

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)

姓名	学号	教学班级	课程名称	专业方向
张子豪	15352427	15M1	Artificial Intelligence	移动互联网

一、实验题目

K近邻与朴素贝叶斯——分类和回归

二、实验内容

1.算法原理

K近邻算法：

将训练集每个文本都用一个特征向量去表示(这个特征向量可以是0-1形式，也可以是tf、tf\_idf等值)。对每一个待处理文本，求出该文本的特征向量与训练集每条文本的特征向量的距离，(可以是曼哈顿距离/欧式距离...也可以用余弦相似度衡量)，然后取出与该待处理文本前k个距离最相近的文本。

对于分类问题，这些被取出的文本的标签的众数就是K近邻算法得到的结果；对于回归问题，则要将距离作为权值进行加权，计算该文本属于某一标签的概率。

朴素贝叶斯算法：

朴素贝叶斯的前提是特征之间强独立，每个属性在判定文本的标签时的概率分布上独立。

对于分类问题，常见的有伯努利模型和多项式模型。伯努利模型中的特征只有0-1，即是否出现在文档中，多项式模型中的特征是单词，值是单词的出现次数。

以多项式模型为例，设某文档 $d=(t_1, t_2, \dots, t_k)$ ， $t_k$ 是该文档中出现过的单词，假如要求标签为joy的概率：

根据贝叶斯定理， $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ ，可知 $P(joy|d) = \frac{P(d|joy) * P(joy)}{P(d)}$ ，由于 $P(d)$ 是常数，归一化

后不产生影响，所以计算分子即可。

先验概率 $P(c)$ =类c下单词总数/整个训练样本的单词总数

类条件概率 $P(t_k | c) = (\text{类c下单词}t_k\text{在各个文档中出现过的次数之和} + \alpha) / (\text{类c下单词总数} + \text{不重复单词总数} * \alpha)$ ，其中 $\alpha$ 的取值在0-1之间

则 $P(joy | d) = P(d | joy) * P(joy) = P(t_1 | c) * P(t_2 | c) * \dots * P(t_k | c) * P(joy)$ 。

对于回归问题， $P(joy | d) = P(t_1 | c, d_1) * P(t_2 | c, d_1) \dots P(t_k | c, d_1) * P(d_1, joy) + P(t_1 | c, d_2) * P(t_2 | c, d_2) \dots P(t_k | c, d_2) * P(d_2, joy) + \dots$ ，其中 $P(d_n, joy)$ 为先验概率。

直接套用这个公式的话，则要求训练集中某个语句拥有测试文本的所有词，否则结果均为0，而这显然是不切实际的，因此需要使用拉普拉斯平滑，将 $P(t_k|c, d_i)$ 改为

$\frac{x_k + \alpha}{\sum_{k=1}^K x_k + K * \alpha}$ ，这样为所有值都手动加了一个数，避免了乘法过程中出现0的情况，alpha的取值同样是0-1。

## 2.伪代码

kNN分类：

- 处理训练集数据
  - 读取所有原始数据，去掉第一行
  - for遍历每一行，istringstream读取被空格分割的每个字符串
  - 处理读取到的最后一个字符串得到最后一个单词与对应情感值
- 获得特征向量vector
  - map+for循环，统计不重复词语
  - for 遍历训练集
    - for 遍历不重复词语
    - if(该训练文本存在这个词语) vector.push\_back(TF\_IDF)
    - else vector.push\_back(0)
- 统计每个测试文本与每个训练文本的距离
  - for 遍历测试集
    - for 遍历训练集
    - 计算距离
- 预测各文本的标签
  - for 遍历合适的k
    - for遍历测试集，对当前这个测试文本
    - 取训练集中前k个与该测试文本最近的文本
    - 根据距离加权，取最大权值对应的标签
    - 输出结果

kNN回归：

- 处理训练集数据
  - 读取所有原始数据，去掉第一行
  - for遍历每一行，istringstream读取被空格分割的每个字符串
  - 处理读取到的最后一个字符串得到最后一个单词与对应概率
- 获得特征向量vector
  - map+for循环，统计不重复词语
  - for 遍历训练集
    - for 遍历不重复词语
    - if(该训练文本存在这个词语) vector.push\_back(TF\_IDF)

else vector.push\_back(0)

