{\beta}^{p,q}\|_1$ and $\frac{\lambda}{\text{dist}(\overline{\boldsymbol{x}}^s, \overline{\boldsymbol{x}}^{s-1}) + 1}$ serves as the coefficient for the regularization term, adaptively adjusted based on the distance between subspaces. In the subsequent simulation experiments and case analyses, λ is set to 1.

Theorem 1, presented later, builds directly upon the foundational results established by Lemma 1 and Lemma 2. By proving the convexity of the loss function through Theorem 1, we ensure the attainment of the global minimum and facilitate the application of optimization algorithms with guaranteed convergence properties. Detailed proofs are provided in the Appendix.

Lemma 1. *For the function* $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$, *where* $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, *and* $\boldsymbol{\beta} \in \mathbb{R}^{d\times 1}$. *For* $\forall \omega \in [0,1]$, *and* $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^{d\times 1}$, *it can be obtained:*

$$f(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) \le \omega f(\boldsymbol{\beta}_1) + (1-\omega) f(\boldsymbol{\beta}_2). \tag{9}$$

Lemma 2. *For the function* $f(\boldsymbol{\beta}) = \|\boldsymbol{\beta}^1 - \boldsymbol{\beta}^2\|_1$, *where* $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2 \in \mathbb{R}^{d\times 1}$, $\boldsymbol{\beta} = ((\boldsymbol{\beta}^1)', (\boldsymbol{\beta}^2)')'$. *For* $\forall \omega \in [0,1]$, *and* $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^{2d\times 1}$, *one can infer:*

$$f(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) \le \omega f(\boldsymbol{\beta}_1) + (1-\omega) f(\boldsymbol{\beta}_2). \tag{10}$$

Theorem 1. *For the Equation (8),* $\forall \omega \in [0,1]$, *and* $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^{d\times k'}$, *it can be derived:*

$$L(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) \le \omega L(\boldsymbol{\beta}_1) + (1-\omega) L(\boldsymbol{\beta}_2). \tag{11}$$

3.4. Multi-stage minimum weight matching

In the subsequent steps, we will synthesize new samples by performing linear and equidistant interpolation between adjacent subspaces. The samples synthesized by the RSIS take into account all sample information comprehensively, and the interpolation rules are designed as follows:

- The two samples used for linear interpolation should originate from distinct and adjacent subspaces.
- Each sample must participate in the interpolation process at least once.
- The interpolation will be performed a total of $\max(\text{num}(D_{(s)}), \text{num}(D_{(s+1)}))$ times.
- The participation frequency in interpolation for all samples must be uniform. Specifically, each sample is allowed to participate in the interpolation process up to $\left\lceil \frac{\max(\text{num}(D_{(s)}), \text{num}(D_{(s+1)}))}{\min(\text{num}(D_{(s)}), \text{num}(D_{(s+1)}))} \right\rceil$ times.

Without loss of generality, we assume that $\text{num}(D_{(s)}) = m$, $\text{num}(D_{(s+1)}) = n$, and $n/2 \le m \le n$. According to the rules of interpolation, there are $C_{n-2k}^2 \prod_{k=0}^{n-m-1} C_m^{n-m}(2m-n)!$ interpolation matching strategies available for selection. Seeking a suboptimal matching strategy is a crucial problem that we need to tackle.

Similarly, without loss of generality, assuming that the feature space is one-dimensional. The average error of samples synthesized through linear and equidistant interpolation between two original samples is measured by:

$$\bar{S}(x^s, x^{s+1}) = \frac{\int_{x^s}^{x^{s+1}} |f(x) - l(x)|\, dx}{|x^{s+1} - x^s|}, \tag{12}$$

where $l(x)$ denotes the linear function passing through points (x^s, y^s) and (x^{s+1}, y^{s+1}), and $f(\cdot)$ represents the real function of x and y.

Theorem 2. *For two adjacent subspaces* $\mathcal{X}^{(s)}$ *and* $\mathcal{X}^{(s+1)}$, *consider* $(x^s, y^s) \in D_{(s)}$ *and* $(x^{s+1}, y^{s+1}) \in D_{(s+1)}$, *with* $y = f(x) + \varepsilon$. *According to (7), assuming the linear fitting error* $\varepsilon' \to 0$. *It can be obtained that:*

$$\mathbb{E}\,(\bar{S}(x^s, x^{s+1})) < \mathbb{E}\left(\frac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}\right). \tag{13}$$

From Theorem 2, under the assumption of the linear fitting error $\varepsilon_i' \to 0$, linear interpolation between samples from any two adjacent subspaces will result in a lower expected average error for the interpolated synthetic samples compared to the original samples, even in the scenario where noise with different distributions.

Theorem 3. *Extending the groundwork established in Theorem 2, let* $y^s = f(x^s) + \varepsilon^s$, $y^{s+1} = f(x^{s+1}) + \varepsilon^{s+1}$. *Assuming the linear fitting error* $\varepsilon' \to 0$. *We have:*

$$\bar{S}(x^s, x^{s+1}) = \begin{cases} \dfrac{\left|\frac{\varepsilon^s}{\varepsilon^s - \varepsilon^{s+1}}\right| \cdot |\varepsilon^s| + \left|\frac{\varepsilon^{s+1}}{\varepsilon^s - \varepsilon^{s+1}}\right| \cdot |\varepsilon^{s+1}|}{2}, & \varepsilon^s \cdot \varepsilon^{s+1} < 0, \\ \dfrac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}, & \varepsilon^s \cdot \varepsilon^{s+1} \ge 0. \end{cases} \tag{14}$$

According to Theorem 3, Formula (12) can be transformed into Formula (14). We can first estimate the error of the samples, and then determine the matching strategy based on the error information by employing the method of bipartite graph minimum weight matching. The noise of the samples can be estimated by the linear fitting function between adjacent subspaces, under the assumption conditions, as can be derived from Equation (7):

$$\varepsilon_i^{s,s+1} = y_i^{s,s+1} - g_{s,s+1}(\boldsymbol{x}_i^{s,s+1}). \tag{15}$$

According to interpolation rules, each sample is mandated to undergo at least one interpolation, and the total number of interpolations is constrained. After optimization with the KNN, the number of samples between subspaces is not always exactly equal, thus this cannot be considered a classical minimum weight perfect matching problem. This problem represents a variant of the weighted matching problem, incorporating features of both balanced matching and minimum weight matching, as it necessitates minimizing the total weight of matching while also ensuring the uniformity of sample matching occurrences. Currently, there is no standard algorithm readily available for this specific problem, necessitating modifications to the traditional minimum weight matching approach to accommodate this target scenario. Based on this, we propose a heuristic multi-stage minimum weight matching algorithm. Note that this method does not guarantee an optimal solution, but it can obtain a reasonable suboptimal solution in a short time.

Continuing with the previous assumption $\text{num}(D_{(s)}) = m$, $\text{num}(D_{(s+1)}) = n$, $n/2 \le m \le n$. In this scenario, each sample in $D_{(s)}$ is considered for either one or two matches, and the solution is approached in two stages. Given a weighted bipartite graph $G = (D_{(s)}, D_{(s+1)}, E)$ with a weight function $\omega : E \to \mathbb{R}$, where $D_{(s)}$, $D_{(s+1)}$ are the sets of vertices on the two sides and $E \subseteq D_{(s)} \times D_{(s+1)}$ is the set of edges. According to Formula (14), let the weight function $\omega(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) = \bar{S}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1})$. Consider the objective function:

$$\begin{aligned}
&\min \sum_{(i,j)\in E} \omega(\boldsymbol{x}_i^s, \boldsymbol{x}_j^{s+1}) v_{i,j}, \\
&s.t. \sum_{j=1}^{n} v_{i,j} \in \{1,2\} \text{ for } \forall\, i, \\
&\quad \sum_{i=1}^{m} v_{i,j} = 1 \text{ for } \forall\, j,
\end{aligned}$$

where $v_{i,j}$ is a binary decision variable. In the first stage, without considering any constraints, the Hungarian algorithm is applied directly to the bipartite graph G to obtain a solution, resulting in a maximum matching M_1 for G. At this point, each vertex $\boldsymbol{x}^s \in D_{(s)}$ is matched once, m vertices from $D_{(s+1)}$ are also matched once, and the remaining $n-m$ vertices in $D_{(s+1)}$ remain unmatched, which are denoted as $D'_{(s+1)}$.

In the second stage, define $G' = (D_{(s)}, D'_{(s+1)}, E')$ with ω, where $E' \subseteq D_{(s)} \times D'_{(s+1)}$. Still employing the Hungarian algorithm for G' to obtain M_2. Since $n/2 \le m \le n$, each vertex in $D'_{(s+1)}$ is involved in one matching in M_2 and $n-m$ vertices from $D_{(s)}$ participate in a matching again. The final matching is obtained $M = M_1 \cup M_2$.

The above describes a two-stage minimum weight matching method for a specific assumed situation, which is also the most common scenario in sample matching between subspaces. If there is a significant difference in the number of samples between the two subspaces, the problem needs to be divided into $\left\lceil \frac{\max(\text{num}(D_{(s)}),\text{num}(D_{(s+1)}))}{\min(\text{num}(D_{(s)}),\text{num}(D_{(s+1)}))} \right\rceil$ stages for resolution, with a more general form presented in Algorithm 2.

Algorithm 2 Multi-stage minimum weight matching.

