PGNN: A Parallel Granular BP Neural Network for Classification

Xianyu Hou (✉ 416410794@qq.com )
Xiamen University of Technology

Yumin Chen
Xiamen University of Technology

Junwen Lu
Xiamen University of Technology

Keshou Wu
Xiamen University of Technology

Nianfeng Zeng
E-success Information Technology Co., Ltd

Research Article

Keywords: Granular computing, Granular representation, Granular neural network, Clustering granulation, Parallel processing

Posted Date: August 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1901755/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
PGNN: A Parallel Granular BP Neural Network for Classification

Xianyu Hou\textsuperscript{1*}, Yumin Chen\textsuperscript{1†}, Junwen Lu\textsuperscript{1†}, Keshou Wu\textsuperscript{1†} and Nianfeng Zeng\textsuperscript{2†}

\textsuperscript{1}College of Computer and Information Engineering, Xiamen University of Technology, 600 Ligong Rd., Xiamen City, 361024, China.
\textsuperscript{2}E-success Information Technology Co., Ltd, 82 Zhanhong Rd., Xiamen City, 361024, China.

\textsuperscript{*}Corresponding author(s). E-mail(s): 416410794@qq.com;
Contributing authors: ymchen@xmut.edu.cn; jwlu@xmut.edu.cn; kswu@xmut.edu.cn; 395373664@qq.com;
\textsuperscript{†}These authors contributed equally to this work.

Abstract

Deep learning models based on BP neural networks are widely used in classification and regression problems. However, they have a wide range of parameters and insufficient local and uncertain information representations. Granular computing has great uncertainty characterization capability and allows input patterns to be abstracted at higher levels. A parallel granular BP neural network is proposed by combining clustered granulation to reconfigure the BP neural network. First, the optimal samples are defined and used as the references for granulating. Features are granulated into granules. And samples are granulated into granular vectors. Secondly, the granule vector is input to the granule BP neural network, and the granules for classification are output after the granule BP forward calculation, and are reduced to the original classification results by the thresholding process. The granular BP neural network defines granular activation functions and a granular loss function to fit the input pattern of granular vectors while retaining the original fully connected neural network structure. The problem to be processed is extended into granules by granulation, and the structured characteristics of the granules make the granular BP neural network computable in parallel. Finally, compared with DNN, SVM,
PGNN: A Parallel Granular BP Neural Network for Classification

DT and NB, PGNN has better classification accuracy and improved computational efficiency and enhanced generalization performance.

Keywords: Granular computing, Granular representation, Granular neural network, Clustering granulation, Parallel processing

1 Introduction

Although significant progress has been made in pattern-based classification, it still faces challenges from humans. Due to the lack of intelligence of computer programs and algorithms, they still can’t handle better than humans when faced with complex tasks [1, 2]. A typical feature of human beings when solving problems is that they divide the problem into different levels of granularity when they encounter complex tasks while using strong memory and comparative ability to handle complex tasks [3]. Zadeh’s paper called "Fuzzy sets and information granularity" marked the birth of a fuzzy set theory. Zadeh argues that the concept of information granularity exists in many fields, such as interval operations in interval theory, uncertainty in control theory, etc. which are manifestations of information granularity [4, 5]. Based on fuzzy set theory, Lin published a paper called "Granular computing" in 1998, detailing the model of granular computing under binary relations, which marked the birth of granular computing [6].

Granular computing is a term with a broad connotation, which covers all theoretical and methodological studies on granules [7–9]. It is considered a collection of fuzzy information granulation, rough set theory, and interval computation [10, 11]. Granules are the most fundamental element in building a granular computational model [12]. By constructing an information granulation, a set of patterns is divided into a series of granules based on similarity, interval, and other features. From an applied perspective, granular computing is a general method for solving structured problems [13]. A key feature of granular computing is the ability to reconstruct input patterns at a higher level of abstraction. Information granules [14, 15] which are taken as the underlying entities of granular computing are pieces of knowledge that describe digital data at a higher level of abstraction.

Methods and techniques of information granulation etc. play important roles in many fields, and most of them propose various clustering and classification algorithms based on the aggregation and aggregation properties of granules [16–19]. In [20], Hu proposed a granulated way based on the neighboring relations, which realizes granulometric calculation in the real number space. In [21], Chen proposed a fuzzy granulation based on single-atom features, which combined with convolutional operations to optimize the weights and obtain good classification performance. In [22], Li proposed a stochastic fuzzy granularity decision tree based on information granules, which combines the ideas of clustering and classification. The method finds the optimal center
by clustering, then classifying fuzzy granules by fuzzy trees, and introducing information entropy and information gain [23, 24] to avoid the over-fitting problem. One can directly innovate to define the ruleset of granules or combine it with other algorithms to predict the results based on the properties of the granules themselves. But classifiers built based on global sample granulation have a limitation. When global granules are input into a neural network, it increases the complexity of the network, the process of building the classifier consumes a lot of time, and the accuracy of the results is reduced due to the complexity of the classifier model [25, 26]. Therefore, constructing a granular classifier with strong classification performance and high efficiency is an area worthy of further study.

