Neurocomputing 74 (2011) 2438–2443

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An OS-ELM based distributed ensemble classification framework in P2P networks

Yongjiao Sun [a,b,*], Ye Yuan [a,b], Guoren Wang [a,b]

[a] Key Laboratory of Medical Image Computing (NEU), MOE, China
[b] College of Information Science and Engineering, Northeastern University, Shenyang 110004, China

ARTICLE INFO

Available online 20 May 2011

Keywords:
Peer-to-Peer (P2P)
Extreme learning machine
OS-ELM
Ensemble classification
Parallel ensemble classification
Quad-tree

ABSTRACT

Although classification in centralized environments has been widely studied in recent years, it is still an important research problem for classification in P2P networks due to the popularity of P2P computing environments. The main target of classification in P2P networks is how to efficiently decrease prediction error with small network overhead. In this paper, we propose an OS-ELM based ensemble classification framework for distributed classification in a hierarchical P2P network. In the framework, we apply the incremental learning principle of OS-ELM to the hierarchical P2P network to generate an ensemble classifier. There are two kinds of implementation methods of the ensemble classifier in the P2P network, *one-by-one* ensemble classification and *parallel* ensemble classification. Furthermore, we propose a data space coverage based peer selection approach to reduce high the communication cost and large delay. We also design a two-layer index structure to efficiently support peer selection. A peer creates a local *Quad-tree* to index its local data and a super-peer creates a global *Quad-tree* to summarize its local indexes. Extensive experimental studies verify the efficiency and effectiveness of the proposed algorithms.

1. Introduction

It is a very important problem to classify and predict data on distributed systems due to higher application demands, for example, automatically organizing Web documents in a Peer-to-Peer (P2P) environment [5]. Although data classifications in centralized environments have been studied extensively in past few years, there still exist several challenging issues for data classification in a P2P network, such as *scalability*, *anytimeness*, and *asynchronism* [6]. Recently, a significant amount of distributed classification techniques have been proposed for classification in P2P networks [1–4]. The key idea of these articles is to learn an ensemble of classifiers on the local training data by combining all the classifiers to achieve high classification accuracies on unseen test data examples.

In this paper, we present a novel P2P ensemble classification framework based on OS-ELM to get an ensemble classifier by combining the classifiers learnt from peers in the P2P network. OS-ELM [7] is an online sequential *extreme learning machine* (ELM) that is an easy-to-use and effective learning algorithm for single hidden layer feedforward neural networks (SLFNs) [8–11]. ELM can only be used for batch learning, while OS-ELM can learn data one-by-one or chunk-by-chunk with varying or fixed chunk length, and the input weights and biases of OS-ELM with additive nodes are randomly generated and the output weights are analytically determined. As such, we can generate a distributed classifier by directly applying the incremental learning principle of OS-ELM.

There are two ways to achieve distributed ensemble learning in a P2P network, *one-by-one* ensemble learning and *parallel* ensemble learning. In the *one-by-one* ensemble learning, each classifier is learnt from all the data one peer by one peer. In the *parallel* ensemble learning, all the classifiers are learnt from all the data in parallel.

However, the *one-by-one* approach has a large network delay, while the *parallel* method has a high communication cost. Therefore, we propose a data space coverage based peer selection approach to solve these two problems. We also design a two-layer index structure to efficiently support peer selection. A peer creates a local *Quad-tree* to index its local data and a super-peer creates a global *Quad-tree* to summarize its local indexes.

The rest of this paper is organized as follows: Section 2 presents the distributed classification framework based on OS-ELM; Section 3 describes the implementation of distributed classification framework; Section 4 proposes the operations of peer dynamic; Section 5 gives the experimental results; and Section 6 concludes the paper and gives the future work.

* Corresponding author at: Key Laboratory of Medical Image Computing (NEU), MOE, Shenyang 110004, China.
E-mail address: gence23@gmail.com (Y. Sun).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.12.040

2. A distributed classification framework based on OS-ELM

ELM is a good learning method to classify data due to its many advantages, such as good generalization performance as well as improving the learning speed of neural network [8,16–18], maximizing the separating margin, and minimizing the training errors [12]. In the traditional P2P distributed classification methods [1–4], an ensemble classifier can be obtained by a two-step approach. The local classifiers are first obtained by training local data in each peer, then an ensemble classier can be obtained by combining these local classifiers. However, this kind of ensemble classifier has a lower classification accuracy since each local classifier only learns partial data. In order to solve this problem, we need an incremental learning method that is the main feature of OS-ELM. OS-ELM can learn data arriving or chunk-by-chunk with varying chunk size. Based on this feature, OS-ELM can learn all the data in a P2P network and get a higher classification accuracy.

