SCHOOL OF DATA AND COMPUTER SCIENCE, SUN YAT-SEN UNIVERSITY

E11 Naive Bayes (C++/Python)

18340052 何泽

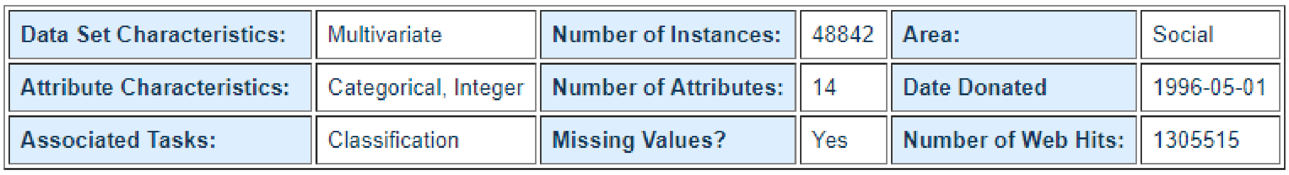
2020 年 11 月 25 日

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.



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 and Results

Code:

```python
import pandas as pd
import numpy as np

attr = {"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}
train = pd.read_csv("dataset/adult.data",names=attr.keys(),index_col=False)
```

```
test = pd.read_csv("dataset/adult.test",names=attr.keys(),index_col=
    False,header=0)
attributes = list(attr.keys())
attributes.remove("fnlwgt")
attributes.remove("capital-gain")
attributes.remove("capital-loss")
train= train[attributes]
test= test[attributes]

def fill(data,flag=1):
    if flag == 0:
        for a in data.columns.values:
            if attr[a]:
                data = data[data[a] != " ?"]
        return data
    else:
        for a in data.columns.values:
            if attr[a]:
                data.loc[data[a] == " ?",a] = data[a].value_counts().
                    argmax()
            else:
                pass
        return data
train = fill(train,1)
test = fill(test,1)

class NB():
    def __init__(self,train,attr):
        self.train = train
        self.attr = attr
        self.prob = {}
        self.prob[" >50K"] = train["salary"].value_counts(normalize=True
            )[" >50K"]
        self.prob[" <=50K"] = 1 - self.prob[" >50K"]
        self.attributes = train.columns.values[train.columns.values != "
            salary"]
        less = train[train["salary"] == " <=50K"]
        more = train[train["salary"] == " >50K"]
```

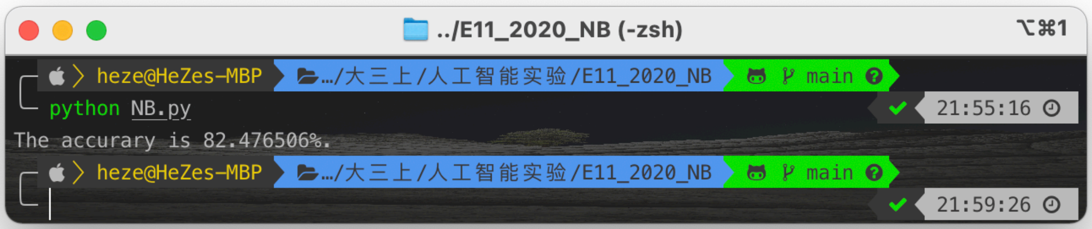
```python
        for a in self.attributes:
            if self.attr[a]:
                a_less = less[a].value_counts()
                a_more = more[a].value_counts()
                V = len(train[a].unique())
                for xi in train[a].unique():
                    self.prob[(xi,"<=50K")] = (a_less.get(xi,0) + 1) / (len(less) + V)
                    self.prob[(xi,">50K")] = (a_more.get(xi,0) + 1) / (len(more) + V)
            else:
                mu_less = np.mean(less[a])
                sigma_less = np.var(less[a])
                self.prob[(a,"<=50K")] = lambda x: 1 / np.sqrt(2*np.pi*sigma_less) * np.exp(-(x-mu_less)**2/(2*sigma_less))
                mu_more = np.mean(more[a])
                sigma_more = np.var(more[a])
                self.prob[(a,">50K")] = lambda x: 1 / np.sqrt(2*np.pi*sigma_more) * np.exp(-(x-mu_more)**2/(2*sigma_more))

    def predict(self,test):
        accurary = 0
        for i, row in test.iterrows():
            prod = np.array([self.prob["<=50K"],self.prob[">50K"]])
            for a in self.attributes:
                xi = row[a]
                if self.attr[a]:
                    prod[0] *= self.prob[(xi,"<=50K")]
                    prod[1] *= self.prob[(xi,">50K")]
                else:
                    prod[0] *= self.prob[(a,"<=50K")](xi)
                    prod[1] *= self.prob[(a,">50K")](xi)
            catagory = "<=50K" if prod.argmax() == 0 else ">50K"
            if catagory == row["salary"][:-1]:
                accurary += 1
        accurary /= len(test)
        print("The accurary is {:f}%.".format(accurary * 100))
        return accurary
```

```

nb = NB(train,attr)
nb.predict(test)
```

Result:



```
../E11_2020_NB (-zsh)                                                        ⌥⌘1
╭─  > heze@HeZes-MBP  …/大三上/人工智能实验/E11_2020_NB   main ?
╰─ python NB.py                                                    ✓  21:55:16
The accurary is 82.476506%.
╭─  > heze@HeZes-MBP  …/大三上/人工智能实验/E11_2020_NB   main ?
╰─ |                                                               ✓  21:59:26
```