Service Classification Based on Improved BP Neural Network

Qiliang Zhu[1], Shangguang Wang[1*], Qibo Sun[1], Ching-Hsien Hsu [2], Fangchun Yang[1]

[1]*State Key Laboratory of Networking and Switching Technology*
[1]*Beijing University of Posts and Telecommunications*
[1]*Beijing, China*
[2] *Department of Computer Science and Information Engineering*
[2]*Chung Hua University*
[2]*Hsinchu 707, Taiwan*
{qiliang, sgwang, qbsun}@bupt.edu.cn; chh@chu.edu.tw; fcyang@bupt.edu.cn

Abstract

With the development of the Internet, several candidate services have emerged for achieving the same task, most of which are functionally identical but different in non-functional properties. Therefore, these services can be classified into different service-quality levels. The so-called Quality of Service (QoS) comprises a set of non-functional properties that can be used to efficiently classify and rank these various services. In this paper, an algorithm called CNBP is proposed to address the problem of automatically classifying services. The core idea of this algorithm is that the weights and biases of a back-propagation network are optimized by a hybrid optimization based on two algorithms: the Nelder-Mead simplex algorithm and the Cuckoo search algorithm. The improved back-propagation (BP) classifier is used to classify candidate services into different QoS levels. Through experiments based on the Quality of Web Services dataset and a comparative analysis with traditional back-propagation networks and three other classification algorithms, we demonstrate that the proposed algorithm performs well in terms of its classification accuracy and stability.

Keywords: service; QoS; classification; BP

1 Introduction

With the development of network technology, numerous Internet-based services have emerged. These services belong to a variety of domains, such as commerce, science, education, games, etc. In general, for any given task, several atomic services can be found with identical functions but different non-functional properties. Owing to the abundance of candidate services, it is unfeasible for users to assess all of these services in order to select the best one. Consequently, selecting the best service to satisfy the requirements of users is an issue that directly affects the performance of service-oriented applications.

The non-functional properties of web services play an important role in service-oriented architectures. These non-functional properties act as distinguishing factors, among services that are functionally identical. The so-called Quality of Service (QoS) comprises a group of non-functional properties, including the price, reliability, availability, safety, throughput, response time, etc. Each property characterizes the service's quality from a certain perspective [1]. Thus, functionally similar services can be classified and ranked according to these QoS properties. However, artificial classification is complex and time consuming. In addition, tagging these services objectively and accurately requires that service requestors are familiar with a variety of categories. Hence, the automatic classification of services has become a common concern and warrants in-depth studies.

In this paper, we propose a method for efficiently identifying services with a set of QoS properties by employing a back-propagation (BP) neural network that is optimized with a hybrid of two algorithms: the Cuckoo search algorithm and the Nelder-Mead simplex algorithm. We use this hybrid algorithm to optimize the initial weights and biases of the BP neural network. Then, the BP classifier is used to classify the candidate services into different QoS levels. Because this hybrid Cuckoo-Nelder-Mead algorithm optimizes a BP neural network, we refer to our proposed algorithm as the "CNBP." Through experiments based on the Quality of Web Services (QWS) dataset and a comparative analysis with the traditional BP network and three other classification algorithms, we demonstrate that the proposed CNBP performs well in terms of its classification accuracy and stability.

The remainder of this paper is organized as follows. In the following section, we provide a detailed overview of the related work. Section 3 describes the correlation algorithm. Section 4 details the service-classification algorithm based on the improved BP neural network. Section 5 presents the experimental results and analysis. Finally, conclusions and future work are provided in Section 6.

2 Related Work

Several researchers have suggested proposals for QoS-based service ranking and classification. In order to

*Corresponding author

assist users in selecting the service that best satisfies their QoS requirements from among several similar services, a QoS-based service-ranking and -selection approach was presented by Yau et al. [2]. Their approach offers high flexibility to the user in terms of the specified requirements. It then selects the service that best satisfies these requirements, rather than simply recommending a service that merely qualifies. Zibin Zheng et al. [3] proposed CloudRank, an approach to ranking cloud services in an optimal way using a greedy algorithm. CloudRank ranks components rather than services, but the algorithm is used to rank a set of items and then treats these explicitly rated items and the unrated items equally. It does not guarantee that the explicitly rated items will be ranked correctly. Non-functional properties of web services are considered as a multi-criteria mechanism that considers multiple non-functional properties as different possible dimensions of ranking. The algorithm proposed in [4] takes into account the associated importance of non-functional properties from the perspective of the users. Because the consumers' QoS requirements are imprecise, uncertain, or ambiguous, user's preferences over some criteria are difficult to be quantified. Almulla et al. [5] proposed a fuzzy model for ranking real-world web services.

In order to obtain service ranking, several researchers have also proposed methods for service classification. Makhlughian et al. [6] presented a method for classifying candidate web services according to different QoS levels, with respect to the requirements and preferences of the user. Their method uses an associative classification algorithm and then ranks the most qualified candidate services based on their functional quality through semantic matching. An approach for automatically classifying services was proposed in [7]. This method is based on the Rocchio algorithm, and each service is considered as a separate document. Text mining and machine-learning techniques have been used for service classifications. The work [8] proposed a knowledge-based solution to the problem by using fuzzy expert systems. Well-known classifiers such as k-nearest neighbor classifiers, probabilistic neural networks, naïve Bayes classifiers, classification and regression trees, TreeNet, decision trees, rough sets, and support vector machines have all been used to classify real-world web services based on their performance [9-12]. However, the authors of these proposals do not specify the details in most of their experiments, especially when using training sets of different sizes. Thus, the overall classification accuracy of these methods is difficult to determine. In this paper, we propose a solution to this problem by using a BP neural network.

