# QiShi Quiz 3

Xiangyu Zhang

June 2019

## 1 Math

### Problem 1

We calculate for more general case. Assume we have $m$ boys and $n$ girls, and we want to calculate the boy-girl adjacent pair in expectation. Denote $Z_i$ to be the number of girls adjacent to $i^{th}$ boy. Then we have

$$\mathbb{P}(Z_i = 2) = \frac{n}{n+m-1}\frac{n-1}{n+m-2}$$
$$\mathbb{P}(Z_i = 1) = \frac{n}{n+m-1}\frac{m-1}{n+m-2} + \frac{m-1}{n+m-1}\frac{n}{n+m-2}$$

So we can conclude

$$\mathbb{E}[Z_i] = \frac{2n(m-1) + 2n(n-1))}{(n+m-1)(n+m-2)}$$

So the expectation of boy-girl pair

$$\mathbb{E}[\sum_{i=1}^{m} Z_i] = \frac{2n^2m + 2nm^2 - 4nm}{(n+m-1)(n+m-2)}$$

### Problem 2

Independence implies uncorrelated, but uncorrelated does not imply independence. For random variable $X \sim N(0,1)$, $X^2$ and $X$ are uncorrelated, but they are not independent.

### Problem 3

The problem can be formulated as MDP. We use $V_t(x)$ the denote the optimal reward we can get starting from the $t^{th}$ round if the value of last toss is $x$. Then we have the dynamic equation

$$V_t(x) = \max\{x, \mathbb{E}_{y\sim U(0,1)}[V_{t+1}(y)]\}$$

By $V_3(x) = x$, we know

$$
\begin{aligned}
V_2(x) &= \max\{x, \mathbb{E}_{y\sim U(0,1)}[V_3(y)]\} = \max\{x, \frac{1}{2}\} \\
V_1(x) &= \max\{x, \mathbb{E}_{y\sim U(0,1)}[\max\{x, 0.5\}]\} = \max\{x, 0.6125\} \\
V_0(x) &= \max\{x, \mathbb{E}_{y\sim U(0,1)}[\max\{x, 0.6125\}]\} = \max\{x, 0.687578125\}
\end{aligned}
$$

So the value of this game is $V_0(0) = 0.687578125$.

## Problem 4

We denote the length of $N$ pieces from left to right to be $x_1, x_2, ..., x_{n-1}, x_n$. Then $x_1 + x_2 + ... + x_n = 1$. We want to calculate the probability, for $0 \leq \alpha \leq 1$,

$$\mathbb{P}(\max_{1\leq i\leq n} x_i \leq \alpha)$$

Denote the $A_i = \{x_i \geq \alpha\}$. Then we have

$$
\begin{aligned}
\{\max_{1\leq i\leq n} x_i \leq \alpha\}^c &= \{x_1 \leq \alpha, x_2 \leq \alpha, ..., x_n \leq \alpha\}^c \\
&= \cup_{1\leq i\leq n}\{x_i \leq \alpha\}^c \\
&= \cup_{1\leq i\leq n}\{x_i \geq \alpha\}
\end{aligned}
$$

So we have

$$
\begin{aligned}
\mathbb{P}(\max_{1\leq i\leq n} x_i \leq \alpha) &= \mathbb{P}(x_1 \leq \alpha, x_2 \leq \alpha, ..., x_n \leq \alpha) \\
&= 1 - \mathbb{P}(\cup_{1\leq i\leq n} x_i \geq \alpha)
\end{aligned}
$$

By De Morgan's law, we know

$$\mathbb{P}(\cup_{1\leq i\leq n} x_i \geq \alpha) = \sum_i \mathbb{P}(A_i) - \sum_{i<j} \mathbb{P}(A_i \cap A_j) + \sum_{i<j<k} \mathbb{P}(A_i \cap A_j \cap A_k) + ...$$

By geometric intuition, we know that, if $\alpha k < 1$,

$$\mathbb{P}(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k}) = (1 - \alpha k)^{n-1}$$