2.1. Review of OS-ELM

Based on batch ELM, OS-ELM develops SLFNs with both additive and radial basis function (RBF) hidden nodes. OS-ELM can learn data one-by-one or chunk-by-chunk with varying or fixed chunk length, and the input weights and biases of additive nodes or centers and impact factors of RBF nodes are randomly generated and the output weights are analytically determined. For N arbitrary distinct samples $(\mathbf{x}_i,\mathbf{t}_i)$, where $\mathbf{x}_i=[x_{i1},x_{i2},\ldots,x_{in}]^T\in\mathbf{R}^n$ and $\mathbf{t}_i=[t_{i1},t_{i2},\ldots,t_{im}]^T\in\mathbf{R}^m$, since standard SLFN with $\tilde{N}$ hidden nodes can approximate these N samples with zero error [9], the output of a SLFN with hidden nodes can be represented by

$$f_{\tilde{N}}(\mathbf{x})=\sum_{i=1}^{\tilde{N}}\beta_i G(\mathbf{a}_i,b_i,\mathbf{x}),\quad \mathbf{x}\in\mathbf{R}^n,\ \mathbf{a}_i\in\mathbf{R}^n \tag{1}$$

where $\mathbf{a}_i$ and b_i are the learning parameters of hidden nodes, β_i is the output weight, and $G(\mathbf{a}_i,b_i,\mathbf{x})$ is the output of the ith hidden node with respect to the input $\mathbf{x}$. For additive hidden node, $G(\mathbf{a}_i,b_i,\mathbf{x})=g(\mathbf{a}_i\cdot\mathbf{x}+b_i)$, $b_i\in R$, where $\mathbf{a}_i$ is the input weight, b_i is the bias of the ith hidden node and $\mathbf{a}_i\cdot\mathbf{x}$ is the inner product. For RBF hidden node $G(\mathbf{a}_i,b_i,\mathbf{x})=g(b_i\|\mathbf{x}-\mathbf{a}_i\|)$, $b_i\in R^+$, where $\mathbf{a}_i$ and b_i are the center and impact factor of ith RBF node. R^+ is the set of all positive real values.

Through selecting the type of node, the activation function, and the number $\tilde{N}$ of hidden nodes, the OS-ELM can be implemented with data $\aleph=\{(\mathbf{x}_i,\mathbf{t}_i)|\mathbf{x}_i\in\mathbf{R}^n,\mathbf{t}_i\in\mathbf{R}^m,i=1,\ldots,N\}$ arrived one-by-one or chunk-by-chunk. The two-step OS-ELM algorithm is presented as follows:

Initialization phase: From the given training set $\aleph=\{(\mathbf{x}_i,\mathbf{t}_i)|\mathbf{x}_i\in\mathbf{R}^n,\mathbf{t}_i\in\mathbf{R}^m,\ i=1,\ldots,N\}$ a small chunk of training data $\aleph_0=\{(\mathbf{x}_i,\mathbf{t}_i)\}_{i=1}^{N_0}$ is used to initialize the learning, where $N_0\geq\tilde{N}$.

(1) For additive hidden nodes, assign random input weights $\mathbf{a}_i$ and bias b_i; for RBF hidden nodes, input weights is center $\mathbf{a}_i$ and impact factor is b_i, where $i=1,\ldots,\tilde{N}$.

(2) Calculate the initial hidden layer output matrix $\mathbf{H}_0$.

$$\mathbf{H}_0=\begin{bmatrix} G(\mathbf{a}_1,b_1,\mathbf{x}_1) & \cdots & G(\mathbf{a}_{\tilde{N}},b_{\tilde{N}},\mathbf{x}_1)\\ \vdots & \cdots & \vdots\\ G(\mathbf{a}_1,b_1,\mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_{\tilde{N}},b_{\tilde{N}},\mathbf{x}_{N_0})\end{bmatrix}_{N_0\times\tilde{N}} \tag{2}$$

(3) Estimate the initial output weight $\beta^{(0)}$:
Considering using the batch ELM algorithm, the solution to minimizing $\|\mathbf{H}_0\beta-\mathbf{T}_0\|$ is given by $\beta^{(0)}=\mathbf{P}_0\mathbf{H}_0^T\mathbf{T}_0$, where $\mathbf{P}_0=(\mathbf{H}_0^T\mathbf{H}_0)^{-1}=\mathbf{K}_0^{-1}$ and $\mathbf{T}_0=[\mathbf{t}_1,\ldots,\mathbf{t}_{N_0}]^T$.

(4) Set $k=0$, where k is the number of chunks that is trained currently.