In practical applications, the standard BP algorithm cannot meet the demands of several problems. Neural networks are relatively inefficient when used to solve complex problems. Therefore, to enhance the performance of a BP network, numerous scholars have proposed training algorithms with a rapid training speed, a global optimal solution, and improved generalization performance. These have been the main objectives when evaluating training algorithms in recent years. To achieve these objectives, several meta-heuristic algorithms have been put forward. Huang et al. [13] presented an iteration-optimization approach for integrating BP neural network with a genetic algorithm, and they employed four strategies for dealing with the possible deficiency of the prediction accuracy resulting from few training patterns. Experiments showed that this method avoids becoming trapped at a local optimum. To improve the search ability, two hybrid algorithms combining two improved particle-swarm optimization algorithms individually with the BP neural network were proposed to train single-hidden-layer feed-forward neural networks in [14]. Yi et al. [15] presented an improved BP network optimized by the Cuckoo search algorithm. Their proposal involved using the Cuckoo search algorithm to simultaneously optimize the initial weights and biases of a BP network. In this paper, an improved algorithm is proposed based on [15]. Since the Cuckoo search algorithm is limited by a slow convergence rate and poor accuracy. So, there is some space for improving its performance when optimizing a BP neural network. We propose a hybrid algorithm based on both the Cuckoo search algorithm and the Nelder-Mead simplex algorithm for BP optimization. The global convergence of the derived algorithm is greatly improved by combining the advantages of these two algorithms.

3 Preliminary

Service classification is essentially a problem of multi-target recognition. BP neural networks are one of the most popular methods for multi-target recognition. However, BP networks have several shortcomings, and in particular they are limited by a slow convergence rate. In order to work around this problem, we propose to use the Cuckoo search algorithm and the Nelder-Mead simplex algorithm to optimize a BP neural network. The specifics of each algorithm are described as follows.

3.1 BP neural network

The BP neural network algorithm is a multi-layer feed-forward network trained with an error-BP algorithm. BP networks are among the most widely applied neural network models. They can be used to learn and store a large number of input-output model mapping relations, and there is no requirement to disclose in advance the mathematical equations that describe these mapping relations. Its learning rule employs the steepest-descent method, in which back propagation is used to achieve the minimum error sum of the square by regulating the weight value and a threshold value for the network. The application of the standard BP network model is converted to a mathematical optimization problem. In other words, the input-output problem of training samples is transformed into a non-linear mathematical optimization problem. As such, the non-linear mapping ability of a BP neural network is considerably strong. BP learning algorithms employ a global optimization approach, which has good generalization ability and resilient fault tolerance. The BP neural network is an important tool for

investigating classification problems, owing to its robust learning ability.

3.2 Cuckoo search algorithm

The Cuckoo search algorithm is a novel meta-heuristic swarm-intelligence optimization algorithm for solving optimization problems [16]. It is inspired by the obligate brood parasitism of some species of cuckoo birds, which lay their eggs in the nests of other host birds (i.e., other species). Some host birds can engage in direct competition with these encroaching cuckoos. For example, if a host bird finds that the eggs are not its own, it will either discard these alien eggs or simply give up its nest, in favor of building a new nest elsewhere. Some cuckoo species, such as the New World brood-parasitic Tapera have evolved in such a way that female parasitic cuckoos are often very specialized at mimicking the colors and patterns of the eggs for some chosen host species.

In addition, the timing of the egg-laying of some species is also critical. Parasitic cuckoos often choose a nest in which the host bird just laid its own eggs. In general, cuckoo eggs hatch slightly earlier than their host eggs. After the first cuckoo chick hatches, the first instinctual action of the host bird is to evict the host eggs so as to increase the amount of food provided by the host bird for the cuckoo chick. A cuckoo chick can mimic the call of the host chicks to gain access to more feeding opportunities.

For a simple description of the Cuckoo search algorithm, the author of [17] presents the following three assumptions:

- Each cuckoo lays one egg at a time and dumps its egg in a randomly chosen nest;
- The best nests with the highest quality of eggs will continue to the next generation;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability of $P_a \in [0,1]$. In this case, the host bird can either discard the egg or give up the nest so as to build a completely new nest in a new location.

When generating new solutions $x^{(t+1)}$ for cuckoo i, a Lévy flight [17] is performed: $x_i^{(t+1)} = x_i^t + \alpha \oplus \text{Le}'\text{vy}(\lambda)$, where $\alpha > 0$ is the step size that should be related to the scale of the problem at hand. In most cases, we can use $\alpha = O(1)$. The product $\oplus$ refers to entry-wise multiplications. Essentially, Lévy flights provide a random walk, whereas their random steps are drawn from a Lévy distribution for large steps: $Le'vy \sim u = t^{-\lambda}, (1 < \lambda \le 3)$. This distribution has an infinite variance with an infinite mean. Here, the consecutive jumps or steps of a cuckoo bird essentially form a random walking process that obeys a power-law step-length distribution with a heavy tail.

The Cuckoo search algorithm is used to optimize the BP neural network's initial weights and biases in order to improve the performance of the BP neural network [15]. However, the Cuckoo search algorithm is limited by a slow convergence rate and poor accuracy. Thus, there is some space for improving its performance when optimizing a BP neural network.