Input: $D_{(s)} = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^m$, $D_{(s+1)} = \{\boldsymbol{x}_i^{s+1}, y_i^{s+1}\}_{i=1}^n$ $(n \ge m)$, $\hat{g}_{s,s+1}$
Output: Interpolation matching strategy M

Estimate the error of $D_{(s)}$ and $D_{(s+1)}$ by $\varepsilon_i^{s,s+1} = y_i^{s,s+1} - g_{s,s+1}(\boldsymbol{x}_i^{s,s+1})$
Define $G_1 = (D_{(s)}, D_{(s+1)}, E)$ with $\omega(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) = \bar{S}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1})$
For $t = 1, 2, \ldots, \left\lceil \frac{n}{m} \right\rceil$ do:
  Utilizing the Hungarian algorithm on bipartite graph G_t to obtain a maximum matching M_t
  $M \leftarrow M \cup M_t$
  Define $D_{\text{matched}} \subset D_{(s+1)}$ which have been matched in M_t
  $D'_{(s+1)} \leftarrow D_{(s+1)} - D_{\text{matched}}$
  Define $G_{t+1} = (D_{(s)}, D'_{(s+1)}, E')$, where $E' \subseteq D_{(s)} \times D'_{(s+1)}$

The computational complexity of our proposed multi-stage minimum weight matching algorithm is analyzed as follows. Given a bipartite graph with m vertices on one side and n vertices on the other side $(m < n)$, the algorithm performs $\lceil n/m \rceil$ stages of minimum weight matching. Each stage uses the Hungarian algorithm, which has a time complexity of $O(m^2 \cdot n)$. Therefore, the overall computational complexity of the multi-stage matching process is $O(\lceil n/m \rceil \cdot m^2 \cdot n)$, which simplifies to $O(n^2 \cdot m)$.

3.5. RSIS overall analysis

Given that the number of subspaces is k', it is necessary to perform k' times of multi-stage minimum weight matching to obtain k' matching strategies, ultimately yielding the total distance of interpolation paths between all samples, denoted as dist_{sum}. An

additional hyperparameter η must be specified, which intuitively represents the ratio of the number of synthetic samples to the original sample size. For two samples from adjacent subspaces, $\{\boldsymbol{x}^s, y^s\}$ and $\{\boldsymbol{x}^{s+1}, y^{s+1}\}$, the synthesized samples are given by $\{\boldsymbol{x}_{(d)}^{s,s+1}, y_{(d)}^{s,s+1}\}_{d=1}^{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil}$, where n is the original sample size, and the term $\frac{n\eta}{\text{dist}_{\text{sum}}}$ is intuitively the number of samples inserted per unit distance, while $\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil$ is the total number of inserted samples between two original samples. Assuming all variables are continuous, the linear interpolation formula is designed as follows:

$$\boldsymbol{x}_{(d)}^{s,s+1} = \boldsymbol{x}^s + d \frac{\boldsymbol{x}^{s+1} - \boldsymbol{x}^s}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1}, \tag{16}$$

$$y_{(d)}^{s,s+1} = y^s + d \frac{y^{s+1} - y^s}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1}. \tag{17}$$

This approach allows for the adaptive control of the quantity of generated samples based on the distance between samples in the feature space, ensuring that inserted samples are linear and equidistant. Theorems 2 and 3 have essentially proved the effectiveness of the RSIS method. To ensure thoroughness and academic rigor, we additionally present a proof in Theorem 4 adhering to the interpolation format specified in Equations (16) and (17). This method is delineated into multiple steps, with the overall operational procedure presented in Algorithm 3.

Algorithm 3 RSIS method.

Input: Dataset $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, Hyperparameters k, η
Output: Synthesized dataset D'

Construct feature dataset: $C = \{\boldsymbol{x}_i\}_{i=1}^n$
Apply the hierarchical clustering algorithm to samples to obtain clusters $D = \{D_s\}_{s=1}^k$
After KNN optimization, obtain: $D = \{D_s\}_{s=1}^{k'}$
Utilize greedy algorithm and 3-opt to obtain the sorted solution: $D_{\text{sorted}} = \{D_{(s)}\}_{s=1}^{k'}$
Perform linear fitting between adjacent subspaces to derive $\{\hat{g}_{s,s+1}\}_{s=1}^{k'-1} \cup \{\hat{g}_{k',1}\}$
For $s = 1, 2, \ldots, k'$ do:
  $D_{(s+1)} = D_{(1)}$ if $s = k'$
  Estimate the error in adjacent subspaces samples from $D_{(s)}, D_{(s+1)}$ by (15)
  Employ the multi-stage minimum weight matching method to determine the interpolation matching strategy
  According to the matching strategy, interpolate between adjacent subspaces samples to synthesize new data
  Add the newly synthesized data to D'

Theorem 4. *For two adjacent subspaces $\mathcal{X}^{(s)}$ and $\mathcal{X}^{(s+1)}$, consider $(\boldsymbol{x}^s, y^s) \in D_{(s)}$ and $(\boldsymbol{x}^{s+1}, y^{s+1}) \in D_{(s+1)}$, with $y = f(\boldsymbol{x}) + \varepsilon$. $\{\boldsymbol{x}_{(d)}^{s,s+1}, y_{(d)}^{s,s+1}\}_{d=1}^{\left\lceil \frac{n\eta}{dist_{sum}} dist(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil}$ are the synthesized samples according to (16) and (17) with $\varepsilon_{(d)}^{s,s+1} = y_{(d)}^{s,s+1} - f(\boldsymbol{x}_{(d)}^{s,s+1})$. Assuming the linear fitting error $\varepsilon' \to 0$, it can be obtained:*

$$\mathbb{E}\left(\frac{\sum_{d=1}^{\left\lceil \frac{n\eta}{dist_{sum}} dist(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} |\varepsilon_{(d)}^{s,s+1}|}{\left\lceil \frac{n\eta}{dist_{sum}} dist(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil}\right) < \mathbb{E}\left(\frac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}\right).$$

According to Theorems 2, 3, and 4, the RSIS method can synthesize samples with less noise compared to the original samples. From Equations (5) and (6), it follows that the samples synthesized by RSIS will have smaller ε_i thereby highlighting the true functional relationships between variables. Moreover, RSIS performs linear interpolation between samples, which does not compromise the information of the original samples. By synthesizing a large number of samples with smaller errors, it extracts more information from the original data. Consequently, RSIS can increase data diversity and representativeness. In summary, the RSIS method is based on three assumptions:

- The function $f(\cdot)$ is continuous.
- The linear fitting error $\varepsilon' \to 0$.
- The features and label variables are continuous.

4. Experiments

For the simulated datasets, which are generated by a known function $f(\cdot)$, we thus use them to test the optimization performance of RSIS; for benchmark datasets, our research focuses on the improvement in model predictive capability after optimization by RSIS.

As a novel data synthesis method, RSIS lacks comparable existing counterparts, primarily because most methods are not tailored to tackle issues related to sample noise and to strengthen the relationships between variables. Consequently, comparative analysis between RSIS and these methods is not presented in this section.

Table 1
Definitions of evaluation metrics.

Metric	Expression
MAE (Mean absolute error)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i-\hat{y}_i\rvert$
WIA (Willmott's index of agreement)	$1-\frac{\sum_{i=1}^{n}(\hat{y}_i-y_i)^2}{\sum_{i=1}^{n}(\lvert\hat{y}_i-\bar{y}\rvert+\lvert y_i-\bar{y}\rvert)^2}$
MAPE (Mean absolute percent error)	$\frac{1}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i-\hat{y}_i}{y_i+0.01}\right\rvert$

Table 2
Simulation datasets.

Simulation Datasets	$\varepsilon_{i,x}$, $\varepsilon_{i,y}$ Distribution	Samples Size	Feature Dimension	Feature Distribution
D_1	$20\%\sim N(0,64)$ $30\%\sim U(-8,8)$ $50\%\sim N(0,1)$	500	10	$N(0,10)$
D_2	$20\%\sim N(0,64)$ $30\%\sim U(-8,8)$ $50\%\sim N(0,1)$	500	10	$T(0,10)$
D_3	$20\%\sim N(0,64)$ $30\%\sim U(-8,8)$ $50\%\sim N(0,1)$	1500	10	$N(0,10)$
D_4	$20\%\sim N(0,64)$ $30\%\sim U(-8,8)$ $50\%\sim N(0,1)$	500	30	$N(0,10)$
D_5	$50\%\sim N(0,64)$ $30\%\sim U(-8,8)$ $20\%\sim N(0,1)$	500	10	$N(0,10)$
D_6	$20\%\sim N(0,64)$ $30\%\sim U(-8,8)$ $50\%\sim N(0,1)$	200	10	$N(0,10)$

4.1. Simulation datasets

4.1.1. Experimental settings

Multiple error criteria have been used to assess optimization performance. The performance metrics are summarized in Table 1, where y_i denotes the real value of original or synthesized samples, while $\hat{y}_i$ represents the observed value from the original dataset or the value of synthesized data obtained through Equation (17).

Due to the RSIS method assumes that all variables and their functional relationships are continuous, we generate relationship matrices $\boldsymbol{M}_1 \in \mathbb{R}^{d\times d'}$, and $\boldsymbol{M}_2 \in \mathbb{R}^{d'\times 1}$, and the model utilized is $y_i = f(\boldsymbol{x}_i) = \tanh(\boldsymbol{x}_i\boldsymbol{M}_1)\boldsymbol{M}_2$.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{18}$$

According to Formula (5), add noise to all variables in the dataset. It is often difficult to determine the distribution of noise, and in most cases, the noise is not identically distributed. We design three different distributions of noise, added to the simulated dataset in certain proportions, to mimic the actual noise conditions of real datasets [41]. We designed six different simulated datasets, and each experiment was repeated 25 times with the average value taken as the result. Detailed information on the simulated datasets is shown in Table 2.

4.1.2. Analysis of optimization effects

Table 3 presents the results for the MAE (first row), WIA (second row), and MAPE (third row) for each simulated experiment's original datasets and for the datasets processed using the RSIS method under varying hyperparameters. To determine whether the datasets were optimized, the Wilcoxon Signed-Rank Test was conducted with a significance level of 0.05 [42]. Signs '+', '−' and '≈' respectively indicate a significant improvement, a significant decline, or no significant change in the metrics after processing with the RSIS method. The last three rows of the table detail the outcomes of the sign rank test, with each tuple representing the count of significant results for the respective metric.

