

2016~2017春季学期机器学习概论

朴素贝叶斯分类器实验

实验报告

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

计43

唐玉涵

2014011328

2016~2017春季学期机器学习概论

朴素贝叶斯分类器实验

实验报告

1、任务说明

在本实验中，我们需要完成一个朴素贝叶斯分类器，并且在一组真实的数据集上测试它的分类性能。具体包括以下几个部分：

1. 在真实数据集上实现一个机器学习的具体算法（特指朴素贝叶斯算法），并且改善它的性能；
2. 为其性能给出合理的评价方式；
3. 分析实验结果。

2、实验设计

我采用自己比较熟悉的java编程语言进行实验，与python相比可能运行时间较快一些。共编写两个类：Sample.java 和 Test.java。Sample是为训练数据和测试数据编写的类，Test为利用朴素贝叶斯方法实现的分类器实现与测试。以下分别对其进行介绍：

Sample类

针对数据集的14个特征属性和1个结果属性，我采用了以下的结构对其进行存储：

```
public class Sample {
    int age;
    String workclass;
    int fnlwgt;
    String education;
    int education_num;
    String marital_status;
    String occupation;
    String relationship;
    String race;
    String sex;
    int capital_gain;
    int capital_loss;
    int hours_per_week;
    String native_country;
    String result;
    String my_result;
    public Sample(){}
    public Sample(int age, String workclass, int fnlwgt, String education,
        int education_num, String marital_status, String occupation,
        String relationship, String race, String sex, int capital_gain, int capital_loss,
        int hours_per_week, String native_country, String result){
        this.age=age;
        this.fnlwgt = fnlwgt;
        this.workclass=workclass;
        this.education=education;
        this.education_num=education_num;
        this.marital_status=marital_status;
        this.occupation=occupation;
        this.relationship=relationship;
        this.race=race;
        this.sex=sex;
        this.capital_gain=capital_gain;
        this.capital_loss=capital_loss;
        this.hours_per_week=hours_per_week;
        this.native_country=native_country;
        this.result=result;
    }
}
```

其中Sample的前15个属性与数据集中属性相对应，最后一个属性my_result为测试集所准备，用来存放对测试数据的预测结果，与result属性进行比较即可对性能作出评价。

需要说明的是，我针对每一个属性都编写了set函数和get函数，这样不直接对Sample类进行取值和修改，符合类的开闭原则，见下图：

```
public int getAge() { return age; }
public void setAge(int age) { this.age = age; }
public String getWorkclass() { return workclass; }
public void setWorkclass(String workclass) { this.workclass = workclass; }
public int getFnlwgt() { return fnlwgt; }
public void setFnlwgt(int fnlwgt) { this.fnlwgt = fnlwgt; }
public String getEducation() { return education; }
public void setEducation(String education) { this.education = education; }
public int getEducation_num() { return education_num; }
public void setEducation_num(int education_num) { this.education_num = education_num; }

public String getMarital_status() { return marital_status; }
public void setMarital_status(String marital_status) { this.marital_status = marital_status; }
public String getOccupation() { return occupation; }
public void setOccupation(String occupation) { this.occupation = occupation; }

public String getRelationship() { return relationship; }
public void setRelationship(String relationship) { this.relationship = relationship; }
public String getRace() { return race; }
public void setRace(String race) { this.race = race; }
public String getSex() { return sex; }
public void setSex(String sex) { this.sex = sex; }
public int getCapital_gain() { return capital_gain; }
public void setCapital_gain(int capital_gain) { this.capital_gain = capital_gain; }
public int getCapital_loss() { return capital_loss; }
public void setCapital_loss(int capital_loss) { this.capital_loss = capital_loss; }
public int getHours_per_week() { return hours_per_week; }
public void setHours_per_week(int hours_per_week) { this.hours_per_week = hours_per_week; }
public String getNative_country() { return native_country; }
public void setNative_country(String native_country) { this.native_country = native_country; }
public String getResult() { return result; }
public void setResult(String result) { this.result = result; }
public String getMy_result() { return my_result; }
public void setMy_result(String my_result) { this.my_result = my_result; }
```

Test类

对训练集和测试集分别采用ArrayList<Sample>的形式进行储存，命名为list和list1。读入训练集数据文件后，逐行扫描并对每行按照逗号分割开存入一个Sample对象中（list中）。

针对测试集，采用同样的方式将数据存入list1中。针对list1中的每一个测试对象，逐个扫描list中的训练对象，首先判断训练对象结果属于“>50K.”还是“<=50K.”类别，然后在这两类下分别逐个扫描训练对象的14个特征属性，统计在该类别下与测试对象各个属性相同的对象个数。同时记录训练集中“>50K.”和“<=50K.”这两类的个数。