Sequential learning phase: Present the $(k+1)$th chunk of new observations $\aleph_{k+1}=\{(\mathbf{x}_i,\mathbf{t}_i)\}_{i=(\sum_{j=0}^{k}N_j)+1}^{\sum_{j=0}^{k+1}N_j}$, where N_{k+1} denotes the number of observations in the kth chunk.

1. Calculate the partial hidden layer output matrix $\mathbf{H}_{k+1}$,

$$\mathbf{H}_{k+1}=\begin{bmatrix} G(\mathbf{a}_1,b_1,\mathbf{x}_{(\sum_{j=0}^{k}N_j)+1}) & \cdots & G(\mathbf{a}_{\tilde{N}},b_{\tilde{N}},\mathbf{x}_{(\sum_{j=0}^{k}N_j)+1})\\ \vdots & \cdots & \vdots\\ G(\mathbf{a}_1,b_1,\mathbf{x}_{\sum_{j=0}^{k+1}N_j}) & \cdots & G(\mathbf{a}_{\tilde{N}},b_{\tilde{N}},\mathbf{x}_{\sum_{j=0}^{k+1}N_j})\end{bmatrix}_{N_{k+1}\times\tilde{N}} \tag{3}$$

where $\mathbf{H}_{k+1}$ is calculated by the $(k+1)$th chunk of data $\aleph_{k+1}$.

2. Set $\mathbf{T}_{k+1}=\left[\mathbf{t}_{(\sum_{j=0}^{k}N_j)+1},\ldots,\mathbf{t}_{(\sum_{j=0}^{k+1}N_j)}\right]_{N_{k+1}\times m}^T$.

3. Calculate the output weight $\beta^{(k+1)}$. We have

$$\mathbf{K}_{k+1}=\mathbf{K}_k+\mathbf{H}_{k+1}^T\mathbf{H}_{k+1} \tag{4}$$

$$\beta^{(k+1)}=\beta^{(k)}+\mathbf{K}_{k+1}^{-1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1}-\mathbf{H}_{k+1}\beta^{(k)}) \tag{5}$$

From Eq. (5), we find that $\mathbf{K}_{k+1}^{-1}$ rather than $\mathbf{K}_{k+1}$ is used to compute $\beta^{(k+1)}$ from $\beta^{(k)}$. The update formula for $\mathbf{K}_{k+1}^{-1}$ is derived using the Woodbury formula [12].

$$\begin{aligned}\mathbf{K}_{k+1}^{-1}&=(\mathbf{K}_k+\mathbf{H}_{k+1}^T\mathbf{H}_{k+1})^{-1}\\&=\mathbf{K}_k^{-1}-\mathbf{K}_k^{-1}\mathbf{H}_{k+1}^T(\mathbf{I}+\mathbf{H}_{k+1}\mathbf{K}_k^{-1}\mathbf{H}_{k+1}^T)^{-1}\times\mathbf{H}_{k+1}\mathbf{K}_k^{-1}\end{aligned} \tag{6}$$

$\beta^{(k+1)}$ can be written by $\mathbf{P}_{k+1}$,

$$\mathbf{P}_{k+1}=\mathbf{P}_k-\mathbf{P}_k\mathbf{H}_{k+1}^T(\mathbf{I}+\mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k \tag{7}$$

$$\beta^{(k+1)}=\beta^{(k)}+\mathbf{P}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1}-\mathbf{H}_{k+1}\beta^{(k)}) \tag{8}$$

where $\mathbf{P}_{k+1}=\mathbf{K}_{k+1}^{-1}$.

4. Set $k=k+1$. Go to Sequential Learning Phase (1).

2.2. OS-ELM for distributed classification

The OS-ELM algorithm can handle data arriving chunk-by-chunk with varying chunk size [7]. By viewing the data of a peer as a chunk of the training dataset, we can easily design a naïve approach to generate a distributed classifier based on OS-ELM. Similar to OS-ELM, a classifier is initialized in a peer by learning the local data, and is propagated to other peers to sequentially learn the remaining data in the network. It is obvious that the naïve approach is not feasible due to a very large network delay. Therefore, we present an effective data space coverage based peer selection approach to avoid the problem.

In a P2P network, a peer manages a data space in which all the local data are located, and there are overlaps among the data spaces. In order to reduce the network delay of the naïve approach, we only need to find minimal peers that cover the whole data space to generate an ensemble classifier. We design a two-layer index structure to efficiently support peer selection. A common peer creates a local *Quad-tree* to index its local data and a super-peer creates a global *Quad-tree* to summarize its local indexes. Next, we introduce the peer selection method based on the two-layer index.