3.3 Nelder-Mead simplex method

The Nelder-Mead simplex method is a commonly used non-linear optimization technique. It is a well-defined numerical method for problems wherein derivatives may be unknown. However, the Nelder-Mead technique is a heuristic search method that can converge to non-stationary points for problems that can be resolved using alternative methods. The method uses the concept of a simplex, which is a special polytope of *N+1* vertices in *N* dimensions. Examples of simplexes include a line segment on a line, a triangle on a plane, a tetrahedron in three-dimensional space, and so forth.

The Nelder-Mead technique generates a new test position by extrapolating the behavior of the objective function measured at each test point arranged as a simplex. The algorithm then opts to replace one of these test points with the new test point, and so the technique progresses. By comparing the value of the objective function of the simplex (*N+1*) vertices, then the point with the worst objective function value will be replaced with the new point. The simplex constantly being updated through the gradual iteration, and the final simplex will be close to the optimal solution. The Nelder-Mead simplex method is described as follows [18]:

Step 1 (Initialization): For the minimization problem of the non-constrained function of *n* variables, $x_0, x_1, \cdots, x_n$ are *n+1* points on the *n* dimensional space. Constitute the initial "simplex", and calculate the value of the function at each vertex of the simplex.

Step 2 (Order): Order the points according to the values at the vertices: $f(x_1) \le f(x_2) \le \cdots \le f(x_{n+1})$. And calculate X_0, the centroid of all points except X_{n+1}.

Step 3 (Reflection): Compute the reflected point, $X_r = X_0 + \alpha(X_0 - X_{n+1})$. If the reflected point is better than the second-worst point, but not better than the best—i.e., $f(X_1) \le f(X_r) < f(X_n)$. Then, obtain a new simplex by replacing the worst point X_{n+1} with the reflected point X_r, and return to Step 2.

Step 4 (Expansion): If the reflected point is the best point so far—that is, $f(X_r) < f(X_1)$ —then compute the expanded point $X_e = X_0 + \gamma(X_0 - X_{n+1})$. If the expanded point is better than the reflected point $f(X_e) < f(X_r)$, then obtain a new simplex by replacing the worst point X_{n+1} with the expanded point X_e, and return to Step 2. Otherwise, proceed to Step 5 (i.e., if the reflected point is not better than second-worst point).

Step 5 (Contraction): If it is certain that $f(X_r) \ge f(X_n)$, then compute the contracted point: $X_c = X_0 + \rho(X_0 - X_{n+1})$. If this contracted point is better than the worst point, i.e. $f(X_c) < f(X_{n+1})$, then obtain a new simplex by replacing the worst point X_{n+1} with the contracted point X_c, and return to Step 2. Otherwise, proceed to step 6.

Step 6 (Reduction). For all but the best point, replace the point with $X_i = X_1 + \sigma(X_i - X_1)$ for all $i \in \{2, \cdots, n+1\}$. Return to Step 2.

The Nelder-Mead simplex algorithm is simple and easy to implement, and it has relatively lower resolution requirements for the analytic properties of objective function.

To improve the local search ability of the algorithm proposed in [15]—that is, the BP network enhanced with a Cuckoo search algorithm (hereafter, the "CSBP")—we propose a hybrid optimization based on both the Cuckoo algorithm and the Nelder-Mead simplex method for BP optimization (hereafter "CN"). Our proposed CNBP method effectively overcomes the shortcomings of both the Cuckoo algorithm and the Nelder-Mead method. By combining the advantages of these two algorithms, the global convergence of the derived algorithm is greatly improved.

The objective of our proposed CNBP is to optimize the BP neural network. With the CNBP, the BP neural network is optimized with the hybrid CN algorithm. Specifically, the BP neural network is considered as the objective function of the CN. In order to obtain the most appropriate weights and biases for initializing the BP network, they are first encoded and then optimized with the hybrid algorithm.

4 Our Service-classification Approach

The CNBP algorithm is designed to solve service-classification problems, with the following main steps. (1) The structure of BP network is determined according to the input vector and the output vector of the problem. Then, we determine the number of input layers, output layers, and hidden layers. (2) The BP neural network is trained with training data. We obtain the weight vector and bias vector according to the nodes of the input, output, and hidden layers. All of the weights and biases are encoded together to determine the structure of each individual in the CN. The Cuckoo search algorithm implements an initial population to determine and calculate the fitness function. According to the value of the fitness function, individuals in the population are arranged in ascending order. The first two individuals are directly entered into the next cycle and the top quarter of the population is optimized with the Nelder-Mead method. Then, the entire population is optimized with the Cuckoo search algorithm. The results of these three parts are assembled together and arranged in ascending order. The top individuals, whose number equals the population size, are assembled together as a new population. In order to find the individual with the best level of fitness, this optimization process is repeated until satisfactory weights and biases are found. (3) Finally, the trained BP neural network is used to classify the services.

Figure 1 shows a flowchart for the proposed CNBP. The proposal contains to basic components: the hybrid CN algorithm and the BP neural network. The CN algorithm is used to optimize the initial weights and biases of the BP neural network. It comprises five tasks: initializing the CN, determining the fitness function, applying the Cuckoo search algorithm, applying the Nelder-Mead algorithm, and assembling the results as a temporary group. A detailed description of these five tasks is presented in the following

