CS 573 Homework 4

Yifan Fei

Computer Science
Purdue University

Instructor: Prof. Bruno Ribeiro

Spring 2018

Section 0:

1

Yes, I discussed the homework with others but came up with my own answers. Their name(s) are Yupeng Han and Meng Liu.

2

I have used online resources to help me answer this question, but I came up with my own answers. Here is a list of the websites I have used in this homework:

UCB1: https://jeremykun.com/2013/10/28/optimism-in-the-face-of-uncertainty-the-ucb1-algorithm/

Q1: Deep Learning

1: The structure of Neural Network

First calculate the Covariance matrix X^TX, treat its row ,column indexes as one hot vectors and the scores as the output value. i.e The input is the one hot encoding of indexes, and the output is the expected value trained by our neural network. The output matrix Y is D by D using 2 for-loop, but for each small calculation, $\hat{y}$ is 1 by 1 since $\hat{y} = \sum_{m=1}^{M} \sigma_m u_m u_m$

Let us define two layers, the first layer for U, and the second layer for Σ. Both have no bias. For the truncated PCA method, we just want M columns of the U matrix and singular value matrix. Then the dimension of U is D by M, Σ is M by M, which means that the weight of the first hidden layer is D by M, and the weight of the second layer is M by M. Notice that I use softmax() as an activation function after layer 1.

The loss function is:

$$L = ||Y_{real} - \hat{Y}||^2$$

$$L = ||X^TX - \hat{Y}||^2 = \sum_{i=1}^{D}\sum_{j=1}^{D}(x_{i,j}^T x_{i,j} - \hat{y}_{i,j})$$

$\hat{Y}$ is the value calculated by the neural network and X^TX is the exact value of the covariance matrix.

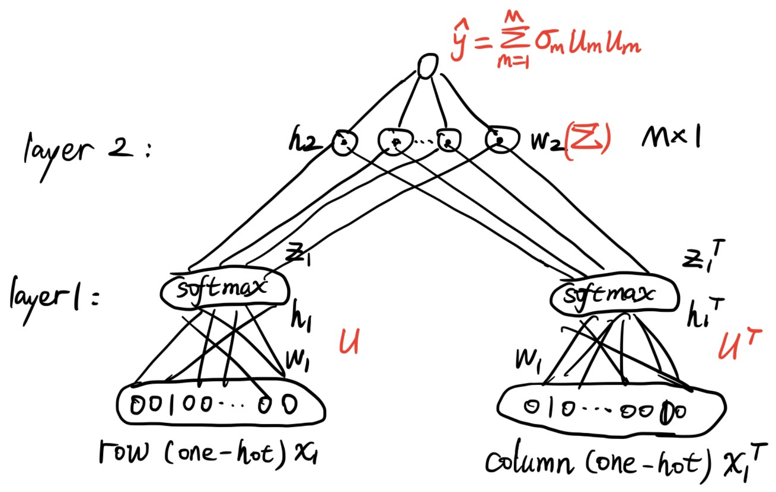


Figure 1: The structure of Neural Network

2: Forward pass

Define the value before layer 1 as h1. h1 passed the sofmax() activation function and z1 is the output of layer 1. Then in layer 2 the row and column vectors meet. No bias in two layers. Weight of layer 1 is U and weight of layer 2 is Σ.

