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

（2017-2018 学年秋季学期）

课程名称：**Artificial Intelligence**

教学班级	**15M1**	专业（方向）	**软件工程（移动信息工程）**
学号	**15352033**	姓名	**陈黄胤**

一、 实验题目

Lab2：K 近邻与朴素贝叶斯——分类和回归

二、 实验内容

1. 算法原理

Ⅰ.KNN_Classification

Step1: 对训练数据集进行读取并处理，提取信息并运算得到不同的特征向量(lab1 中的 ONE-HOT、TF、TF-IDF 等等)

Step2: 与 Step1 中的操作一致，提取验证(测试)数据集的特征向量。

Step3: 计算验证(测试)数据集特征向量到每一个训练数据集特征向量的距离，其中的距离有多种定义(包括欧拉距离、曼哈顿距离、余弦距离、切比雪夫距离等等)，而且可以根据需要，对上述距离进行参数调整处理。

Step4: 对 Step3 中的距离进行排序，根据距离(欧拉距离/曼哈顿距离越小代表越相似,余弦距离越接近 1 代表越相似)，取与验证(测试)集最相似的 K 组训练集数据作为匹配数据。

Step5: 统计 Step4 中提取出来的 K 组数据的特征 label，在这 K 组中出现频率最高的数据 label，即为 KNN 的分类预测结果。

Ⅱ.KNN_Regression

Step1 -> Step4: 与 KNN_Classification 一致

Step5: 对于 Step4 中提取出来的 K 组数据的各属性特征值按照公式

P’=P(Train)*Dis(这里的 Dis 是余弦距离，若是欧拉距离则乘以倒数)求解，然后将 K 组的 P’求和即可得到验证(测试)数据集这个属性的回归预测值。

Ⅲ.NB_Classification

Step1: 对训练数据集进行读取并处理，提取信息并运算得到 ONE-HOT 矩阵，并标记好对应数据的 label

Step2: 对验证(测试)数据集进行读取，对于每一行验证(测试)集进行遍历，求取

每一个 label 的可能性。对于出现的单词，求取这个单词在训练集上 label 为当前所求 label 的句子中所有单词的比例，把验证(测试)集的每一个单词从训练集中比例求出，然后将每个单词的这个结果累乘，就能得到这个 label 的预测概率。按以上方法，求得所有 label 的概率，对概率进行排序，取最高概率的 label 作为 NB 分类预测值。

Ⅳ.NB_Regression

Step1: 对训练数据集进行读取并处理，提取信息并运算得到 TF 矩阵，并存储好对应 label 的概率。

Step2: 对验证(测试)数据集进行读取，对于每一行验证(测试)集进行遍历，求取每一行验证(测试)数据集每一个 label 的概率。对于出现的单词，求取这个单词在训练集上每一行句子中的 TF 矩阵值乘积，然后再乘以这个训练集数据行对应 label 的几率，对每一个训练集数据进行累加就能得到这个 label 的预测概率。按以上方法，求得这一行验证(测试)集的所有 label 的概率，然后输出即可。

