Final Part 4---By Qi Yao(qy508)

Part A:

Boosting algorithm usually has three different algorithms like **Adaboost**、**GBDT**、**XGBOOST**. Here I take Adaboost algorithm as the example for parallel processing boosting algorithm. **Adaboost** is an ensemble learning method which iteratively includes a strong classifier from a series of weak classifiers. During each iteration, it employs a simple learning algorithm to get a single learner for the iteration. While the final ensemble classifier is a weighted linear combination of those weak classifiers which we have to adapt the classifier weight votes. And those weights correspond to the correctness of the classifiers, i e, if a classifier with lower error ratio gets higher weight in the final strong classifier.[1]

And if we propose two or more workers to construct the boosting classifiers. While each of the worker can only access to specified subset of the training data. If we just separate the dataset, because of the data distribution, the result got from each iteration may differs much from each other. So we can modify the order of classifier according to the weight get merge the classifier with similar weight together to get a good a result. We can express the algorithm in this way:

$D^{P}_{n^{P}} = \{(x^{p}_{1}, y^{p}_{1}), (x^{p}_{2}, y^{p}_{2}), (x^{p}_{3}, y^{p}_{3})..., (x^{p}_{n^{p}}, y^{p}_{n^{p}})\}$ is the data set for p th worker where $p \in \{1, 2...., M\}$ and n^{p} is the number of data points in the p th worker. The workers compute the ensemble classifier H^{p} by computing all the T times iterations of standard Adaboost algorithm on their respective data sets. H^{p} can be defined as below:

$\{(h^{P(1)}, \alpha^{P(1)}), (h^{P(2)}, \alpha^{P(2)}), (h^{P(3)}, \alpha^{P(3)})...., (h^{P(T)}, \alpha^{P(T)})\}$ where $h^{P(T)}$ is the weak classifier of p th worker at t th iteration. $\alpha^{P(T)}$ is the corresponding weight of the weak classifier. Then we can further choose to process the boost in the following ways. And we can reorder the weak classifiers with its corresponding weight in each machine and we can merge the classifiers and the weight for each iteration. And finally we can get the final strong classifier with the weak classifier and its distribution weight.

And the algorithm can express the parallel algorithm with the following pseudo code:

Input:separate the data into M subsets to give to M workers.
Iteration times:T
For p= 1 to M do:
H^{p} ***=Adaboost(D,T)***
Take H^{p} ***=the weak classifiers in the*** H^{p} ***sorted by the correctness***
End For
For t= 1 to T do

$$h^{t} = Merge(h^{1,t}, ..., h^{M,t})$$

$$\alpha^{t} = \frac{1}{M}\sum_{1}^{M}\alpha^{P,t}$$

End For
Return H=Sum($\alpha^{t}h^{t}$)

Note that the number of rounds for the final classifier is same as the number of iterations of the workers' internal Adaboost. However, the t th round of the final classifier doesn't necessarily merge the t th iteration results of the workers. **The intuition of the sorting the workers' weak classifier with respect to their weight is to combine with the weak classifiers with the similar correctness in the same sorted level.** And this will guarantee the similar classifier to be merged during each boosting iteration. And this stimulate the standard boosting method.
Also You can find that h^t is a ternary classifier. It should be noted that the ternary classifier provides the algorithm the ability of using any number of available working nodes in the distributed setting. In the pseudo code, the weights of the corresponding classifiers are averaged to get the weight of the ternary classifiers which are generated in T rounds. This algorithm returns the sum of the combination of classifiers with their corresponding weighted distributed. The strategy of distributing the data and computations in the worker nodes and we can ensure the nodes independent of each other and let the boost algorithm to be processed in parallel.

Reference:
[1].https://en.wikipedia.org/wiki/AdaBoost

Part B:

Advice from Student's advice:

1. I think you can include more topics like about NonSQL and Hadoop. And I think you can choose some courses to help students to understand the basic principles behind hadoop and Spark. Therefore, I think this will help the further student to get used to the new tools much more smoothly. Besides this, I think you can also give some exercise about new language like Scala. This will help a lot to write the Spark program.

2. About the course examples, I think you can ask for some ideas from students at the beginning of the semester. And choose some topics which they are interested. E g. do a survey at first. From the other side, I think you can also do some courses about R in details. Because I think data visualization is also very important for big data course.

3. For my own experience I had this semester, I took much time to learning so basic stuff by myself with many learning material. So I think you can prepare some instructions from blog or something else.

4. Finally, I have a little idea that you can let them do a little spark program from the beginning. Provide them with a base of spark. Meanwhile, I believe most students don't pay much attention on this course except the midterm or the final exam. So I think you can add them some little assignments in the process and this may "push" them in some way and help them learn something at least.

In the end, thanks for this semester's learning and I hope I did a great job. Hope best for your future course and career. :)