Parallelized Random Forests Learning

CSE 633 Course Project

Jie Hu, Department of CSE
Outline

• Random Forests
• Parallelized Implementation
• Experimental Results
Random Forest

- **Random forests** (RF) are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

- Given a test sample as the input, the output of the random forest is the combination of results from all individual trees.

$$P(c|v) = \sum_{t=1}^{T} P_t(c|v)$$
Sequential Random Forest Learning

- Selection of training samples for each tree
- Tree Construction

**Algorithm 1** RandomForest(examples $\langle e_1, \ldots, e_n \rangle$, features $F$, no. of trees $t$)

1. for $i = 1 \ldots t$ do
2. for $j = 1 \ldots n$ do
3. $e'_j \leftarrow e_{\text{rand}}(1,n)$
4. $T_i \leftarrow \text{RandomTree}(\langle e'_1, \ldots, e'_n \rangle, F)$
5. return $T_1, \ldots, T_t$

**Algorithm 2** RandomTree(examples $\langle e_1, \ldots, e_n \rangle$, features $F$)

1. if TerminalNode($\langle e_1, \ldots, e_n \rangle$) then
2. $T \leftarrow \text{MakeLeaf}(\langle e_1, \ldots, e_n \rangle)$
3. else
4. $K \leftarrow \text{RandomSplitFunctionPool}(\langle e_1, \ldots, e_n \rangle, F)$
5. $k \leftarrow \text{BestSplit}(\langle e_1, \ldots, e_n \rangle, K)$
6. $\{El, Er\} \leftarrow \text{Distribute}(\langle e_1, \ldots, e_n \rangle, k)$
7. $Tl \leftarrow \text{RandomTree}(El, F)$
8. $Tr \leftarrow \text{RandomTree}(Er, F)$
9. $T \leftarrow \{k, Tl, Tr\}$
10. return $T$
Parallelized Implementation

• Why parallelize?
  - If we train the random forest sequentially, the time complexity is $O(t|K|n\log n)$.
    t: number of trees  K: number of chosen features  n: size of training set
  - Increasing the number of trees can improve the accuracy of the classifier, but elongate the training time as well.
  - Training subsets of trees in the separate processor can largely decrease the training time.
Parallelized Random Forest Learning

Master

Load training data

Randomly selected training data with replacement

Worker1

Worker2

Workerm

Learnt sub-forest

Constructing the whole forest

Sub-forest learning

Bootstrap aggregating

Master

Training process

Load test data

Test process

Master

Test result
Parallelized Random Forest Learning

Algorithm 1 RandomForest(examples $<e_1,\ldots,e_n>$, features $F$), no. of trees $t$, no. of cores $m$

1. for $k=1 \ldots m$ do
2. for $i=1 \ldots t/m$ do
3. for $j=1 \ldots n$ do
4. $e'_j \leftarrow e_{\text{rand}}(1,n)$
5. $T_{(k-1)*t/m+i} \leftarrow \text{RandomTree}(<e_1,\ldots,e_n>, F)$
6. return $T_1, \ldots, T_t$

Algorithm 2 RandomTree(examples $<e_1,\ldots,e_n>$, features $F$)

1. if terminalNode($<e_1,\ldots,e_n>$) then
2. $T \leftarrow \text{MakeLeaf}(<e_1,\ldots,e_n>)$
3. else
4. $K \leftarrow \text{RandomSplitFunctionPool}(<e_1,\ldots,e_n>, F)$
5. $k \leftarrow \text{BestSplit}(<e_1,\ldots,e_n>, K)$
6. $\{El,Er\} \leftarrow \text{Distribute}(<e_1,\ldots,e_n>, k)$
7. $Tl \leftarrow \text{RandomTree}(El, F)$
8. $Tr \leftarrow \text{RandomTree}(Er,F)$
9. $T \leftarrow \{k, Tl, Tr\}$
10. return $T$
Experiments

• I implement the random forest algorithm using MPI in C
• The random forest is learnt on gene expression data which contains 3 categories and 1363 samples.
Experimental Results

• Accuracy
Experimental Results (Contd.)

• Time

![Graph showing time vs. number of cores with different nTree values]
Experimental Results (Contd.)

- Speed up

![Graph showing Speed up vs Number of cores for different nTree values]
### Experimental Results (Contd.)

- Running time (ms)

<table>
<thead>
<tr>
<th>n_trees n_nodexn_processors</th>
<th>120</th>
<th>180</th>
<th>240</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x2</td>
<td>3050</td>
<td>4410</td>
<td>5780</td>
<td>7150</td>
</tr>
<tr>
<td>2x2</td>
<td>1430</td>
<td>2010</td>
<td>2520</td>
<td>3060</td>
</tr>
<tr>
<td>4x2</td>
<td>870</td>
<td>1140</td>
<td>1410</td>
<td>1690</td>
</tr>
<tr>
<td>8x2</td>
<td>690</td>
<td>870</td>
<td>1040</td>
<td>1220</td>
</tr>
<tr>
<td>2x8</td>
<td>390</td>
<td>500</td>
<td>600</td>
<td>720</td>
</tr>
<tr>
<td>12x2</td>
<td>540</td>
<td>670</td>
<td>740</td>
<td>850</td>
</tr>
<tr>
<td>2x12</td>
<td>220</td>
<td>290</td>
<td>360</td>
<td>390</td>
</tr>
</tbody>
</table>
Conclusion and Future work

• Random forest learning is implemented using C in MPI.

• By using parallel methods, we can improves the accuracy of the classification using less time.

• We can apply this parallel methods on larger dataset and try to parallelize the construction for each decision tree.
Thank you. Q & A