Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **11**, Number **1**, February **2026**

Application and Performance Evaluation of Multi-Objective Trust Region Algorithms for Machine Learning Model Optimization

Guo-Ming Zhang[1]
[1]College of Computer Science,
Guangdong Business and Technology University, Zhaoqing 526060, P. R. China
zhangmingarticle@126.com

Bao-Li Liu[2,*]
[2]Aviation Engineering College,
Air Force Engineering University, Xi'an 710051, P. R. China
719299492@qq.com

Boris Kim[3]
[3]School of Information Technology and Engineering
St. Paul University Philippines, Tuguegarao 3500, Philippines
nc8050@163.com

[*]Corresponding author: Bao-Li Liu

Received October 02, 2024, revised January 26, 2025, accepted April 20, 2025.

Abstract. *BP neural network (BPNN) is the most representative machine learning model, and most of the existing researches optimize BPNN by heuristic algorithms, but it cannot guarantee that the weights and biases searched each time are the global optimal solutions, which leads to a long training time. To address the above issues, this paper utilizes the multi-objective trust region (TR) method to optimize the BPNN. Firstly, the multi-objective TR is improved by using self-correcting geometric method, which takes the distance factor into consideration when choosing the interpolation points for replacement, and at the same time, the Lagrange function is used to maintain the good geometric properties of the interpolation point set, so as to enhance the approximation effect of the model function. Relied on this, the weight and threshold correction model of BPNN is established, and a TR centered at the given point is determined, and then the approximation function is approximated within this range and the target point is obtained. Then the quadratic equation of the cost function is constructed as an unconstrained optimization problem and addressed by the conjugate gradient method, which not only improves the training speed but also ensures the overall convergence of the algorithm. The performance evaluation outcome on the public dataset UCI Abalone indicates that the AUC value and training time of the proposed BPNN optimization model are 0.976 and 0.0669s, respectively, which are better than the comparison model, verifying the effectiveness of the suggested model.*
Keywords: BP neural network; Machine learning; Multi-objective trust region; Self-correcting geometry; Conjugate gradient

1. **Introduction.** Machine learning (ML) is a science that aims to study how to use computers to simulate or implement human studying activities. Since the 1980s, ML has become one of the most important methods for realizing artificial intelligence and has attracted much attention in the field of artificial intelligence [1]. Many ML algorithms can

be described as optimization problems. In this paper, we focus on the most representative BP neural network (BPNN) algorithm, which has been applied in many fields such as data mining [2], pattern recognition [3], image processing [4], and so on.

However, BPNN suffers from slow convergence of errors, correction of weights and thresholds, difficulty in determining the framework of the neural network, selection of the initial values of each parameter, and the problem of falling into local minima [5]. This paper will focus on the optimization of the slow convergence speed of BPNN and the correction of weights and thresholds, and find a more optimal solution in the limited time and the solution space that can be explored. In this way, higher accuracy and better convergence can be achieved.

1.1. **Related work.** Optimization for BPNN algorithm is gathered in optimization theory on one hand, while on the other hand optimization for learning algorithm controls the speed of parameter change in other perspectives [6]. Wang et al. [7] used conjugate gradient algorithm to optimize the output of weights of BPNN to avoid the network from falling into local minima prematurely. Rebentrost et al. [8] proposed a Newton descent method based on learning rate improvement was used to enhance the BP neural network algorithm, but the computational consumption is large. Li et al. [9] investigated the impact of momentum factor on the BPNN algorithm, and the outcome indicated that appropriate improvement of the momentum factor can accelerate the convergence of the whole network. Fernandes and Yen [10] utilized links between BPNN nodes for pruning, merging two implicit layer nodes when their outputs have a large correlation, but are prone to falling into local optima. Li et al. [11] proposed a grey correlation analysis to analyze the correlation between the nodes and the output of the whole network, and then delete the nodes with the least correlation to optimize the network structure.

Subsequently, many scholars used heuristic algorithms (HA) to seek the extreme values of the target function of the BPNN. Huang et al. [12] suggested a Genetic Algorithm (GA) based optimization of the BPNN, which discarded the original gradient descent method, and adopted the GA to find the near-optimal solution in the solution space, but with high time complexity. Pan et al. [13] suggested a neural network algorithm that combines BPNN with Particle Swarm Optimization (PSO) algorithm but the computation is time consuming. García-Ródenas et al. [14] used PSO algorithm and gravitational search algorithm to train the BPNN algorithm. Gupta and Raza [15] applied the forbidden algorithm to BPNN to obtain a globally optimal solution. Kuang et al. [16] applied the ACO algorithm for continuum function optimization problem to BPNN but the convergence was poor. Zhang et al. [17] used GA to obtain a new population of BPNN weights and biases, and then replaced the new population by an annealing operation of Simulated Annealing (SA) Algorithm, and repeated the iterations up to the preset number of iterations, but the training time is long. Li et al. [18] used Monte Carlo comparison method to compare the optimization of GA and SA algorithms on BPNN, and the optimization performance of GA is better than that of SA. Zhao et al. [19] proposed to combine the whale optimization algorithm with BP neural network to improve the performance of the algorithm, but the solution speed is slow.

The above HA optimization model still has many shortcomings, such as the inability to guarantee the global optimal solution of the objective issue, the main advantage of the trust domain (TR) algorithm over the HA algorithm is that it can provide more reliable optimization solutions, and theoretically has better convergence and stability. Mao et al. [20] combined the idea of TR to estimate the fitness of the BPNN, thus reducing the number of true computations of the fitness values. Li and He [21] combined TR with

an evolutionary search algorithm to correct the regularization coefficients of the BPNN, which improved the network performance.

1.2. **Contribution.** In summary, most of the existing models for optimizing BPNN based on HA are letting in the solution space according to their respective heuristics, but there is no guarantee that the optimal solution can be searched every time. Therefore, this paper introduces a multi-objective TR algorithm with better convergence and stability to optimize the BPNN.