The BP neural network is the fundamental model of deep learning, which has been used in pattern recognition problems widely. In supervised learning, the perceptron takes a sample vector as input, obtains the predicted classes by limiting the model output, and optimizes the model by finding decision bounds that minimize misclassification [27]. The BP neural network is a multi-layer perceptron. It can achieve a powerful ability to handle nonlinear data through nonlinear activation functions.

The uncertainty quantification of fuzzy sets and the classification power of neural networks, and their application to pattern recognition problems, have attracted great interest in infusion methods of fuzzy sets and neural networks [28, 29]. In many applications of fuzzy sets, the uncertainty of the input data can be quantified and represented as intervals. Interval analysis is the basis of fuzzy sets and can explain the tolerance of a sample. In the papers [30–32], authors propose a parallel fuzzy max-min neural network that divides the large problem into individual sub-problems, and the final model gains a great advantage in terms of time efficiency.

The construction of granules using equivalence relations is a global granulation method, which has a high time and space complexity [33, 34]. To solve this problem, the mean samples of each class are introduced to construct the optimal samples. The optimal sample is used as the reference sample, and all samples are granulated against the reference samples. The optimal sample is local concerning the whole samples, so this is a local granulation method. The parallel granular BP neural network (PGNN) is then constructed based on the granular form of the input, which is structured in parallel. The model solves the time-consuming problem encountered when the data is expanded in granular form and retains a better accuracy on top of that. Experiments show that the parallel model proposed in this paper achieves good generalization on small samples as well. And the input of the model is changed from a one-dimensional pattern space to a high-dimensional pattern space, where the input samples are transformed into the granular vectors. Combining this feature with parallel processing provides an efficient way of processing.

The PGNN has the following innovative features:

1. The input samples can be either one-dimensional raw samples or high-dimensional granular vectors.
2. An optimal sample granulation method is innovated to facilitate sample expansion and improve model generalization performance.

3. Parallel computation is possible, and making model training more efficient.

The rest of the paper is organized as follows. Section II gives a detailed description of clustering methods and BP neural networks. Section III defines the granule and granular operations. Section IV gives a detailed description of the PGNN, focusing on the model structure of the parallel network and the use of the granulation and degranulation layers in the network. Section V gives an experimental analysis that verifies the advantages of the model in several aspects.

2 Preliminaries

2.1 Related work

2.1.1 Cluster methods

Various clustering algorithms are the most commonly used analysis methods in machine learning. In clustering algorithms, such as the k-means algorithm, the samples are divided into k clusters in the sample space, and each cluster has a center that represents the class of the cluster \([35, 36]\). Combined with the idea of similarity, by finding the center of each class in the sample, and then calculating the similarity between classes based on the class center, we can maximize the intra-class similarity and minimize the inter-class similarity.

2.1.2 BP neural network

The BP neural network is a basic model of deep learning. Unlike the traditional perceptron, each node in the middle layer has a functional operation relationship with the nodes in the lower layer, and the parameters of the network are trained by the BP algorithm and loss function to finally obtain the parameter values that make the error close to the minimum. The BP neural network has nonlinear mapping capability and good adaptive and generalization capabilities. A classical BP neural network architecture consists of three layers, i.e. the input layer, the hidden layer, and the output layer, as shown in Figure 1.

2.2 Optimal sample granulation

Let the sample space be \(U = (X, C)\), where \(X = \{x_1, x_2, ..., x_n\}\) is the training sample set; The set of conditional features is \(C = \{c_1, c_2, ..., c_m\}\). Given a single sample \(x \in X\) and a single feature \(c \in C\), \(v(x, c) \in [0, 1]\) denotes the normalized value of sample \(x\) on feature \(c\).

**Definition 1**

Given a sample space \(U = (X, C)\), the sample \(\bar{x}_j\) is the mean sample of a certain class of sample sets \(X_j \in X\). For any sample \(x \in X_j\), the Euclidean
distance between the sample $x$ and the mean sample $\bar{x}_j$ is:

$$
d(x, \bar{x}_j) = \sqrt{\sum (x_i - \bar{x}_j)^2}.
$$

**Definition 2**
The sample $x_i$ satisfying $\min\{d(x_1, \bar{x}_j), d(x_2, \bar{x}_j), ..., d(x_n, \bar{x}_j)\}$ is defined as the optimal sample, denoted as the reference sample $p_k$. The selected optimal sample set is used as the reference sample set, denoted as:

$$
P = \{p_1, p_2, ..., p_k\} \subseteq X,
$$

where $k$ is the number of classes.

