Quiz 2

Math/Stat

1. χ^2 test

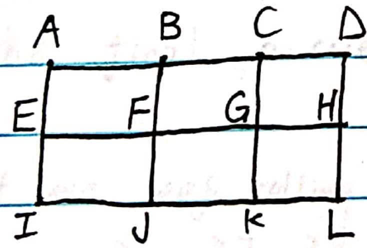
Let's say we have k variables $Z_1, Z_2, \cdots, Z_k$ where $Z_i \sim N(0,1)$ and they are i.i.d. Them the sum $X = \sum_{i=1}^{k} Z_i^2$ follows χ_k^2 distribution.

Pearson's χ^2 test

Assume we have n observations from a random sample. We classify those points into k classes. If the ~~i-t~~ occurance probability of i-th class is p_i and the number of points in i-th class is X_i, we have

$$\sum_{i=1}^{k} \frac{(X_i - np_i)^2}{np_i} \sim \chi_{k-1}^2$$

2. Euler path means a walk through the graph which uses every edge exactly once. And there is a conclusion that a graph has an Euler path if and only if there are at most two vertices with odd degree.



From the graph above, we have ~~vertexs~~ vertices B, C, E, H, J, K in total six points. So in order to make it a new graph that an ~~eu~~ Euler path exists, we can draw ~~e~~ edges BC and JK to make the degree of these vertices even. So we can now calculate the length of the ~~shortest~~ shortest path as $15+2\times2=19$ since we

Scanned by CamScanner

need to pass BC and JK twice.
One possible solution is

$E \to A \to B \to C \to B \to F \to E \to I \to J \to K \to J \to F \to \cancel{G}$
$\to G \to C \to D \to H \to G \to K \to L \to H$

3. $P(X>a \mid X+Y>0)$

$$= \frac{P(X>a,\ X+Y>0)}{P(X+Y>0)}$$

$P(X+Y>0) = 0.5$ because of symmentry

To calculate $P(X>a,\ X+Y>0)$

we have $P(X>a,\ X+Y>0)$

$$= \int_a^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \int_{-x}^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dx$$

$$= \int_a^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} N(x)\, dx$$

$$= \int_a^\infty N'(x) N(x)\, dx$$

$$= \frac{1}{2} N(x)^2 \Big|_a^\infty$$

$$= \frac{1}{2}(1 - N^2(a))$$

So the final answer is $1 - N^2(a)$

4. (1) If we keep rolling, after the first roll, our payment will be

sum	36	37	38	39	40	41
payment	0	37	38	39	40	41
P	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

And for those last five ~~six~~ cases, we can roll one more time since we will at most get 47, which is smaller than the next square number 49.

So our payment will be larger than

$\frac{1}{6}(37+38+39+40+41) + 3.5 \cdot \frac{5}{6} = 35.417$

So we will keep rolling.

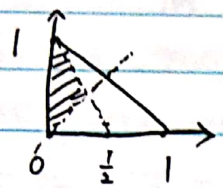
(2) Let's ~~assu~~ denote the sum we get before our last throw as X. We can list the conditional probability of having Y as the final sum conditioning on X as below.

X\Y	43	44	45	46	47	48
42	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
41	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	0
40	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0
39	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	0	0
38	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0
37	1	0	0	0	0	0

So as we can see, 43 is the most probable amount of dollar I win when I stop.

(3)

5 Let's denote the length of first and second part as x and y relatively. We can take a look at the case when x is the smallest



$x+y<1$, $x>0$, $y>0$

$x<y$

$x<1-x-y$

We can calculate the expectation of x as below

$$E(x) = \int_0^{\frac{1}{3}} \frac{1}{\frac{1}{2}} \int_x^{1-2x} x \, dy \, dx$$

$$= 2\int_0^{\frac{1}{3}} (1-3x)x \, dx$$

$$\cancel{-\frac{x}{3}(1-3x)^2 \Big|_0^{\frac{1}{3}}}$$

$$\cancel{= \frac{1}{3}}$$

$$= 2\left(\frac{1}{2}x^2 - x^3\right)\Big|_0^{\frac{1}{3}}$$

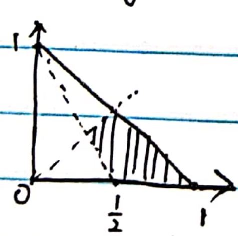
$$= \frac{1}{27}$$

So the average size of the smallest is $\frac{1}{27} \cdot 3 = \frac{1}{9}$

~~Because of symmentry, we know the average o size of the middle-sized is~~ $\cancel{\frac{1}{3}}$