Table 3
Coputational results of optimization effects.

Datasets	Before RSIS	After RSIS								
		k:	5	10	15	5	10	15	5	10	15
		η:	1	1	1	5	5	5	10	10	10
	17.79		13.82+	14.47+	15.48+	13.14+	14.42+	15.75+	13.01+	14.48+	15.89+
D_1	0.290		0.319+	0.315+	0.313+	0.341+	0.320+	0.315+	0.344+	0.319+	0.314+
	5.160		3.908+	3.947+	4.210+	3.919+	3.967+	4.207+	3.974+	4.317+	4.191+
	19.97		16.01+	16.34+	16.97+	15.11+	16.06+	16.71+	14.95+	16.05+	16.78+
D_2	0.241		0.249+	0.259+	0.259+	0.259+	0.271+	0.265+	0.260+	0.273+	0.265+
	6.179		4.484+	4.793+	5.095+	4.671+	4.813+	4.893+	4.603+	4.925+	4.913+
	18.37		13.77+	14.35+	14.95+	13.21+	14.34+	14.89+	13.12+	14.36+	14.99+
D_3	0.313		0.333+	0.339+	0.338+	0.345+	0.345+	0.339+	0.345+	0.344+	0.336+
	5.143		3.446+	3.466+	4.318+	3.187+	3.284+	3.880+	3.185+	3.225+	6.409+
	19.29		19.29≈	15.71+	15.07+	19.29≈	14.26+	14.15+	19.29≈	13.95+	14.01+
D_4	0.280		0.280≈	0.302+	0.312+	0.280≈	0.325+	0.340+	0.280≈	0.331+	0.345+
	6.643		6.643≈	4.973+	4.578+	6.643≈	4.303+	4.097+	6.643≈	4.112+	3.991+
	31.88		22.15+	22.34+	23.00+	21.17+	21.94+	22.88+	21.01+	21.90+	22.87+
D_5	0.190		0.211+	0.216+	0.220+	0.237+	0.239+	0.236+	0.240+	0.239+	0.237+
	9.508		6.221+	6.557+	6.607+	6.149+	6.641+	6.679+	6.102+	6.769+	6.714+
	18.64		15.12+	16.09+	16.89+	14.61+	14.04+	16.91+	14.53+	16.05+	16.99+
D_6	0.307		0.328+	0.329+	0.330+	0.333+	0.332+	0.332+	0.334+	0.332+	0.331+
	3.958		2.943+	3.347+	3.505+	2.841+	3.271+	3.596+	2.795+	3.302+	3.577+
+			(5,5,5)	(6,6,6)	(6,6,6)	(5,5,5)	(6,6,6)	(6,6,6)	(5,5,5)	(6,6,6)	(6,6,6)
−			(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
≈			(1,1,1)	(0,0,0)	(0,0,0)	(1,1,1)	(0,0,0)	(0,0,0)	(1,1,1)	(0,0,0)	(0,0,0)

Based on the results of the Wilcoxon Signed-Rank Test, it is evident that the RSIS method has improved performance across nearly all metrics for given datasets, with no instances of significant decline observed. For dataset D_4 under the conditions of $k = 5, \eta = 1, 5, 10$, no significant changes were noted. This is primarily attributed to the fact that as feature dimensionality increases, the samples of most clusters become too sparsely distributed in the feature space. During the KNN optimization phase, all samples were merged into a single cluster, omitting the subsequent phase of RSIS. However, this issue was mitigated with an increase in k.

Compared to η, the optimization effect of the RSIS method is more sensitive to the hyperparameter k, which will be further substantiated in the subsequent section. With an appropriately selected value for k, elevating the value of η may amplify the method's optimization efficacy.

Experimental results also show that RSIS's performance continues to optimize effectively, as the content of noise with large variances increases. Hence, it can deal with more complex noise and has good robustness. Moreover, the RSIS method is capable of significantly optimizing samples even in datasets with heavy-tailed distributions or smaller sample sizes, demonstrating its excellent adaptability.

4.1.3. Hyperparameter analysis

To more clearly demonstrate the optimization effect of RSIS and the impact of hyperparameter value, we subtracted the pre-optimization metrics from the post-optimization metrics and calculated the performance of the simulated datasets under various hyperparameters, as shown in Fig. 5. (a), (b) and (c) present the performance of different metrics at various k values when $\eta = 1$, while (d), (e) and (f) in Fig. 5 display the performance across different η values at the k value where MAE is minimized.

In the vast majority of cases, all metrics have been optimized. Compared to MAE and WIA, MAPE exhibits greater fluctuations. The primary reason is the RSIS method synthesizing a substantial number of values close to zero. Given that MAPE is particularly sensitive to near-zero true values, especially for the simulated dataset D_3, this leads to larger variations in MAPE.

The performance of each metric under different hyperparameter η values is more stable compared to k, and in most cases, as the value of η increases, the optimization effect is further enhanced, subsequently leading to a gradual stabilization. Under different values of k, the optimization effect of MAE presents a U-shape; MAPE is optimized in most scenarios; and WIA initially increases and then fluctuates slightly within a stable range.

(g) and (h) in Fig. 5 represent the proportion of errors of different magnitudes to the total sample size before and after optimization by the RSIS method, with the left part of the graph illustrating the proportion of negative errors and the right part showing the proportion of positive errors. It is evident that the proportion of samples with larger errors has significantly decreased following optimization by the RSIS method, while the proportion of samples with smaller errors has significantly increased.

In summary, our experimental results further validate that the RSIS method can effectively increase data diversity and representativeness, and highlight the true functional relationships between variables by synthesizing a large number of samples with smaller errors.

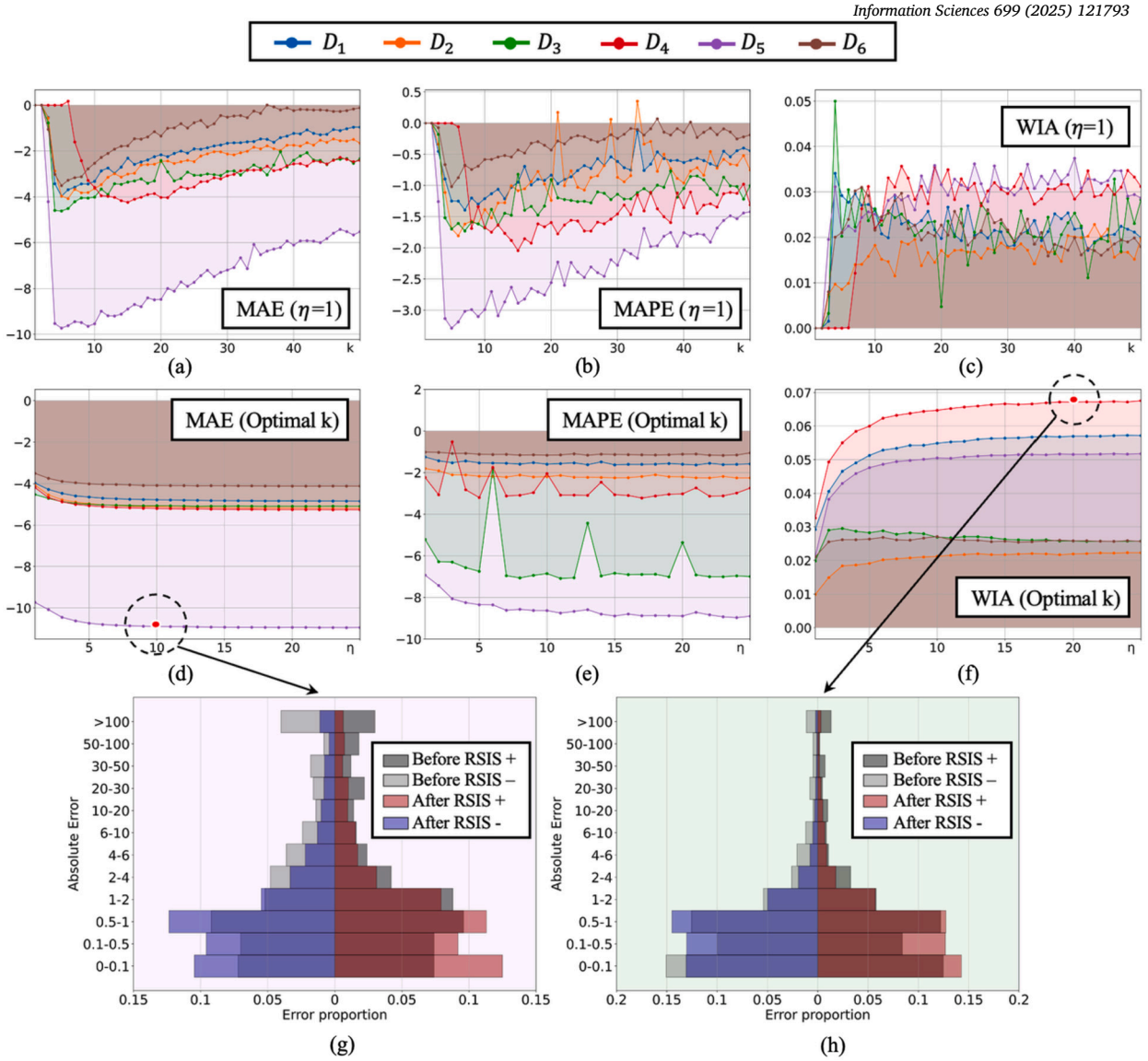

Fig. 5. Optimization effect of the RSIS method with different hyperparameters.

