Towards Positive Unlabeled Learning for Parallel Data Mining: A Random Forest Framework

Conference Paper · December 2014

DOI: 10.1007/978-3-319-14717-8_45

2 authors:

Chen Li
Monash University (Australia)
21 PUBLICATIONS **94** CITATIONS
SEE PROFILE

Xue-Liang Hua
Monash University (Australia)
1 PUBLICATION **1** CITATION
SEE PROFILE

Towards Positive Unlabeled Learning for Parallel Data Mining: A Random Forest Framework

Chen Li[1,2] and Xue-Liang Hua[3,⋆]

[1] Department of Biochemistry and Molecular Biology, Monash University, VIC 3800, Australia
[2] College of Information Engineering, Northwest A&F University, Yangling 712100, China
Chen.Li@monash.edu
[3] Faculty of Information Technology, Monash University, VIC 3800, Australia
Xueliang.Hwa@gmail.com

Abstract. Parallel computing techniques can greatly facilitate traditional data mining algorithms to efficiently tackle learning tasks that are characterized by high computational complexity and huge amounts of data, to meet the requirement of real-world applications. However, most of these techniques require fully labeled training sets, which is a challenging requirement to meet. In order to address this problem, we investigate widely used Positive and Unlabeled (PU) learning algorithms including PU information gain and a newly developed PU Gini index combining with popular parallel computing framework - Random Forest (RF), thereby enabling parallel data mining to learn from only positive and unlabeled samples. The proposed framework, termed PURF (Positive Unlabeled Random Forest), is able to learn from positive and unlabeled instances and achieve comparable classifcation performance with RF trained by fully labeled data through parallel computing according to experiments on both synthetic and real-world UCI datasets. PURF is a promising framework that facilitates PU learning in parallel data mining and is anticipated to be useful framework in many real-world parallel computing applications with huge amounts of unlabeled data.

Keywords: PU information gain, PU Gini index, random forest, parallel data mining.

1 Introduction

Processing of huge amounts of data and high-complexity computations are the two emerging major challenges faced by many real-world applications of data mining in recent years. Accordingly, parallel computing techniques are developed and widely used to address data mining tasks that require time-consuming computation and processing of large databases by dispatching the learning task to several CPUs / jobs to allow separate running. A large number of experiments in different research areas and applications have benefited from the success and effectiveness of these developed parallel algorithms [1][2][3]. Among these, random forest (RF) [4] is one such attractive parallel algorithm that can be readily extended for parallel computing purposes by generating many trees in different CPUs / jobs.

⋆ Corresponding author.

X. Luo, J.X. Yu, and Z. Li (Eds.): ADMA 2014, LNAI 8933, pp. 573–587, 2014.

Experiments on many real-world datasets have proved that RF can achieve satisfactory performance and flexible extendibility for parallel computing. However, to date, the majority of parallel algorithms are trained by fully labeled instances, while it is well known that labeling data is time-consuming and requires considerable effort. Prediction of post-translational modification sites (PTMs) provides is a good example that illustrates the limitation and incompatibility of traditional classifications that rely on fully labeled samples: First, millions of protein sequences from different species are available; Second, identifying (labeling) PTMs in each of these proteins requires extensive experimental efforts; and finally, researchers sometimes are more interested in certain types of PTMs (e.g. phosphorylation sites) that might be better regarded as positive samples for data mining purposes.

With the goal of tackling the tasks described above, in this study, we propose a novel computational framework, termed PURF (Positive Unlabeled Random Forest), for parallel computing to learn from positive and unlabeled samples effectively. This framework combines PU learning techniques including widely used PU information gain (PURF-IG) [5] and newly developed PU Gini index (PURF-GI) with an extendable parallel computing algorithm (i.e. RF). Empirical experiments performed on both synthetic and real-world UCI datasets indicate that both these two PU learning techniques perform comparably with RF model trained by fully labeled data, suggesting that in the case of parallel computing, PURF has a strong capability to learn data from large amounts of samples with unknown class and lack of labeled negative samples. To the best of our knowledge, this work represents the first study of positive and unlabeled learning with both PU information gain and PU Gini index for the parallel data mining scenario.

The rest of this paper is organized as follows: Section 2 will review some representative works in terms of PU Learning and parallel computing. A new PU Gini index will be proposed in Section 3. The corresponding algorithms for PURF and its parallel implementation will be described in Section 4. We conduct experiments to compare the performance of supervised RF, PURF-GI and PURF-IG in Section 5, which is followed by conclusions and future work drawn in Section 6.

2 Related Works

We briefly summarize three major directions in this area: **(1) Single classifier for PU learning.** Denis *et al*. developed a new information gain-based method with only positive and unlabeled data and applied it to decision tree [5]. Similar technique was also developed with Naive Bayesian learner in [6]. **(2) Strategies for PU learning (such as two-step strategy and unlabeled sample weighting).** In this direction, traditional classifiers or algorithms will not be changed or modified. Elkan *et al*. developed a method to weigh unlabeled data with the assumption that training data are an incomplete set of positive and unlabeled data [7]. Margin maximization in SVM was also used to incrementally label negative data with the aid of MC (Mapping Convergence) proposed by Yu. [8]. For text classification, a two-step strategy was widely used, with which reliable negative documents were extracted from unlabeled documents and used to train a classifier together with labeled positive documents [9][10][11]. Then previously labeled

negative documents could be refined and more reliable negative documents would be retrieved. This process would not stop until a satisfactory threshold was reached.