~~The average size of the largest piece is~~ $\cancel{1-\frac{1}{9}-\frac{1}{3}=\frac{5}{9}}$

We can now take a look at the case when x is the largest



$x+y<1$, $x>0$, $y>0$

$x>y$

$x>1-x-y$

$$E(x) = 2\left(\int_{\frac{1}{3}}^{\frac{1}{2}} \int_{1-2x}^{x} x \, dy \, dx + \int_{\frac{1}{2}}^{1} \int_0^{1-x} x \, dy \, dx\right)$$

$$= 2\left(\int_{\frac{1}{3}}^{\frac{1}{2}} (3x-1)x \, dx + \int_{\frac{1}{2}}^{1} (1-x)x \, dx\right)$$

$$= 2\left[\left(x^3 - \frac{1}{2}x^2\right)\Big|_{\frac{1}{3}}^{\frac{1}{2}} + \left(\frac{1}{2}x^2 - \frac{1}{3}x^3\right)\Big|_{\frac{1}{2}}^{1}\right]$$

$$= 2 \cdot \frac{11}{108} = \frac{11}{54}$$

So the average size of the largest is $\frac{11}{54} \cdot 3 = \frac{11}{18}$

The average size of the middle-sized is

$1 - \frac{11}{18} - \frac{1}{9} = \frac{5}{18}$

6 (1) Let's denote Y_i as below

$$Y_i = \begin{cases} 1 & \text{if } i\text{-th person gets his own hat} \\ 0 & \text{else} \end{cases}$$

$$Y = \sum_i Y_i$$

$$E(Y) = E(\Sigma Y_i) = \sum_i E(Y_i) = N \cdot \frac{1}{N} = 1$$

(2) $$E(Y^2) = \sum_i E(Y_i^2) + \sum_i \sum_{j \neq i} E(Y_i Y_j)$$

$$= N \cdot \frac{1}{N} + N(N-1)\frac{1}{N} \cdot \frac{1}{N-1}$$

$$= 2$$

$$Var(Y) = E(Y^2) - E^2(Y) = 1$$

(3)

7 Intuitively, the lower bound of R^2 will be the case when X_1 is orthogonal to the residual of Model 2 $-(Y, X_2)$

$R^2_{min} = R_2^2 = 0.2$

The upper bound of R^2 will be the case when X_1 is orthogonal to X_2

$R^2_{max} = R_1^2 + R_2^2 = 0.1 + 0.2 = 0.3$

So $0.2 \leq R^2 \leq 0.3$

8. Given n, we can find integer k where $2^{k-1} < n \leq 2^k$. Then we throw the fair coin k times and record the result series (if head we record 1, tail record 0). Then we convert the binary number to a decimal number. If the result is 0, we return head. If the result is larger than 1 and smaller than n, we return tail. Else we repeat the steps above until we return a valid result.

9. Let X denote the average number of bridges that the man has to cross.

We have

$$X = \sum_{i=1}^{9} \left(\frac{1}{2}\right)^i (i+X) + \left(\frac{1}{2}\right)^9 \cdot 9$$

$$2^9 X = \sum_{i=1}^{9} 2^{9-i} (i+X) + 9$$

$$\Rightarrow X = 2^{10} - 2 - 9 + 9$$

$$= 2^{10} - 2$$

$$= 1022$$

Scanned by CamScanner

quiz2_code

June 10, 2019

0.0.1 10

The constant function named “fun” takes a constant pointer pointing to a constant interger as the parameter and returns a constant pointer pointing to a constant interger.

