processes

MDPI

Article

Research on Imbalanced Data Regression Based on Confrontation

Xiaowen Liu [1,2] and Huixin Tian [1,2,*]

[1] School of Control Science and Engineering, Tiangong University, Tianjin 300387, China
[2] Tianjin Key Laboratory of Intelligent Control of Electrical Equipment, Tiangong University, Tianjin 300387, China
* Correspondence: tianhuixin@tiangong.edu.cn

Abstract: The regression model has higher requirements for the quality and balance of data to ensure the accuracy of predictions. However, there is a common problem of imbalanced distribution in real datasets, which directly affects the prediction accuracy of regression models. In order to solve the problem of data imbalance regression, considering the continuity of the target value and the correlation of the data and using the idea of optimization and confrontation, we propose an IRGAN (imbalanced regression generative adversarial network) algorithm. Considering the context information of the target data and the disappearance of the deep network gradient, we constructed a generation module and designed a composite loss function. In the early stages of training, the gap between the generated samples and the real samples is large, which easily causes the problem of non-convergence. A correction module is designed to train the internal relationship between the state and action as well as the subsequent state and reward of the real samples, guide the generation module to generate samples, and alleviate the non-convergence of the training process. The corrected samples and real samples are input into the discriminant module. On this basis, the confrontation idea is used to generate high-quality samples to balance the original samples. The proposed method is tested in the fields of aerospace, biology, physics, and chemistry. The similarity between the generated samples and the real samples is comprehensively measured from multiple perspectives to evaluate the quality of the generated samples, which proves the superiority of the generated module. Regression prediction is performed on the balanced samples processed by the IRGAN algorithm, and it is proven that the proposed algorithm can improve the prediction accuracy in terms of the imbalanced data regression problem.

Keywords: imbalanced data; imbalanced regression; IRGAN

check for updates

Citation: Liu, X.; Tian, H. Research on Imbalanced Data Regression Based on Confrontation. *Processes* **2024**, *12*, 375. https://doi.org/10.3390/pr12020375

Academic Editor: Olympia Roeva

Received: 21 January 2024
Revised: 8 February 2024
Accepted: 11 February 2024
Published: 13 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In theoretical research, it is usually assumed that the dataset is balanced. In contrast, in real life, imbalanced datasets are common, which brings challenges to data analysis in almost all research fields [1]. Ignoring the problem of data imbalance, it is unreasonable to predict directly using the original dataset. A characteristic of imbalanced data is that the sample size of a certain category is much smaller than that of other categories. Most of the samples are in the normal range, and the minority samples are in the abnormal range. This results in the prediction model being more inclined to predict the minority samples at majority sample intervals, thus reducing the prediction accuracy of the minority samples [2]. However, the information extracted from the minority samples is usually more valuable than the information extracted from the majority. Therefore, it is vital to correctly deal with the problem of data imbalance to improve the effect of the prediction model, which has become an important topic in current research. At present, many scholars have proposed solutions to the problem of data imbalance. These methods are mainly divided into two categories: solutions for classification problems and solutions for regression problems [3].

The imbalanced classification problem refers to the imbalance of the number of samples in different categories in the classification problem [4]. At present, many studies have been carried out to solve the problem of imbalanced classification data [5]. The existing solutions include resampling [6–10], ensemble learning [11–15], sample generation [16–22], and so on. Among them, resampling and ensemble learning both start from the local neighborhood of sample points, without considering the overall distribution of the original dataset. Sampling from the distribution of data to increase the number of minority samples is an ideal imbalanced data processing method, in which the idea of using a generator to generate minority samples is applied. In terms of generating samples, the most typical sample generation algorithms are variational auto-encoder (VAE) and generative adversarial network (GAN). In order to learn the distribution information of the original sample, Diederik P et al. [16] proposed the VAE algorithm. The algorithm includes an encoder and a decoder. The encoder is used to learn the distribution of the original samples, and the decoder is used to generate samples that conform to the distribution of the original samples. However, the VAE model has the disadvantage of poor generalization ability. Goodfellow et al. [17] proposed the GAN algorithm. The GAN consists of two networks: a generator and a discriminator. The goal of the generator is to generate samples similar to real samples as much as possible to deceive the discriminator. The purpose of the discriminator is to distinguish the samples generated by the generator from the real samples. When the Nash equilibrium state is reached, the performance of the generator and the discriminator is optimal, and the generator can generate high-quality samples. However, the GAN has the disadvantage of training instability due to gradient disappearance and mode collapse [18]. In order to improve stability, a large number of variants of GANs have been developed, such as those that change the model structure, including internal and external structures, add other input conditions, change the loss function, etc. For internal structural changes, the deep convolutional generative adversarial network (DCGAN) [19] uses convolutional neural networks and deconvolutional neural networks to construct a generator and a discriminator, respectively, and provide experimental guidance on how to establish a stable GAN network. For conditional settings, the conditional generative adversarial network (CGAN) [20] adds conditional variables to the generator and discriminator at the same time so that the generation of sample data is based on conditional variables. For the change of loss function, the Wasserstein generative adversarial network (WGAN) [21] replaces the JS divergence with the Wasserstein distance to estimate the distance between the real samples' distribution and the generated samples' distribution, making the adversarial learning of the model more stable. Bao et al. [22] proposed the CVAE-GAN model, which is based on the CVAE model and uses CGANs to optimize the generator so that the generated samples are both real and diverse. Although a large number of studies have been carried out, these studies mainly focus on categorical data with discrete target variables.

The imbalanced regression problem occurs when the frequency of some target values in the regression dataset is extremely low and it is easy for the model to ignore these very few target values, resulting in the poor prediction performance of the model on these samples [23]. For the problem of regression data imbalance, the method of solving the problem of classification data imbalance is mainly applied directly to the imbalanced regression task [24]. Torgo et al. [25] applied the SMOTE algorithm that generates classified samples for the regression problem, but this method is the same as the SMOTE algorithm that deals with the imbalance of classified data. Because the SMOTE algorithm cannot learn the data distribution of imbalanced datasets, it is easy to produce the problem of distribution marginalization. Branco et al. [26] proposed the REBAGG algorithm, which is an ensemble method based on bagging and combines data preprocessing methods to solve the problem of imbalanced data in regression tasks. Despite their effectiveness, these methods do not take into account the regression characteristics of the sample. In the regression problem, the model predicts the value rather than the category. The imbalanced regression problem needs to study the distribution of the data more carefully in order to predict the value more accurately. The imbalanced regression problem is more complex

than the imbalanced classification problem. By analyzing the relationship between the distribution of the target value of the regression sample and the test error of the prediction model, Yang et al. [27] proposed the concept of deep imbalanced regression by analyzing the relationship between the target value distribution of regression samples and the test error of the prediction model. According to the characteristics of imbalanced regression data, they proposed label distribution smoothing (LDS) and feature distribution smoothing (FDS). However, this method still uses the method of data interpolation when generating minority regression samples, which is prone to overfitting. Ren et al. [28] proposed a new loss function called balanced mean squared error (BMSE) to solve the problem of imbalanced regression. Specifically, the BMSE loss function uses a weighting method to assign different degrees of importance to different samples. Rahul et al. [29] proposed the spatial-SMOTE algorithm, which handles the oversampling of rare events by preserving the importance of the spatial distribution of data. Nevertheless, the above two algorithms are only applicable to specific datasets. By analyzing the existing research, it is found that the current research on imbalanced regression mainly focuses on model integration and data interpolation. When using a model ensemble to solve the problem of imbalanced regression data, the method of solving the problem of imbalanced classification is directly applied to the problem of imbalanced regression, without considering the continuity of the sample target value. When using the data interpolation method to generate minority regression samples, the regression characteristics of the original samples are not considered. Additionally, the existing methods have problems such as slow convergence speed and a lack of wide applicability when processing data.

