Pattern Recognition and Machine Learning: Homework 6

Qingru Hu 2020012996

April 10, 2023

Problem 1

I use `hmmlearn` module to build HMM models.

(1)

I use the `CategoricalHMM` model in `hmmlearn` to train the dataset, and I obtain from fitting the initial , the transition and emission probabilities, shown respectively in , Fig.2 and Fig.3.

Dice Type	Dice 1	Dice 2
Initial Prob	0.618	0.382

Table 1: The initial probabilities

Dice Type	Dice 1	Dice 2
Dice 1	0.888	0.112
Dice 2	0.156	0.844

Table 2: The transition probabilities

Dice/Point	1	2	3	4	5	6
Dice 1	0.158	0.164	0.184	0.171	0.191	0.132
Dice 2	0.120	0.098	0.096	0.108	0.088	0.491

Table 3: The emission probabilities

The code is shown as below.

```
import numpy as np
from hmmlearn import hmm

data = np.load('sequences.npy')
X = data.reshape(200*30, 1)
lens = np.ones(data.shape[0])*30
lens = lens.astype(int)
model = hmm.CategoricalHMM(n_components=2, random_state=10)
model.fit(X, lens)
model.score(X)
```

```
# -10434.902086730863
```

(2)

Forward Algorithm

The probability of observing sequence 6 6 6 6 using forward algorithm is $p = 0.015$.

```
  iprob = model.startprob_
  tprob = model.transmat_
  eprob = model.emissionprob_

  for t in range(4):
      if t==0:
          a0 = eprob[0, 6]*iprob[0]
          a1 = eprob[1, 6]*iprob[1]
      else:
          a0 = eprob[0, 6]*(a0*tprob[0, 0] + a1*tprob[1, 0])
          a1 = eprob[1, 6]*(a0*tprob[0, 1] + a1*tprob[1, 1])
  p = a0 + a1
  # p = 0.014626307201743518
```

Backward Algorithm

The probability of observing sequence 6 6 6 6 using backward algorithm is $p = 0.015$.

```
  iprob = model.startprob_
tprob = model.transmat_
eprob = model.emissionprob_

for t in [3, 2, 1, 0]:
    # as = np.zeros([2, 2])
    if t==3:
        b0 = 1
        b1 = 1
    else:
        b0 = tprob[0, 0]*eprob[0, 6]*b0 + tprob[0, 1]*eprob[1, 6]*b1
        b1 = tprob[1, 0]*eprob[0, 6]*b0 + tprob[1, 1]*eprob[1, 6]*b1
p = a0 + a1
# p = 0.014626307201743518
```

(3)

This player is cheating and he switched his dice on his 12th roll.

```
seq = np.array([3, 2, 1, 3, 4, 5, 6, 3, 1, 4, 1, 6, 6, 2, 6])
seq = seq.reshape(1, -1)
model.decode(seq)
```

```
# log_prob = -28.45720629383466,
# state_sequence = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]
```

Problem 2

2.1

2.2

2.3

Decision Tree

criterion	Q_G			Q_L			Q_L		
max_depth	I	B	U	I	B	U	I	B	U

2.4