- 统计每个测试文本与每个训练文本的距离
  - for 遍历测试集
  - for 遍历训练集
  - 计算距离
- 计算测试集各文本的概率
  - for 遍历合适的k
  - for遍历测试集，对当前这个测试文本
  - 取训练集中前k个与该测试文本最近的文本
  - for 遍历每个标签 根据距离加权，计算测试文本属于该标签的概率
  - 归一化
  - 输出结果

贝叶斯分类：

- 处理训练集数据
  - 读取所有原始数据，去掉第一行
  - for遍历每一行，istringstream读取被空格分割的每个字符串
  - 处理读取到的最后一个字符串得到最后一个单词与对应情感值
- 获得所有标签种类数量
  - for 遍历训练集
  - set.insert(标签)
  - set.size()
- 获得所有不重复单词
- 获得标签-该标签下的单词总数 map<string,int>存储
- 获得所有先验概率  $P(c) = \text{类}c\text{下文件总数} / \text{整个训练样本的文件总数}$
- 预测各文本标签
  - for 遍历测试集 对当前文本
    - 去除掉训练集中不存在的词语，并相应减小句子长度
    - if(句子长度=0) 所有单词训练集都不存在，无法预测，不如joy一下
    - 根据多项式模型计算各标签概率
    - 取概率最大的标签作为预测结果

贝叶斯回归：

- 处理训练集数据
  - 读取所有原始数据，去掉第一行
  - for遍历每一行，istringstream读取被空格分割的每个字符串
  - 处理读取到的最后一个字符串得到最后一个单词与对应概率
- 获得所有不重复单词
- 计算训练集tf向量，并使用拉普拉斯平滑
- 计算测试集各文本概率

- 去除掉训练集中不存在的词语，并相应减小句子长度
- if(句子长度=0) 所有单词训练集都不存在，无法预测，所有标签概率六等分
- $P(\text{joy}|\text{d})=P(\text{t1}|\text{c},\text{d1})*P(\text{t2}|\text{c},\text{d1})\dots P(\text{tk}|\text{c},\text{d1})*P(\text{d1},\text{joy})+ P(\text{t1}|\text{c},\text{d2})*P(\text{t2}|\text{c},\text{d2})\dots P(\text{tk}|\text{c},\text{d2})*P(\text{d2},\text{joy})+\dots$
- .....
- 计算出所有标签概率，归一化，输出结果

### 3.关键代码截图(带注释)

kNN:

```
//求余弦相似度;为正且值越大则差距越小;为负则代表差距较大,负数的绝对值越大则差距越大
double dot=0;
for(int i=0;i<a.size();i++) dot+=a[i]*b[i];
double a_=0,b_=0;
for(int i=0;i<a.size();i++)
{
    a_+=a[i]*a[i];
    b_+=b[i]*b[i];
}
if(a_==0||b_==0) return 999999;//某一向量为零向量,两向量正交,无任何相似可言
return dot/sqrt(a_*b_);
```

```
//KNN分类
for(int k=1;k<=50;k++)//遍历寻找合适的k值
{
    int true_num=0;//分类正确的数量
    for(int i=0;i<verification.size();i++)//遍历验证集
    {
        map<string,double> probability_mood;//标签-该标签出现概率
        stable_sort(verification[i].distance.begin(),verification[i].distance.end(),cmp);//根据
距离排序
        for(int j=0;j<k;j++)//取topK个标签,根据距离进行加权
        probability_mood[verification[i].distance[j].mood]+=verification[i].distance[j].dist;
        string ret_mood;//预测结果
        double max_ret=0;
        for(map<string,double>::iterator
it=probability_mood.begin();it!=probability_mood.end();it++)//寻找概率最大的标签
        {
            if(it->second>max_ret)
            {
                max_ret=it->second;
                ret_mood=it->first;
            }
        }
        verification[i].mood_from_train=ret_mood;
        if(ret_mood==verification[i].mood) true_num++;
    }
    cout<<"k="<<k<<"    "<<"rate:"<<(double>true_num/verification.size())<<endl;
}
}
```

```

//KNN回归
for(int i=0;i<verification.size();i++)//对这条语句，取离他最近的k个邻居
{
    sort(verification[i].distance.begin(),verification[i].distance.end(),cmp);//根据距离排序
    for(int m=0;m<6;m++) verification[i].mood_from_train[m]=0;//遍历每种标签，并将结果初始化为0
    for(int j=0;j<k;j++)//取topK个标签
    {
        for(int m=0;m<6;m++)//遍历每种标签
        {
            verification[i].mood_from_train[m]+=verification[i].distance[j].dist*verification[i].distance[j]
            .mood[m];//由于使用余弦作为距离衡量，因此用余弦值乘概率就可以完成加权
        }
    }
    double sum=0;
    for(int m=0;m<6;m++) sum+=verification[i].mood_from_train[m];
    for(int m=0;m<6;m++)
    {
        verification[i].mood_from_train[m]/=sum;//归一化
        out<<verification[i].mood_from_train[m]<<"\t";
    }
    out<<endl;
}

