Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 325192, 15 pages
http://dx.doi.org/10.1155/2015/325192

Research Article

Improving ELM-Based Service Quality Prediction by Concise Feature Extraction

Yuhai Zhao, Ying Yin, Gang Sheng, Bin Zhang, and Guoren Wang

College of Information Science and Engineer, Northeastern University, Shenyang 110819, China

Correspondence should be addressed to Yuhai Zhao; zhaoyuhai@ise.neu.edu.cn

Received 20 August 2014; Revised 10 November 2014; Accepted 12 November 2014

Academic Editor: Jiuwen Cao

Web services often run on highly dynamic and changing environments, which generate huge volumes of data. Thus, it is impractical to monitor the change of every QoS parameter for the timely trigger precaution due to high computational costs associated with the process. To address the problem, this paper proposes an active service quality prediction method based on extreme learning machine. First, we extract web service trace logs and QoS information from the service log and convert them into feature vectors. Second, by the proposed EC rules, we are enabled to trigger the precaution of QoS as soon as possible with high confidence. An efficient prefix tree based mining algorithm together with some effective pruning rules is developed to mine such rules. Finally, we study how to extract a set of diversified features as the representative of all mined results. The problem is proved to be NP-hard. A greedy algorithm is presented to approximate the optimal solution. Experimental results show that ELM trained by the selected feature subsets can efficiently improve the reliability and the earliness of service quality prediction.

1. Introduction

The advantage of composite Web services is that it realizes a complex application by connecting multiple component services seamlessly. However, in real applications, Web service lives in a highly dynamic environment, and both the network condition and the operational status of each of the component Web services (WSs) may change during the lifetime of a business process itself. The instability brought by various uncertain factors often makes the composite services failed or interrupted temporally. Therefore, it is very important to ensure the normal execution of the composite service applications and provide a reliable software system [1].

As one of the promising technologies to address the above issue, Web service quality prediction has become an important research problem and has attracted a lot of attention in recent years. The goal is to perceive in advance whether the invoked services will fail or be interrupted by monitoring and evaluating service quality fluctuation. In SOA infrastructure, Web service prediction aims to optimally select the high quality service in advance to ensure the reliable execution of system. A number of Web service prediction models have been proposed, such as ML-based methods [2–4], QoS-aware based methods [5], and collaborative filtering-based methods [6, 7]. These models are often implemented by monitoring and evaluating the quality of composite services. In spite of improving the quality of composite services to some extent, there methods still have three major drawbacks. First, most of the traditional ML-based prediction models [3], such as support vector machines (SVM) and artificial neural networks (ANN), are more sensitive to the user-specified parameters. Second, the prediction models based on QoS monitoring, such as Naive Bayes and Markov model [8, 9], often assume that sequences in a class are generated by an underlying model M and the probability distributions are described by a set of parameters. However, these parameters are obtained by predicting QoS during the whole lifecycle of the services and will therefore lead to high overhead costs. In another sequence distance based prediction method, such as collaborative filtering [6, 7], a function measuring the similarity between a pair of sequences is necessary. However, how to select an optimal similarity function is far from trivial, as it will introduce numerous parameters and measures for distances which may be rather subjective.

As a powerful prediction model, extreme learning machine (ELM for short) was originally developed based on single-hidden layer feedforward neural networks (SLFNs) in [10]. Compared with the conventional learning machines, it is of extremely fast learning capacity and good generalization capability. Thus, ELM, with its variants, has been widely applied in many fields. For example, in [11], ELM was applied for plain text classification by using the one-against-one (OAO) and one-against-all (OAA) decomposition scheme. In [12], an ELM-based XML document classification framework was proposed to improve classification accuracy by exploiting two different voting strategies. A protein secondary structure prediction framework based on ELM was proposed in [13, 14] to provide good performance at extremely high speed. References [15, 16] evaluated the multicategory classification performance of ELM on three microarray datasets. The results indicate that ELM produces comparable or better classification accuracies with reduced training time and implementation complexity compared to artificial neural networks methods and support vector machine methods.

In this paper, we introduce ELM into Web service QoS prediction. To our best knowledge, it has never been addressed by any previous work. However, it is not trivial to integrate ELM into Web services quality prediction. Some issues need further consideration, for example, how to model the execution information of Web services to facilitate the usage of ELM on the data and how to train ELM in as short time as possible to get a model of high prediction accuracy so that we can conduct an on-line Web service QoS prediction.

Our contributions include that (1) we devise a method to extract web service trace logs and QoS information from the service log and convert them into feature vectors; (2) we propose a concept, namely, EC rule, based on which we are enabled to trigger precaution as soon as possible with high confidence; (3) we develop an efficient prefix tree based mining algorithm together with some effective pruning rules to mine such rules; (4) we further study how to extract a set of diversified features as the representative of all mined results based on ELM.

The rest of this paper is organized as follows. Section 2 gives a brief overview of ELM. Section 3 presents ELM-based QoS prediction framework. Section 4 studies the feature vectors representation of Web services. Section 5 defines the EC rules and proposes the mining algorithm. In Section 6, we study the problem of diversified feature selection and present the greedy solution. In Section 7, the experimental evaluation results are reported. Finally, Section 8 concludes this paper.

2. A Brief Introduction to ELM

ELM (extreme learning machine) is a generalized single hidden-layer feedforward network. In ELM, the hidden-layer node parameter is mathematically calculated instead of being iteratively tuned; thus, it provides good generalization performance at thousands of times faster speed than traditional popular learning algorithms for feedforward neural networks [12].

Given N arbitrary distinct samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$, standard SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$f(x) = \sum_{i=1}^{L} \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}), \tag{1}$$

where $\mathbf{a}_i$ and b_i are the learning parameters of hidden nodes and β_i is the weight connecting the ith hidden node to the output node. $g(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of the ith hidden node with respect to the input x. In our case, sigmoid type of additive hidden nodes is used. Thus, (1) is given by

$$f(x) = \sum_{i=1}^{L} \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}) = \sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad (j = 1, \ldots, N), \tag{2}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the bias of the ith hidden node, and $\mathbf{o}_j$ is the output of the jth node [10].

If an SLFN with activation function $g(x)$ can approximate the N given samples with zero errors that $\sum_{j=1}^{L} \|\mathbf{o}_j - \mathbf{t}_j\| = 0$, there exist β_i, $\mathbf{a}_i$, and b_i such that

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \ldots, N. \tag{3}$$

Equation (3) can be expressed compactly as follows:

$$\mathbf{H}\beta = \mathbf{T}, \tag{4}$$

where

$$\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_L, b_1, \ldots, b_L, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L}, \tag{5}$$

$$\beta = \left[\beta_1^T, \ldots, \beta_L^T\right]_{m \times L}^T, \qquad \mathbf{T} = \left[\mathbf{t}_1^T, \ldots, \mathbf{t}_L^T\right]_{m \times N}^T.$$

$\mathbf{H}$ is called the hidden layer output matrix of the network. The ith column of $\mathbf{H}$ is the ith hidden nodes output vector with respect to inputs $x_1, x_2, \ldots, x_N$ and the jth row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input x_j.

For the binary classification applications, the decision function of ELM [17] is

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{L} \beta_i g(\mathbf{a}_i, b_i, \mathbf{x})\right) = \operatorname{sign}(\beta \cdot h(x)). \tag{6}$$

$h(\mathbf{x}) = [g(\mathbf{a}_1, b_1, \mathbf{x}), \ldots, g(\mathbf{a}_L, b_L, \mathbf{x})]^T$ is the output vector of the hidden layer with respect to the input $\mathbf{x}$. $h(\mathbf{x})$ actually

(1) **for** $i = 1$ to L **do**
(2) randomly assign input weight w_i
(3) randomly assign bias b_i
(4) **end for**
(5) calculate $\mathbf{H}$
(6) calculate $\beta = \mathbf{H}^{\dagger}\mathbf{T}$

Algorithm 1: ELM.

maps the data from the d-dimensional input space to the L-dimensional hidden layer feature space $\mathbf{H}$.

In ELM, the parameters of hidden layer nodes, that is, w_i and b_i, can be chosen randomly without knowing the training datasets. The output weight $\mathbf{L}$ is then calculated with matrix computation formula $\mathbf{L} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{H}$.

ELM tends to reach not only the smallest training error but also the smallest norm of weights [18]. Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, activation function $g(x)$, and hidden node number L, the pseudocode of ELM [10] is given in Algorithm 1.

