机器学习第二次实验报告——朴素贝叶斯分类器

March 28, 2018

1 问题描述

本次实验的目标为用朴素贝叶斯算法根据训练数据实现对于测试数据的分类，同时根据预测结果给出分类器在测试数据上的正确率。同时，实验要求对于结果利用拉普拉斯平滑的方法进行改进，以免对于一些测试数据出现概率为零的情况。

2 解决方案

2.1 原理介绍

贝叶斯算法主要是通过计算样本的后验概率，从而推断出样本属于哪一类别，即后验概率最大的类别即为样本预测的类别。

根据朴素贝叶斯公式，样本的后验概率可以表示为：

$$P(C_i|X) = \frac{P(C_i) * P(X|C_i)}{P(X)}$$

根据假设条件，各个样本出现的概率认为是相等的，而 $P(X|C_i)$ 可以进一步表示为

$$P(X|C_i) = P(x_1|C_i) * P(x_2|C_i) * ...P(x_K|C_i)$$

因此有

$$P(C_i|X) \propto P(x_1|C_i) * P(x_2|C_i) * ...P(x_K|C_i) * P(C_i)$$

而 $P(x_n|C_i)$ 的值可以根据样本分布进行估计，经过归一化后可以求出样本不同类别的后验概率。

2.2 朴素贝叶斯算法具体实现

2.2.1 数据预处理

为方便分析训练集数据的统计特性，首先对于原始数据进行预处理。具体方法为将原始数据按照类被划分，分别提取不同类别的数据在不同维度上的统计特征。对于连续型数据，由于后续需要根据高斯公式计算后验概率，因此需要对计算时的参数进行参数估计，这里采用极大似然估计的方式进行参数估计。根据贝叶斯公式：

$$P(\theta|X) = \frac{P(\theta) * P(X|\theta)}{P(X)}$$

在极大似然估计中，假设 $P(\theta)$ 和 $P(X)$ 是与 θ 无关的, 因此使 $P(\theta|X)$ 最大的 θ 即为使 $P(X|\theta)$ 最大的 θ, 根据高斯公式，推导出估计的均值和方差分别为

$$\mu = \frac{1}{n} * \sum_{i=1}^{n} x_i$$

$$\sigma^2 = \frac{1}{n} * \sum_{i=1}^{n} (x_i - \bar{x})^2$$

根据上述公式对样本数据进行参数估计，进而求得后验概率。

首先定义函数 createdData($filename$), 将原始数据按照类别分类。

```
def createData(filename):
    with open(filename) as train_f:
        train_data=list(csv.reader(train_f))
    #devide the train_data by classes
    separated ={}
    for i in range(1,len(train_data)):
        vector = train_data[i]
        # The label for the sample
```

```
        class_num=int(vector[-1])
        #The first sample in a class
        if (class_num not in separated):
            separated[class_num] = []
        separated[class_num].append(vector)
    #for each class, create data on each demension
    separated_data={}
    for class_num in separated:
        separated_data[class_num]={}
        for demen_num,demen_name in enumerate(train_data[0]):
            #print (type(demen_name))
            if demen_name!='Label' and demen_name!='No.':
                separated_data[class_num][demen_name]=np.array(separated[class_num])[:,demen_num]
    return len(train_data),separated_data
```

首先定义字典变量 *separated*，键为样本的类别标签 *class_num*，值为该类别的样本组成的数组。再定义字典变量 *separated_data*，键为样本的类比标签 *class_num*，值为以维度 *demen_name* 为键，该维度的所有数据为值的字典。经过预处理后，可以通过 *separated_data*[*class_num*][*demen_name*] 获取样本中类别标签为 *class_num*, 特征为 *demen_name* 的数据。

再定义函数 pro_data(*train_data*) 提取数据特征。

```
def pro_data(train_data):
    pro_data={}
    for class_num in train_data.keys():
        pro_data[class_num]={}
        for demen_name in train_data[class_num].keys():
            #print (demen_name)
            pro_data[class_num][demen_name]={}
            #load the raw data
            sample_data= train_data[class_num][demen_name]
            #print (type(sample_data))
            temp_data=[]

            #print (temp_data)
            # for continuous data  calculate the mean value and standard deviation
            if demen_name=='Density' or demen_name=='SugerRatio':
                #convert the type from string to float
                temp_data=sample_data.astype('float')
                #calculate the mean value
                mean_val=np.mean(np.array(temp_data))
                pro_data[class_num][demen_name]['mean']=mean_val
                std_val=np.std(np.array(temp_data))
                #calculate the standard deviation
                pro_data[class_num][demen_name]['std']=std_val
                pro_data[class_num][demen_name]['sample']=temp_data
            else:
                temp_data=sample_data.astype('int')
                pro_data[class_num][demen_name]['sample']=temp_data
    return pro_data
```

对于每个特征上的所有样本数据，首先调用函数 astype('*float*') 或 astype('*int*') 将其转化为整型或浮点型。对于连续的数据特征 *Density* 和 *SugerRatio*，用 np.mean 和 np.std 分别计算数据估计出的均值和标准差。

