Solutions to Qishi Quiz Two

Si Chen, Yupeng Li, Jie Wang, Xian Xu and Yuanda Xu

October 12, 2015

1 Math/Stat

Problem 1 (Yupeng Li)

A χ^2 test (or chi-square test), is to test whether the sampling distribution is a chi-square distribution with the null hypothesis is true. A chi-squared test can then be used to reject the hypothesis that the data are independent.

The chi-squared distribution (also chi-square or χ^2-distribution) with k degrees of freedom is the distribution of a sum of the squares of k independent standard normal random variables Z. $Q = \sum_i Z_i^2$.

Due to the central limit theorem, the sum or average of random variables are iid and normally distributed, which is often used as the test samples, and makes the chi-squared test valid in many cases.

Problem 2 (Jie Wang)

Recall a well known fact that one can visit each edge of a (connected) graph G **exactly once** if and only if the number of vertices of odd degree is either 0 or 2 (the degree of a vertex v of G is the number of edges passing through that vertex in question). Our graph G in question has 6 vertices of degree 3, namely T_1, T_2 on the top side, B_1, B_2 one the bottom side, L on the left side and R on the right side. Thus there is no way to find a path to visit all edges exactly once. However, if we make two auxiliary edges connecting T_1, T_2 and B_1, B_2, then there are only two vertices of degree 3 left, namely L and R. One can find a path to visiting each edge of the new graph G' exactly once. This path has length 19 since the new graph has 19 edges. This means on the original graph G, there exists a path of length 19 to visit all edges (the path will visit edges T_1T_2 and B_1B_2 twice and all other edges once). It is easy to show this is the shortest path possible.

Problem 3 (Si Chen)

It follows that

$$\begin{aligned}
\Pr(X \le x | X+Y > 0) &= 2 * \Pr(X \le x, X+Y > 0) \\
&= 2 * \int_{-\infty}^{x} \Pr(X+Y > 0 | X = z)\phi(z)dz \\
&= 2 * \int_{-\infty}^{x} \Phi(z)\phi(z)dz.
\end{aligned}$$

Therefore the density of $X|X+Y > 0$ is

$$2\Phi(x)\phi(x) = \frac{1}{\pi} e^{-\frac{x^2}{2}} \int_{-\infty}^{x} e^{-\frac{y^2}{2}} dy.$$

Problem 4 (Yupeng Li)

Problem 5 (Jie Wang)

Let X and Y be the random variable representing two of pieces, the thrid piece has length $1 - X - Y$. Thus X and Y are i.i.d with $\mathcal{U}(0,1)$ distribution. We compute the c.d.f $F_{\max}$ of $\max X, Y, 1-X-Y$ as follows:

$$
\begin{aligned}
F_{\max}(t): &= \mathbb{P}(\max\{X, Y, 1-X-Y\} \le t \mid X+Y \le 1) \\
&= \mathbb{P}(X \le t,\ Y \le t,\ 1-X-Y \le t \mid X+Y \le 1) \\
&= \frac{\mathbb{P}(X \le t, Y \le t, 1-t \le X+Y \le 1)}{\mathbb{P}(X+Y \le 1)}
\end{aligned}
$$

We easily compute that

$$
F_{\max}(t) = \begin{cases} 0 & t \le \frac{1}{3}, \\ (3t-1)^2 & \frac{1}{3} < t \le \frac{1}{2}, \\ 1 - 3(1-t)^2 & \frac{1}{2} < t \le 1. \end{cases}
$$

We further compute the density function and then the expectation

$$
\mathbb{E}_{\max} = \frac{11}{18}.
$$

One can similarly compute the c.m.f of $\min\{X, Y, 1-X-Y\}$:

$$
F_{\min}(t) = \begin{cases} 1 - (1-3t)^2 & 0 \le t \le \frac{1}{3} \\ 1 & t > \frac{1}{3}. \end{cases}
$$

The expectation is $\mathbb{E}_{\min} = \frac{1}{9}$. The middle piece therefore has expectation $\frac{5}{18}$.

