School of Data and Computer Science, Sun Yat-sen University

E11 Naive Bayes (C++/Python)

18340013 Conghao Chen

November 24, 2020

Contents

1 Datasets

The UCI dataset (`http://archive.ics.uci.edu/ml/index.php`) is the most widely used dataset for machine learning. If you are interested in other datasets in other areas, you can refer to `https://www.zhihu.com/question/63383992/answer/222718972`.

Today's experiment is conducted with the **Adult Data Set** which can be found in `http://archive.ics.uci.edu/ml/datasets/Adult`.

Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	1305515

You can also find 3 related files in the current folder, `adult.name` is the description of **Adult Data Set**, `adult.data` is the training set, and `adult.test` is the testing set. There are 14 attributes in this dataset:

>50K, <=50K.

1. age: continuous.
2. workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. fnlwgt: continuous.
4. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 5. 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5. education-num: continuous.
6. marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
8. relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. sex: Female, Male.

11. capital−gain: continuous.

12. capital−loss: continuous.

13. hours−per−week: continuous.

14. native−country: United−States, Cambodia, England, Puerto−Rico, Canada, Germany, Outlying−US(Guam−USVI−etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican−Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El−Salvador, Trinadad&Tobago, Peru, Hong, Holand−Netherlands.

Prediction task is to determine whether a person makes over 50K a year.

2 Naive Bayes

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that **the value of a particular feature is independent of the value of any other feature**, given the class variable.

For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

$$P(y \mid x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n \mid y)}{P(x_1, ..., x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i \mid y, x_1, ..., x_{i-1}, x_{x+1}, ..., x_n) = P(x_i \mid y)$$

, for all i, this relationship is simplified to

$$P(y \mid x_1, ..., x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, ..., x_n)}$$

Since $P(x_1, ..., x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

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

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$, the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

- When attribute values are discrete, $P(x_i \mid y)$ can be easily computed according to the training set.
- When attribute values are continuous, an assumption is made that the values associated with each class are distributed according to Gaussian i.e., Normal Distribution. For example, suppose the training data contains a continuous attribute x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_k be the mean of the values in x associated with class y_k, and let σ_k^2 be the variance of the values in x associated with class y_k. Suppose we have collected some observation value x_i. Then, the probability distribution of x_i given a class y_k, $P(x_i \mid y_k)$ can be computed by plugging x_i into the equation for a Normal distribution parameterized by μ_k and σ_k^2. That is,

$$P(x = x_i \mid y = y_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}}$$

3 Task

- Given the training dataset `adult.data` and the testing dataset `adult.test`, please accomplish the prediction task to determine whether a person makes over 50K a year in `adult.test` by using Naive Bayes algorithm (C++ or Python), and compute the accuracy.
- Note: keep an eye on the discrete and continuous attributes.
- Please finish the experimental report named `E11_YourNumber.pdf`, and send it to `ai_2020@foxmail.com`