$$h_1 = W_1 x$$
$$z_1 = softmax(h_1)$$
$$h_2 = W_2 z_1 z_1^T$$
$$\hat{Y} = h_2$$

Actually,

$$\hat{y} = \sum_{m=1}^{M} \sigma_m u_m u_m$$

$$U = W_1$$
$$\Sigma = W_2$$

3: Code

Turn-in

4:PCA

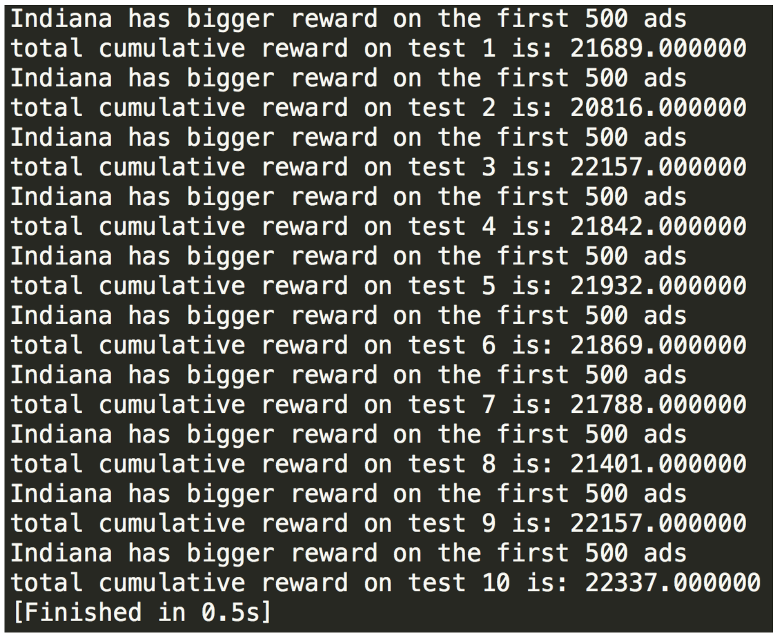
Q2: Multi-armd Bandits

a: Initial try

Python Simulation on case 1

```
import random
# Case 1
IN_list=[0]*(4875504+1608298) # 0 stands for adult
Ca_list=[0]*(500908+100815)   # 0 stands for adult

for i in range(1608298):
    IN_list[i]=1 # 1 stands for pop

for i in range(100815):
    Ca_list[i]=1 # 1 stands for pop

reward_List=[]
test_num = 10

for i in range(test_num):
    IN_sample = random.sample(IN_list,9500)
    Indi_test  = IN_sample[:500]
    Indi_9000 = IN_sample[501:9000]
    rew_indi = Indi_test.count(1)

    Ca_sample = random.sample(Ca_list,9500)
    Cali_test  = Ca_sample[:500]
    Cali_9000  = Ca_sample[501:9000]
    rew_cali = Cali_test.count(1)

    count_0, count_1 = 0, 0
    cost = 10000*0.01

    if Indi_test.count(1)>Cali_test.count(1):
        print('Indiana has bigger reward on the first 500 ads')
        total_count_indi = IN_sample.count(1)
        total_count_cali = Cali_test.count(1)

    else:
        print('District of Columbia has bigger reward on the first 500 ads')
        total_count_indi = Indi_test.count(1)
        total_count_cali = Ca_sample.count(1)

    revenue = (total_count_cali + total_count_indi)*0.01*10
    cumulative_reward = revenue - cost
    reward_List.append(cumulative_reward)
    print('total cumulative reward on test %d is: %f' %  (i+1,cumulative_reward))
```

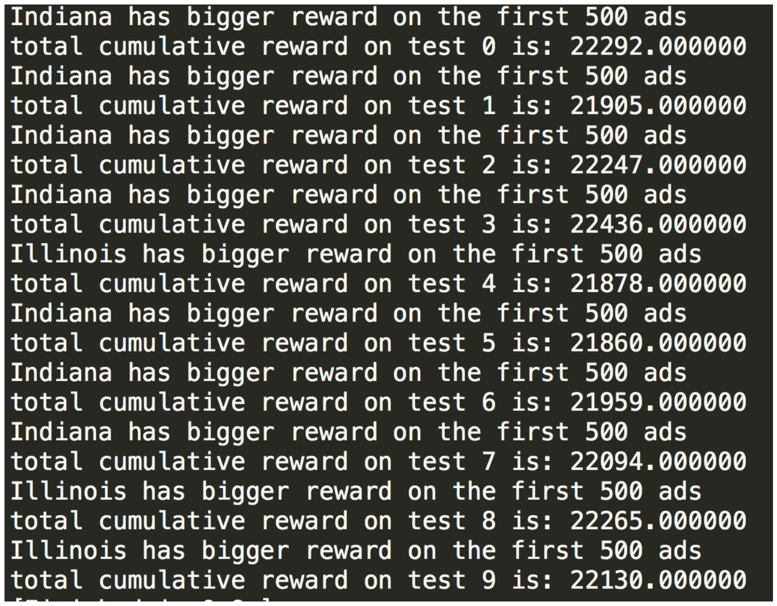
I run 10 test on the data with random function and found that Indiana seems to always have bigger reward on the first 500 ads. The result is shown below. The reward is around 2150.



```
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 1 is: 21689.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 2 is: 20816.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 3 is: 22157.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 4 is: 21842.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 5 is: 21932.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 6 is: 21869.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 7 is: 21788.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 8 is: 21401.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 9 is: 22157.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 10 is: 22337.000000
[Finished in 0.5s]
```