2. 伪代码

```
Function KNN_Classification()
{
    Lab1.GET_TFIDF(); //采用 lab1 中的方式获取训练集与测试集的 tfidf 矩阵
    For i=0 to i=test_set.size()   //遍历测试集，对每句进行求解
        For j=0 to j=train_set.size()
            Dis[j]=Get_distance(tfidf[i],tfidf[j]) //这里的 dis 函数求解余弦距离，点乘除
                                                   以模乘
        Sort(Dis,Dis+j,cmp);   //cmp 定义为按距离降序排序
        For i=0 to i=k //k 为 KNN 中的 K 取值
            Count_label[ Dis[i].label ]++;
        Print Count_label.max().label   //输出计数最多的 label
}

Function KNN_Regression()
{
    Lab1.GET_TFIDF(); //采用 lab1 中的方式获取训练集与测试集的 tfidf 矩阵
    For i=0 to i=test_set.size()   //遍历测试集，对每句进行求解
        For j=0 to j=train_set.size()
            Dis[j]=Get_distance(tfidf[i],tfidf[j]) //这里的 dis 函数求解余弦距离，点乘除
                                                   以模乘
        Sort(Dis,Dis+j,cmp);   //cmp 定义为按距离降序排序
        For i=0 to i=k //k 为 KNN 中的 K 取值
            Label_rate[ Dis[i].label ]+=Dis*Regression[dis[i].index]; //乘以对应的回归
                                                   //值，Dis 由于是余弦距离作为优化参数乘进去
        For i=0 to i=5
            Label_rate[i]/=Label_rate.sum()   //归一化概率
            Print Label_rate[i]   //输出这一个 label 的概率
```

```
}

Function NB_Classification()
{
    Lab1.GET_ONEHOT(); //采用 lab1 中的方式获取训练集与测试集的 onehot 矩阵
    For i=0 to i=test_set.size()   //遍历测试集，对每句进行求解
        For j=0 to j=train_set.size()
            Label_rate[label[k]]*=num(train[j].word.label == label[k]) /num(word.label ==
                                    label[k])   //求取这个单词，在 label 相同训练集句子中
                                                //的占这些句子总单词的比例
                                                //并将测试集中每个单词的这个值累乘起来
        For k=0 -> k=5
        Label_rate[label[k]]*=train_rate[label[k]] //乘上这个标签在训练集上出现的比例
        Print Label_rate.max().label   //输出概率最高的 label
}

Function NB_Regression ()
{
    Lab1.GET_TF(); //采用 lab1 中的方式获取训练集与测试集的 TF 矩阵
    For i=0 to i=test_set.size()   //遍历测试集，对每句进行求解
        For j=0 to j=train_set.size()
            If(train[j] in test_set !=0)
            tmp[label[k]]*=TF[j] //累乘出现的单词在每一个训练集句子上的 TF
        Label_rate[label[k]] += tmp[label[k]]*Regression[j] //乘上训练集的回归值后累加
        For i=0 to i=5
            Label_rate[i]/=Label_rate.sum()   //归一化概率
            Print Label_rate[i]   //输出这一个 label 的概率
}
```

3. 关键代码截图（带注释）

KNN:

```
77  double get_dis(vector<double> a, vector<double> b ) //获得两个特征向量的距离
78  {
79      double dot=0,alen=0,blen=0;
80      for(int i=0;i<a.size();i++)
81      {
82          dot += (a[i]*b[i]);
83          alen += (a[i]*a[i]);
84          blen += (b[i]*b[i]);
85      }
86      return (dot/(sqrt(alen)*sqrt(blen)));  //返回余弦距离
87  }
```

```
//classification
dis *__dis=new dis[Ts];
for(int j=0;j<Ts;j++)
{
    __dis[j]._dis=get_dis(SET.TFIDF[i],SET.TFIDF[j]);  //获取两个向量的距离
    __dis[j].index=j;  //存储在总表中位置
}

sort(__dis,__dis+Ts,cmp);  //由于使用余弦距离，所以这里使用降序排序
```

```
126         for(int j=0;j<14;j++) k=14
127         {
128             cnt[Train_label_list[__dis[j].index]]+=__dis[j]._dis;       //分类计数，将余弦距离作为系数
129             for(int k=0;k<6;k++)
130             {
131                 regre_res[k]+=Regression[__dis[j].index][k]*__dis[j]._dis;   //同时进行回归值计算，
132             }                                                   //将训练集概率乘上余弦距离，叠加到各个标签的概率值上
133         }
```

```
for(map<string,double>::iterator iter=cnt.begin(); iter!=cnt.end(); iter++)
{
    if(iter->second >= max)
    {
        max=iter->second;
        ans=iter->first;
    } //查找分类计数值最大的标签，并将之输出
}
out1 << i-Ts+1 << "," << ans << endl;
```

```
//regression
for(int k=0;k<6;k++)
{
    regre_sum+=regre_res[k];
}
for(int k=0;k<6;k++)
{
    if(regre_sum!=0)
    regre_res[k]=regre_res[k]/regre_sum;
}                                   //回归值归一化
```