Considering the defects of existing methods for imbalanced regression problems, this paper proposes the IRGAN algorithm. This algorithm primarily addresses two tasks: (1) focusing on the problem of the number of samples in the original regression samples varying greatly in different target value intervals, a combination of optimization and confrontation ideas is used to generate regression samples; (2) combining the generated samples and the original samples as new samples for regression prediction. The algorithm includes four parts: generation, correction, discriminant, and regression modules. Due to the imbalance of data distribution, firstly, considering the contextual information of data, the generation module is designed. Combined with the characteristics of the regression problem, a composite loss function is designed to guide the generation of regression samples to further ensure the quality of the generated samples. In the early stages of training, the gap between the generated samples and the real samples is large, which makes it easy for the training process to not converge. By introducing the correction module, using the optimization idea to implement the decision and combining it with the deep neural network, the internal relationship between the state and action and the subsequent state and reward of the real samples is trained to guide the generation module and improve the quality and convergence speed of the generated samples. Then, based on the confrontation idea, the generation module and the discriminant module are continuously optimized to make the generation module and the discriminant module reach the Nash equilibrium. At this time, the generation module can generate high-quality samples to balance the original samples. Finally, the regression prediction is realized by using the regression module in the algorithm. In short, in order to solve the problem of imbalanced regression of data, we propose the IRGAN algorithm. The effectiveness and feasibility of the algorithm are demonstrated by experimental verification in the fields of aerospace, biology, physics, and chemistry. The main contributions of this paper are as follows:

(1) For the problem of imbalanced data regression, this study takes into account the continuity of the target variable and the correlation between the data and uses a method that combines optimization and the confrontation idea to generate regression samples.

(2) According to the continuous and imbalanced characteristics of the target variables of the original regression data, the generation module is designed to generate the samples closer to the original samples.

(3) Focusing on the problems of the gap between the generated samples and the real samples being large and the training process not converging at the initial stages of training, a correction module is designed to guide the generation module.

2. Algorithm Design

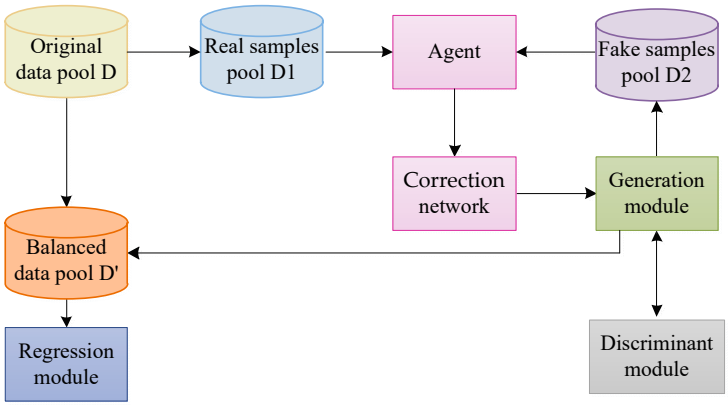
In traditional regression problems, the predicted numerical variables usually have a balanced distribution. That is, the frequency of the occurrence of each value in the dataset is roughly equal. Therefore, in the prediction process, the algorithm can optimize the model by minimizing criteria such as average error or mean square error. However, in imbalanced regression problems, the distribution of the predicted numerical variables is imbalanced. This means that the frequency of some values in the dataset may be much higher than other values. Due to the imbalance of training data, the traditional regression prediction model may tend to predict the category with a large number of samples, which leads to a poor prediction effect of the model regarding minority samples [30]. In order to improve the regression prediction accuracy of balanced data, we designed the IRGAN algorithm. This algorithm is designed to tackle two tasks. first, for the problem of the number of samples in the original regression samples varying greatly in different target value intervals, high-quality minority samples are generated to balance the original samples. Then, the generated samples and the original samples are combined as new samples for regression prediction. It contains four kinds of samples: (1) original data pool D, storing all the original samples; (2) real samples pool D1, which is used to collect real samples obtained during the interaction between the agent and the environment; (3) fake samples pool D2, which is used to collect samples generated by the generation module; the real samples in pool D1 and fake samples in pool D2 together provide training samples for the agent; (4) balanced data pool D′, which stores the balanced samples. The balanced samples include the original samples and the samples generated by the generated module after training. The data in the sample pool is finally used for regression prediction. The algorithm consists of four modules. (1) The generation module is used to generate minority samples. (2) The correction module includes an agent and a correction network. Similar to the human brain, the agent can perceive environmental information and make optimal decisions. The correction network is used to train the relationship between the state action pair (s, a) and the subsequent state reward pair (s', r), and guide the generation module to improve performance and accelerate the convergence speed of model training. (3) The discriminant module determines whether the input samples are real or fake, feeds back the discriminant information to the generation module, and optimizes the generation module in order to improve the quality of the generated samples. (4) The regression module performs regression prediction for the balanced samples.

The overall research framework diagram is shown in Figure 1a, and the model structure of the IRGAN algorithm is shown in Figure 1b. The following sections will introduce the four parts of the generation module: the discriminant module, the correction module, and the regression module.

