A Gegenbauer Neural Network with Regularized Weights
Direct Determination for Classification

Jie He, Tao Chen, Zhijun Zhang, Senior Member, IEEE
School of Automation Science and Engineering,
South China University of Technology
Guangzhou, China

Abstract—Single-hidden layer feed forward neural networks
(SLFNs) are widely used in pattern classification problems, but
a huge bottleneck encountered is the slow speed and poor perfor-
mance of the traditional iterative gradient-based learning algo-
rithms. Although the famous extreme learning machine (ELM)
has successfully addressed the problems of slow convergence, it
still has computational robustness problems brought by input
weights and biases randomly assigned. Thus, in order to over-
come the aforementioned problems, in this paper, a novel type
neural network based on Gegenbauer orthogonal polynomials,
tered as GNN, is constructed and investigated. This model could
overcome the computational robustness problems of ELM, while
still has comparable structural simplicity and approximation
capability. Based on this, we propose a regularized weights
direct determination (R-WDD) based on equality-constrained
optimization to determine the optimal output weights. The R-
WDD tends to minimize the empirical risks and structural risks
of the network, thus to lower the risk of over fitting and improve
the generalization ability. This leads to a final GNN with
R-WDD, which is a unified learning mechanism for binary and
multi-class classification problems. Finally, as is verified in the
various comparison experiments, GNN with R-WDD tends to
have comparable (or even better) generalization performances,
computational scalability and efficiency, and classification ro-
bustness, compared to least square support vector machine (LS-
SVM), ELM with Gaussian kernel.

Index Terms—Extreme learning machine (ELM), least square
support vector machine (LS-SVM), regularization neural net-
work, Gegenbauer neural network (GNN), Gegenbauer ortho-
gonal polynomials.

I. INTRODUCTION

Researches on artificial feed-forward neural networks
(FNNs) have become increasingly active and popular, for
it is one of the most powerful tools in artificial intelligence
field. What’s more, it has been widely studied and applied in
a great number of engineering fields, such as the genetic
research [1], the prediction of rainfall [2], the analysis of
biological emotion [3] and the humanoid robots control [4].
Because FNN has formidable approximation capability to
approximate complex and nonlinear functions directly through
training on samples, a computational model containing system
characteristics could be built up more conveniently without
knowing the exact information of the objective system by
training on input samples.

For the development of this algorithm, many researchers
have explored the abilities of multi-layer FNNs [5–7].
Leshno has proved that multi-layer FNNs with activation
functions are able to approximate any function, including
continuous functions and non-sequence functions [8]. For the
past few years, the single-hidden layer feed-forward neural
networks (SLFNs) model has been proposed by Huang et al.
[9–11], which can approximate functions with zero errors
by learning $S$ diverse samples. SLFNs have at most $N$ hidden
neurons and the hidden layer biases.

For traditional SLFNs, the parameters need to be tuned
iteratively to ensure the optimality of the network and frequent
parameters tuning will lead to a complicated relationship
between the parameters of different layers. As a solution, the
famous Back-propagation (BP) algorithm based on gradient-
descend has been proved to be the most effective tool for
iterative learning algorithm in SLFNs [12]. However, the
BP-type training algorithm has some inherent weaknesses
resulting in inevitable slow operations and poor performances,
such as slow convergence [13], local minima [14].

To address this problem, Huang et al. propose and investi-
igate a simple but effective learning algorithm for the SLFNs,
called extreme learning machine (ELM). It has been observed
that the ELM has unparalleled advantages such as extremely
training speed, and excellent generalization performance [9–
11]. Unlike the traditional BP algorithm, ELM did not pay
much attention to the input weights and the neuron biases, and
the input weights and hidden layer biases can be arbitrarily
assigned [15, 16].

Although the ELM enjoys its promising merits in applica-
tion prospects, it still has several shortcomings that ELM could
not easily overcome. The first one is that because of due to
the mechanisms of randomly assigning weights, the output
of ELM fluctuates and the performances could be unstable.
The second problem ELM involving is that the hidden layer
matrix $H$ could be not full column rank or even ill-conditioned
when the input weights and biases are generated randomly,
which may incur serious computational robustness problems
in obtaining optimal output weights [17].

Therefore, in order to overcome the aforementioned weak-
nesses of BP-type FNNs and ELM, in this paper, we focus on
utilizing Gegenbauer orthogonal polynomials as activation
functions, and constructing orthogonal polynomials FNN with
the simplicity, such as fixed input weights and biases. Such
settings have been proved effective in [13, 19]. Further, to
avoid lengthy training and determine optimal output weights
directly, a method called weights direct determination (WDD)
based on pseudo-inverse in previous works [13, 19] is firstly
reviewed. However, the WDD only focuses on empirical risk
minimization, which might easily lead to a too complicated
model and higher risks of over-fitting [20, 21]. Also, in
the previous works, the proposed networks are single output networks, which may be not consistent with the facts that FNNs with multiple output are the most effective way to tackle multi-classification problems.

In this paper, a novel FNN termed as Gegenbauer neural network (GNN) is constructed and investigated based on Gegenbauer polynomials series and polynomials function approximation. Then, based on equality-constrained optimization, a regularized WDD (R-WDD) is proposed with a regularized term to prevent the model from being too complicated. Also, based on the number of network output nodes, a unified learning mechanism of R-WDD for binary and multi classification problems is given. Besides, with regard to computational scalability and efficiency, two different solutions to R-WDD are given, depending on the scale of the problems and the dimensionality of hypothesis feature space. Specifically, the main contributions of this paper lie in the following facts.

- The main merit ELM algorithm is its extremely fast training speed. In this paper, the proposed GNN has similar calculation complexity to ELM, mainly involving an inversion of a square matrix. Thus, it has the comparable excellent training speed as ELM.
- Compared with ELM, the proposed algorithm is stable and has lower computational risks, since the input weights are fixed and the tensor products of Gegenbauer orthogonal polynomials are utilized as activation functions.
- Experiments on several real-world data sets have illustrated that the GNN has comparable or even better generalization performance in binary classification problems and multi-classification problems, compared with least square support vector machine (LS-SVM) with Gaussian kernel and ELM with Gaussian kernel.
- Experiments have shown that GNN has great computational scalability and efficiency, still performs very well when there only exists very few neurons in the network.
- Experiments on data sets contaminated with random noises prove the better classification robustness of GNN.