Figure 2: 10 Simulations on Case 2

Python Simulation on case 2 Using the similar code we can find that some test IN wins, with the others IL wins. The result is shown below. The reward is around 2150.



```
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 0 is: 22292.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 1 is: 21905.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 2 is: 22247.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 3 is: 22436.000000
Illinois has bigger reward on the first 500 ads
total cumulative reward on test 4 is: 21878.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 5 is: 21860.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 6 is: 21959.000000
Indiana has bigger reward on the first 500 ads
total cumulative reward on test 7 is: 22094.000000
Illinois has bigger reward on the first 500 ads
total cumulative reward on test 8 is: 22265.000000
Illinois has bigger reward on the first 500 ads
total cumulative reward on test 9 is: 22130.000000
```

Figure 3: 10 Simulations on Case 2

This procedure is not a fully dynamic process as UCB1 and is not stable. We can see in case 1 the proportion of under-18 pop in District of Columbia is so small. There might be a case when 500 ads go to District of Columbia happens to have more reward than under-18 in Indiana, leading to the remaining 9000 ads going to that bad-performance district. However in case 2 the population distribution in 2 states are about the same and is more stable under this simple algorithm. By doing experiment for 1000 times, I found there is a win for District of Columbia in case 1, which prove my guess.

b:UCB1 algorithm

UCB_1 uses the strategy of playing arm with the largest upper bound. For this problem, there are 2 arms for case 1 and case 2.

suppose $Y_1, \ldots, Y_n$ are independent random variables whose values lie in [0,1] and whose expected values are μ_i. Define:

$$Y = \frac{1}{n}\sum_i Y_i, \mu = E(Y) = \frac{1}{n}\sum_i \mu_i$$

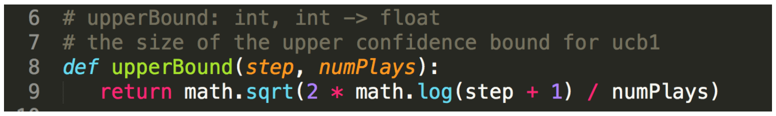
The Chernoff-Hoeffding inequality gives an exponential upper bound on the probability that the value of Y deviates from its mean:

$$P(Y + a < \mu) \leq e^{-2na^2}$$

The variable Y is the empirical average payoff for action j over all the times we tried it. Define n_j as the number of times we played action j so far, and by using $a = a(j,T) = \sqrt{2log(T)/n_j}$, we have:

$$P(Y > \mu + a) \leq T^{-4}$$

Which converges to zero very quickly. Here the upper bound is calculated by a simple function:



```
# upperBound: int, int -> float
# the size of the upper confidence bound for ucb1
def upperBound(step, numPlays):
    return math.sqrt(2 * math.log(step + 1) / numPlays)
```

Figure 4: function upperbound()

UCB1 Algorithm:

1. Play each of the k actions once(Here k=2), giving initial values for empirical mean payoffs $\overline{y}_i$ of each action i.
2. For the next each step, play the action j maximizing $\overline{y}_i + \sqrt{2log(T)/n_j}$
3. Observe the reward $Y_{j,t}$ and update the empirical mean for each action.
4. Repeat step(2) and (3) until 10000 runs end.

For this question, use random function in python to generate the reward for each action.

The cumulative Reward for case 1 using UCB1 is 22250.02. The cumulative Reward for case 2 using UCB1 is 21740.02. This result is not expected. Intuitively, the under-18 pop has larger portion in case 2 than that in case 1. I would expect case 2 will have a bigger reward. Also I expect UCB1 algorithm can beat the simple algorithm in part 1. The reason why this does not make a difference is that both part 1 and 2 are lead by Indiana state. The probability strength of Indiana state makes it like only put ads on Indiana state for both algorithms.

