Lab 3 Thuật toán Naive Bayes

ThS. Lê Thị Thùy Trang

2024-12-30

1. Chuẩn bị môi trường thực hành

Sinh viên sử dụng ngôn ngữ `python` để xây dựng một mô hình Naive Bayes giải quyết một bài toán dự đoán đơn giản.

Để thực hiện bài thực hành, sinh viên cài đặt gói `numpy` trong `Python`.

Trước tiên, cài đặt `pip` - công cụ quản lý gói đi kèm với `Python`, giúp cài đặt các thư viện như `numpy`. Kiểm tra `pip` đã được cài đặt chưa, sử dụng lệnh:

```
pip --version
```

Nếu chưa, cài đặt `pip` bằng:

```
python -m ensurepip --upgrade
```

Sau khi đã có `Python` và `pip`, cài đặt `numpy` bằng lệnh sau:

```
pip install numpy
```

2. Mô hình phân loại Naive Bayes

Cho tập dữ liệu huấn luyện mô hình phân loại nhị phân Naive Bayes gồm các thuộc tính "Outlook", "Temperature", "Humidity", "Wind":

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Overcast	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes

Với tập dữ liệu này, hãy phát triển chương trình sử dụng mô hình phân loại Naive Bayes để dự đoán xem ngày thứ 11 (D11), bạn A có thể chơi tennis hay không?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D11	Sunny	Cool	High	Strong	???

Chương trình bao gồm các hàm chính:

- Hàm `create_train_data`: tạo dữ liệu huấn luyện dưới dạng một mảng `numpy`.
- Hàm `compute_prior_probablity`: tính xác suất tiên nghiệm của các lớp (`yes` và `no`).
- Hàm `compute_conditional_probability`: tính xác suất có điều kiện của các đặc trưng dựa trên lớp.
- Hàm `train_naive_bayes`: huấn luyện mô hình Naive Bayes bằng cách tính xác suất tiên nghiệm và xác suất có điều kiện.
- Hàm `get_index_from_value`: trả về chỉ số của một số giá trị trong danh sách các giá trị duy nhất của một đặc trưng.
- Hàm `prediction_play_tennis`: dự đoán lớp (`yes` hoặc `no`) cho một mẫu dữ liệu mới dựa trên mô hình Naive Bayes đã huấn luyện.

2.1 Tạo dữ liệu huấn luyện

Dữ liệu bao gồm các đặc trưng: `Outlook`, `Temperature`, `Humidity`, `Wind` và nhãn là `Play Tennis`.

Mỗi hàng đại diện cho một mẫu dữ liệu.

Nhãn Play Tennis có hai giá trị: `yes` (có chơi) và `no` (không chơi).

Hàm `create_train_data` sẽ tạo ra một mảng `numpy` chứa các mẫu dữ liệu với các đặc trưng và nhãn tương ứng.

```
import numpy as np

def create_train_data():

  data=[['Sunny','Hot', 'High', 'Weak', 'no'],
        ['Sunny','Hot', 'High', 'Strong', 'no'],
        ['Overcast','Hot', 'High', 'Weak', 'yes'],
        ['Rain','Mild', 'High', 'Weak', 'yes'],
        ['Rain','Cool', 'Normal', 'Weak', 'yes'],
        ['Rain','Cool', 'Normal', 'Strong', 'no'],
        ['Overcast','Cool', 'Normal', 'Strong', 'yes'],
        ['Overcast','Mild', 'High', 'Weak', 'no'],
        ['Sunny','Cool', 'Normal', 'Weak', 'yes'],
        ['Rain','Mild', 'Normal', 'Weak', 'yes']
        ]
  return np.array(data)

train_data = create_train_data()
print(train_data)
```

Sau đó, cần hoàn thành những hàm sau đây:

2.2 Tính toán xác suất tiên nghiệm (Prior Probability)

Xác suất tiên nghiệm là tỷ lệ số lượng mẫu thuộc lớp đó so với tổng số mẫu:

$$P(y) = \frac{\text{Số lượng mẫu thuộc lớp } y}{\text{Tổng số mẫu}}$$

Q1: Hoàn thiện hàm `compute_prior_probability` để tính xác suất tiên nghiệm của các lớp P("Play Tennis" = "yes") và tính P("Play Tennis" = "no")

```
def compute_prior_probablity(train_data):
  y_unique = ['no', 'yes']
  prior_probability = np.zeros(len(y_unique))
  # code here

  return prior_probability

prior_probablity = compute_prior_probablity(train_data)
print("P("Play Tennis" = no)", prior_probablity[0])
print("P("Play Tennis" = yes)", prior_probablity[1])
```