Problem 6 (Xian Xu)

(1)

Let X_i denote an indicator variable which is equal to 1 if the ith person picks his/her own hat. Clearly,

$$
\mathrm{E}(X_i) = \Pr(X_i = 1) = \frac{(N-1)!}{N!} = \frac{1}{N}.
$$

Therefore

$$
\mathrm{E}(Y) = \sum_{i=1}^{N} \mathrm{E}(X_i) = N * \frac{1}{N} = 1.
$$

(2)

For person i and j, the covariance between X_i and X_j is

$$
\begin{aligned}
\mathrm{Cov}(X_i, X_j) &= \mathrm{E}(X_i X_j) - \mathrm{E}(X_i)\mathrm{E}(X_j) \\
&= \Pr(X_i = 1, X_j = 1) - \frac{1}{N^2} \\
&= \frac{(N-2)!}{N!} - \frac{1}{N^2} \\
&= \frac{1}{N^2(N-1)}.
\end{aligned}
$$

Therefore

$$
\mathrm{Var}(Y) = \sum_{i=1}^{N} \mathrm{Var}(X_i) + 2 \sum_{1 \le i < j \le N} \mathrm{Cov}(X_i, X_j) = N * \left(\frac{1}{N} - \frac{1}{N^2}\right) + N(N-1) * \frac{1}{N^2(N-1)} = 1.
$$

(3)

Start from simple cases: $\mathrm{E}[R(2)] = 2$; $\mathrm{E}[R(3)] = 1 + \frac{1}{3}\mathrm{E}[R(3)] + \frac{1}{2}\mathrm{E}[R(2)]$,which gives us $\mathrm{E}[R(3)] = 3$. Therefore my educated guess for the general case would be $\mathrm{E}[R(N)] = N$, and I'll prove that by induction.

Suppose the conclusion holds for all $n < N$. When $n = N$, let's continue to use Y as the number of people who select their own hats at the first round. Then it follows from the conditional-expectation formula that

$$\begin{aligned}
\mathrm{E}[R(N)] &= \sum_{k=0}^{N} \mathrm{E}[R(N)|Y = k]\mathrm{Pr}(Y = k) \\
&= 1 + \mathrm{Pr}(Y = 0) * \mathrm{E}[R(N)] + \sum_{k=1}^{N} \mathrm{E}[R(N-k)]\mathrm{Pr}(Y = k) \\
&= 1 + \mathrm{Pr}(Y = 0) * \mathrm{E}[R(N)] + \sum_{k=1}^{N} (N-k)\mathrm{Pr}(Y = k) \\
&= 1 + \mathrm{Pr}(Y = 0) * \mathrm{E}[R(N)] + [1 - \mathrm{Pr}(Y = 0)] * N - \sum_{k=0}^{N} k\mathrm{Pr}(Y = k) \\
&= 1 + \mathrm{Pr}(Y = 0) * \mathrm{E}[R(N)] + [1 - \mathrm{Pr}(Y = 0)] * N - \mathrm{E}(Y) \\
&= \mathrm{Pr}(Y = 0) * \mathrm{E}[R(N)] + [1 - \mathrm{Pr}(Y = 0)] * N,
\end{aligned}$$

where the last equality follows from part (1). Now it is super clear that $\mathrm{E}[R(N)] = N$. QED.

(4)

Again let's start from simple cases: $\mathrm{E}[S(2)] = 4 = \frac{1}{2}*2*(2{+}2)$; $\mathrm{E}[S(3)] = 3{+}\frac{1}{3}\mathrm{E}[S(3)]{+}\frac{1}{2}\mathrm{E}[S(2)] = \frac{10}{3} = \frac{1}{2}*3*(3{+}2)$. Therefore my educated guess for the general case would be $\mathrm{E}[S(N)] = \frac{1}{2}N(N+2)$, and I'll still prove that by induction.