The peer selection can be done in two steps. (1) A super-peer selects minimal common peers based on its *Quad-tree* index in a

greedy way. The data spaces containing only one common peer are first found out from the *Quad-tree* index. If these found data spaces covers the data space managed by the super-peer, then these peers are just selected as the minimal selected peers. Otherwise, the data spaces containing two or more common peers are continuously found out till that the found data spaces covers the super-peer data space. (2) In a similar way, all the super-peers collaborate to find the minimal common peers that cover the whole data space from the selected common peers in the first step.

Next, we illustrate the peer selection procedure using the example of a *Quad-tree* index in a super-peer as shown in Fig. 1. First, the data spaces *ABD*, *AAA*, and *DAB* containing only one common peer are found out from the *Quad-tree* index. From the figure, we can see that the data spaces managed by the common peers P_1 and P_2 include *AD*, *BC*, *CB*, and *DB* as well. It is obvious that they do not cover the data spaces *BB* and *CC*. We continue to check the data spaces that involve two common peers from the remaining data spaces. In this case, the data spaces *BB* and *CC* both containing the common peers P_3 and P_5 are considered. P_3 involves the data spaces *AD*, *BB*, *CB*, and *CC* and P_5 involves the data spaces *BB* and *CC*. P_3 is selected as an additional minimal peers since P_3 is involved in more data spaces than P_5. If P_3 and P_5 involve the same data spaces, then the super-peer randomly selects P_3 or P_5. For the example in Fig. 1, P_1, P_2, and P_3 are selected as the minimal peers for sequential learning, and therefore avoiding high communication costs and large delay in a P2P network.

Algorithm 1 shows the distributed classifier based on OS-ELM. Firstly, a super-peer generates an initial classifier by calculating parameters $\mathbf{H}_0$ and $\beta^{(0)}$ on its local training dataset. Secondly, the super-peer searches from the selected common peers a common peer not to be learned, and propagates the initial classifier to the searched common peer. Thirdly, the common peer sequentially learns the initial classifier using its observations and calculates the new parameters on its local data. Finally, the common peer sends the new classifier to its associated super-peer.

Algorithm 1. Distributed classifier.

Require:
Training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m,\ i = 1, \ldots, N\}$;
Parameters $\mathbf{a}_i$ and b_i;
Ensure:
A distributed classifier;
1: $p = 0$;
2: **for** each selected peer **do**
3: Calculate the initial hidden layer output matrix $\mathbf{H}_p$;
4: Estimate the initial output weight $\beta^{(p)}$;
5: The super-peer searches from the selected common peers a common peer not to be learned;
6: **if** All the selected peers complete learning; **then**
7: Break;
8: **else**
9: Propagate the classifier to the next common peer;
10: **end if**
11: $p = p + 1$;
12: **end for**

3. Implementation of the distributed classification framework

This section proposes two ensemble distributed classification algorithms, *one-by-one* and *parallel* ensemble classifiers, to implement the distributed classification framework described in the last section.

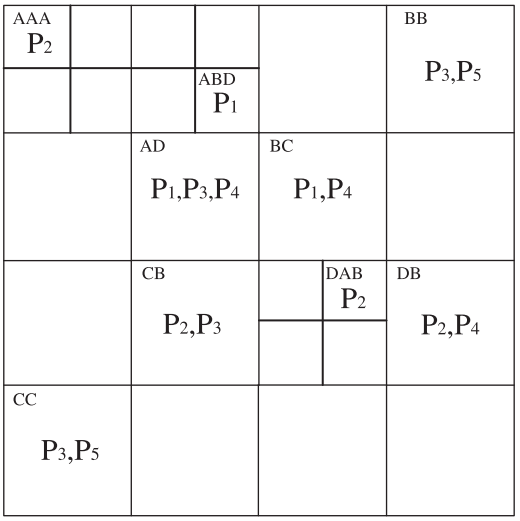

Fig. 1. An example of a *Quad-tree* index in a super-peer.

3.1. One-by-one ensemble classification in P2P networks

Let $P = \{p_1, p_2, \ldots, p_n\}$ be the selected peers by the peer selection method, and let $SP = \{sp_1, \ldots, sp_u\}$ be the super-peers in P and $CP = \{cp_1, \ldots, cp_v\}$ be the common peers in P, $u + v = n$. Let $C = \{c_1, \ldots, c_u\}$ be the initial classifiers in SP, where c_i denotes the initial classifier of super-peer sp_i. Assume that each super-peer in the P2P network has the list P of the selected peers. Then the *one-by-one* classification algorithm is described in the following three steps.

In the first step, sp_i $(1 \le i \le u)$ in SP propagates its classifier c_i to a common peer cp_j $(1 \le j \le v)$ of CP.

In the second step, once a common peer cp_j receives a classifier from super-peer sp_i, it begins to learn using its sequential input data. Then, cp_j sends back the new classifier to super-peer sp_i.