Firstly, to deal with the issue that multi-objective TR fails to approximate the effective point set as the computation of the objective function increases, the multi-objective TR is improved by using the self-correcting geometry method, which takes into account the distance factor when choosing the interpolated points to be replaced and each unsuccessful iteration leads to an improvement in the geometry of the interpolated set, which leads to an improvement in the approximation of the model function. Then, based on the improved multi-objective TR to establish the weight and threshold correction model of BPNN, approximate the approximation function in the range of TR and obtain the objective point, construct the quadratic equation of the cost function, treat it as an unconstrained optimization problem, and solve it by using the conjugate gradient method to obtain the optimal weights and thresholds. Finally, the performance evaluation experiments on UCI Abalone show that the proposed model has low training and testing time, which not only improves the solution speed, but also ensures the overall convergence of the model.

2. Theoretical analysis.

2.1. **BP neural network.** BPNN is the ML model learned by the error backpropagation approach, which is extensively adopted in different domains due to the easy function and uncomplicated network framework. The framework of BPNN usually consists of input, obscured and output levels [22], the amount of neural nodes in every level is decided according to the true condition, the neural nodes in the same level are not connected to each other, and the neural nodes in the lower level can only transmit activation information to the neural nodes in the higher level. The Framework of BPNN is shown in Figure 1.

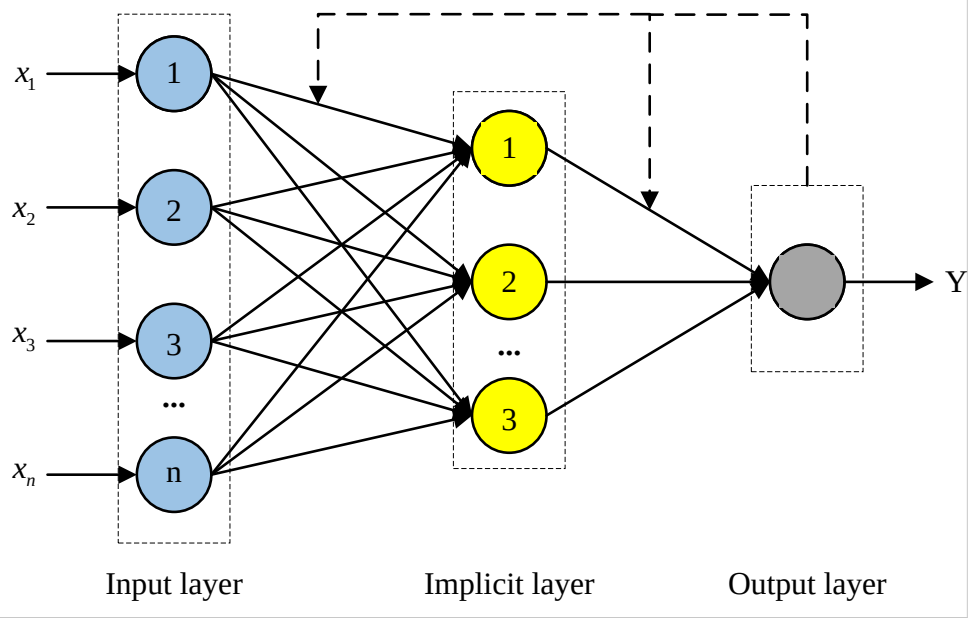

Figure 1. The model framework of BPNN

The input level is with n nodes, the implicit level has q nodes, the output level is with m nodes, the input is an n-dimensional characteristic vector $X = (x_1, x_2, \ldots, x_n)^T$, the output is an m-dimensional characteristic vector $Y = (y_1, y_2, \ldots, y_m)^T$, V is the connection weight among the input level and the obscured level, and W is the connection weight between the obscured level and the output level. The input data is first passed into the input level of the BPNN, then forward to the implicit level, and after processing through the activation operation of the implicit level, it is passed to the output level, and eventually the forecasting outcome is output by the processing of the output level.

BPNN is not perfect and requires a large amount of iterative calculation during training. In addition, the performance of BPNN is easily affected to the option of original weights. If the original weights are not properly chosen, it may lead to network training failure or fall into local minima. Thus, this article will intend to enhancing the above mentioned BPNN defects to enhance the training speed of BPNN and save time consumption.

2.2. **Trust region algorithm.** The TR method and HA are two major methods for solving nonlinear optimization problems [23]. Unlike the HA method, TR constructs a quadratic model of the target function near the current point and chooses a neighbor of the current point as the trust domain. TR is able to provide relatively reliable optimization solutions because it bases its calculations on an explicit mathematical model in each iteration, rather than simply on heuristics or experience. The structure of TR method is indicated in Figure 2.