So we can conclude

$$
\begin{aligned}
\mathbb{P}(\cup_{1\leq i\leq n} x_i \geq \alpha) &= \sum_i \mathbb{P}(A_i) - \sum_{i<j} \mathbb{P}(A_i \cap A_j) + \sum_{i<j<k} \mathbb{P}(A_i \cap A_j \cap A_k) + ... \\
&= \sum_{i=1}^{n} \binom{n}{i} (-1)^{i+1} (1 - \alpha i)^{n-1}
\end{aligned}
$$

So we deduce the distribution for longest length as

$$\mathbb{P}(\max_{1\leq i\leq n} x_i \leq \alpha) = \sum_{i=0}^{n} \binom{n}{i} (-1)^{i} (1 - \alpha i)^{n-1}$$

## Problem 5

The matrix $cov(X, Y, Z)$ is positive semidefinite. So we can conclude

$$1 - r^2 \geq 0$$
$$1 - 2r^2 \geq 0$$

So we can conclude $-\frac{\sqrt{2}}{2} \leq r \leq \frac{\sqrt{2}}{2}$.

## Problem 6

Initially we assume the drunk man is at 0, and the left door at $-1$ and right door at 99. Denote $Z_t$ to be the location the drunk man stands at time $t$. Denote $\tau$ to be the stopping time when the drunk man hit the left door or the right door.

(a) Notice $Z_t$ and $Z_t^2 - t$ are two martingale, so we conclude

$$\mathbb{E}[Z_\tau] = 0$$
$$\mathbb{E}[Z_\tau^2 - \tau] = 0$$

which implies

$$\mathbb{P}[Z_\tau = 99] = \frac{1}{100}, \mathbb{P}[Z_\tau = -1] = \frac{99}{100}$$
$$\mathbb{E}[\tau] = \mathbb{E}[Z_\tau^2] = \frac{99}{100} \times 1 + \frac{1}{100} \times 99^2 = 99$$

(b) If the drunk man stands at the left door, and he moves left in next step, he will hit the left door and stay at the same place. We calculate $T$, the average steps the drunk man takes to go home when he starts from 0, and $S$, the average steps the drunk man takes to go home when he starts from the left door.

By the answer in part (a), we have

$$S = \mathbb{E}[\tau] + \mathbb{P}[Z_\tau = -1]T$$
$$T = 1 + \frac{1}{2}T + \frac{1}{2}S$$

so we have

$$S = 10098, T = 10100$$

## Problem 7

We denote matrix $X$ as the feature matrix, and $Y$ as the prediction vector $Y$. The underlying model is given by

$$Y = X\beta + \epsilon$$

Ridge regression means we optimize

$$||Y - X\beta||^2 + \lambda \, ||\beta||^2$$

to recover $\beta$.

Ridge regression is used when we want to leverage the trade off between the prediction accuracy and model complexity to avoid overfitting.

### Problem 8

We denote matrix $X$ as the feature matrix, and $Y$ as the prediction vector $Y$. The underlying model is given by

$$Y = X\beta + \epsilon$$

Lasso regression means we optimize

$$||Y - X\beta||^2 + \lambda|\beta|$$

to recover $\beta$.

Lasso regression is used when the parameter $\beta$ is sparse.

### Problem 9

We use $R$ to denote the return. The return is a random variable with density

$$\mathbb{P}[R = \sum_{i=1}^{n} 2^i] = \frac{1}{2^{n+1}}$$

Return itself is heavy hail, so the expectation of $R$ is $\infty$. To decide how much money we would like to invest into this game, we propose the following approach. Take $_\alpha$ to be the upper-$\alpha$ quantile of the return,

$$\mathbb{P}[R \le R_\alpha] = 1 - \alpha$$

Then based on the personal preference of the risk, we suggest to invest

$$\mathbb{E}[R|R \le R_\alpha]$$

## 2 Programming

### Problem 10

Sorry I do not understand what 'default' means in this problem.