```

贝叶斯:

```

//贝叶斯分类
map<string,double> probability;//标签-该标签对应概率
//对这条文本，求每种情感的概率
for(set<string>::iterator it=all_mood.begin();it!=all_mood.end();it++)//遍历每种情感
{
    for(int j=0;j<tmp.size();j++)//遍历删去训练集中不存在的单词后的句子
    {
        if(!probability[*it]) probability[*it]=
        (Get_num_of_times_word_occurs(*it,tmp[j])+0.37)/(num_of_mood[*it]+0.37*not_repeat_words.size());
        else probability[*it]*=
        (Get_num_of_times_word_occurs(*it,tmp[j])+0.37)/(num_of_mood[*it]+0.37*not_repeat_words.size());
    }
    probability[*it]*=priori_probability[*it];
}
string ret;
double max_ret=0;
for(map<string,double>::iterator it=probability.begin();it!=probability.end();it++)//寻找最大概率
{
    if(it->second>max_ret)
    {
        max_ret=it->second;
        ret=it->first;
    }
}

```

```

//贝叶斯回归
double ret[6]={0,0,0,0,0,0};
for(int c=0;c<6;c++)//遍历每种情感
{
    for(int j=0;j<train.size();j++)//当前情感下遍历训练集所有文本
    {
        double tmp1=train[j].mood[c];//先验概率
        //遍历这个文本下每个训练集出现过的单词
        for(int k=0;k<tmp.size();k++)
        {
            tmp1*=train[j].laplace[tmp[k]];
        }
        ret[c]+=tmp1;
    }
}
double sum=0;
for(int j=0;j<6;j++) sum+=ret[j];
if(sum!=0) for(int j=0;j<6;j++) ret[j]/=sum;
for(int j=0;j<6;j++) out<<ret[j]<<"\t";
out<<endl;

```

## 4.创新点 & 优化(如果有)

kNN:

- 使用余弦相似度衡量距离，也就是向量夹角，不需要进行归一化，而且较欧氏距离来说，在这份训练集上的表现余弦要优秀一些。
- 使用tf\_idf值作为文本的特征向量，tf\_idf不仅能衡量某一特定文件内的高词语频率，也能衡量该词语在整个文件集合中的低文件频率，较one\_hot而言属性要优秀的多。
- 取出topK后使用距离进行加权，取权值较大者，不取众数。这样不会被数量众多但实际上与测试样本近似于正交的训练样本干扰，我只是进行了简单的相加运算作为加权，但实际上的一组余弦值为1的向量肯定要比两组余弦值为0.5的向量要准确的多的多，由于时间问题因此没有多加探索，只是实现了一个简单的思想，但饶是这样准确率也有4%的提升。
- 其实还有一些小优化，比如取出topK个标签之后，去除掉其中余弦值小于1e20的向量，也就是近似于正交的向量；排序使用stable\_sort而不用sort。但是在使用上面所说的加权的方法后，这些优化对结果的提升都很小了，因此没有在代码中写出，只在这里提及一下。