3. The ELM-Based QoS Prediction Framework

In order to immediately comprehend our idea, we illustrate the whole process of ELM-based Web service QoS prediction shown in Figure 1. As shown, the process consists of four major phases: (1) preprocess, which records the composite service execution log information, extracts multidimensional QoS attributes, and converts them into service feature vectors; (2) the EC rules mining, where a prefix tree based algorithm is proposed to mine the candidate feature sets, namely, the EC rules; (3) diversified feature selection, where a small subset of diversified features are extracted from all the rules to construct a classifier of high prediction accuracy, that is, F-ELM; (4) feature updating, where the process periodically updates the prefix tree with the QoS values changing.

(1) Preprocess. At first, the system needs to collect large amounts of composite service execution information, aiming to mine useful knowledge for prediction. The original service log includes a variety of structural and unstructural data information, such as service trace logs, quality of service (QoS) information, service invocation relationships, and Web service description language (WSDL). These sets of information are typically heterogeneous, of multiple data types, and high dynamic. Thus, in order to extract the useful feature vectors, a preprocess step is necessary. This part will be discussed in Section 4.

(2) The EC Rules Mining. Since the goal is to conduct an online Web service QoS prediction, the rules should be concise so as to response the predictor as early as possible. By the proposed EC rules, we are enabled to trigger the prediction as soon as possible with high confidence. An efficient prefix tree based mining algorithm together with some effective pruning rules is developed to mine such rule. This part will be described in Section 5.

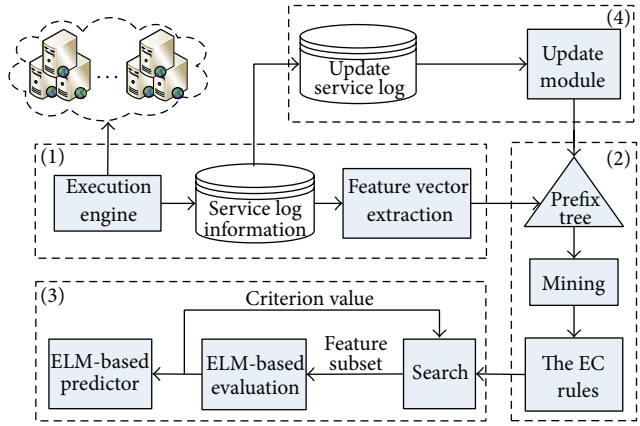


Figure 1: ELM-based QoS prediction process.

(3) Diversified Feature Selection. Too many rules increase the chance for model overfitting and decrease the generalization performance of a model. Thus, in this step, we study how to extract a small subset of diversified features as the representative of all mined results. By an ELM-based evaluation, the feature subset of the highest score is utilized to construct the predictor, that is, F-ELM. This part will be described in Section 6.

(4) Features Updating. Further, when a new service sequence is input, the update module judges QoS status of the service sequences. If the status of a service attribute changes greatly, the update module sends the updating request to the prefix tree according to a certain strategy. Besides judging the status, the feature values of each node in the prefix tree are recalculated periodically. In this paper, we exploit the strategy mentioned in [19] to address the issue.

In what follows, we mainly focus on steps (1)~(3) one by one.

4. Proprecessing

Once a service-oriented application or a composite service is deployed in a running environment, the application can be executed in many execution instances. Each execution instance is uniquely identified with an identifier (i.e., id). In each execution instance, a set of service components can be triggered. Due to various internet uncertain factors, there possibly exist a large number of sets of potential exception status information. We record the triggered events of the Web service failure information in a log. It is helpful for service quality management by extracting execution status information from the execution log to predict the service reliability.

Web service QoS information often includes many attributes. For example, the literature [20] lists twelve attributes to depict service QoS, for example, response time,

Table 1: Composite QoS status information.

A	Composite QoS	Description
0	$\langle av^0, exe^0\rangle$	Server unavailable, runtime delay
1	$\langle av^{0.5}, exe^0\rangle$	Server available intermittently, runtime delay
2	$\langle av^1, exe^0\rangle$	Server available, runtime delay
3	$\langle av^{0.5}, exe^1\rangle$	Server available intermittently, normal execution
4	$\langle av^1, exe^1\rangle$	Server available, normal execution
5×	$\langle av^0, exe^1\rangle$	Server unavailable, normal execution

Table 2: An example of service execution instances.

ID	Execute log entry	Count	Class
1	S_1^0	3	Failed with type A
2	$S_1^1S_3^1$	3	Failed with type B
3	$S_1^2S_2^2S_3^2$	2	Failed with type C
4	$S_1^1S_2^1S_3^1$	3	Failed with type B
5	$S_1^4S_2^2S_3^2S_4^2$	2	Failed with type C
6	$S_1^4S_2^2S_3^2$	2	Failed with type C
7	$S_1^4S_2^3S_3^4S_4^2$	2	Successful
8	$S_1^4S_2^3S_3^4$	3	Successful

availability, throughput, successability, reliability, compliance, latency, service name, WSDL address, documentation, and service classification. To simplify the explanation, we assume that there are just two QoS attributes for each component service in this paper, that is, the availability attribute (av) and the execution time attribute (exe). We further suppose that there are three possible states for av, that is, inaccessible, intermittently accessible, and accessible, denoted by av^0, $av^{0.5}$, and av^1, respectively, and two states for exe, that is, delayed execution and normal execution, denoted by exe^0 and exe^1, respectively. As such, we obtain five possible groups of service execution statuses as shown in Table 1: $\langle av^0, exe^0\rangle$, denoted by S^0, corresponding to the status of server unavailable and runtime delay; $\langle av^{0.5}, exe^0\rangle$, denoted by S^1, the status of server available intermittently and runtime delay; $\langle av^1, exe^0\rangle$, denoted by S^2, the status of server available and runtime delay; $\langle av^{0.5}, exe^1\rangle$, denoted by S^3, the status of server available intermittently but normal execution; and $\langle av^1, exe^1\rangle$, denoted by S^4, the status of server available but normal execution. Note that the status $\langle av^0, exe^1\rangle$, denoted by S^5, does not exist in practice. This is because an unavailable service is not executed.

Given the QoS status representation in Table 1, we extract Web service trace logs and QoS information from the service log and convert them into feature vectors by the following way. Let $S = \{S_1, S_2, \ldots, S_n\}$ be the candidate service component set and S_i^j the status j of service S_i. For every record in the web service log, we replace each individual component service by the corresponding status such that every record could be converted into a sequence of feature vectors. Table 2 exemplifies a service execution dataset of 15 failed executions, 5 successful executions, and 3 failure types. For example, column 2 in row 5 denotes an execution sequence "$S_1^4S_2^2S_3^2S_4^2$," which first invokes service S_1 of the status s_4 and then service S_2 of the status s_2, service S_3 of the status s_2, and service S_4 of the status s_2. Column 3 indicates that the sequence was executed twice, and column 4 shows that this execution failed with error type C.

5. The EC Rules Mining

In the last step, we have modeled the data as a sequence dataset. Next, we detail how to mine the candidate features for on-line Web service QoS prediction from the sequence dataset. Different from the sequence feature used in other domains, we require that the features in the context of on-line Web service QoS prediction should be of two important properties: (1) sequential character and (2) conciseness. This is because on-line Web service QoS prediction is a temporal process, where the prediction should be triggered as soon as possible.

5.1. Basic Definition. In this section, we first give some basic concepts and the problem statement.

Definition 1 (*feature*). Let $\mathcal{D}$ be service execution log with service set $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$. Let $\mathcal{F} = \{S_1^i S_2^j \cdots S_l^k\} \subseteq \mathcal{S}$ $(l = 1, 2, \ldots, n)$ be a component service or a subset of execution sequences containing status information. We call a set of component services with status information a Feature. Note that S_l^k is the status k of service S_l.

Given a feature $\mathcal{F} = s_1^i s_2^j, \ldots, s_m^k$, we say feature $\mathcal{F}$ appears in a sequence $\mathcal{S}$ if there exists $1 \le i_0 \le n-m+1$ such that $\mathbf{a}_{i_0} = s_1^i$, $\mathbf{a}_{i_0+1} = s_2^j, \ldots,$ $\mathbf{a}_{i_0+m-1} = s_m^k$. However, a feature $\mathcal{F}$ may appear several times in a sequence. For example, $\mathcal{F} = \{acd\}$ appears twice in sequence $\mathcal{S} = \{$aacdf gacde$\}$.Below, we give the minimum prefix length definition.

Definition 2 (*minimum prefix length*). Given feature $\mathcal{F}$ and a sequence $\mathcal{S}$, where $\mathcal{F} \subseteq \mathcal{S}$, the minimum prefix length is the length from initial position of $\mathcal{S}$ to the first matched position of $\mathcal{F}$ (MPL$(\mathcal{F}, \mathcal{S})$ for short).

