E3-EM

October 12, 2018

In this assignment, you are asked to actually implement the EM algorithm derived in class. Your implementation will be using Python. Recall that the basic setup was that we imagine that there are two coins in a bag. Repeatedly, we pick one out and flip it 10 times, then put it back in. We derived an algorithm to look at all of the sequences of 10 flips, and figure out the probability that each coin comes up heads.

One thing that might help you is

```
scipy.stats.binom.pmf (numHeads, numTrials, probOfHeads)
```

This function allows you to compute the binomial probability of seeing the specified number of heads in the specified number of trials.

Grading:

Point allocation: * 0-20 Points for myEM function implementation * 15 Points for successfully running Task 1 * 15 points for successfully running Task 2

Start with the following code:

```
In [9]: import numpy as np
        import scipy.stats

        # one coin has a probability of coming up heads of 0.2, the other 0.6
        # this is the truth we will use to generate the sequences of coin flips
        coinProbs = np.zeros (2)
        coinProbs[0] = 0.2
        coinProbs[1] = 0.6

        # reach in and pull out a coin numTimes times
        numTimes = 100

        # flip it numFlips times when you do
        numFlips = 10

        # flips will have the number of heads we observed in 10 flips for each coin
        flips = np.zeros (numTimes)
        for coin in range(numTimes):
                which = np.random.binomial (1, 0.5, 1);
                flips[coin] = np.random.binomial (numFlips, coinProbs[which], 1);

        # initialize the EM algorithm
```

```
        coinProbs[0] = 0.79
        coinProbs[1] = 0.51
```

Using this code as a start, write some Python code that runs 20 iterations of the EM algorithm that we derived. At the end of each iteration, print out the current probabilities of the two coins.

```
In [ ]: # define the EM algorithm
        def myEM():
            for iters in range (20):
                # my code goes here
```

Now run the function

```
In [ ]: myEM()
```

My results are below. Note that your results might be different, since the data are randomly generated.

```
[0.6913887786871945, 0.3205189463068871]
[0.6349440972384687, 0.2573384274377747]
[0.6044228758697338, 0.22727544075146722]
[0.5887711493028852, 0.21171140477915998]
[0.5808172323753603, 0.20354444512168343]
[0.5767500765491345, 0.1992741649770466]
[0.5746569298008388, 0.19705084180978027]
[0.5735756856294365, 0.19589616580905417]
[0.5730161693988925, 0.1952972264380479]
[0.5727264119901841, 0.1949867278135566]
[0.5725763063104846, 0.19482580210801304]
[0.5724985350227417, 0.19474240699153128]
[0.5724582385320893, 0.1946991922060499]
[0.5724373587164968, 0.19467679917115008]
[0.5724265396099921, 0.19466519569444465]
[0.5724209335381654, 0.19465918311774882]
[0.5724180286651543, 0.19465606758891035]
[0.572416523458053, 0.19465445322203573]
[0.5724157435100066, 0.1946536167097004]
[0.5724153393668321, 0.19465318325643205]
```

This time, reduce numFlips to 2.
Reset the initial estimates for the probabilities.

```
In [11]: numFlips = 2
         coinProbs[0] = 0.2
         coinProbs[1] = 0.6
```

Regenerate the data and reinitialize the starting probabilities

```
In [12]: # flips will have the number of heads we observed in 10 flips for each coin
         flips = np.zeros (numTimes)
         for coin in range(numTimes):
                 which = np.random.binomial (1, 0.5, 1);
                 flips[coin] = np.random.binomial (numFlips, coinProbs[which], 1);

         # initialize the EM algorithm
         coinProbs[0] = 0.79
         coinProbs[1] = 0.51
```

and rerun the EM algorithm

```
In [ ]: myEM()
```

My results from the second run are:

```
[0.57170075073339, 0.24386416623514545]
[0.5343869486316972, 0.19806150349976756]
[0.534088640515714, 0.18205211266965377]
[0.539898119875198, 0.1726941015450789]
[0.5457572835473563, 0.16592423331454195]
[0.5505645667426521, 0.160770869382013]
[0.554271213920473, 0.15683662462603218]
[0.5570534369577226, 0.15385618787812252]
[0.5591128976731792, 0.151617730583195]
[0.5606255624759454, 0.14994883671785908]
[0.5617317646002122, 0.1487117602279271]
[0.5625387932685745, 0.14779881984552143]
[0.563126836563891, 0.14712733303164222]
[0.5635550717541402, 0.14663467479903525]
[0.563866863281927, 0.14627389344325]
[0.5640938678280079, 0.14601005302192102]
[0.564259151161806, 0.14581730319970954]
[0.5643795051676515, 0.1456765952081389]
[0.5644671507716209, 0.14557393495341814]
[0.5645309820954078, 0.14549906478102514]
```

Remarkably, the EM algorithm does a pretty job, even with just two flips of each coin!!
Turn in your code (& results) in a Jupyter Notebook.