In the third step, if a super-peer sp_i receives from a common peer cp_j a classifier that has been already learnt by cp_j, then sp_i sends the classifier to the next common peer in the list CP till all the selected common peers have completed learning.

Till now, each super-peer in the set of SP gets a classifier that have learnt all the selected common peers. The ensemble classifier is finally obtained by merging all v classifiers in the voting method [15]. Algorithm 2 shows the *one-by-one* approach.

Algorithm 2. *One-by-one* ensemble of classifiers.

1: sp_i propagates its classifier c_i to cp_j;
2: cp_j begins to learn and sends back the new classifier to super-peer sp_i;
3: **if** a super-peer sp_i receives the message of cp_j; **then**
4: sp_i puts cp_j into waiting sequence;
5: **end if**
6: **if** Waiting sequence = null; **then**
7: Wait the completed learning information;
8: **else**
9: sp_i sends inactive message to the waiting candidates;
10: **end if**
11: sp_i propagates c_i to cp_{j+1};

3.2. Parallel ensemble classification in P2P networks

We first introduce necessaries for *parallel* ensemble classification, then propose detail steps for the *parallel* algorithm.

From Eqs. (6) to (8), we obtain

$$\beta^{(k+1)} = \beta^{(k)} + (\mathbf{H}_k^T\mathbf{H}_k + \mathbf{H}_{k+1}^T\mathbf{H}_{k+1})^{-1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta^{(k)}) \quad (9)$$

It is known that $\beta^{(k)}$, $\mathbf{H}_k$ and $\mathbf{H}_{k+1}$ are the main parameters to obtain $\beta^{(k+1)}$. The $\beta^{(k)}$ and $\mathbf{H}_k$ are obtained in the previous peers, while $\mathbf{H}_{k+1}$ is calculated by $\beta^{(k)}$ and $\mathbf{H}_k$ in the current peer. As such, if each selected common peer can calculate $\beta^{(k)}$ and $\mathbf{H}_k$, then each super-peer can analytically calculate $\beta^{(k+1)}$ using OS-ELM, and therefore a final classifier can be obtained in a parallel way. The *parallel* ensemble classification works in following three steps.[1]

In the first step, each selected common peer calculates the initial hidden layer output matrix $\mathbf{H}_0$ and estimates the initial output weight $\beta^{(0)}$ using its local training dataset.

In the second step, all the selected common peers propagate $\mathbf{H}_0$ and $\beta^{(0)}$ to their associated super-peers.

In the third step, all the super-peers broadcast their $\mathbf{H}_0$ and $\beta^{(0)}$ to obtain an ensemble of the classifiers.

Algorithm 3 shows the detail of *parallel* ensemble classification.

Algorithm 3. *Parallel* ensemble of classifiers.

Require:
Training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$;
Parameters $\mathbf{a}_i$ and b_i;
Ensure:
An ensemble of classifiers;
1: Assign $\mathbf{a}_i$ and b_i;
2: Calculate the initial hidden layer output matrix $\mathbf{H}_0$, and estimate the initial output weight $\beta^{(0)}$ in each selected common peer;
3: Propagate $\mathbf{H}_0$ and $\beta^{(0)}$ to super-peers;
4: Super-peers broadcast their $\mathbf{H}_0$ and $\beta^{(0)}$ propagated by their associated selected common peers each other;
5: Estimate multiple output weights $\beta^{(k+1)}$;

4. Churn issues in P2P networks

Since a P2P system is a dynamic network, our proposed distributed classification algorithms should adapt to the dynamic scenario. In this section, we deal with dynamic operations for common peers.[2]

Join: Consider that peer *A* wants to join the super-peer P2P network. Peer *A* first finds a super-peer *B* with closest ID to *A* in the P2P network using the routing algorithm in [13]. Then peer *A* updates its routing table according to super-peer *B*'s. Finally super-peer *B* inserts *A* into the sequential learning queue if *A* is a selected peer based on the peer selection method.

Departure: Consider that peer *A* wants to leave from the super-peer P2P network. The super-peer *B* of peer *A* takes over *A*'s data for classification if *A* is a selected peer.

Failure: Each super-peer maintains a local queue and some statistical information, by considering some factors such as which common peers are selected peers; which common peers are most frequently learned; and which peers are often failed. A super-peer analyzes failure probabilities of all the associated common peers according to these factors and keeps *k* common peers with the highest probability of failure in the front of the queue. The super-peer periodically monitors the *k* common peers to see if they fail. Once a peer fails, then the super-peer takes over the data of the failed common peer for classification.

