AI Programming Assignment #2 李明峻 0416206

Method : First，I want to specify my method to test this experiments，as traditional supervised learning，we have to separate the datasets into training data(induction part) and testing data(inference part) to accomplish all the process，first of all，we have to read all the detail about the input type and form，classify the attributes and the categories of each input data，after processing all the input data and store it，now we can start to train the datasets to build the model，and I use the Iris datasets to experiment，this datasets has 150 datas，so I separate 120 datas to be the training set and the other 30 datas be the testing data，on the other hand，I also implement the K-fold cross validation (K = 5) to average the accuracy，the method is that we apply totally different training set and testing set of all the data between each step，I think this can help the accuracy of precision.
Then next step we can implement random forest to build our training model，the method is that we have to build a lot of decision trees as our model，prepared to be tested afterwards，for 'each' testing data，it would be tested through all the decision trees that was built in the training step，such that we can gain lots of results from the model，among this model we have to vote the answer as the prediction of this testing data，and finally comparing the result with the real answer. And the process of building the decision tree is to compute the Gini's impurity of each attributes，the detail is in the spec. of this homework，so I skip describing here.

This is the process of cross validation Bellow

```
void Data_cross_validation(vector<Flower> &flower){
    // random shuffle
    srand(time(NULL));
    for(int i = 0; i < flower.size(); i ++){
        flower[i].randNum = (rand() % RAND_SIZE);
    }
    sort(flower.begin(), flower.end(), compare);

    // K-fold cross validation (K = 5)
    int trainNum = (flower.size()) * 4 / 5;
    int testNum = (flower.size()) / 5;
    vector<Flower> TrainFlower;
    vector<Flower> TestFlower;
    TrainFlower.resize(trainNum);
    TestFlower.resize(testNum);
    float final[5];

    for(int i = 0; i < 5; i++){
        if(i == 0){
            for(int j = 0; j < TrainFlower.size(); j++){
                TrainFlower[j] = flower[j];
            }
        }
        else if(i == 1){
            for(int j = 0; j < 90; j++){
                TrainFlower[j] = flower[j];
            }
            for(int j = 120; j < 150; j++){
                TrainFlower[j-30] = flower[j];
            }
        }
        else if(i == 2){
            for(int j = 0; j < 60; j++){
                TrainFlower[j] = flower[j];
            }
            for(int j = 90; j < 150; j++){
                TrainFlower[j-30] = flower[j];
            }
        }
        else if(i == 3){
            for(int j = 0; j < 30; j++){
                TrainFlower[j] = flower[j];
            }
            for(int j = 60; j < 150; j++){
                TrainFlower[j-30] = flower[j];
            }
        }
```

```
        // Build random forest
        vector<dtree> Root;
        Root.resize(FOREST_NUM);
        for(int j = 0; j < FOREST_NUM; j++){
            Root[j] = NULL;
            dtree tnode = new treenode;
            vector<Flower> train = TrainFlower;
            TrimTrainData(train);
            tnode->ClassifyFeature = ComputeIG(train);
            srand(time(NULL));
            for(int i = 0; i < train.size(); i ++){
                train[i].randNum = (rand() % RAND_SIZE);
            }
            sort(train.begin(), train.end(), compare);
            tnode->leave = false;
            Root[j] = tnode;
            BuildTree(Root[j], train);
        }

        if(i == 0){
            for(int j = 0; j < TestFlower.size(); j++){ // test data
                TestFlower[j] = flower[j+trainNum];
            }
        }
        else if(i == 1){
            for(int j = 90; j < 120; j++){
                TestFlower[j-90] = flower[j];
            }
        }
        else if(i == 2){
            for(int j = 60; j < 90; j++){
                TestFlower[j-60] = flower[j];
            }
        }
        else if(i == 3){
            for(int j = 30; j < 60; j++){
                TestFlower[j-30] = flower[j];
            }
        }
        else if(i == 4){
            for(int j = 0; j < 30; j++){
                TestFlower[j] = flower[j];
            }
        }

        final[i] = TestData(Root, TestFlower);
    }
```

