CMPE 300 - Fall 2020
MPI Programming Project
Due: 20.01.2021 23:59

1 Introduction

In this project, you are going to experience parallel programming with C/C++/Python using MPI library. You will implement a parallel algorithm for feature selection using Relief.

2 Relief

In supervised machine learning, you have labeled input data which consist of feature values and class label for every instance. Class label is a single target field whereas there may be thousands of features. The aim of machine learning is to predict the class label by learning a pattern of features. Irrelevant features may decrease the accuracy of your decision models. Feature selection is the process where you decide to utilize a subset of the whole feature set which contributes most to your prediction variable. It is a pre-processing step before the main decision process.

Some feature selection methods assumes independence of features and judges only according to the label variable. However, in real life, features can depend on each other or act in correlation with each other. Relief is a feature weighting algorithm which estimates the quality of attributes considering the strong dependencies between them. For feature selection, we can utilize the Relief weights and select a subset of the features with highest weights.

Below is a pseudocode for Relief algorithm:

```
Input: dataset (for each training instance a vector of feature values and the class value)
        n ← number of training instances
        a ← number of features (not including class variable)
        m ← number of iterations

initialize all feature weights W[A]=0.0
for i=1 to m do
        randomly select a target instance Ri
        find a nearest hit H and nearest miss M (instances)
        for A=0 to a-1 do
                W[A] = W[A] - diff(A,Ri,H)/m + diff(A,Ri,M)/m
        end for
end for

Output: the vector W of feature scores that estimate the quality of features
```

In this pseudocode, the helper function diff is defined as:

$$\text{diff(A,I}_1, I_2) = \frac{|value(I_1,A) - value(I_2,A)|}{max(A) - min(A)}$$

Relief calculates a weight vector W. It starts with a zero vector and updates W thru iterations. In every iteration, Relief selects a target instance. It calculates one nearest same class and one

nearest other class instance to the target instance and updates W by these instances. By this way, weights resolve the differentiation between given classes. The maximum and minimum values of A in diff function are determined over the entire set of instances. It ensures that weight updates fall between 0 and 1.

After m iterations, features with the highest weights are selected as a relevant subset for the problem. This elimination both helps to get rid of confusing results from irrelevant features and increases performance of the machine learning system as less features are used in calculations.

You can read in more detail about Relief and its variations from "Relief-based feature selection: Introduction and review" paper by Urbanowicz et al, 2018. You can check a mini example from Medium.com.

3 Problem Description

New datasets are getting bigger and bigger with petabytes of instances. Therefore, there is a growing need for a parallel processing point of view for handling these datasets. In this project, you will implement a parallel feature selection using Relief.

There will be p processors in your system, 1 master and p-1 slave. All file I/O is handled with the master processor. Slave processors can only print to console.

The input file is tab separated and will be as follows:

```
P
N A M T
... N lines of input data
```

P is the total number of processors (remember 1 master, P-1 slave). Processor ids start from 0, P0 is the master. N is the number of instances. A is the number of features. Feature ids start from 0 and feature values are all numeric with decimal places up to 4. For each instance, after A features, there is a single class variable. We will have 0 or 1 as target class value. M is the iteration count for weight updates. T is the resulting number of features. Top T features will be selected from Relief algorithm. The rest is N lines of input data.

The master reads the input file and stores input data. It divides the input data into equal length partitions and sends each partition to a slave. Data lines are used in order in this process (will be clear with the example). The master also sends M and T to all slaves. Each slave runs Relief on its partition for M iterations and selects top T features. For simplicity, you will not use randomly selected target instance in iterations. Instead, the instances will be used in sequential order as target instance among iterations (will be clear with the example). For distance formula, you will use Manhattan distance. After iterations, slaves print the ids of the resulting T features in ascending order of ids. They send the list of top features to the master. The master unites all lists and eliminates duplicates. It prints the resulting set in ascending order of ids. Each processor prints its output as a single line. Console prints are space separated.

Manhattan distance formula:

$$\text{ManhattanDistance}([x_1, y_1, z_1], [x_2, y_2, z_2]) = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

We will provide you small-medium-large test files for your development tests. Grading will be done with unseen test files.

4 Example

An example input file is as follows:

```
3
10 4 2 2
6.0 7.0 0.0 7.0 0
16.57 0.83 19.90 13.53 1
0.0 0.0 9.0 5.0 0
11.07 0.44 18.24 15.52 1
5.0 5.0 5.0 7.0 0
16.55 0.25 10.68 17.12 1
7.0 0.0 1.0 8.0 0
17.44 0.01 18.55 17.52 1
5.0 3.0 4.0 5.0 0
16.80 0.72 10.55 13.62 1
```

There will be 3 processors for this example, 1 master (P0) and 2 slaves (P1, P2). The input data has 10 instances with 4 features and a class value each, feature0-feature3. The class label, which is 0 or 1, comes at the end of each data line. For Relief algorithm, there will be 2 iterations and at the end each slave processor selects top 2 features.

P0 reads the input lines and sends #input_lines / #slave_processors = 10/2 = 5 instances to each processor. As P0 partitions the input lines in order, the first 5 lines will be assigned to P1 and the last 5 lines will be assigned to P1. P0 also sends M=2 and T=2 information to the slaves.

P1 and P2 runs the Relief algorithm. They first define and initialize the weight array (W) of size feature count, 4 in this example. Consider P1 for the first iteration, it selects 1st instance (6.0 7.0 0.0 7.0 0) as the target instance. For the second iteration, P1 selects 2nd instance (16.57 11.83 19.90 13.53 1) as the target instance. In each iteration, the distance between the target instance and all other instances in that slave processor is calculated by using Manhattan distance. The nearest hit (nearest instance with the same class label) and the nearest miss (nearest instance with the other class label) are selected and used in weight updates.

When slaves complete their iterations, they select two features with the highest weights in W array (because T=2). They print the result to console.
P1 prints:

```
Slave P1 : 2 3
```

P2 prints:

```
Slave P2 : 0 2
```

Slaves send their results to the master. The master unites them and prints the united list:

```
Master P0 : 0 2 3
```

Each processor prints its output as a single line. Console prints are space separated. The whole print may be as following for this example (of course ␣will not be visible):

```
Slave␣P1␣:␣2␣3
Slave␣P2␣:␣0␣2
Master␣P0␣:␣0␣2␣3
```

5 Submission Details

- The deadline for submitting the project is 20.01.2021 - 23:59. The deadline is strict.
- This is an individual project. Your code should be original. Any similarity between submitted projects or to a source from the web will be accepted as cheating.
- Your code should be sufficiently commented and indented.
- You should write a header comment in the source code file and list the following information.

```
Student Name: Ali Veli
Student Number: 2008123456
Compile Status: Compiling/Not Compiling
Program Status: Working/Not Working
Notes: Anything you want to say about your code that will be helpful in the grading process.
```

- You must prepare a document about the project, which is an important part of the project. You should talk about your approach in solving the problem. Follow the guidelines given in the "Programming Project Documentation" link on https://www.cmpe.boun.edu.tr/ gungort/informationstudents.htm.
- Submit your project and document as a compressed archive (zip) to Moodle. Your archive file should be named in the following format: "StudentId.zip", i.e. 2008123456.zip. You don't need to submit any hard copies.
- If you have any further questions, send an e-mail to the course assistant: ozlem.simsek@boun.edu.tr