

中山大学数据科学与计算机学院  
移动信息工程专业-人工智能  
本科生实验报告  
(2016 学年秋季学期)

课程名称: Artificial Intelligence

教学班级	14M2	专业(方向)	移动互联网
学号	14353205	姓名	刘万里

## 一、实验题目

分类: 必须实现 1-NN, 使用欧氏距离, 在 246 个训练集文本上进行训练, 再在 1000 个测试文本上进行预测, 得到所有测试文本的情感预测结果, 与标准答案(test 文本第二列)进行对比, 将准确率记录在实验报告上

回归: 使用 k-NN 处理回归问题, 得出所有 500 个测试文本属于每个情感(一共 6 种)的概率, 计算出在验证集上的相关系数(使用 validation.xlsx 文件进行计算), 并记录在实验报告中

## 二、实验内容

## 1. 算法原理

分类: 在实验 1 的基础上做出适当修改, 用一个 `vector<vector<int>>` 的数据结构存储每一行的字符串在 OneHot 矩阵中出现的下标, 最后在进行测试的时候把当前测试行和每一个样本行计算距离, 用这个数据结构便可以计算而不用计算 OneHot 矩阵上每一列的距离, 以降低算法复杂度。最后找到的最小的距离就可以决定分类结果。

回归: 在第一个任务的基础上, 无需做出很大改动, 在读取训练集的时候用一个二维矩阵保存下每一句话的 6 个概率。在读取到验证集的字符串时, 对训练集的每一行保存下相互间的距离, 用一个二维矩阵存储。最后读取完成后对验证集的每一行根据已得到的矩阵按公式计算出对应的概率, 最后对 6 个概率进行归一化。

## 2. 伪代码

分类:

```
int MinLine = 0, MinDis = 999999;
```

```
for(int r = 0; r < 246; r++){ //与训练文本的每一行求欧式距离
```

```
    if(这行的字符串个数 > 3)
```

```
        //这里是自己选择的优化, 因为 2 个字母的样本容易使所有测试样本都和它距离最小
```

```
        {
```

```
            得到当前行和第 r 行的 dis;
```

```
            if(dis < MinDis){
```

```
                MinDis = dis;
```

```
                MinLine = r;
```

```
            }
```

```
        }
```

```
    }
```

```
    输出这行的感情预测值到文本;
```

```
    if(预测值和正确答案相等) RightNums++;
```

```
}
```

```
最后分类正确率 = RightNums / 测试样本个数;
```



回归:

```
for(int r = 0; r < 246; r++){ //与训练文本的每一行求欧式距离
    得到距离最小值 dis;
    DIS[i-246][r] = dis;    //得到距离矩阵
}

for(int i = 0 ; i < 500; i++) //实际上代表第 i + 246 行
{
    double SUM[6], all = 0;
    for(int j = 0 ; j < 6 ; j++)
    {
        double sum = 0;
        sum = 该行对训练文本的每一行的第 J 种感情的概率的和;
        SUM[j] = sum;    //得到这种感情的概率
        all += sum;
    }
    归一化并得到 6 个概率的值;
}
```

关键代码截图（带注释）

分类:

```
if(i >= 246){ //test.txt来分类咯;
    int MinLine = 0, MinDis = 999999;
    for(int r = 0; r < 246; r++){ //与训练文本的每一行求欧式距离
        if(NumsPerLines[r].size() > 3)
        {
            int dis = 0, c = 0, k = 0, equals = 0;
            for(; c < NumsPerLines[i].size(); c++){ //当前行的字符串
                for(; k < NumsPerLines[r].size(); k++){
                    if(NumsPerLines[i][c] < NumsPerLines[r][k]){
                        break;
                    }
                    else if(NumsPerLines[i][c] == NumsPerLines[r][k]){
                        equals++;
                    }
                }
            }
            dis = NumsPerLines[i].size() + NumsPerLines[r].size() - equals * 2;
            if(dis < MinDis){
                MinDis = dis;
                MinLine = r;
                //cout<<"smaller, dis = "<<dis<<endl;
            }
        }
    }
    OneHot<<"min line = "<<MinLine<<"; MinDis = "<<MinDis<<"; emotion = "<<Emo[MinLine]<<"; ";
    if(Emo[MinLine] == Emo[i]) RightNums++;
}
```