Definition 3 (*weight Intra_Support with early factor*). Let $\mathcal{F}$ be a feature and let C_k be a class. The weight intraclass support with early factor of feature $\mathcal{F}$ in class C_k is the ratio of the sum of the reciprocals of the minimum prefix lengths containing $\mathcal{F}$ in C_k to the number of data in class C_k. wis^e is an abbreviation of weight *Intra_Support* with early factor. Consider

$$\text{wis}^e(\mathcal{F} \longrightarrow C_k) = \frac{\sum_{\mathcal{F}\in\mathcal{S}(\mathcal{F}\to C_k)} \left(1/\text{MPL}\left(\mathcal{F}, \mathcal{S}\left(\mathcal{F} \to C_k\right)\right)\right)}{\left\|C_k\right\|}. \quad (7)$$

Input: data set D, α (minimum wis^e), β and α
Output: Feature Sets (FS)
(1) Set FS = ϕ
(2) Count support of 1-features in every class
(3) Generate 1-feature set(F_1)
(4) Count support of 1-features in different class
(5) Select 1-features respectively and add them to FS
(6) new feature set ← Generate(2-feature set(F_2))
(7) **while** new feature set is not empty **do**
(8) Count wis$^e(\mathcal{F}, C_k)$ of candidates in new feature set
(9) For each feature $\mathcal{F}$ in $(l+1)$-feature set
(10) **Applying pruning 1:** IF wis$^e(\mathcal{F} \rightarrow C_k) < \alpha$
(11) remove feature $\mathcal{F}$;
(12) Else if there is a superset $\mathcal{F}a$ of feature $\mathcal{F}$ in l-feature set
(13) **Applying pruning 2:** that wis$^e(\mathcal{F}a)$ = wis$^e(\mathcal{F})$ or
(14) **Applying pruning 3:** $\log((\sigma + \text{wis}^e(\mathcal{F} \rightarrow C_k))/(\sigma + \text{wis}^e(\mathcal{F} \rightarrow \neg C_k)) \geq \beta$
(15) Then remove feature $\mathcal{F}$;
(16) Select optimal features to FS;
(17) **ENDIF**
(18) **end while**
(19) new feature set ← Generate(next level features sets)
(20) Return FS;
Function 1 Generate($l+1$)-feature Set
(21) Let $(l+1)$-feature set be empty set
(22) (Note: Obey by the $\text{CI}_{k-1} * \text{CI}_{k-1}$ Method to Merge)
(23) **for** each pair of features pP_{l-1} and $P_{l-1}q$ in l-feature set **do**
(24) Insert candidate $P_{l-1} \cdot pq$ in $(l+1)$-feature set;
(25) **for** all $P_l \subset P_{l-1}pq$ **do**
(26) **if** P_l does not exist in l-feature set **then**
(27) Then remove candidate $P_{l-1}pq$
(28) **end if**
(29) Return $(l+1)$-feature set
(30) **end for**
(31) **end for**

Algorithm 2: The EC-Miner algorithm.

wis$^e(\mathcal{F} \cup C_k)$ denotes the support of features $\mathcal{F}$ and C_k emerging simultaneously, wis$^e(\mathcal{F} \cup \neg C_k)$ denotes the support of features $\mathcal{F}$ and $\neg C_k$ emerging simultaneously; that is, wis$^e(\mathcal{F} \cup \neg C_k)$ = wis$^e(F)$ − wis$^e(\mathcal{F} \cup C_k)$. We say that wis$^e(\mathcal{F})$ is frequent if wis$^e(\mathcal{F}) \geq \alpha$, where α is user-specific minimum frequent threshold.

Definition 4 (*discriminative feature*). Let $\mathcal{F}$ be a feature and let C_k be a class. The discriminative power $\mathcal{F}$, denoted by DF, is calculated as follows:

$$\text{DF}(\mathcal{F}) = \log\left(\frac{\sigma + \text{Supp}(\mathcal{F} \longrightarrow C_k)}{\sigma + \text{Supp}(\mathcal{F} \longrightarrow \neg C_k)}\right), \quad (8)$$

where σ is a regulation factor. Specially, we say that $\mathcal{F}$ is discriminate feature if DF($\mathcal{F}$) $\geq \beta$, where β is a user-specific minimum discriminative threshold.

The rationale behind Definition 4 is intuitive. If a feature $\mathcal{F}$ often occurs in class C_k but rarely in other classes (i.e., $\neg C_k$), we consider it a feature well discriminating C_k from the other classes. Moreover, since Supp($\mathcal{F} \rightarrow \neg C_k$) may be zero, we add a regulation factor σ to avoid this case.

Definition 5 (*concise feature*). Given a specific class label C_k and a discriminate feature $\mathcal{F}$, the discriminative power of $\mathcal{F}$ is no less than that of a longer feature $\mathcal{F}'$ and conf($\mathcal{F}' \rightarrow C_k$) $\geq$ conf($\mathcal{F} \rightarrow C_k$), we say that $\mathcal{F}$ is concise with respect to C_k, where conf($\mathcal{F} \rightarrow C_k$) = wis$^e(\mathcal{F}C_k)$/wis$^e(\mathcal{F})$.

Definition 5 is also understandable. This is because if we have a shorter feature $\mathcal{F}'$, the discriminative power of which is no less than that of a longer feature $\mathcal{F}$ such that $\mathcal{F}' \sqsubseteq \mathcal{F}$, there is no need to use $\mathcal{F}$ instead of $\mathcal{F}'$ for classification. That is, we prefer a feature of shorter size but stronger discriminative power. In this sense, we refer to such a feature as a concise feature.

Problem Statement. Given a Web service execution log D, a minimum frequent threshold α, a regulation factor σ, and a minimum discriminative threshold β, our goal is to find all sequence rules satisfying both Definitions 3 and 5, that is, *the EC rules*.

5.2. The EC-Miner Algorithm. In this section, we detail the proposed EC rule mining algorithm, namely, EC-Miner. The main idea is formalized in Algorithm 2.

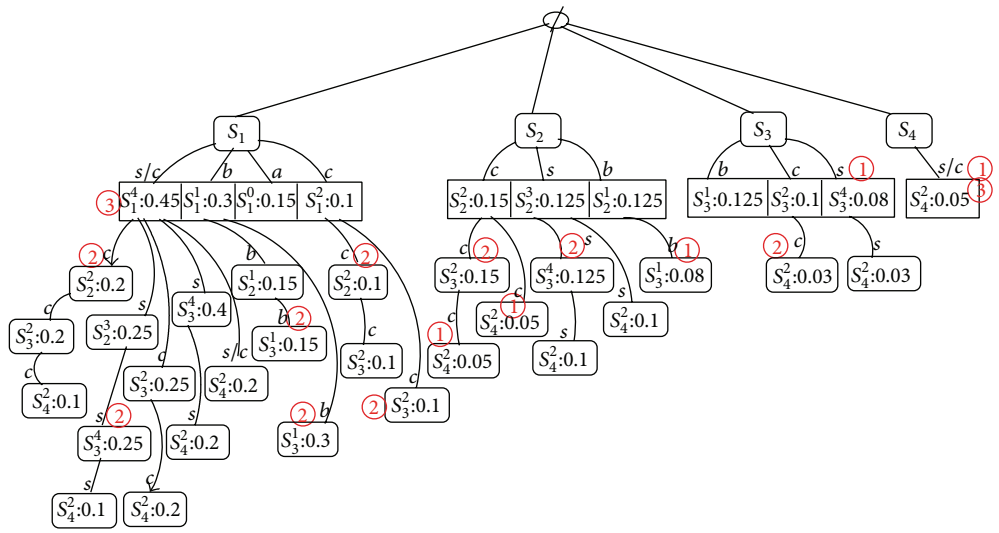


Figure 2: The EC rule mining.

The mining process is exemplified by a prefix tree as shown in Figure 2, which is built on Table 2 with $\text{wis}^e = 0.1$. As seen from Figure 2, there are four services s_1, s_2, s_3, and s_4 at level 1. We obtain different status information for each service in descending order at level 2. Different from the traditional support computing method, we consider both concise and early characteristics. Therefore, the obtained order using wis^e for each item is also different from traditional approaches. For example, at first, the algorithm scans Table 2 once and computes the wis^e of each item. After computing, we generate the candidate early feature for the second level. We can see 11 candidate 1-features at level 2. The wis^e of each one item is $s_1^4 : 0.45$, $s_1^1 : 0.3$, $s_1^0 : 0.15$, $s_2^2 : 0.15$, $s_2^3 : 0.125$, $s_3^1 : 0.125$, $s_1^2 : 0.1$, $s_3^2 : 0.1$, and so on, where the number after colon denotes weight of *Intra_Support* with early factor (wis^e). Next, we generate the candidate early 2-features for the third level. For example, $s_1^4 s_2^3 : 0.25$ denotes the wis^e of $s_1^4 s_2^3$ which is 0.25. The feature $\mathcal{F}$ with solid box represents the corresponding rule. For example, class s with solid box under $s_1^4 s_2^3$ means the rule $s_1^4 s_2^3 \rightarrow$ successful can be deduced.