**Definition 3**
Given a sample space $U = (X, C)$, for any sample $x \in X$ and a reference sample $p \in P$ and any single feature $c \in C$, the similarity between $x$ and $p$ with respect to feature $c$ is defined as follows:

$$
r = S_c(x, p) = 1 - | v(x, c) - v(p, c) |,
$$

where $v(x, c)$ denotes the value of $x$ on the feature $c$.

**Definition 4**
Given a sample space $U = (X, C)$, for any sample $x \in X$ and a reference sample set $P$, and any single feature $c \in C$, then $x$ is granulated on the feature $c$ in the reference sample and the conditional granule formed is defined as:

$$
g_c(x) = \{g_c(x)_{j=1}^k = \{r_j\}_{j=1}^k = \{r_1, r_2, ..., r_k\},
$$

where $r_j \in [0, 1]$. The granule consists of a nucleus, $g_c(x)$ is called a granule and $g_c(x)_j = r_j$ is called the $j$th nucleus.
Definition 5
Let $U = (X, C)$ be the sample space, and for any sample $x \in X$, any conditional feature subset $A \subseteq C$, let $A = \{a_1, a_2, ..., a_m\}$, the granular vector of $x$ on the feature subset $A$ is defined as:

$$G_A(x) = (g_{a_1}(x), g_{a_2}(x), ..., g_{a_m}(x))^T,$$

where $g_{a_m}(x)$ is the granule of sample $x$ on feature $a_m$. For computational convenience, the feature set is labeled with integers, and the granular vector can be expressed as:

$$G(x) = (g_1(x), g_2(x), ..., g_m(x))^T.$$

2.3 Granular operations
The optimal samples are used as the reference samples and then the training samples are granularized with the reference samples, which will construct the granules and granular vectors. This section defines the correlation metrics and operations of granules and establishes the operational basis of the granular neural network.

Definition 6
Let $g_a(x) = \{s_j\}_{j=1}^k$ and $g_b(x) = \{t_j\}_{j=1}^k$ be two granules of sample $x$ on feature a and feature b respectively. The addition, subtraction, multiplication and division operations of the two granules are defined as follow:

$$g_a(x) + g_b(x) = \{s_j + t_j\}_{j=1}^k;$$

$$g_a(x) - g_b(x) = \{s_j - t_j\}_{j=1}^k;$$

$$g_a(x) * g_b(x) = \{s_j * t_j\}_{j=1}^k;$$

$$g_a(x) / g_b(x) = \{s_j / t_j\}_{j=1}^k.$$

Definition 7
Let $g_a(x) = \{s_j\}_{j=1}^k$ and $g_b(y) = \{t_j\}_{j=1}^k$ be two granules of sample $x$ on feature a and feature b respectively. The addition, subtraction, multiplication and division operations of the two granules are defined as follow:

$$g_a(x) + g_b(y) = \{s_j + t_j\}_{j=1}^k;$$

$$g_a(x) - g_b(y) = \{s_j - t_j\}_{j=1}^k;$$

$$g_a(x) * g_b(y) = \{s_j * t_j\}_{j=1}^k;$$
\[ g_a(x) / g_b(y) = \{s_j/t_j\}_{j=1}^{k}. \]

The result of two granular operations is a granule. **Definition 6** is defined for operations on different granules on different sets of features for the same sample, while **Definition 7** is applied to operations on granule on the same set of features for different samples.

**Definition 8**

Let the two granular vectors be \( G(x) = (g_1(x), g_1(x), ..., g_m(x))^T \) and \( G(y) = (g_1(y), g_1(y), ..., g_m(y))^T \). Then the dot product of these two granular vectors is defined as follow:

\[ G(x) \cdot G(y) = G(x)^T G(y) = g_1(x) * g_1(y) + g_1(x) * g_1(y) + ... + g_m(x) * g_m(y). \]

The result of the dot product of two granular vectors is one granule. Therefore, it is possible to construct a weight granular vector by having the sample granular vector perform a dot product operation with the weight granular vector. Let the granular vector be \( G(x) = (g_1(x), g_2(x), ..., g_m(x), 1)^T \) and the weight granular vector be \( W = (w_1, w_2, ..., w_m, b)^T \), then its dot product is expressed as:

\[ W \cdot G(x) = w_1 * g_1(x) + w_2 * g_2(x) + ... + w_m * g_m(x) + b. \]

