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

E12 Naive Bayes (C++/Python)

17341111 Xuehai Liu

2019 年 12 月 1 日

目录

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`.

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 `E12_YourNumber.pdf`, and send it to `ai_201901@foxmail.com`

4 Codes and Results

还是老样子，先对数据进行预处理，然后构造朴素贝叶斯网络，最后训练并进行预测。
第一部分：预处理，这部分主要将本来由字符串表示的数据转化为数字形式。

```
import pandas as pd
import numpy as np

def createDataSet(path):
    matrix =pd.read_csv(path,header = None )
    matrix = matrix.as_matrix().tolist()
    dataset = []
    labels = ["age", "workclass", "fnlwgt", 'education', 'education-num', \
              'marital-status', 'occupation', 'relationship', 'race', 'sex', \
              'capital-gain', 'capital-loss', 'hours-per-week', 'native-country']
    for line in matrix:
        dataset.append(line)
    return dataset,labels

#第一步构建初始数据集
myData, labels = createDataSet("adult.data")
test_data, test_labels = createDataSet("adult.test")
```

```
print(myData[:2])

#此函数将数据集的label列全部数值化。
def dealData(dataset,label):
    import copy
    newdataset = copy.deepcopy(dataset)
    list = [dataset[i][label] for i in range(len(dataset))]
    dict ={}
    j = 0
    for i, str in enumerate(list):
        if(str in dict.keys()):
            list[i] = dict[str]
        else:
            dict[str] = j
            list[i] = j
            j += 1
    for i in range(len(dataset)):
        newdataset[i][label] = list[i]
    return newdataset


#此函数对之前的数据集所有变量数值化，并将最终结果保存到filename的位置。
def preDeal(dataset,filename):
    print(dataset[:10])
    work = dealData(dataset, 1)
    edu = dealData(work, 3)
    mar = dealData(edu, 5)
    occ = dealData(mar, 6)
    rel = dealData(occ, 7)
    race = dealData(rel, 8)
    sex = dealData(race,9)
    native = dealData(sex,13)
    label = dealData(native,14)
    myData = label

    #myData = ageData
    print(myData[:10])
    import csv
    with open (filename,'w',newline= '')as f:
        writer = csv.writer(f)
        for line in myData:
            writer.writerow(line)
        f.close()

preDeal(myData,"adult_22-1.data")
preDeal(test_data,"adult_22-1.test")
```

第二部分：朴素贝叶斯网络单元：用类的形式，主要实现了拟合函数 fit，预测函数 predict 与计算 acc 函数。对连续的数据使用高斯密度函数进行计算条件概率。

```
import numpy as np


#Y_size = 5
#X_size = [1,2,3]
class Bayes:
    def __init__(self,X_size,Y_size):
        self.X_size = X_size
        self.Y_size = Y_size
        self.continuousList = [0,2,4,10,11,12]
        self.prob = {}

    def __calcGaussion__(self,x,avg,sigma):
        return 1/(np.sqrt(2*np.pi*sigma) * np.exp(-(x-avg)**2/(2*sigma) ) )

    #计算数据集下label特征的均值和方差
    def calcAvgtheta(self,dataset, label, y):
        list = [dataset[i][label] for i in range(len(dataset)) if dataset[i][14] == y]
        sum = 0
        for i in list:
            sum += i
        avg = sum / len(dataset)

        sum2 = 0
        for i in list:
            sum2 += (i - avg) ** 2
        theta = sum2 / len(dataset)
        return avg, theta

    def fit(self,train_data):
        #先验概率矩阵，描述每个Yi的先验概率
        self.Prior = np.zeros(self.Y_size)

        #条件概率矩阵，三维分别表示：Y的序号，特征X的序号，特征X的值
        self.Cond = []

        #初始化条件概率矩阵
        for i in range(self.Y_size):
            change = []
            for j in range(len(self.X_size)):
                change.append([0]*self.X_size[j])
            self.Cond.append(change)
        print(self.Cond)
        #count
```

```
    for i in range(len(train_data)):
        #统计Yi的出现次数
        self.Prior[train_data[i,-1]] += 1
        #对于x个特征，分别计算每个Y对应 Xj 的条件概率，此处先对各个P(Y,Xj)出现的次数
                                                    统计，作为分子
        for j in range(len(self.X_size)):
            self.Cond[train_data[i,-1]][j][train_data[i,j]] += 1
        #对于连续值的特征，需要使用概率分布来进行计算。
    #P [i][j][k] = numpy.random.normal(loc = avg, scale= theta )


    for i in range(self.Y_size):
        for j in range(len(self.X_size)):
            if (j not in self.continuousList):
                #不连续特征
                for k in range(self.X_size[j]):
                    #分母是Xj的属性个数+Yi的出现次数，计算得到条件概率P(Yi,Xj)
                    # +1 防止出现概率为0的情况（即laplacian 平滑）
                    self.Cond[i][j][k] = (self.Cond[i][j][k]+1 )/(self.Prior[i] +
                                                                self.X_size[j])
            else:
                #连续特征，使用高斯概率密度函数计算 P (Xj|Yi)
                avg,sigma = self.calcAvgtheta(train_data,j,i)
                self.prob[(j,i)] = lambda x:1/(np.sqrt(2*np.pi*sigma) * np.exp(-(x-
                                                        avg)**2/(2*sigma) ) )

        #得到每个Yi的先验概率：用每个Yi的出现次数/ 数据集长度
        self.Prior[i] = (self.Prior[i]+1)/ (len(train_data)+self.Y_size)

def predict(self,test_data):#a vector
    index = -1.
    maxp = -1
    #预测时使用公式计算，取P(yi)* 累乘P(Xj|yi)的最大值对应的y作为预测值
    for i in range(self.Y_size):
        pro = 1
        for j in range(len(self.X_size)):
            if(j not in self.continuousList): # 不连续变量
                pro *=  self.Cond[i][j][test_data[j]]
            else:
                pro *=  self.prob[(j,i)](test_data[j])
        pro  *= self.Prior[i]
        if(pro>maxp):
            maxp = pro
            index = i
    return index

def accuracy(self,test_data):
```

```
        acc = 0
        for i in  range(len(test_data)):
            #对于输入的数据集，计算每一个vector对应的predict值，并计算acc
            pre = self.predict(test_data[i])
            if(pre == test_data[i,-1]):
                acc += 1
        return acc/len(test_data)
```