2.3 Tính xác suất có điều kiện (Conditional Probability)

Hàm `compute_conditional_probability` tính xác suất có điều kiện của các đặc trưng dựa trên lớp. Xác suất có điều kiện là xác suất của một giá trị thuộc tính xuất hiện trong một lớp cụ thể.

`x_conditional_probabilityp[j, k]`: Xác suất có điều kiện của đặc trưng $X = x$ khi biết nhãn kết quả $Y = y$, được tính bằng:

$$P(x|y) = P(X = x|Y = y) = \frac{\text{số lần} X = x \text{ xuất hiện khi } Y = y}{\text{số lần } Y = y \text{ xuất hiện}}$$

Q2: Hãy hoàn thiện đoạn mã sau để tính xác suất có điều kiện

```
#this function is used to compute the conditional probabilities
#input: train data
#output: conditional probabilities and list of feature names
def compute_conditional_probability(train_data):
  y_unique = ['no', 'yes']
  conditional_probability = []
  list_x_name = []
  for i in range(0,train_data.shape[1]-1):
    x_unique = np.unique(train_data[:,i])
    print("x_unique", x_unique)

    list_x_name.append(x_unique)

    # code here

    conditional_probability.append(x_conditional_probability)
  return conditional_probability, list_x_name
```

Q3: Đoạn code sau đây, thực thi hai công việc:

- Tính toán xác suất có điều kiện (Conditional Probability) và lấy danh sách các giá trị duy nhất của từng thuộc tính.
- In ra các giá trị duy nhất của từng thuộc tính để kiểm tra và hiểu rõ cấu trúc dữ liệu

Hãy cho biết kết quả của đoạn code?

```
train_data = create_train_data()
_, list_x_name  = compute_conditional_probability(train_data)
print("x1 = ",list_x_name[0])
```

```
print("x2 = ",list_x_name[1])
print("x3 = ",list_x_name[2])
print("x4 = ",list_x_name[3])
```

a)

x1 = ['Cool' 'Hot' 'Mild']

x2 = ['Overcast' 'Rain' 'Sunny']

x3 = ['High' 'Normal']

x4 = ['Strong' 'Weak']

b)

x1 = ['Overcast' 'Rain' 'Sunny']

x2 = ['Cool' 'Hot' 'Mild']

x3 = ['High' 'Normal']

x4 = ['Strong' 'Weak']

c)

x1 = ['Strong' 'Weak']

x2 = ['Cool' 'Hot' 'Mild']

x3 = ['High' 'Normal']

x4 = ['Overcast' 'Rain' 'Sunny']

d)

x1 = ['Overcast' 'Rain' 'Sunny']

x2 = ['Cool' 'Hot' 'Mild']

x3 = ['Strong' 'Weak']

x4 = ['High' 'Normal']

Hàm `get_index_from_value` để tìm chỉ số (index) của một giá trị trong danh sách các giá trị duy nhất của một thuộc tính.

```
#This function is used to return the index of the feature name
def get_index_from_value(feature_name, list_features):
  return np.where(list_features == feature_name)[0][0]
```

Q4: Đoạn code sau đây thực hiện các công việc:

- Tính toán xác suất có điều kiện và lấy danh sách các giá trị duy nhất của từng thuộc tính bằng cách gọi hàm `compute_conditional_probability(train_data)`.
- Tìm và in ra chỉ số của các giá trị cụ thể (`Overcast`, `Rain`, `Sunny`) trong danh sáchh giá trị duy nhất của đặc trưng `Outlook`.