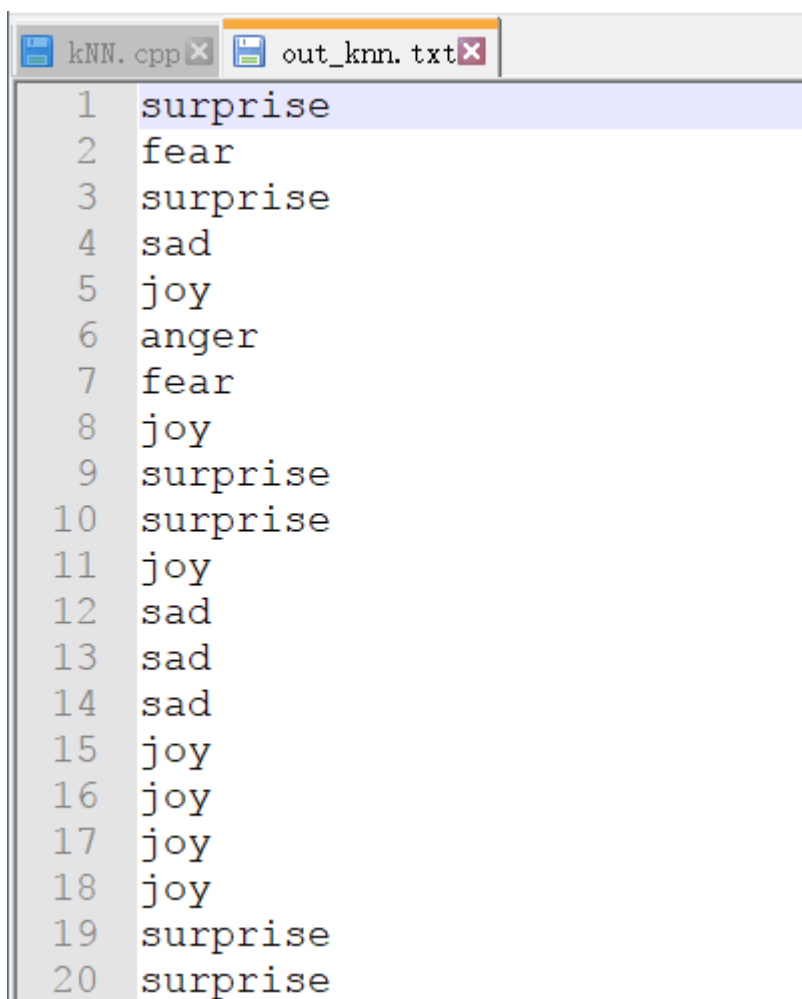
贝叶斯:

- 基于多项式模型，对我建立的模型寻找合适的alpha值，这里我采用的分别是0.37与0.0001（分类与回归），准确率和相关系数都有较大提升。

## 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）

kNN分类，取k=11，预测结果如下图所示



The image shows a code editor window with two tabs: 'kNN.cpp' and 'out\_knn.txt'. The 'out\_knn.txt' tab is active, displaying a list of 20 emotion labels, each preceded by a line number from 1 to 20. The labels are: surprise, fear, surprise, sad, joy, anger, fear, joy, surprise, surprise, joy, sad, sad, sad, joy, joy, joy, joy, surprise, and surprise. The first line is highlighted in light blue.

```
1 surprise
2 fear
3 surprise
4 sad
5 joy
6 anger
7 fear
8 joy
9 surprise
10 surprise
11 joy
12 sad
13 sad
14 sad
15 joy
16 joy
17 joy
18 joy
19 surprise
20 surprise
```