经过预处理后，将数据输出：

```
The feature of the data in class 0 are :
   Color :
                sample [2 1 3 3 1 3 2 3 1]
   SugerRatio :
                std 0.10162980509649036
                sample [0.091 0.267 0.057 0.099 0.161 0.198 0.37  0.042 0.103]
                mean 0.1542222222222222
   Touch :
                sample [1 2 1 2 1 1 2 1 1]
   Root :
                sample [2 3 3 1 2 2 2 1 1]
   Texture :
                sample [1 2 3 3 2 2 1 3 2]
   Knocks :
                sample [2 3 3 1 1 2 1 1 2]
   Umbilicus :
                sample [2 3 3 3 1 1 2 3 2]
   Density :
                std 0.18358252424965876
                sample [0.666 0.243 0.245 0.343 0.639 0.657 0.36  0.593 0.719]
                mean 0.4961111111111117
The feature of the data in class 1 are :
   Color :
                sample [1 2 2 1 3 1 2 2]
   SugerRatio :
                std 0.09440570692495238
                sample [0.46  0.376 0.264 0.318 0.215 0.237 0.149 0.211]
                mean 0.27875
   Touch :
                sample [1 1 1 1 1 2 2 1]
   Root :
                sample [1 1 1 1 1 2 2 2]
   Texture :
                sample [1 1 1 1 1 1 2 2]
   Knocks :
                sample [1 2 1 2 1 1 1 1]
   Umbilicus :
                sample [1 1 1 1 1 2 2 2]
   Density :
                std 0.12086536931644233
                sample [0.697 0.774 0.634 0.608 0.556 0.403 0.481 0.437]
                mean 0.57375
```

2.2.2 计算条件概率

提取训练集数据特征后，定义函数 cal_prob($file\_name, train\_data$) 计算测试集数据属于不同类别标签的条件概率。

```python
def cal_prob(file_name,train_data):
    with open(file_name) as test_f:
        test_data=list(csv.reader(test_f))
    total_corr= 0
    print (test_data)
    for i in range(1,len(test_data)):
        class_prob={}
        pred_class=0
        pred_prob=0
        #Calculate the possibility for each class
        for class_num in train_data.keys():
            res=1

            #Calculate the likelihood on each demension
            for demen_num,demen_name in enumerate(test_data[0]):
                if demen_name!='Label' and demen_name!='No.':
                    x = float(test_data[i][demen_num])
                    #When the data follows continuous distribution
                    if demen_name=='Density' or demen_name=='SugerRatio' :
                        #Find the data in training dataset for the class on this demension
                        mean= train_data[class_num][demen_name]['mean']
                        std=train_data[class_num][demen_name]['std']
                        exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(std,2))))
                        prob = (1 / (math.sqrt(2*math.pi) * std)) * exponent
                        res=res*prob
                    #When the data follows discrete distribution
                    else:
                        x = int(test_data[i][demen_num])
                        sample_data = train_data[class_num][demen_name]['sample']
                        is_equal=(sample_data==x)
                        prob = sum(is_equal.astype(int))/len(sample_data)
                        res = res * prob
            temp_prob=res*(len(train_dataset[class_num])/total_trainnum)
            class_prob[class_num]=temp_prob
```

```
            if(temp_prob>pred_prob):
                pred_prob = temp_prob
                pred_class = class_num
        if pred_class==int(test_data[i][-1]):
            total_corr+=1
    return total_corr/(len(test_data)-1)
```

对于测试集的每个样本 i，由贝叶斯公式可知，其在不同类别标签 $class\_num$ 下的概率正比于该类别的概率与该类别的条件下不同特征维度上测试样本数据出现的概率的乘积。对于每个类别标签 $class\_num$ 下的每个特征 $demen\_name$, 若该特征是连续的，根据训练样本在该特征下的均值 $mean$ 和方差 std, 根据高斯公式

$$P(x_i|y_k) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

计算出条件概率 $prob$。若特征是非连续的，首先通过矩阵的逻辑运算 $is\_equal = (sample\_data == x)$ 得到样本数据与测试数据是否相等的布尔矩阵，将该矩阵转换为数值矩阵求和后除以样本数量即可得到对应的条件概率 $prob$。对所有特征的条件概率累乘后，还要乘以此类别标签的概率即可得到该测试样本属于此类别标签的概率 (未归一化)。

各样本数据未归一化的后验概率为：

```
{0: 3.665180426076142e-05, 1: 0.03606615594376714}
{0: 0.0008581798773291161, 1: 0.0}
{0: 0.013351340700013266, 1: 0.024981771443384718}
{0: 0.006516473953368407, 1: 0.10057479304436875}
{0: 0.0008581798773291161, 1: 0.0}
{0: 0.0003277297680532618, 1: 0.008042932562393766}
{0: 0.0058449185425527945, 1: 0.0021930254760241202}
{0: 0.0017293659872335099, 1: 0.445647434789868}
{0: 0.0010478951262823698, 1: 0.11251728902050441}
```