[1] In the original OS-ELM, the parameters $\mathbf{a}_i$ and b_i of Eq. (1) are randomly generated. In order to perform learning in parallel, it is required that each selected common peer generates the same values of $\mathbf{a}_i$ and b_i.

[2] We only consider dynamic operations of common peers, since super-peers are stable in the P2P network.

5. Performance evaluation

In this section, we evaluate the performances of the algorithms proposed in this paper. All the evaluations are carried out in MATLAB 7.0 in an AMD Athlon Dual Core processor with 2.6 GHz and 2 GB RAM.

All the experiments are conducted using three real datasets download from the UCI repository [14], including Image Segment, Satellite Image, and Primate Splice-Junction Gene Sequences (DNA). Let *Ncp* and *Nsp* be the number of common peers and the number of super-peers in the P2P network, respectively. The P2P network connects the *Nsp* super-peers in a random graph topology. In our experimental setting, the P2P network has at least 500 common peers while *Nsp* is set to $10\% \times Ncp$. We use the sigmoidal additive activation function $G(\mathbf{a},b,\mathbf{x}) = 1/(1+\exp(-(\mathbf{a}\cdot\mathbf{x}+b)))$ and the Gaussian RBF activation function $G(\mathbf{a},b,\mathbf{x}) = \exp(-\|\mathbf{x}-\mathbf{a}\|^2/b)$ for OS-ELM.

Our experiments mainly evaluate the following two aspects: (1) the accuracy and response time of P2P classification in the cases of *Quad-tree* index and no index, and (2) the effect of parameters on the performance. For the first aspect, we compare our proposed algorithms with centralized OS-ELM. For the second aspect, we mainly consider effect of Quad-tree height on the performance of the peer selection method, and effect of parameter *Ncp* on communication costs of the *one-by-one* and *parallel* ensemble classification algorithms.

5.1. Performance without Quad-tree indexes

In this experiment, we compare the accuracy and response time of our algorithms with centralized OS-ELM in the case of no *Quad-tree* index. To test the classification accuracy, 80% data in each peer are randomly selected to train classifiers with two activation functions, while the rest 20% data are used as the test examples to compute the accuracies of the classifiers. The number of hidden neurons is set to 180, 400 and 200 for datasets Image Segment, Satellite Image, and DNA, respectively. Table 1 shows classification accuracies of our approaches and centralized OS-ELM. From the table we can see that the accuracy in a static P2P network is higher than that in a dynamic P2P network. As described in Section 4, a super-peer analyzes failure probabilities of all the associated common peers and takes over only *k* peer failures with the highest failure probabilities, and therefore not all the failed peers are used to train the classifiers. Contrastively, the accuracy in a static P2P network comes close to the accuracy of centralized OS-ELM.

Table 1
Classification accuracy in % in the case of no index.

Datasets	Activation functions	Static P2P networks		Dynamic P2P networks		Centralized OS-ELM
		One-by-one	Parallel	One-by-one	Parallel	
Image Segmentation	Sigmoid	92.90	93.05	88.35	86.97	93.00
	RBF	92.57	92.21	87.63	89.45	92.70
Satellite Image	Sigmoid	86.70	86.93	84.36	85.62	87.02
	RBF	87.97	87.10	83.78	83.41	87.95
DNA	Sigmoid	90.32	90.23	85.24	85.51	91.12
	RBF	90.45	90.98	86.64	85.25	91.79

Table 2 shows the training time for different approaches. Obviously, the *parallel* ensemble classification algorithm needs shorter training time than the *one-by-one* method. In addition, since a part of failure peers are not taken over by super-peers, the training time of dynamic P2P networks is shorter than the static environment.

5.2. Performance with Quad-tree indexes

In this experiment, we compare the accuracy and response time of our algorithms with centralized OS-ELM in the case of *Quad-tree* index. Table 3 gives classification accuracies of our proposed approaches. By comparing Table 3 and Table 1, we can see that the classification accuracy in a P2P network with *Quad-tree* index is slightly lower than the classification accuracy in the case of no index. This is because super-peers select minimal common peers to cover the whole data space, and part of data are not learnt.

Table 4 shows the response time of our approaches with *Quad-tree* index. From the table, we can see that *Quad-tree* plays an important role in a dynamic P2P network. Since super-peers can choose effective common peers to replace the whole network with the help of the global *Quad-tree* index, the number of peers that participate in computation is significantly reduced, and therefore the communication costs are reduced dramatically.

Table 2
Training time in seconds in the case of no index.