**kNN**回归，取 $k=11$ ，预测概率如图所示

kNN.cpp		rate11.txt				
1	0.0896983	0.100955	0.227858	0.218501	0.124772	0.238216
2	0.0616184	0.0211718	0.251058	0.248634	0.197197	0.220321
3	0.0536897	0.0328258	0.230663	0.21343	0.150016	0.319376
4	0.0865584	0.0667209	0.231172	0.109189	0.374371	0.131989
5	0.0369429	0.0265863	0.127448	0.398402	0.181554	0.229066
6	0.25834	0.0725724	0.20682	0.0326029	0.263191	0.166474
7	0.173082	0.0759794	0.229444	0.118474	0.225687	0.177334
8	0.0559259	0.06372	0.106693	0.497441	0.0825895	0.193631
9	0.0537391	0.0868339	0.202278	0.183107	0.20041	0.273631
10	0.0513095	0.0645575	0.176968	0.290307	0.155628	0.261229
11	0.0535228	0.00740045	0.151451	0.461543	0.0560925	0.269991
12	0.028188	0.0242119	0.255671	0.129785	0.340364	0.221779
13	0.157074	0.135692	0.11145	0.0671052	0.288428	0.24025
14	0.0264717	0.0208472	0.0721253	0.0428806	0.557183	0.280492
15	0.104862	0.0644184	0.164589	0.380457	0.144125	0.141549
16	0.0809696	0.0338758	0.069752	0.44709	0.110816	0.257497
17	0.199182	0.146739	0.110061	0.297695	0.0797217	0.166602
18	0.0154913	0.0118861	0.147635	0.493286	0.124064	0.207637
19	0	0	0.313369	0.0718491	0.154062	0.460721
20	0.0796876	0.144579	0.169039	0.220806	0.178184	0.207704
21	0.175324	0.158741	0.230192	0.113995	0.233959	0.0877884
22	0.0441031	0.05848	0.173136	0.274479	0.161183	0.288619
23	0.0277842	0.0282021	0.135719	0.405387	0.143998	0.258909
24	0.112028	0.0579414	0.18456	0.0885928	0.329094	0.227784
25	0.0720502	0.0205009	0.205808	0.281762	0.199768	0.220111
26	0.108043	0.0651978	0.056556	0.408404	0.0576203	0.304178
27	0.0942359	0.063007	0.151273	0.268013	0.254857	0.168614
28	0.0583076	0.0761195	0.0727573	0.333304	0.102347	0.357164
29	0.122329	0.0747778	0.194594	0.0484935	0.533021	0.0267834
30	0.104577	0.0414898	0.277032	0.0247956	0.435412	0.116694

贝叶斯分类，预测结果如下：



1	fear
2	fear
3	surprise
4	sad
5	joy
6	sad
7	fear
8	joy
9	surprise
10	joy
11	surprise
12	sad
13	sad
14	sad
15	joy
16	joy
17	joy
18	joy
19	surprise
20	surprise

贝叶斯回归，预测概率如下：

1	0.104034	0.0933526	0.223631	0.212501	0.18659	0.179891
2	0.0347918	0.0440512	0.196864	0.213321	0.130927	0.380045
3	0.0676884	0.0437504	0.259307	0.186882	0.178238	0.264135
4	0.0778795	0.0465549	0.212219	0.165912	0.333796	0.163638
5	0.0530091	0.0327964	0.17461	0.293294	0.231703	0.214587
6	0.11035	0.05818	0.12254	0.280627	0.190384	0.23792
7	0.172144	0.0816298	0.360817	0.0187661	0.277614	0.0890299
8	0.0869912	0.0869911	0.0434959	0.521648	0.108689	0.152185
9	0.00275679	0.0435416	0.248927	0.167677	0.0080555	0.529042
10	0.0795594	0.0784255	0.244658	0.105921	0.288738	0.202698
11	0.0250172	0.0110777	0.0553959	0.522918	0.0871164	0.298475
12	0.0265053	0.0442088	0.24785	0.0265053	0.557612	0.0973195
13	0.0908982	0.111731	0.0997678	0.0603677	0.371827	0.265408
14	0.000215272	0.000536581	0.000967257	0.000110409	0.788733	0.209437
15	0.0850032	0.0717617	0.193899	0.2554	0.201119	0.192818
16	1.89898e-007	1.07713e-007	5.90693e-007	0.738598	1.13385e-006	0.2614
17	0.111982	0.0500573	0.151153	0.306664	0.151856	0.228288
18	0.00657887	0.0038806	0.0855595	0.565397	0.176937	0.161647
19	0.035302	0.0215912	0.259229	0.149669	0.169771	0.364438
20	0.0275726	0.232511	0.0675935	0.188819	0.0706347	0.41287
21	0.133316	0.124413	0.222227	0.217812	0.137821	0.164411
22	2.40142e-005	2.08823e-005	0.0933609	0.546616	0.0800013	0.279977
23	0.0447323	0.0302366	0.043508	0.504482	0.118186	0.258856
24	0.0755198	0.164616	0.123692	0.103736	0.350523	0.181914
25	0.0328101	0.0092503	0.0600799	0.501771	0.0683213	0.327767
26	0.0966611	0.0637613	0.177711	0.214591	0.233256	0.21402
27	0.161233	0.107839	0.238099	0.0289408	0.34119	0.122698
28	0.0355692	0.0334836	0.0402799	0.237394	0.0467676	0.606506
29	0.0970649	0.0965051	0.206559	8.86281e-006	0.586959	0.0129024