Suppose the conclusion holds for all $n < N$. When $n = N$, it follows from the conditional-expectation formula that

$$\begin{aligned}
\mathrm{E}[S(N)] &= \sum_{k=0}^{N} \mathrm{E}[S(N)|Y = k]\mathrm{Pr}(Y = k) \\
&= N + \mathrm{Pr}(Y = 0) * \mathrm{E}[S(N)] + \sum_{k=1}^{N} \mathrm{E}[S(N-k)]\mathrm{Pr}(Y = k) \\
&= N + \mathrm{Pr}(Y = 0) * \mathrm{E}[S(N)] + \sum_{k=1}^{N} \frac{1}{2}(N-k)(N-k+2)\mathrm{Pr}(Y = k) \\
&= N + \mathrm{Pr}(Y = 0) * \mathrm{E}[S(N)] + [1 - \mathrm{Pr}(Y = 0)] * \frac{1}{2}N(N+2) - (N+1) * \mathrm{E}(Y) + \frac{1}{2} * \mathrm{Var}(Y) \\
&= \mathrm{Pr}(Y = 0) * \mathrm{E}[S(N)] + [1 - \mathrm{Pr}(Y = 0)] * \frac{1}{2}N(N+2),
\end{aligned}$$

where the last equality follows from part (1) and (2). Now it is super clear that $\mathrm{E}[S(N)] = \frac{1}{2}N(N+2)$. QED.

(5)

The expected number of false selections made by one of the N people is equal to

$$\frac{\mathrm{E}[S(N)] - N}{N} = \frac{1}{2}N.$$

Problem 7 (Si Chen)

Introducing a new feature into linear regression will not decrease the R^2 value, so the R^2 of model 3 should not be smaller than 0.2. If X_1 and X_2 are totally uncorrelated, R^2 will be 0.3. The range should be [0.2,0.3]

Problem 8 (Xian Xu)

This solution is motivated by `http://jeremykun.com/2014/02/12/simulating-a-biased-coin-with-a-fair-coin/`. The Python code is presented as follows:

```
def biasedCoin(binaryDigitStream, fairCoin):
    for d in binaryDigitStream:
        if fairCoin() != d:
            return d

def binaryDigitsStream(fraction):
    while True:
        fraction *= 2
        yield int(fraction)
        fraction = fraction % 1

def fairCoin():
    return random.choice([0,1])
```

The main function biasedCoin() takes two arguments: one is binaryDigitsStream(), an iterator representing the binary expansion of probability $\frac{1}{n}$; the other is fairCoin(), which returns 1 or 0 with equal probability.

The biasedCoin() function returns 1(heads) with probability $\frac{1}{n}$, and the reasoning is as follows: we use fairCoin() to generate a series of random bits, until one of our random bits is different from the corresponding bit in the binary expansion of $\frac{1}{n}$. If we stop after i steps, that means that the first $i-1$ bits in the two binary sequences were the same, which happens with probability $\frac{1}{2^{i-1}}$. Given that this happens, in the ith step we will return the ith bit of $\frac{1}{n}$; let us denote this bit by b_i. Therefore the probability of returning 1 is $\sum_{i=1}^{\infty} \frac{b_i}{2^i}$, which is the binary expansion of $\frac{1}{n}$.

Problem 9 (Yuanda Xu)

2 Programming

Problem 10 (Yupeng Li)

This is about the const in variable, function and class members.

The first const int *: the fun return type is a pointer of int type and the returned int value is constant.

The second const fun(): returned pointer of fun is constant, meaning the value of pointer cann't be changed.

The third const int*: like the first, argument is a pointer to int type and the value of int is constant.

The forth const& p: p is a pointer with same reference as the passed variable and it's value cann't be change.

