Sevde Sarıkaya

INTRODUCTION:

Relief Algorithm is an algorithm used for Feature Selection in Supervised Learning. In supervised machine learning, we have labeled input data with feature values and class label for every record. Our model predicts the class label of an instance thanks to these inputs. However, some irrelevant features may affect the accuracy of our model. That's why we need feature selection. Thanks to Relief, we can select features by considering the dependencies between features. That helps us with handling noise in our data or avoiding overfitting in our machine learning model. Moreover, after feature selection our total data gets smaller which fastens the training of our model.

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
```

There are different ways to find difference but we use this one:

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

PROGRAM EXECUTION:

I used C++ with MPI library in this project. To be able run my project, you need to have MPI in your computer. You can simply install it in Linux by “sudo apt install openmpi-bin” command.

My openmpi version: mpirun (Open MPI) 2.1.1

GCC version : gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0

compile code with mpic++ as below:

```
> mpic++ -o relief ./relief.cpp
```

run code as below with -np parameter:

```
> mpirun --oversubscribe -np <P> relief<inputfile>
```

After termination, it will give you features selected by each slave processor and the master in the order.

INPUT:

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

Here, P is the total number of processors.

N is the number of instances given.

A is the number of features. At the end of every instance there is a class variable.

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 instances.

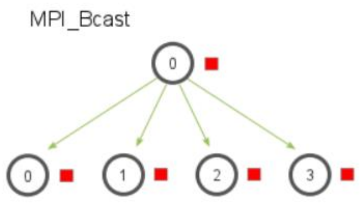
OUTPUT:

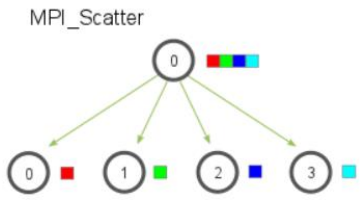
Prints the selected feature by the each processor.

For example:

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

PROGRAM STRUCTURE:

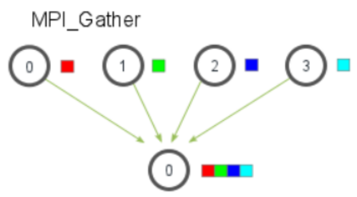




Main consists of **3 parts**.
First part is reading input with ifstream. Then, broadcasting the paratemers via MPI_Bcast to the slave processors from master. After reading all instances, master distributes the data equally to the processors via MPI_Scatter function. This process done by the master.

Second part is done by the slave processors. They take the instances, apply Relief algorithm and selects features.

Third part is done by the master. It gathers the selected features by the slave processors via MPI_Gather and prints them out in the order of feature ID.



RELIEF ALGORITHM:

- To calculate the differences by the given method we need to find the maximum and minimum values of the instances that belong to the current processor. This process is done by **findMaxnMinFeatures** function and it returns maximum of features and minimum of features vectors as a pair.

- To calculate the distance between instances, we use Manhattan Distance (**manhattanDistance**). It takes 2 instances, calculates the distance and returns the value.

- **diff** function basically calculates the difference.

Relief function uses the helper functions above.
First of all, it selects an instance sequentially. Then, finds its nearest hit and nearest miss.

Nearest Hit: Nearest instance with the same class number. (Distance is calculated by Manhattan)
Nearest Miss: Nearest instance with the other class number.

After finding the nearest hit and nearest miss instances, it updates the weights of the features by the given function above.

An iteration is done and it continues with the next iteration.

Examples:

Input:

```
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
```

The number of processors: 3
The number of instances: 10
The number of features: 4
Iteration count: 2
Top 2 features should be chosen.

Output:

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

Improvements & Extensions:

Some randomness would make our selection better. For example, in some natural data an event may be occurred in some of the instances. However, taking instances subsequently while applying Relief may result in not being able to recognizing the important events.

Difficulties:

The project was fun. I haven’t encountered with difficulties. However, learning some MPI from scratch might be a little difficult. It took some time.

Conclusion:

In this project, we apply Relief Algorithm with Master & Slave processors. Relief Algorithm is powerful algorithm for feature selection. We speed it up by parallel programming.