We can use the wis^e-based pruning 1 (see the details in Section 5.3) to prune some rules (lines 2–5). We can prune some redundancy rules by applying pruning rule 1 (lines 8~10); for example, candidates s_2^1, s_4^2, $s_3^2 s_4^2$, $s_3^4 s_4^2$ are removed since $\text{wis}^e(s_3^1) = 0.08 < \alpha = 0.1$, $\text{wis}^e(s_4^2) = 0.05 < \alpha = 0.1$, $\text{wis}^e(s_3^2 s_4^2) = 0.03 < \alpha = 0.1$, $\text{wis}^e(s_3^4 s_4^2) = 0.03 < \alpha = 0.1$. In Figure 2, the pruning rule 1 is applied which is marked by ①. Further, if all features under threshold are pruned, the rules containing these features will be pruned.

Then, we perform the concise-based pruning rule 2 (lines 2–5, Algorithm 2), which will also be explained in Section 5.3. For instance, $(s_1^2 s_2^2)$ in candidate $(s_1^2 s_2^2; c)$ is terminated because $\text{wis}^e(s_1^2) = \text{wis}^e(s_1^2 s_2^2) = 0.1$. In Figure 2, the pruning rule 2 is applied which is marked by ②.

At last but not the least, discriminative-based pruning rule 3 is very important but not difficult to be understood (see Section 5.3). For example, candidate feature s_1^4 is removed by line 14 because $\text{wis}^e(s_1^4 \rightarrow c) = \text{wis}^e(s_1^4 \rightarrow \text{successful})$. In Figure 2, the pruning rule 3 is applied which is marked by ③. A complete pseudocode for mining optimal EF sets is presented in Algorithm 2.

Algorithm 2 discusses the wis^e-based pruning, concise-based pruning, and discriminative-based pruning. Most of the existing algorithms find an interesting rule set by post-pruning. However, this may be very inefficient especially when the minimum support is low since it will generate an amount of redundancy rules. Our EC-Miner algorithm makes use of the interestingness measure property to efficiently prune uninteresting rules and saves only the maximal interesting rules instead of all ones. This distinguishes it from other association rule mining algorithms.

Function 1 is a function to generate candidate item sets. All generated candidates are built on the prefix tree structure. We adopt the $CI_{k-1} * CI_{k-1}$ merge strategy [21] to obtain the candidate item sets. After rules have been formed, we can prune many redundancy rules.

5.3. The Pruning Strategies. To improve the efficiency of EC-Miner, we devise a series of pruning rules.

Pruning Rule 1. In pruning by wis^e: given wis^e, a feature $\mathcal{F}$ and all its possible proper supersets $\mathcal{F}a$, and class C_k, if $0 \leqslant \text{wis}^e(\mathcal{F} \rightarrow C_k) \leqslant \alpha$, then $\mathcal{F} \rightarrow C_k$ and $\mathcal{F}a \rightarrow C_k$ are all not the EC rules.

Proof. Once $0 \leqslant \text{wis}^e(\mathcal{F} \rightarrow C_k) \leqslant \alpha$ is observed, it is not necessary to search for more specific rules $\mathcal{F}a \rightarrow C_k$. Because $\text{wis}^e(\mathcal{F}a \rightarrow C_k) \leq \text{wis}^e(\mathcal{F} \rightarrow C_k) \leqslant \alpha$. So, target $C_k \in C$ will be terminated in candidate rule $\mathcal{F} \rightarrow C_k$. □

Instead of the global support, pruning 1 describes the intraclass support of a feature with respect to a specific class. This is because a feature $\mathcal{F}$ in C_k is hardly frequent if C_k is rare in service execution log. Thus, pruning rule 1 can reduce the redundancy rules greatly. This is different from association rules.

Pruning Rule 2. In pruning by conciseness, given wis^e, a feature $\mathcal{F}$ and all its possible proper supersets $\mathcal{F}a$, and class C_k, if $\text{wis}^e(\mathcal{F}) = \text{wis}^e(\mathcal{F}a)$, then feature $\mathcal{F}a$ and all its proper supersets can be pruned.

Proof. In the proof, we show that $\text{confidence}(\mathcal{F} \rightarrow C_k) > \text{confidence}(\mathcal{F}a \rightarrow C_k)$

$$\begin{aligned}\text{Conf}(\mathcal{F} \longrightarrow C_k) &= \frac{\text{wis}^e(\mathcal{F} \cup C_k)}{\text{wis}^e(\mathcal{F})} = \frac{\text{wis}^e(\mathcal{F} \cup C_k)}{\text{wis}^e(\mathcal{F}a)} \\ &> \frac{\text{wis}^e(\mathcal{F}a \cup C_k)}{\text{wis}^e(\mathcal{F}a)} = \text{Conf}(\mathcal{F}a \longrightarrow C_k).\end{aligned} \tag{9}$$

□

Pruning Rule 3. In pruning by discrimination, given a feature $\mathcal{F}$, if $\log(\sigma + \text{wis}^e(\mathcal{F} \rightarrow C_k))/(\sigma + \text{wis}^e(\mathcal{F} \rightarrow \neg C_k)) \geq \beta$, then $\mathcal{F}$ will not be the discriminative prediction rules. σ and β are appointed by user.

Proof. If $\log(\sigma+\text{wis}^e(\mathcal{F} \rightarrow C_k))/(\sigma+\text{wis}^e(\mathcal{F} \rightarrow \neg C_k)) \geq \beta$, then $\mathcal{F}$ is relative frequent in different class. However, we say feature $\mathcal{F}$ does not have the ability to distinguish different class because it does not satisfy Definition 4. □

The above pruning rules are very efficient since they only generate a subset of frequent features with great interestingness instead of all ones. Finally, the EC rules set is significantly smaller than an association rule set but is still too large for decision practitioners to review them all. Next, we give an ELM-based diversified feature selection method to further reduce the size of EC rules.

6. ELM-Based Diversified Feature Selection

As ever mentioned, the EC-Miner algorithm generates a set of optimal feature sets (rules); however, their number may be still a little large. An enormous number of features impose a great challenge on understanding and further analyzing the classification or prediction results. In this section, we study how to construct a classifier of high classification (prediction) accuracy by extracting a small number of feature sets as the representative of all mined results.

In the context of feature selection data analysis, most of the current methods adopt such a framework that ranks the attributes according to their individual discriminative power to the target class and then selects top-k ranked attributes. These methods cannot remove redundant features. It is pointed out in a number of studies [22] that simply combining highly ranked features often does not form a better feature set because these features could be highly correlated. The drawback of redundancy among selected features is twofold. On one hand, the selected feature set can have a less comprehensive representation of the target class than one of the same size but without redundant features; on the other hand, redundant features may unnecessarily increase the size of the selected feature set, which may reduce the classifier performance. Besides incapability of handling redundant features, in most ranking based methods, the number of features to be selected is arbitrarily determined.

To address the above issues, we propose an ELM-based diversified feature selection method in this section. Before describing it, we first give a diversity function as follows:

$$\begin{aligned}\text{Div}(X_1, X_2) = &\left(1 - \frac{|T(X_1) \cap T(X_2)|}{|T(X_1) \cup T(X_2)|}\right) \\ &\cdot \left(1 - \frac{|\text{LCS}(X_1, X_2)|}{|I(X_1) \cup I(X_2)|}\right),\end{aligned} \tag{10}$$

where $T(X_1)$ is the set of samples which contain X_1 as a significant chain, $I(X_1)$ is the set of items involved in X_1, $\text{LCS}(X_1, X_2)$ is the longest common feature of X_1 and X_2, and symbol "| |" denotes the length of a pattern.

In (10), the diversity between two early rules X_1 and X_2, that is, $\text{Div}(X_1, X_2)$, is measured from two aspects: support sequences and involved items. If X_1 and X_2 have few common support sequences, they should have high diversity. Similarly, if LCS of X_1 and X_2 is short, they should have high diversity.

Based on (10), we can construct a diversity graph in the following way. For a list of results $\text{FS} = \{X_1, X_2, \ldots\}$, the corresponding diversity graph, denoted as $G(\text{FS}) = (V, E)$, is an undirected graph such that, for any result $X_i \in \text{FS}$, there is a corresponding node $v_i \in V$ and, for any two results $X_i \in \text{FS}$ and $X_j \in \text{FS}$, there is an edge $(v_i, v_j) \in E$ if and only if $\text{Div}(X_i, X_j) \leq \gamma$ (a user-specified threshold). The problem of finding a set of diversified rules, which represent all mined results, is now equivalent to find an independent dominating set of $G(\text{FS})$. Further, we require the number of the selected features as few as possible to reduce the complexity of classifier. Thus, the problem of diversified feature selection can be viewed as an instance of finding minimum independent dominating set of $G(\text{FS})$, which is NP-hard [23].