The fifth const: the fun member in the class is a constantt member function, saying it cann"t change any class member in the fun function and the object which use this fun function will be required as constant object too.

Problem 11 (Si Chen)

Idea comes from `http://www.algolist.net/Data_structures/Singly-linked_list/Removal`. There are three cases, which can occur while removing the node:

a. Remove first. It can be done in two steps:
   - Update head link to point to the node, next to the head.
   - Dispose removed node.

b. Remove last. In this case, last node (current tail node) is removed from the list. This operation is a bit more tricky, than removing the first node, because algorithm should find a node, which is previous to the tail first. It can be done in three steps:
   - Update tail link to point to the node, before the tail. In order to find it, list should be traversed first, beginning from the head.
   - Set next link of the new tail to NULL.
   - Dispose removed node.

c. General case.
   - Update next link of the previous node, to point to the next node, relative to the removed node.
   - Dispose removed node.

The C++ code is as follows:

```
void SinglyLinkedList::removeFirst() {
    if (head == NULL) return;
    else {
        SinglyLinkedListNode *removedNode;
        removedNode = head;
        if (head == tail) {
            head = NULL;
            tail = NULL;
        } else {
            head = head->next;
        }
        delete removedNode;
     }
}

void SinglyLinkedList::removeLast() {
    if (tail == NULL) return;
    else {
        SinglyLinkedListNode *removedNode;
        removedNode = tail;
```

```
        if (head == tail) {
            head = NULL;
            tail = NULL;
        } else {
            SinglyLinkedListNode *previousToTail = head;
            while (previousToTail->next != tail)
            previousToTail = previousToTail->next;
            tail = previousToTail;
            tail->next = NULL;
        }
        delete removedNode;
    }
}

void SinglyLinkedList::removeNext(SinglyLinkedListNode *previous) {
    if (previous == NULL) removeFirst();
    else if (previous->next == tail) {
        SinglyLinkedListNode *removedNode = previous->next;
        tail = previous;
        tail->next = NULL;
        delete removedNode;
    } else if (previous == tail) return;
    else {
        SinglyLinkedListNode *removedNode = previous->next;
        previous->next = removedNode->next;
        delete removedNode;
    }
}
```

Problem 12 (Jie Wang)

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

template<typename T>

class Matrix
{
public:
Matrix(int A, int B, T t):RowNumber(A),ColNumber(B) //constructor innitialize A*B matrix with entries t.
{
MyMatrix.resize(A);
for(int i=0;i<A;i++)
MyMatrix[i].resize(B,t);


}
Matrix(const Matrix& OneMatrix):RowNumber(OneMatrix.RowNumber),ColNumber(OneMatrix.ColNumber) //copy
    constructor
{
MyMatrix.resize(RowNumber);
for(int i=0;i<RowNumber;i++)
MyMatrix[i].resize(ColNumber);
for(int i;i<RowNumber;i++)
for(int j=0;j<ColNumber;j++)
MyMatrix[i][j]=OneMatrix.MyMatrix[i][j];
}
```

```
Matrix& operator=(const Matrix& OneMatrix) //overloading assignment operator
{
if(RowNumber!=OneMatrix.RowNumber||ColNumber!=OneMatrix.ColNumber)
{cout<<"The matrices do not match."<<endl;
throw -1;
}
for(int i=0;i<RowNumber;i++)
for(int j=0;j<ColNumber;j++)
MyMatrix[i][j]=OneMatrix.MyMatrix[i][j];
return *this;


}

Matrix operator+(const Matrix& AnotherMatrix) //overloading +
{
if(RowNumber!=AnotherMatrix.RowNumber||ColNumber!=AnotherMatrix.ColNumber)
{cout<<"The matrices do not match."<<endl;
throw -1 ;
}
else
{Matrix A(RowNumber,ColNumber,0);
for(int i=0;i<RowNumber;i++)
for(int j=0;j<ColNumber;j++)
A.MyMatrix[i][j]=MyMatrix[i][j]+AnotherMatrix.MyMatrix[i][j];
return A;
}