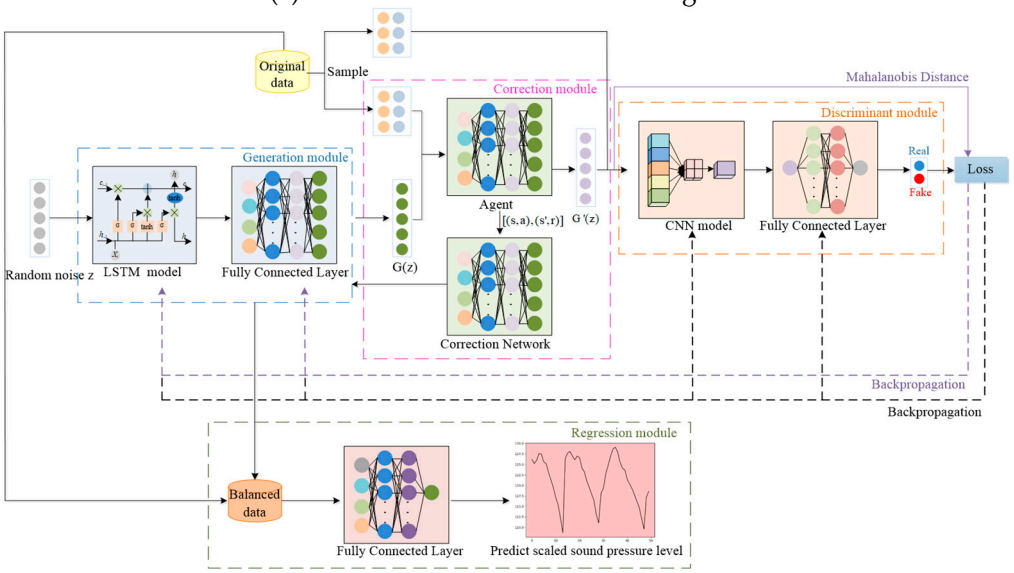
2.1. Generation Module

The function of the generation module is to generate regression data. Firstly, the random Gaussian noise, z, is introduced into the generation module as an input signal. The purpose of inputting the random Gaussian noise is to increase the diversity of the generation module and improve the exploration ability, so as to better simulate the distribution of data. The Gaussian noise is output to generate sample $G(z)$ after processing by the neural network. The loss function, $\mathrm{E}_{z\sim p(z)}[\log D(G(z))]$, of the generation module is to minimize the JS divergence between the generated distribution and the real distribution. However, when there is no overlap between the two distributions or when the overlap is extremely small, the JS divergence is constant, and the generation module is be updated. The loss function was at a constant value, resulting in the disappearance of the gradient. Focusing on the problem of gradient disappearance in the training process, this paper designs the

loss function of the generation module and uses the Mahalanobis distance to calculate the distance between the generated data and the real data, which is more suitable for generating regression data, and effectively alleviates the problem of gradient disappearance in the training process.



(**a**) Overall research framework diagram



(**b**) LCRGANRL algorithm model structure diagram

Figure 1. The sketch of LCRGANRL algorithm.

The loss function formula of the generation module is divided into two parts. They are written for the generated data and the distance between the generated data and the real data. The overall loss function is shown in (1).

$$L_{\mathrm{G}} = \mathrm{E}_{x \sim pdata(x)}[\log D(G(z))] + \alpha(X - G(z))' V^{-1}(x - G(z)) \quad (1)$$

(1) The first part of the generation module loss function is for the generated data, as shown in (2).

$$L_{\mathrm{G}}(\mathrm{G}(z)) = \mathrm{E}_{z \sim p(z)}[\log D(G(z))] \quad (2)$$

where $G(z)$ is the generated fake sample, which begins with random noise and then is generated by the generation module. For the generation module, its goal is to generate a generated sample closer to the real sample so that the discriminant module considers it to be a real sample—that is, if $D(G(z))$ is closer to 1, then $\log D(G(z))$ is close to 0, which can minimize the loss function of the generation module. The smaller the loss of the generating module is, the more realistic the generated samples are, and the discriminant module cannot be identified.

(2) Considering the correlation between the characteristics of regression data, in the second part of the generation module loss function, the Mahalanobis distance is added to measure the distance between the real samples and the generated samples. The Mahalanobis distance is a commonly used distance index in metric learning. It is used as a similarity index between data like Euclidean distance, Manhattan distance, and Hamming distance, but it can deal with the problem of non-independent and identical distributions between dimensions in high-dimensional data. Upon adding this item to the loss function of the generation module, the formula is as follows:

$$L_G(X, G(z)) = \alpha(X - G(z))' \mathbf{V}^{-1}(X - G(z)) \tag{3}$$

where $\mathbf{V}$ is the covariance matrix, and $\mathbf{V}^{-1}$ is the inverse matrix of the covariance matrix. The Mahalanobis distance adds the consideration of the correlation between features, and it is invariant to all nonsingular linear transformations. This shows that it is not affected by the selection of feature dimension and considers the influence of dimension on the sample distance, which can measure the distance between the real samples and the generated samples more scientifically. The loss function value of the generated module is calculated, and then the distance between the real samples and the generated samples is continuously reduced by back propagation so that the generated sample is closer to the real sample, thereby improving the quality of the generated sample. Increasing this item is conducive to improving the convergence speed of the model and is also more conducive to regression prediction. α is a hyperparameter that can adjust the weight of the Mahalanobis distance in the loss function of the generation module. Its influence on the model will be discussed in subsequent experiments. In addition, the Mahalanobis distance has superior smoothing characteristics relative to the JS divergence. Even if the overlap of the two distributions is extremely small, the Mahalanobis distance can also reflect the distance between them. Therefore, after adding this item, the loss function of the generation module will not become constant with training and the gradient can be continuously updated, which effectively alleviates the problem of gradient disappearance during training.

2.2. Correction Module

At the beginning of training, the samples generated by the generation module are only theoretically feasible data samples and the gap with the real samples is large, which easily causes instability in the training process, so it needs to be corrected by the correction module. The function of the correction module is to guide the generation module. The correction module is composed of an agent and a correction network. Specifically, the samples are selected by combining real samples and virtual samples, and the data is provided to the agent for training the action value function network, so as to find the optimal strategy. The correction network analyzes the relationship among the state, action, and reward.

The real samples in the learning process are defined as a pair of states and actions and subsequent states and rewards. The state at the previous moment corresponds to the corresponding action, which is called the state action pair (s, a). The state at the next moment and the reward are called the subsequent state reward pair (s', r). Therefore, the real samples, $D_x = [s, a, s', r]$, can be divided into two parts:

$$D_x = [(s, a), (s', r)] = [x_1, x_2] \tag{4}$$

where x_1 denotes the state action pair, and x_2 denotes the subsequent state reward pair. The input is x_1, and the output is x_2. The correction network is used to train the internal relationship between x_1 and x_2. Consistent with the real samples, the samples $G_Z = [s_z, a_z, s_z', r_z]$ generated by the generation module can also be divided into two parts:

$$G_z = [s_z, a_z, s_z', r_z] = [G_1(z), G_2(z)] \tag{5}$$

where $G_1(z)$ represents the generated state action pair, and $G_2(z)$ represents the generated subsequent state reward pair. In order to improve the quality of the generated samples, based on the generated $G(z)$, the relationship between the generated $G_1(z)$ and $G_2(z)$ should be consistent with the relationship in the real samples $[x_1, x_2]$. Therefore, combined with the generated sample $G_1(z)$, $G_1(z)$ is input into the correction module, and the output is used as the constructed subsequent state reward pair $G_2(z)'$. The goal is to make the generated subsequent state reward pair, $G_2(z)$, and the constructed subsequent state reward pair, $G_2(z)'$, have a high similarity. Further, the generation module is promoted to generate more high-quality samples and accelerate the convergence speed of model training. The loss function formula of the correction module is as follows:

$$L_{\mathrm{u}} = \sum_i p(i) \log \frac{1}{q(i)} - \sum_i p(i) \log \frac{1}{p(i)} = \sum_i p(i) \log \frac{p(i)}{q(i)} \tag{6}$$

where p denotes the generated subsequent state reward pair, $G_2(z)$, and q denotes the constructed subsequent state reward pair, $G_2(z)'$.