Since it is difficult to find the optimal solutions, we adopt a greedy algorithm to address this problem. Given the result set FS and a set of selected results FS', the algorithm incrementally selects patterns from $\text{FS} - \text{FS}'$ with diversity guarantee. A pattern $X_i \in \text{FS} - \text{FS}'$ is selected if $\forall X_j \in \text{FS}'$, $\text{Div}(X_i, X_j) > \delta$. If there are several such alternative X_i's in a selection, the one corresponding to a node of the most neighbors is selected. Note that, at beginning, the set FS' is empty. The algorithm picks the most significant pattern, that is, an irreducible sequence of the largest confidence value, and inserts it to FS'. As seen from what we mentioned, the final selected FS' may be more than one in the process. For example, there may be several sequences of the largest confidence value and there may be more than one node of the same number of neighbors. In such case, we use ELM

Input: a set of feature sets(FS)
Output: The selected feature subset FS′
(1) Let X be the feature set of the largest confidence
(2) FS′ = {X}
(3) **while** there is a node in FS − FS′ not dominated by FS **do**
(4) Find a pattern $X_i \in$ FS − FS′ s.t. $\forall X_j \in$ FS′, Div$\left(X_i, X_j\right) > \delta$, and the number of the neighbors of node X_i is largest
FS′ = FS′ ∪ {X_i}
(5) **end while**
(6) using ELM evaluates every possible FS′
(7) the FS′ of the highest accuracy on ELM;

Algorithm 3: The FS algorithm.

Table 3: QWS datasets attributes and their description.

Attributes	# Attributes name	Description	Units
X_1	Response time	Time taken to send a request and receive a response	ms
X_2	Availability	Number of successful invocations/total invocations	%
X_3	Throughput	Total number of invocations for a given period of time	Invokes/s
X_4	Successability	Number of responses/number of request messages	%
X_5	Reliability	Ratio of the number of error messages to total messages	%
X_6	Compliance	Number of successful invocations/total invocations	%
X_7	Best practices	The extent to which a WSDL document follows WSDL documentation	%
X_8	Latency	Time taken for the server to process a given request	ms
X_9	Documentation	Measure a documentation (i.e., description tags) in WSDL	%
X_{10}	Service classification	Levels representing service offering qualities (1 through 4)	Classifier
X_{11}	Service name	Name of the Web services	None
X_{12}	WSDL address	Location of the Web service definition language (WSDL) file on web	None

to evaluate every possible candidate. The one of the largest prediction accuracy is selected. Algorithm 3 formalizes the process.

The greedy algorithm can be viewed as a hybrid of the filter model and the wrapper model in feature selection, which achieves a better trade-off between the two. Better than the filter model, it explicitly removes redundancy among the selected features and determines the number of the selected features automatically. Compared with the wrapper model, it is of less computation cost.

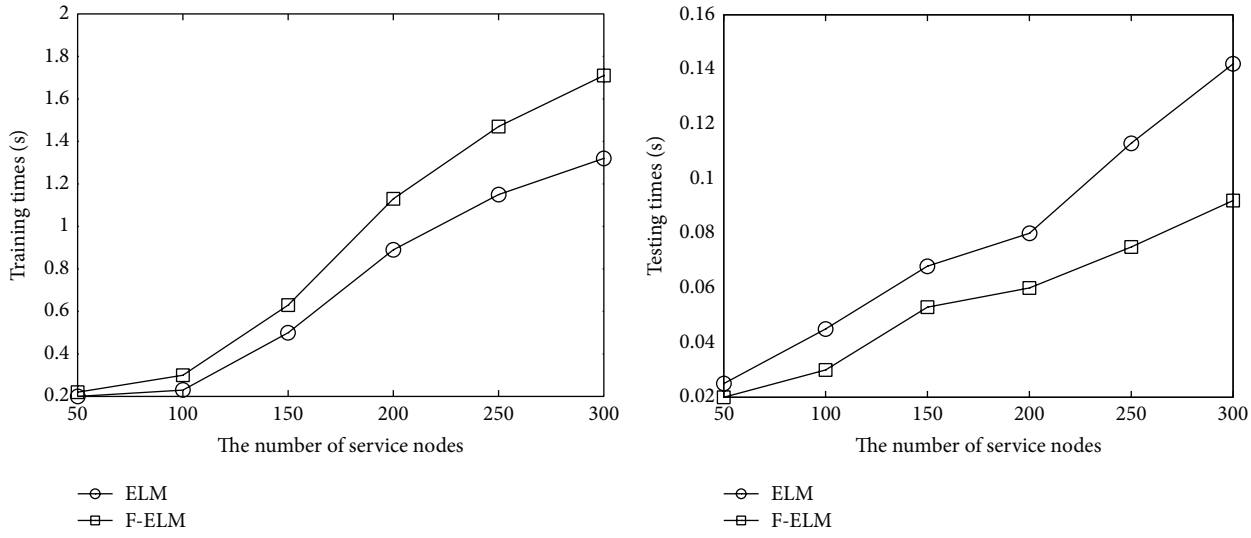
7. Experiments Result Analysis

In this section, we design a series of experiments to verify the performance of the proposed method. For brevity, we refer to the algorithm of diversified feature selection based ELM as F-ELM. We select two different scenarios: one is Web service quality prediction and the other is Web service fault diagnosis prediction.

We provide two kinds of datasets. For the real dataset, we use E. AI-Mari and Dr. QH. Mahmouds' QoS dataset [20] (downloaded from http://www.uoguelph.ca/~qmahmoud/qws/), which includes twelve attributes (x1 to x12) as shown in Table 3, where the attributes x1 to x10 are used as explanatory variables and the attribute x10 is used as the target variable. However, attributes x11 and x12 are ignored as they do not contribute to the analysis.

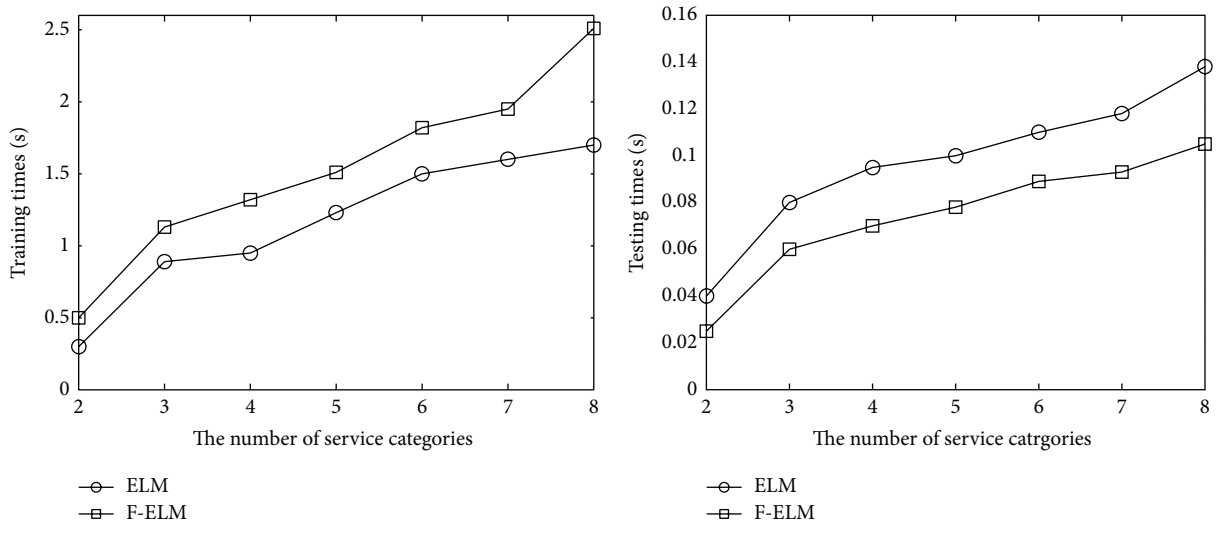
For artificial datasets, we get the Web service datasets by simulating a network environment and general network topology graph by ERITETool: with two input parameters: number of network nodes (#Web service), number of embedded classes (*#class*), percent of embedded fault rate (%). The system selects composite service by matching I/O operation.

7.1. Analysis of Efficiency. In this set of experiments, we refer to the diversified feature selection based ELM as F-ELM and the original ELM as ELM. The efficiency of F-ELM is studied by showing how response time varies with service nodes and service categories. In Figure 3, we compare the training time and the testing time between F-ELM and ELM with respect to the same categories (number of categories is 3) when the numbers of Web service are increasing (from 50 to 300). In Figures 4(a) and 4(b), we compare the training time and the testing time of F-FLM and ELM respectively, where the number of service categories varies from 2 to 8 while the number of Web services is fixed to 200.