4.2. Benchmark datasets

To assess the optimization performance in prediction of the RSIS method, four benchmark datasets were utilized [43–46]. Samples were randomly selected in varying quantities from each dataset and divided into training and testing sets. Following processing of the training data with the RSIS approach, a suite of machine learning models was subsequently trained on this dataset, which included the original training data as well as the data synthesized by RSIS. The models' predictive performance were subsequently assessed on the testing set. For each set of experiments, we repeated the process 25 times and took the average value as the experimental result, and a Wilcoxon signed-rank test was conducted thereafter. In addition, we removed features that cannot be directly used, such as "Datetime" for Bike Sharing Demand, "Month" for Forest Fires, and so on. We selected a range of machine learning models for prediction, including KNN, Gradient Boosting Decision Tree (GBDT), Multilayer Perceptron (MLP), and Support Vector Regression (SVR), with SVR employing a Radial Basis Function (RBF) kernel.

The computational results for the four models applied to the benchmark datasets are presented in Table 4, including the MAE and MAPE (in brackets) for each testing set. RSIS consistently delivers strong performance across various benchmarks and demonstrates exceptional adaptability with different models, generally improving their prediction accuracy. It can also be seen from Table 4 that the effects of normalization and the varied sizes of training sets do not impact the optimization efficacy of RSIS. Fig. 6 demonstrates that after processing with the RSIS method, the prediction outcomes of each model are closer to the actual values, and even poorly performing prediction models (such as SVR) still achieve a significant improvement in predictive performance.

Table 4
Experimental results for benchmark datasets.

Datasets	Optimizing	Training (Testing) set size	Models KNN	MLP	SVR	GBDT
Bike Sharing Demand	-	500 (1000)	5.99 (0.16)	2.31 (0.14)	64.33 (2.34)	5.25 (0.05)
		1000 (1000)	4.44 (0.12)	1.65 (0.12)	42.98 (1.44)	4.45 (0.04)
		2000 (1000)	3.66 (0.11)	0.47 (0.02)	26.85 (0.99)	3.37 (0.04)
	RSIS	500 (1000)	3.66+ (0.14+)	0.29+ (0.02+)	12.27+ (1.44+)	2.92+ (0.05≈)
		1000 (1000)	2.94+ (0.12≈)	0.18+ (0.01+)	8.32+ (0.83+)	2.94+ (0.05≈)
		2000 (1000)	2.66+ (0.09+)	0.15+ (0.006+)	4.74+ (0.34+)	2.45+ (0.04≈)
Air Quality	-	500 (1000)	105.38 (0.12)	104.23 (0.12)	327.52 (0.52)	95.38 (0.11)
		1000 (1000)	101.86 (0.11)	98.97 (0.11)	306.72 (0.49)	89.29 (0.10)
		2000 (1000)	95.76 (0.11)	92.80 (0.10)	268 (0.44)	89.24 (0.10)
	RSIS	500 (1000)	101.43+ (0.11+)	89.85+ (0.10+)	218.93+ (0.31+)	89.92+ (0.10+)
		1000 (1000)	96.52+ (0.11≈)	88.77+ (0.10+)	174.02+ (0.30+)	87.52+ (0.10≈)
		2000 (1000)	90.24+ (0.10+)	86.18+ (0.10≈)	144.60+ (0.20+)	84.18+ (0.09+)
Facebook Metrics (Normalized)	-	100 (200)	0.079 (0.189)	0.072 (0.162)	0.101 (0.187)	0.004 (0.013)
		200 (200)	0.065 (0.152)	0.070 (0.112)	0.084 (0.148)	0.002 (0.006)
		300 (200)	0.051 (0.124)	0.055 (0.098)	0.077 (0.141)	0.001 (0.004)
	RSIS	100 (200)	0.078≈ (0.181+)	0.061+ (0.102+)	0.105− (0.198−)	0.002+ (0.007+)
		200 (200)	0.067− (0.151≈)	0.048+ (0.093+)	0.086− (0.153−)	0.001+ (0.004+)
		300 (200)	0.056− (0.125≈)	0.030+ (0.053+)	0.076≈ (0.136+)	0.001≈ (0.003≈)
Forest Fires (Normalized)	-	50 (200)	0.064 (0.540)	0.177 (0.169)	0.093 (0.083)	0.063 (0.053)
		150 (200)	0.060 (0.055)	0.161 (0.158)	0.080 (0.076)	0.058 (0.054)
		300 (200)	0.050 (0.039)	0.093 (0.081)	0.076 (0.067)	0.055 (0.044)
	RSIS	50 (200)	0.062+ (0.052+)	0.144+ (0.104+)	0.124− (0.114−)	0.077− (0.066−)
		150 (200)	0.059≈ (0.054≈)	0.058+ (0.054+)	0.082≈ (0.078≈)	0.056+ (0.052+)
		300 (200)	0.048+ (0.037+)	0.068+ (0.057+)	0.073+ (0.063+)	0.054≈ (0.043≈)
	+		8 (7)	12 (11)	7 (8)	9 (5)
	−		2 (0)	0 (0)	3 (3)	1 (1)
	≈		2 (5)	0 (1)	2 (1)	2 (6)

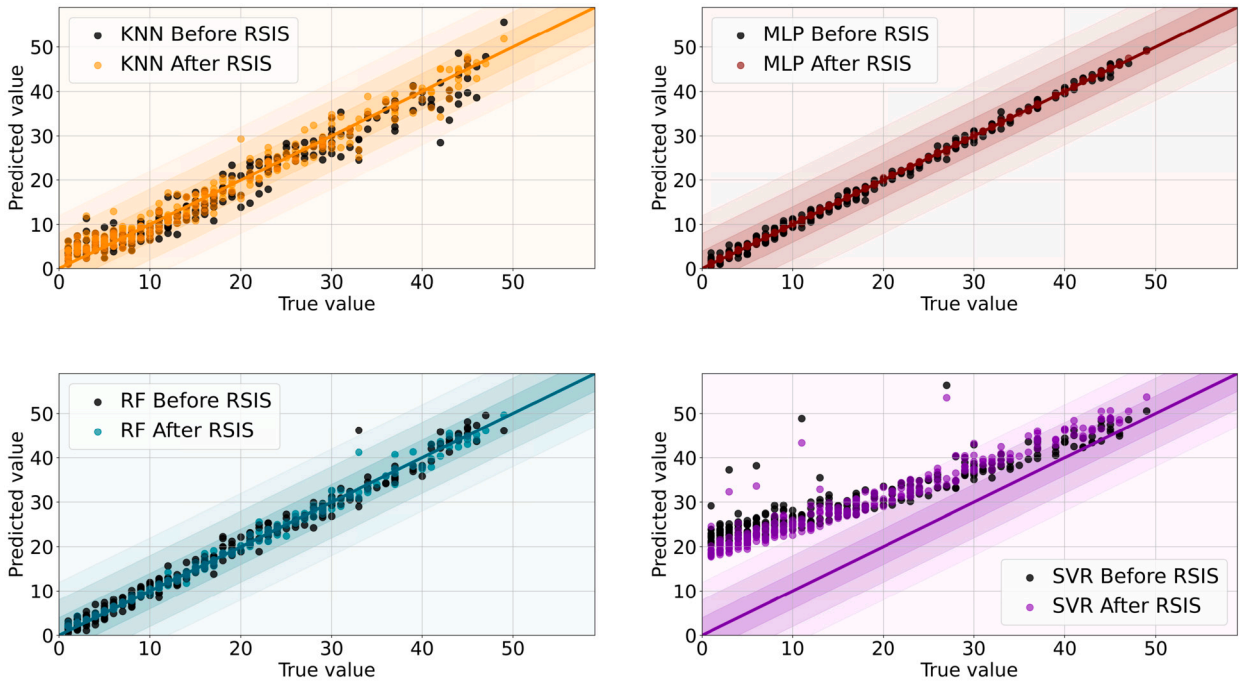

Fig. 6. Comparison of prediction effects on the partly testing set of Bike Sharing Demand.

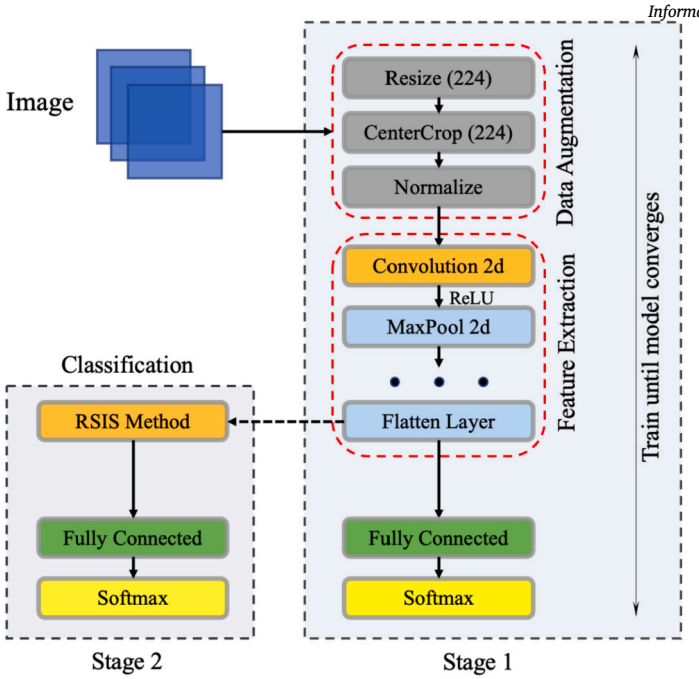

Fig. 7. RSIS method combined with CNN framework.

It is worth mentioning that the dataset contains many variables which are not continuous. Moreover, for sparse datasets (Facebook Metrics and Forest Fires), it is difficult to guarantee that the linear fitting error $\varepsilon' \to 0$ when interpolating between subspaces. This indicates that even if there are violations of the RSIS assumptions in practical applications, this method may still achieve good optimization results.