On the other hand, the research works of parallel data mining techniques can be categorized into following three groups: **(1) Modification of traditional data mining techniques.** For apriori association rules mining, there exist three types of methods, including Count Distribution, Data Distribution and Candidate Distribution [2][12][13]. Zaki *et al*. proposed a parallel algorithm for vertical associate rules mining based on Eclat, in order to aggregate and take advantage of limited computing resources [14]. FP-tree [15] is another classic model to mine association rules. Two algorithms have been developed to grow multiple FP-trees in a parallel manner to different parts of transactions [16][17]. Moreover, Parthasarathy *et al*. proposed a new parallel association rules mining technique in shared-memory system [1]. To tackle imbalanced data mining task, Cheung *et al*. proposed a novel parallel algorithm using a distributed nothing-parallel system to explore association rules [18]. **(2) Parallel computing in computer applications.** For graphics processing, Li *et al*. developed three methods based on Compute Unified Device Architecture (CUDA) to accelerate three data mining algorithms (CU-Apriori, CU-KNN and CU-K-means) in CUDA [2]. For semantic retrieval, Chen *et al*. proposed a parallel computing approach to build engineering concept space, thereby addressing the scalability issue related to large-scale information retrieval [3]. **(3) Parallel computing in bioinformatics.** Parallel computing has been widely used in bioinformatics and computational biology, such as molecular dynamic simulation [19][20] and tool development for biological data analysis [21].

It should be noted that for most of the parallel algorithms, the models are trained by fully labeled data that is difficult to collect in real-world applications.

3 Positive Unlabeled Gini Index

As mentioned in our previous study [22], the PU learning task can be described as follows. Let us write a dataset S ($|S| = n$) that has only positive and unlabeled samples, $S = \{s_1, s_2, \ldots, s_n\}$, where $s_i =< A_i, C_i, y_i >$ is a sample in $S(1 \leq i \leq n)$. Here, $A_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\}$ is an attribute vector with m attributes; C_i denotes the class label of s_i (positive class: $C_i = +1$; negative class: $C_i = -1$); y_i represents whether C_i is known to the model (class known: $y_i = +1$; class unknown: $y_i = -1$). Since for positive unlabeled learning, only positive samples are labeled, if $y_i = +1$, then $C_i = +1$; while if $y_i = -1$, the true class label of s_i is unknown. The PU learning aims at distinguishing positive class ($C_i = +1$) from non-positive class ($C_i \neq +1$) in a given dataset. In the case of binary classification, non-positive class is the negative class. While in the case of multi-class classification, non-positive class can be the set of all classes except the positive class.

Random forest [4] is an ensemble of decision trees built on random bootstrapping of training datasets. Final classification decision of a test sample is determined by the average value of the possibility of all classes. Since RF uses Gini index [23] to decide the split criteria, inspired by estimation method of proportion of positive and negative samples proposed by Denis *et al*. [5], we propose the PU Gini index in this study that explores only positive and unlabeled samples.

Given a learning task with only positive and unlabeled samples, let $PosLevel$ denote the proportion of positive samples in the learning task and S_{node} as the dataset filtered in the current node node in tree T. Note that $PosLevel$ is the proportion of positive samples in this learning scenario. Therefore, $PosLevel$ is unknown and can vary. The details of $PosLevel$ will be discussed in the following section. According to the definition of Gini index, we have

$$PUGini(S_{node}) = 1 - \sum_{i=1}^{|C|} p_i^2. \tag{1}$$

Under the PU learning scenario, there are only two classes: positive class and non-positive class. Let p_1 and p_2 denote the estimates of proportion of positive and non-positive samples in S_{node}, respectively. According to the PU information gain in [5], we have

$$p_1 = \min\{\frac{|POS_{node}|}{|POS|} \times PosLevel \times \frac{|UNL|}{|UNL_{node}|}, 1\}$$

$$p_2 = 1 - p_1, \tag{2}$$

where $|POS|$ and $|UNL|$ are the numbers of positive and unlabeled samples in the training dataset, respectively. $|POS_{node}|$ and $|UNL_{node}|$ are the numbers of positive and unlabeled samples filtered into the current node, respectively. Then for each attribute $a_j (1 \leq j \leq m)$ in A, according to the split point, S_{node} can be divided into two subsets, $S_{node}(a_j \leq splitPoint)$ and $S_{node}(a_j > splitPoint)$. Therefore we have

$$\begin{aligned} PUGini_{a_j} = & \frac{|S_{node}(a_j \leq splitPoint)|}{|S_{node}|} \times \\ & PUGini(S_{node}(a_j \leq splitPoint)) + \\ & \frac{|S_{node}(a_j > splitPoint)|}{|S_{node}|} \times \\ & PUGini(S_{node}(a_j > splitPoint)). \end{aligned} \tag{3}$$

Finally, for the current node with the attribute $a_j (a_j \in A)$, the PU Gini can be described as follows

$$\Delta PUGini_{a_j}(S_{node}) = PUGini(S_{node}) - PUGini_{a_j}. \tag{4}$$

Among m attributes, the one with the maximal $\Delta PUGini_{a_j}(S_{node})$ should be chosen as the splitting attribute.

4 Positive Unlabeled Random Forest

4.1 Framework

In this section, we describe the developed framework based on RF to learn from positive and unlabeled samples. In particular, PURF uses bootstrapping (with replacement)

to select the training samples for each tree in forest. Therefore after bootstrapping, approximately 2/3 of samples are collected as the training data. In each PU tree T_i in forest, there are nine sub-trees built in accordance with different values of $PosLevel$ (ranging from 0.1 to 0.9 with a step size of 0.1). Then the rest 1/3 of the data will be used to determine the final output of each tree T_i in forest from the nine sub-trees. Algorithm 1 shows the framework of the proposed PURF. Step 1 is used to

Algorithm 1. PURF Framework

Input:

training dataset including positive and unlabeled samples, S;
the feature space of S, A;
the number of trees in forest, n_{tree};

Output:

positive and unlabeled random forest F.