使用上面统计的数目，利用贝叶斯公式计算该测试对象分属“>50K.”和“≤50K.”的概率（相对值，其和并不为1），相对大的那个结果即记为其预测结果。所依据的原理如下：

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y)$$

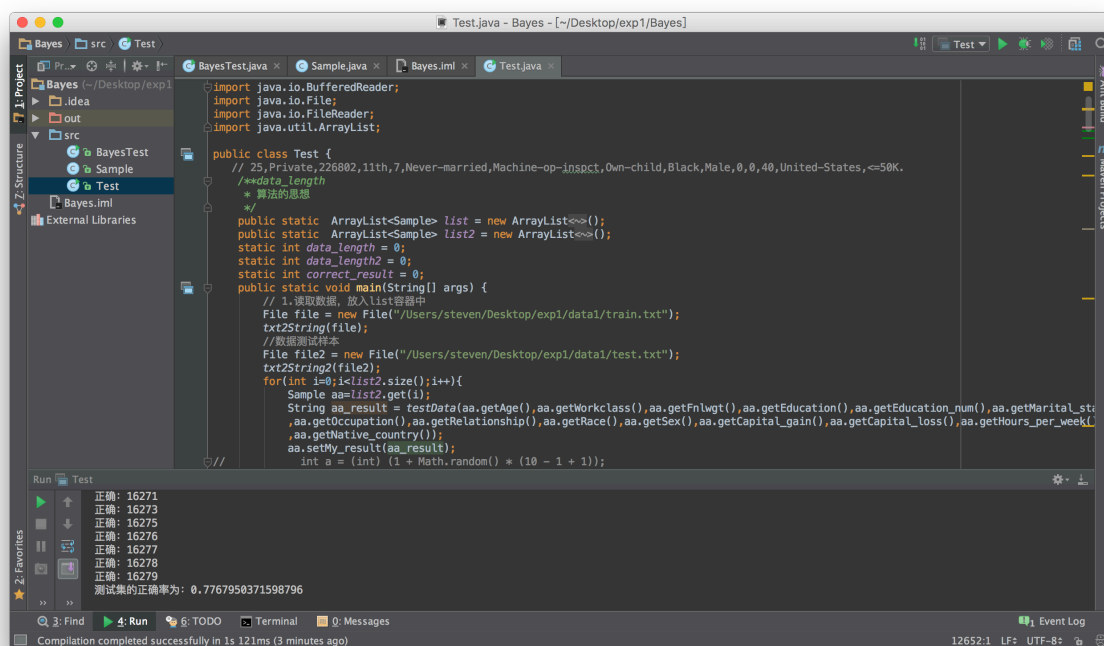
为了对分类的性能做出评价，我才用了准确率作为衡量指标，即

$$\bullet \text{ Accuracy} = \frac{\text{number of correctly classified records}}{\text{number test records}}$$

统计针对全体测试集数据的正确预测个数，除以测试集大小，即可得到该算法的准确率。

3、实验结果

如上述方法实现的朴素贝叶斯分类器（未经过优化），得到的准确率约为77.67%，见下图：



The screenshot shows an IDE window titled "Test.java - Bayes - [~/Desktop/exp1/Bayes]". The code defines a `Test` class with a `main` method. It reads training data from `data1/train.txt` and test data from `data1/test.txt`. The `main` method iterates over the test data, uses the `testData` method to get a prediction, and compares it with the actual result. The output shows 8 correct predictions out of 10 test records, resulting in an accuracy of 0.7767950371598796.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;