Hãy cho biết kết quả của đoạn code?

```
train_data = create_train_data()
_, list_x_name  = compute_conditional_probability(train_data)
outlook = list_x_name[0]
i1 = get_index_from_value("Overcast", outlook)
i2 = get_index_from_value("Rain", outlook)
```

```
i3 = get_index_from_value("Sunny", outlook)

print(i1, i2, i3)
```

a) 1 2 0

b) 0 1 1

c) 0 1 2

d) 0 2 3

Q5: Đoạn code sau tính toán xác suất có điều kiện của $P(Outlook = Sunny|PlayTennis = Yes)$. Hãy cho biết kết quả?

```
train_data = create_train_data()
conditional_probability, list_x_name  =
compute_conditional_probability(train_data)
# Compute P("Outlook"="Sunny"|Play Tennis"="Yes")
x1=get_index_from_value("Sunny",list_x_name[0])
print("P('Outlook'='Sunny'|Play Tennis'='Yes') = ",
np.round(conditional_probability[0][1, x1],2))
```

a) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.27

b) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.47

c) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.37

d) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.17

Q6: Đoạn code sau tính toán xác suất có điều kiện của $P(Outlook = Sunny|PlayTennis = No)$. Hãy cho biết kết quả?

```
train_data = create_train_data()
conditional_probability, list_x_name  =
compute_conditional_probability(train_data)
# Compute P("Outlook"="Sunny"|Play Tennis"="No")
x1=get_index_from_value("Sunny",list_x_name[0])
print("P('Outlook'='Sunny'|Play Tennis'='No') = ",
np.round(conditional_probability[0][0, x1],2))
```

a) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.5

b) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.4

c) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.3

d) P('Outlook'='Sunny'|Play Tennis'='Yes') = 0.2

2.4 Huấn luyện mô hình

- Chức năng: Huấn luyện mô hình Naive Bayes bằng cách tính toán xác suất tiên nghiệm và xác suất có điều kiện.
- Đầu vào: Dữ liệu huấn luyện (train_data).
- Đầu ra:

  `prior_probability`: Xác suất tiên nghiệm của các lớp.

  `conditional_probability`: Xác suất có điều kiện của các đặc trưng.

`list_x_name`: Danh sách các giá trị duy nhất của từng đặc trưng.

Q7: Hãy hoàn thiện đoạn code sau đây để huấn luyện mô hình

```
############################
# Train Naive Bayes Model
############################
def train_naive_bayes(train_data):
    # Step 1: Calculate Prior Probability
    # code here

    # Step 2: Calculate Conditional Probability
    # code here

data = create_train_data()
prior_probability,conditional_probability, list_x_name = train_naive_bayes(data)
```

2.5 Hàm `prediction_play_tennis`

Với mỗii giá trị của đặc trưng trongg X, tìm chỉ số tương ứng trong danh sách các giá trị đặc trưng.

Tính toán xác suất hậu nghiệm bằng công thức:

$$P(y|X) = P(y)\prod_{i=1}^{4} P(x_i|y)$$

So sánh xác suất hậu nghiệm cho hai nhãn `no` và `yes`, nhãn nào có xác suất cao hơn thì trả về.

Q8: Hoàn thiện hàm `prediction_play_tennis` để hỗ trợ bạn A trả lời câu hỏi có nên đi chơi Tennis không trong trường hợp dưới đây?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D11	Sunny	Cool	High	Strong	???

```
####################
# Prediction
####################
def prediction_play_tennis(X, list_x_name, prior_probability, conditional_probability):

    x1=get_index_from_value(X[0],list_x_name[0])
    x2=get_index_from_value(X[1],list_x_name[1])
    x3=get_index_from_value(X[2],list_x_name[2])
    x4=get_index_from_value(X[3],list_x_name[3])

    # code here

    # print(p0, p1)

    if p0>p1:
        y_pred=0
    else:
        y_pred=1
```

```
    return y_pred

# prediction_play_tennis()
```

Q9: Đến đây ta đã xây dựng được một mô hình Naive Bayes hoàn chỉnh, ta sử dụng mô hình để trả lời câu hỏi A có đi chơi Tennis không trong trường hợp: X = [Sunny, Cool, High, Strong]. Hãy cho biết kết quả của đoạn code sau đây:

```
X = ['Sunny','Cool', 'High', 'Strong']
data = create_train_data()
prior_probability,conditional_probability, list_x_name = train_naive_bayes(data)
pred =  prediction_play_tennis(X, list_x_name, prior_probability, conditional_probability)

if(pred):
  print("A should go!")
else:
  print("A should not go!")
```

a) A should not go!

b) A should go!