Datasets	Activation functions	Static P2P networks		Dynamic P2P networks		Centralized OS-ELM
		One-by-one	Parallel	One-by-one	Parallel	
Image Segmentation	Sigmoid	147.56	0.187	128.23	0.143	133.02
	RBF	283.51	0.778	252.18	0.620	207.97
Satellite Image	Sigmoid	185.97	0.465	176.53	0.387	137.52
	RBF	369.61	2.775	352.87	2.367	304.33
DNA	Sigmoid	178.30	0.206	155.63	0.156	141.37
	RBF	195.30	0.323	177.32	0.165	176.48

Table 3
Classification accuracy in % in the case of Quad-tree indexes.

Datasets	Activation functions	Static P2P networks		Dynamic P2P networks		Centralized OS-ELM
		One-by-one	Parallel	One-by-one	Parallel	
Image Segmentation	Sigmoid	84.35	85.18	80.24	81.05	93.00
	RBF	85.65	84.41	81.10	83.56	92.70
Satellite Image	Sigmoid	80.70	81.93	78.56	77.88	87.02
	RBF	81.54	82.18	80.83	80.46	87.95
DNA	Sigmoid	83.62	84.38	81.64	81.22	91.12
	RBF	83.75	85.30	82.08	83.00	91.79

Table 4
Training time in seconds in the case of Quad-tree indexes.

Datasets	Activation functions	Static P2P networks		Dynamic P2P networks		Centralized OS-ELM
		One-by-one	Parallel	One-by-one	Parallel	
Image Segmentation	Sigmoid	65.31	0.163	58.65	0.102	133.02
	RBF	78.43	0.469	72.65	0.254	207.97
Satellite Image	Sigmoid	70.63	0.242	65.35	0.121	137.52
	RBF	126.36	1.178	88.80	0.684	304.33
DNA	Sigmoid	68.41	0.125	55.77	0.090	141.37
	RBF	72.85	0.265	59.50	0.200	176.48

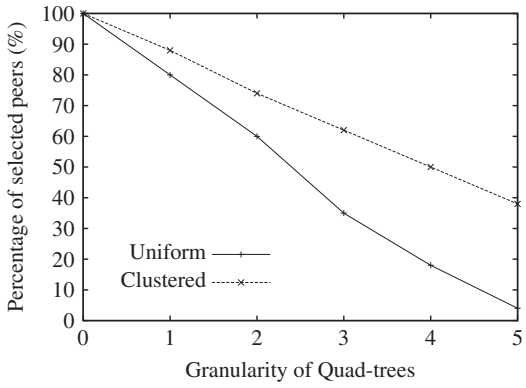

Fig. 2. Percentage of selected peers vs. heights of *Quad-tree*.

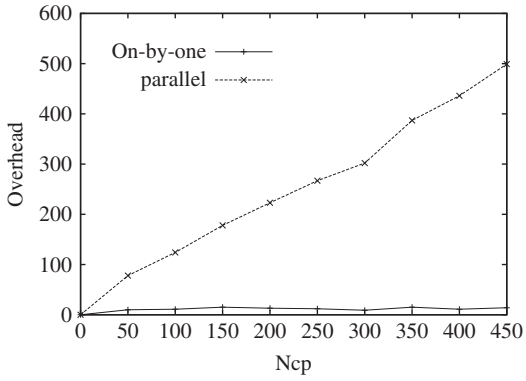

Fig. 3. Communication costs vs. *Ncp*.

5.3. Effect of other parameters

In this section, we study the effect of Quad-tree height on the performance of the peer selection method and the effect of parameter *Ncp* on the communication costs of the *one-by-one* and *parallel* ensemble classification algorithms. Fig. 2 gives the percentage of the selected peers in all the common peers for different data distributions while varying the height of *Quad-tree*. From this figure we can see that the number of selected peers decrease obviously with increasing of the height of *Quad-tree*. *Uniform* data distribution has a worse peer selection performance than *clustered* data distribution. This is because data are almost distributed in different sub-spaces when the data are uniformly distributed.

Fig. 3 gives the network overhead for our algorithms while varying the number of common peers *Ncp*. In this figure, we can observe that no matter how many peers in the network, the overhead in *one-by-one* ensemble algorithm is very small. However, the network overhead of the *parallel* ensemble algorithm increases with increasing of *Ncp*.

6. Conclusions and future work

This paper proposes an OS-ELM based ensemble classification framework for distributed classification in a super-peer P2P

network. In the framework, we apply the incremental learning principle of OS-ELM to the hierarchical P2P network to generate an ensemble classifier. A data space coverage based peer selection approach and a two-layer index structure are also proposed to reduce the high communication cost and large delay. We implement the framework using two kinds of ensemble classifiers in the P2P network, *one-by-one* ensemble classification and *parallel* ensemble classification. Finally, extensive experimental studies verify the efficiency and effectiveness of the proposed algorithms. In the future, we plan to propose new approaches to deal with the classification of high-dimensional data in P2P networks.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grant nos. 60873011, 60933001 and 61025007), National Basic Research Program of China (Grant no. 2011CB302200-G), the 863 High Technology Program (Grant no. 2009AA01Z150), and the Fundamental Research Funds for the Central Universities (Grant nos. N090104001 and N090304007).