(a) Training time comparison (b) Testing time comparison

Figure 3: Training/testing time comparison between F-ELM and ELM.



(a) Training time comparison (b) Testing time comparison

Figure 4: Training/testing time comparison between F-ELM and ELM.

As seen from Figures 3(a) and 4(a), training time decreases with service nodes increasing. We note that the total training time of F-ELM is a bit longer than original ELM when the service nodes are increasing. It is the same as the scenario where the service categories are increasing. This is because the increasing of service nodes (service categories) may lead to more rules to be evaluated and pruned in ELM-based diversified feature selection.

However, both Figures 3(b) and 4(b) show that the testing time of F-ELM outperforms that of ELM and the advantage becomes more substantial with a larger dataset (category). This is because ELM has to perform a time-consuming check for all feature sets. However, F-ELM only performs a series of early and concise interesting features. Although the test time changes little, F-ELM is still constantly faster than ELM.

7.2. Classification Accuracy. The following evaluation criteria are used to measure the performance of F-ELM, ELM, and SVM.

Accuracy denotes the proportion of the correctly classified service sequences in the whole service sequence sets

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{N}. \tag{11}$$

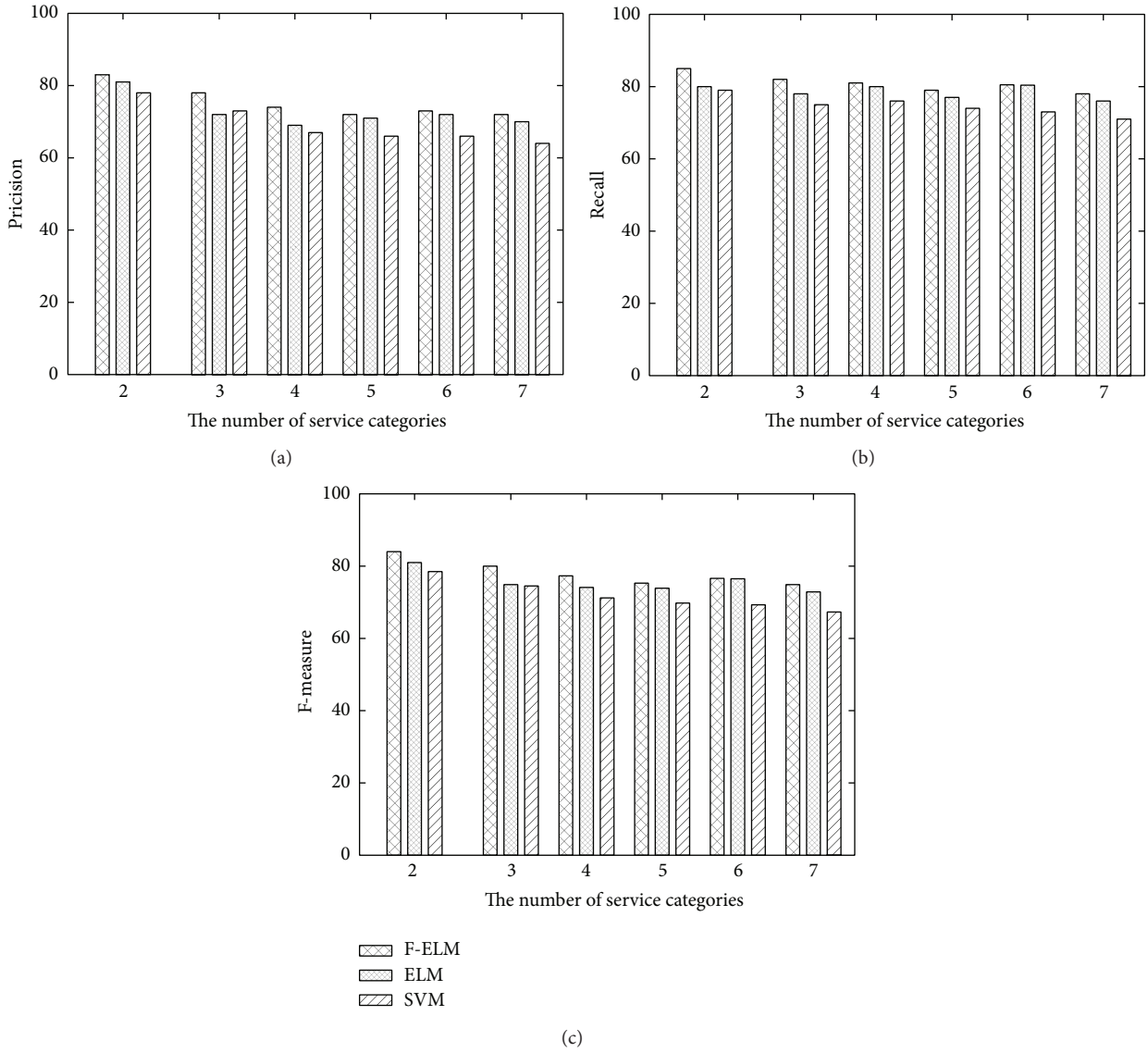


Figure 5: Performance comparison versus number of categories.

Precision denotes the proportion of the correctly classified service sequences with respect to a specific class

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{12}$$

Recall denotes the proportion of the correctly classified service sequences with respect to a specific class

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{13}$$

F-measure (F_1 score) is the harmonic mean of precision and recall. Since precision and recall cannot reach mathematical optimum, F_1 score measures both of the two criteria and assumes that the weight of precision is equal to the weight of recall

$$F_1 = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}. \tag{14}$$

For artificial dataset, Figure 5 presents the classification comparison result of F-ELM, the original ELM algorithm, and SVM with different categories changing. The precision comparison result is presented in Figure 5(a).

The recall comparison result is presented in Figure 5(b). Figure 5(c) shows the comparison result of F_1, scores. Figure 6 presents the classification comparison result of F-ELM, ELM, and SVM with different datasets changing. The precision comparison result is shown in Figure 6(a). The recall comparison result is present in Figure 6(b). Figure 6(c) shows the comparison result of F_1 scores. All six figures demonstrate that F-ELM is better than ELM and SVM on each of the three criteria in terms of both the categories vary and the datasets change.

The features selected by ELM-based diversified feature selection just involve six attributes out of the original twelfth ones. To show how the selected features of the six attributes

