Hidden Markov Model

CMPT 498/820 Machine Learning
Tutorial 6

Najeeb Khan

November 2, 2016

1 Hidden Markov Models

This tutorial uses `hmmlearn` package which can be installed by running `<pip install hmmlearn>` in a command line interface. For the assignment you can't use `hmmlearn`.

1.1 Imports

```
In [1]: #some imports
        import re
        import numpy as np
        import pandas as pd
        from hmmlearn.hmm import MultinomialHMM
        from collections import Counter
        from sklearn.metrics import accuracy_score
        #from mleFile import mleOutcomes, mleTrans
```

1.2 Preprocessing

```
In [2]: data = pd.read_csv('training_Magic.txt', sep="\t", header = None)
        data.columns = ["state", "outcome"]
```

1.3 Counting/Training

```
In [3]: # code in mleFile is part of the assignment questions 1 & 2
        # Outcome Probabilities of fair  and cheat state
        #P_fair_win, P_cheat_win = mleOutcomes(data)
        P_fair_win, P_cheat_win = 0.507874015748, 0.397408207343

        P_fair_lose = 1 - P_fair_win
        P_cheat_lose = 1 - P_cheat_win

        print (P_fair_win)
        print (P_fair_lose)
        print (P_cheat_win)
        print (P_cheat_lose)
```

```
0.507874015748
0.49212598425200005
0.397408207343
0.602591792657
```

```
In [4]: # Calculated state transition probabilities.
        #f2f, f2c, c2c, c2f = mleTrans(data)
        f2f, f2c, c2c, c2f = 0.5196850393700787, 0.48031496062992124, \
                                0.474025974025974, 0.525974025974026
        print(f2f, f2c, c2c, c2f)
```

```
0.5196850393700787 0.48031496062992124 0.474025974025974 0.525974025974026
```

1.4 Define HMM

Here we define an HMM model with 2 states: `n_components=2`, and multinomial outcome distributions: `MultinomialHMM`

```
In [5]: model=MultinomialHMM(n_components=2)
```

What is the probability of being in fair state at the begining of time?

```
In [6]: model.startprob_ = [0.5, 0.5]
```

How frequently transitions occurs between states?

```
In [7]: model.transmat_= np.array([
                                    # Next state
                                    #   0       1
                                      [c2c,   c2f], # Current state 0
                                      [f2c,   f2f]  # Current state 1
                                  ])
```

What the different outcomes/values that can be observed in each state?

```
In [8]: model.n_features = 2 # Number of possible outcomes
        model.emissionprob_ = np.array([
                              # Outcome lose,            win
                                      [P_cheat_lose,   1-P_cheat_lose], # State cheat
                                      [1 - P_fair_win, P_fair_win    ]  # State fair
                                 ])
```

Generate some data using the defined HMM model

```
In [9]: #X, Z = model.sample(n_samples=500)
```

1.5 Labelling Data

```
In [10]: # reading some outcome data file
         observations=[]
         with open('testing_Magic.txt') as f:
             rawdata=f.readlines()
             observations = [1*(s.strip()=='win') for s in rawdata]
         observations=np.array(observations).reshape(len(observations),1)
```

```
In [11]: # Q5 asks to implement forward-backward algorithm
         # and use that to decode states
         Zp=model.decode(observations, algorithm="viterbi")
```

```
In [12]: c=Counter(Zp[1])
         c
```

```
Out[12]: Counter({0: 442, 1: 358})
```