The remainder of the article is organized as follows. Section II presents and analyzes the theoretical basis of the GNN, and the GNN with single output nodes (SOGNN) and the GNN with multiple output nodes (MOGNN) are constructed and discussed respectively. In Section III the proposed R-WDD is discussed and analyzed based on equality constrained-optimization. Section IV presents several comparison experiments on LS-SVM, ELM and GNN with R-WDD. And the final the remarks are given in Section V.

II. THEORETICAL BASIS FOR GNN

In this section, the fundamentals of Gegenbauer Polynomials are first presented. Then, the theoretical basis of Gegenbauer polynomials series (GPS) for multivariate function approximation is briefly analyzed. Furthermore, based on the analysis, the GNN model is constructed and investigated.

A. Fundamentals of Gegenbauer Polynomials

Definition 1. The general expression of Gegenbauer polynomials can be defined as below [22, 23]:

\[
g_n^\lambda(x) = \sum_{k=0}^{n} \frac{1}{k!(n-k)!} \frac{(2\lambda)n(2\lambda + n)k}{(\lambda + 1/2)^k} (x - 1)^k, n \in \mathbb{N} \tag{1}
\]

where \( \lambda \) is a parameter of the polynomials that satisfies \( \lambda > 0 \), \( n \in \mathbb{N} \) is the degree of the Gegenbauer polynomials and \( (\lambda)_n = \lambda(\lambda+1)\cdots(\lambda+k-1) = \Gamma(\lambda+k)/\Gamma(\lambda) \) and \( \Gamma(\cdot) \) is a Gamma function. Equivalently, Gegenbauer polynomials can also be defined by the recurrence relations below [23]:

\[
\begin{align*}
g_n^\lambda(x) &= 1, \quad n = 0, \\
g_n^\lambda(x) &= 2\lambda x, \quad n = 1, \\
g_n^\lambda(x) &= \alpha_n x g_{n-1}^\lambda(x) - \beta_n g_{n-2}^\lambda(x), \quad n \geq 2
\end{align*}
\]

where \( \alpha_n = 2(n+\lambda-1)/n \) and \( \beta_n = (n+2\lambda-2)/n \).

According to orthogonal polynomials theories [24, 25], Gegenbauer polynomials are orthogonal with respect to weighted function \( \rho(x) \geq 0 \) within the interval \( x \in [0,1] \), and have the orthogonal properties as below:

\[
[g_i^\lambda(x), g_j^\lambda(x)] = \int_0^1 \rho(x)g_i^\lambda(x)g_j^\lambda(x)dx = \left\{ \begin{array}{ll} 0, & i \neq j \\
\int_0^1 \rho(x)(g_i^\lambda(x))^2 dx > 0, & i = j \end{array} \right. \tag{3}
\]

where \( \rho(x) \) satisfies that \( \int_0^1 \rho(x)dx > 0 \) and there exists \( \int_0^1 x^n \rho(x)dx > 0, n \in \mathbb{N} \).

B. GPS for Multivariate Function Approximation

First, a function approximation theorem, i.e., Weierstrass approximation theorem, is presented for the further discussion.

Theorem 1. (Weierstrass Approximation Theorem)

Let \( \phi \in C(U, \mathbb{R}) \) with \( U = [a, b] \). Then there exists a sequence of polynomials \( p_n(x) \) converging uniformly to \( \phi(x) \) on \( [a, b] \):

\[
|\phi - p_n(x)| < \xi \tag{4}
\]

with \( \forall \xi > 0 \). And it can also be generalized to a multivariate case. Set \( S_m = \{ (x_1, \ldots, x_m) \in \mathbb{R}^k | a < x_t < b, t = 1, 2, \ldots, m \} \). Let \( \phi \in C(S_m, \mathbb{R}) \). Then there is a sequence of polynomials \( p_{n_1,\ldots,n_m}(x_1, \ldots, x_m) \) converging uniformly to \( \phi(x) \) on \( S_m \):

\[
|\phi - p_{n_1,\ldots,n_m}(x_1, \ldots, x_m)| < \xi \tag{5}
\]

with \( \forall \xi > 0 \).

Proof: The proof of famous Weierstrass approximation theorem has been completed in [26]. Thus, the proof of Theorem 1 is completed.

Based on the Gegenbauer polynomials and theories of function approximation, i.e., Theorem 1 and polynomial interpolation [19, 27, 28], a univariate continuous function \( \phi(x) \) defined on \( x \in [0,1] \) can be approximated by the \( N \)-order (or to say the total number of polynomials used) GPS:
\[ \phi(x) = p_N(x) + \xi_\phi(N) \]
\[ = \sum_{i=0}^{N} \mu_i g_i^N(x) + \xi_\phi(N), \]  
(6)

where \( \mu_i \) is the coefficient corresponding to \( g_i^N(x) \), \( p_N(x) \) denotes \( N \)-order GPS and \( \xi_\phi(N) \) is the error function related to \( N \). According to approximation theories \cite{11, 12} and Theorem 2, the error function could converge to zero when \( N \) is large enough, which could be expressed as:

\[ \lim_{N \to +\infty} \xi_\phi(N) = 0. \]  
(7)

For the function approximation of GPS in \( m \)-dimension, we have the following lemma \cite{29, 30}.

**Lemma 1.** Let \( D = [0,1]^{m \times 1} \) and \( \phi(x) \) is a continuous real-valued function with \( m \) variables on \( D \) (i.e., \( x = [x_1, x_2, \ldots, x_m]^T \in [0,1]^{m \times 1} \)). Then, \( B_n(\phi; x) \), the generalized multivariate Bernstein polynomials of \( \phi(x) \), converges uniformly to \( f(x) \) and could be structured as follows:

\[ B_n(\phi; x) = \sum_{k_i=0}^{n} \cdots \sum_{k_m=0}^{n} \phi \left( \frac{k_1}{n}, \ldots, \frac{k_m}{n} \right) \prod_{i=1}^{m} b_{n,k_i}(x_i), \]  
(8)

where \( b_{n,k_i}(x_i) = C_{n}^{k_i} x_i^{k_i} (1-x_i)^{n-k_i} \) with \( t = 1, 2, \ldots, m \), and \( C_{n}^{k_i} \) is binomial coefficient (i.e., \( C_{n}^{k_i} = n!/[k_i!(n-k_i)!] \)).