**Definition 9**

Given a granule \( g(x) = \{r_j\}_{j=1}^{k} \), the function on the granule is defined as follow:

\[ f(g(x)) = f(\{r_j\}_{j=1}^{k}) = \{f(r_1), f(r_2), ..., f(r_k)\}. \]

**Definition 10**

Let the granule be \( g(x) = \{r_j\}_{j=1}^{k} \). Then the sigmoid, tanh, ReLU, and ELU of the granule are defined as follow respectively:

\[
\begin{align*}
    f(g(x)) &= \{1 / (1 + e^{-r_j})\}_{j=1}^{k}; \\
    f(g(x)) &= \{e^{r_j} - e^{-r_j} / (e^{r_j} + e^{-r_j})\}_{j=1}^{k}; \\
    f(g(x)) &= \{max(0, r_j)\}_{j=1}^{k}; \\
    f(g(x)) &= \left\{ \begin{array}{ll}
    r_j, & r_j > 0 \\
    a(r_j - 1), & r_j \leq 0
    \end{array} \right\}_{j=1}^{k}.
\end{align*}
\]

The activation function introduces the nonlinear factors for the neurons, allowing the neural network to approximate any nonlinear functions at will so that the neural network can be applied to the nonlinear models. The granular activation function is a function on granules, whose input is a granule and whose output is also a granule.
3 Proposed Methods

3.1 Model Structure

In this paper, we propose a parallel granular BP neural network (PGNN) model. The model is an extension of the BP neural network with a three-layer structure: the granular layer, the working layer, and the threshing layer. Figure 2 shows the structure of the model.

After the sample set $X = \{x_1, x_2, x_3, ..., x_n\}$ is input into the granular layer, single feature granulation is performed on sample $X$. The reference sample set for granulation is $P = \{p_1, p_2, p_3, ..., p_k\}$. The classes of the samples were encoded with One-Hot. According to Definition 1 and Definition 2, a similarity measure function is used to select the optimal sample that is defined as follows:

$$p_{\text{opt}} = x_i, \text{when } d(x_i, \bar{x}) = \min \{d(x_i, \bar{x})\}_{i=1}^n,$$

where $\bar{x}$ is the mean of the samples in the same class and $d$ is the Euclidean distance between $x$ and $\bar{x}$.

The output result of granular layer is a set of granular vectors $M_P = \{G(x_1), G(x_2), G(x_3), ..., G(x_n)\}$, where each $G(x_i)$ is the granulated result of sample $x_i$. According to Definition 3, the similarity-based granulation function is as follows:

$$S_c(x, p) = 1 - | v(x, c) - v(p, c) |.$$  

After granulation, the granular vectors having the same class are divided into different sets of classes, which are the inputs to the subsequent model.

The middle layer of the model is $k$ parallel workers, which are three-layer BP neural networks adapted to the input in the form of granular vectors, they receive the set of inputs split from the granular layer and generate a series of granular vectors. The workers are defined by the following function:

$$W(G(x_i), \theta, \{f_1(G(x_i), f_2(G(x_i))\}),$$
where $G(x_i)$ is the conditional granular vector output from the granulation layer, $\theta$ is a user-specified hyperparameter that indicates that there are $\theta$ reference samples that influence the complexity of this workstation, and $f_j(G(x_i))$ is the activation function of the network.

Subsequently, the granular vectors generated by all workers are output to the threshing layer, corresponding to the threshing function for $x$ and $p$ on the single feature $c$ as:

$$T_c(x, p) = \begin{cases} 1 - r_c + v(p, c), v(x, c) - v(p, c) \geq 0 \\ r_c - 1 + v(p, c), v(x, c) - v(p, c) < 0 \end{cases}.$$  \(4\)

After threshing, the two-dimensional granular vector $G_t = \{g_{t1}, g_{t2}, ..., g_{tk}\}$ is generated, where the $k$ represents that there are $k$ reference samples. Each $g_{ti}$ represents the prediction result corresponding to the $i$th reference sample, and the result corresponds to the class output of the sample. Each granular class function is defined as follows:

$$g_{ci} = \begin{cases} \max(g_{ti}) = \max(g_{ti}^j) \\ g_{ti}^j = 0, \text{others} \end{cases},$$  \(5\)

where $j$ is the $j$th nucleus in granule $g_{ti}$.