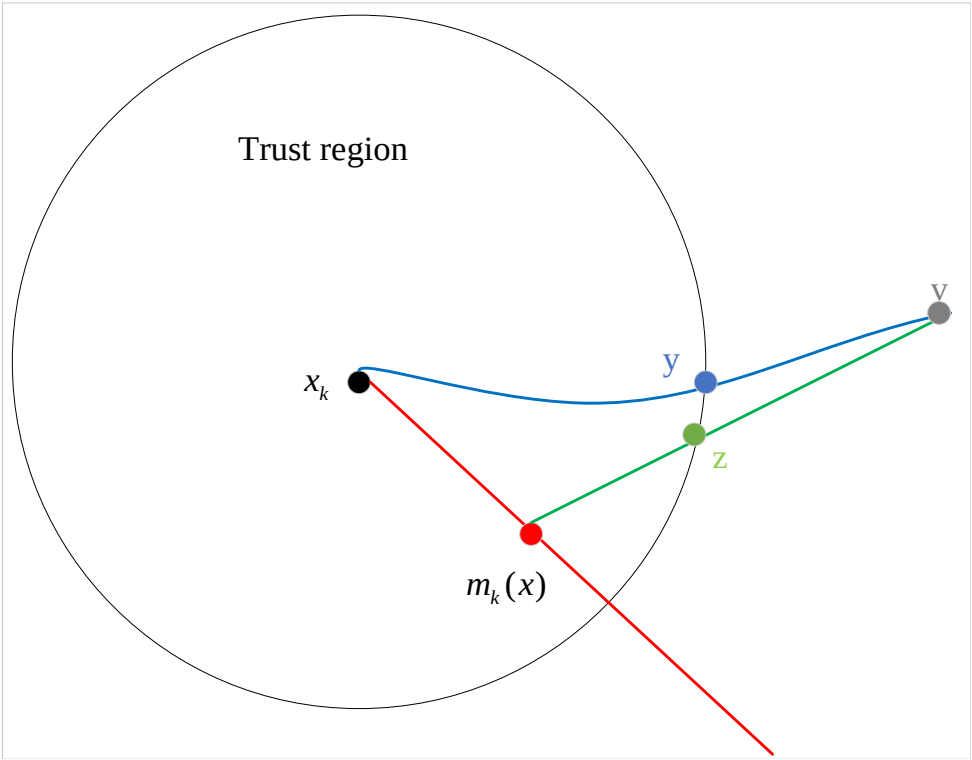

Figure 2. The structure of TR method

The quadratic model of the target function $f(x)$ at the current iteration point x_k is as follows:

$$m_k(p) = f(x_k) + g_k^T p + \frac{1}{2} p^T B_k p \tag{1}$$

where $g_k = \nabla f(x_k)$, B_k is $\nabla^2 f(x_k)$ or its approximation, and obviously $m_k(0) = f(x_k)$. Thus, the subproblem of the TR is as follows:

$$\min\{m_k(p) : p \in R^n, \|p\| \le \Delta_k\} \iff \min\left\{g_k^T p + \frac{1}{2} p^T B_k p : p \in R^n, \|p\| \le \Delta_k\right\} \quad (2)$$

where $\Delta_k > 0$ is the radius of the TR.

Let the optimal solution of the subproblem be p_k, then the approximation between $m_k(x)$ and $f(x)$ is described by the following equation:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \quad (3)$$

For the update of the TR radius, given the parameter $0 < \lambda_1 < \lambda_2 < 1$, if $\rho_k < \lambda_1$, then $m_k(x)$ is considered to be a poor approximation of $f(x)$, so $x_k + p_k$ cannot be used as a new iteration point, and the TR radius needs to be reduced. If $\rho_k > \lambda_2$ and very close to 1, then $m_k(x)$ and $f(x)$ are very close, and $x_k + p_k$ can be used as a new iteration point, and the TR radius can be expanded in the next iteration; in other cases, the TR radius is kept unchanged.

3. **Multi-objective trust domain algorithm based on self-correcting geometry.**

The optimization of BPNN is not a simple single-objective optimization, but in essence, it is a multi-objective optimization process for weights and thresholds [24], this paper will use the multi-objective TR algorithm to optimize and solve it. However, the target of the multi-objective TR optimization issue is to approximate the entire set of valid points, but if the objective function is computationally intensive, approximating the entire set of valid points becomes impractical. Based on this, this paper utilizes the self-correcting geometric method [25] to improve the TR algorithm, which takes the distance factor into account when choosing the replacement interpolation points, thus reducing the function computation; meanwhile, the Lagrangian function is used to maintain the good geometric properties of the interpolation point set.

Since the effectiveness of the function approximation of the multi-objective TR model and the convergence of the algorithm are closely related to the geometric properties of the interpolated point sets, this paper applies the self-correcting geometric method to the optimization of TR. Assume that the objective functions are $f_i, i = 1, 2, \ldots, q$, the initial point is x_0, the initial TR radius is Δ_0, the initial equalized set of interpolated points Y_0, and the parameters $0 < \lambda_1 \le \lambda_2 < 1, 0 < \mu_1 \le \mu_2 < 1, \mu > 1, \Lambda > 1$. The specific steps are as follows.

Step 1: Construct the initial model function $m_i^k, i = 1, 2, \ldots, q$ by making $k = 0$. In this paper, the quadratic model is used to replace the objective function in the original TR, and the derivative information of f_i must be available, but since the basic quadratic model function is the Taylor model, in which the derivative information of f_i is not available, the Taylor model can not be used as the quadratic model, and the interpolation method based on the quadratic Lagrange polynomials [26] is used here to construct the quadratic model for f_i.

Let P_n^2 denote the polynomial space consisting of all polynomials in R^n with number less than or equal to 2. Given a balanced set of interpolating points $Y = \{y^1, y^2, \ldots, y^p\} \subset R^n$ and a set of interpolating basis functions $L = \{l_1, l_2, \ldots, l_p\}$, where $l_i \in P_n^2$ is a quadratic Lagrangian polynomial basis, and a model function $m_i^k(x)$ of f_i as in Equation (4) if $i = j$, then $l_i(y^j) = 1$, and otherwise $l_i(y^j) = 0$, and each interpolating point satisfies $m_i(y^j) =$

$f_i(y^j)$, then the subproblem of multi-objective optimization (MOP) is $\min_{x\in R^n} m^k(x)$.

$$m_i^k(x) = m_L(x; f_i, Y_k) = \sum_{j=1}^{p} f_i(y^j) l_j(x) \tag{4}$$

Step 2: Calculate the ideal point $p_i^k = \min_{x\in B_k} m_i^k(x)$, where p^k is the ideal point of m^k in B_k. p^k gives the direction in which to reduce $m_i^k(x)$. The goal is to move as far as possible in the direction of p^k within the TR radius, which not only ensures that the model has a good enough approximation, but also serves as a kind of step size control.