2.3. Discriminant Module

The goal of the discriminant module is to extract the features of the input data and distinguish the authenticity of the sample as much as possible. The real and fake samples are put into the discriminant module, and then the loss value of the discriminant module is calculated. The loss includes the real loss and the fake loss. The loss function of the discriminant module is shown in (7).

$$L_{\mathrm{D}} = \mathrm{E}_{x \sim pdata(x)}[\log D(X)] + \mathrm{E}_{\mathrm{z} \sim p(\mathrm{z})}[\log(1 - D(G(z))] \tag{7}$$

(1) The loss function of the first part of the discriminant module refers to the real samples as the input of the discriminant module. The loss function after input is shown in (8).

$$L_{\mathrm{D}}\Big(X) = \mathrm{E}_{x \sim pdata(x)}[\log D(X)] \tag{8}$$

where X is the real sample. For the discriminant module, its goal is to determine whether the real sample is real. The closer the discriminant value, $D(X)$, of the real sample is to 1, the closer the $D(X)$ is to 0, which can minimize the loss of the discriminant module.

(2) The loss function of the second part of the discriminant module refers to the generated sample as the input of the discriminant module. The loss function after input is shown in (9).

$$L_{\mathrm{D}}(G(z)) = \mathrm{E}_{\mathrm{z} \sim p(\mathrm{z})}[\log(1 - D(G(z))] \tag{9}$$

where $G(z)$ is the generated sample. For the discriminant module, its goal is to determine the generated sample as fake. The closer the discriminant value, $D(G(z))$, of the generated sample is to 0, the closer the $\log(1 - D(G(z))$ is to 0, so as to minimize the loss that the discriminant module discriminates as fake.

2.4. Regression Module

The regression module is the last module of the algorithm, which is mainly responsible for the regression prediction of the balanced samples. When using neural networks for regression, it is necessary to establish a mapping from input features to continuous output values. The formula is as follows: $\mathbf{y} = f(\mathbf{W} \bullet \mathbf{X} + \mathbf{b})$. where $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias, and f usually represents the activation function. However, its application scope is far more than that. f can represent a variety of types of functions, such as the regularization function, normalized function, and so on. The regression process usually involves multi-layer nonlinear mapping. It is assumed that the size of the processed matrix

is $m \times n$, where m is the number of samples, and n is the number of features. The input $\mathbf{X}$ and output matrix $\mathbf{Y}$ of regression prediction in neural network are defined as follows:

$$\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{1n-1} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn-1} \end{pmatrix} \tag{10}$$

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \tag{11}$$

Before constructing the model, we first need to divide the input dataset into a training set and a test set, and then divide the data from each interval according to the ratio of 7:3. A total of 70% of the data is used to train the model to grasp the potential rules of the data, and the remaining 30% is used as a test set to evaluate the performance of the model in new situations. This segmentation strategy helps the model make accurate predictions on unknown data and enhances its generalization ability. In the model training phase, the loss function is used to measure the size of the prediction error, which is a key indicator to evaluate the performance of the model. The loss function is shown in (12).

$$L_{\mathrm{R}} = \frac{1}{N}\sum_{i=1}^{N} \mid y_i - \widehat{y}_i \Big| \tag{12}$$

where y_i is the real value, $\widehat{y}_i$ is the predicted value, and N is the number of samples. Compared with other loss functions, this loss function is more robust to outliers and is very useful in feature selection and model interpretation. It can help identify the most important features, simplify the model, and improve the generalization ability. The loss function calculates the average value of the prediction error of all samples, which reflects the accuracy of the model prediction results. By optimizing this loss function, the model parameters can be adjusted to make the predicted value as close as possible to the real value, thereby improving the prediction ability of the model.

3. Experiments and Analysis

This paper proposes a new algorithm, IRGAN, which is used to solve the problem of imbalanced regression of data. Firstly, in order to verify the effect of the proposed algorithm on sample generation, this paper selects three comparison models: CGAN, VAE, and CVAE-GAN. Then, regression predictions are made separately for the imbalanced samples and the balanced samples using different algorithms, demonstrating the effectiveness of the proposed algorithm in improving prediction accuracy.

3.1. Datasets and Evaluation Indicators

3.1.1. Datasets

Four datasets were selected for this experiment. These datasets are the Airfoil Self-Noise data from the NASA dataset, Abalone data, Yacht Hydrodynamics data, and Concrete Compressive Strength data from the UCI dataset. Figure 2 shows the probability density distribution of the target values of these datasets.

(1) Airfoil Self-Noise: This NASA dataset is obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. The dataset contains 1503 samples, with the following inputs—frequency, chord length, free-stream velocity, and suction side displacement thickness, while the only output is the scaled sound pressure level. The minimum scale sound pressure level is 103.4 dB, and the maximum scale sound pressure level is 141 dB. The distribution of target values is shown in Figure 2a, showing a significant data imbalance.

(2) Abalone: The Abalone dataset aims to predict the age of abalone by measuring its properties. The dataset contains 4177 of abalone's physical properties and corresponding

ages. As shown in Figure 2b, the width of each histogram is 1 year. The minimum age of these samples is 1 year old, and the maximum age is 29 years old. The number of samples in each histogram varies from 22 to 2769. It can be seen from the figure that the dataset shows an obvious data imbalance.

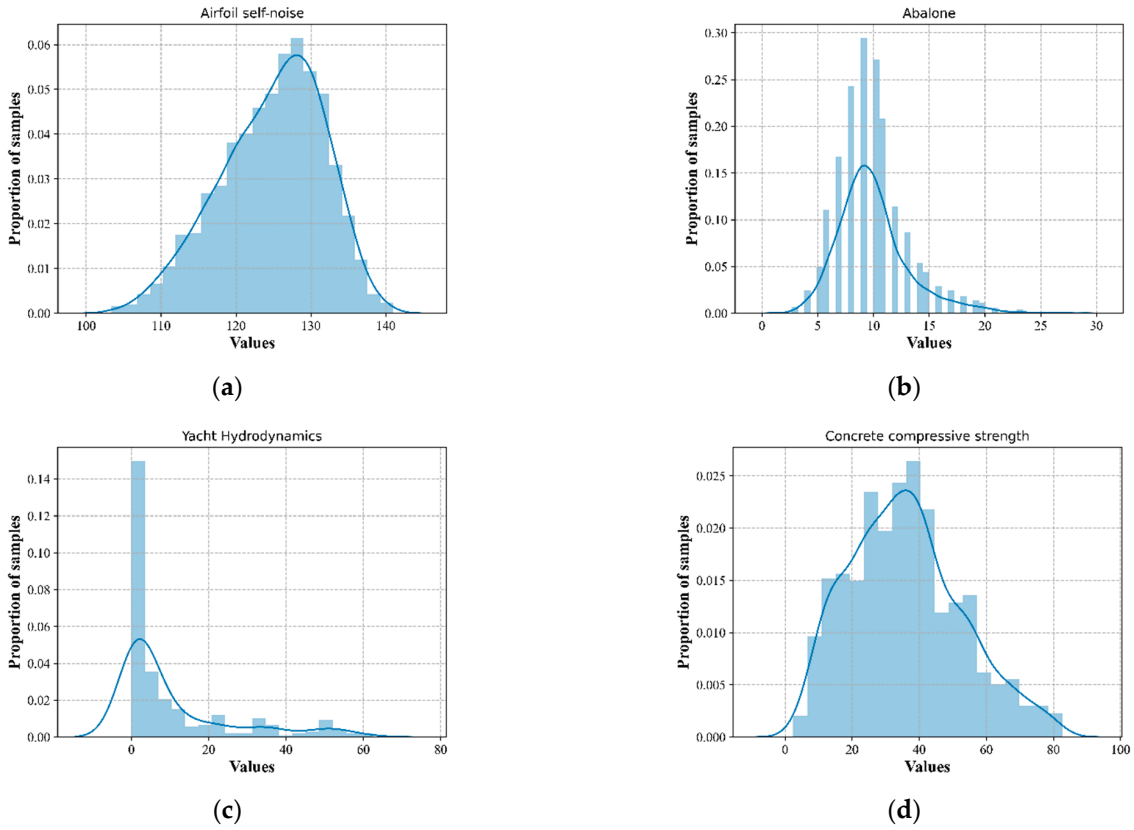


Figure 2. Data target value distribution diagram.