NB:

```
//classify
for(int i=0;i<V_SET.SENTENCE_LIST.size();i++)
{
    map<string,double> rating;

    for(int j=0;j<6;j++)
    {
        if(rating[_labels[j]] == 0)
        rating[_labels[j]] = 1;  //待累乘数归1

        for(int k=0;k<V_SET.SENTENCE_LIST[i].word_list.size();k++)
        {
            rating[_labels[j]]*=(double)( word_cnt[SET.WORD_INDEX[V_SET.SENTENCE_LIST[i].word_list[k].c
        }  //求取这个单词，在label相同训练集句子中的占这些句子总单词的比例
        rating[_labels[j]]*=s_label_cnt[_labels[j]]/(double)Ts;  //乘上这个label在训练集的比例
    }
```

```
[i].word_list[k].content]][_labels[j]]+0.44)/(double)(label_cnt[_labels[j]]+0.44*SET.len);

集的比例
```

```
double max=0;
int maxi=0;
for(int j=0;j<6;j++)
{
    if(rating[_labels[j]]>max)
    {
        max=rating[_labels[j]];
        maxi=j;
    }
}  //寻找并输出最大的rating值，并输出对应的label值
out1 << i+1 << "," << _labels[maxi] << endl;
```

```
//regression
for(int i=0;i<V_SET.SENTENCE_LIST.size();i++)
{
    map<string,double> rating;

    for(int j=0;j<6;j++)
    {
        for(int p=0;p<SET.SENTENCE_LIST.size();p++)
        {
            double tmp=1;
            for(int k=0;k<V_SET.SENTENCE_LIST[i].word_list.size();k++)
            {
                if(SET.WORD_INDEX[V_SET.SENTENCE_LIST[i].word_list[k].content]!=0)  //!important 忽略掉没出现过的单词
                tmp*=(SET.L_TF[p][SET.WORD_INDEX[V_SET.SENTENCE_LIST[i].word_list[k].content]]);
                //累乘拉普拉斯平滑后的TF矩阵
            }
            rating[_labels[j]]+=tmp*Regression[p][j]; //最后乘上训练集对应回归值后累加到预测概率上
        }
    }
```

```
}
double sum=0;
for(int j=0;j<6;j++)
{
    sum+=rating[_labels[j]]; //求和
}
out2 << i+1 << ",";
for(int j=0;j<6;j++)
{
    out2 << rating[_labels[j]]/sum << ",";
}
out2 << endl; //回归值归一化
```

4. 创新点&优化（如果有）

①KNN 计算向量距离的时候选择了余弦距离，对验证集测试发现效果比欧式距离要好许多。

②KNN 计算句子特征向量的时候采用了 TF-IDF 作为特征向量，把单词出现的频率以及单词在文章中的稀有度纳入了特征分析，通过验证集测试发现，比 ONE-HOT 作为特征向量有了很大的提升。

③KNN 统计邻近 K 点距离的时候采用了权值优化，考虑到虽然 K 点都应该纳入结果的考虑范围之中，但是由于这 K 个点本身也有远近之分，应该给予他们一个权值用于区分。

④NB 采用了拉普拉斯平滑解决了零概率的问题，并通过循环执行，找到了较优的拉普拉斯平滑系数，从而大大提高了预测的准确度。其中 NB 分类计算每一个累乘概率时采用了分子加上 0.44，分母加上 0.44*总单次数的优化方式。而 NB 的回归则在计算 TF 矩阵的时候，令分子为（单词出现次数+0.009），分母为（句子单词数+0.009*单词总表大小）进行优化。