Then this is the function I compute Gini's value

```
float ComputeGini(vector<Flower> flower, int thre, int type){
    float f1Pro, f2Pro, setosaPro, versicolorPro, virginicaPro, f1Gini, f2Gini, gini;
    vector<Flower> f1, f2;
    for(int i = 0; i < flower.size(); i++){
        if(type == 1){
            if(flower[i].sepal_len <= thre) f1.push_back(flower[i]);
            else f2.push_back(flower[i]);
        }
        else if(type == 2){
            if(flower[i].sepal_wid <= thre) f1.push_back(flower[i]);
            else f2.push_back(flower[i]);
        }
        else if(type == 3){
            if(flower[i].petal_len <= thre) f1.push_back(flower[i]);
            else f2.push_back(flower[i]);
        }
        else if(type == 4){
            if(flower[i].petal_wid <= thre) f1.push_back(flower[i]);
            else f2.push_back(flower[i]);
        }
    }
    f1Pro = (flower.size() == 0) ? 0 : ((float)f1.size() / (float)flower.size());
    f2Pro = (flower.size() == 0) ? 0 : ((float)f2.size() / (float)flower.size());

    int setosaCnt = 0, versicolorCnt = 0, virginicaCnt = 0;
    for(int i = 0; i < f1.size(); i++){
        if(f1[i].type == "Iris-setosa") setosaCnt++;
        else if(f1[i].type == "Iris-versicolor") versicolorCnt++;
        else if(f1[i].type == "Iris-virginica") virginicaCnt++;
    }
    setosaPro = (f1.size() == 0) ? 0 : ((float)setosaCnt / (float)f1.size());
    versicolorPro = (f1.size() == 0) ? 0 : ((float)versicolorCnt / (float)f1.size());
    virginicaPro = (f1.size() == 0) ? 0 : ((float)virginicaCnt / (float)f1.size());
    f1Gini = 1.0 - (setosaPro*setosaPro + versicolorPro*versicolorPro + virginicaPro*virginicaPro);

    setosaCnt = 0;
    versicolorCnt = 0;
    virginicaCnt = 0;
    for(int i = 0; i < f2.size(); i++){
        if(f2[i].type == "Iris-setosa") setosaCnt++;
        else if(f2[i].type == "Iris-versicolor") versicolorCnt++;
        else if(f2[i].type == "Iris-virginica") virginicaCnt++;
    }
    setosaPro = (f2.size() == 0) ? 0 : ((float)setosaCnt / (float)f2.size());
    versicolorPro = (f2.size() == 0) ? 0 : ((float)versicolorCnt / (float)f2.size());
    virginicaPro = (f2.size() == 0) ? 0 : ((float)virginicaCnt / (float)f2.size());
    f2Gini = 1.0 - (setosaPro*setosaPro + versicolorPro*versicolorPro + virginicaPro*virginicaPro);

    gini = f1Pro * f1Gini + f2Pro * f2Gini;
    return gini;
}
```

```
Classification ComputeIG(vector<Flower> &info){
    if(info.size() == 0){
        Classification cla;
        cla.feature = "serious_error";
        return cla;
    }

    Classification Ac;
    //compute sepal_len Gini's impurity
    vector<GINI> G; // gini value of all threshold
    G.resize(info.size()-1);
    sort(info.begin(), info.end(), compare1);
    for(int i = 0; i < info.size()-1; i++){
        float thre = (info[i].sepal_len + info[i+1].sepal_len) / 2;
        G[i].threshold = thre;
        G[i].gini = computeGini(info, thre, 1);
    }
    sort(G.begin(), G.end(), compareGini);
    Ac.feature = "sepal_len";
    Ac.Threshold = G[0].threshold;
    Ac.Ig = G[0].gini;

    //Compute sepal_wid Gini's impurity
    sort(info.begin(), info.end(), compare2);
    for(int i = 0; i < info.size()-1; i++){
        float thre = (info[i].sepal_wid + info[i+1].sepal_wid) / 2;
        G[i].threshold = thre;
        G[i].gini = computeGini(info, thre, 2);
    }
    sort(G.begin(), G.end(), compareGini);
    if(G[0].gini < Ac.Threshold){
        Ac.feature = "sepal_wid";
        Ac.Threshold = G[0].gini;
        Ac.Ig = G[0].gini;
    }

    //Compute petal_len gini's impurity
    sort(info.begin(), info.end(), compare3);
    for(int i = 0; i < info.size()-1; i++){
        float thre = (info[i].petal_len + info[i+1].petal_len) / 2;
        G[i].threshold = thre;
        G[i].gini = computeGini(info, thre, 3);
    }
    sort(G.begin(), G.end(), compareGini);
    if(G[0].gini < Ac.Threshold){
        Ac.feature = "petal_len";
        Ac.Threshold = G[0].gini;
        Ac.Ig = G[0].gini;
    }

    //Compute petal_wid Gini's impurity
    sort(info.begin(), info.end(), compare4);
    for(int i = 0; i < info.size()-1; i++){
        float thre = (info[i].petal_wid + info[i+1].petal_wid) / 2;
        G[i].threshold = thre;
        G[i].gini = computeGini(info, thre, 4);
    }
    sort(G.begin(), G.end(), compareGini);
    if(G[0].gini < Ac.Threshold){
        Ac.feature = "petal_wid";
        Ac.Threshold = G[0].gini;
        Ac.Ig = G[0].gini;
    }
    if(Ac.Ig == 0){
        Ac.same = true;
    }
    else Ac.same = false;
    return Ac;
}
```