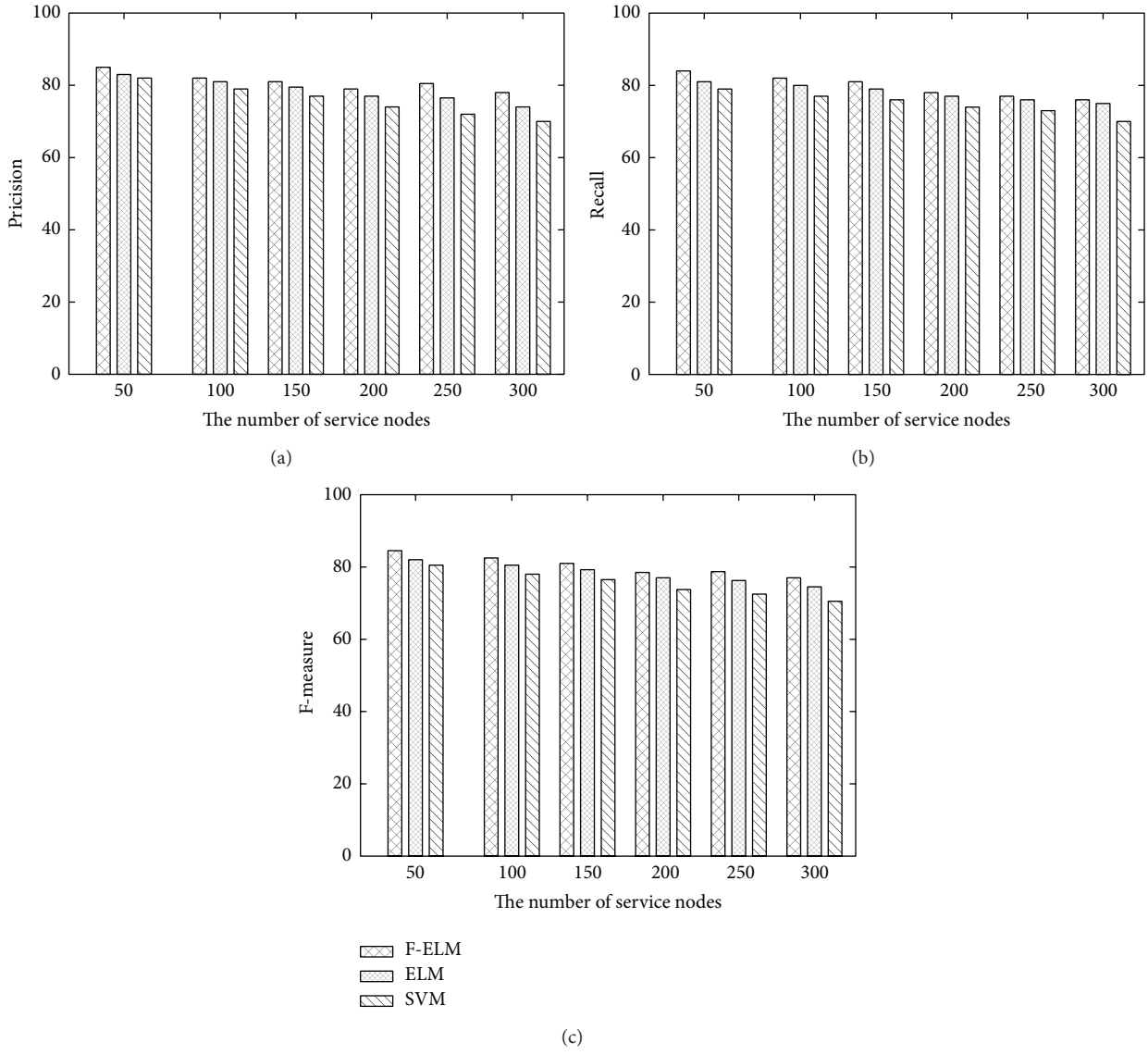


Figure 6: Performance comparison versus number of nodes.

affect the performance of a classifier, we compared the training time, the testing time, and the accuracies of six different classifiers, that is, ELM, SVM, CART, J48, Treenet, and BPNN, on the features of the six attributes and all the original attributes, respectively. The results are shown in Table 4. As seen, the performance of a classifier on the features of the six attributes is always better than that on the features of all the original attributes. This confirms that these classifiers can benefit from the selected features. Moreover, F-ELM behaves the best among all the introduced classifiers on the same attributes setting. This is because F-ELM exploits the relationship among attributes as the features and it removes the redundancy among the selected features while the other methods do not. t-test is utilized to evaluate whether the accuracy difference between F-ELM and a comparative method is statistically significant. Since 10-fold cross-validation is used and $t_{0.01}(49)$ is about 2.678, the values larger than 2.678 indicate a statistically significant difference. Thus, F-ELM does outperform the comparative methods on effectiveness. We also conduct the accuracy comparison of different algorithms on a real microarray dataset, that is, Leukemia dataset, which contains 7129 genes, 38 training samples, and 34 testing samples. The results are reported in Table 5. Since all the t-test values are larger than $t_{0.01}(33) = 2.733$, F-ELM still outperforms other comparative methods on accuracy in statistical significance. Thus, it is reasonable to say that the proposed method could be applied in a wider range of applications.

Additionally, we conducted a set of experiments for comparing the proposed feature selection method with six other feature selection methods, which are often used as comparative methods in machine learning for feature selection studies. The six methods are information gain (IG), twoing rule (TR), sum minority (SM), max minority (MM), Gini

Table 4: Accuracy comparison of different algorithms.

Algorithm	# Attributes	Training time	Testing time	Accuracy	t-test
F-ELM	6	1.15	0.26	82.25%	N/A
ELM	12	1.82	0.52	81.2%	5.103
SVM	6	2.3	0.69	82.1%	5.211
SVM	12	2.9	0.89	80.55%	5.060
CART	6	3.1	0.67	74.1%	5.533
CART	12	3.5	0.88	72.1%	5.667
J48	6	3.41	0.96	66.72%	6.025
J48	12	4.01	1.2	63.77%	6.222
Treenet	6	2.22	0.74	77.2%	5.732
Treenet	12	2.79	0.99	75.4%	5.547
BPNN	6	1.87	0.77	63.1%	6.267
BPNN	12	2.12	0.97	60.1%	6.467

Table 5: Accuracy comparison on Leukemia dataset.

Algorithm	# Attributes	Training time	Testing time	Accuracy	t-test
F-ELM	6	2.05	0.46	88.24%	N/A
ELM	12	3.26	0.93	85.29%	7.033
SVM	6	2.71	0.81	82.35%	7.194
SVM	12	3.42	1.05	82.35%	7.194
CART	6	4.37	0.94	76.47 %	7.780
CART	12	4.94	1.24	73.53%	7.989
J48	6	4.71	1.33	67.65%	8.493
J48	12	5.54	1.66	61.76%	8.771
Treenet	6	3.65	1.22	76.47%	7.780
Treenet	12	4.23	1.57	73.53%	7.876
BPNN	6	3.13	1.29	64.71%	8.643
BPNN	12	3.56	1.63	61.76%	9.116

Table 6: Accuracy comparison versus different feature selection algorithms.

Algorithm	EF	IG	TR	SM	MM	GI	SV
ELM	82.55%	81.47%	79.83%	77.66%	78.08%	81.72%	82.3%
SVM	82.1%	80.64%	79.0%	78.6%	77.1%	79.65%	81.03%
CART	74.1%	71.03%	71.75%	72.01%	73.11%	71.68%	71.68%
J48	66.72%	64.37%	59.76%	55.51%	58.41%	61.79%	59.63%
Treenet	77.2%	73.8%	77.7%	74.48%	74.73%	73.68%	74.67%
BPNN	63.1%	60.63%	58.8%	58.5%	59.07%	59.34%	60.73%

index (GI), and sum of variance (SV), respectively. The results are reported in Table 6, where ELM-based feature selection is abbreviated as EF. As seen from Table 6, EF approach provides the highest accuracy on all classifiers. This is mainly because the features selected by EF can be considered as a diversified coverage of all the original features, which provide more complete but less redundant information than the comparative methods.

In summary, as seen from Tables 4 and 5, when applying ELM in service QoS prediction or in other applications (such as microarray data classification), we can always obtain the best results. This confirms the effectiveness of ELM in a wide of applications. Further, this can be explained in such a way that the proposed F-ELM extracts the concise features of high discriminative power for each category while the other methods do not.

7.3. Effects of the Parameters. In this section, we study how the parameters affect the performance of ELM-based diversified feature selection. In Figure 7(a), the running time of the feature selection decreases with α increasing. This is because the larger α makes more rules pruned. The reduced search

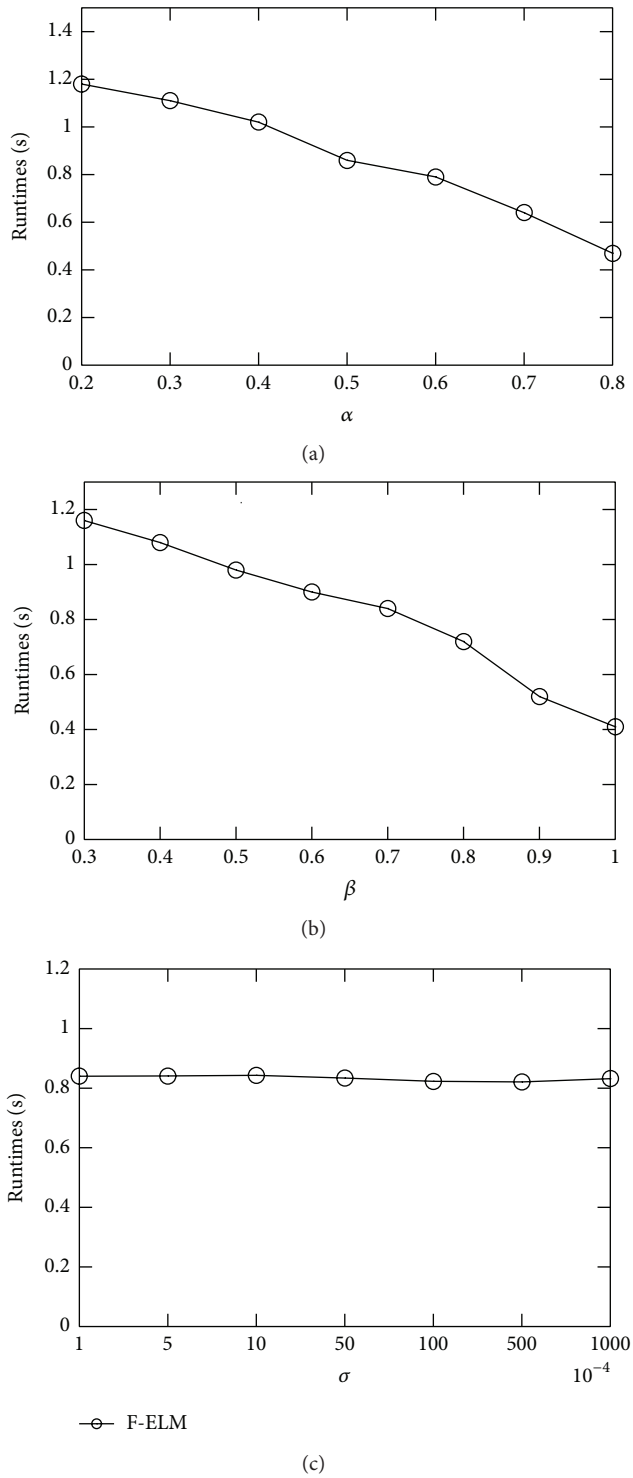


Figure 7: Runtime versus parameters.