回归:



```
vector<int> Temp_Vec;
for(it2=repeatTimes.begin();it2!=repeatTimes.end();it2++) //为了在vector中有序,借用这个map
{
    Three_Var item(Lines,it2->first); //利用机构体记录这个单词出现的行数,下标
    smat.push_back(item);
    ThreeVarNums++; //结构体vector中放入的个数
    Temp_Vec.push_back(it2->first); //这个结构存储每一行拥有的字符串的下标;
}
NumsPerLines.push_back(Temp_Vec);
Lines++; //第几行计数
if(Lines == 246){
    first = true;
    Input.close();
    Input.open("Dataset_validation.txt");
}

if(i >= 246){ //test.txt来分类咯;
    int MinLine = 0;
    Info << i << " ";
    for(int r = 0; r < 246; r++){ //与训练文本的每一行求欧式距离
        int dis = 0;
        int c = 0, k = 0, equals = 0;
        for(; c < NumsPerLines[i].size(); c++){ //当前行的字符串
            for(; k < NumsPerLines[r].size(); k++){
                if(NumsPerLines[i][c] < NumsPerLines[r][k])
                    break;
                else if(NumsPerLines[i][c] == NumsPerLines[r][k])
                    equals++;
            }
        }
        dis = NumsPerLines[i].size() + NumsPerLines[r].size() - equals * 2; //得到这个距离了
        DIS[i-246][r] = dis; //得到距离矩阵
        Info<<dis << " ";
    }
    Info << endl;
}

for(int i = 0; i < 500; i++) //实际上代表第 i + 246 行
{
    //Pro << "text"<< i+1 << ": ";
    double SUM[6], all = 0;
    for(int j = 0; j < 6; j++)
    {
        double sum = 0;
        for(int k = 0; k < 246; k++)
        {
            sum += P[k][j]/ DIS[i][k];
        }
        SUM[j] = sum;
        all += sum;
    }
    for(int j = 0; j < 6; j++) //归一化
    {
        Pro << SUM[j]/all<<" ";
    }
    Pro << endl;
}
```

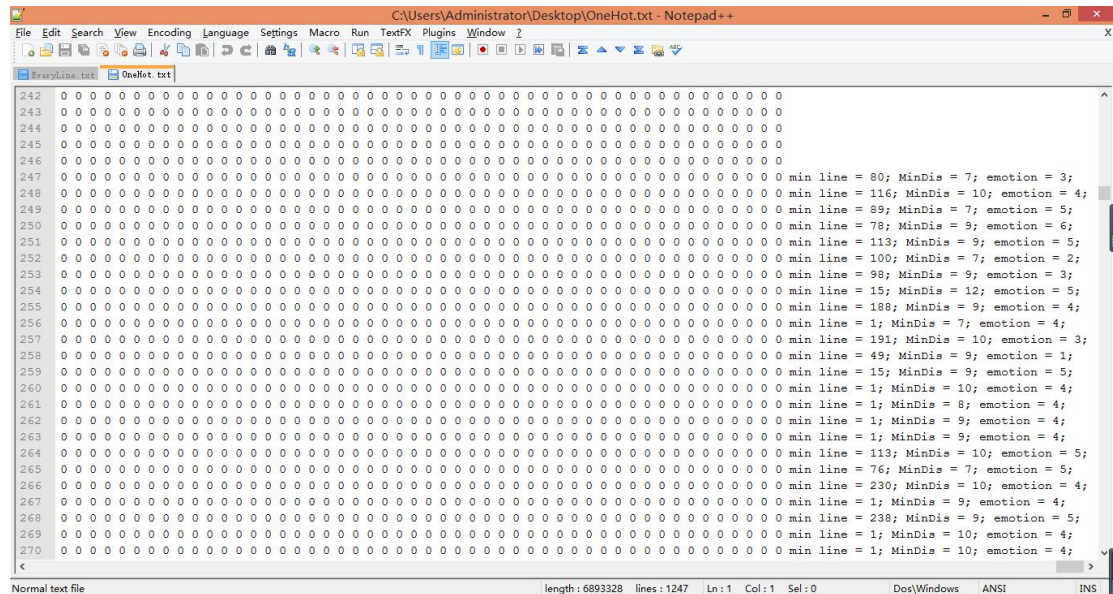
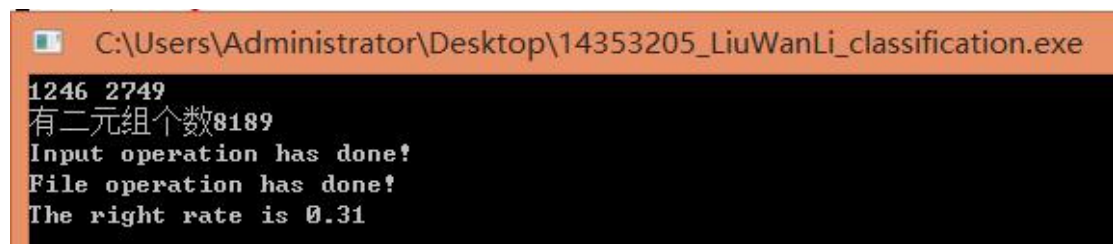
### 3. 创新点&优化

无

### 三、实验结果及分析

#### 1. 实验结果展示示例（可图可表可文字，尽量可视化）

分类：

回归：

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
id	anger	disgust	fear	joy	sad	surprise			anger	disgust	fear	joy	sad	surprise
1	0.123373	0.085446	0.136417	0.27603	0.210747	0.167987		r	0.05686954	0.1078458	0.262546	0.239433	0.233311	0.155688
2	0.123609	0.08686	0.132815	0.277089	0.211557	0.16807		average	0.175949007					
3	0.124564	0.087497	0.132817	0.278603	0.210711	0.165808		evaluation	极弱相关 加油哦					