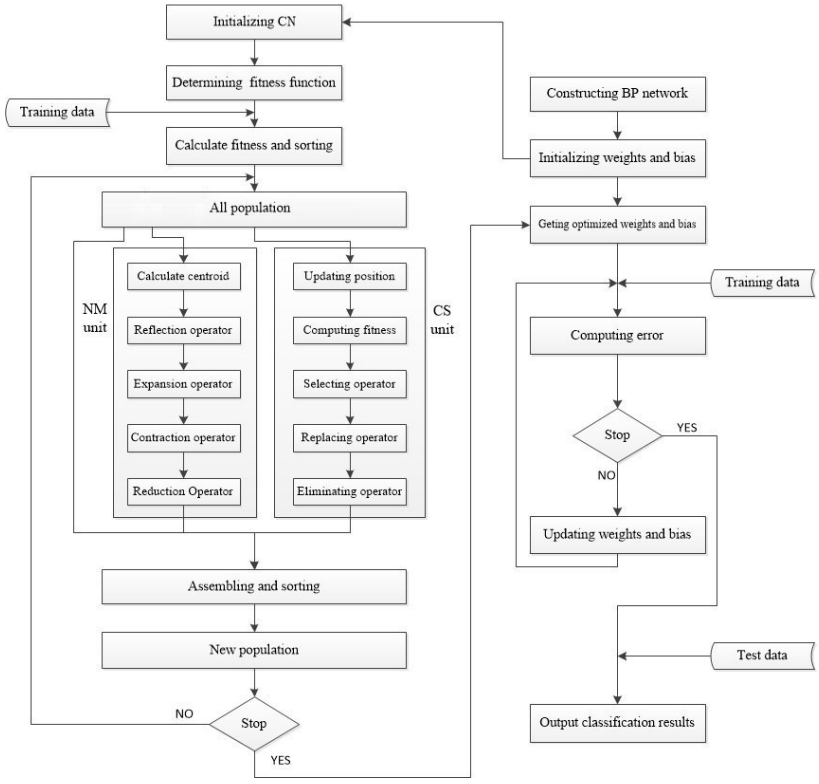

Figure 1. Flowchart for the proposed CNBP

subsections.

4.1 Initializing the CN

When the BP neural network is constructed, we can determine the number of weights and biases based on the number of input layers, output layers, and hidden layers. Then, the connection weights between the input layer and the hidden layer, the connection weights between the hidden and output layers, and the bias of the hidden and output layers are encoded together as an individual for the Cuckoo algorithm.

4.2 Determining the fitness function

The optimization goal is to find the optimal weights and biases with which the BP neural network can achieve a suitable performance when making predictions. In other words, the final goal is to find the set of weights and biases that minimizes the global sum of the absolute errors between the desired output and the predicted output. We define the fitness function as follows:

$$F = \frac{1}{2m}\sum_{k=1}^{m}\sum_{o=1}^{q}(d_o(k) - y_o(k))^2 \qquad (1)$$

where F is the value of the fitness function, m is the number of samples, q is the number of output layers, and $d_o(k)$ and $y_o(k)$ denote the desired output and the predicted output of the k-th sample, respectively, for node o in the BP network.

4.3 Applying the Cuckoo algorithm

After calculating the fitness value of all individuals and sorting the population according to this fitness value, all of the individuals are optimized with the Cuckoo search algorithm. To do so, the algorithm proceeds as follows: the position is updated, the fitness is computed, and the individuals are selected, replaced and eliminated. Subsequently, we can obtain a new population with n individuals, where n is the size of the original population.

4.4 Applying the Nelder-Mead algorithm

Some of the individuals (the number of individuals is approximately $b = n/4$) with a lower fitness value are removed from the population and optimized with the Nelder-Mead algorithm. This task involves four basic operations: reflection, expansion, contraction, and reduction (see Sec. 3.3, above). As a result, superior points are obtained near the best point, and the worst point is replaced by a better point. Moreover, the optimal point is found in the correct direction. Thus, b individuals are generated from this process.

4.5 Assembling the results in a temporary group

A temporary group is then assembled after the above tasks are complete. This group comprises three parts: the best two individuals from the initial population, the b individuals that result from applying the Nelder-Mead algorithm, and the n individuals from the Cuckoo search. Thus, the temporary group contains t individuals ($t = 2+b+n$). In order to generate the new population, all of the individuals are sorted in ascending order, and the top n are selected as the new population.

Based on the above description, the basic steps for the CNBP are summarized in Algorithm 1. These steps are executed repeatedly until the optimal point is found or until the iterations terminate. The optimal weights and biases are then used to construct the BP neural network. The service data can be classified with the BP network that has been trained with the training data. Because the initial weights and biases of the BP neural network are optimized with the CN algorithm, the BP neural network is expected to offer an improved classification performance.

Algorithm 1 CNBP

Begin
Step 1 Initialization BP, with weights w1, w2, and biases b1, b2.
Step 2 Training BP neural network with training data
  2.1 Initializing the CN
    $x = (x_1, x_2, \ldots, x_n)^T = ((w_1{}^T, w_2{}^T, b_1{}^T, b_2{}^T))^T$
  2.2 Calculate fitness function f(x),
    Generate initial population of n host nests x_i
    While $(t \leq MaxGeneration)\&(f(x) \geqslant goal)$ do
    Sort all the cuckoos
    Chose the best a cuckoos from x, denoted y_a
    Chose the best b cuckoos from x, denoted z_b
  2.3 Execute NM algorithm to z
    Calculate the centroid z_0
    Compute reflected nest z_r
    If $f(z_1) \leq f(z_r) < f(z_n)$,replace $z_{n+1}$with z_r
    Else If$f(z_r) < f(z_1)$,compute the expanded nest z_e
      If$(f(z_e) < f(z_r)$,replace z_{n+1} with z_e
      Else Compute contracted nest: z_c
        If $(f(z_c) < f(z_{n+1})$,replace z_{n+1} with z_e
        Else for all but the best nest,
        replace the nest with
        $z_i = z_1 + \sigma(z_i - z_1)$
        End
      End
    End
  End
  2.4 Execute CS algorithm to x
    Randomly select a cuckoo i
    implement Levy flights to replace its solution
    Evaluate its fitness $f(x_i)$
    Randomly choose another nest j
    If $f(x_i) < f(x_j)$,replace j by the new solution
    A fraction p_a of the worse nests are abandoned and new ones are built
    Keep the best solutions (or nests with quality solutions)
  2.5 Assembling $X = (x, y, z)$and sorting by f(X)
    Taken out top n to the next generation
  End while
  Output optimal results and decode them
Step 3 classification by trained BP network
End

5 Experiment

In this section, we describe several experiments that we conducted on a real-world dataset. In order to verify the effectiveness of our proposed algorithm, we compared our approach with several commonly used classification methods.