1: $F = \phi$;
2: **for** each $i \in [1, n_{tree}]$ **do**
3: initialize a tree T_i with only one node (the root);
4: $S_{train} = Bootstrapping(S)$ [4];
5: $S_{validate} = S - S_{train}$;
6: **for** each $j \in [1, 9]$ **do**
7: $PosLevel = \frac{j}{10}$;
8: $T_{ij} = BuildTree(T_{ij}, PosLevel, S_{train}, A)$;
9: $T_i = T_i \bigcup T_{ij}$
10: $T_\theta = GetBestTree(T_i, S_{validate})$
11: **end for**
12: $F = F \bigcup T_\theta$;
13: **end for**
14: **return** F;

initiate the forest F, $F = \{T_1, T_2, \ldots, T_{n_{tree}}\}$. From steps 2 to 13, a forest F with n_{tree} trees will be generated and returned. As mentioned above, each tree $T_i (1 \leq i \leq n_{tree})$ in F has nine sub-trees with different $PosLevel$ values. The function $BuildTree(T_{ij}, PosLevel, S_{train}, A) (1 \leq j \leq 9)$ is used to generate sub-trees for T_i. Then one of the nine trees will be chosen as the best tree according to the estimate value by $GetBestTree(T_i, S_{validate})$, which will be discussed in the next section. Details of the $BuildTree(T_{ij}, PosLevel, S_{train}, A)$ are shown in Algorithm 2. The key part of Algorithm 2 is to calculate the PU Gini index according to formula (4) or PU information gain [5]. Then the attribute in A with the maximal PU Gini index will be chosen as the splitting attribute. The growth of the tree halts when one of the two criteria is met: (1) the samples filtered into this node are 'pure' (i.e., positive or unlabeled) or (2) the number of samples in the current node is smaller than a pre-set threshold. Then the class of the leaf is determined by p_1 and p_2 from formula (2).

4.2 Tree Selection

As discussed above, each PU decision tree $T = \{T_1, T_2, \ldots, T_9\}$ generated in the forest F has nine sub-trees based on nine values of $PosLevel$ and one sub-tree should

Algorithm 2. $BuildTree(T, PosLevel, S_{train}, A)$

// Refer to Algorithm 1 for the details of input parameters;
1: Generate new feature subspace $A^{'} \in A$ by random sampling with replacement;
2: **for** each $A^{'}_{m} \in A$ **do**
3: Calculate PU Gini index (formula (4)) or PU information gain according to [5];
4: **end for**
5: Choose attribute A_{split} with the maximal PU Gini index / PU information gain value as splitting attribute;
6: Compute the splitting value $splittingPoint$ for A_{split};
7: Generate child nodes splitting from $splittingPoint$;
8: **for** each child node **do**
9: $S_{train} = getDataInCurrentNode()$;
10: $BuildTree(T, PosLevel, S_{train}, A)$;
11: **end for**

be chosen as a representative decision tree of T in the forest F. Here, based on [5] and our previous study [22], we use the following four statistics to calculate $e(T)$ to choose the best sub-tree:

1. Number of positive samples in $S_{validate}$, n_{POS};
2. Number of unlabeled samples in $S_{validate}$, n_{UNL};
3. Number of positive samples to be predicted as non-positive, $n_{POS \longrightarrow NP}$;
4. Number of unlabeled samples to be predicted as positive, $n_{UNL \longrightarrow POS}$.

Then for each T_k in T,

$$e(T_k) = \frac{n_{POS \longrightarrow NP}}{n_{POS}} + \frac{n_{UNL \longrightarrow POS}}{n_{UNL}}. \tag{5}$$

Finally, the output of T, T_θ is

$$\theta = \underset{k}{\operatorname{argmin}}(e(T_k)). \tag{6}$$

5 Empirical Study

In order to evaluate the performance of our proposed algorithm, we conducted benchmarking experiments using both synthetic and real-world datasets to assess the ability of PURF to learn from unlabeled samples and compared with RF trained using fully labeled data.

We have implemented PURF with Python[1] based on the scikit-learn package[2] and both two PU techniques including PU Gini index (PURF-GI) and PU information gain (PURF-IG) were tested as split functions. RF models used to compared with PURF-GI and PURF-IG were implemented with Gini index and information gain, respectively. The experiments were conducted on two Apple iMac machines with Intel Core i5 Quad

[1] `http://www.python.org/`
[2] `http://scikit-learn.org/stable/`

Core CPU, with 12G and 24 G RAM, respectively and one Apple MacBook Pro with Intel Core Duo CPU, and 4G RAM. The machine with Intel Core Duo CPU was only used to conduct the experiments of running time comparison based on different number of cores.

We measured the classification performance of PURF-GI, PURF-IG and RF using Accuracy, F1 and AUC, which are commonly used for measuring the performance of PU classifiers [7][22]. Meanwhile, the time and space complexity were also evaluated by calculating the running time and counting the number of leaves and inner nodes of both algorithms.

5.1 Datasets and Performance Evaluation

Both synthetic and real-world datasets were used to evaluate the performance of our proposed PURF algorithm. We used a waveform generator to construct synthetic data, while the three real-world datasets (Spambase[3], Breast Cancer[4] and Ionosphere[5]) described in Table 1 were downloaded from the UCI website.