Step 3: Calculate the test points. The ideal point is given by geometric scalar [27] optimization as shown in Equation (5), where $v^k = f(x_k) - p^k \in R_+^q$. It is easy to see that as long as x_k is not a Pareto critical point of the MOP [27], the direction v^k is not a zero vector, such that $t_k^+ = 0, x_k^+ = x_k$. Starting from the existing iteration point x_k, the local ideal point p^k is computed, and the test point x_k^+ is calculated by moving in that direction via Equation (5).

$$\min_{x\in R^n} m^k(x) = \min\left\{t \in R \mid f(x_k) + tv^k - m^k(x) \in R_+^q, x \in B_k\right\} \tag{5}$$

Step 4: Generate a search direction using the local ideal points, and move along this direction as much as possible to provide a candidate point for the next iteration. Test point x_k^+ is received as the next iteration point only when the improvement condition is satisfied. Based on the results of the test point reception, the TR and model function are adjusted. If the test point is successfully received, if $t_k^+ = 0$ or $m_{\max}^k(x_k) - m_{\max}^k(x_k^+)$, let $\tilde{\rho}^k = 0$, otherwise compute $f_i(x_k^+)$ and the ratio $\rho_i^k = (f_i(x_k) - f_i(x_k^+))/(m_i^k(x_k) - m_i^k(x_k^+))$. Let $\tilde{\rho}^k = \min_{i=1,\ldots,q} \rho_i^k$, if $\tilde{\rho}^k \geq \lambda_1$, let $x_{k+1} = x_k^+$, otherwise let $x_{k+1} = x_k$.

Step 5: Update the interpolated point set. If the iteration is successful, $\tilde{\rho}^k \geq \lambda_1$ then makes $Y_{k+1} = Y_k \cup \{x_k^+\} \setminus \{y^r\}$, where $y^r = \arg\max_{y^j\in Y_k} \|y^j - x_k^+\|$. If $\tilde{\rho}^k < \lambda_1$, and the sets F_k and C_k are non-empty, then F_k, C_k and y^r are as follows:

$$F_k = \{y^j \in Y_k \mid |y^j - x_k| > \beta\Delta_k\} \tag{6}$$

$$C_k = \{y^j \in Y_k \setminus \{x_k\}\} \tag{7}$$

$$y^r = \arg\max_{y^j\in F_k\cup C_k} \|y^j - x_k^+\|^2 |l_j(x_k^+)| \tag{8}$$

Step 6: The TR radius is updated using the self-correcting geometry method. If the TR radius is small enough with respect to the model gradient and all the iteration points are contained in the TR, each unsuccessful iteration leads to an improvement in the geometry of the iterated set of points, which improves the model function approximation. The updated TR radius is shown below:

$$\Delta_{k+1} \in \begin{cases} [\beta_1\Delta_k, \beta_2\Delta_k], & \text{if } \tilde{\rho}^k < \lambda_1 \\ [\beta_2\Delta_k, \Delta_k], & \text{if } \lambda_1 \leq \tilde{\rho}^k < \lambda_2 \\ [\Delta_k, \infty], & \text{if } \tilde{\rho}^k \geq \lambda_2 \end{cases} \tag{9}$$

Step 7: Update the model function. When $\tilde{\rho}^k \geq \lambda_1$, this implies that $f_i(x_k) - f_i(x_k^+) > 0$, so test point x_k^+ not only ensures that each objective function decreases, but also ensures that this decrease is sufficient, at which point it receives x_k^+ when $\tilde{\rho}^k < \lambda_1$, test point x_k^+ is not received as the next iteration point and reduces the radius of the trust domain to recalculate the model function $m_i^{k+1}(x)$, such that $k = k + 1$, go to Step 2.

4. **Machine learning model optimization based on multi-objective trust region algorithm.**

4.1. **BPNN model design based on multi-objective trust region method optimization.** Most of the existing BPNN optimization models utilize HA algorithms to let the solution space be searched according to their respective heuristics, but it is not guaranteed that the weights and biases from each search end up in the same minimum, which is a waste of computational resources. This paper will use the multi-objective TR proposed in the previous section to optimize the BPNN model, the optimization process is shown in Figure 3.

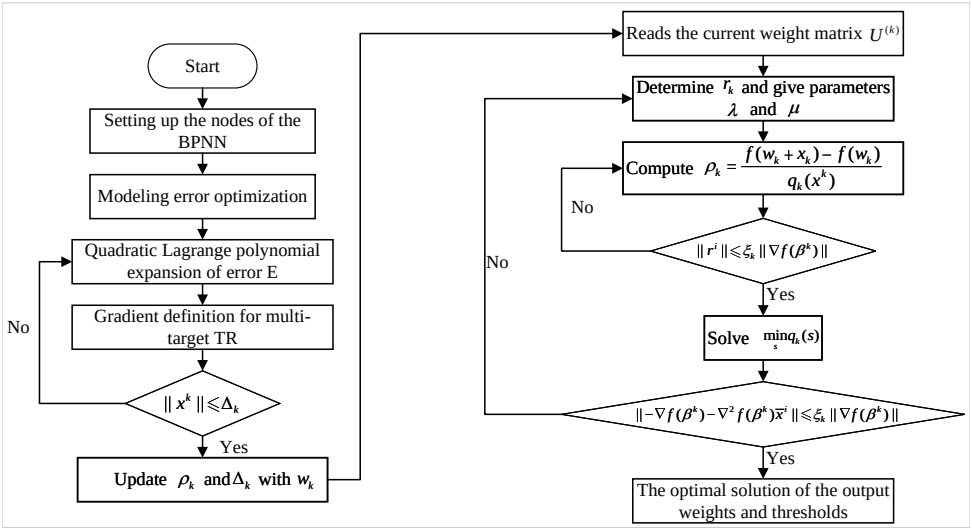

Figure 3. Multi-objective TR optimization process for BPNN models

Firstly, a circular domain centered at a given point is determined, and then the approximation function is approximated within this range to obtain the target point, and if the error accuracy requirement is not satisfied, the above process is repeated until the optimal point is found. The optimized BPNN model corrects the network weights and thresholds with the TR direction of the approximation function during the learning process.

The key to optimization of BPNN is to obtain as many minima of the objective function as possible in finite time and to circumvent repeated local minima. The algorithm changes in two parts, one is the strategy for letting the global solution space, and the other is to compare the letting weights and biases with the letting local minima. This paper will first establish a multi-objective TR to build a weight and threshold correction model, and then construct the quadratic equation of the cost function as an unconstrained optimization issue, which is addressed by the conjugate gradient method [28], which not only improves the training speed, but also ensures the overall convergence of the algorithm.