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）

对 validation_set 进行分类预测

KNN 分类：

	A	B	C	D	E	F	G	H
298	man rides	surprise	joy	0				
299	tourist spo	sad	sad	1				
300	asia near p	joy	joy	1				
301	pakistan in	joy	joy	1				
302	australian	joy	joy	1				
303	planned ce	fear	fear	1				
304	two koreas	joy	joy	1				
305	books on s	surprise	surprise	1				
306	a mistrial f	anger	joy	0				
307	porn ring s	disgust	disgust	1				
308	madonna s	joy	joy	1				
309	lawmakers	anger	fear	0				
310	bill thanks	joy	joy	1				
311	japanese w	disgust	sad	0				
312	french subi	fear	joy	0				
313				149		准确数		
314				0.4791		准确率		
315								
316								
317								
318								
319								
320								

NB 分类：

	A	B	C	D	E	F	G
298	man rides	surprise	joy	0			
299	tourist spo	sad	sad	1			
300	asia near p	joy	joy	1			
301	pakistan in	joy	joy	1			
302	australian	joy	joy	1			
303	planned ce	fear	fear	1			
304	two koreas	joy	joy	1			
305	books on s	surprise	joy	0			
306	a mistrial f	anger	joy	0			
307	porn ring s	disgust	disgust	1			
308	madonna s	joy	joy	1			
309	lawmakers	anger	joy	0			
310	bill thanks	joy	joy	1			
311	japanese w	disgust	sad	0			
312	french subi	fear	joy	0			
313				144		准确数	
314				0.463023		准确率	
315							
316							
317							

KNN 回归：

id	anger	disgust	fear	joy	sad	surprise			anger	disgust
1	0.086427	0.10272	0.211883	0.226179	0.163944	0.208848		r	0.3701041	0.349752572
2	0.054632	0.018982	0.227082	0.306362	0.182588	0.210353		average	0.405762653	
3	0.063324	0.040392	0.246957	0.19566	0.155202	0.298465		evaluation	低度相关 666	
4	0.089915	0.072324	0.22971	0.108444	0.367843	0.131764				
5	0.05261	0.031209	0.142237	0.370547	0.174749	0.228648				
6	0.269728	0.087416	0.203738	0.028407	0.257083	0.153629				
7	0.170091	0.074114	0.234169	0.124262	0.22139	0.175973				
8	0.06303	0.058581	0.102142	0.48985	0.076982	0.209415				
9	0.048976	0.078841	0.208167	0.212734	0.192357	0.258926				
10	0.044482	0.055268	0.170055	0.31831	0.145954	0.265931				
11	0.051601	0.008549	0.154742	0.463845	0.056868	0.264395				
12	0.039657	0.026781	0.254811	0.127293	0.330627	0.220831				

NB 回归：

	A	B	C	D	E	F	G	H	I	J	K
1	id	anger	disgust	fear	joy	sad	surprise			anger	disgust
2	1	0.100655	0.087649	0.213104	0.225187	0.187965	0.18544		r	0.314988863	0.24294562
3	2	0.0436	0.04255	0.190332	0.233043	0.136363	0.354113		average	0.354033376	
4	3	0.070098	0.044242	0.238456	0.20695	0.184085	0.256169		evaluation	低度相关 666	
5	4	0.081059	0.051263	0.18816	0.217633	0.271444	0.19044				
6	5	0.053025	0.029773	0.174249	0.318811	0.200813	0.223328				
7	6	0.109919	0.057436	0.126591	0.276714	0.19918	0.23702				
8	7	0.160379	0.078417	0.326803	0.049704	0.271561	0.113136				
9	8	0.08699	0.08699	0.043499	0.521645	0.108689	0.152187				
10	9	0.009805	0.046987	0.244442	0.172837	0.027532	0.498398				
11	10	0.076873	0.078285	0.238187	0.113227	0.285076	0.208353				
12	11	0.037375	0.001895	0.106321	0.319555	0.190857	0.343997				
13	12	0.026505	0.044209	0.24785	0.026505	0.557612	0.09732				
14	13	0.105576	0.105071	0.11109	0.094305	0.295259	0.288698				
15	14	0.000289	0.000705	0.001277	0.000164	0.788066	0.209499				
16	15	0.084524	0.069067	0.191609	0.259729	0.200782	0.194289				

