CMPE 300 – Fall 2020
MPI Programming Project

2018400285
Bengisu Özaydın

27th January, 2020

INTRODUCTION

This project is written in C++ using MPI library.

In supervised machine learning, the aim is to predict class labels of instances by learning from their feature values. The number of features can be numerous. Not all features contribute equally to the prediction, also too many features cause slow learning, noisy data etc. Therefore, feature selection is used as a pre-processing step to decide on the most important features.

In this project parallel programming using MPI is implemented using Relief algorithm, which is used in weighing features considering the dependencies between them and selecting the ones that contribute to the prediction the most.

REQUIRED ENVIRONMENT

In order to run this program, you must own a machine that runs C++. An MPI library such as OpenMPI must be installed.

COMPILATION

After opening a terminal and changing directory to the path of the project, compile the program using:

```
mpic++ -o cmpe300_mpi_2018400285 ./cmpe300_mpi_2018400285.cpp
```

RUNNING

Run the program using:

```
mpirun --oversubscribe -np <num_of_processors>
./cmpe300_mpi_2018400285 <inputfile>
```

INPUT

Should follow the format:

P
N A M T
⋯ N lines of input data

Where:
P: total number of processors (1 being the master, P-1 being the slaves)
N: number of instances
A: number of features
M: iteration count in Relief algorithm
T: resulting number of features after selection

! Please make sure that N is divisible by P-1, since master divides the input equally among processors.

OUTPUT

One line for each slave that is as follows:

Slave P<rank of processor> : <numbers representing selected features by this slave in ascending order>

Followed by one line by master that is as follows:

Master P0 : <combination of all features selected by all slaves without duplicates in ascending order>

IMPLEMENTATION DETAILS

The implementation consists of two parts: the master and the slaves.

MPI_Send and MPI_Recv functions are used to communicate between master and slaves.

Master:

- Reads the input file
- Sends each slave N / (P-1) instances
- Receives the selected feature indices in ascending order from each slave
- Merges these in a set to remove duplicates
- Prints all selected feature indices in ascending order to the console

Slaves:

- Receive instances sent by the master
- Initialize an array of zeroes that represent weights of features.
- Run Relief algorithm (Relief algorithm depends on how well a certain feature can distinguish between close instances)
  - For the M iteration count, select an instance (for this project, instances are chosen sequentially using mod in case the numbers do not fit)
  - Find nearest hit (same class) and nearest miss (different class): Manhattan Distance is used here.

    ```
    double manhattan_distance(double features_1[], double
    features_2[], int size){
      double distance = 0;
      for(int i = 0; i < size-1; i++){
        distance += fabs(features_1[i] - features_2[i]);
      }
      return distance;
    }
    ```
  - Update weight vector using a helper diff function.
    - If the result of this diff function between selected instance and nearest hit is large, then attribute is not very successful in contributing to the prediction.
    - If the result of this diff function between selected instance and nearest miss is large, then attribute is desirable since it is successful in contributing to the prediction by differentiating opposite classes.
- Sort weight vector
- Choose T features with the greatest weight
- Print all selected feature indices in ascending value to the console
- Send selected feature indices to master in ascending order

DIFFICULTIES ENCOUNTERED

If you get error no 2 in large input files, run the following code in terminal:

```
export OMPI_MCA_btl=self,tcp
```