Table 1. real-world datasets

Dataset	#Ins.	#Attr.	Positive class	#Positive ins.
Spambase	4601	57	spam	1813
Breast cancer	699	9	malignant	241
Ionosphere	351	34	g	225
			b	126

For transferring from fully labeled data to positive and unlabeled data, we defined the probability $transfer(s)$ that one positive instance s becomes unlabeled with the given percentage of unlabeled data $ratio(UNL)$ and the assumption that $ratio(UNL)$ is larger than the proportion of negative samples in the training dataset:

$$transfer(s) = \frac{|S_{train}| \times (1 - ratio(UNL))}{|S_{train_{pos}}|}, \tag{7}$$

where $|S_{train}|$ is the number of training samples, while $|S_{train_{pos}}|$ represents the number of positive samples in $|S_{train}|$. If $r > transfer(s)$, s will be transferred to unlabeled sample.

10-fold cross-validation was applied to all experiments conducted in this study. In each fold, we generated PU datasets for $ratio(UNL) = 70\%, 80\%$ and 90% using the fully labeled dataset with formula (7). Therefore, the performance of PURF-GI/PURF-IG and RF reported in this paper were averaged over the forest generated on these fully labeled and PU datasets of the ten folds. In addition, all the experiments except the

[3] `http://archive.ics.uci.edu/ml/datasets/Spambase`

[4] `http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)`

[5] `http://archive.ics.uci.edu/ml/datasets/Ionosphere`

parallel analysis were performed in a parallel manner with quad cores and 10 jobs. The number of jobs is a pre-defined parameter in the Python implementation[6] of RF.

5.2 Portrait of PURF-GI

In this section, we would like to apply both synthetic and real-world UCI datasets to summarise the character of PURF with newly developed PU Gini index in terms of different sizes of training data, tree selection method and time/space complexity.

Number of Samples n_{sample}. The purpose of this group of experiments is to examine the relationship between the number of training samples and the corresponding classification performance of PURF-GI and RF. To do this, we used Waveform Generator to build synthetic datasets. By default, datasets constructed by the waveform generator have three classes: 0, 1 and 2. In this section, every class was considered as positive class once, and the other two were combined as negative class. We constructed three different datasets 20 times whose $n_{sample} = 1,000$, $5,000$ and $10,000$, respectively. When conducting experiments, we fixed $n_{tree} = 20$ and $ratio(UNL) = 70\%$, 80% and 90%, respectively. The results are shown in Fig. 1. With the increase of n_{sample}, the performance (in terms of Accuracy, AUC and F1) of both PURF-GI and RF generally increased. In general, despite the different $ratio(UNL)$, the performances of PURF-GI and RF were very close to each other, especially for AUC value. For example, when the positive class was 0 and $n_{sample} = 10,000$, the differences of Accuracy, AUC and F1 between PURF-GI and RF were 4.9%, 0.48% and 2.1%, respectively. Note that despite the $ratio(UNL) = 90\%$ (approximately $10,000 \times 0.9 \times 0.9 = 8,100$ samples were unlabeled in 10-fold cross-validation), our proposed PURF-GI still achieved a very close performance to that of RF trained using fully labeled data.

Experiments on Tree Selection. To check whether formulas (5) and (6) are capable of selecting best sub-tree among nine, we conducted a group of experiments based on three real-world UCI datasets with $ratio(UNL) = 80\%$ and $n_{tree} = 1$. For each sub-tree ($PosLevel$ from 0.1 to 0.9), we built and tested PURF-GI on 10 folds. The average Accuracy, F1 and AUC at certain $PosLevel$ are reported in Fig. 2. For each fold at a certain $PosLevel$, we recorded the real selected $PosLevel$ by formulas (5) and (6) and calculated the average values. Then we obtained nine average values of selected $PosLevel$, of which the ranges are shown in the dark areas of Fig. 2. It is clear that the dark areas are all around the $PosLevel$s with best performance, which shows the effectiveness of formulas (5) and (6) for selecting the best sub-tree.

Time and Space Complexity Analysis. In this section, we analysed the time and space complexity of PURF-GI and RF. All the experiments in this section were conducted with the Quad cores and 10 jobs. Fig. 3 shows the time consumed by PURF-GI and RF to generate models on the three real-world datasets with different n_{tree} and $ratio(UNL)$, respectively. It is obvious that, both PURF-GI and RF have linear time complexity. The reason that PURF-GI consumed more time than RF is that in each tree in the forest of PURF-GI, there are a total of nine sub-trees to generate. We also listed the average numbers of the inner nodes and leaves of single tree in both PURF-GI and