public class Test {
    // 25,Private,226802,11th,7,Never-married,Machine-op-inspct,Own-child,Black,Male,0,0,40,United-States,<=50K.
    /**data_length
     * 算法的思想
     */
    public static ArrayList<Sample> list = new ArrayList<>();
    public static ArrayList<Sample> list2 = new ArrayList<>();
    static int data_length = 0;
    static int data_length2 = 0;
    static int correct_result = 0;
    public static void main(String[] args) {
        // 1.读取数据，放入list容器中
        File file = new File("/Users/steven/Desktop/exp1/data1/train.txt");
        txt2String(file);
        //数据测试样本
        File file2 = new File("/Users/steven/Desktop/exp1/data1/test.txt");
        txt2String2(file2);
        for(int i=0;i<list2.size();i++){
            Sample aa=list2.get(i);
            String aa_result = testData(aa.getAge(),aa.getWorkclass(),aa.getFnlwgt(),aa.getEducation(),aa.getEducation_num(),aa.getMarital_status(),aa.getOccupation(),aa.getRelationship(),aa.getRace(),aa.getSex(),aa.getCapital_gain(),aa.getCapital_loss(),aa.getHours_per_week(),aa.getNative_country());
            aa.setMy_result(aa_result);
            int a = (int) (1 + Math.random() * (10 - 1 + 1));
        }
    }
}
```

Run Test

正确: 16271
正确: 16273
正确: 16275
正确: 16276
正确: 16277
正确: 16278
正确: 16279
测试集的正确率为: 0.7767950371598796

可见数据的分类性能比较普通。由于训练集中“ $\leq 50K$.”的数据比例占到了76.07%，与之相对应的若直接预测测试集数据全部为“ $\leq 50K$.”也可得到76%左右的准确率，提升空间比较大。

同时，我将过程中每一个测试对象分属“ $> 50K$.”和“ $\leq 50K$.”的概率打印出来，惊奇地发现其中有很多两个概率均为0。这时我的算法将其判断为“ $\leq 50K$.”，即类似于上段所述直接预测得结果的方法，准确率也比较相近。研究其原因，发现是“零概率”问题造成的，针对其以及其他部分的优化详见下一部分。

4、实验优化与分析

1. 训练集大小的影响

实验所给出的全部训练集合大小为32561条数据，集合数目较大，下对训练集的大小对分类性能的影响进行研究：

分别使用5%，50%和100%的训练集数据来进行训练，5%和50%的数据均为从训练集中随机产生，并各测试5组，求其中最大、最小和平均准确率。见下表：

	5%	50%
1	81.47%	81.75%
2	82.20%	81.24%
3	80.83%	81.61%
4	82.59%	81.60%
5	81.71%	81.65%
最大	82.59%	81.75%
最小	80.83%	81.24%
平均	81.76%	81.57%

作为对比，100%的训练数据集合准确率为81.68%（加入了下面几个分点所述优化，故比76.07%提高了不少）。

分析：

从准确率的结果上看，5%的数据集合优于100%的数据集合优于50%的数据集合，最好的准确率甚至达到了82.59%，但是结果的波动较大，不很稳定。

可能的解释是朴素贝叶斯并不因更多的数据而执行的更好。朴素贝叶斯算法需要足够的数据来了解每个属性的概率关系，这些属性独立于输出变量。而模型忽略给定属性间的交互关系，我们不需要这些有交互关系的样本，因此通常情况下比其他算

法需要更少的数据。更进一步，它不会因小规模样本而产生过拟合的数据。所以如果没有太多的训练数据，朴素贝叶斯算法也许是比较好的选择。

2. 零概率问题

上一部分已发现算法中存在大量的“零概率”问题，这种问题出现的原因是：很多测试集中的数据的某一属性在训练集中可能并没有（一个重要的例子是，一个分类属性具有数值，然而却没有出现在训练集中。这种情况下，模型会赋0概率并且不能够进行预测），即有下式：

$$\hat{P}(y = c | x_1, \dots, x_i = k, \dots, x_n) = 0$$

这显然并不合理，将直接导致测试对象概率为0，无法给出比较合理的预测结果。

优化方法：采用拉普拉斯平滑方法对数据进行处理：

$$\hat{P}(x_i = k | y = c) = \frac{\#\{y=c, x_i=k\} + \alpha}{\#\{y=c\} + M\alpha}$$

上式子中 α 取1，即为拉普拉斯平滑，其原理为假定训练样本很大时，每个分量 x 的计数加1造成的估计概率变化可以忽略不计，但可以方便有效的避免零概率问题。

这样对概率计算进行平滑处理后，实验结果的准确率得到了很大的提升，由76.07%提升为81.07%。

其他平滑处理的方式限于时间原因，未进行尝试，但拉普拉斯算法在理论上可靠，对原计算改动不大，实际效果也比较好，推荐作为处理时的首选。

3. 连续属性和缺失属性的处理

连续属性：

针对6个连续的数值型特征属性，我才用了分类离散法和高斯分布估值法。分类法的效果较好，实验结果的准确率可达到81.68%，而用高斯分布估值法反而准确率有所下降，只有77.87%。

缺失属性：

针对缺失属性，我采用了三种处理方法，直接去掉训练集中带有缺失属性的数据：准确率81.09%；将其分为一类：准确率81.68%；用该特征出现最多的代替它：准确率81.62%。