According to Lemma 1 for a target function \( \phi(x) \), we have the following result:

\[ \lim_{n \to +\infty} B_n(\phi; x_1, \ldots, x_m) = \phi(x_1, \ldots, x_m). \]  
(9)

Then, on the basis of polynomial interpolation theory \cite{29}, we give the following theorem.

**Theorem 2.** For a real-valued function with \( m \) variables continuous on \( [0,1]^{m \times 1} \), it could be approximated by GPS uniformly.

**Proof:** Without loss of generality, in light of Eq. 6 and 7, \( b_{n,k_i}(x_i) \) can be approximated by GPS as follows:

\[ b_{n,k_i}(x_i) = \sum_{i=0}^{\infty} \vartheta_{t,i,k_i} g_i^{\lambda}(x_i), \]  
(10)

where \( \vartheta_{t,i,k_i} \) denotes the coefficient corresponding to \( g_i^{\lambda}(x_i) \).

Then, in light of the lemma given above, we could substitute the Eq. 10 into Eq. 8:

\[ \phi(x) = \sum_{k_i=0}^{n} \cdots \sum_{k_m=0}^{n} \varphi_\phi \prod_{t=1}^{\infty} \sum_{i=0}^{\infty} \vartheta_{t,i,k_i} g_i^{\lambda}(x_i) + \xi_\phi(n) \]
\[ = \sum_{k_i=0}^{n} \cdots \sum_{k_m=0}^{n} \varphi_\phi \prod_{t=1}^{n} \sum_{i=0}^{n} \vartheta_{t,i,k_i} g_i^{\lambda}(x_i) + \tilde{\xi}_\phi(n) \]
\[ = \sum_{k_i=0}^{n} \cdots \sum_{k_m=0}^{n} \varphi_\phi \prod_{t=1}^{m} \sum_{i=0}^{m} \vartheta_{t,i,k_i} g_i^{\lambda}(x_i) + \tilde{\xi}_\phi(n) \]
\[ = L \sum_{t=1}^{L} w_l G_l^{\lambda}(x) + \tilde{\xi}_\phi(n), \]  
(11)

where \( n_t \) is the total amount of Gegenbauer polynomials utilized to approximate \( b_{n,k_i}(x_i) \), \( \xi_\phi(n) \) and \( \tilde{\xi}_\phi(n) \) is the residual error functions related to function \( \phi(x) \), \( c_{k_1, \ldots, k_m} \) is the coefficient for \( \prod_{t=1}^{n} g_l^{\lambda}(x_t) \), \( G_l^{\lambda}(x) \), termed as the Gegenbauer Elementary function (GEF) in this paper, is the \( m \)-order tensor product of the one-dimensional Gegenbauer polynomials, which could be expressed as:

\[ G_l^{\lambda}(x) = G_{k_1, \ldots, k_m}^{\lambda}(x) = g_{k_1}^{\lambda}(x_1) \cdots g_{k_m}^{\lambda}(x_m), \]
\[ w_l \] is the weights for \( G_l^{\lambda}(x) \), and \( L = \prod_{t=1}^{n} n_t \) is the total amount of Gegenbauer polynomials tensor product (i.e., \( \{ G_l^{\lambda}(x) \} \)) utilized to approximate \( \phi(x) \).

Besides, according to Eq. 11 and Eq. 7, the approximation residual error function of \( \phi(x) \) using GPS would converge to zero:

\[ \lim_{n, n_t \to +\infty} \tilde{\xi}_\phi(n) = 0. \]  
(12)

Thus, the proof of GPS for multivariate function approximation is thus completed.

**Note** that the \( n, n_t, \ldots, n_t \) should be large enough to guarantee the approximation capability of the GPS. Therefore, according to Theorem 2 a series of GEFs \( \{ G_l^{\lambda}(x) \} \), sorted by graded lexicograph \cite{13, 14}, with optimal weights \( w_l \) could minimize the residual error function \( \xi_\phi(n) \) and best approximate the function \( \phi(x) \).

**C. Network Architecture**

Based on the analysis in Section II-B the SOGNN and MOGNN are hereby constructed and investigated. The output patterns of both forms are then illustrated.

1) **SOGNN Model:** According to the theoretical analysis above, the architecture of SOGNN is constructed. As is shown in Fig. II, the network architecture of SOGNN consists of three layers, including input layer, hidden layer and output layer. Let \( x = [x_1, x_2, \ldots, x_m] \) denotes the \( m \)-dimensional input vector, and all of the input data should be normalized to \([0,1]\). In this model, the \( m \) neurons in input layer and the single neuron in output layer are both chosen linear in our model. Besides, the activation functions in hidden layer are a series of GEFs \( \{ G_l^{\lambda}(x) \mid l = 1, 2, \cdots, L \} \), and \( L \) denotes the total amount of
hidden neurons. Thus, the output of SOGNN in this case can be expressed as follow:

\[ \psi(x) = \sum_{l=1}^{L} w_l G_l^T(x) = G(x)w, \quad (13) \]

where \( G(x) = [G_1^T(x), G_2^T(x), \ldots, G_L^T(x)] \) denotes the Gegenbauer activation vector with respect to \( x \), \( G(x) \) actually maps the \( m \)-dimensional input space to the \( L \)-dimensional hidden layer feature space, and \( w = [w_1, w_2, \ldots, w_L]^T \in \mathbb{R}^L \) denotes the connecting weights vector.