(3) Yacht Hydrodynamics: The Yacht Hydrodynamics dataset predicts the hydrodynamic performance of sailing yachts from dimensions and velocity. The input variables are the longitudinal position of the buoyancy center, the prism coefficient, the length-displacement ratio, the beam-draft ratio, the length-beam ratio, and the Froude number. The output variable is the residual resistance per unit displacement weight. The minimum value is 0, and the maximum value is 79.68. As shown in Figure 2c, a large number of data values are concentrated in the area close to 0, and the frequency of high values gradually decreases. This distribution indicates that the target value shows a clearly skewed distribution and skews to the lower value.

(4) Concrete Compressive Strength: Concrete is the most important material in civil engineering. The compressive strength of concrete is a highly nonlinear function of age and ingredients. This dataset contains 1030 samples, and the input variables cover a variety of components in the concrete mixture: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age of concrete. The output variable is the compressive strength of concrete, for which the minimum value is 2.33 MPa, and the maximum value is 82.6 MPa. It can be seen from Figure 2d that the frequency of the target value is higher in some specific intervals, while it is relatively low in other intervals, indicating that the data distribution is imbalanced.

3.1.2. Evaluation Indicators

This section will introduce the evaluation indicators related to the experiment, including those that measure the similarity between generated samples and real samples, such as Jensen–Shannon divergence (JS divergence), maximum mean discrepancy (MMD distance),

and cosine similarity, and comprehensively evaluate the similarity between generated samples and real samples from multiple perspectives. In addition, this section will also explore the mean square error (MSE), which is a test error indicator that evaluates the performance of the prediction model.

(1) JS divergence

The JS divergence is a probability statistical method, which can calculate the difference between the distribution of two variables in the same sample space. The JS divergence is obtained by changing the Kullback–Leibler divergence (KL divergence), which makes up for some shortcomings of the KL divergence. The KL divergence measures the statistical correlation between two variables from the perspective of entropy. Entropy can represent the distance between two variables in information theory. If U and V denote the distribution of two random variables, then the KL divergence from U to V is as follows:

$$D_{KL}(U||V) = \int U(t) \log \frac{U(t)}{V(t)} dt \geq 0 \tag{13}$$

where only when U and V are exactly the same, the inequality on the right takes the equal sign. It is not difficult to see that the KL divergence is asymmetric—that is, $D_{\mathrm{KL}}(U||V) \neq D_{\mathrm{KL}}(V||U)$. Considering that the value range of the KL divergence is from 0 to infinity and that the KL divergence is not symmetrical, it is impossible to truly and accurately evaluate the dispersion degree of the two variable distributions. As a variant form of the KL divergence, the JS divergence effectively solves these two problems by constructing the average probability distribution of two probability distributions. The calculation formula of JS divergence between U and V is as follows:

$$D_{\mathrm{JS}}(U||V) = \frac{1}{2} D_{\mathrm{KL}}(U||\frac{U+V}{2}) + \frac{1}{2} D_{\mathrm{KL}}(V||\frac{U+V}{2}) \tag{14}$$

It can be seen that $D_{\mathrm{JS}}(U||V) = D_{\mathrm{JS}}(V||U)$, which means that the JS divergence is symmetrical. In addition, the value range of the JS divergence is 0 to 1. The higher the distribution similarity of the two variables, the closer the JS divergence value between them is to 0; otherwise it is closer to 1. The JS divergence effectively eliminates the disadvantages of the asymmetric KL divergence and its too large value interval. Compared with the KL divergence, it can more accurately evaluate the similarity of the distribution of two variables.

(2) MMD: MMD is a method used to measure the difference between two probability distributions. Its basic idea is to map the samples in two distributions into a feature space, and then compare the average performance difference between the two distributions in the feature space. Let p and q be two independent probability distributions. If F denotes the mathematical expectation of $f(x)$ to x under the probability density, then the distribution difference between p and q can be represented by (15).

$$MMD(F, p, q) \triangleq \sup_{f \in F} ||E_{x \sim p}[f(x)] - E_{y \sim q}[f(y)]|| \tag{15}$$

The size of the MMD value is related to the similarity between the two probability distributions. Specifically, when the value of MMD is close to 0, the difference between the mean values of the two distributions, p and q, in the feature space is very small—that is, their distributions in the feature space are very similar. On the contrary, when the value of the MMD is large, the difference between the mean values of p and q in the feature space is large—that is, their distribution in the feature space is quite different, and they can be considered to be dissimilar. Therefore, the value of MMD can be used as an index to measure the similarity between two probability distributions. The closer to 0, the higher the similarity, and the farther away from 0, the lower the similarity.

(3) Cosine similarity

Cosine similarity is a common method used to characterize the difference in the direction of continuous data distribution. It measures the similarity between two vectors

based on the angle between them. Suppose x_{ij} is the jth element of the ith training sample, $\hat{x}_{ij}$ is the corresponding generated sample, and m is the number of elements of a single sample. The cosine similarity calculation formula of the training samples and the generated samples is shown in (16).

$$\cos(x_i, \hat{x}_i) = \frac{\sum_{j=1}^{m} x_{ij}\hat{x}_{ij}}{\sqrt{\sum_{j=1}^{m} x^2{}_{ij}}\sqrt{\sum_{j=1}^{m} \hat{x}_{ij}^2}} \tag{16}$$

The value range of cosine similarity is between -1 and 1: the closer the value is to 1, the more similar the two vectors are; the closer the value is to -1, the more opposite the two vectors are; and a value of 0 indicates that there is no similarity between the two vectors.

(4) MSE

The mean square error (MSE) can calculate the mean square error between the predicted value and the real value of the model. This experiment uses this index to evaluate the prediction accuracy and generalization ability of the regression module. The smaller the MSE value is, the closer the prediction result is to the real value and the better the prediction effect is. Additionally, the MSE is also a crucial indicator to measure the generalization ability of the model, which helps to evaluate the ability of the model to adapt to new data. If the MSE value of the model is small, it shows that the model has strong predictive ability for new data and has good generalization. The calculation formula is as follows:

$$\text{MSE} = \frac{1}{\text{n}}\sum_{i=1}^{n} (y_{i,p} - y_i)^2 \tag{17}$$

where n is the number of samples in the test sample set, and y_i and $y_{i,p}$ represent the value corresponding to the ith sample in the test sample set and the model prediction value, respectively.

