Hw2 Report

Environment gcc 5.11,Cs 工作站,Ubuntu

Language c++

Code explanation

Part 1 Build k-d tree

the feature of each layer of the tree follow the sequence of 0,1,2,3,4,5,6,7,8,0,1,2,…

feature num of each node is presented by attr

calculate the median of the feature

```
dat node::calculatemedium(int n)    //feature n
{
    vector<dat> tmp;
    tmp.assign(s.begin(),s.end());
    sortclass = n;
    sort(tmp.begin(),tmp.end(),cmp);
    return tmp[tmp.size()/2];                //指定 feature 的中位數
}
```

Save the median data in the tree->med then delete it from following set

use this median to partition the dataset to 2 subset(data that its value of the feature < median =>left else =>right)

then use 2 new subset to build left and right subtree(repeat build tree)

➔ dctree* dctree::buildtree(dctree *tree,node set)

if we get the subset that is null(no data belongs to this subtree)

```
if(set.s.size()==0)
    {
        tree->attr = -1;            //  -1->不算葉節點
    tree->left = new dctree();      //建構子指定 attr = -1(所以她的左右子樹也不算節點)
    tree->right = new dctree();
        return tree;
    }
```

Subset 只剩一個 data -> 葉節點

```
    if(set.s.size()==1)                                    {
        tree->attr=9;                    //attr =9 ->葉節點
        tree->left = new dctree();
        tree->right = new dctree();
        tree->med = set.s[0];
        return tree;
```

```
    }
```

Part2 implement knn by kd tree

Algorithm I perceived:

1. 找到輸入的 data 經過 kdtree 走到的末節點(searctree,此點為暫時最近點)
2. 將此暫時最近點的 data 加入 knn 這個 vector (getktmp)
   有加入 knn 的點->attr 設為-1(代表不再是節點)
3. 追溯樹 直到他回到樹的 root
   若是 tree 的 root 到輸入點距離>輸入點到 parent 分割之後形成的 hyperplane 距離=>兩者維度為 parent->attr 的值相減為點到平面距離
   代表有可能在他的兄弟子樹(besttemp->bro)中，有更近的點(比輸入點到 parent 代表的點還要近)
   需進入兄弟節點搜索完才能回到 parent, o.w 回到 parent 且因此兄弟節點沒有更近點->不用搜索->所以 attr 為-1(不再是樹的節點)

```
 void getktmp(dctree *best,int k)    //讓 knn 維持在前 k 個最近的
 {
      if(knn.size()<k)      //未滿 k 個 加進去 sort
      {
      knn.push_back(best->med);
      sort(knn.begin(),knn.end(),cmp1);
      }
      else if(best->med.dist<knn[knn.size()-1].dist)      //已經滿 k 個 但出現更近的
      {
           knn.erase(knn.end());                    //刪掉最遠的
           knn.push_back(best->med);            //把新的加進去 sort
           sort(knn.begin(),knn.end(),cmp1);
      }
  }
dctree* knnfunc(dctree *tree,int k,dat q)
{
     dctree *besttemp = tree->searchtree(q,tree);     //找到末節點
     while(besttemp!=tree->parent)            //當未回溯到 root 完畢
     {
     dat besttmp = besttemp->med;
     besttemp->med.dist = distance1(q,besttmp);
     if(besttemp->attr!=-1||besttemp->parent == 0)    //若 attr 不為-1(還是節點) 代表第一次當最佳點且有資料在裡面
```

```
    {
        getktmp(besttemp,k); //將暫時最佳點加進去 knn
        besttemp->attr = -1;    //已加過 attr 設為-1(避免重複判斷)
    }
    else          //attr == -1 已不是節點 所以直接回到上一層
    {besttemp = besttemp->parent; continue;
    }
    if(besttemp->parent!=tree->parent)      //不是 root
    {
        dat pp = besttemp->parent->med;
    int att = besttemp->parent->attr;
        if(besttemp->bro->attr>=0&&distance1(q,tree->med)
>fabs(pp.dimension[att]-q.dimension[att])) //另一邊可能有更近勹 且兄弟節點存
在
        {
            besttemp = besttemp->bro;
            besttemp = knnfunc(besttemp,k,q); //進入兄弟節點查詢 查完之後
會回到原本的 besttemp 的 parent
        }
        else
        {
            besttemp->bro->attr = -1;              //兄弟節點未存在更近點
            besttemp = besttemp->parent;         //回到 parent
        }
    }
    else                                          //此最佳點為 root 此時回到原本的
tree 的位置,root knn 也判斷完畢
    {
            besttemp = besttemp->parent;
        }
    }
    return besttemp;
}
```

```
dctree* dctree::searchtree(dat q,dctree* tree)    //搜尋末節點
{
    if(tree->left->attr == -1&&tree->right->attr == -1)    //有可能為葉節點或者不是節點
    {
        if(tree->attr>=0)       //葉節點
        return tree;
        else
        return searchtree(q,tree->bro);       //不是節點  兄弟節點存在
    }
    if(tree->right->attr<0)                  //右節點不存在
        return searchtree(q,tree->left);
    else if(tree->left->attr<0) return searchtree(q,tree->right);    //左節點不存在
    else if(q.dimension[tree->attr]<tree->med.dimension[tree->attr]) //比中位數小
    {
    return searchtree(q,tree->left); //往左走
    }
    else if(q.dimension[tree->attr]>=tree->med.dimension[tree->attr]) //比中位數大
    {
        return searchtree(q,tree->right); //  往右走
    }
}
```

Part 3 predict

找到 knn 裡面出現頻率最高的那個 class -> 預測他 看他有沒有跟測資真正 class 相同

算測資裡面有預測正確的比率