If the number of samples is \( S \), their outputs can be written in matrix form, i.e.,

\[ \Psi(x) = GW, \quad (14) \]

where \( \Psi(x) = [\psi(x_1), \psi(x_2), \ldots, \psi(x_s)]^T \) and

\[ G = \begin{bmatrix} G_1(x_1) & G_2(x_2) & \cdots & G_L(x_s) \\
G_1^T(x_1) & G_2^T(x_2) & \cdots & G_L^T(x_s) \\
\vdots & \vdots & \ddots & \vdots \\
G_1^T(x_s) & G_2^T(x_s) & \cdots & G_L^T(x_s) \end{bmatrix} \in \mathbb{R}^{S \times L}, \]

Based on Theorem 2 (specifically, Eq. (11) and (12)), with \( L \) large enough and optimal weights, \( \psi(x) \) can best approximate the target function \( \phi(x) \). Also, in this model, all the input weights are fixed to be 1, and the threshold values for all neurons are fixed to be 0.

Notably, the aforementioned parametric settings can effectively simplify the network architecture, lower the computational complexity, make it more convenient for future implementation on hardware, and have been proved effective in previous works [18], [19].

Even more, such parametric simplification, on one hand, maintains the approximation capability and the optimality of the network since the network (shown in Fig. 1) with the aforementioned parametric settings well matches the theoretical analysis in Eq. (14) and (13). On the other hand, such fixed parametric settings could also improve the robustness of the model when faced with disturbance in input. According to [32], [33], when the input weights and hidden layer biases are randomly assigned, the robustness could be poor, since the changes of the output weights matrix sometimes could be very large when input disturbance is encountered, which will result in increasing risk of network.

Further, with activation functions being orthogonal elementary functions, our model prevents the hidden activation matrix \( G \) from being not full column rank and ill-conditioned, according to orthogonal polynomials theory [34]. Also, with \( G^T G \) being invertible, the direct computing of \( (G^T G)^{-1} \) can be easily obtained. Notably, this sets the basis of the computational robustness of the proposed method in Section III-B [35].

2) MOGNN Model: As is shown in Fig. 2 the MOGNN model is constructed based on SOGNN. To be more specific, the MOGNN with \( K \) outputs consists of \( K \) SOGNN. The output of MOGNN is \( K \)-dimensional vector, where \( \psi(x) = [\psi(x)_1, \psi(x)_2, \ldots, \psi(x)_K] \) and the \( k \)th output of MOGNN \( \psi(x)_k \) is the output of the \( k \)th SOGNN. The parametric settings is the same as the settings of SOGNN. The output of \( k \)th output node is given below:

\[ \psi(x)_k = \sum_{l=1}^{L_k} w_{lk} G_l^T(x) = G(x)w, \quad (15) \]

III. PROPOSED REGULARIZED WEIGHTS DIRECT DETERMINATION

In this section, the detailed explanation of our proposed R-WDD for GNN is presented. First, the original WDD method is reviewed, after which the main drawbacks of it are discussed. Then, based on equality-constrained optimization, a method minimizing the empirical risks and structural risks with better generalization ability is provided, and the analysis of it is carried out in cases of two forms of GNN, respectively. Finally, the unified R-WDD for GNN weights determination under different cases are given.

A. Review of the Original Weights Direct Determination

According to the previous works in [18], [19], for a given GNN model, the objective loss function of WDD is:

\[ \text{minimize: } \| \Psi - \Phi \|^2 = \| GW - \Phi \|^2. \quad (16) \]

where \( \Phi = [\phi(x_1), \phi(x_2), \ldots, \phi(x_s)]^T \) is the label of \( s \) samples.

With WDD, the solution of the optimal output weights can be calculated by the following equation:

\[ w = (G^T G)^{-1} G^T \Phi = G^T \Phi, \quad (17) \]

where \( (G^T G)^{-1} G^T \) is the Moore-Penrose generalized inverse of activation matrix \( G \), also termed as \( G^+ \). According to [36], there are several methods to calculate the Moore-Penrose generalized inverse of a matrix: orthogonal projection method, iterative method and singular values decomposition (SVD).

Since the WDD directly calculates the least-square solution involving an inversion of a matrix, the speed of calculation can be guaranteed. Further, according to the previous works [18], [19], the networks equipped with WDD have demonstrated
outstanding performances, such as excellent approximation ability, easy implementation.

However, although WDD method has aforementioned merits, it still has a main drawback. As is shown Eq. (16), the WDD only focuses on empirical risk minimization and no measure is taken to prevent the model hypothesis from being too complicated, which might easily lead to an over-fitting model [20], [21].

B. Equality-constrained Optimization Problems for GNN

Due to the aforementioned drawbacks of the original WDD method, in this paper, our model does not only tend to seek the smallest training error but also seek the smallest norm of output weights. According to the famous Barlett’s theory [21], a good model with generalization ability should be the one could reach a best trade-off between the empirical risk and the structural risk. Generally, the empirical risk is the training error of the neural network, which could be represented by $\ell_2$-norm, i.e., $\| \xi \|^2 = \| \Psi - \Phi \|^2$; the structural risk is the $\ell_2$-norm of output weights, i.e., $\| w \|^2$, which denotes the complexity of the model [37]. Thus, the optimization object and the mathematical model could be formulated as below:

$$\text{minimize: } \| Gw - \phi \|^2_2 \text{ and } \| w \|^2_2. \quad (18)$$

According to the ridge regression theory [38], it would make solution much stabler and tending to have better generalization ability since a positive value added to the diagonal of $(G^T G)^{-1} G^T$ or $G^T(GG^T)^{-1}$. Also, according to the statistical learning theory [33], [39], the real risk of a model should be the sum of the empirical one and the structural one, and these two kinds of risk should be considered together in discussions. Since the proposed GNN has two forms (i.e., SOGNN and MOGNN), in the following, the solution for each form would be discussed respectively. After that, it can be seen the solution for SOGNN is actually one specific case of MOGNN. Thus, these two solutions are finally unified, which leads to the final solution of R-WDD.