3.2. Experiments and Analysis

Firstly, the optimal hyper-parameter configuration of the algorithm is selected through experiments. On this basis, the experimental results are presented and analyzed in detail. The experiment is divided into two main parts: in the first part, the similarity between the generated samples and the original samples of the proposed model, VAE, CGAN, and CVAE-GAN models is compared by using evaluation indicators such as the JS divergence, MMD, and cosine similarity, so as to evaluate the performance of the new model; the second part is used to further verify the effectiveness of the proposed algorithm by comparing the regression prediction test error of the original data with the data processed by different sample generation algorithms.

3.2.1. Model Structures and Parameters

The generation module in the IRGAN algorithm is composed of a long short-term memory neural network (LSTM) and a fully connected layer. When processing sequence data, LSTM can capture the dependencies in the sequence through its own loop structure. A fully connected layer neural network is added after the LSTM layer, and its main purpose is to balance the dimension. The correction module is composed of an agent and a correction network. The essence of both the correction module and the regression module is also a fully connected neural network. The discriminant module is composed of a convolutional neural network (CNN) and a fully connected layer. The CNN uses convolutional operations, which makes the discriminant module have strong feature extraction abilities. The output layer of the discriminant module is a fully connected layer, which integrates the eigenvalues extracted by the previous network and flattens the elements on each feature surface to perform the final discriminant task. In addition, in the process of model training, the discriminant ability of the default discriminant module is stronger than the data generation

ability of the generation module so that the discriminant module can guide the generation module to learn in the right direction, and thus, the usual practice is to set different learning rates for the discriminant module and the generation module, respectively. That is, the learning rate of the discriminant module is set to be greater than that of the generation module to speed up the convergence speed of the discriminant module. In this experiment, the learning rate of the generated module is set to 0.001, and the learning rate of the discriminant module is set to 0.002. The batch size is 50 with generation and discriminant module alternating training. After about 100 cycles, the training process tends to be stable.

We set the fully connected layer, long short-term memory neural network, convolutional neural network, deconvolution neural network, hidden layers, and the number of neurons per layer to FC, LSTM, CNN, DCNN, N_HL, and e-f-g. The model structures and parameters involved in this section are shown in Table 1.

Table 1. Structures and parameters of all models in the experiment.

	Datasets	Model Structure
Generation module	Airfoil Self-Noise	N_HL = 3, LSTM:15-20-5
	Abalone	N_HL = 3, LSTM:20-25-8
	Yacht Hydrodynamics	N_HL = 3, LSTM:10-15-7
	Concrete Compressive Strength	N_HL = 3, LSTM:15-20-9
Discriminant module	Airfoil Self-Noise	N_HL = 2, CNN:15-20
	Abalone	N_HL = 2, CNN:20-32
	Yacht Hydrodynamics	N_HL = 2, CNN:10-15
	Concrete Compressive Strength	N_HL = 2, CNN:20-25
Correction module	Airfoil Self-Noise	Agent: FC:5-15-5 Correction network: FC:2-20-2
	Abalone	Agent: FC:8-25-8 Correction network: FC:2-30-2
	Yacht Hydrodynamics	Agent: FC:7-10-7 Correction network: FC:2-15-2
	Concrete Compressive Strength	Agent: FC: 9-20-9 Correction network: FC:2-25-2
Regression module	Airfoil Self-Noise	FC:4-6-1
	Abalone	FC:7-10-1
	Yacht Hydrodynamics	FC:6-8-1
	Concrete Compressive Strength	FC:8-10-1

In order to apply the VAE, CGAN, and CVAE-GAN models to imbalanced regression problems, this paper changes the generator target value to continuous value according to the characteristics of regression samples.

3.2.2. Model Parameter Comparison Experiment

Firstly, from the perspective of the parameters of the model itself, this paper discusses the influence of the value of the parameter, α, in the IRGAN algorithm on the performance of the model. Through a large number of experiments, six typical values of $\alpha = 0$, $\alpha = 0.2$, $\alpha = 0.4$, $\alpha = 0.8$, $\alpha = 1.2$, and $\alpha = 1.5$ were selected as the discussion objects. In this section, the similarity between the generated samples and the real samples is used to evaluate the generation ability of the model under each parameter. Specifically, we calculate the JS divergence, MMD, and cosine similarity between the generated samples and the real samples under different parameters. In order to avoid the error caused by the randomness of the single experimental results, we carried out ten experiments, and the experimental results were processed by the mean method. The results are shown in Table 2.

Table 2. Similarity measurement results of models under different parameters.

(**a**) Airfoil Self-Noise			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
IRGAN (α = 0)	0.4649	0.8394	0.7264
IRGAN (α = 0.2)	0.4533	0.7736	0.7431
IRGAN (α = 0.4)	0.4228	0.6676	0.7731
IRGAN (α = 0.8)	**0.3272**	**0.5361**	**0.8124**
IRGAN (α = 1.2)	0.3598	0.5498	0.8013
IRGAN (α = 1.5)	0.3842	0.6514	0.7905
(**b**) Abalone			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
IRGAN (α = 0)	0.5021	0.5978	0.8358
IRGAN (α = 0.2)	0.4882	0.4051	0.8453
IRGAN (α = 0.4)	0.4585	0.3812	0.8624
IRGAN (α = 0.8)	**0.3833**	**0.2647**	**0.9069**
IRGAN (α = 1.2)	0.4097	0.3087	0.8876
IRGAN (α = 1.5)	0.4308	0.3567	0.8721
(**c**) Yacht Hydrodynamics			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
IRGAN (α = 0)	0.6409	1.4909	0.5417
IRGAN (α = 0.2)	0.6193	1.4065	0.5703
IRGAN (α = 0.4)	0.5916	1.3018	0.6054
IRGAN (α = 0.8)	**0.2099**	**1.0472**	**0.6489**
IRGAN (α = 1.2)	0.3132	1.1832	0.6286
IRGAN (α = 1.5)	0.4670	1.2155	0.6121
(**d**) Concrete Compressive Strength			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
IRGAN (α = 0)	0.4263	0.5567	0.8185
IRGAN (α = 0.2)	0.4131	0.5521	0.8379
IRGAN (α = 0.4)	0.3989	0.4838	0.8537
IRGAN (α = 0.8)	**0.2291**	**0.3438**	**0.8901**
IRGAN (α = 1.2)	0.2342	0.3583	0.8728
IRGAN (α = 1.5)	0.3206	0.4491	0.8694

The experimental results show the following.

1. The JS divergence and MMD are negatively correlated with cosine similarity. The smaller the JS divergence and MMD, the larger the cosine similarity. According to the above introduction, we know that the higher the distribution similarity of the two variables, the closer the JS divergence value and MMD are to 0 and that the cosine similarity value is close to 1, and the angle tends to 0. It shows that the closer the generated samples are to the real samples, the better the model generation effect.
2. By comparing the experimental data in Table 2, we find that regardless of any value of α, the experimental results are better than when α takes 0. As mentioned above, when α takes 0, it is equivalent to not changing the loss function of the generation module. This shows that adding the Mahalanobis distance to the loss function of the generated module and considering the correlation between the data can improve the quality of the generated samples.