5. Further research on CNN integration

The preceding research primarily focused on synthesizing data using RSIS for feature vector data. This chapter delves into the application of the RSIS method in combination with the CNN framework in the field of image prediction.

5.1. Method overview

Consider a dataset of RGB images $D = \{\boldsymbol{I}_i, \boldsymbol{L}_i\}_{i=1}^{n}$, where $\boldsymbol{L}_i$ is the one-hot vector representing the category of $\boldsymbol{I}_i$. Let the feature extraction function of a CNN be $f : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{d}$, where $H \times W \times C$ denotes the dimensions and channel number of the image, and d represents the dimension of the extracted features. Through the feature extraction operation of CNN, an image $\boldsymbol{I}_i$ is transformed into a feature vector $\boldsymbol{v}_i = f(\boldsymbol{I}_i)$. Using the RSIS method, we obtain $\{\boldsymbol{v}_i, \boldsymbol{L}_i\}_{i=1}^{n'}$, where n' is the sample size after adding synthesized feature vectors. Finally, all samples are placed into a fully connected network for iterative training.

The combination of the RSIS method with the CNN framework necessitates changes to traditional model training approaches. Consequently, we have designed a two-stage training strategy, as illustrated in Fig. 7.

In the first stage, following traditional training protocols, the model performs forward propagation and calculates loss using input images. It then uses backpropagation to calculate gradients for updating model parameters, continuing this process until convergence.

In the second stage, original images are processed by the feature extractor, trained in the initial stage, to generate a dataset of feature vectors for each image. Subsequently, the RSIS method is employed on this dataset to synthesize new data, which is then exclusively retrained in the fully connected layer until convergence is reached. Notably, this stage updates only the fully connected layer's parameters, leaving the feature extraction layers unchanged during the training process.

5.2. Analysis of optimization effects

To assess the optimization performance in prediction, four image datasets (MNIST [47], FashionMNIST [48], CIFAR-10 [49], and SVHN [50]) and CNN frameworks were utilized, including ResNet50, VGG16, DenseNet121, and MobileNet-v2. All models were initialized with pre-trained parameters. We randomly selected 200 and 1,000 samples as the training and testing sets, respectively. In the second stage, new synthesized samples were added to the training set, expanding the original dataset to 500, 1,000, and 3,000 samples to explore the optimization effects at different scales. The process was repeated 25 times, with the average value taken as the experimental result, followed by a Wilcoxon signed-rank test. The optimization effect of the RSIS method was measured by the variation in the test error rate E_{var}:

Table 5
Experimenal results for image datasets.

Models	After RSIS	Datesets			
		MNIST	FashionMNIST	CIFAR-10	SVHN
ResNet50	500	1.70%+	0.30%+	0.50%+	3.10%+
	1000	1.90%+	1.30%+	1.60%+	3.60%+
	3000	2.70%+	1.50%+	2.10%+	3.60%+
VGG16	500	0.20%+	0.10%−	0.30%−	0.80%−
	1000	0.50%+	0.90%+	0.40%+	0.00%≈
	3000	0.70%+	1.10%+	0.40%+	0.00%≈
DenseNet121	500	0.70%+	0.20%+	0.00%≈	4.70%+
	1000	1.00%+	0.30%+	0.50%+	5.10%+
	3000	1.70%+	0.60%+	1.00%+	5.40%+
MobileNet-v2	500	1.30%+	0.70%−	0.20%+	3.30%+
	1000	1.50%+	0.40%+	1.80%+	5.20%+
	3000	1.70%+	1.20%+	2.50%+	5.70%+
+		12	10	10	9
−		0	2	1	1
≈		0	0	1	2

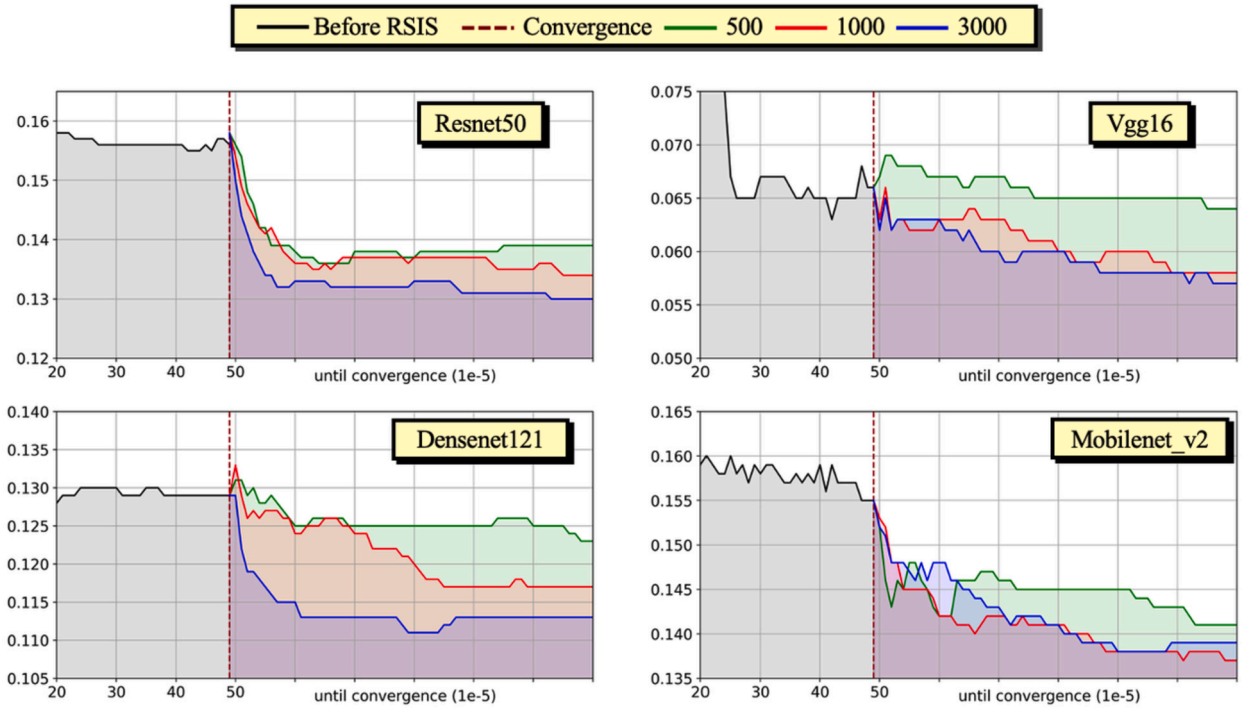

Fig. 8. Iterative comparison of prediction errors.

$$E = \frac{1}{n}\sum_{i=1}^{n} I(\hat{L}_i \neq L_i), \tag{19}$$

$$E_{\text{var}} = E_{\text{stage1}} - E_{\text{stage2}}, \tag{20}$$

where E_{stage1} represents the error rate on the testing set at the end of the first stage, and E_{stage2} is the error rate post the second stage. It can be seen from Table 5 that the RSIS significantly enhances the model's generalization capability in most cases. However, the benefits derived from the RSIS method vary according to the model's adaptability and compatibility with specific datasets. For instance, DenseNet121 demonstrates significant generalization improvement and a pronounced reduction in prediction errors on the SVHN dataset.

We selected the MNIST dataset, and examined the iterative changes in E_{stage2}, as illustrated in Fig. 8. The section right of the red dashed line illustrates the subsequent iterative changes in E_{stage2} for three distinct synthesized sample sizes, following optimization with RSIS. It is evident that in most scenarios, the error significantly decreases in the second stage.

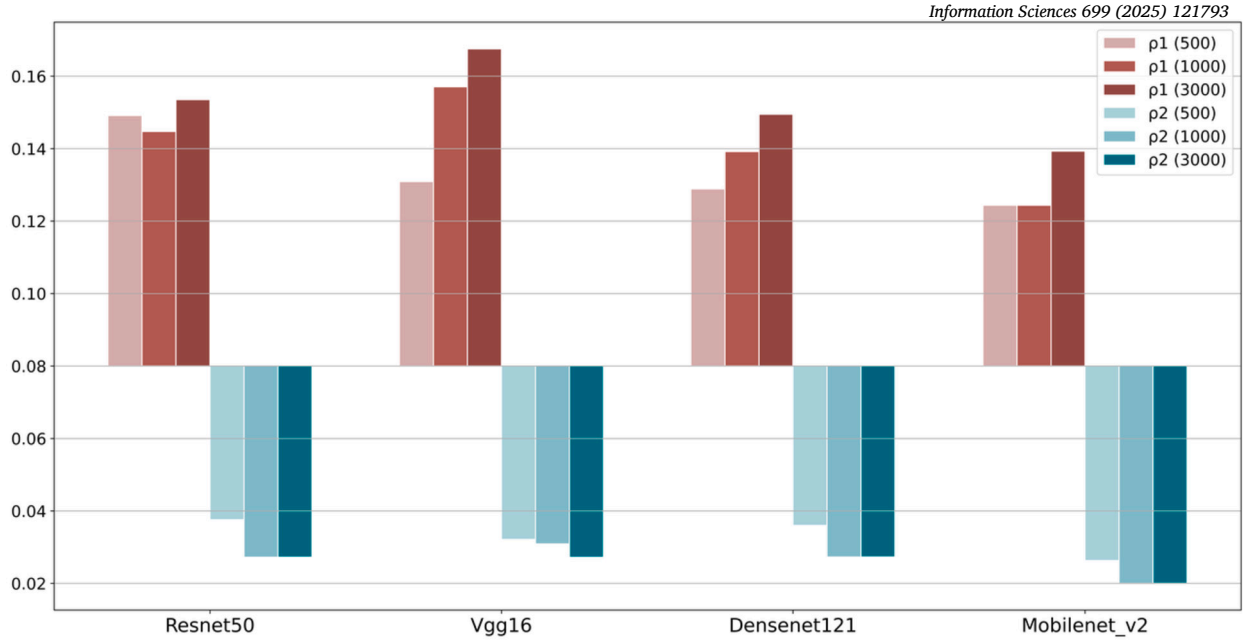

Fig. 9. The predictive performance of four models across metrics ρ_1 and ρ_2.