References

[1] H.-H. Ang, V. Gopalkrishnan, S. Hoi, W.-K. Ng, Cascade RSVM in peer-to-peer networks, in: 2008 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML'2008), Antwerp, Belgium, September 15–19, 2008.
[2] V. Gorodetskiy, O. Karsaev, V. Samoilov, S. Serebryakov, Agent-based service oriented intelligent P2P networks for distributed classification, in: 2006 International Conference on Hybrid Information Technology (ICHIT'2006), Cheju Island, Korea, November 9–11, 2006.
[3] P. Luo, H. Xiong, K. Lü, Z. Shi, Distributed classification in peer-to-peer networks, in: 2007 ACM SIGKDD International Conference on Knowledge Discovery in Databases (KDD'2007), San Jose, CA, USA, August 12–15, 2007.
[4] H.-H. Ang, V. Gopalkrishnan, S. Hoi, W.-K. Ng, A. Datta, Classification in P2P networks by bagging cascade RSVMs, in: 2008 Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P'2008), Auckland, New Zealand, August 23, 2008.
[5] S. Siersdorfer, S. Sizov, Automatic document organization in a P2P environment, in: 2006 European Conference on IR Research (ECIR'2006), London, UK, April 10–12, 2006.
[6] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, H. Kargupta, Distributed data mining in peer-to-peer networks, IEEE Internet Computing 10 (4) (2006) 18–26 (Special issue on Distributed data mining).
[7] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (6) (2006) 1411–1423.
[8] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks (IJCNN'2004), Budapest, Hungary, July 25–29, 2004.
[9] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[10] G.-B. Huang, C.-K. Siew, Extreme learning machine: RBF network case, in: Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004), Kunming, China, December 6–9, 2004.
[11] G.-B. Huang, C.-K. Siew, Extreme learning machine with randomly assigned RBF kernels, International Journal of Information Technology 11 (1) (2005) 16–24.
[12] G.-H. Golub, C.-F.V. Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[13] E. Tanin, A. Harwood, H. Samet, Using a distributed quadtree index in peer-to-peer networks, International Journal on Very Large Data Bases (VLDB Journal) 16 (2) (2007) 165–178.
[14] C. Blake, C. Merz, UCI Repository of Machine Learning Databases, Department of Information and Computer Science, University of California, Irvine, CA, 1998 [Online]. Available: 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉.
[15] P.-K. Chan, S.-J. Stolfo, A comparative evaluation of voting and meta-learning on partitioned data, in: 1995 International Conference on Machine Learning (1995'ICML), Tahoe City, CA, USA, July 9–12, 1995.
[16] N.-Y. Liang, P. Saratchandran, G.-B. Huang, N. Sundararajan, Classification of mental tasks from EEG signals using extreme learning machines, International Journal of Neural Systems 16 (1) (2006) 29–38.
[17] C.-W.T. Yeu, M.-H. Lim, G.-B. Huang, A. Agarwal, Y.S. Ong, A new machine learning paradigm for terrain reconstruction, IEEE Geoscience and Remote Sensing Letters 3 (3) (2006) 382–386.
[18] R. Zhang, G.-B. Huang, N. Sundararajan, P. Saratchandran, Multi-category classification using extreme learning machine for microarray gene expression cancer diagnosis, IEEE/ACM Transactions on Computational Biology and Bioinformatics 4 (3) (2007) 485–495.

Yongjiao Sun received the B.S. degree and M.S. degree in Computer Science from Northeastern University, China, in 2006 and 2008 respectively. Currently, he is a Ph.D. candidate at Computer Science Department of the Northeastern University. His main research interests in P2P data management, and probabilistic database.

Ye Yuan received the B.S. degree and M.S. degree in Computer Science from Northeastern University, China, in 2004 and 2007 respectively. Currently, he is a Ph.D. candidate at the Computer Science Department of Northeastern University, and he is also a visiting scholar at Department of Computer Science and Engineering in Hong Kong University of Science and Technology. His research interests include graph database, probabilistic database, P2P data management, and data piracy.

Guoren Wang is a lecturer at Northeastern University, China. He received his B.E., M.E. and Ph.D. degrees from Northeastern University in 1988, 1991 and 1996, respectively. His research interests include machine learning, data mining, data management, bioinformatics and multimedia technology. He has published more than 80 papers in international conferences and journals.