After determining the class of each granule by the granular class function, the output resultant granule becomes the class granule and the decision granular vector becomes the class granular vector $G_c = \{g_{c1}, g_{c2}, ..., g_{ck}\}$. This changes the model decision output which can be compared with the original class, and then eventually the class of the sample output is determined by majority election. This class output can be either a fuzzy or a clear value. After determining the class, the fuzzy output is obtained by averaging the class granule, which can be used as the probabilistic output of the corresponding class; or directly assigning a value of 1 to the nucleus of the corresponding class, which is the encoding of the class.

### 3.2 Learning algorithms

The model in this paper uses gradient descent to solve the granular vector of the weights in the PGNN. Multiple conditional values of each sample are granulated to form a granular vector, which corresponds to multiple input granules; multiple decision values of each sample are granulated to form a granular vector as well, which corresponds to multiple output granules. Therefore, the PGNN model is constructed. The learning process of the model is described by a gradient descent method as follows:

<table>
<thead>
<tr>
<th>Algorithm 1 PGNN Learning algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: The training set $T = (x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)$, where $x_i$ is an $n$-dimensional feature vector; $y_i$ is a $l$-dimensional decision vector; $x \in X \subseteq \mathbb{R}^n$, $y_i \in \mathbb{R}^l$, $i = 1, 2, \ldots, t$; learning rate is $\eta(0 &lt; \eta \leq 1)$.</td>
</tr>
</tbody>
</table>
Output: The weights \( w \), \( v \) and the granular threshold \( \theta \) of PGNN.

1. According to \textbf{Function 1}, \( k \) samples are constructed from the training set \( T \) as the optimal reference set \( P = \{ p_1, p_2, \ldots, p_k \} \), where the number of samples \( k \) corresponds to the user-specified value of \( \theta \);
2. Inputting the training set \( T \) and the reference set \( P \) into the granular layer and performing single-feature granulation on the samples, and the output of the granulation is the granulation matrix \( G_T = \{ G(x_1), G(x_2), \ldots, G(x_t) \} \), where the granular vector \( G(x_i) = \{ g_1(x_i), g_2(x_i), \ldots, g_n(x_i) \} \); granules \( g_j(x_i) = \{ r_1, r_2, \ldots, r_k \} \) is a set;
3. Constructing the working layer with the granular vector \( G_T \) as the sample input to the overall, each input being \( n \)-dimensional and the output being \( l \)-dimensional (corresponding to the class of the sample), and randomly initialize the weights and thresholds of the working layer;
4. If the number of iterations is less than the maximum number of iterations, perform steps 5, 6 and 7, otherwise, go to 8;
5. For \( x \in X \), the conditional granular vector \( G(x) = \{ g_1(x), g_2(x), \ldots, g_n(x) \} \) is input into the parallel working layer for forwarding calculation; after the working layer calculation, the output decision granular vector \( G_d = \{ g_{d1}, g_{d2}, \ldots, g_{dl} \} \);
6. The decision granular vector \( G_d = \{ g_{d1}, g_{d2}, \ldots, g_{dl} \} \) is input into the threshing layer and perform the threshing operation to form the class granules \( G_c = \{ g_{c1}, g_{c2}, \ldots, g_{cl} \} \);
7. Updating the weights \( w \) and \( v \) of PGNN;
8. After each \( x \in X \) has performed step 5 and step 6, calculate the granular loss function: \( E = \frac{1}{kn} \sum_{i=1}^{n} \sum_{j=1}^{l} (c_{xi} - g_{cj})^2 \), if \( E > \zeta \) go to step 4, otherwise go to step 8;
9. Outputting the weights \( w \) and \( v \) of PGNN and the reference set \( P \).

For each training sample, the learning algorithm inputs a conditional granular vector based on similarity and outputs a threshing class result. Each sample corresponds to \( l \) class regression values, which can form a class granule.

4 Experiments

In this paper, several UCI datasets are used for experiments, and the specific information of the datasets is shown in Table 1. To test the effectiveness of the algorithm, the convergence of the PGNN is first analyzed; Secondly, several linear and nonlinear datasets will be selected. Compared with DNN based on the BP algorithm and several classical classification algorithms to test the nonlinear performance of the PGNN. Finally, the article compares the temporal performance of PGNN and DNN.
Table 1 Datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Dimension</th>
<th>Class</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>4</td>
<td>3</td>
<td>150</td>
</tr>
<tr>
<td>Wine</td>
<td>13</td>
<td>3</td>
<td>177</td>
</tr>
<tr>
<td>Dry Bean</td>
<td>16</td>
<td>7</td>
<td>13612</td>
</tr>
<tr>
<td>SPECT</td>
<td>22</td>
<td>2</td>
<td>267</td>
</tr>
<tr>
<td>Heart</td>
<td>13</td>
<td>2</td>
<td>270</td>
</tr>
</tbody>
</table>