3. Comparing the experimental results under the five values of $\alpha = 0$, $\alpha = 0.2$, $\alpha = 0.4$, $\alpha = 0.8$, $\alpha = 1.2$ and $\alpha = 1.5$, it is found that as the value increases, the experimental effect gradually improves, but when it exceeds a certain value, the experimental effect begins to deteriorate. This shows that it is not enough to only consider the Mahalanobis distance between the generated data and the real data when generating samples. Because the goal of the generator is to generate samples closer to real data, it is more important to consider the difference between the generated data distribution and the real data distribution. Through a large number of experiments, we determined that when α value is 0.8, the samples generated by IRGAN achieved the best results in JS divergence, MMD, and cosine similarity. Based on this, $\alpha = 0.8$ is defaulted for model training in subsequent experiments.

3.2.3. Comparative Experiment of Different Models

After discussing the parameters of the model itself, we now compare the proposed model with other models. In this section, the similarity indicators between the generated samples and the real samples are used to evaluate the generation ability of each model. Specifically, we calculate the JS divergence, MMD, and cosine similarity between the generated samples and the real samples of each model. In order to avoid the error caused by the randomness of the single experimental results, we carried out ten experiments, and the experimental results were processed by the mean method. The results are shown in Table 3.

Table 3. Similarity measurement results of different models.

(**a**) Airfoil Self-Noise			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
CGAN	0.6895	1.2152	0.3412
VAE	0.5494	0.9054	0.4813
CVAE-GAN	0.4798	0.8829	0.5979
IRGAN	**0.3272**	**0.5361**	**0.8124**
(**b**) Abalone			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
CGAN	0.6948	0.9876	0.5497
VAE	0.6579	0.8734	0.6534
CVAE-GAN	0.6142	0.7521	0.7216
IRGAN	**0.3833**	**0.2647**	**0.9069**
(**c**)Yacht Hydrodynamics			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
CGAN	0.7968	1.6487	0.3981
VAE	0.7543	1.6243	0.4652
CVAE-GAN	0.6721	1.5621	0.5213
IRGAN	**0.2099**	**1.0472**	**0.6489**
(**d**) Concrete Compressive Strength			
Model	**Evaluation Metric**		
	JS	**MMD**	**Cosine Similarity**
CGAN	0.8856	0.9876	0.7210
VAE	0.7892	0.7521	0.7654
CVAE-GAN	0.6543	0.6409	0.7907
IRGAN	**0.2291**	**0.3438**	**0.8901**

By comparing with the classical algorithm, it is found that the CVAE-GAN increases the learning of sample distribution information on the basis of the CGAN, and uses the confrontation idea to optimize the generation effect on the basis of the VAE model. Therefore, the effect of the CVAE-GAN model is better than that of the VAE and CGAN. Based on these models, the IRGAN algorithm further integrates the optimization idea and introduces the guidance mechanism of the correction network to make the similarity and authenticity of the generated samples perform better. This is also evident from the table of the experimental results. Compared with the other three models, IRGAN performs lower in terms of the JS divergence and MMD values, and higher regarding cosine similarity, which further confirms the effectiveness and superiority of the algorithm in generating regression samples.

3.2.4. Regression Prediction Experiment

After proving the superiority of the proposed algorithm in data generation, the regression prediction effect is further verified by experiments. Specifically, we perform regression prediction on the original samples and the samples balanced by different generation algorithms. The data from each interval of the input dataset is divided into a training set and a test set according to the principle of 7:3, and the number of iterations is set to 100. In order to ensure the reliability of the experimental results, ten experiments were carried out on the original samples and the balanced samples, respectively, and all the test errors were averaged. The corresponding average error graph of the test dataset of the original samples and the balanced samples is shown in Figure 3.

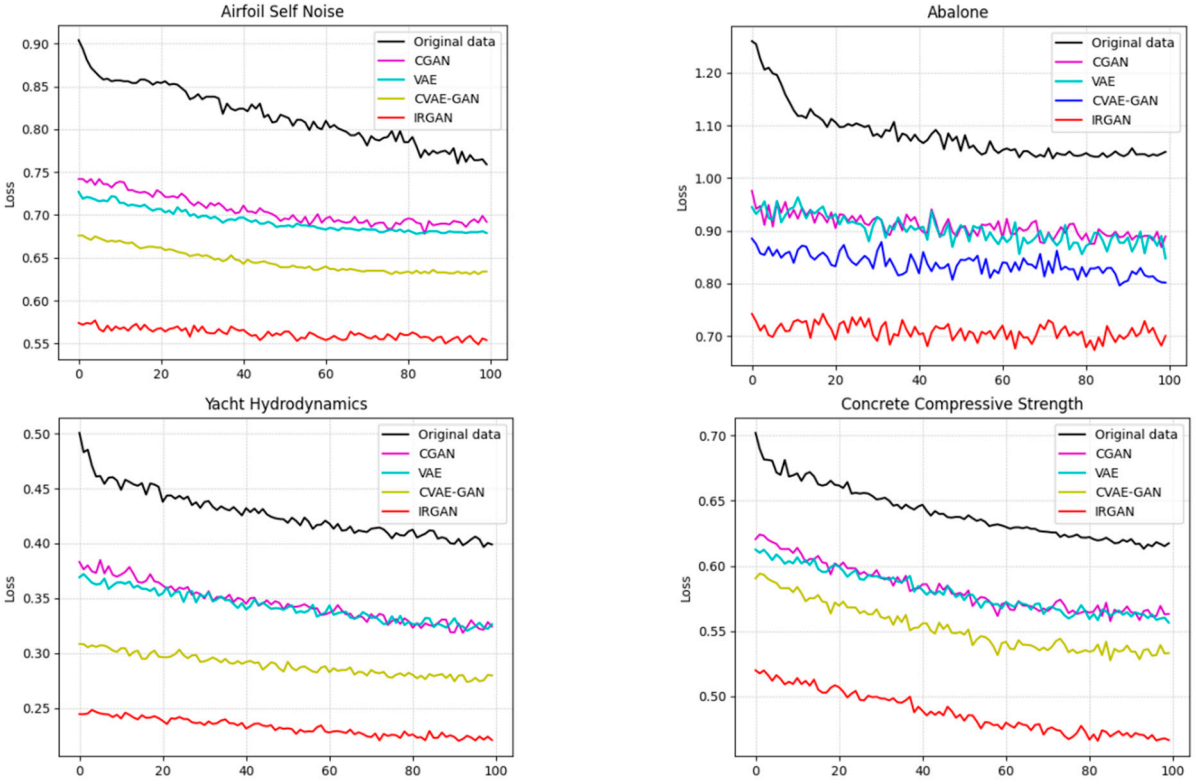


Figure 3. Test error curves of different models.

(1) By observing the regression error curve of the test set, it can be found that the error curve of the original data is above the test error of the balanced samples, that is, the test error of the balanced samples is significantly lower than that of the original data. This shows that the accuracy of regression prediction can be significantly improved by adding minority samples to balance the original dataset.