4 Codes

```
import os, sys, time
import pandas as pd
import numpy as np

attrlist = {"age":0, "workclass":1, "fnlwgt":0, "education":1,
    "education-num":0, "marital-status":1, "occupation":1,
    "relationship":1, "race":1, "sex":1, "capital-gain":0,
    "capital-loss":0, "hours-per-week":0, "native-country":1,
    "salary":0} # 0: 连续, 1: 离散

traindata =
    pd.read_csv("adult.data",names=attrlist.keys(),index_col=False)
testdata =
    pd.read_csv("adult.test",names=attrlist.keys(),index_col=False,
header=0)

def preprocessing(data):
    attributes = list(attrlist.keys())
    # 删除一些影响较小属性
    attributes.remove("fnlwgt")
    attributes.remove("capital-gain")
    attributes.remove("capital-loss")
    return data[attributes]

def fulldata(data):
    # 填充缺失数据，方法是选择该列出现最多的数据填入该位置。
    for a in data.columns.values:
        if attrlist[a]:  # 离散
            data.loc[data[a] == " ?", a] =
                data[a].value_counts().argmax()  # 众数
        else:  # 连续则跳过
            pass
```

```
    return data

# 处理数据
traindata = preprocessing(traindata)
testdata = preprocessing(testdata)
traindata = fulldata(traindata)
testdata = fulldata(testdata)

class NB():
    # 贝叶斯分类器实现
    def __init__(self,traindata,attrlist):
        self.traindata = traindata
        self.attrlist = attrlist
        # 计算概率P(x_i|y)
        self.prob = {}
        self.prob[" >50K"] =
            traindata["salary"].value_counts(normalize=True)[" >50K"]
        self.prob[" <=50K"] = 1 - self.prob[" >50K"]
        self.attributes =
            traindata.columns.values[traindata.columns.values !=
            "salary"]
        leq = traindata[traindata["salary"] == " <=50K"]
        geq = traindata[traindata["salary"] == " >50K"]
        for a in self.attributes:
            if self.attrlist[a]: # 离散
                numofleq = leq[a].value_counts()
                numofgeq = geq[a].value_counts()
                N = len(traindata[a].unique())
                for xi in traindata[a].unique():
                    # 拉普拉斯平滑处理
                    self.prob[(xi," <=50K")] = (numofleq.get(xi,0) + 1)
                        / (len(leq) + N)
                    self.prob[(xi," >50K")] = (numofgeq.get(xi,0) + 1)
```

```
                    / (len(geq) + N)
            else: # 连续情况则用高斯分布
                muofleq = np.mean(leq[a])
                sigmaofleq = np.var(leq[a])
                self.prob[(a," <=50K")] = lambda x:
                    np.exp(-(x-muofleq)**2/(2*sigmaofleq)) /
                    np.sqrt(2*np.pi*sigmaofleq)
                muofgeq = np.mean(geq[a])
                sigmaofgeq = np.var(geq[a])
                self.prob[(a," >50K")] = lambda x:
                    np.exp(-(x-muofgeq)**2/(2*sigmaofgeq)) /
                    np.sqrt(2*np.pi*sigmaofgeq)

    def predict(self,testdata):
        # 预测
        accuracy = 0
        for i, row in testdata.iterrows():
            # 遍历行数据，计算概率P(y|x1,...,xn)
            prod = np.array([self.prob[" <=50K"],self.prob[" >50K"]])
            for a in self.attributes:
                xi = row[a]
                if self.attrlist[a]: # 离散
                    prod[0] *= self.prob[(xi," <=50K")]
                    prod[1] *= self.prob[(xi," >50K")]
                else: # 连续
                    prod[0] *= self.prob[(a," <=50K")](xi)
                    prod[1] *= self.prob[(a," >50K")](xi)

            # 找到最大概率的那一个
            if prod.argmax() == 0:
                catagory = " <=50K"
            else:
                catagory = " >50K"
```

```
            if catagory == row["salary"][:-1]:
                accuracy += 1

        accuracy /= len(testdata)
        print("Accuracy: {:.2f}%".format(accuracy * 100))
        return accuracy
start = time.time()
nb = NB(traindata, attrlist)
nb.predict(testdata)
end = time.time()
print('Runtime: ' + str(end-start) + 's')
```

5 Results

该程序前面预处理数据部分与上一次的 DT 一模一样;Bayes 分类器的代码也有标注释, 比较容易理解. 程序运行时间平均在 3 秒左右. 训练后的结果如图：

```
C:\Users\czh\.conda\envs\Pycharm\python.exe "D:/Pycharm/PyCharm 2020.2.2/NB.py"
Accuracy: 82.48%
Runtime: 3.010920286178589s
```

可以看到准确率为 82.48%, 相对于 DT 来讲是要低一些. 目前的实现没有任何优化，完全是按照课件一步步实现的.