Define the BPNN model input layer node as x_i, implicit level node as y_j, output level node as z_l. The implicit level node output is as follows:

$$y_j = f\left(\sum_i w_{ji} x_i - \theta_j\right) \tag{10}$$

where w_{ji} is the network weight between the input level node and the implicit level node; θ_j stands for the activation operation, generally using the sigmoid operation. The output formula of the output level node is as follows:

$$z_l = f\left(\sum_j v_{lj} y_j - \theta_l\right) \tag{11}$$

where v_{lj} is weights among the implicit level nodes and the output level nodes. The error of the output layer node is as follows:

$$E = \frac{1}{2}\sum_{l}\left\{t_l - f\left[\sum_{j} v_{lj} f\left(\sum_{i} w_{ji}x_i - \theta_j\right) - \theta_l\right]\right\}^2 \tag{12}$$

The above is the derivation of the error model of the network weights in the learning process of BP NN, and the optimization of the network weights is relied on the principle of error minimization to enhance the learning effect of the network. The error optimization model based on error minimization is established as follows:

$$\min E = \frac{1}{2}\sum_{l}\left\{t_l - f\left[\sum_{j} v_{lj} f\left(\sum_{i} w_{ji}x_i - \theta_j\right) - \theta_l\right]\right\}^2 \tag{13}$$

Define a point $U^{(k)} = (w_{ji}^{(k)}, v_{lj}^{(k)})^T$ on the two-dimensional search plane, expand the expression of error E at point $U^{(k)}$ based on the interpolation method of quadratic Lagrange polynomials, and take the quadratic approximation to obtain the approximated error function E' as follows:

$$E' = E(U^{(k)}) + \nabla E(U^{(k)})^T d + \frac{1}{2} d^T \nabla^2 E(U^{(k)}) d \tag{14}$$

where $d = U - U^{(k)}$, can be understood as the step length between neighboring search points. Accordingly, the response multi-objective TR model $\min E', \|d\| \leq r_k$, r_k is the TR radius, which is generally taken as a constant.

At this time, the unknown terms in the TR model are the gradient of the error E and the Hessian matrix [29], according to the definition of the gradient, and combined with the actual problem of multi-objective TR solving, the gradient can be expressed as follows:

$$\nabla E = \left(\frac{\partial E}{\partial w_{ji}}, \frac{\partial E}{\partial v_{lj}}\right)^T \tag{15}$$

The partial derivatives of the network weights w_{ji} and v_{lj} are obtained according to the form of the error function. The error function E takes partial derivatives with respect to w_{ji} as follows:

$$\frac{\partial E}{\partial w_{ji}} = \sum_{l}\sum_{j} \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_{ji}} \tag{16}$$

where each of the three terms on the right-hand side of the equation can be reduced to $\partial E/\partial z_l = -(t_l - z_l)$, $\partial z_l/\partial y_j = f'(\sum_j v_{lj}y_j - \theta_l)\cdot v_{lj}$, and $\partial y_j/\partial w_{ji} = f'(\sum_i w_{ji}x_i - \theta_j)\cdot x_i$. The error function E takes partial derivatives with respect to v_{lj} as $\partial E/\partial v_{lj} = (\partial E/\partial z_l)\cdot(\partial z_l/\partial v_{lj})$, where $\partial z_l/\partial v_{lj}$ can be simplified as follows:

$$\frac{\partial z_l}{\partial v_{lj}} = f'\left(\sum_{j} v_{lj}y_j - \theta_l\right)\cdot y_j \tag{17}$$

The above are the partial derivatives of the two unknowns with respect to the error, respectively, and Equation (18) is the Hessian matrix of the unknowns:

$$G = \nabla^2 E = \begin{pmatrix} \frac{\partial^2 E}{\partial w_{ji}^2} & \frac{\partial^2 E}{\partial w_{ji}\partial v_{lj}} \\ \frac{\partial^2 E}{\partial v_{lj}\partial w_{ji}} & \frac{\partial^2 E}{\partial v_{lj}^2} \end{pmatrix} \tag{18}$$

The gradient and Hessian matrix are put into the trust region model, that is, the algorithm improvement of BPNN weight w is completed. The optimization process of the

network threshold ξ is similar to the derivation of the weight value, which will not be described here.

4.2. **BPNN model solving based on optimization of multi-objective trust domain approach.** According to the established TR optimization model, the process in which the output weight w minimizes the cost function is solved. The quadratic model of multi-objective TR is as follows:

$$q_k(x) = \nabla f(w_k)^T x + \frac{1}{2} x^T \nabla^2 f(w_k) x \tag{19}$$

where the difference of the approximate cost function is $f(w_k + x_k) - f(w_k)$, w_k is the iterative variable that outputs the weights, and x_k is the search direction. It is always possible to find a x_k that satisfies the TR constraint $\|x^k\| \leq \Delta_k$ and minimizes $q_k(x)$. Update w_k and Δ_k by comparing the ratio ρ_k between the true reduction in the cost function and the forecasting reduction in the binary model.

$$\rho_k = \frac{f(w_k + x_k) - f(w_k)}{q_k(x^k)} \tag{20}$$

If ρ_k is large enough, it means that the computed x_k is in the direction where the value of the cost function decreases faster, so s^k is received and w_k is updated according to Equation (21).

$$w_{k+1} = \begin{cases} w_k + x_k, & \rho_k > \eta_0 \\ w_k, & \rho_k \leq \eta_0 \end{cases} \tag{21}$$

where η_0 is a given value. The TR constraint is updated according to Equation (22).

$$\begin{cases} \Delta_{k+1} \in [\lambda_1 \min\{\|x^k\|, \Delta_k\}, \lambda_2 \Delta_k], & \rho_k \leq \mu_1 \\ \Delta_{k+1} \in [\lambda_1 \Delta_k, \mu_3 \Delta_k], & \rho_k \in (\mu_1, \mu_2) \\ \Delta_{k+1} \in [\Delta_k, \mu_3 \Delta_k], & \rho_k \geq \mu_2 \end{cases} \tag{22}$$

where $0 < \lambda_1 \leq \lambda_2 < 1, 0 < \mu_1 \leq \mu_2 < 1$ are constants.