5.1 Experimental setup

We implemented experiments employing MATLAB 8.3 on an IBM server with an Intel Xeon E5-2670 eight-core 2.60-GHz CPU and 32 GB of RAM.

The publicly available Quality of Web Services dataset (QWS) [19-21] was used for classifying services with our proposed CNBP algorithm. As illustrated in Table 1, each web service includes nine quality attributes that describe the QoS in a particular area. Services in the QWS dataset were classified into four categories: platinum (high quality), gold, silver, and bronze (low quality). This classification was based on the overall quality rating provided by Web Services Resource Framework (WSRF). Each web service

was grouped into a specific service group. This classification is helpful for differentiating between various services that offer the same functionality. In the experiment, we used a sample experimental dataset consisting of 364 functionally similar web services. As described in Table 1, the last dimension represents a class identifier, and the first nine represent the attributes of the respective web services. Among these 364 samples, Samples 1-41, 42-141, 142-261, and 262-364 belonged to the first, second, third, and fourth category, respectively. Each category was then divided into training data and testing data.

Table 1 QWS parameters

ID	Attribute	Description	Units
1	Response Time	Time taken to send a request and receive a response	ms
2	Availability	Number of successful invocations/total invocations	%
3	Throughput	Total Number of invocations for a given period of time	invokes/s
4	Success ability	Number of response / number of request messages	%
5	Reliability	Ratio of the number of error messages to total messages	%
6	Compliance	The extent to which a WSDL document follows WSDL	%
7	Best Practices	The extent to which a Web service follows WS-I Basic	%
8	Latency	Time taken for the server to process a given request	ms
9	Documentation	Measure of documentation (i.e. description tags) in WSDL	%
10	WsRF	Web Service Relevancy Function: a rank for Web	%
11	Service Classification	Levels representing service offering qualities (1 through 4)	Classifier

5.2 Performance of our algorithm

In order to demonstrate that our proposal offers a significant improvement, it was compared with CSBP and traditional BP. According to the number of services, attributes, and categories, the neurons for the input, output, and hidden layers were set as 9, 4, and 8, respectively. The four neurons in the output layer represent four species, which can be expressed as [1 0 0 0], [0 1 0 0], [0 0 1 0], and [0 0 0 1]. Here, each code represents a category. The length of the encoded string for each individual was 116, according to the following equation: 9×8+8×4+8+4=116. Because the neural network uses a logistic function, the input range must be within 0 and 1. The property values must be normalized for all of the samples in the QWS dataset. In this experiment, all the parameters were set as follows. For the BP network, the learning rate was $lr = 0.2$ and the training error was $goal = 0.0001$. For the CSBP, the BP network used the same parameters as the basic BP, and the Cuckoo algorithm had a discovery rate of $p_a = 0.2$ with a population size of NP=50 and maximum generations of $Maxgen = 10$. For the CNBP, the Cuckoo algorithm used the same parameters as the CSBP, and the Nelder-Mead technique had a reflection coefficient of α=1, an expansion coefficient of γ=2, a contraction coefficient of β=0.5, and a reduction coefficient of σ=0.5. Each algorithm was run 100 times and we recorded the average results.

In the first experiment, differently sized training samples were selected for classification. Between 10% and 90% of the entire dataset was used for training. For each algorithm, the training time was set as *epochs*=10. As shown in Figure 2, the classification accuracy gradually improved with an increased number of training samples. Moreover, the classification accuracy of the proposed CNBP was significantly higher than the other two, especially when the training set was small. When only 10% of the dataset was used for training, the classification accuracy of CNBP was 67.3%, whereas it was a mere 61.4% and 55.6% for CSBP and BP, respectively. The reason for this is that the hybrid CN algorithm converged faster and with fewer errors. With an increase in training samples, more information can be used. Consequently, the classification accuracy of the other two algorithms increased significantly (BP: 93.4%, and CSBP: 94.6%). Similarly, the classification accuracy of the proposed CNBP increased significantly (96.7%), again outperforming the other two.

