COMP7107-Management of Complex Data Types

Assignment 1：Band join and similarity computation

Ping Yifeng 3036196594
woodp620@connect.hku.hk

1. Info and Setup

This assignment is written in C++ 17. CMake 3.25 or above is needed to build this project. You can find the GitHub page for this assignment through the following link: https://github.com/frankpyf/COMP7107_management_of_complex_data_types.

After running the setup file, there should be a build directory that includes the executables. To run the executable, you need to put all the data files into the build directory. Now we are ready to go.

2. Result

This dataset has 581012 records, and each record has 54 attributes and 1 categorical variable. Among these attributes, 10 of them are quantitative ones and 44 of them are qualitative ones. The categorical variable represents 7 forest cover types, tagged from 1 to 7.

2.1 Part 1: Band Join

To get the number of pairs, I implemented an algorithm with a time complexity of O(N2) as instructed. To optimize the algorithm a little bit, I keep track of the last record that the difference in the first column (Elevation) is equal to or smaller than k in one iteration so that I don't have to iterate through the records all over again in the next iteration. This is reasonable because the records are sorted based on the first column. C++ code is as follows:

```
int right = 0;
int nums = 0;
// we are talking of pairs, so we only need to iterate to the element before the last one
for(int left = 0; left < sorted_records.size() - 1; ++left)
{
    while((sorted_records[right].quantative_attributes[0] - sorted_records[left].quantative_attributes[0]) <= k)
    {
        if(right == (sorted_records.size() - 1))
            break;
        ++right;
    }
    --right;
    nums += (right - left);
}
return nums;
```

After testing the program with different small values of k, the results are shown in table 1.

k	0	1	2	3	4	5
qualified pairs	199138468	568355289	937077919	1327371987	1704671089	2073456300

table 1 the number of observation record pairs whose difference in the first column is at most k

2.2 Part 2: Similarity computation

1) Similarity function

I first normalize every quantitative attribute with min-max normalization. Then, I use the general approach for combining similarities because there are two types of attributes.

For each quantitative attribute, I compute its distance between each pair of records and use the distance to compute the similarity.

```
// Quantative attributes
for(int i = 0; i < 10; ++i)
{
        float distance = std::abs(f1.quantative_attributes[i] - f2.quantative_attributes[i]);
        similarity += 1.0f / (1.0f + distance);
        ++sum_delta;
}
```

For the qualitative attributes, it is similar to Jaccard method. In other words, we exclude the 0 and 0 cases.

```
// Binary attributes
for(int i = 0; i < 4; ++i)
{
        if(f1.wilderness_area[i] == 0 && f2.wilderness_area[i] == 0)
                continue;
        similarity += f1.wilderness_area[i] * f2.wilderness_area[i];
        ++sum_delta;
}
```

Finally, divide the variable similarity by sum_delta.

2) Observations

As shown in table 2, the minimum, maximum and average similarity of the random samples are smaller than the ones of the classified samples. This observation is intuitive.

	Random	Type=1	Type=2	Type=3	Type=4	Type=5	Type=6	Type=7
min	0.499	0.572	0.581	0.676	0.693	0.580	0.697	0.533
max	0.996	0.999	0.998	0.999	0.999	0.999	0.999	0.999
avg	0.684	0.770	0.824	0.801	0.810	0.73	0.811	0.736

table 2 minimum, maximum and average similarity

Another thing to notice is that, among the similarity of each cover type, Lodgepole Pine forest (cover type = 2) has the highest average similarity and Aspen forest (cover type = 5) has the lowest. Although maximum similarities are similar, there is a certain difference in the minimum and average similarity of each cover type. However, this observation is based only on my sample, not all the records. Thus, other sample may not be the case.