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

E11 Naive Bayes (C++/Python)

18340215 张天祎

2020 年 11 月 26 日

目录

1 Datasets **2**

2 Naive Bayes **3**

3 Task **4**

4 Codes and Results **4**

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

```
import csv
import math
import numpy as np
from collections import Counter

def load(file_name):
    with open(file_name, 'r') as file:
        reader = csv.reader(file)
        rows = []
        for row in reader:
            if row != [] and len(row) == 15:
                tem = []
                for ind, word in enumerate(row):
                    if ind == 0:
                        tem.append(word)
                    elif ind == len(row) - 1 and word[-1]=='.':
                        tem.append(word[1:-1])
                    else:
                        tem.append(word[1:])
                rows.append(tem)
        return rows


def load_attributes(file_name):
    with open(file_name, 'r') as file:
        reader = csv.reader(file)
        rows = [row for row in reader if row != []]
        attributes = {}
        con_list = []
        dis_map = {}
        for row in rows:
            if row[0][0] != '|':
                row[-1] = row[-1][:-1]
                if row[0].find(':') == -1:
                    labels = row
                else:
```

```
                    tem = row[0].split(':')
                    attribute = tem[0]
                    row[0] = tem[1]
                    attributes[attribute] = [word[1:] for word in row]
        for ind, (attribute, val) in enumerate(attributes.items()):
            if val == ['continuous']:
                con_list.append(ind)
            else:
                dis_map[ind] = val
        return labels, con_list, dis_map


class NB():
    def __init__(self, labels, con_list, dis_map, train):

        train_data = [i[:-1] for i in train]
        train_label = [i[-1] for i in train]

        # 处理标签
        self.label_cpt = {}
        count = Counter(train_label)
        data_size = len(train)
        for i in count:
            self.label_cpt[i] = count[i]/data_size

        # 处理连续数据
        self.con_list = {}
        for i in con_list:
            for j in self.label_cpt:
                # self.con_list[(i,j)] = (np.mean([float(k[i]) for k in train if int(k[i]) != 0 and k[-1] == j]), math.sqrt(np.var([float(k[i]) for k in train if int(k[i]) != 0 and k[-1] == j])))
                self.con_list[(i,j)] = (np.mean([float(k[i]) for k in train if k[-1] == j]), math.
```

```
sqrt(np.var([float(k[i]) for k in train if k[-1] == j])))
        # 处理离散数据
        self.dis_cpt = {}
        for i in dis_map:
            count = dict(Counter([j[i] for j in train_data]))
            unknow = count.get('?', 0)
            for j in count:
                if j != '?':
                    count[j] = count[j] / (data_size - unknow)
            for j in dis_map[i]:
                if count.get(j, "") == "":
                    count[j] = 0
            if count.get('?', "") != "":
                count.pop('?')
            self.dis_cpt[i] = count

    def test(self, test_data):
        rst = []
        # print(self.con_list)
        for test in test_data:
            ph = {}
            for label in self.label_cpt:
                ph[label] = self.label_cpt[label]
                for ind, i in enumerate(test):
                    if ind not in [k[0] for k in self.con_list]:
                        if i != '?':
                            ph[label] *= self.dis_cpt[ind][i]
                        else:
                            ph[label] *= max(self.dis_cpt[ind].items(), key=lambda x:x[1])[1]
                    else:
```

```
                        if float(i) != 0:
                            u = self.con_list[(ind,label)][0]
                            sig = self.con_list[(ind,label)][1]
                            ph[label] *= np.exp(-(float(i) - u)** 2 /(2* sig**2))/(math.sqrt(2*math.pi)*sig)
                        # else:
                        #     sig = self.con_list[(ind, label)][1]
                        #     ph[label] *= np.exp(0) / (math.sqrt(2 * math.pi) * sig)
            rst.append(max(ph.items(), key=lambda x:x[1])[0])
        return rst

def main():
    train = load('dataSet/adult.data')
    labels, con_list, dis_map = load_attributes('dataSet/adult.names')

    nb = NB(labels, con_list, dis_map, train)

    test = load('dataSet/adult.test')
    test_label = nb.test([i[:-1] for i in test])
    length = len(test)
    count = 0
    for i in zip(test_label, [i[-1] for i in test]):
        if i[0]==i[1]:
            count += 1
    print("Accuracy:", count/length)


if __name__ == "__main__":
    main()
```

```
Accuracy: 0.8049259873472145


Process finished with exit code 0
```

- 结果如图
- 朴素贝叶斯分类器的效果比决策树分类器的效果略差，原因是该问题模型和决策树模型更相似。
- 若不使用连续数据，结果约 0.77，感觉贝叶斯分类器比决策树分类器更适合处理连续数据。