Fig. 9 presents the predictive performance of four models across metrics ρ_1 (Misclassification Improvement Ratio) and ρ_2 (Negative Conversion Rate) on the FashionMNIST dataset.

$$\rho_1 = \frac{\text{num}(C_{\text{stage2}} \cap M_{\text{stage1}})}{\text{num}(M_{\text{stage1}})}, \tag{21}$$

$$\rho_2 = \frac{\text{num}(M_{\text{stage2}} \cap C_{\text{stage1}})}{\text{num}(C_{\text{stage1}})}, \tag{22}$$

where M_{stage1} and M_{stage2} denote the sample misclassified sets before and after optimization, while C_{stage1} and C_{stage2} denote the sample correctly sets classified before and after optimization, respectively. In the majority of scenarios, ρ_1 tends to increase with the growth of synthesized samples in the training set, a trend that is particularly pronounced in VGG16 and DenseNet121. Conversely, ρ_2 generally decreases as the training set expands, with this effect being especially notable in VGG16.

6. Conclusion

This paper introduces a data synthesis method based on linear and equidistant interpolation within feature subspaces, which synthesizes samples with reduced noise without sacrificing the original sample information and adaptively interpolates between feature subspaces to enhance data diversity and representativeness. Furthermore, our experimental results reveal that the RSIS method significantly optimizes samples by synthesizing samples with reduced noise and simultaneously decreasing the proportion of high-noise samples. On benchmark datasets, it notably enhances the models' generalization capabilities. By innovatively integrating the RSIS method with the convolutional neural network framework, we have further improved the model's predictive performance.

RSIS can also be combined with methods from various fields, such as evolutionary algorithms, serving as a mutation strategy for differential evolution algorithms. However, RSIS is not suitable for studies on sample distribution because it alters the distribution of variables. Theoretically, if the hyperparameter η is set to infinity, variables will tend to a uniform distribution. Exploring how to significantly optimize samples without disrupting their distribution is worth pursuing. Moreover, the requirement of RSIS to flatten image data, i.e., matrix and tensor data, limits its ability to preserve positional information. Future research should address challenges associated with various data types, including heterogeneous and functional data, to enhance diversity while maintaining intrinsic distribution characteristics.

CRediT authorship contribution statement

Yukun Du: Writing – original draft, Software, Methodology. **Yitao Cai:** Writing – original draft, Software, Formal analysis. **Xiao Jin:** Writing – review & editing, Writing – original draft, Data curation. **Haiyue Yu:** Writing – review & editing, Funding acquisition, Formal analysis, Data curation. **Zhilong Lou:** Writing – review & editing, Supervision. **Yao Li:** Supervision, Data curation. **Jiang Jiang:** Visualization, Supervision. **Yongxiong Wang:** Validation, Supervision.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grants 72301286 and 72431011.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yongxiong Wang reports financial support was provided by This research was supported by a grant from Natural Science Foundation Shanghai under Grand 22ZR1443700. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of Lemma 1

Consider the function $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$, where $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2$ denotes the L_2 norm of $\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}$. Then, for $\forall \omega \in [0,1]$, and $\forall \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^{d\times 1}$, we have:

$$
\begin{aligned}
&f(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta_2}) - \omega f(\boldsymbol{\beta_1}) - (1-\omega) f(\boldsymbol{\beta_2}) \\
&= ||y - X(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2)||_2^2 - \omega||y - X\boldsymbol{\beta}_1||_2^2 - (1-\omega)||y - X\boldsymbol{\beta}_2||_2^2 \\
&= ||\omega y + (1-\omega)y - X(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2)||_2^2 - \omega||y - X\boldsymbol{\beta}_1||_2^2 - (1-\omega)||y - X\boldsymbol{\beta}_2||_2^2 \\
&= ||\omega(y - X\boldsymbol{\beta}_1) + (1-\omega)(y - X\boldsymbol{\beta}_2)||_2^2 - \omega||y - X\boldsymbol{\beta}_1||_2^2 - (1-\omega)||y - X\boldsymbol{\beta}_2||_2^2 \\
&= \omega^2||y - X\boldsymbol{\beta}_1||_2^2 + (1-\omega)^2||y - X\boldsymbol{\beta}_2||_2^2 + 2\omega(1-\omega)(y - X\boldsymbol{\beta}_1)'(y - X\boldsymbol{\beta}_2) - \omega||y - X\boldsymbol{\beta}_1||_2^2 - (1-\omega)||y - X\boldsymbol{\beta}_2||_2^2 \\
&= \omega^2||y - X\boldsymbol{\beta}_1||_2^2 + (1 - 2\omega + \omega^2)||y - X\boldsymbol{\beta}_2||_2^2 + (2\omega - 2\omega^2)(y - X\boldsymbol{\beta}_1)'(y - X\boldsymbol{\beta}_2) - \omega||y - X\boldsymbol{\beta}_1||_2^2 - (1-\omega)||y - X\boldsymbol{\beta}_2||_2^2 \\
&= \omega^2\left(||y - X\boldsymbol{\beta}_1||_2^2 - 2(y - X\boldsymbol{\beta}_1)'(y - X\boldsymbol{\beta}_2) + ||y - X\boldsymbol{\beta}_2||_2^2\right) + (1-2\omega)||y - X\boldsymbol{\beta}_2||_2^2 + 2\omega(y - X\boldsymbol{\beta}_1)'(y - X\boldsymbol{\beta}_2) - \\
&\omega||y - X\boldsymbol{\beta}_1||_2^2 - (1-\omega)||y - X\boldsymbol{\beta}_2||_2^2 \\
&= \omega^2||(y - X\boldsymbol{\beta}_1) - (y - X\boldsymbol{\beta}_2)||_2^2 + 2\omega(y - X\boldsymbol{\beta}_1)'(y - X\boldsymbol{\beta}_2) - \omega||y - X\boldsymbol{\beta}_1||_2^2 - \omega||y - X\boldsymbol{\beta}_2||_2^2 \\
&= \omega^2||(y - X\boldsymbol{\beta}_1) - (y - X\boldsymbol{\beta}_2)||_2^2 - \omega||(y - X\boldsymbol{\beta}_1) - (y - X\boldsymbol{\beta}_2)||_2^2 \\
&= \omega(\omega - 1)||(y - X\boldsymbol{\beta}_1) - (y - X\boldsymbol{\beta}_2)||_2^2 \le 0.
\end{aligned}
$$

From inequality $f(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) - \omega f(\boldsymbol{\beta}_1) - (1-\omega) f(\boldsymbol{\beta}_2) \le 0$, simplification yields the convexity condition:

$$f(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) \le \omega f(\boldsymbol{\beta}_1) + (1-\omega) f(\boldsymbol{\beta}_2).$$

Appendix B. Proof of Lemma 2

Considering the linear structure of $f(\boldsymbol{\beta})$ with respect to the L_1 norm, we commence the proof by examining the function at a convex combination of $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, and we have:

$$
\begin{aligned}
f(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) &= ||(\omega\boldsymbol{\beta}_1^1 + (1-\omega)\boldsymbol{\beta}_2^1) - (\omega\boldsymbol{\beta}_1^2 + (1-\omega)\boldsymbol{\beta}_2^2)||_1 \\
&= ||\omega(\boldsymbol{\beta}_1^1 - \boldsymbol{\beta}_1^2) + (1-\omega)(\boldsymbol{\beta}_2^1 - \boldsymbol{\beta}_2^2)||_1 \\
&\le \omega||\boldsymbol{\beta}_1^1 - \boldsymbol{\beta}_1^2||_1 + (1-\omega)||\boldsymbol{\beta}_2^1 - \boldsymbol{\beta}_2^2||_1 \\
&= \omega f(\boldsymbol{\beta}_1) + (1-\omega) f(\boldsymbol{\beta}_2).
\end{aligned}
$$

Appendix C. Proof of Theorem 1

Let $\lambda_{p,q} = \dfrac{\lambda}{\mathrm{dist}(\overline{\boldsymbol{x}}^p, \overline{\boldsymbol{x}}^q) + 1}$, $L(\boldsymbol{\beta}) = \gamma_1(\boldsymbol{\beta}) + \gamma_2(\boldsymbol{\beta})$, where

$$
\begin{aligned}
\gamma_1(\boldsymbol{\beta}) &= \sum_{s=2}^{k'-1} \lambda_{s,s-1} L_1(\boldsymbol{\beta}^{s,s+1}, \boldsymbol{\beta}^{s-1,s}) + \lambda_{k',1} L_1(\boldsymbol{\beta}^{1,2}, \boldsymbol{\beta}^{k',1}) + \lambda_{k'-1,k'} L_1(\boldsymbol{\beta}^{k',1}, \boldsymbol{\beta}^{k'-1,k'}), \\
\gamma_2(\boldsymbol{\beta}) &= \sum_{s=1}^{k'-1} L_2(\boldsymbol{\beta}^{s,s+1}) + L_2(\boldsymbol{\beta}^{k',1}),
\end{aligned}
$$

we have:

$$L(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) = \gamma_1(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2) + \gamma_2(\omega\boldsymbol{\beta}_1 + (1-\omega)\boldsymbol{\beta}_2).$$