2. 评测指标展示即分析（如果实验题目有特殊要求，否则使用准确率）

对于 KNN：

如果取 K=1 时：

id	anger	disgust	fear	joy	sad	surprise			anger	disgust
1	0	0	0.7524	0.1238	0.1238	0		r	0.283613358	0.224154942
2	0.2	0.0696	0.3913	0	0.1652	0.1739		average	0.328902639	
3	0	0	0.384638	0.076908	0.076908	0.461546		evaluation	低度相关 666	
4	0.208179	0.253075	0.232677	0	0.236676	0.069393				
5	0.105289	0.092091	0.171083	0	0.526247	0.105289				
6	0.026505	0.044209	0.24785	0.026505	0.557612	0.09732				
7	0.174883	0.081992	0.420758	0	0.273173	0.049195				
8	0.086991	0.086991	0.043496	0.521648	0.108689	0.152185				
9	0	0.041696	0.249975	0.166683	0	0.541646				
10	0	0	0	0.8545	0	0.1455				
11	0.0375	0	0.1	0.3125	0.2	0.35				

301	pakistan in	joy	fear	0	
302	australian	joy	joy	1	
303	planned ce	fear	joy	0	
304	two koreas	joy	joy	1	
305	books on s	surprise	joy	0	
306	a mistrial f	anger	sad	0	
307	porn ring s	disgust	joy	0	
308	madonna	joy	joy	1	
309	lawmakers	anger	fear	0	
310	bill thanks	joy	joy	1	
311	japanese w	disgust	fear	0	
312	french subi	fear	joy	0	
313				139	准确数
314				0.446945	准确率
315					
316					

如果取 K=40 时：

298	man rides	surprise	joy	(Ctrl)	
299	tourist spo	sad	surprise	0	
300	asia near p	joy	joy	1	
301	pakistan in	joy	joy	1	
302	australian	joy	joy	1	
303	planned ce	fear	surprise	0	
304	two koreas	joy	joy	1	
305	books on s	surprise	joy	0	
306	a mistrial f	anger	joy	0	
307	porn ring s	disgust	disgust	1	
308	madonna	joy	joy	1	
309	lawmakers	anger	fear	0	
310	bill thanks	joy	joy	1	
311	japanese w	disgust	sad	0	
312	french subi	fear	joy	0	
313				139	准确数
314				0.446945	准确率
315					

	A	B	C	D	E	F	G	H	I	J
1	id	anger	disgust	fear	joy	sad	surprise			anger
2	1	0.100006	0.08872	0.240532	0.214444	0.173668	0.18263		r	0.35674794
3	2	0.075833	0.041337	0.249628	0.250333	0.189259	0.193611		average	0.379005532
4	3	0.060295	0.03846	0.244232	0.206392	0.15017	0.300452		evaluation	低度相关 666
5	4	0.089915	0.072324	0.22971	0.108444	0.367843	0.131764			
6	5	0.085255	0.040233	0.149072	0.347601	0.154113	0.223727			
7	6	0.168946	0.065171	0.148104	0.206121	0.217743	0.193914			
8	7	0.140616	0.064188	0.229858	0.119297	0.273094	0.172946			
9	8	0.068379	0.061947	0.106977	0.469211	0.079784	0.213701			
10	9	0.056342	0.082228	0.220752	0.201944	0.192942	0.245793			
11	10	0.049588	0.047847	0.152726	0.302483	0.172036	0.275321			
12	11	0.066986	0.024556	0.160946	0.395153	0.113783	0.238576			

在这个范围内迭代求解正确率，发现 K=13 时能获得最佳的正确率和最高回归相关度

对于 NB：

进行拉普拉斯平滑后：

	A	B	C	D	E	F	G
298	man rides	surprise	joy	0			
299	tourist spo	sad	sad	1			
300	asia near p	joy	joy	1			
301	pakistan in	joy	joy	1			
302	australian p	joy	joy	1			
303	planned ce	fear	fear	1			
304	two koreas	joy	joy	1			
305	books on s	surprise	joy	0			
306	a mistrial f	anger	joy	0			
307	porn ring s	disgust	disgust	1			
308	madonna s	joy	joy	1			
309	lawmakers	anger	joy	0			
310	bill thanks	joy	joy	1			
311	japanese w	disgust	sad	0			
312	french subi	fear	joy	0			
313				144		准确数	
314				0.463023		准确率	
315							
316							
317							