## 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）

**kNN**分类，取 $k=1\sim 50$ ，计算准确率，可以看出 $k=11$ 时达到峰值：

```
k=1  rate:0.430868
k=2  rate:0.430868
k=3  rate:0.401929
k=4  rate:0.434084
k=5  rate:0.450161
k=6  rate:0.446945
k=7  rate:0.463023
k=8  rate:0.485531
k=9  rate:0.475884
k=10 rate:0.466238
k=11 rate:0.488746
k=12 rate:0.475884
k=13 rate:0.488746
k=14 rate:0.482315
k=15 rate:0.485531
k=16 rate:0.472669
k=17 rate:0.469453
k=18 rate:0.469453
k=19 rate:0.466238
k=20 rate:0.472669
k=21 rate:0.463023
k=22 rate:0.463023
k=23 rate:0.456592
k=24 rate:0.453376
k=25 rate:0.463023
k=26 rate:0.466238
k=27 rate:0.463023
k=28 rate:0.459807
k=29 rate:0.463023
k=30 rate:0.466238
k=31 rate:0.469453
k=32 rate:0.466238
k=33 rate:0.459807
k=34 rate:0.456592
k=35 rate:0.466238
k=36 rate:0.466238
k=37 rate:0.466238
k=38 rate:0.456592
k=39 rate:0.459807
k=40 rate:0.459807
k=41 rate:0.459807
k=42 rate:0.453376
k=43 rate:0.453376
k=44 rate:0.453376
k=45 rate:0.459807
k=46 rate:0.459807
k=47 rate:0.459807
k=48 rate:0.459807
k=49 rate:0.463023
k=50 rate:0.463023
```

**kNN**回归，同样取 $k=11$ ，相关系数如下：

	anger	disgust	fear	joy	sad	surprise
r	0.391303469	0.348815551	0.450006654	0.449390068	0.437341144	0.42323656
average	0.416682241					
evaluation	低度相关 666					

贝叶斯分类，准确率如下：

rate: 0.469453

贝叶斯回归，相关系数如下：

	anger	disgust	fear	joy	sad	surprise
r	0.318081265	0.248435142	0.35648226	0.404801925	0.398496613	0.364529188
average	0.348471066					
evaluation	低度相关 666					

## 四、思考题

1.同一测试样本的各个情感概率总和应该为1 如何处理？

答：进行归一化处理，对所有概率求和后计算每个概率的占比即可。

2.在矩阵稀疏程度不同的时候，曼哈顿距离与欧氏距离表现有什么区别，为什么？

答：个人认为没有区别，也可能是我不知道。

3.伯努利模型和多项式模型分别有什么优缺点？

答：多项式模型较伯努利模型而言，能更好的衡量单词出现次数对模型的影响。伯努利模型只计算单词有没有出现，所以文本中重复出现的单词与只出现过一次的单词在伯努利模型中影响相同；而多项式模型计算单词出现次数，那么只出现过一次的单词相较重复出现的单词没有优势，因此就这方面而言多项式模型预测更准确。但是相应的，伯努利模型实现起来要比多项式模型容易一些。

4.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？

答：之前全词典中都没有出现这个词，那么这个词在基于训练集得到的模型中是无法被预测的，应该将这个词从测试集中删去。