Based on Lemma 1 and Lemma 2, we obtain:

$$L_1(\omega\boldsymbol{\beta}_1^{q,r}+(1-\omega)\boldsymbol{\beta}_2^{q,r},\omega\boldsymbol{\beta}_1^{p,q}+(1-\omega)\boldsymbol{\beta}_2^{p,q})\le\omega L_1(\boldsymbol{\beta}_1^{q,r},\boldsymbol{\beta}_1^{p,q})+(1-\omega)L_1(\boldsymbol{\beta}_2^{q,r},\boldsymbol{\beta}_2^{p,q}),$$
$$L_2(\omega\boldsymbol{\beta}_1^{p,q}+(1-\omega)\boldsymbol{\beta}_2^{p,q})\le\omega L_2(\boldsymbol{\beta}_1^{p,q})+(1-\omega)L_2(\boldsymbol{\beta}_2^{p,q}),$$

which further implies:

$$\begin{aligned}&\gamma_1(\omega\boldsymbol{\beta}_1+(1-\omega)\boldsymbol{\beta}_2)\\&\le\sum_{s=2}^{k'-1}\lambda_{s,s-1}(\omega L_1(\boldsymbol{\beta}_1^{s,s+1},\boldsymbol{\beta}_1^{s-1,s})+(1-\omega)L_1(\boldsymbol{\beta}_2^{s,s+1},\boldsymbol{\beta}_2^{s-1,s}))+\lambda_{k',1}(\omega L_1(\boldsymbol{\beta}_1^{1,2},\boldsymbol{\beta}_1^{k',1})+(1-\omega)L_1(\boldsymbol{\beta}_2^{1,2},\boldsymbol{\beta}_2^{k',1}))\\&=\omega\gamma_1(\boldsymbol{\beta}_1)+(1-\omega)\gamma_1(\boldsymbol{\beta}_2),\\&\gamma_2(\omega\boldsymbol{\beta}_1+(1-\omega)\boldsymbol{\beta}_2)\le\sum_{s=1}^{k'-1}(\omega L_2(\boldsymbol{\beta}_1^{s,s+1})+(1-\omega)L_2(\boldsymbol{\beta}_2^{s,s+1}))+\omega L_2(\boldsymbol{\beta}_1^{k',1})+(1-\omega)L_2(\boldsymbol{\beta}_2^{k',1})\\&\qquad=\omega\gamma_2(\boldsymbol{\beta}_1)+(1-\omega)\gamma_2(\boldsymbol{\beta}_2).\end{aligned}$$

Hence, we have:

$$\begin{aligned}L(\omega\boldsymbol{\beta}_1+(1-\omega)\boldsymbol{\beta}_2)&=\gamma_1(\omega\boldsymbol{\beta}_1+(1-\omega)\boldsymbol{\beta}_2)+\gamma_2(\omega\boldsymbol{\beta}_1+(1-\omega)\boldsymbol{\beta}_2)\\&\le\omega\gamma_1(\boldsymbol{\beta}_1)+(1-\omega)\gamma_1(\boldsymbol{\beta}_2)+\omega\gamma_2(\boldsymbol{\beta}_1)+(1-\omega)\gamma_2(\boldsymbol{\beta}_2)\\&=\omega L(\boldsymbol{\beta}_1)+(1-\omega)L(\boldsymbol{\beta}_2).\end{aligned}$$

Appendix D. Proof of Theorem 2

Since $\varepsilon'\to 0$, it follows from Equation (7) that:

$$y^{s,s+1}=g_{s,s+1}(x^{s,s+1})+\varepsilon.$$

Transform Equation (12) to:

$$\bar{S}(x^s,x^{s+1})=\frac{\int_{x^s}^{x^{s+1}}|g(x)-l(x)|dx}{|x^{s+1}-x^s|}.$$

According to the Law of Iterated Expectations (LIE):

$$\mathbb{E}(\bar{S}(x^s,x^{s+1}))=\mathbb{E}(\bar{S}(x^s,x^{s+1})|\varepsilon^s\cdot\varepsilon^{s+1}<0)P(\varepsilon^s\cdot\varepsilon^{s+1}<0)+\mathbb{E}(\bar{S}(x^s,x^{s+1})|\varepsilon^s\cdot\varepsilon^{s+1}\ge 0)P(\varepsilon^s\cdot\varepsilon^{s+1}\ge 0).$$

We can simplify $\bar{S}(x^s,x^{s+1})$ using basic geometric area calculations. If $\varepsilon^s\cdot\varepsilon^{s+1}<0$, then $\exists\, x'\in(x^s,x^{s+1})$, such that $g(x')=l(x\prime)$. It follows that:

$$\mathbb{E}(\bar{S}(x^s,x^{s+1})|\varepsilon^s\cdot\varepsilon^{s+1}<0)=\frac{\mathbb{E}(|\varepsilon^s|)\cdot|x^s-x'|+\mathbb{E}(|\varepsilon^{s+1}|)\cdot|x^{s+1}-x'|}{2|x^{s+1}-x^s|}.$$

If $\varepsilon^s\cdot\varepsilon^{s+1}\ge 0$, then

$$\begin{aligned}\mathbb{E}(\bar{S}(x^s,x^{s+1})|\varepsilon^s\cdot\varepsilon^{s+1}\ge 0)&=\frac{\mathbb{E}(|\varepsilon^s|+|\varepsilon^{s+1}|)\cdot|x^{s+1}-x^s|}{2|x^{s+1}-x^s|}\\&=\frac{\mathbb{E}(|\varepsilon^s|+|\varepsilon^{s+1}|)}{2}.\end{aligned}$$

Substituting into the original formula, it can be derived:

$$\mathbb{E}(\bar{S}(x^s,x^{(s+1)}))=\frac{\mathbb{E}(|\varepsilon^s|)\cdot|x^s-x'|+\mathbb{E}(|\varepsilon^{s+1}|)\cdot|x^{s+1}-x'|}{2|x^{s+1}-x^s|}P(\varepsilon^s\cdot\varepsilon^{s+1}<0)+\frac{\mathbb{E}(|\varepsilon^s|+|\varepsilon^{s+1}|)}{2}P(\varepsilon^s\cdot\varepsilon^{s+1}\ge 0).$$

Note that $P(\varepsilon^s\cdot\varepsilon^{s+1}<0)+P(\varepsilon^s\cdot\varepsilon^{s+1}\ge 0)=1$, and

$$\begin{aligned}\frac{\mathbb{E}(|\varepsilon^s|)\cdot|x^s-x'|+\mathbb{E}(|\varepsilon^{s+1}|)\cdot|x^{s+1}-x'|}{2|x^{s+1}-x^s|}&=\frac{\mathbb{E}(|\varepsilon^s|)\cdot|\frac{x^s-x'}{x^{s+1}-x^s}|+\mathbb{E}(|\varepsilon^{s+1}|)\cdot|\frac{x^{s+1}-x'}{x^{s+1}-x^s}|}{2}\\&<\frac{\mathbb{E}(|\varepsilon^s|)+\mathbb{E}(|\varepsilon^{s+1}|)}{2},\end{aligned}$$

imply that:

$$\mathbb{E}(\bar{S}(x^s, x^{s+1})) < \mathbb{E}(\frac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}).$$

Appendix E. Proof of Theorem 3

If $\varepsilon^s \cdot \varepsilon^{s+1} < 0$, then $\exists\, x' \in (x^s, x^{s+1})$, such that $g(x') = l(x')$. It follows that:

$$\begin{aligned}\bar{S}(x^s, x^{s+1}) &= \frac{|\varepsilon^s| \cdot |x^s - x'| + |\varepsilon^{s+1}| \cdot |x^{s+1} - x'|}{2|x^s - x^{s+1}|} \\ &= \frac{|\varepsilon^s| \cdot |\frac{x^s - x'}{x^s - x^{s+1}}| + |\varepsilon^{s+1}| \cdot |\frac{x^{s+1} - x'}{x^s - x^{s+1}}|}{2}.\end{aligned}$$

Based on the properties of similar triangles, we can infer that:

$$\begin{aligned}|\frac{x^s - x'}{x^s - x^{s+1}}| &= |\frac{\varepsilon^s}{\varepsilon^s - \varepsilon^{s+1}}|, \\ |\frac{x^{s+1} - x'}{x^s - x^{s+1}}| &= |\frac{\varepsilon^{s+1}}{\varepsilon^s - \varepsilon^{s+1}}|.\end{aligned}$$

Substituting into the original formula, it can be derived:

$$\begin{aligned}\bar{S}(x^s, x^{s+1}) &= \frac{|\frac{\varepsilon^s}{\varepsilon^s - \varepsilon^{s+1}}| \cdot\ |\varepsilon^s| + |\frac{\varepsilon^{s+1}}{\varepsilon^s - \varepsilon^{s+1}}| \cdot |\varepsilon^{s+1}|}{2} \\ &< \frac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}.\end{aligned}$$

If $\varepsilon^s \cdot \varepsilon^{s+1} \geq 0$, then

$$\begin{aligned}\bar{S}(x^s, x^{s+1}) &= \frac{(|\varepsilon^s| + |\varepsilon^{s+1}|) \cdot |x^{s+1} - x^s|}{2|x^{s+1} - x^s|} \\ &= \frac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}.\end{aligned}$$

Appendix F. Proof of Theorem 4

Since $\varepsilon' \to 0$, it follows from Equation (7) that:

$$y^{s,s+1} = g_{s,s+1}(\boldsymbol{x}^{s,s+1}) + \varepsilon.$$

According to (16), (17):

$$\begin{aligned}\varepsilon_{(d)}^{s,s+1} &= y_{(d)}^{s,s+1} - f(\boldsymbol{x}_{(d)}^{s,s+1}) \\ &= y^s + d\frac{y^{s+1} - y^s}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1} - f(\boldsymbol{x}^s + d\frac{\boldsymbol{x}^{s+1} - \boldsymbol{x}^s}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1}) \\ &= y^s + d\frac{y^{s+1} - y^s}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1} - g(\boldsymbol{x}^s + d\frac{\boldsymbol{x}^{s+1} - \boldsymbol{x}^s}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1}).\end{aligned}$$