Fig. 3 Classification Error

4.1 Convergence analysis

We use the full training set for granulation and training to test whether the granular classification error converges and compare the convergence with the DNN algorithm. The granular classification error is adopted as the granular mean square error:

$$g - MSE = \frac{1}{kn} \sum_{i=1}^{n} \sum_{j=1}^{l} (c_{xi} - g_{cj})^2.$$  

The conventional classification error using mean square error as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (c_{xi} - y_{prd_i})^2.$$  

The experiment was conducted on the Heart dataset, and the experimental results are shown in Figures 3 and 4. In Figure 3, the horizontal axis indicates the number of iterations, and the vertical axis indicates the classification error. In Figure 4, the horizontal axis indicates the number of iterations, and the vertical axis indicates the subtraction of errors before and after the iterations.

As can be seen from Figures 3-4, the error of PGNN converges to near the very small point around 800 epochs as the original error, and the convergence process is also smoothly decreasing. The experiment shows that the
PGNN converges with the same classification training process as the BP neural network.

4.2 Effect of granular reference set

4.2.1 Effect of optimal sample

In the granular classification model, training is first performed to obtain a weight granular vector. Then, the sample to be tested is granulated into a granular vector, which is operated with the weight granular vector. Finally, the prediction result is obtained, which is a granule instead of a real number. Suppose there are n training samples, whose \( Y \) values be \( Y = \{y_1, y_2, \ldots, y_n\} \). After the sample \( x \) is granulated and predicted, a predicted granular vector is obtained, set as \( G_{dg} = \{g_{dg1}, g_{dg2}, \ldots, g_{dgk}\} \). Each \( g_{dg} \) is converted to \( g_c \) by the threshing layer, and then the final output class value is determined using a voting strategy, expressed by the following equation:

\[
y_t = VIM(g_{c1}, g_{c2}, \ldots, g_{ck}).
\]

For each class granule \( g_{ci}(0 \leq i \leq k) \), its mean square error is compared with the true class, and then the gradient descent algorithm is used to reduce the error. This allows a comparison with DNN. In this experiment, we compared three PGNN models (corresponding to different ways of sample selection) with DNN, where PGNN denotes the selection of optimal samples, PGNN_center denotes the selection of sample centers and PGNN_random denotes random selection, and obtained ten training results respectively. The Train_acc is used to denote the training set accuracy and the Test_acc denotes the test set accuracy. The results are shown in Table 2 and Figures 5-6 below. The values in Table 2 represent the highest accuracies in the training set and the test set.

According to the experimental results in Table 2, PGNN has the highest accuracy on the training set of Iris, Wine, and Dry Bean respectively; while on the training set of SPECT, both random selection and sample center obtain
the highest accuracy with an accuracy of 0.9623; meanwhile, on the Heart dataset, DNN performs the best, and PGNN is second. And in the testing set, the advantage of PGNN’s high generalizability comes out, it can be seen that the best accuracy is obtained on the Iris, Wine, Dry Bean, and SPECT. It is about two percentage points higher than random selection on Iris and Wine, while the highest difference in accuracy is obtained on the test sets of Dry Bean and SPECT reached 0.0227 and 0.0558, respectively. On the last dataset, i.e. Heart, DNN performed better than the other three algorithms, but only about 0.009 higher than PGNN.

Table 2 Results of different methods on the dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Train acc</th>
<th>Test acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>PGNN</td>
<td>0.9997</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>PGNN-center</td>
<td>0.9667</td>
<td>0.9667</td>
</tr>
<tr>
<td></td>
<td>PGNN_Random</td>
<td>0.9416</td>
<td>0.9667</td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td>0.9833</td>
<td>1.0000</td>
</tr>
<tr>
<td>Wine</td>
<td>PGNN</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>PGNN-center</td>
<td>0.9787</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>PGNN_Random</td>
<td>1.0000</td>
<td>0.9722</td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Dry Bean</td>
<td>PGNN</td>
<td>0.9425</td>
<td>0.9401</td>
</tr>
<tr>
<td></td>
<td>PGNN-center</td>
<td>0.9111</td>
<td>0.9174</td>
</tr>
<tr>
<td></td>
<td>PGNN_Random</td>
<td>0.9235</td>
<td>0.9258</td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td>0.9336</td>
<td>0.9346</td>
</tr>
<tr>
<td>SPECT</td>
<td>PGNN</td>
<td>0.9481</td>
<td>0.8889</td>
</tr>
<tr>
<td></td>
<td>PGNN-center</td>
<td>0.9623</td>
<td>0.8519</td>
</tr>
<tr>
<td></td>
<td>PGNN_Random</td>
<td>0.9623</td>
<td>0.8333</td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td>0.9378</td>
<td>0.8331</td>
</tr>
<tr>
<td>Heart</td>
<td>PGNN</td>
<td>0.9120</td>
<td>0.9259</td>
</tr>
<tr>
<td></td>
<td>PGNN-center</td>
<td>0.8843</td>
<td>0.8889</td>
</tr>
<tr>
<td></td>
<td>PGNN_Random</td>
<td>0.8889</td>
<td>0.8704</td>
</tr>
<tr>
<td></td>
<td>DNN</td>
<td>0.9336</td>
<td>0.9346</td>
</tr>
</tbody>
</table>