The solution of Equation (13) is then transformed in the inner level into an approximate solution x^k, i.e., $\min_s q_k(s)$, of the trust domain subproblem using the conjugate gradient method. For a given $\xi_k < 1, \Delta_k > 0$, make $\bar{x}^0 = 0, r^0 = -\nabla f(\beta^k), d^0 = r^0$, determine whether $\|r^i\| \leq \xi_k \|\nabla f(\beta^k)\|$ is satisfied, if so obtain an approximate solution $x^k = \bar{x}^i$ in the iterative direction and the algorithm ends, otherwise compute $\alpha_i = \|r^i\|^2 / (((d^i)^T \nabla^2 f(\beta^k) d^i))$ and update $\bar{x}^{i+1} = \bar{x}^i + \alpha_i d^i$.

Thus, the main computational effort of the conjugate gradient algorithm for solving the optimal solution x_k of the TR subproblem $\min_s q_k(s)$ is the product of the Hessian matrix of the cost function and the vector $\nabla^2 f(w^k) d^i$. In practice, however, the product of the Hessian array and the vector is computed only once at each iteration of the algorithm. Because $r^i = -\nabla f(\beta^k) - \nabla^2 f(\beta^k) \bar{x}^i$, then Eq. (23) satisfies the stopping criterion.

$$\| - \nabla f(\beta^k) - \nabla^2 f(\beta^k) \bar{x}^i \| \leq \xi_k \|\nabla f(\beta^k)\| \tag{23}$$

This means that $\bar{x}^i$ satisfying the stopping criterion is an optimal approximate solution to the TR subproblem. In the optimization process, the TR constraints $\|x\| \leq \Delta$ need to be carefully considered, so in the finite-step conjugate gradient iteration, $q_k(\bar{x}^i + \alpha_i d^i) < q_k(\bar{x}^i)$ that satisfies the stopping criterion must be able to find an iteration point on the TR boundary that satisfies $\bar{x}^i$. Therefore, the BPNN optimized by the multi-objective TR guarantees that the approximate solution is good enough, and at the same time it can make the multi-objective TR converge.

5. **Performance evaluation.**

5.1. **Prediction performance comparison.** This paper uses the UCI Abalone dataset [30] from Data Mining and Machine Learning to estimate the performance of the BPNN model optimized by the multi-objective TR method (ETR-BP). The Abalone dataset contains a total of 4,377 instances with 30 classes of labels, and the number of abalone's rings is the same as the tree rounds give in addition to the abalone's actual age, which can be viewed as the classes of the multiclassification problem, and it is more convenient to perform performance evaluation after BPNN optimization. The above dataset is classified into training set and testing set by 6:4. The running environment of the experiment is as follows: Intel (R) Core (TM) i7-4720HQ CPU@2.60GHz, 8GB of RAM, Ubuntu 14.04 operating system, python programming language, python 2.7 compilation environment. The learning rate in the experiment was set to 0.0001, $\lambda_1 = 0.25$, $\lambda_2 = 0.75$.

For the purpose of experimental analysis, the BPNN optimization models from literature [12], literature [13] and literature [21], denoted as GA-BP, BP-PSO and TR-BP, respectively, are selected for comparative tests. In this paper, various optimization models are evaluated using accuracy, precision, recall, F1 and AUC values and the evaluation results are shown in Table 1. The Accuracy and F1 of ETR-BP were 0.927 and 0.921, which improved 11.2% and 10.6% compared to GA-BP, 8.8% and 9.1% compared to BP-PSO, and 5.6% and 3.4% compared to TR-BP, respectively, and ETR-BP showed the best performance in predictive accuracy. In addition, AUC is the area below the ROC curve, and the closer the value of AUC is to 1, the better the accuracy of the classification prediction is. ETR-BP has an AUC value of 0.976, which is the closest to 1. It is followed by TR-BP, with an AUC value of 0.942, and the worst performer is GA-BP, with an AUC value of only 0.861. In summary, ETR-BP not only optimizes the multi-objective TR, but also optimizes the BPNN using the improved multi-objective TR to find the optimal solution of weights and thresholds by iterative searching within the TR, which has the most ideal prediction effect.

Table 1. Comparison of prediction performance of different BPNN optimization models

Model	Accuracy	Precision	Recall	F1	AUC
GA-BP	0.815	0.822	0.809	0.815	0.861
BP-PSO	0.839	0.817	0.828	0.822	0.885
TR-BP	0.871	0.861	0.896	0.878	0.942
ETR-BP	0.927	0.904	0.938	0.921	0.976

In this paper, the implied nodes are incremented from 20 to 200, and each model is simulated 50 times, and the average of the simulation results is taken, and the results are implied in Figure 4. From Figure 4, it can be seen that the training time of GA-BP and BP-PSO increases rapidly as the size of the nodes in the implicit layer increases. This is because both GA and PSO are heuristic optimization algorithms, and the performance of the algorithms is affected by the heuristic function, which has a high time complexity, and it needs to consume more resources and time to optimize the BPNN. The training time of TR-BP and ETR-BP increases more slowly, and this training time advantage will be more obvious with the increase of node size. The difference is that ETR-BP uses the optimized multi-objective TR to correct the weights and thresholds, which verifies the effectiveness of ETR-BP.

5.2. **Prediction accuracy analysis.** To evaluate the performance of ETR-BP more comprehensively, this paper compares the prediction accuracies of the trained and tested

Figure 4. Comparison of simulation results of different models with implied node changes

Root Mean Square Error ($RMSE$) of different models, as shown in Table 2. The training and testing $RMSE$ of ETR-BP are 0.0669 and 0.0691, respectively, and the results of training and testing are the closest to each other, which not only have good fitting ability but also show high prediction accuracy on the test set. TR-BP has the next best performance, with an $RMSE$ difference of 0.0185 between the training and test sets, which combines TR with an evolutionary search algorithm to achieve correction of the regularization coefficients of the BPNN, but does not take into account the optimization of the BPNN's weights and thresholds, and does not improve the TR, and thus the prediction error is larger than that of ETR-BP. The difference in $RMSE$ between the training set and the test set of BP-PSO is 0.0219. PSO utilizes only the information of the group optimum and the individual optimum in each iteration step and does not fully utilize the other information obtained in the computation process, which leads to insufficiently high prediction accuracy. GA-BP performs the worst, with a single GA encoding that does not comprehensively represent the constraints of the BPNN optimization problem, which leads to biased or incompletely accurate predictions from the model.