1) SOGNN for Classification: Based on the analysis in Section II-C, the SOGNN can approximate any continuous target function in the variables space. For classification problems, the expected output of SOGNN can be close enough to the target class labels in the corresponding regions. Thus, the objective function of proposed R-WDD for SOGNN can be described as below:

$$\text{minimize: } \frac{\gamma}{2} \| w \|^2_2 + \frac{1}{2} \sum_{s=1}^{S} \xi_s^2,$$

subject to: $\sum_{l=1}^{L} w_l G_{l}^{\alpha}(x_s) - \phi_s - \xi_s = 0, s = 1, 2, \ldots, S,$

where the weight factor $\gamma$ is the regularizer. By tuning the values of $\gamma$, the proportion of the empirical risk and the structural risk can be adjusted for reaching the best trade-off. Based on KKT theorem [40], to solve the R-WDD optimization problem for SOGNN is equivalent to solve the following Lagrangian equation:

$$L(w, \xi, \alpha) = \frac{\gamma}{2} \| w \|^2_2 + \frac{1}{2} \sum_{s=1}^{S} \xi_s^2$$

subject to: $\sum_{k=1}^{K} \sum_{s=1}^{S} \alpha_{s,k} G(x_s)w - \phi_{s,k} + \xi_{s,k} = 0, s = 1, 2, \ldots, S, k = 1, 2, \ldots, K,$

where $\alpha_{s,k} = \frac{\alpha_{s,1}}{\alpha_{s}}, \alpha_{s,2}, \ldots, \alpha_{s,K}$ is the target vector corresponding to sample $s$, $\xi_s = [\xi_{s,1}, \xi_{s,2}, \ldots, \xi_{s,K}]$ is the error vector, $w$ is a matrix of $L \times K$, where $w_{k}$ is the vector of the output weights corresponding to $k$th output node and $\| \cdot \|_F$ is Frobenius norm.

Similar to SOGNN, the Lagrangian equation for R-WDD optimization for MOGNN is:

$$L(w, \xi, \alpha) = \frac{\gamma}{2} \| w \|^2_2 + \frac{1}{2} \sum_{s=1}^{S} \xi_s^2$$

subject to: $G(x_s)w - \phi_s + \xi_s = 0, s = 1, 2, \ldots, S,$

where $\phi_s = [\phi_{s,1}, \phi_{s,2}, \ldots, \phi_{s,K}]$ is the target vector corresponding to sample $s$, $\xi_s = [\xi_{s,1}, \xi_{s,2}, \ldots, \xi_{s,K}]$ is the error vector, $w$ is a matrix of $L \times K$, where $w_{k}$ is the vector of the output weights corresponding to $k$th output node and $\| \cdot \|_F$ is Frobenius norm.

2) MOGNN for Classification: According to Section II-C, the MOGNN is actually a generalized form of the SOGNN. For $K$-classification problems, the number of the output nodes of MOGNN should be $K$, and when the original class label of sample is $k$, the expected output of MOGNN should be a $K$ dimensional vector and the $k$th component of it should be close to 1, while others should be close to $-1$. Thus, the objective function of proposed R-WDD for MOGNN can be described as below:

$$\text{minimize: } \frac{\gamma}{2} \| w \|^2_2 + \frac{1}{2} \sum_{s=1}^{S} \xi_s^2$$

subject to: $G(x_s)w - \phi_s + \xi_s = 0, s = 1, 2, \ldots, S,$

where $\phi_s = [\phi_{s,1}, \phi_{s,2}, \ldots, \phi_{s,K}]$ is the target vector corresponding to sample $s$, $\xi_s = [\xi_{s,1}, \xi_{s,2}, \ldots, \xi_{s,K}]$ is the error vector, $w$ is a matrix of $L \times K$, where $w_{k}$ is the vector of the output weights corresponding to $k$th output node and $\| \cdot \|_F$ is Frobenius norm.

Similar to SOGNN, the Lagrangian equation for R-WDD optimization for MOGNN is:

$$L(w, \xi, \alpha) = \frac{\gamma}{2} \| w \|^2_2 + \frac{1}{2} \sum_{s=1}^{S} \xi_s^2$$

subject to: $G(x_s)w - \phi_s + \xi_s = 0, s = 1, 2, \ldots, S,$

where $\alpha_{s,k} = \frac{\alpha_{s,1}}{\alpha_{s}}, \alpha_{s,2}, \ldots, \alpha_{s,K}$ is the target vector corresponding to sample $s$, $\xi_s = [\xi_{s,1}, \xi_{s,2}, \ldots, \xi_{s,K}]$ is the error vector, $w$ is a matrix of $L \times K$, where $w_{k}$ is the vector of the output weights corresponding to $k$th output node and $\| \cdot \|_F$ is Frobenius norm.
can also obtain the KKT optimality conditions of Eq. (23) as follows:
\[
\frac{\partial L(w, \xi, \alpha)}{\partial w} = 0 \quad \Rightarrow \quad \gamma w - G^T \alpha = 0, \tag{24a}
\]
\[
\frac{\partial L(w, \xi, \alpha)}{\partial \xi} = 0 \quad \Rightarrow \quad \xi - \alpha = 0, \tag{24b}
\]
\[
\frac{\partial L(w, \xi, \alpha)}{\partial \alpha} = 0 \quad \Rightarrow \quad Gw - \Phi + \xi = 0. \tag{24c}
\]

It can be seen from the Eq. (21) and Eq. (24) that to solve the R-WDD optimization for SOGNN is actually to solve one particular case of the R-WDD optimization for MOGNN when the number of output nodes \(k\) is fixed to 1. To be more specific, some of the matrices relevant to \(K\) in MOGNN reduce to vectors in SOGNN. Thus, the solutions of weights for SOGNN and MOGNN could be analyzed together. It is worth mentioning that the size of the activation matrix \(G\) remains the same in the both cases, and it only depends on the number of training samples \(S\) and the neuron number \(L\).

C. Proposed Regularized Weights Direct Determination

Based on the aforementioned KKT optimality conditions for SOGNN and MOGNN, the unified solution of output weights for SOGNN and MOGNN can be obtained. However, due to the efficiency concerns, different solutions to the above KKT optimality conditions can be obtained based on the size of the training data sets.