(2) From the test error curve, it can be seen that the comparison effect of the error curve is consistent with the previous analysis, and the CVAE-GAN algorithm is better than the VAE and CGAN models. However, the IRGAN algorithm designs the correction module to guide the generation module, and changes the loss function of the generation module, which enhances the similarity between the generated samples and the real samples, making it more suitable for the generation of regression samples. Therefore, the test error on the figure is the smallest.

(3) It can be seen from Figure 3 that the IRGAN model has good results on four imbalanced regression datasets in the fields of aerospace, biology, physics, and chemistry, which not only proves the effectiveness of the algorithm but also shows its wide applicability in many application fields.

4. Conclusions

Data imbalance is an important problem in machine learning. When the number of minority samples is much smaller than the number of majority samples, traditional machine learning algorithms often find it difficult to effectively identify minority samples, resulting in the poor accuracy and stability of prediction results. Therefore, we propose the IRGAN algorithm to solve the regression problem of data imbalance. The algorithm comprehensively uses four modules of generation, correction, discriminant, and regression to effectively generate high-quality minority samples to balance the original samples, thereby improving the accuracy of regression prediction. The effectiveness and practicability of the IRGAN algorithm are demonstrated by experimental verification in the fields of aerospace, biology, physics, and chemistry. Our research provides a new direction for solving the problem of imbalanced regression, but its limitation is that the datasets used all contain offline data. Therefore, future research can consider using a real-time environment. In this process, it is necessary to ensure the performance and improve the calculation speed of the algorithm as much as possible to meet the real-time requirements. This may require exploring more efficient algorithm implementations, optimizing model structures, or using hardware acceleration.

Author Contributions: Conceptualization, X.L. and H.T.; methodology, H.T.; software, X.L.; writing—original draft preparation, X.L.; writing—review and editing, H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Tianjin Research Innovation Project for Postgraduate Students (NO. 2022SKYZ348).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kaur, H.; Pannu, H.S.; Malhi, A.K. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 79. [CrossRef]
2. Ma, T.; Lu, S.; Jiang, C. A membership-based resampling and cleaning algorithm for multi-class imbalanced overlapping data. *Expert Syst. Appl.* **2024**, *240*, 122565. [CrossRef]
3. Tian, H.X.; Tian, C.Z.; Li, K.; Jia, W.A. Unbalanced regression sample generation algorithm based on confrontation. *Inf. Sci.* **2023**, *642*, 119157. [CrossRef]
4. Petinrin, O.; Saeed, F.; Salim, N. Dimension Reduction and Classifier-Based Feature Selection for Oversampled Gene Expression Data and Cancer Classification. *Processes* **2023**, *11*, 1940. [CrossRef]
5. Pei, X.; Su, S.; Jiang, L.; Chu, C.; Gong, L.; Yuan, Y. Research on rolling bearing fault diagnosis method based on generative adversarial and transfer learning. *Processes* **2022**, *10*, 1443. [CrossRef]
6. Yen, S.J.; Lee, Y.S. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Syst. Appl.* **2009**, *36*, 5718–5727. [CrossRef]
7. Zhai, Z. Auto-encoder generative adversarial networks. *J. Intell. Fuzzy Syst.* **2018**, *35*, 3043–3049. [CrossRef]
8. Liu, X.Y.; Wu, J.; Zhou, Z.H. Exploratory undersampling for class-imbalance learning. *IEEE Trans. Syst. Man Cybern. Part B-Cybern.* **2008**, *39*, 539–550.

9. Chawla, N.V.; Bowyer, K.W.; Hall, L.O. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
10. Seiffert, C.; Khoshgoftaar, T.M.; Van Hulse, J. Hybrid sampling for imbalanced data. *Integr. Comput. Aided Eng.* **2009**, *16*, 193–210. [CrossRef]
11. Guo, Y.; Chu, Y.; Jiao, B. Evolutionary dual-ensemble class imbalance learning for human activity recognition. *IEEE Trans. Emerg. Top. Comput. Intell.* **2021**, *6*, 728–739. [CrossRef]
12. Ganaie, M.A.; Hu, M.; Malik, A.K.; Tanveer, M. Ensemble deep learning: A review. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105151. [CrossRef]
13. Chawla, N.V.; Lazarevic, A.; Hall, L.O. SMOTEBoost: Improving prediction of the minority class in boosting. In Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Knowledge Discovery in Databases: PKDD 2003, Cavtat-Dubrssovnik, Croatia, 22–26 September 2003; Proceedings 7. Springer: Berlin/Heidelberg, Germany, 2003; pp. 107–119.
14. Sun, Y.; Kamel, M.S.; Wong, A.K. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognit.* **2007**, *40*, 3358–3378. [CrossRef]
15. Ren, Z.; Zhu, Y.; Liu, Z.; Feng, K. Few-shot GAN: Improving the performance of intelligent fault diagnosis in severe data imbalance. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 3516814. [CrossRef]
16. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
17. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 2672–2680.
18. Zhang, D.; Ma, M.; Xia, L. A comprehensive review on GANs for time-series signals. *Neural Comput. Appl.* **2022**, *34*, 3551–3571. [CrossRef]
19. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
20. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
21. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 214–223.
22. Bao, J.; Chen, D.; Wen, F. CVAE-GAN: Fine-grained image generation through asymmetric training. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2745–2754.
23. Wang, L.; Han, M.; Li, X. Review of classification methods on unbalanced data sets. *IEEE Access* **2021**, *9*, 64606–64628. [CrossRef]
24. Moniz, N.; Ribeiro, R.; Cerqueira, V.; Chawla, N. Smoteboost for regression: Improving the prediction of extreme values. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–3 October 2018; pp. 150–159.
25. Torgo, L.; Ribeiro, R.P.; Pfahringer, B.; Branco, P. Smote for regression. In Proceedings of the Portuguese Conference on Artificial Intelligence, Angra do Heroísmo, Portugal, 9–12 September 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 378–389.
26. Branco, P.; Torgo, L.; Ribeiro, R.P. Rebagg: Resampled bagging for imbalanced regression. In Proceedings of the Second International Workshop on Learning with Imbalanced Domains: Theory and Applications, Dublin, Ireland, 10–14 September 2018; pp. 67–81.
27. Yang, Y.; Zha, K.; Chen, Y.; Wang, H.; Katabi, D. Delving into deep imbalanced regression. In Proceedings of the International Conference on Machine Learning, Virtual Event, 18–24 July 2021; pp. 11842–11851.
28. Ren, J.; Zhang, M.; Yu, C.; Liu, Z. Balanced MSE for Imbalanced Visual Regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 21–24 June 2022; pp. 7926–7935.
29. Gavas, R.D.; Das, M.; Ghosh, S.K.; Pal, A. Spatial-SMOTE for handling imbalance in spatial regression tasks. *Multimed. Tools Appl.* **2024**, *83*, 14111–14132. [CrossRef]
30. Liu, F.; Dai, Y. Product processing quality classification model for small-sample and imbalanced data environment. *Comput. Intell. Neurosci.* **2022**, *2022*, 9024165. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.