}

Matrix operator-(const Matrix& AnotherMatrix) //overloading -
{
if(RowNumber!=AnotherMatrix.RowNumber||ColNumber!=AnotherMatrix.ColNumber)
{cout<<"The matrices do not match."<<endl;
throw -1 ;
}
else
{Matrix A(RowNumber,ColNumber,0);
for(int i=0;i<RowNumber;i++)
{ for(int j=0;j<ColNumber;j++)
A.MyMatrix[i][j]=MyMatrix[i][j]-AnotherMatrix.MyMatrix[i][j];
}
return A;
}

}
Matrix operator*(const Matrix& AnotherMatrix) //overloading matrix multiplication
{
if(ColNumber!=AnotherMatrix.RowNumber)
{throw "The matrices can not be multiplied";

}
Matrix A(RowNumber,AnotherMatrix.ColNumber,0);
for(int i=0;i<RowNumber;i++)
for(int j=0;j<AnotherMatrix.ColNumber;j++)
{ for(int k=0;k<ColNumber;k++)
A.MyMatrix[i][j]+=MyMatrix[i][k]*AnotherMatrix.MyMatrix[k][j];
}
return A;

}
```

```
T& operator()(int A, int B)
{   if(A>=RowNumber||B>=ColNumber||A<0||B<0)

throw "Position not available";


return MyMatrix[A][B];
}

void print()
{
for(int i=0;i<RowNumber;i++)
{ cout<<"\n";
for(int j=0;j<ColNumber;j++)
cout<<MyMatrix[i][j]<<'\t';}
cout<<'\n';
}
private:
int RowNumber;
int ColNumber;
vector<vector<T>> MyMatrix;
};
```

Problem 13 (Xian Xu)

The answer to first part is NO. When calling a constructor, the caller needs to know the exact type of the object to be created, and thus they cannot be virtual.

That being said, it is still possible to realize a similar function as a virtual constructor. See the following two URLs for implementation details:

- http://www.geeksforgeeks.org/advanced-c-virtual-constructor/
- http://www.geeksforgeeks.org/advanced-c-virtual-copy-constructor/

Problem 14 (Yuanda Xu)

Problem 15 (Yupeng Li)

```
string getmaxPalindrome(string s) {
    bool isPalindrome[s.size()][s.size()];

    int maxlen = 0, maxst=0;
    for (int i = 0; i < s.size(); i++) {
        isPalindrome[i][i] = true;
    }
    for (int i = 0; i < s.size() - 1; i++) {
        isPalindrome[i][i + 1] = (s[i] == s[i + 1]);
        if (isPalindrome[i][i+1]) {maxlen = 1;maxst=i;}
    }

    for (int length = 2; length < s.size(); length++) {
        for (int start = 0; start + length < s.size(); start++) {
            isPalindrome[start][start + length] = isPalindrome[start + 1][start + length - 1] && s[start]
                 == s[start + length];
            if (isPalindrome[start][start+length]&& (length > maxlen)) {
                maxlen = length;
                maxst = start;
```

```
        }
      }
    }
    return s.substr(maxst, maxlen+1);
}
```

Problem 16 (Xian Xu)

Assume there are no leading or trailing spaces in the input sentence, and the words are always separated by a single space. The idea is to first reverse each word and then reverse the whole string. Here is a C++ implementation:

```
void reverseWords(string& s) {
    string::iterator begin = s.begin();
    string::iterator end = s.begin();

    // First reverse each word
    while (end != s.end()) {
        if (*end == ' ') {
            reverse(begin,end);
            begin = end + 1;
        }
        end++;
    }
    reverse(begin,end);

    // Then reverse the whole string
    reverse(s.begin(),s.end());
}
```

Problem 17 (Yuanda Xu)

Problem 18