1) Small Training Data Sets: When \(S\) is not huge, or more specifically, less than \(L\), by substituting Eq. (21a) and (24a), and Eq. (21b) and (24b) into Eq. (21c) and (24c) respectively, we can get:
\[
(\gamma I + GG^T)\alpha = \Phi, \tag{25}
\]
where \(I\) is an identity matrix, whose size is \(S \times S\), and the same as the size of \(GG^T\).

By substituting Eq. (25) into Eq. (21a) and (24a), and getting rid of \(\alpha\), the following equation can be obtained:
\[
w = G^T(\gamma I + GG^T)\Phi, \tag{26}
\]
when the number of training samples is not large, the calculation of R-WDD mainly involves an inversion of a \(S \times S\) matrix. It is worth mentioning: for SOGNN, the \(w\) and \(\Phi\) is column vectors; for MOGNN, the \(w\) and \(\Phi\) evolve to matrices.

2) Huge Training Data Sets: When \(S\) is huge, i.e., greater than \(L\), by substituting Eq. (21a) and (24a), and Eq. (21b) and (24b) respectively, the explicit expression for \(\alpha\) can be obtained as follows:
\[
\alpha = -(Gw - \Phi). \tag{27}
\]
Based on this, we can obtain the following solution for weights vector \(w\), by substituting Eq. (27) into Eq. (24a):
\[
w = (\gamma I + G^TG)^{-1}G^T\Phi, \tag{28}
\]
where \(I\) is an \(L \times L\) identity matrix, the same as the size of \(G^TG\). Thus, the calculation mainly involves an inversion of an \(L \times L\) matrix, and the speed only hinges on \(L\) when \(S\) grows large.

Based on the aforementioned discussion under two circumstances and Eq. (13), the final network output equation is:
\[
\Psi(x) = \begin{cases} 
G(x)w = G^T(\gamma I + GG^T)\Phi, & S \leq L \\
G(x)w = (\gamma I + GG^T)^{-1}G^T\Phi, & S > L
\end{cases} \tag{29}
\]
As is shown in the Eq. (29), the calculation of R-WDD always only involves an inversion of a matrix of relatively small size, thus the the rapidness of R-WDD calculation is guaranteed.

IV. EXPERIMENTS AND ANALYSIS

In this section, to verify the performance of the GNN with proposed R-WDD, we first compare the generalization performance of GNN with different algorithms (i.e., LS-SVM with Gaussian kernel and ELM with Gaussian kernel, another two famous algorithm based on equality-constrained optimization) on real-world benchmark binary, multi-class classification cases. To make it more intuitive, the decision boundaries of different algorithms are presented to demonstrate the classification capability of the proposed method. Then, we also compare the computational scalability of three algorithms, and the computational efficiency of GNN and ELM. Since model’s sensitivity to hyper parameter selection is also one key feature of algorithms, experiments between GNN and ELM are conducted to test model’s tolerance for the hyper parametric changes. At last, we conduct an experiment in which different amplitude noises are added to test the robustness to random noises of the different algorithms.

The simulations in Subsection IV-B and IV-F are carried out in MATLAB R2018b environment running in Intel(R) Core(TM) i7-8700T 2.40GHz CPU with 16-GB RAM. The simulations in Subsection IV-D and IV-E are carried out in MATLAB R2018a environment running in Intel(R) Core(TM) i7-8700T 2.40GHz CPU with 16-GB RAM. The codes for LS-SVM and ELM are downloaded from [41] and [42] respectively.

A. Benchmark Data Sets and Hyper Parameters Selections

To verify the classification performance of different algorithms, 5 binary classification cases and 5 multi-classification cases have been tested in out experiments. All data sets are downloaded from the UCI Machine Learning Repository [43].

According to [16], the selection of hyper parameters greatly influences the performance of the algorithms. For LS-SVM with the popular Gaussian kernel (i.e., \(K(u, v) = \exp(-\delta|u-v|^2\)) , it mainly has two hyper parameters which are the cost parameter \(C\) (preventing the model from being too complex) and kernel parameter \(\delta\). ELM using Gaussian kernel, has three hyper parameters, the cost parameter \(C\), kernel parameter \(\delta\) and number of nodes \(L\). For the proposed GNN with R-WDD, it also has three hyper parameters which are the regularizer \(\gamma\), the parameter for GPS \(\lambda\) and the number of neurons \(L\). In this paper, the value of \(\lambda\) is fixed to 0.05. Also, the number of nodes for ELM and the proposed GNN with R-WDD for binary and multi-classification problems are fixed to 1000.
Specifically, for a binary problem, only one LS-SVM, one ELM with single output node and one SOGNN are trained; for a $K$-class multi-classification problem, it need to train $K$ binary LS-SVM, while for ELM and GNN, only one network with $K$ output nodes is required. Thus, ELM and GNN are more flexible and applicable mechanisms in multi-classification.

In experiments, for LS-SVM and ELM, the hyper parameters ($\delta$ and $C$) are searched from 31 different values in the fashion of 4-fold cross-validation grid search, which will result in 961 different combinations of ($\delta$, $C$). And the ranges of values of $\delta$ and $C$ are $\{2^{-30}, 2^{-28}, \ldots, 2^{20}, 2^{30}\}$. The value of $\gamma$ for GNN is also searched from 31 different values in $\{2^{-30}, 2^{-28}, \ldots, 2^{20}, 2^{30}\}$. At last, the specifications of each experiment (including the divisions of data sets, the number of features and classes of data set, and the hyper parameters selections for each algorithm) are reported in TABLE I and II.

Note that for each experiment, fifty trials are conducted to ensure the results are not accidental values, and the average result is reported. Besides, all the data are reshuffled at each trial and all the data are normalized before experiments.

### B. Performance Comparison on Real-World Benchmark Data Sets

TABLE II and IV present the performance comparisons of LS-SVM with Gaussian kernel, ELM with Gaussian kernel and GNN with R-WDD. Note that the simulation results include the mean testing classification accuracies (Acc), corresponding mean standard deviations (Std) and the average training time, and the best results are highlighted in boldface.

For binary classification problems, GNN could achieve comparable or even better generalization performance as LS-SVM and ELM with comparably fast learning speed. Take the results of Banana data set as instance.