Table 2. Comparison of training and testing RMSE for different models

Model	GA-BP	BP-PSO	TR-BP	ETR-BP
Training	0.1238	0.1082	0.0851	0.0669
Testing	0.1795	0.1301	0.1036	0.0691

Figure 5 shows the comparison of training time and testing time spent by different models when the implicit level nodes are taken as 100, and the training and testing time of ETR-BP is 0.14s and 0.0193s, respectively, with a difference of 0.1207s, which has a faster training speed and testing speed. In contrast, the training and testing times of TR-BP are 0.2219s and 0.0301s, which are 0.0819s and 0.0108s more than ETR-BP, respectively. The training and testing time of BP-PSO increased by 0.1519s and 0.036s compared to ETR-BP. The training and testing time of GA_BP increased by 0.2588s and 0.0713s compared to ETR-BP. Combined with Table 2 and Figure 5, it can be seen that

as the number of training samples and the data size increase, the advantage of training speed of ETR-BP will be more obvious. Therefore, it shows that ETR-BP can still show low time consumption for larger data sizes.

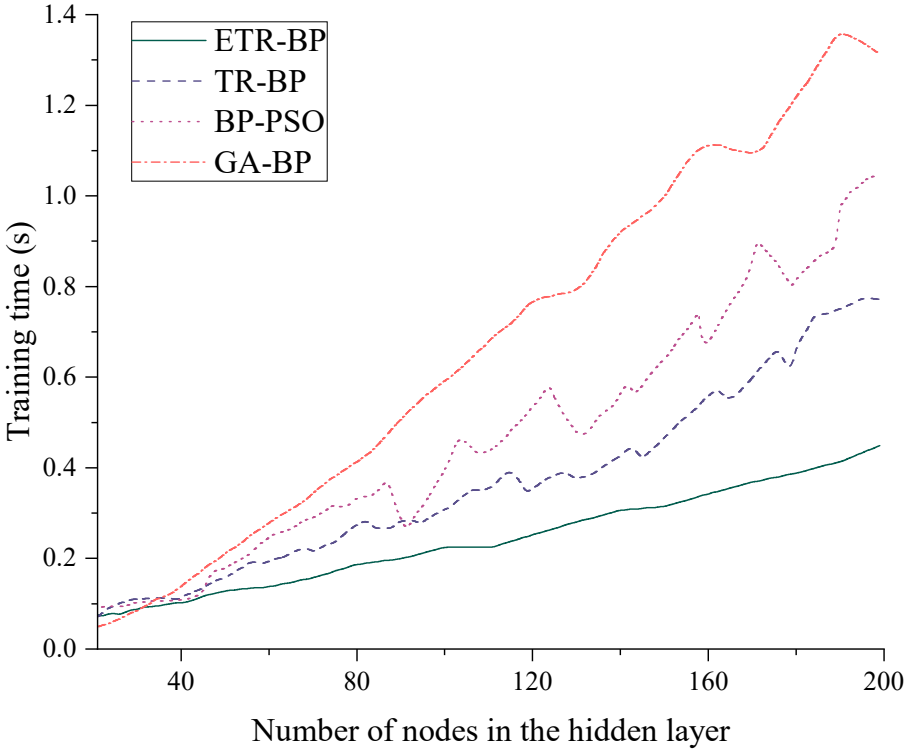

Figure 5. Training and testing time of different BPNN optimized models

6. **Conclusion.** Neural network algorithm is an important part of the ML model, this paper focuses on the most representative ML model BPNN. For the existing BPNN optimization model that cannot guarantee that the optimal solution can be searched every time, resulting in the problem of slow solution speed, this paper uses the multi-objective TR algorithm to optimize the BPNN.

Firstly, for the issue that the multi-objective TR cannot approximate the effective point set, the multi-objective TR is improved by using the self-correcting geometric method. When choosing the interpolation points for replacement, the maximized Lagrange polynomial function is used to improve the equilibrium of the interpolation point set, and the distance factor is taken into account, so as to improve the approximation effect of the model function. On this basis, the weight and threshold correction model of BPNN is established, the approximation function is approximated within the TR range and the target point is obtained. If the target point does not satisfy the error accuracy requirements, the above process is repeated until the optimal target point is found. Then the quadratic equation of the cost function is constructed as an unconstrained optimization problem, and the conjugate gradient method is used to find the optimal solution. The performance evaluation outcome indicates that the suggested model has high prediction accuracy and low training time, improves the solution speed to a larger extent, and exhibits excellent convergence.

Acknowledgment. This work is partially supported by the construction of a blockchain technology development practical teaching base for new types of employment positions in the field of software engineering based on industry-academia collaboration (No. 220905844283406).

REFERENCES

[1] J. M. Helm, A. M. Swiergosz, H. S. Haeberle, J. M. Karnuta, J. L. Schaffer, V. E. Krebs, A. I. Spitzer, and P. N. Ramkumar, "Machine learning and artificial intelligence: definitions, applications, and future directions," *Current Reviews in Musculoskeletal Medicine*, vol. 13, pp. 69-76, 2020.

[2] X. Sun, and Y. Lei, "Research on financial early warning of mining listed companies based on BP neural network model," *Resources Policy*, vol. 73, 102223, 2021.

[3] R. Pan, W. Gao, J. Liu, and H. Wang, "Automatic recognition of woven fabric pattern based on image processing and BP neural network," *The Journal of the Textile Institute*, vol. 102, no. 1, pp. 19-30, 2011.

[4] W. Kung, "Research on signal processing technology optimization of contact image sensor based on BP neural network algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 4, pp. 3911-3919, 2020.

[5] X. Yang, J. Zhou, and D. Wen, "An optimized BP neural network model for teaching management evaluation," *Journal of Intelligent & Fuzzy Systems*, vol. 40, no. 2, pp. 3215-3221, 2021.