Let $\dfrac{d}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil + 1} = m$, we have:

$$\begin{aligned}\varepsilon_{(d)}^{s,s+1} &= y^s - g(\boldsymbol{x}^s) + m \cdot (y^{s+1} - g(\boldsymbol{x}^{s+1}) - y^s + g(\boldsymbol{x}^s)) \\ &= \varepsilon^s + m \cdot (\varepsilon^{s+1} - \varepsilon^s) \\ &= (1 - m) \cdot \varepsilon^s + m \cdot \varepsilon^{s+1}.\end{aligned}$$

Since $0 < m < 1$, we can infer that:

$$\mathbb{E}(\frac{\sum_{d=1}^{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} |\varepsilon_{(d)}^{s,s+1}|}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil}) = \frac{\sum_{d=1}^{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} \mathbb{E}|\varepsilon_{(d)}^{s,s+1}|}{\left\lceil \frac{n\eta}{\text{dist}_{\text{sum}}} \text{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil}$$

$$
\begin{aligned}
&= \frac{\sum_{d=1}^{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} \mathbb{E}|(1-m)\cdot\varepsilon^s + m\cdot\varepsilon^{s+1}|}{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} \\
&< \frac{\sum_{d=1}^{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} \mathbb{E}(|(1-m)\cdot\varepsilon^s| + |m\cdot\varepsilon^{s+1}|)}{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} \\
&= \frac{\mathbb{E}|\varepsilon^s| \sum_{d=1}^{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} (1-m)}{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} + \frac{\mathbb{E}|\varepsilon^{s+1}| \sum_{d=1}^{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} m}{\left\lceil \frac{n\eta}{\mathrm{dist}_{\mathrm{sum}}} \mathrm{dist}(\boldsymbol{x}^s, \boldsymbol{x}^{s+1}) \right\rceil} \\
&= \mathbb{E}\left(\frac{|\varepsilon^s| + |\varepsilon^{s+1}|}{2}\right).
\end{aligned}
$$

Data availability

Data will be made available on request.

References

[1] N. Pavlidou, A.H. Vinck, J. Yazdani, B. Honary, Power line communications: state of the art and future trends, IEEE Commun. Mag. 41 (4) (2003) 34–40.
[2] D. Cabric, S.M. Mishra, R.W. Brodersen, Implementation issues in spectrum sensing for cognitive radios, in: Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, vol. 1, 2004, Ieee, 2004, pp. 772–776.
[3] D. Karimi, H. Dou, S.K. Warfield, A. Gholipour, Deep learning with noisy labels: exploring techniques and remedies in medical image analysis, Med. Image Anal. 65 (2020) 101759.
[4] W. Jiang, Applications of deep learning in stock market prediction: recent progress, Expert Syst. Appl. 184 (2021) 115537.
[5] X. Chu, I.F. Ilyas, S. Krishnan, J. Wang, Data cleaning: overview and emerging challenges, in: Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 2201–2206.
[6] L. Berti-Equille, T. Dasu, D. Srivastava, Discovery of complex glitch patterns: a novel approach to quantitative data cleaning, in: 2011 IEEE 27th International Conference on Data Engineering, IEEE, 2011, pp. 733–744.
[7] A. Buades, B. Coll, J.-M. Morel, Non-local means denoising, Image Process. Line 1 (2011) 208–212.
[8] M. Lukasik, S. Bhojanapalli, A. Menon, S. Kumar, Does label smoothing mitigate label noise?, in: International Conference on Machine Learning, PMLR, 2020, pp. 6448–6458.
[9] C. Liguori, A. Paolillo, A. Ruggiero, D. Russo, Outlier detection for the evaluation of the measurement uncertainty of environmental acoustic noise, IEEE Trans. Instrum. Meas. 65 (2) (2015) 234–242.
[10] J.W. Osborne, Best Practices in Data Cleaning: A Complete Guide to Everything You Need to do Before and After Collecting Your Data, Sage Publications, 2012.
[11] M.N.K. Sikder, F.A. Batarseh, Outlier detection using ai: a survey, AI Assur. (2023) 231–291.
[12] E. Peterfreund, M. Gavish, Multidimensional scaling of noisy high dimensional data, Appl. Comput. Harmon. Anal. 51 (2021) 333–373.
[13] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, mixup: beyond empirical risk minimization, arXiv preprint, arXiv:1710.09412, 2017.
[14] H. Alqahtani, M. Kavakli-Thorne, G. Kumar, Applications of generative adversarial networks (gans): an updated review, Arch. Comput. Methods Eng. 28 (2021) 525–552.
[15] D. Nie, R. Trullo, J. Lian, L. Wang, C. Petitjean, S. Ruan, Q. Wang, D. Shen, Medical image synthesis with deep convolutional adversarial networks, IEEE Trans. Biomed. Eng. 65 (12) (2018) 2720–2730.
[16] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, A.P. Sales, Generation and evaluation of synthetic patient data, BMC Med. Res. Methodol. 20 (2020) 1–40.
[17] F.B. Hildebrand, Introduction to Numerical Analysis, Courier Corporation, 1987.
[18] J.-P. Lewis, Algorithms for solid noise synthesis, in: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, 1989, pp. 263–270.
[19] D. Walawalkar, Z. Shen, Z. Liu, M. Savvides, Attentive cutmix: an enhanced data augmentation approach for deep learning based image classification, arXiv preprint, arXiv:2003.13048, 2020.
[20] A. Bhosekar, M. Ierapetritou, Advances in surrogate based modeling, feasibility analysis, and optimization: a review, Comput. Chem. Eng. 108 (2018) 250–267.
[21] N. Nasios, A.G. Bors, Variational learning for Gaussian mixture models, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 36 (4) (2006) 849–862.
[22] N. Friedman, M. Goldszmidt, T.J. Lee, Bayesian network classification with continuous attributes: getting the best of both discretization and parametric fitting, in: ICML, vol. 98, Citeseer, 1998, pp. 179–187.
[23] O. Arandjelovic, R. Cipolla, Incremental learning of temporally-coherent Gaussian mixture models, 2005.
[24] G.M. Raab, B. Nowok, C. Dibben, Practical data synthesis for large samples, J. Priv. Confid. 7 (3) (2016) 67–97.
[25] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[26] H. Ling, K. Okada, Diffusion distance for histogram comparison, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, IEEE, 2006, pp. 246–253.
[27] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A.A. Bharath, Generative adversarial networks: an overview, IEEE Signal Process. Mag. 35 (1) (2018) 53–65.
[28] S. Bond-Taylor, A. Leach, Y. Long, C.G. Willcocks, Deep generative modelling: a comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models, IEEE Trans. Pattern Anal. Mach. Intell. 44 (11) (2021) 7327–7347.
[29] M. Li, J. Lin, Y. Ding, Z. Liu, J.-Y. Zhu, S. Han, Gan compression: efficient architectures for interactive conditional gans, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 5284–5294.
[30] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1125–1134.
[31] A. Van Den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: International Conference on Machine Learning, PMLR, 2016, pp. 1747–1756.
[32] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., Conditional image generation with pixelcnn decoders, Adv. Neural Inf. Process. Syst. 29 (2016).

[33] T. Elguebaly, N. Bouguila, Bayesian learning of finite generalized Gaussian mixture models on images, Signal Process. 91 (4) (2011) 801–820.
[34] S.J. Roberts, D. Husmeier, I. Rezek, W. Penny, Bayesian approaches to Gaussian mixture modeling, IEEE Trans. Pattern Anal. Mach. Intell. 20 (11) (1998) 1133–1142.
[35] N. Bouguila, D. Ziou, A Dirichlet process mixture of generalized Dirichlet distributions for proportional data modeling, IEEE Trans. Neural Netw. 21 (1) (2009) 107–122.
[36] S. Hong, D. Yang, J. Choi, H. Lee, Inferring semantic layout for hierarchical text-to-image synthesis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7986–7994.
[37] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Lof: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 93–104.
[38] N.M. Razali, J. Geraghty, et al., Genetic algorithm performance with different selection strategies in solving tsp, in: Proceedings of the World Congress on Engineering, vol. 2, International Association of Engineers, Hong Kong, China, 2011, pp. 1–6.
[39] Ş. Gülcü, M. Mahi, Ö.K. Baykan, H. Kodaz, A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem, Soft Comput. 22 (2018) 1669–1685.
[40] Q. Sun, Z. Ge, A survey on deep learning for data-driven soft sensors, IEEE Trans. Ind. Inform. 17 (9) (2021) 5853–5866.
[41] Y. Guo, W. Wang, X. Wang, A robust linear regression feature selection method for data sets with unknown noise, IEEE Trans. Knowl. Data Eng. 35 (1) (2021) 31–44.
[42] G. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[43] W. Cukierski, Bike sharing demand, https://kaggle.com/competitions/bike-sharing-demand, 2014.
[44] S. Vito, Air quality, UCI Machine Learning Repository, https://doi.org/10.24432/C59K5F, 2016.
[45] S. Moro, P. Rita, B. Vala, Facebook metrics, UCI Machine Learning Repository, https://doi.org/10.24432/C5QK55, 2016.
[46] P. Cortez, A. Morais, Forest fires, UCI Machine Learning Repository, https://doi.org/10.24432/C5D88D, 2008.
[47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[48] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv:1708.07747 [cs.LG], 2017.
[49] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[50] Y. Netzer, T. Wang, A. Coates, et al., Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, vol. 2011, 2011, p. 7.