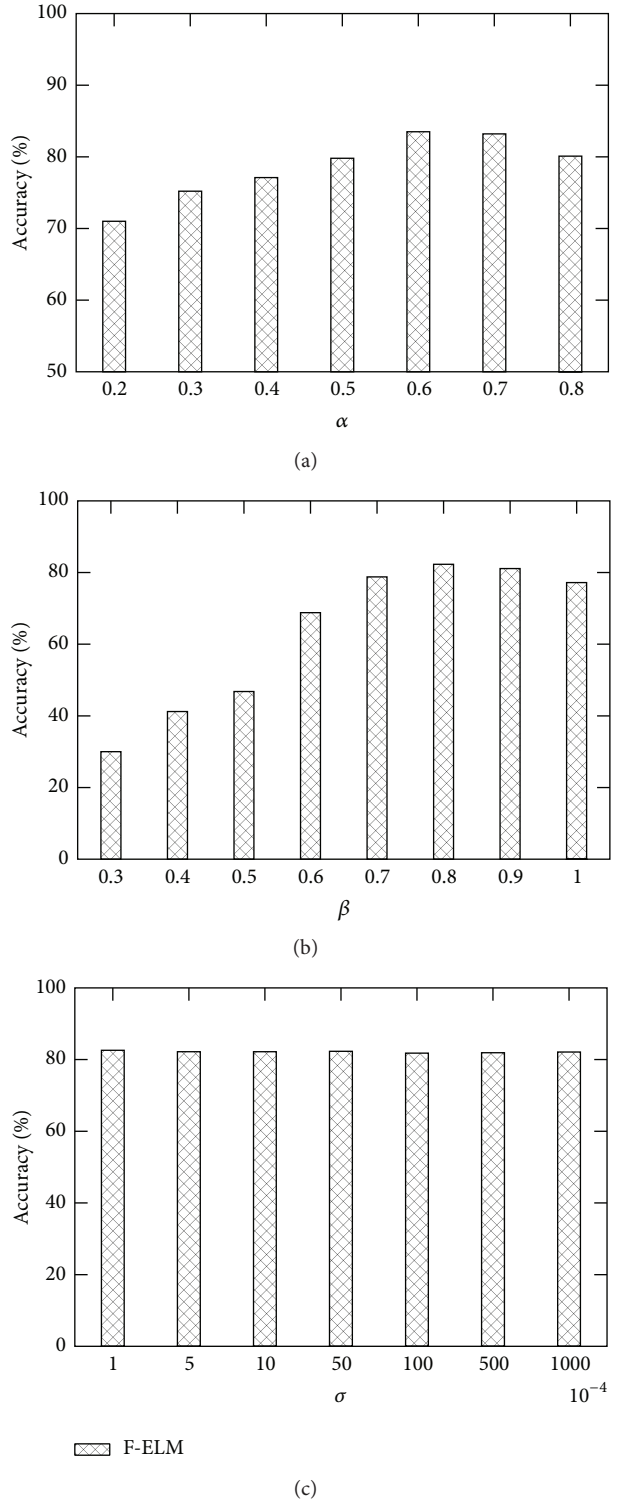


Figure 8: Accuracy versus parameters.

space leads to the less running time. However, this does not indicate that we should choose α as large as possible. Figure 8(a) shows that too large α may deteriorate classification accuracy. This is because many rules of potentially high usability will be pruned at a high α level. Also, the accuracy at a too low α level is not very good due to the "overfitting" problem. Figures 7(b) and 8(b) give how β affects the feature selection performance. The cases are similar as those in Figures 7(a) and 8(a), respectively. That is, the running time of the feature selection decreases with β increasing, and too large and too low β may deteriorate classification accuracy. The results can also be explained in a similar way as those

for Figures 7(a) and 8(a), respectively. Differently, Figures 7(c) and 8(c) show that σ rarely affects the running time of the feature selection and classification accuracy. This is because σ is introduced just for avoiding the case where the denominator of (8) is zero. σ is often set to a very low value, the effect of which is dominated by other values setting in (8).

8. Conclusions

In this paper, we propose an ELM-based service quality prediction framework. Considering the highly dynamic and the uncontrollable circumstances, the service quality prediction is required to be triggered as soon as possible in the proposed framework. By developing the prefix tree based algorithm, EC-Miner, a series of candidate rule sets are first found, where both the earliness and conciseness of the rules are considered. Then, an ELM-based diversified feature selection algorithm is proposed to fine the candidate rule set. A small subset of high-quality features are discovered as the representative of the whole candidate rule set. A greedy algorithm is presented to approximate the optimal solution. Experimental results show that the proposed approach significantly improves the efficiency and the effectiveness of ELM with respect to some widely used feature selection techniques.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China under Grants nos. 61100028, 61272182, 61073062, and 61073063; State Key Program of National Natural Science of China (61332014); the New Century Excellent Talents in University Award (NCET-11-0085); the Fundamental Research Funds for the Central Universities under grants (no. 130504001); the Ph.D. Programs Foundation of Ministry of Education of China (young teacher) (no. 20110042120034).

References

[1] L. Zhang, J. Zhang, and C. Hong, *Services Computing*, Tsinghua University Press, Beijing, China, 2007.

[2] F. Wang, L. Liu, and C. Dou, "Stock market volatility prediction: a service-oriented multi-kernel learning approach," in *Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12)*, pp. 49–56, June 2012.

[3] J. W. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Machine Learning Press, 3rd edition, 2012.

[4] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, June 2014.

[5] A. Goldman and Y. Ngoko, "On graph reduction for QoS prediction of very large web service compositions," in *Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12)*, pp. 258–265, June 2012.

[6] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 573–579, 2013.

[7] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service QoS prediction with location-based regularization," in *Proceedings of the IEEE 19th International Conference on Web Services (ICWS '12)*, pp. 464–471, Honolulu, Hawaii, USA, June 2012.

[8] J. Wu, L. Chen, H. Jian, and Z. Wu, "Composite service recommendation based on bayes theorem," *International Journal of Web Services Research*, vol. 9, no. 2, pp. 69–93, 2012.

[9] J. Park, H. Yu, K. Chung, and E. Lee, "Markov chain based monitoring service for fault tolerance in mobile cloud computing," in *Proceedings of the 25th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '11)*, pp. 520–525, March 2011.

[10] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[11] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[12] X.-G. Zhao, G. Wang, X. Bi, P. Gong, and Y. Zhao, "XML document classification based on ELM," *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, 2011.

[13] G. Wang, Y. Zhao, and D. Wang, "A protein secondary structure prediction framework based on the Extreme Learning Machine," *Neurocomputing*, vol. 72, no. 1–3, pp. 262–268, 2008.

[14] J. Cao and Z. Lin, "Bayesian signal detection with compressed measurements," *Information Sciences*, vol. 289, pp. 241–253, 2014.

[15] R. Zhang, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Multi-category classi fi cation using an Extreme Learning Machine for microarray gene expression cancer diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 3, pp. 485–495, 2007.

[16] Y. Zhao, J. Y. Xu, G. Wang, L. Chen, B. Wang, and G. Yu, "Maximal subspace co-regulated gene clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 83–98, 2008.

[17] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.

[18] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, no. 1–4, pp. 306–314, 2005.

[19] Y. Zhao, G. Wang, X. Zhang, J. X. Yu, and Z. Wang, "Learning phenotype structure using sequence model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 667–681, 2014.

[20] R. Mohanty, V. Ravi, and M. R. Patra, "Web-services classification using intelligent techniques," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5484–5490, 2010.

[21] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Datt Mining*, Addison-Wesley, 2nd edition, 2014.

[22] J. Biesiada and W. Duch, "Feature selection for high-dimensional data—a person redundancy based filter," in

Computer Reconginition System, Advances in Soft Computing, pp. 242–249, Springer, Berlin, Germany, 2008.

[23] D. Zuckerman, "On unapproximable versions of NP-complete problems," *SIAM Journal on Computing*, vol. 25, no. 6, pp. 1293–1304, 1996.