第三部分：训练与预测。画图表现了最后的结果。

```
import  numpy as np
import  pandas as pd
import matplotlib.pyplot as plt
from Bayes_network import *

train_data = pd.read_csv("adult_22-1.data",header = None)
test_data =  pd.read_csv("adult_22-1.test",header = None)
train_data = np.array(train_data)
test_data = np.array(test_data)

#统计得到每一个特征X对应的值的个数，作为X_size
def countlabels(dataset):
    max = [ 0  for i in range(15)]
    for row in dataset:
        for i,num in enumerate(row):
            if( num+1 > max[i]):
                max[i] = num + 1
    return max[:-1]


X_size = countlabels(train_data)
Y_size = 2

if __name__ =='__main__':


    #模型构建
    bayes = Bayes (X_size,Y_size)

    #模型拟合
    bayes.fit(train_data)

    #预测
    acc = bayes.accuracy(test_data)
    print(acc)

    train_set = np.zeros(6,dtype = int)
    accur_set = np.zeros(6,dtype = float)
```

```
#分批次进行训练,观察数据量对模型准确率的影响
for i in range(6):
    if i==0:
        continue
    train_set[i] =(int)(i*(len(train_data)/5) )
    print(train_set[i])
    bayes.fit(train_data[:train_set[i],:])
    accur_set[i] = bayes.accuracy(test_data)

print(accur_set)
plt.plot(train_set,accur_set)
plt.xlabel("train data size")
plt.ylabel("accuracy")
plt.show()
```

预测结果:

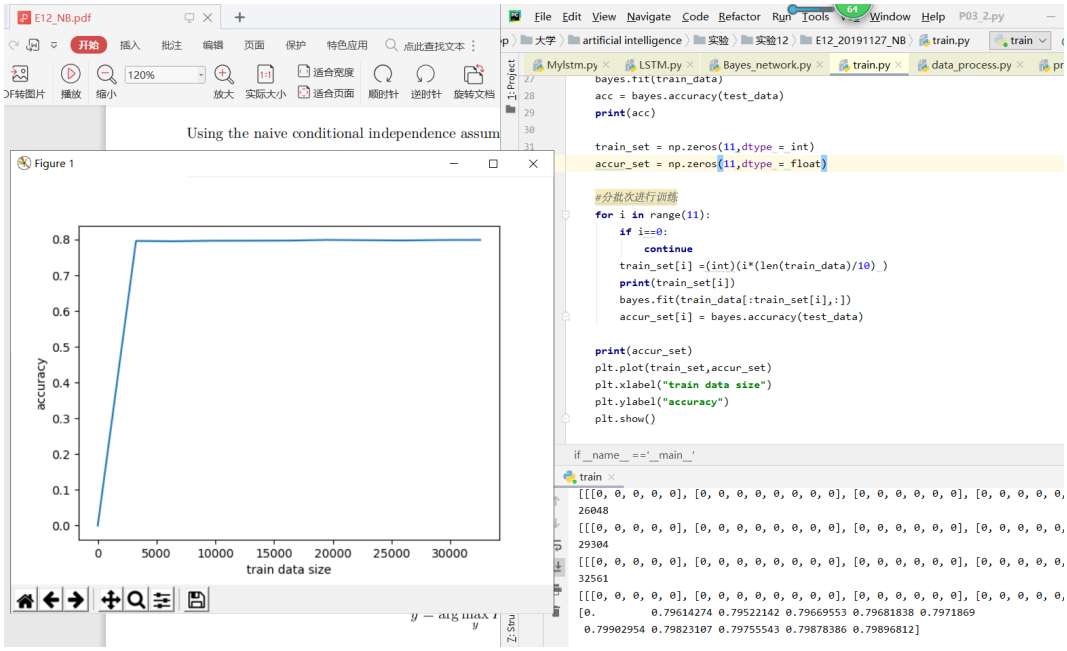

图 1: Result