[6] `https://pythonhosted.org/joblib/generated/joblib.Parallel.html`

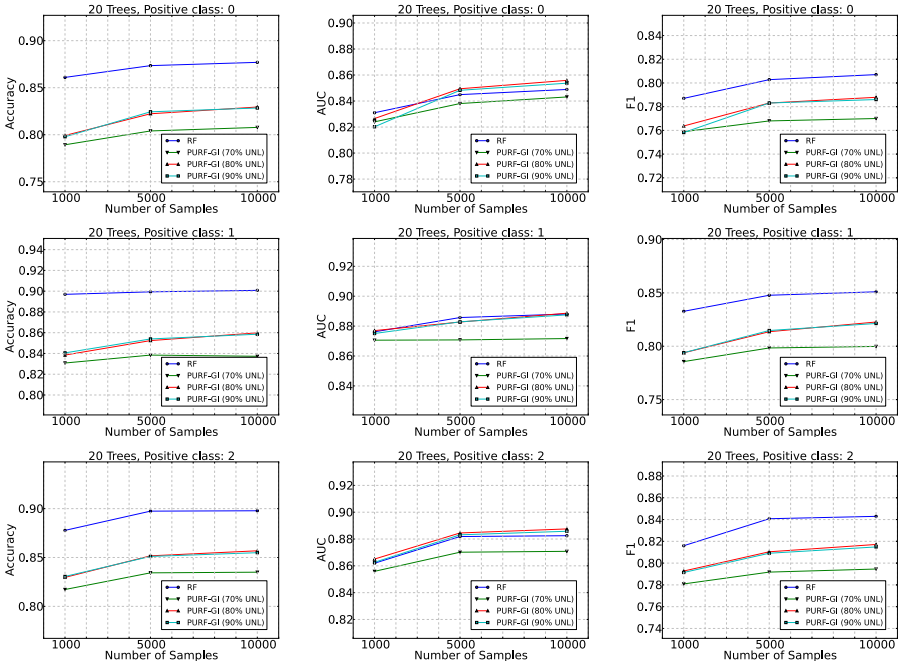

Fig. 1. Experimental results of n_{sample} with different positive classes for PURF-GI

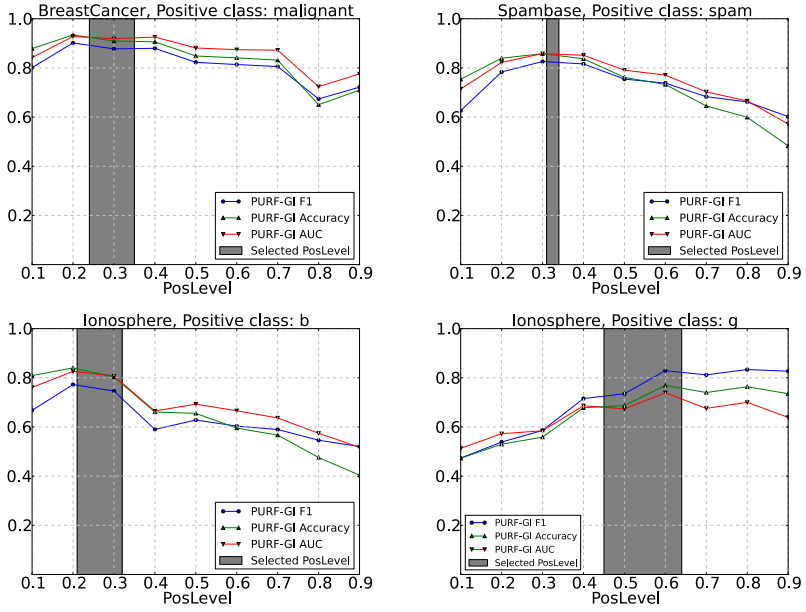

Fig. 2. Experimental results of tree selection on three UCI datasets for PURF-GI

RF in Table 2. All the results were generated based on 10-fold cross-validation of the three UCI datasets with $n_{tree} = 20$ and $ratio(UNL) = 70\%$, 80% and 90%. Since Gini index generates binary splits, $\#inner\,nodes = \#leaves - 1$.

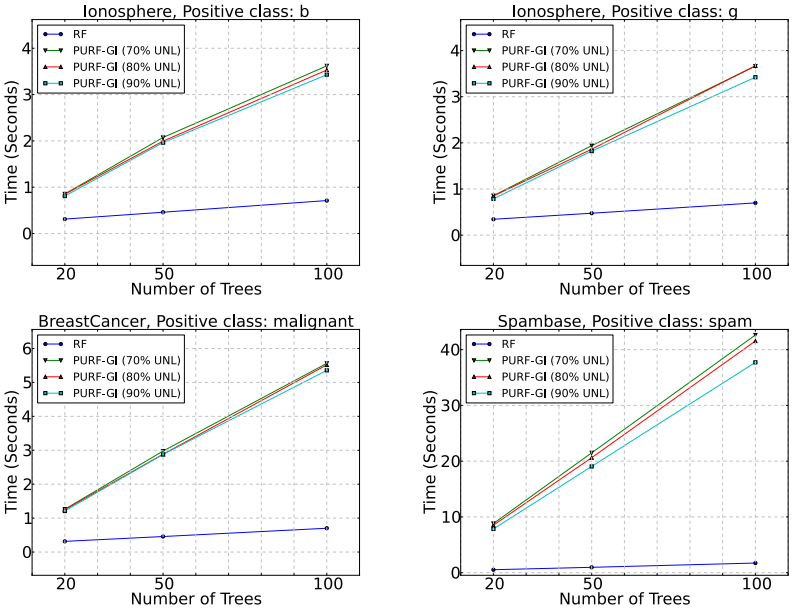

Fig. 3. Time consumed in real-world datasets with 10 jobs and Quad cores for PURF-GI

Table 2. The number of inner nodes and leaves of single tree in PURF-GI and RF

$ratio(UNL)$	**Breast cancer**		**Spambase**		**Ionosphere_b**		**Ionosphere_g**	
	#inner nodes	#leaves	#inner nodes	#leaves	#inner nodes	#leaves	#inner nodes	#leaves
Fully Labeled	25.50	26.50	260.48	261.48	21.37	22.37	21.64	22.64
70%	20.48	21.48	515.99	516.99	33.65	34.65	13.12	14.12
80%	11.10	12.10	126.80	127.80	19.30	20.30	9.38	10.38
90%	5.16	6.16	65.46	66.46	10.02	11.02	6.42	7.42

5.3 Parallel Computing Analysis of PURF-GI

Since PURF is a parallel computing framework, in this section, we went on to examine the effect of two important parameters on the time complexity in the parallel computing setup, which are the numbers of CPU cores and jobs. First, we fixed the number of CPU cores at 4. Then we calculated the consumed time based on the three UCI datasets with $ratio(UNL) = 70\%$, 80% and 90%, $n_{tree} = 50$. Fig. 4 shows the time consumed with the increase of number of jobs (from 1 to 10). As expected, the more jobs PURF-GI had, the less time PURF-GI consumed. For those runs with more than 4 jobs allocated, the time to build the forest with the same amount of trees is almost the same. This is because we used 4 CPU cores and at any point only 4 jobs could be concurrently processed.

