Qishi Quiz 3 Solution

Liao Zhu

1 Math

Question 1

This can be calculated by the indicator. For a more general case, we have m boys and n girls. Denote I_i to be the number of girls adjacent to i^{th} boy. Then we have

$$P(I_i = 2) = \frac{n(n-1)}{(n+m-1)(n+m-2)} \tag{1}$$

$$P(I_i = 1) = \frac{2n(m-1)}{(n+m-1)(n+m-2)} \tag{2}$$

$$P(I_i = 0) = \frac{m(m-1)}{(n+m-1)(n+m-2)} \tag{3}$$

So the expectation of boy-girl pair

$$E(\sum_{i=1}^{m} I_i) = \sum_{i=1}^{m} E(I_i) = \frac{2mn(n+m-2)}{(n+m-1)(n+m-2)} \tag{4}$$

Question 2

Independence implies uncorrelated, but uncorrelated does not imply independence. So independence is stronger. For example, for $X \sim Uniform(-1,1)$, X^2 and X are uncorrelated, but they are not independent.

Question 3

We solve this reversely.

$$E_3 = \frac{1}{6}\sum_{i=1}^{6} i = \frac{7}{2} \tag{5}$$

$$E_2 = \frac{1}{6}(4+5+6) + \frac{1}{2}E_3 = \frac{17}{4} \tag{6}$$

$$E_1 = \frac{1}{6}(5+6) + \frac{2}{3}E_2 = \frac{14}{3}. \tag{7}$$

The value is $\dfrac{14}{3}$.

Question 4

Question 5

The matrix $cov(X, Y, Z)$ is positive semi-definite. Calculate the principal minors to have

$$1 - r^2 \geq 0 \tag{8}$$
$$1 - 2r^2 \geq 0 \tag{9}$$

the results are $-\frac{\sqrt{2}}{2} \leq r \leq \frac{\sqrt{2}}{2}$.

Question 6

1. Consider a random walk X_t starting from 0 and absorbing at -1 or 99. τ is the stopping time it first hit -1 or 99. Then X_t and $X_t^2 - t$ are two martingales, therefore,

$$E[X_\tau] = 0 \tag{10}$$
$$E[X_\tau^2 - \tau] = 0 \tag{11}$$

   which implies

$$P[X_\tau = 99] = \frac{1}{100} \tag{12}$$
$$P[X_\tau = -1] = \frac{99}{100} \tag{13}$$
$$E[\tau] = E[X_\tau^2] = \frac{99}{100} \times 1 + \frac{1}{100} \times 99^2 = 99 \tag{14}$$

2. Now -1 is not absorbing. If it hits -1, with 0.5 probability it will stay there, and 0.5 probability it will go right. Denote T be the steps it takes to go to 99 from 0, and S by the steps it takes to go to 99 from -1. Condition on the first step from -1, we have

$$ES = 0.5ES + 0.5ET + 1 \tag{15}$$

   condition on the X_τ we have

$$ET = E(E(T|X_\tau)) = E(P(X_\tau = -1)(\tau + S) + P(X_\tau = 99)\tau) \tag{16}$$
$$= E\tau + P(X_\tau = -1)ES \tag{17}$$
$$= 99 + \frac{99}{100}ES \tag{18}$$

   By these two equations, we know that $ET = 10098$.

Question 7

In the setting $Y = X\beta + \epsilon$, the ridge estimator is

$$\hat{\beta}_{Ridge} = \underset{\beta}{argmin}(\|Y - X\beta\|_2 + \lambda \|\beta\|_2) \tag{19}$$
$$= (\lambda I + X'X)^{-1}X'Y \tag{20}$$

This solves the problem when $X'X$ is singular. In low dimension, singular $X'X$ is caused by colinearity. In high dimension, $X'X$ is always singular. Ridge regression shrinks the β with a fixed rate and reduces over-fitting. However it does not gives a sparse solution.

Question 8

In the setting $Y = X\beta + \epsilon$, the lasso estimator is

$$\hat{\beta}_{Ridge} = \underset{\beta}{argmin}(\|Y - X\beta\|_2 + \lambda \|\beta\|_1) \tag{21}$$

Lasso regression is used when the parameter β is sparse and high dimension.

Question 9

We interpret the question as "all first i tosses are head", otherwise one will pay any finite money to play the game since there is nothing to lose and the gain is exponential. 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 ∞. To decide how much money we would like to invest into this game, we propose the following approach. Take R_α to be the upper-α quantile of the return,

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

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

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

and α here can be chosen by individuals due to their own risk aversion.

2 Programming

Question 10

Question 11

In python, we can use the following code

```
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

```
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.

```
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.

```
#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

Start from a seed X_0 and generate random numbers by

$$X_{n+1} = (aX_n + b) \mod m \tag{22}$$

where a, b are large integers, and $(a, b, m) = 1$. This will generate random numbers from 0 to m - 1. To measure the quality of random numbers, we can use statistic checks. For example, use auto-correlation Ljung-box test to measure dependency, and use chi-square test to measure uniformality.

Question 16

The is equivalent to find:

$$i, j = \underset{2^i \geq 6^j}{argmin} \frac{2^i - 6^j}{2^i} \tag{23}$$

However, this value has a limit 0 when j goes to infinity (can be proved by pigeon hole rule after taking the log), so there is no minimizer. Several good choices of (i, j) can be (3, 1), (13, 5).

Question 17

The program is the below:

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

while True:
    Go left
```