## Problem 11

The implementation is in Python.

```python
import threading

# Based on tornado.ioloop.IOLoop.instance() approach.
# See https://github.com/facebook/tornado
class SingletonMixin(object):
  __singleton_lock = threading.Lock()
  __singleton_instance = None

  @classmethod
  def instance(cls):
    if not cls.__singleton_instance:
      with cls.__singleton_lock:
        if not cls.__singleton_instance:
          cls.__singleton_instance = cls()
    return cls.__singleton_instance
```

## Question 12

We use dynamic programming in this question. The state we store is tuple $(t, r)$, which $t$ is the current time, and $r$ is the remaining transaction time. We use $P(t, r)$ to store the maximum profit if we start at time t and could complete at most $r$ transaction in the future. The value we are interested is $P(0, k)$.

We use dynamic programming, and start from $t = n$.

```python
def max_profit(prc, k):
    assert prc is not None
    prc_diff = [prc[i + 1] - prc[i] for i in range(len(prc) - 1)]
    n = len(prc_diff)

    curr_in_old = [0 for _ in range(k + 1)]
    curr_out_old = [0 for _ in range(k + 1)]
    curr_in_new = [0 for _ in range(k + 1)]
    curr_out_new = [0 for _ in range(k + 1)]
    # At i-th step, with max j transactions so far,
    # curr_in[j] is the max profit with prc_diff[j] in our trade
    # curr_out[j] is the max profit with prc_diff[j] not in our
    trade

    for i in range(n):
        for j in range(1, k + 1, 1):
            curr_in_new[j] = prc_diff[i] + max(curr_out_old[j - 1],
                                               curr_in_old[j])
            curr_out_new[j] = max(curr_in_old[j], curr_out_old[j])

        curr_in_old = curr_in_new.copy()
        curr_out_old = curr_out_new.copy()

    return max(curr_in_new[k], curr_out_new[k])
```

## Question 13

The trick is to use bit-wise xor.

```python
    def appear_once(array):
        assert array

        value = 0
        for num in array:
            value = value ^ num
        return value
```

## Question 14

In C++, it is legal for a non-virtual function to can a virtual function. But this is not recommended. SEE THE CODE BELOW.

```cpp
#include <iostream>
using namespace std;

class Base
{
public:
  virtual void print()
  {
    cout << "Base class print function \n";
  }
  void invoke()
  {
    cout << "Base class invoke function \n";
    this -> print();
  }
};

class Derived: public Base
{
public:
  void print()
  {
    cout << "Derived class print function \n" ;
  }
  void invoke()
  {
    cout << "Derived class invoke function \n";
    this -> print(); // called under non - virtual function
  }
};

int main()
{
  Base *b = new Derived;
  b -> invoke();
  return 0;
}
```

The output should be

```
Base class invoke function
Derived class print function
```

### Question 15

We usually use multiplication, addition, and module arithmatic operation to generate pseudo-random number. Start from a seed $X_0$ and generate random numbers by

$$X_{n+1} = (aX_n + b) \mod m \tag{1}$$

But constant $a, b$ and $m$ need to be selected with caution. To test the randomness, we could use various of statistical testing method.

### Question 16

To toss a coin $i$ times, we have in total $2^i$ possible outcome. To roll a dice $j$ times, we have in total $6^j$ possible outcome. so to increase the efficiency, it is equivalent to find

$$i, j = \arg\min_{2^i \geq 6^j} \frac{2^i - 6^j}{2^i}$$

But we can prove

$$0 = \inf_{2^i \geq 6^j} \frac{2^i - 6^j}{2^i}$$

and inf is not achievable. So personally I think $i = 3, j = 1$ is a good choice.

### Question 17

The program is the below:

```
while(not at-zero):
    Go left
    Go right
    Go left

while True:
    Go left
```

## 3 Reference

[1] http://faculty.wwu.edu/sarkara/dirac.pdf

[2] https://math.stackexchange.com/questions/13959/if-a-1-meter-rope-is-cut-at-two-uniformly-randomly-chosen-points-what-is-the-av