CSYE7374 HW2

Instructor: Dr. Handan Liu

2020-02-29

The following parts are required to be written in python file (not in Jupyter) and run in parallel mode on the Discovery Cluster.

Part I: 30 points

1. Use Pool.apply() to get the row wise common items in list_a and list_b. And print the result.
   list_a = [[1, 2, 3], [5, 6, 7, 8], [10, 11, 12], [20, 21]]
   list_b = [[2, 3, 4, 5], [6, 9, 10], [11, 12, 13, 14], [21, 24, 25]]

2. Use Pool.map() to run the following python scripts in parallel. And print the result.
   Script names: 'script1.py', 'script2.py', 'script3.py'
   Note: you can put any content in the three scripts.

Part II: 30 points

Parallel tuning C parameter in SVM:
Implement a naive parallel k-fold cross-validation algorithm in Python for tuning the "C" parameter in Support Vector Machines. The serial code "k-fold-serial.py" is given here. The dataset is "optdigits.txt".
Please parallelize the serial code by using Pool and Process of **multiprocessing**. You can put the 3 modes in one file; or in separately 3 files.
And run the serial mode and two parallelism modes on the processors of 2, 4, 8 to get the elapsed time for a plot of two types of parallelism (Pool and Process) (in one figure) showing the elapsed time (y-axis) as the number of processors (x-axis) is varied, including 1 processor with the serial code.

Part III: 40 points

Tuning Learning Rate and the Number of Trees in XGBoost:
When creating gradient boosting models with XGBoost using the scikit-learn wrapper, the **learning_rate** parameter can be set to control the weighting of new trees added to the model. In this part, you are required to explore the relationship of the learning rate and the number of decision trees by evaluating a grid of parameter pairs. The number of decision trees will be varied from 100 to 500 and the learning rate varied on a log10 scale from 0.0001 to 0.1.

Please use the grid search capability in scikit-learn to evaluate the effect on logarithmic loss of training a gradient boosting model with different learning rate values.

```
n_estimators = [100, 200, 300, 400, 500]
learning_rate = [0.0001, 0.001, 0.01, 0.1]
```

You are going to work on the "train.csv" and the following estimators/utilities in scikit-learn are required to use in this part: XGBoost Classifier, Grid Search CV with (Stratified)KFold (K=10) and other necessary tools.

Please run your python script on the cluster. On any compute node, set the parallel jobs for GridSearchCV on 1, 2, 4, 8 and 16 processors to get the elapsed time for plotting the elapsed time (y-axis) as the number of processors (x-axis) is varied.

After evaluation, please print the following outputs with formats:
1) print the output of the "best score" and "best parameters" of the gridsearchcv attributes;
2) print the means (mean_test_score) with the best parameters (best_params_).
And give a plot of each learning rate as a series showing log loss performance (y-axis) as the number of trees (x-axis) is varied.