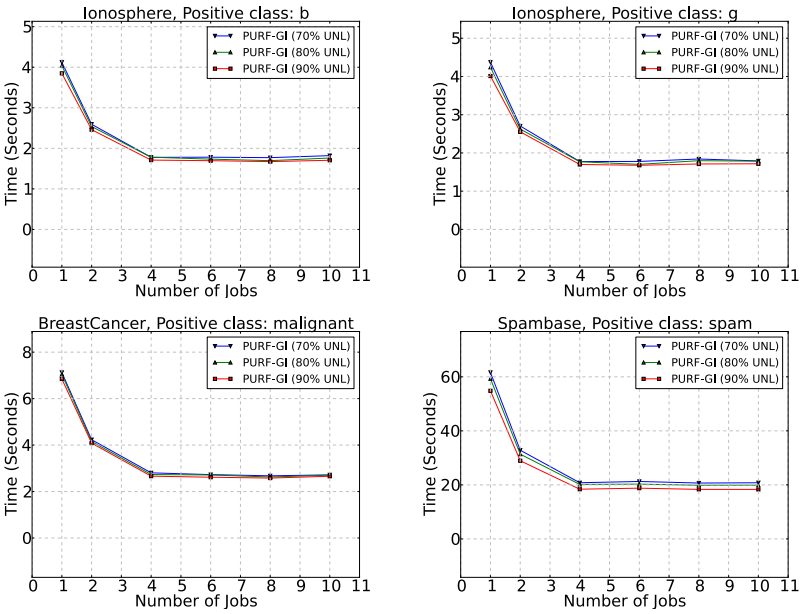

Fig. 4. Time consumed in real-world datasets with different numbers of jobs for PURF-GI

Table 3. Time (second) consumed with different numbers of CPU cores for PURF-GI

#CPU cores	$ratio(UNL)$	Spambase	Breast cancer	Ionosphere_b	Ionosphere_g
Core Duo	70%	59.447	7.557	4.647	4.765
	80%	58.169	7.493	4.535	4.737
	90%	53.652	7.349	4.365	4.567
Quad	70%	21.493	2.830	1.840	1.868
	80%	20.591	2.853	1.794	1.866
	90%	19.085	2.750	1.740	1.773

Then, we fixed the number of jobs at 10 and changed the number of CPU cores from 2 to 4. Then we run PURF-GI on the UCI datasets with $n_{tree} = 50$. The results are listed in Table 3. The first column is the number of CPU cores. The $ratio(UNL)$ is shown in the second column. Time consumed is shown for each dataset (for the ionosphere data, there were two positive classes: b and g.) in columns 3 to 6. These results suggest that PURF can be appropriately applied in the parallel computing scenario.

5.4 PURF-GI/PURF-IG *versus* RF on Real-World Datasets

To illustrate the scalability of PURF to real-world applications, in this section, we performed several experiments using UCI datasets to compare the performance of PURF-GI/PURF-IG to RF. We actually built up separate 10-fold cross-validation sets for the performance comparison of PURF-GI/PURG-IG with RF for each dataset, respectively. The $ratio(UNL)$ ranged from 70% to 90% and $n_{tree} = 20, 50$ and 100.

For the breast cancer dataset, we set 'malignant' as the positive class. Fig. 5 and Fig. 6 show the performance of PURF-GI and PURF-IG on this dataset when $ratio(UNL) = 70\%$, 80% and 90% in terms of Accuracy, AUC and F1, respectively. It can be seen that, even when $ratio(UNL) = 90\%$, the performance of PURF-GI and PURF-IG was still comparative to that of RF. For example, the differences of Accuracy, AUC and F1 between PURF-GI and RF when $ratio(UNL) = 90\%$ and $n_{tree} = 20$ were only 1.2%, 1.0% and 1.8%, respectively. It is also noticeable that, when $n_{tree} = 50$ and $ratio(UNL) = 80\%$, the performance of PURF-GI was even better that of RF.

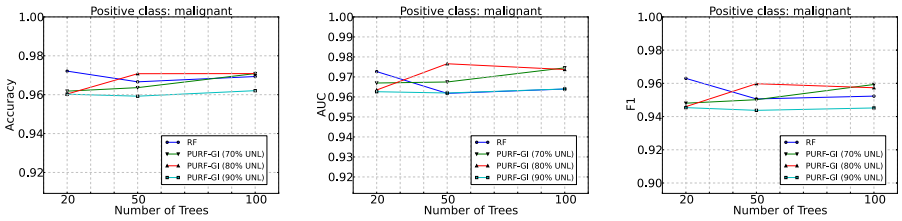

Fig. 5. Experimental results of Breast Cancer dataset for PURF-GI

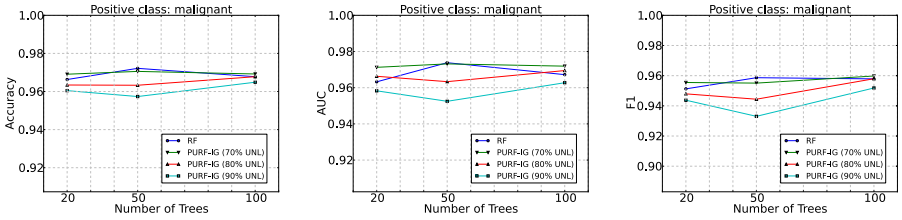

Fig. 6. Experimental results of Breast Cancer dataset for PURF-IG

For the Spambase dataset, 'spam' was set to be the positive class. Fig. 7 and Fig. 8 display the performance of PURF-GI/PURF-IG vs. RF when $ratio(UNL) = 70\%$, 80% and 90%, respectively. Similarly, both PURF-GI and PURF-IG achieved satisfactory performance when compared with RF based on this dataset. For instance, the difference of accuracy between PURF-GI and RF when $n_{tree} = 50$ and $ratio(UNL) = 80\%$ was only 3.1%.