- For Banana data set, the best Acc is achieved by LS-SVM, while the Acc by GNN is only 0.11% less. Also, GNN demonstrates comparably extremely fast training speed as ELM, with difference within 0.01s.

For multi-classification problems, the solutions achieved by GNN are better than LS-SVM and ELM, with higher Acc, and stabler with lower Std. To be illustrative, take the results of Glass and Ecoli data sets as examples.

- For Glass data set, Acc of GNN is much higher, with 5.46% and 4.47% higher than the results achieved by LS-SVM and ELM.
- For Ecoli data set, Acc of the GNN is at least 2% higher than LS-SVM and ELM, while Std is much lower. Although the training time is a little longer, the training speed of GNN with R-WDD is very fast and accessible.

### C. Comparison of Decision Boundaries

As a more intuitive proof of GNNs classification capability, we also conduct a simple experiment aiming at showing the decision boundaries of different algorithms on a nonlinear separable data classes. For this experiment, it is conducted on a synthetic two-dimensional, two class data set called double moon, with a pattern of two moons intertwining. Since it is a binary classification problem, the pattern vectors of Class 1 is tagged “+1” labels and ones of Class 2 received “-1” labels.

For this experiment, all the classifiers are trained with a data set with 3000 samples. The decision boundary of each algorithm is recorded in Fig. 3 (a)-(c), respectively. Note that this simple experiment is set to demonstrate the classification ability of each classifier intuitively, thus all the samples were used to train and the testing accuracy rate were not given.

### D. Computational Scalability and Efficiency

One of the biggest merits of ELM originates from its excellent computational scalability and efficiency, which means the training speed of ELM maintains when the scale of problem grows and the number of hidden neurons $L$ increases. In this subsection, experiments on Liver data sets are conducted to verify the computational scalability and efficiency of GNN. To be specific, since the experiments here is set to evaluate the computational scalability and efficiency of algorithms,
TABLE III
PERFORMANCE COMPARISON OF LS-SVM, ELM AND GNN: BINARY CLASS DATA SETS

| Data Sets  | LS-SVM | ELM | GNN |
|------------|--------|-----|-----|
|            | Testing | Testing | Training | Testing | Testing | Training | Testing | Testing | Training |
|            | Acc (%) | Std (%) | Time (s) | Acc (%) | Std (%) | Time (s) | Acc (%) | Std (%) | Time (s) |
| Australian | 72.08   | 3.44  | 0.2462   | 74.69   | 1.42   | 0.2267   | 75.91   | 3.11   | 0.2639   |
| Banana     | 89.63   | 0.33  | 0.1241   | 89.04   | 0.46   | 0.1124   | 89.52   | 0.61   | 0.1206   |
| Diabetes   | 77.01   | 2.41  | 0.1550   | 77.52   | 2.46   | 0.1328   | 77.29   | 2.05   | 0.1421   |
| Liver      | 71.17   | 3.92  | 0.2541   | 74.87   | 1.44   | 0.2817   | 72.68   | 4.04   | 0.2370   |
| Ionosphere | 91.58   | 2.74  | 0.2151   | 92.12   | 4.73   | 0.1761   | 91.03   | 2.66   | 0.2240   |

Fig. 3. The decision boundaries of each algorithm. It can be observed that three algorithms can classify the nonlinear separable data set well.

TABLE IV
PERFORMANCE COMPARISON OF LS-SVM, ELM AND GNN: MULTI-CLASS DATA SETS

| Data Sets | LS-SVM | ELM | GNN |
|-----------|--------|-----|-----|
|           | Testing | Testing | Training | Testing | Testing | Training | Testing | Testing | Training |
|           | Acc (%) | Std (%) | Time (s) | Acc (%) | Std (%) | Time (s) | Acc (%) | Std (%) | Time (s) |
| Iris      | 96.22   | 2.36  | 0.0231   | 96.04   | 2.37   | 0.0222   | 96.58   | 2.34   | 0.0257   |
| Glass     | 67.32   | 5.04  | 0.0417   | 68.41   | 4.27   | 0.0326   | 72.68   | 4.94   | 0.0421   |
| Wine      | 97.63   | 1.82  | 0.0343   | 98.48   | 4.46   | 0.0326   | 98.54   | 1.63   | 0.0457   |
| Ecoli     | 85.93   | 3.52  | 0.0344   | 87.48   | 2.91   | 0.0382   | 89.55   | 2.23   | 0.0481   |
| Vehicle   | 83.19   | 1.93  | 0.1029   | 83.16   | 1.89   | 0.0931   | 84.05   | 1.75   | 0.1266   |

1) Computational Scalability: In the simple example shown in Fig. 4, we can see the GNN has comparable (or even better) computational scalability as ELM. Specifically, according to the analysis in [16], for LS-SVM the main computational cost comes from its calculating Lagrange multipliers $\alpha$, which is highly associated with the size of data sets $S$. However, since ELM and GNN with R-WDD have different equations calculating the output weights when $S$ is great (refer Eq. (29)), the computational cost of ELM and GNN reduces dramatically. Thus, ELM and GNN has better computational scalability, compared to LS-SVM.

2) Computational Efficiency: The results of comparison on the computational efficiency, has been shown in Fig. 5, we can see the training times consumed by both ELM and GNN do not increase remarkably, as $L$ increases. Even more, the training time consumed by GNN grows a bit slower than ELM with a smoother slope.

Based on the above two experiments and the analysis above, it can be seen that the GNN still has extremely fast training speed when the $S$ and $L$ becomes very large. Thus, it can be concluded that the GNN has great computational scalability and efficiency, with regard to the size of training data sets $S$ and the number of neurons $L$. 