[6] J. Wang, Y. Wen, Z. Ye, L. Jian, and H. Chen, "Convergence analysis of BP neural networks via sparse response regularization," *Applied Soft Computing*, vol. 61, pp. 354-363, 2017.

[7] J. Wang, B. Zhang, Z. Sun, W. Hao, and Q. Sun, "A novel conjugate gradient method with generalized Armijo search for efficient training of feedforward neural networks," *Neurocomputing*, vol. 275, pp. 308-316, 2018.

[8] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, "Quantum gradient descent and Newton's method for constrained polynomial optimization," *New Journal of Physics*, vol. 21, no. 7, 073023, 2019.

[9] P. Li, S. Zhang, B. Zhong, J. Wu, H. Zhang, Y. Chen, Y. Fu, Q. Wang, and Q. Li, "Service quality evaluation of bus lines based on improved momentum back-propagation neural network model: A study of Hangzhou in China," *IET Intelligent Transport Systems*, vol. 15, no. 7, pp. 958-972, 2021.

[10] F. E. Fernandes, and G. G. Yen, "Automatic searching and pruning of deep neural networks for medical imaging diagnostic," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5664-5674, 2020.

[11] L. Li, Y. Fu, J. C. Fung, H. Qu, and A. K. Lau, "Development of a back-propagation neural network and adaptive grey wolf optimizer algorithm for thermal comfort and energy consumption prediction and optimization," *Energy and Buildings*, vol. 253, 111439, 2021.

[12] H.-X. Huang, J.-C. Li, and C.-L. Xiao, "A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm," *Expert Systems with Applications*, vol. 42, no. 1, pp. 146-155, 2015.

[13] X. Pan, Y. Niu, Y. Zhao, P. Huang, and Y. Wu, "Parameter calibration method of clustered-particle logic concrete DEM model using BP neural network-particle swarm optimisation algorithm (BP-PSO) inversion method," *Engineering Fracture Mechanics*, vol. 292, 109659, 2023.

[14] R. García-Ródenas, L. J. Linares, and J. A. López-Gómez, "Memetic algorithms for training feedforward neural networks: an approach based on gravitational search algorithm," *Neural Computing and Applications*, vol. 33, no. 7, pp. 2561-2588, 2021.

[15] T. K. Gupta, and K. Raza, "Optimizing deep feedforward neural network architecture: A tabu search based approach," *Neural Processing Letters*, vol. 51, no. 3, pp. 2855-2870, 2020.

[16] Y. Kuang, R. Singh, S. Singh, and S. P. Singh, "A novel macroeconomic forecasting model based on revised multimedia assisted BP neural network model and ant Colony algorithm," *Multimedia Tools and Applications*, vol. 76, pp. 18749-18770, 2017.

[17] D. Zhang, W. Li, X. Wu, and X. Lv, "Application of simulated annealing genetic algorithm-optimized back propagation (BP) neural network in fault diagnosis," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 10, no. 4, 1950024, 2019.

[18] D. Li, F. Huang, L. Yan, Z. Cao, J. Chen, and Z. Ye, "Landslide susceptibility prediction using particle-swarm-optimized multilayer perceptron: Comparisons with multilayer-perceptron-only, bp neural network, and information value models," *Applied Sciences*, vol. 9, no. 18, 3664, 2019.

[19] J. Zhao, H. Nguyen, T. Nguyen-Thoi, P. G. Asteris, and J. Zhou, "Improved Levenberg–Marquardt backpropagation neural network by particle swarm and whale optimization algorithms to predict the deflection of RC beams," *Engineering with Computers*, vol. 38, no. 5, pp. 3847-3869, 2022.

[20] C. Mao, R. Lin, D. Towey, W. Wang, J. Chen, and Q. He, "Trustworthiness prediction of cloud services based on selective neural network ensemble learning," *Expert Systems with Applications*, vol. 168, pp. 114390, 2021.

[21] H. Li, and H. He, "Multiagent trust region policy optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 12873-12887, 2023.

[22] T.-Y. Wu, H. Li, S. Kumari, and C.-M. Chen, "A Spectral Convolutional Neural Network Model Based on Adaptive Fick's Law for Hyperspectral Image Classification," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 19-46, 2024.

[23] T.-Y. Wu, A. Shao, and J.-S. Pan, "CTOA: Toward a Chaotic-Based Tumbleweed Optimization Algorithm," *Mathematics*, vol. 11, no. 10, 2339, 2023.

[24] T.-Y. Wu, H. Li, and S.-C. Chu, "CPPE: An Improved Phasmatodea Population Evolution Algorithm with Chaotic Maps," *Mathematics*, vol. 11, no. 9, 1977, 2023.

[25] L. Laskowski, and J. Jelonkiewicz, "Self-correcting neural network for stereo-matching problem solving," *Fundamenta Informaticae*, vol. 138, no. 4, pp. 457-482, 2015.

[26] W.-H. Luo, X.-M. Gu, L. Yang, and J. Meng, "A Lagrange-quadratic spline optimal collocation method for the time tempered fractional diffusion equation," *Mathematics and Computers in Simulation*, vol. 182, pp. 1-24, 2021.

[27] D. Li, S. Feng, S. Nie, J. Ma, and C. Yuan, "Scalar and vectorial vortex filtering based on geometric phase modulation with a q-plate," *Journal of Optics*, vol. 21, no. 6, 065702, 2019.

[28] G. C. Bento, J. Cruz Neto, L. Meireles, and A. Soubeyran, "Pareto solutions as limits of collective traps: an inexact multiobjective proximal point algorithm," *Annals of Operations Research*, vol. 316, no. 2, pp. 1425-1443, 2022.

[29] M. Fatemi, "A new efficient conjugate gradient method for unconstrained optimization," *Journal of Computational and Applied Mathematics*, vol. 300, pp. 207-216, 2016.

[30] A. A. Abro, W. A. Sıddıque, M. S. H. Talpur, A. K. Jumani, and E. Yaşar, "A combined approach of base and meta learners for hybrid system," *Turkish Journal of Engineering*, vol. 7, no. 1, pp. 25-32, 2023.