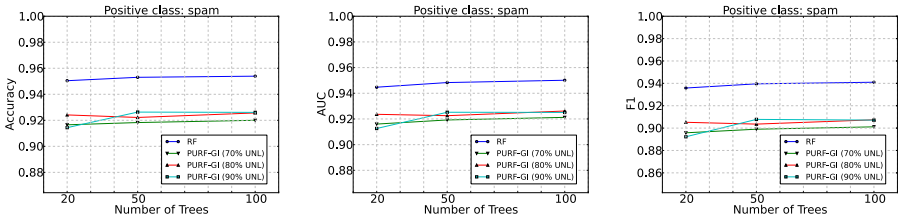

Fig. 7. Experimental results of Spambase dataset for PURF-GI

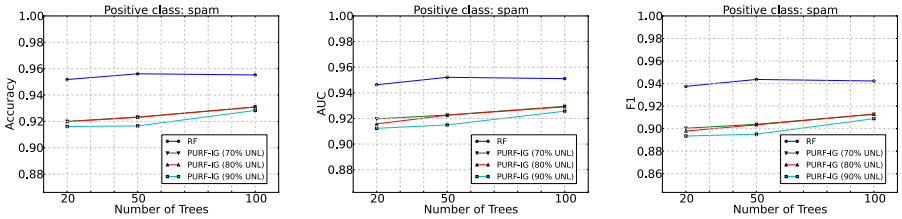

Fig. 8. Experimental results of Spambase dataset for PURF-IG

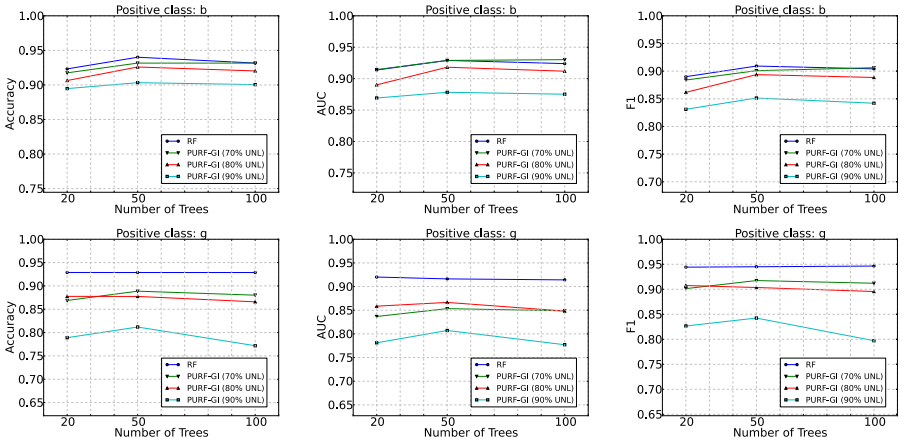

Fig. 9. Experimental results of Ionosphere dataset for PURF-GI

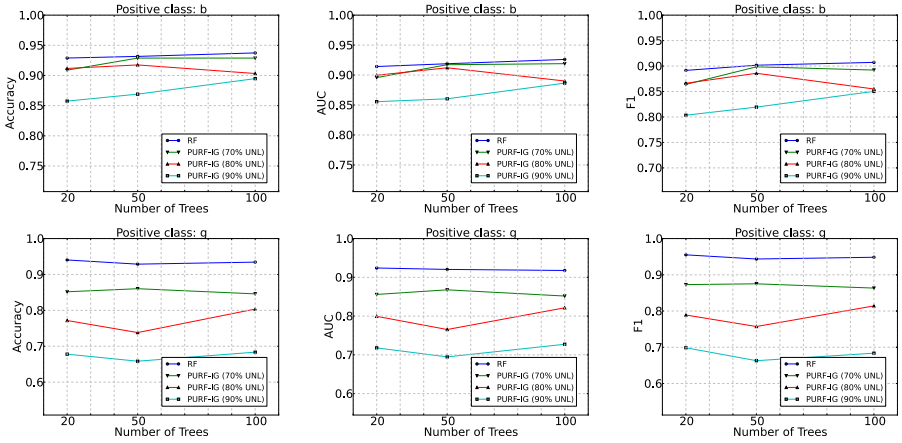

Fig. 10. Experimental results of Ionosphere dataset for PURF-IG

For the ionosphere dataset, we set each class as the positive class in turn. The performance of PURF-GI *vs.* RF and PURF-IG *vs.* RF on this dataset is shown in Fig. 9 and Fig. 10, respectively. When 'b' was set as the positive class, the Accuracy, AUC and F1 of both PURF-GI and PURF-IG were very close to those of RF with $ratio(UNL) = 70\%$ and 80%. When $ratio(UNL) = 90\%$, the performance dropped down. A possible reason for this is that the number of positive samples was very limited in this dataset - there were only a few positive samples that led to poor performance of both PURF-GI and PURF-IG. On the other hand, when 'g' was set to be the positive class, the performance was generally poor.