Fig. 4. Computational scalability of different algorithms: this experiment runs on the amplified Liver data set (20000 samples), and the number of neurons of GNN and ELM is 100. We can see that as $S$ grows, the training time consumed by LS-SVM exponentially increases, while the training time consumed by ELM (Gaussian kernel) and GNN increases very slowly.

we simply amplify size of the Liver data set by randomly copying the samples. To avoid accidental results, each data point plotted is the average values of 100 trials.
E. Sensitivity to Hyper Parameters

Axiomatically, the selections of the hyper parameters are vital to the performance of the classification algorithms. However, if the model is not very sensitive to changes of hyper parameters, i.e., the best generalization performance could be achieved in a wide range of combinations of hyper parameters, the model would have better utility value, since it require less user intervention in real-world implementation. Thus, in this subsection, the ensuing experiments are conducted to test the sensitivity of ELM and GNN to hyper parameters. Even more, the generalization performances of ELM and GNN when $L$ is very small are also explored. To avoid accidental results, each data point is the average values of 100 trials and the ELM’s Gaussian parameters and GNN’s GPS parameter follow the optimal settings listed in Subsection IV-A.

1) Sensitivity to Hyper Parameters: The experiments are conducted on a binary problem (Liver data set) and a multi-class problem (Vehicle data set) to show GNNs sensitivity to the combinations of two hyper parameters (i.e., $\gamma$ and $L$). Similarly, as a comparison, the same experiments on ELM are conducted to test the sensitivity to the combinations of two similar hyper parameters (i.e., $\delta$ and $L$).

According to the results shown in Fig. (6) and Fig. (7), the best performance of GNN could be achieved in a wider range of combinations of $(\gamma, L)$. Especially, the performance of GNN is not sensitive to $L$, and can still perform well even when there only exists a very few neurons in the network. Thus, only $\gamma$ need to be specified to achieve best performance. However, the ELM is relatively more sensitive to the combinations of $(\delta, L)$, and good performance could be achieved only when the $L$ is large enough.

2) Performance when $L$ is small: From the Fig. (6) and Fig. (7), we can also see that when neurons are few, the generalization performance of GNN are still good, while the generalization performance of ELM grows slowly when $L$ grows. Thus, experiments on the same two data sets are conducted to compare the generalization performance of these two algorithms when $L$ is small. The basic experimental settings also follow the ones in Subsection IV-A except $(\delta,L)$ for ELM and $(\gamma,L)$ for GNN.

From the results shown in Fig. (8) and Fig. (9), it is clear that the GNN works still very well with high Accs, when the hypothesis feature space is not complicated (i.e., $L < 50$), while for ELM, we can performance deteriorates greatly when lacking enough neurons. Thus, with appropriate values of $\gamma$, the approximation capability of GNN is much better than ELM, which needs more neurons to ensure its approximation capability. As a conclusion, GNN has much better generalization performance than ELM when the network hypothesis feature space is simple (i.e., $L$ is small).

Thus, through the two aforementioned experiments, we can conclude the that the GNN is less sensitive to the changes of hyper parameters than ELM. What is more, it still performs excellently when the dimensionality $L$ of the hypothesis feature space is small. Thus, the model proposed has great utility values and is easy to implement.

F. Comparison of Classification Robustness

To investigate the classification robustness of the proposed GNN with R-WDD, we conduct the following experiments on the same real-world benchmark multi-classification data sets as Subsection IV-B. Then, we contaminate the data sets by...
adding random noises with amplitude of 5%, and 10% to the training sets. And the test set should be noise free since this experiment is to evaluate the classification robustness of the algorithms when noises exist. All experiment settings follow the settings in Subsection IV-A.

TABLE V

PERFORMANCE COMPARISON OF LS-SVM, ELM AND GNN: MULTI-CLASS DATA SETS WITH 5% RANDOM NOISES

| Data Sets | LS-SVM | ELM | GNN |
|-----------|--------|-----|-----|
| Iris      | 94.31  | 95.21 | 96.02 |
| Glass     | 63.21  | 63.76 | 68.55 |
| Wine      | 93.81  | 96.57 | 97.77 |
| Ecoli     | 80.42  | 82.75 | 85.41 |
| Vehicle   | 79.21  | 79.51 | 81.89 |

TABLE VI

PERFORMANCE COMPARISON OF LS-SVM, ELM AND GNN: MULTI-CLASS DATA SETS WITH 10% RANDOM NOISES

| Data Sets | LS-SVM | ELM | GNN |
|-----------|--------|-----|-----|
| Iris      | 91.87  | 92.46 | 94.59 |
| Glass     | 60.19  | 61.72 | 66.77 |
| Wine      | 92.73  | 95.94 | 98.21 |
| Ecoli     | 78.80  | 80.13 | 84.51 |
| Vehicle   | 75.27  | 77.37 | 79.25 |

Fig. 7. The hyper parametric sensitivity test runs on vehicle data set (multi-class). GNN can achieve good generalization performance in a wider range of combinations of $\gamma, L$ than ELM. GNNs performance is only sensitive to the values of $\gamma$, while ELM could only achieve best performance in a narrow range of combinations of $(\delta, L)$.

Fig. 8. The experiment runs on Liver data set (binary). The ELM starts with a low generalization accuracy when the neurons are very few, and the accuracy grows slowly, while the GNN has relatively high generalization accuracy when $L$ is larger than 10.
As verifications, the comparison experiments on various real-world benchmark data sets have proved GNN has the comparable (or better) generalization performance, especially in multi classification problems, compared with LS-SVM, ELM. Besides, the excellent computational scalability and efficiency of GNN have been confirmed in the comparison experiments. As for the hyper-parametric experiments, less sensitivity of GNN to the combinations of hyper parameters has been observed. Especially, the generalization ability of GNN hardly relies on the values of $L$ and good performances could be achieved even though the dimensionality of the hypothesis feature space $L$ is very small. Finally, the experiments on contaminated data sets also verified the classification robustness of the proposed method.

**V. Conclusion**

In this paper, to tackle the slow operations of BP-type algorithms and computational robustness problems, a novel type GNN model with structural simplicity has been proposed and investigated, based on GPS and polynomials function approximation. Then, in order to overcome the disadvantages of easy over-fitting and improve model’s generalization performance, a R-WDD method, based on equality-constrained optimization, has been proposed to minimize networks’ empirical risk and structural risk. For better computational scalability and efficiency, two solutions to the R-WDD optimization are given, depending on the scale of the problems and the complexity of the model.