0.0.2 11

```
In [7]: class Node:
            def __init__(self, val):
                self.val = val
                self.next = None

In [10]: class Single_List:
             def __init__(self):
                 self.head = None

             def __init__(self, num_list):
                 self.head = Node(num_list[0])
                 cur = self.head
                 for num in num_list[1:]:
                     node = Node(num)
                     cur.next = node
                     cur = node

             def delete(self, val):
                 if self.head.val == val:
                     self.head = self.head.next
                 else:
                     cur = self.head
                     while cur.next:
                         if cur.next.val == val:
                             break
                         cur = cur.next
                     if cur.next:
                         cur.next = cur.next.next
                     else:
                         print("Can't find the value!!!")
```

```
    def print_list(self):
        cur = self.head
        num_list = []
        while cur:
            num_list.append(cur.val)
            cur = cur.next
        print(num_list)
```

```
In [11]: if __name__ == "__main__":
             num_list = [1,2,3,4,5,6,7,8]
             single_list = Single_List(num_list)
             single_list.print_list()
             single_list.delete(5)
             single_list.print_list()
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 6, 7, 8]
```

0.0.3 12

```
In [ ]: #include <iostream>
        #include <vector>
        using namespace std;

        class Matrix{
        private:
            vector<vector<double>> mat;
        public:
            Matrix(int n, int m, double a[]){
                for (int i=0;i<n;i++)
                    {
                        vector<double> k;
                        for (int j=0;j<m;j++)
                            k.push_back(a[i*n+j]);
                        mat.push_back(k);
                    }
            }

            void set(int i, int j, double t){
                mat[i-1][j-1] = t;
            }

            double get(int i, int j){
                return mat[i-1][j-1];
            }
        };
```

```
int main(){
    double num[]={1000, 2, 3, 17, 50, 20};
    Matrix mat = Matrix(3,2,num);
    cout << mat.get(1,2) << endl;
    mat.set(1,2,5);
    cout << mat.get(1,2) << endl;
    return 0;
}
```

0.0.4 13

The constructor can't be virtual because we need to know the type of the return variable when we run the constructor.

0.0.5 14

It is okay since the function to be called is decided at run-time using the vptr and vtable.

0.0.6 15

```
In [12]: class Solution(object):
             def longestPalindromeSubseq(self, s):
                 """
                 :type s: str
                 :rtype: int
                 """
                 num_list = [[0 for j in range(len(s))] for i in range(len(s))]
                 for i in range(len(s)-1, -1, -1):
                     for j in range(i,len(s)):
                         if i==j:
                             num_list[i][j] = 1
                         else:
                             num_list[i][j] = max(num_list[i+1][j], num_list[i][j-1], num_list
                 return num_list[0][len(s)-1]
```

0.0.7 16

```
In [13]: class Solution(object):
             def reverseWords(self, s):
                 """
                 :type s: str
                 :rtype: str
                 """
                 word_list = s.split(' ')
                 word_list = [word for word in word_list if word != '']
                 return ' '.join(word_list[::-1])
```

0.0.8 17

```python
In [14]: class Solution(object):
             def maxProfit(self, prices):
                 """
                 :type prices: List[int]
                 :rtype: int
                 """
                 if len(prices)==0:
                     return 0
                 profit1 = float('-inf')
                 profit2 = 0
                 profit3 = float('-inf')
                 profit4 = 0
                 for x in prices:
                     profit1 = max(profit1, -x)
                     profit2 = max(profit2, profit1+x)
                     profit3 = max(profit3, profit2-x)
                     profit4 = max(profit4, profit3+x)
                 return profit4
```

0.0.9 18

```python
In [15]: class Solution(object):
             def findCircleBook(self, N, mat):
                 """
                 :type M: List[List[int]]
                 :rtype: int
                 """
                 self.final_result = []
                 self.N = N
                 cur = 1
                 marked = {}
                 result_list = [1]
                 self.mat = mat
                 self.DFS(cur, marked, result_list)
                 if self.final_result:
                     return self.final_result[0]
                 else:
                     return 'No solution'

             def DFS(self, cur, marked, result_list):
                 for i in self.mat[cur-1]:
                     if marked.get(i, 0)==0:
                         marked[i] = 1
                         result_list.append(i)
                         if len(marked) == self.N and i==1:
                             self.final_result.append(tuple(result_list))
```

```
                    elif len(marked) < self.N and i != 1:
                        self.DFS(i, marked, result_list)
                    del marked[i]
                    result_list.pop(-1)

if __name__ == "__main__":
    N = 4
    mat = [[2,3], [3,4], [1,4], [3]]
    sol = Solution()
    print(sol.findCircleBook(N, mat))
```

```
(1, 2, 4, 3, 1)
```