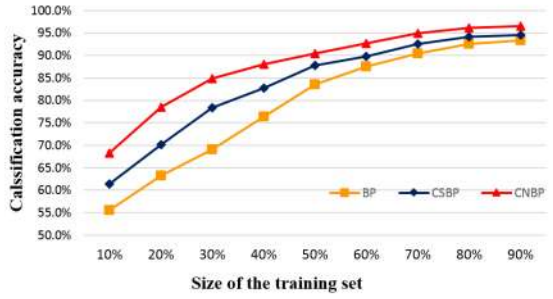

Figure 2. Classification accuracy with different training sets

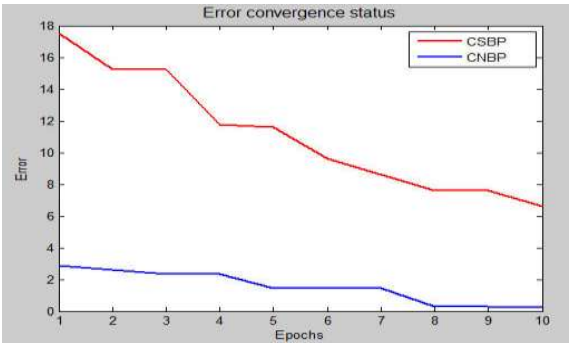

Figure 3. Error convergence status of CSBP and CNBP

Figure 3 shows the error-convergence status of the Cuckoo search algorithm and the hybrid CN algorithm,

Because there were too many initial errors to describe them easily in the figure, the number of errors was recorded only after the first step in the optimization. After the initial optimization, there were far fewer initial errors with the CN than with the Cuckoo search algorithm. Furthermore, the error convergence with CN was very fast during the entire optimization process. The hybrid algorithm converged quickly and with few errors (indeed coming close to 0). The errors from the Cuckoo search algorithm converged slowly (from 18 to 6, approximately). Therefore, the CN is significantly better than the Cuckoo search algorithm in terms of its optimization capability and effect.

With better initial weights and biases to guide the BP network, the performance will be improved. Because the initial weights and biases for a traditional BP network are random, its effect was demonstrably worse. We also found that the gap in the classification accuracy of different algorithms became gradually narrower as the number of training samples increased. This means that the CNBP algorithm has an obvious advantage in cases where there are few training samples.

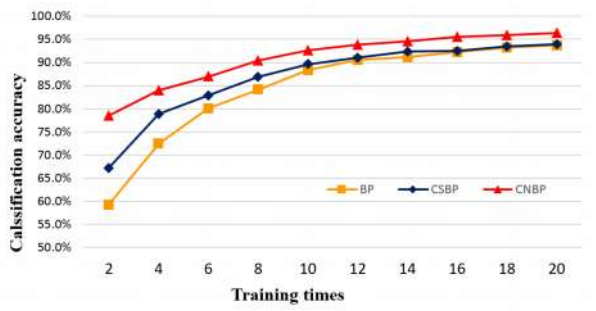

Figure 4. Classification accuracy with different training times

The second experiment involved analyzing the impact of the number of training times on the classification accuracy. For this experiment, we trained the classifier between 2 and 20 times (i.e., we reiterated Step 2 in Algorithm 1 between 2 and 20 times), using 60% of the dataset for training. The classification accuracy with different training times is shown in Figure 4. The CNBP's classification accuracy was again the highest of the three methods. Its superiority is obvious, especially with fewer training times. When training the classifier only twice, for instance, the proposed algorithm was approximately 78% accurate. By contrast, the Cuckoo search algorithm was somewhat accurate, at approximately 67%, and the traditional BP fell below 60% accuracy, which is a considerably inferior.

The BP algorithm did not have a sufficient number of training iterations to guide its subsequent classifications. Therefore, it could not achieve the desired results without ample training. The CSBP also performed poorly, despite optimized weights and biases. Its performance was significantly affected by a slow error convergence. The proposed CNBP, on the other hand, used a hybrid optimization algorithm to find the optimal weights and biases, resulting in a superlative performance. Thus, it is the most stable of the three methods, even when trained only a few times. Nevertheless, as the number of training times increased (e.g., with more than 10), the advantages of our optimized BP network were not especially obvious. For example, with 20 training times, the classification accuracy of the proposed CNBP was 96.3%, whereas it was 93.9% and 93.7% for the CSBP and BP, respectively.

In the second experiment, the difference in classification time resulting from different training times is shown in Figure 5. The classification time increased as the classifiers were subjected to more training, and the classification time of each method was not significantly different. When the classifiers were trained twice, the improved BP algorithm indeed higher than that of the traditional BP algorithm. The reason for this is that optimizing the initial weights and biases requires considerable time. The classification time for the CSBP was not especially stable, owing to the search mechanism of the Cuckoo search algorithm. On the whole, the difference between all three algorithms is nominal. Figure 6 illustrates the classification accuracy for each class, when the classifiers were each trained four times. The classification accuracy of each class is consistent with the overall classification accuracy. The proposed CNBP's classification accuracy for each class was significantly higher than the other two. The subtle differences between the various types are due to blur and the randomness of the samples.

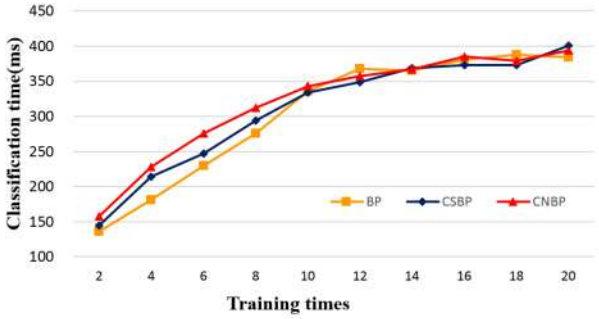

Figure 5. Classification time of different training times

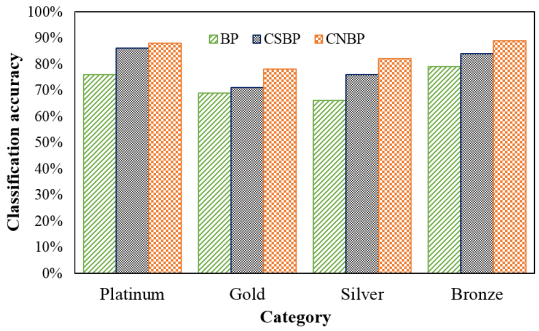

Figure 6. Classification accuracy of each category

5.3 Comparison with other algorithms

In order to evaluate the performance of the proposed CNBP, we compared it with a naïve Bayes (NB) algorithm [11], a k-nearest neighbor (KNN) algorithm [12], and a

support vector machine (SVM) [12]. In this experiment, we used three metrics [10]: accuracy, precision, and recall.