Figure 5 shows the results of 10 training results on the training set (the data are similarly divided 10 times). The blue line representing PGNN performs significantly better than the other three algorithms on the Iris and Dry Bean datasets, and the accuracy is the best at most times. The training results of DNN on Heart and Wine are better and generally higher than the other three algorithms. Finally, the training results on SPECT data are more confusing, except for times 1, 3, 4, and 7, where DNN performs better than the other three algorithms, at other times, each of the four algorithms has its advantages.

Turning to the results of the test set (Figure. 6), the PGNN algorithm performs significantly better than the other algorithms on the Iris, Wine, SPECT, and Dry Bean datasets, and achieves the highest accuracy at most times. In the Heart, the four algorithms have the best performance at all times.

Based on these 10 results, an interesting phenomenon is that the PGNN algorithm is more adaptive to unknown samples than the DNN algorithm.
Defining the adaptability of the algorithms is as follows:

\[ S = \frac{1}{n} \sum_{i=1}^{n} | t_i - \bar{t} |, \]  

(7)

where \( \bar{t} \) is the mean value of the prediction accuracy. Then for the four algorithms, the adaptabilities are evaluated as follows in Table 3.

As we can see, the adaptation of PGNN is better than DNN on the Iris, Heart, and Wine, while on SPECT and Dry Bean DNN is a little higher, considering the randomness of the experiments, the adaptation of these two algorithms is the same. Although PGNN does not always perform the best in terms of adaptability, the combined ability of PGNN is the best when combined with the prediction results of PGNN in these five datasets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Iris</th>
<th>Heart</th>
<th>Wine</th>
<th>SPECT</th>
<th>Dry Bean</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGNN</td>
<td>0.0140</td>
<td>0.0353</td>
<td>0.0117</td>
<td>0.0260</td>
<td>0.0040</td>
</tr>
<tr>
<td>PGNN_random</td>
<td>0.0367</td>
<td>0.0333</td>
<td>0.0444</td>
<td>0.0326</td>
<td>0.0093</td>
</tr>
<tr>
<td>PGNN_center</td>
<td>0.0267</td>
<td>0.0282</td>
<td>0.0333</td>
<td>0.0260</td>
<td>0.0056</td>
</tr>
<tr>
<td>DNN</td>
<td>0.0334</td>
<td>0.0554</td>
<td>0.0167</td>
<td>0.0240</td>
<td>0.0039</td>
</tr>
</tbody>
</table>

According to the above analysis, although PGNN has no obvious advantage over DNN in the training set, PGNN generally performs better than DNN in the test set, with an average higher accuracy of about 0.02. And combined
with the adaptation analysis, PGNN has a higher comprehensive ability than DNN on Iris, Heart, and Wine. In general, PGNN has better generalization performance than the other three models.

### 4.2.2 Effect of the number of optimal samples

For the optimal sample, it can select the sample with the smallest distance with the mean vector within each class. After the initial selection, the option to continue selecting a new best-fit reference sample in the sample space excluding the selected reference sample is still available. But the more reference samples are not better, because, for each worker, different reference samples will affect the performance of the workers. Selecting better reference samples will have better generalization performance after network aggregation. However, if a poor reference sample is selected, it will lead to a decrease in network performance. Based on this problem, different values of the optimal number of reference samples \( \theta \) are chosen. The experiments are conducted on the SPECT dataset, and the worker in the hidden layer was a three-layer BP neural network with a single layer of 20 nodes. The experimental results are shown in Figure 7.

As shown in Figure 7, with the accuracy of DNN as the baseline, the accuracy of PGNN is higher than or equal to the baseline when \( \theta = \{1, 2, 3, 4, 5\} \), and the best accuracy is obtained at \( \theta = 2 \). This is because, at \( \theta = 2 \), the number of reference samples is equal to the number of sample classes, and the selected reference samples are the most suitable samples within the two classes so that the generalization performance of the constructed model is the best.
It can be seen that the model is sensitive to the selection of the $\theta$ value, so it is necessary to adjust the $\theta$ value in the proposed model.