6 Conclusions and Future Work

In this paper, for application in the parallel computing scenario, we have developed a novel framework termed PURF (Positive unlabeled Random Forest) with PU learning techniques that enables parallel data mining to learn from only positive and unlabeled samples. Using the designed novel Gini index for PU learning as well as PU information gain [5], we generated PURF-GI and PURF-IG with the PU Gini index and PU information based on RF, respectively. Empirical assessment on real-world UCI datasets has strongly indicated that even provided with a high percentage of unlabeled data, PURF-GI and PURF-IG were able to achieve a accpetatble performance compared to RF. Our results on both synthetic and real-world dataset also indicate that PURF is powerful in learning from positive and unlabeled data. Experiments on real-world datasets showed that even with 90% unlabeled data, both PURF-GI and PURF-IG had a strong ability to distinguish positive from non-positive data. We also evaluated the performance and consumed time for calculation of PURF-GI in both parallel and non-parallel situations. As expected, time analysis demonstrated that parallel PURF could indeed save considerable time through running multiple CPU cores and jobs. It is our expectation that this new framework will be increasingly used as a powerful approach to facilitate the processing and learning of positive and unlabeled data in the future. In Gini index, for a nominal attribute A, binary partition is calculated. If we assume that $|A| = n$, the number of binary partition of A should be $2^n - 2$ (power set and empty set are excluded). Therefore in this research, we did not consider nominal attributes in our training datasets for simplicity, which is a limitation of this work. In the future, based on PURF-GI, we will investigate methods to deal with nominal attributes, thereby making it more suitable for real-world applications.

References

1. Parthasarathy, S., Zaki, M.J., Ogihara, M., Li, W.: Parallel data mining for association rules on shared-memory systems. Knowledge and Information Systems 3, 1–29 (2001)
2. Li, J., Liu, Y., Liao, W., Choudhary, A.: Parallel data mining algorithms for association rules and clustering. In: International Conference on Management of Data (2008)
3. Chen, H., Schatz, B., Ng, T., Martinez, J., Kirchhoff, A., Lin, C.: A parallel computing approach to creating engineering concept spaces for semantic retrieval: The Illinois digital library initiative project. IEEE Transactions on Pattern Analysis and Machine Intelligence 18, 771–782 (1996)

4. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
5. Letouzey, F., Denis, F., Gilleron, R.: Learning from positive and unlabeled examples. In: Arimura, H., Sharma, A.K., Jain, S. (eds.) ALT 2000. LNCS (LNAI), vol. 1968, pp. 71–83. Springer, Heidelberg (2000)
6. Calvo, B., Larranaga, P., Lozano, J.A.: Learning Bayesian classifiers from positive and unlabeled examples. Pattern Recognition Letters 28, 2375–2384 (2007)
7. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2008), pp. 213–220 (2008)
8. Yu, H.: Single-Class Classification with Mapping Convergence. Machine Learning 61, 49–69 (2005)
9. Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S.: Building text classifiers using positive and unlabeled examples. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003), pp. 179–186 (2003)
10. Fung, G.P.C., Yu, J.X., Lu, H., Yu, P.S.: Text classification without negative examples revisit. IEEE Transactions on Knowledge and Data Engineering 18, 6–20 (2006)
11. Yu, H., Han, J., Chang, K.C.C.: PEBL: web page classification without negative examples. IEEE Transactions on Knowledge and Data Engineering 16, 70–81 (2004)
12. Agrawal, R., Shafer, J.C.: Parallel mining of association rules. IEEE Transactions on Knowledge and Data Engineering 8, 962–969 (1996)
13. Han, E., Karypis, G., Kumar, V.: Scalable parallel data mining for association rules, vol. 26. ACM (1997)
14. Zaki, M.J., Parthasarathy, S., Li, W.: A localized algorithm for parallel association mining. In: Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 321–330 (1997)
15. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM SIGMOD Record 29, 1–12 (2000)
16. Zaïane, O.R., El-Hajj, M., Lu, P.: Fast parallel association rule mining without candidacy generation. In: Proceedings IEEE International Conference on Data Mining (ICDM 2001), pp. 665–668 (2001)
17. Pramudiono, I., Kitsuregawa, M.: Tree structure based parallel frequent pattern mining on PC cluster. In: Mařík, V., Štěpánková, O., Retschitzegger, W. (eds.) DEXA 2003. LNCS, vol. 2736, pp. 537–547. Springer, Heidelberg (2003)
18. Cheung, D.W., Lee, S.D., Xiao, Y.: Effect of data skewness and workload balance in parallel data mining. IEEE Transactions on Knowledge and Data Engineering 14, 498–514 (2002)
19. Kalé, L., Skeel, R., Bhandarkar, M., Brunner, R., Gursoy, A., Krawetz, N., Phillips, J., Shinozaki, A., Varadarajan, K., Schulten, K.: NAMD2: greater scalability for parallel molecular dynamics. Journal of Computational Physics 151, 283–312 (1999)
20. Sanbonmatsu, K.Y., Tung, C.S.: High performance computing in biology: multimillion atom simulations of nanoscale systems. Journal of Structural Biology 157, 470–480 (2007)
21. D'Agostino, N., Aversano, M., Chiusano, M.L.: ParPEST: a pipeline for EST data analysis based on parallel computing. BMC Bioinformatics 6, S9 (2005)
22. Li, C., Zhang, Y., Li, X.: OcVFDT: one-class very fast decision tree for one-class classification of data streams. In: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data (SensorKDD 2009), pp. 79–86 (2009)
23. Steinberg, D., Colla, P.: CART: tree-structured non-parametric data analysis. Salford Systems, San Diego (1995)