	A	B	C	D	E	F	G	H	I	J	K
1	id	anger	disgust	fear	joy	sad	surprise			anger	disgust
2	1	0.100655	0.087649	0.213104	0.225187	0.187965	0.18544		r	0.314988863	0.24294562
3	2	0.0436	0.04255	0.190332	0.233043	0.136363	0.354113		average	0.354033376	
4	3	0.070098	0.044242	0.238456	0.20695	0.184085	0.256169		evaluation	低度相关 666	
5	4	0.081059	0.051263	0.18816	0.217633	0.271444	0.19044				
6	5	0.053025	0.029773	0.174249	0.318811	0.200813	0.223328				
7	6	0.109919	0.057436	0.126591	0.276714	0.19318	0.23702				
8	7	0.160379	0.078417	0.326803	0.049704	0.271561	0.113136				
9	8	0.08699	0.08699	0.043499	0.521645	0.108689	0.152187				
10	9	0.009805	0.046987	0.244442	0.172837	0.027532	0.498398				
11	10	0.076873	0.078285	0.238187	0.113227	0.285076	0.208353				
12	11	0.037375	0.001895	0.106321	0.319555	0.190857	0.343997				
13	12	0.026505	0.044209	0.24785	0.026505	0.557612	0.09732				
14	13	0.105576	0.105071	0.11109	0.094305	0.295259	0.288698				
15	14	0.000289	0.000705	0.001277	0.000164	0.788066	0.209499				
16	15	0.084524	0.069067	0.191609	0.259729	0.200782	0.194289				

实现的时候进行了拉普拉斯平滑的系数迭代，找到了最佳的拉普拉斯平滑系数。具体平滑值见优化和创新点处。

四、 思考题

1. 为什么是倒数呢？

如果是欧拉距离的话要取倒数，因为距离越小说明相似度越高，概率所占的比重应该更大，所以取倒数可以实现距离越小权重越高。而如果使用余弦距离则不需要取导数，因为余弦距离越接近 1 则相似度越高，权重也应该越高。

2. 同一测试样本的各个情感概率总和应该为 1 如何处理

将初始所得的概率相加，然后把每个初始概率除以这个概率和即可实现概率的归一化。

3. 在矩阵稀疏程度不同时，曼哈顿距离和欧式距离表现有什么区别，为什么？

当稀疏程度较小时，曼哈顿距离可能会比欧式距离更加稳定一些，因为欧式距离的相同率会更高，会出现大量的不同个体间存在相同的欧式距离，这时欧式距离来筛选特征就

会缺乏准确度，但是曼哈顿距离对比欧式距离会掩盖特征之间的邻近关系，使用曼哈顿距离会忽略掉特征之间的一些关系。当矩阵稀疏程度很大的时候，曼哈顿距离和欧式距离的表现逐渐趋于一致，两者的区别越来越小。

4. 伯努利模型和多项式模型分别有什么优缺点

伯努利模型非 0 即 1，对于一个训练集忽略了样本的频率信息，只单纯分析了是否出现过，并不考虑他在句子中出现的频率，优点是确定方法简单，运算起来很简洁。
多项式模型忽略了同一个单词在一个句子中多次出现的情况，默认了他们的出现是独立事件，然而事实上并非如此，一个单词多次出现很有可能要大幅提高对应概率的比例，而不只是单纯的线性提升。优点是对比伯努利模型，考虑到了样本内部的频率信息。

5. 如果测试集中出现了一个之前全词典中没有出现过的词该如何解决

直接忽略这个一个词，这一个词不参与概率累乘。