The accuracy refers to the proportion of correct predictions among the total number of predictions. It is defined as follows:

$$Accuracy = \frac{\sum_{i=1}^{k} n_{ii}}{\sum_{i=1,\, j=1}^{k} n_{ij}} \tag{2}$$

The precision is the proportion of true positives against all of the positive results (i.e., both true positives and false positives). It is defined as follows:

$$Precision(C_i) = \frac{n_{ii}}{\sum_{j=1}^{k} n_{ji}} \tag{3}$$

The recall is the proportion of positive cases that were correctly identified, and this metric is defined as follows:

$$Recall(C_i) = \frac{n_{ii}}{\sum_{j=1}^{k} n_{ij}} \tag{4}$$

In the above three formulas, the *Accuracy* represents the value of the classification accuracy, the $Precision(c_i)$ is the precision of the classification for a certain class i, and the $Recall(c_i)$ indicates the recall rate for the classification of a certain class i. Furthermore, in these formulae, k is the number of services to be classified, n_{ij} indicates the number of services where the correct QoS value is i, and the predicted result is j.

For each algorithm, we performed ten-fold cross-validation. For the CNBP, we set the number of epochs at 10, and the remaining parameters were the same as they were for the previous experiments. The parameters for the three contrastive algorithms were kept the same as they were in their respective publications (viz., [11] and [12]). As can be seen in Table 2, the total classification accuracy of the proposed CNBP was significantly higher than the other three algorithms (at more than 98%). The classification accuracy of the other classifiers was as follows: NB: 88.1%; SVM: 81.4%; and KNN: 68.3%. Moreover, the classification accuracy of each class also confirmed the advantages of the proposed method.

Table 2 Classification accuracy of different algorithms

Algorithms	NB	KNN	SVM	CNBP
Accuracy	88.1%	68.3%	81.4%	98.4%

Likewise, the proposed CNBP exhibited higher values than the other three classification algorithms in terms of precision and recall, as shown in Figures 7 and 8, respectively. For each QoS level, the classification precision and recall for the CNBP was stable and maintained equilibrium. However, because of the complex training process, the proposed CNBP required more training time than the other three algorithms. Therefore, the proposed algorithm is somewhat more time consuming. In general, however, a web-services classifier is trained offline and does not require continuous training. Thus, service classification itself does not consume additional time.

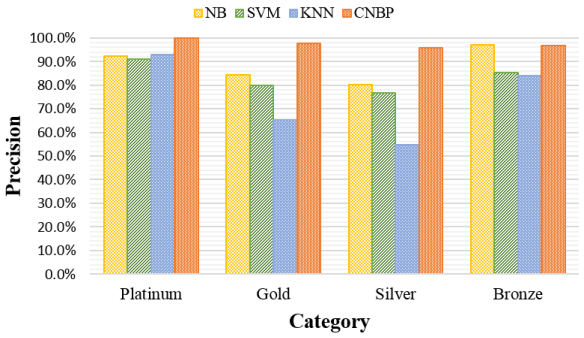

Figure 7. Precision of each classification algorithm

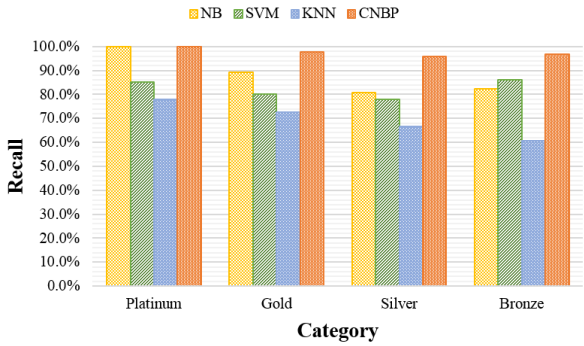

Figure 8. Recall of each classification algorithm

6 Discussion

The classification performance of the proposed CNBP was evaluated in the previous section. Because the initial weights and biases of CNBP are optimized by a hybrid algorithm, our CNBP outperformed all other algorithms in terms of classification accuracy, precision, and recall. In addition, CNBP has good ability of self-learning and adaptive, and strong robustness, so it is a good solution for service classification. However, our algorithm also has some shortcomings:

- There are too many parameters need to be set in CNBP, so it is not easy to get the best performance;
- Because our CNBP algorithm is complex and time consuming, it is not conducive to real-time classification, but only for offline classification.

In summary, we provide a feasible algorithm for service classification. However, there is some room for improving the classification accuracy and reducing the classification time.

7 Conclusion

In order to address the problem of service classification, this paper proposed the CNBP algorithm. With this proposal, the weights and biases of a BP network are

optimized with a hybrid optimization based on the Nelder-Mead simplex method and the Cuckoo search algorithm. This hybrid algorithm effectively overcomes the shortcomings of both algorithms. By combining the two algorithms, the global convergence is greatly improved. Moreover, by using this algorithm, services can be effectively classified into different service-quality levels.