### 4.3 Classified performance comparison

Eighty percent of the dataset for this experiment was used for training and the other twenty percent of the samples were used for validation. The classification effectiveness of the model is evaluated using the training set accuracy and the test set accuracy. It is also compared with classical classification algorithms such as Support Vector Machine (SVM), Gaussian naive Bayes (NB), and Decision Tree (DT). Five experiments were performed for each algorithm and the mean and the range of errors are shown in Table 4.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DNN</th>
<th>SVM</th>
<th>NB</th>
<th>DT</th>
<th>PGNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>0.989±0.010</td>
<td>0.967±0.023</td>
<td>0.975±0.025</td>
<td>0.967±0.023</td>
<td>1.000±0.000</td>
</tr>
<tr>
<td>Wine</td>
<td>1.000±0.000</td>
<td>0.989±0.011</td>
<td>1.000±0.000</td>
<td>0.934±0.001</td>
<td>1.000±0.000</td>
</tr>
<tr>
<td>Dry Bean</td>
<td>0.925±0.003</td>
<td>0.919±0.014</td>
<td>0.899±0.005</td>
<td>0.894±0.001</td>
<td>0.931±0.004</td>
</tr>
<tr>
<td>SPECT</td>
<td>0.813±0.003</td>
<td>0.935±0.000</td>
<td>0.895±0.004</td>
<td>0.973±0.000</td>
<td>0.889±0.003</td>
</tr>
<tr>
<td>Heart</td>
<td>0.856±0.037</td>
<td>0.790±0.021</td>
<td>0.870±0.013</td>
<td>0.784±0.023</td>
<td>0.861±0.023</td>
</tr>
<tr>
<td>Average</td>
<td>0.917±0.011</td>
<td>0.920±0.014</td>
<td>0.927±0.009</td>
<td>0.910±0.010</td>
<td>0.936±0.006</td>
</tr>
</tbody>
</table>

According to Table 4, it can be seen that PGNN performs best on the testing sets of Iris, Wine, and Dry Bean, with the accuracies of 1.000, 1.000, and 0.931 respectively. The DT algorithm performs the best on the SPECT dataset, with an accuracy of 0.973. On the Heart dataset, PGNN performs better, with much higher accuracy than the SVM and DT, and slightly lower than the NB algorithm. Taken together, the average accuracy of PGNN on the five datasets is higher than that of DNN, SVM, NB, and DT algorithms, and the fluctuation range is smaller than other algorithms, so PGNN has stronger generalization and stability.
4.4 Time performance comparison of PGNN and DNN

PGNN is a parallel model that enhances the generalization performance of the model by processing the output of the aggregated model, and because of this feature of parallel processing, PGNN and DNN are different in terms of time performance. The time efficiency of PGNN is about $\frac{1}{2}$ that of DNN on Iris, SPECT, and Heart with small sample size, and about $\frac{1}{3}$ that of DNN on Dry Bean and Wine. Therefore, PGNN has a large improvement in time efficiency compared with DNN. A comparison of the time efficiency of PGNN and DNN is shown in Table 5.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PGNN</th>
<th>DNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>2.808</td>
<td>5.978</td>
</tr>
<tr>
<td>Wine</td>
<td>3.964</td>
<td>10.384</td>
</tr>
<tr>
<td>Dry Bean</td>
<td>121.858</td>
<td>407.021</td>
</tr>
<tr>
<td>SPECT</td>
<td>3.453</td>
<td>6.384</td>
</tr>
<tr>
<td>Heart</td>
<td>6.375</td>
<td>13.430</td>
</tr>
</tbody>
</table>

5 Conclusions

The values involved in the traditional classification model are real numbers. Starting with the study of granulation of samples, this paper proposes a new classification model of the ensemble by defining the form of the granule and the granular vector. First, the single feature optimal sample granulation method is introduced to construct information granules and granular vectors in the information system, and the size metric and operation rules of granular vectors are defined. Then the granular vectors and their associated operation rules are further proposed. By applying the granular vectors, the PGNN model is proposed which has a working layer. The working layer is a parallel neural network adapted to the input in the form of granular vectors. Second, the PGNN is optimally solved to improve the generalization performance and efficiency of the model by aggregating the output of the workstation. Finally, experimental analysis is conducted to show the feasibility and effectiveness of the PGNN. In future work, the study of granular affiliation function, further reduce the sample size by fuzzy affiliation function, and then increase the number of samples by more reference samples to further improve the generalization performance of the model, and also introduce a pruning step to reduce the size of the model, which will further improve the accuracy and efficiency of the model.

References