c:Plot and Comparison

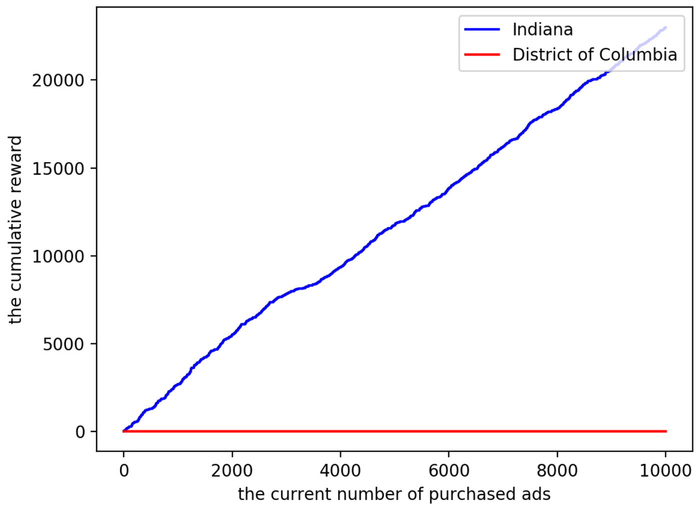


Figure 5: Case 1

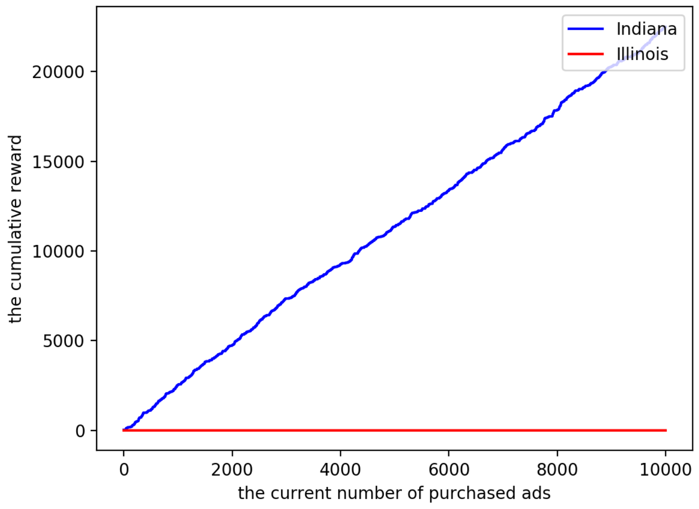


Figure 6: Case 2

We can see the curve is zigzag due to the random but is similar to a straight line with specific slope. Plus the reward of Indiana is increasing shapely, while the red curve(District of Columbia in case 1 and Illinois in case 2) remains about 0.

However, after doing many experiments, I find that in some other case, things might be opposite.

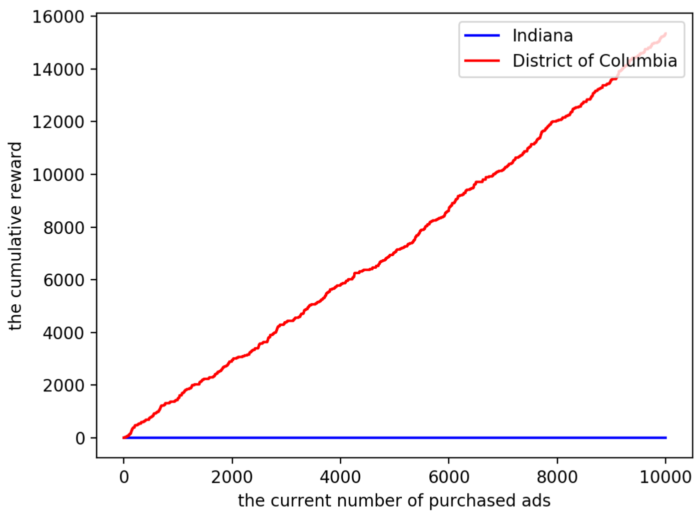


Figure 7: Case 1

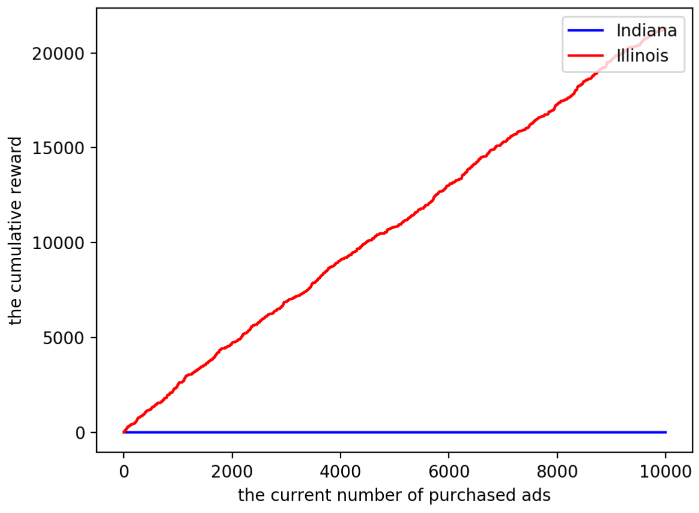


Figure 8: Case 2

One difference for case 1 with 2 different result is that when the line of Indiana is over District of Columbia, the cumulative reward ends at about 22000, but when the line of Indiana is below District of Columbia, the cumulative reward ends at about 14660.

Another difference between 2 cases is that the initial slope of the line in case 1 is bigger than that in case 2. The reason is that the gap of the proportion of young population of 2 states in case 1 is bigger than that in case 2, causing bigger "momentum" in initial state in case 1.