We conducted a series of experiments based on the QWS dataset, demonstrating that this algorithm is more effective and stable than the CSBP and traditional BP for service classification. As described in the experiment, the error convergence was considerably fast during the entire optimization process with the hybrid algorithm, and it converged with few errors. With better initial weights and biases to guide the BP network, the proposed CNBP achieved a superior performance. A comparative analysis with three other classification algorithms demonstrated that the proposed CNBP outperformed the other method in terms of its classification accuracy, precision, and recall. Despite these results, however, there is some room for improving the classification accuracy and the training time of the proposed method.

In future research, we shall focus on improving the algorithm itself, and we will compare it with other algorithms. Moreover, we will attempt to design a service classifier that is more accurate and less time consuming. Based on the results of this future research, we hope to offer contributions in the field of service recommendation.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61202435 and 61272521, the Natural Science Foundation of Beijing under Grant No. 4132048, and SRFDP (20110005130001). Shangguang Wang is the corresponding author.

References

[1] Ran, Shuping, A model for web services discovery with QoS, *ACM Sigecom exchanges,* Vol. 9, No. 1, 2003, pp.1-10.

[2] Yau, Stephen S., and Yin Yin, Qos-based service ranking and selection for service-based systems, *Proc. IEEE Services Computing*, Washington, United States, July, 2011, pp.56-63

[3] Zheng Zibin, Yilei Zhang, and Michael R. Lyu, CloudRank: A QoS-Driven Component Ranking Framework for Cloud Computing, *Proc. Reliable Distributed Systems,* New Delhi, Punjab, India, Oct., 2010, pp.184-193.

[4] Toma, Ioan, Roman, Dumitru, Fensel, Dieter, Sapkota, Brahmanada, and Gomez, Juan Miguel, A Multi-criteria Service Ranking Approach Based on Non-Functional Properties Rules Evaluation, *Proc. International conference on Service-Oriented Computing,* Vienna, Austria, Sep., 2007, pp.435-441.

[5] Almulla, Mohammed, Kawthar Almatori, and Hamdi Yahyaoui, A qos-based fuzzy model for ranking real world web services, *Proc. IEEE Web Services,* Washington, USA, July, 2011, pp.203-210.

[6] Makhlughian, Molood, Seyyed Mohsen Hashemi, Yousef Rastegari, and Emad Pejman, Web service selection based on ranking of qos using associative classification, *International Journal on Web Service Computing,* Vol. 9, No. 1, 2012, pp.1-14.

[7] Crasso, Marco, Alejandro Zunino, and Marcelo Campo, Awsc: An approach to web service classification based on machine learning techniques, *Revista Iberoamericana de Inteligencia Artificial* , Vol. 12, No. 37, 2008, pp.25-36.

[8] Almulla, Mohammed A., On Classification of Web Services Using Fuzzy Expert System, *Journal of Algorithms & Computational Technology,* Vol. 6, No. 4, 2012, pp.673-686.

[9] Sonawani, Shilpa, and Debajyoti Mukhopadhyay, A Decision Tree Approach to Classify Web Services using Quality Parameters, *Proc. The International Conference on Web Engineering and Application,* Bhubaneshwar, India, Dec., 2013, pp.1-9.

[10] Own, Hala S., and Hamdi Yahyaoui, Rough set based classification of real world Web services, *Information Systems Frontiers* , 2014, pp.1-11.

[11] Mohanty, Ramakanta, V. Ravi, and M. R. Patra, Classification of Web Services Using Bayesian Network, *Journal of Software Engineering and Applications* ,Vol. 5, No. 4, 2012, pp.291-296.

[12] Mustafa, A. Syed, and Y. S. Kumaraswamy, Performance Evaluation of Web-services Classification, *Indian Journal of Science and Technology*, Vol. 7, No. 10, 2014, pp.1674-1681.

[13] Huang, Han-Xiong, Jiong-Cheng Li, and Cheng-Long Xiao, A proposed iteration optimization approach integrating back propagation neural network with genetic algorithm, *Expert Systems with Applications*, Vol. 42, No. 1, 2015, pp.146-155.

[14] Han, Fei, and Jian-Sheng Zhu, Improved Particle Swarm Optimization Combined with Backpropagation for Feedforward Neural Networks, *International Journal of Intelligent Systems*, Vol. 28, No. 3, 2013, pp.271-288.

[15] Yi, Jiao-hong, Wei-hong Xu, and Yuan-tao Chen, Novel Back Propagation Optimization by Cuckoo Search Algorithm, *The Scientific World Journal*, 2014, doi:10.1155/2014/878262

[16] Gandomi, Amir Hossein, Xin-She Yang, and Amir Hossein Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, *Engineering with Computers*, Vol. 29, No. 1, 2013, pp.17-35.

[17] Yang, Xin-She, and Suash Deb, Cuckoo search via Lévy flights, *Proc. World Congress on Nature & Biologically Inspired Computing,* Coimbatore, India, Dec., 2009, pp.210-214.

[18] Singer, Saša, and John Nelder, Nelder-mead algorithm, *Scholarpedia*, Vol. 4, No. 7, 2009, pp.2928.

[19] Al-Masri, Eyhab, and Qusay H. Mahmoud, Discovering the best web service, *Proc. International*

Conference on World Wide Web, May, 2007, pp. 1257-1258.

[20] Al-Masri, Eyhab, and Qusay H. Mahmoud, QoS-based Discovery and Ranking of Web Services, *Proc. International Conference on Computer Communications and Networks*, Hawaii, United States, Aug., 2007, pp.529-534.

[21] Al-Masri, Eyhab, and Qusay H. Mahmoud, Investigating Web Services on the World Wide Web, *Proc. International Conference on World Wide Web*, Beijing, China, Apr., 2008, pp.795-804.