为进一步预测样本类别并计算正确率，对每一个测试数据未归一化的最大的后验概率值对应的类别标签即为其预测的类别，与测试样本的实际类别比较后即可得到正确预测的数量，除以测试样本总数即为正确率。

2.3 实验改进

对于以上输出的为未进行归一化的后验概率，部分类别标签出现了为 0 的情况。这种情况产生的原因主要是对于离散变量，若测试样本的数据在所有训练样本中都没有出现过，则计算出的概率为零。这实际上是不合理的，样本中没有出现的数据不代表实际不会出现，因此用拉普拉斯平滑的方法对分类器进行改进。

引入的拉普拉斯平滑的公式如下：

$$P(X^j = a_{jl}|y = c_k) = \frac{\sum_{i=1}^{N} I(X_i^j = a_{jl}, y_i = c_k) + \lambda}{\sum_{i=1}^{N} I(y_i = c_k) + L_j * \lambda}$$

$$P(y = c_k) = \frac{\sum_{i=1}^{N} I(y_i = c_k) + \lambda}{N + k * \lambda}$$

其中 L_j 表示第 j 维特征的所有选择个数，a_jl 表示第 j 维特征的第 l 个选择，而 k 代表所有类别标签数，λ 是拉普拉斯平滑因子，通常取 1。

引入拉普拉斯平滑后，为计算测试集数据属于不同类别标签的条件概率，首先需要计算训练数据在不同特征上的特征数量。为此定义函数 cal_feanum($file\_name$)，该函数的返回值是一个字典，键为特征名称，值为训练数据在该特征上的特征数。

```
def cal_feanum(file_name):
    res={}
    with open(file_name) as train_f:
        train_data=np.array(list(csv.reader(train_f)))
    for demen_num,demen_name in enumerate(train_data[0]):
        if demen_name!='Label' and demen_name!='No.'and demen_name!='Density'and demen_name!='SugerRati
            all_sample = train_data[1:,demen_num]
            #print(all_sample)
            res[demen_name]= np.unique(all_sample).shape[0];
    return res;
```

对于每一个特征，变量 all_sample 表示训练集在该特征上的所有样本数据，np.unique(all_sample) 返回 all_sample 所有不重复的元素，其长度即表示训练数据在该特征上的样本数量。

该函数的输出结果为：

```
Root    3
Color    3
Umbilicus    3
Touch    2
Knocks    3
Texture    3
```

然后修改原函数，当数据在特征上离散分布时，将概率修改为：

```
prob = sum(is_equal.astype(int))+1/(len(sample_data)+fea_num[demen_name])
```

最后在该类别上的后验概率为：

```
temp_prob = res*((len(train_dataset[class_num])+1)/(total_trainnum+len(train_data.keys())))
```

经过拉普拉斯平滑后，训练数据集在各类别上的未归一化的概率为：

```
{0: 4.203038088311189e-05, 1: 0.018122286074940586}
{0: 0.0010544100031711996, 1: 3.2930362518104097e-06}
{0: 0.011961423741612838, 1: 0.01867957897475863}
{0: 0.007005705989422777, 1: 0.07580424599393631}
{0: 0.0010544100031711996, 1: 3.2930362518104097e-06}
{0: 0.000402667966123624, 1: 0.0060139287691576395}
{0: 0.0052959196240495235, 1: 0.002023966089681861}
{0: 0.0022310402358140247, 1: 0.20993063191337416}
{0: 0.0012817830113131322, 1: 0.07269036587997696}
```

2.4 实验结果

拉普拉斯平滑前的结果为：

```
The prediction result of the 1th data is 1
The prediction result of the 2th data is 0
The prediction result of the 3th data is 1
The prediction result of the 4th data is 1
The prediction result of the 5th data is 0
The prediction result of the 6th data is 1
The prediction result of the 7th data is 0
The prediction result of the 8th data is 1
The prediction result of the 9th data is 1
```

对应预测精度为：88.89%
拉普拉斯平滑后的结果为：

```
The prediction result of the 1th data is 1
The prediction result of the 2th data is 0
The prediction result of the 3th data is 1
The prediction result of the 4th data is 1
The prediction result of the 5th data is 0
The prediction result of the 6th data is 1
The prediction result of the 7th data is 0
The prediction result of the 8th data is 1
The prediction result of the 9th data is 1
```

对应预测精度为：88.89%

3 实验分析

计算实例的概率时，如果某个量在训练集中没有出现过，会导致整个实例的概率结果是 0，但这显然是不合理的，因为在训练集中没有出现过的数据不代表实际不会出现。

拉普拉斯平滑提出用加 1 的方法估计没有出现过的现象的概率，这样如果训练样本集数量充分大时，并不会对结果产生影响，并且解决了后验概率为 0 的问题，使得实验结果更加准确。