After computing Gini's value，we have to choose which classification is proper for splitting the node，and record the information likes gini's value and threshold of that attribute，in order to classify the data when splitting the node.

```
void BuildTree(dtree root, vector<Flower> &f){
    dtree leftChild;
    dtree rightChild;

    if(root->ClassifyFeature.same){ // leave node
        root->left = NULL;
        root->right = NULL;
        root->leave = true;
        //cout << "leave" << endl;
        root->flowerType = f[0].type;
        return;
    }

    float thre = root->ClassifyFeature.Threshold;
    vector<Flower> f1,f2;

    if(root->ClassifyFeature.feature == "sepal_len"){
        for(int i = 0; i < f.size(); i++){
            if(f[i].sepal_len < thre) f1.push_back(f[i]);
            else f2.push_back(f[i]);
        }
    }
    else if(root->ClassifyFeature.feature == "sepal_wid"){
        for(int i = 0; i < f.size(); i++){
            if(f[i].sepal_wid < thre) f1.push_back(f[i]);
            else f2.push_back(f[i]);
        }
    }
    else if(root->ClassifyFeature.feature == "petal_len"){
        for(int i = 0; i < f.size(); i++){
            if(f[i].petal_len < thre) f1.push_back(f[i]);
            else f2.push_back(f[i]);
        }
    }
    else if(root->ClassifyFeature.feature == "petal_wid"){
        for(int i = 0; i < f.size(); i++){
            if(f[i].petal_wid < thre) f1.push_back(f[i]);
            else f2.push_back(f[i]);
        }
    }

    leftChild = new treenode; // build left subtree
    leftChild->ClassifyFeature = ComputeIG(f1);
    if(leftChild->ClassifyFeature.feature == "serious_error"){
        root->left = NULL;
        root->right = NULL;
        root->leave = true;
        root->flowerType = f[0].type;
        //cout << "sleave" << endl;
        return;
    }
    leftChild->leave = false;
    root->left = leftChild;
    BuildTree(root->left, f1);

    rightChild = new treenode; // build right subtree
    rightChild->ClassifyFeature = ComputeIG(f2);
    if(rightChild->ClassifyFeature.feature == "serious_error"){
        root->left = NULL;
        root->right = NULL;
        root->leave = true;
        root->flowerType = f[0].type;
        //cout << "sleave" << endl;
        return;
    }
    rightChild->leave = false;
    root->right = rightChild;
    BuildTree(root->right, f2);
}
```

Once the attribute and its threshold is chosen，we can build the decision tree according to these information，if the value of data is less than the threshold，then go to the left child of the node，otherwise go to the right child of the node. Follow this rule，we can build the decision tree.

Results:

This is the accuracy of 5-fold cross validation (5 results).

Random forest tree number : 5

```
[lmjun860804@linux1 AI]$ ./1
Choose datasets you want to train :
1.Iris
1
5-fold cross validation result :
0.9256666 0.7863333 0.9354545 0.7553982 0.8033333
[lmjun860804@linux1 AI]$
```

Random forest tree number : 30

```
[lmjun860804@linux1 AI]$ ./1
Choose datasets you want to train :
1.Iris
1
5-fold cross validation result :
0.6854985 0.8563245 0.9033291 0.8 0.6253333
[lmjun860804@linux1 AI]$
```

Observation:

1. Number of trees in the forest

From my perspective，I think that increase the random forest size will also increase the accuracy of results，but my experiment didn't show this phenomenon，I think the reason that cause this happened is that I choose the data subsets randomly every time the program run，so in the different time when the program execute，there must be a distinct result，maybe sometimes the training dataset was chosen in dense，so the decision tree will lose its precision in high probability，so the accuracy will also decrease，or maybe the testing dataset and training dataset varies in lots of attributes(this means two dataset are totally different classification)，this will also cause the low accuracy. The solution of this problem might be testing this program for several times that can average the peak phenomenon of the dataset. Then I believe the accuracy would be better.

2. Parameters used during tree induction, such as how many attributes to consider at each node splitting

I consider all the attributes(parameters) when finding proper classification for splitting node，because I think considering more various factors can view the model in full perspective，but this is suitable for large data amount，in this Iris dataset，the data amount is too small，so the performance isn't that good.

Experience:

In this assignment，we implement a simple supervised learning model，this concept is very important not only in artificial intelligence but in machine learning and deep learning and reinforcement learning issues，we can also apply this concepts to a lot of real world problems，after practicing each step of the algorithm，I can totally understand the process of learning and testing，this assignment is really helpful for people who want to know the supervised learning problems.