12^{th} TuiJinBan, Quiz 2
Group B: Jiayang Gao *2019.05.27*

1. What do you know about χ^2 test?

   Answer.

   - Application 1: for $X \sim N(\mu, \sigma^2)$, test if $\sigma = \sigma_0$.
     - Test statistic: $(n-1)\frac{s^2}{\sigma_0^2} = \frac{\sum_i (x_i - \bar{x})^2}{\sigma_0^2}$, where s is the sample StdDev $s^2 = \frac{\sum_i (x_i - \bar{x})^2}{n-1}$.
     - degrees of freedom: $n-1$.
   - Application 2: test whether two categorical features are independent.
     - Feature A and B have m and n categories. Test if A, B independent. Null hypothesis is independent.
     - Let O_{ij} be observed frequency for $i \in A$ and $j \in B$. Total N data points. Let $p_i = \sum_j O_{ij}/N$, $q_j = \sum_i O_{ij}/N$.
     - Test statistics:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \frac{(O_{ij} - p_i q_j N)^2}{p_i q_j N}$$

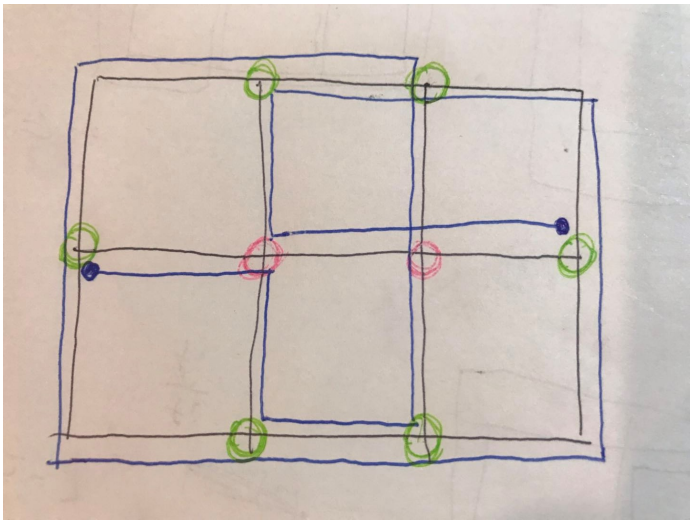
     - degrees of freedom: $(m-1)(n-1)$.
   - Application 3 (generalization): test goodness of fit (whether data comes from a family of distributions)
     - n categories, s number of co-variate for the class of distribution ($s = 2$ for normal and $s = 0$ for discrete uniform distribution.)
     - Test statistics:

$$\sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i},$$

       where O_i is observed frequency of i, E_i is the expected frequency of the distribution with fitted parameters.
     - degrees of freedom: $n - s - 1$.

   □

2. Given a 2 by 3 grid (which has 6 blocks and 17 edges), shortest route to visit all edges (assuming edge length is 1).



Answer. Note that the green nodes have 4 edges, and red nodes have 3 edges. Every visit to a node can cover 2 edges (except the first and the last node, which cover 1 edge.) So

- The green nodes and the red nodes have to be visited at least twice.
- Therefore the shortest path contains at most $12 + 8 = 20$ nodes.

The shortest path satisfy

- The first nodes and the last node have to be green node. (Because they only cover 1 edge, so we have to choose nodes which have odd number edges.)
- Only the edges connected by two green nodes can be visited twice. The green nodes used here don't include the nodes that are used as the first and the last nodes.

With these two criteria, it's not hard to get the shortest path given by the blue line on the plot.

□

3. X, Y are iid $N(0,1)$, calculate $P(X|X+Y>0)$, try not use density function of joint distribution.

Answer.

$$f(X=x|X+Y>0) = \frac{f(X=x, X+Y>0)}{P(X+Y>0)} = \frac{f(X=x)P(Y>-x)}{1/2} = 2\phi(x)\Phi(x).$$

Here, ϕ and Φ are pdf and cdf of $N(0,1)$. □

4. You have a six sided dice, you can keep rolling the dice and you get the dollars equal to the mount of the sum. However, if at some point, the sum is a square number, you must stop and will get zero dollars.

   (1) If at some point, your sum is 35, should you stop or keep rolling?

   (2) in (1), if you choose to continue and this is your strategy: you will keep rolling until you exceed 43, what is the most probable amount of dollar you win when you stop?

   (3) Is there a best strategy for this game, any number that you should stop?

   Answer.

   (1) If continue, then you hit a square number at get 0 if you roll a 1. Otherwise the game won't be forced to stop at the next round, and you won't have the risk of being forced to stop until you get a sum of at least 43 (because the next whole square after 36 is 49.)

   So consider the payoff of the following strategy: "continue this round, and continue until you have a sum at least 43". Then the payoff is at least

   $$\frac{1}{6} * 0 + \frac{5}{6} * 43 = 35.83 > 35.$$

   So there exists a naive strategy which have expected payoff greater than 35, which is the payoff you get if you stop now. So you should keep rolling.

   (2) The answer is 43.

   First, consider the case when first roll doesn't end up at 36. So the sum after the first roll is $S_1 \in \{37, \ldots, 41\}$ with equal probability and the game continues.

   Then the second round gives a sum of $S_2 \in \{38, \ldots, 47\}$, with 43 having the largest probability among the stopped numbers $43, \ldots, 47$. This is because starting from all S_1 numbers, there is $1/6$ possibility of hitting 43. This is not true for $44, \ldots, 47$.

   Similarly we can show that in each round, if the game continues, no matter what the current sum is, there is $1/6$ probability of hitting 43 in the next round. So 43 is the most probable number to stop with if we don't hit 36.

   To compare the probability of 43 with 36, note that the previous argument shows the probability of hitting 43 is strictly greater than $1/6$ (because the probability of hitting 43 in the second round is $1/6$, and there's positive probability of hitting 43 in the later rounds.) While the probability of hitting 36 is simply $1/6$. So 43 has higher probability than 36.

   (3) A good strategy is in the class of "roll until you exceed $k^2 - 6$ and then stop". Because if the current sum is greater than $(k-1)^2$ and less than $k^2 - 6$, the next roll is guaranteed to be safe so we can keep rolling. We now try to determine an upper bound on k.

(Note that this class of strategy may not be the optimal strategy, because the optimal strategy may ask you to stop at k^2-i but continue at k^2-i-1 for $i<6$.)

After we determine an upper bound on k, we can then use dynamic programming to get the exact optimal strategy. (And the upper bound will serve as a boundary condition in the DP.)

Let $V(k)$ be the maximum expected *future* payoff under optimal strategy if you're at $\{k^2-1,\dots k^2-6\}$. We have

$$V(k)\le \max\{0, V(k+1)+\frac{5}{6}((k+1)^2-1-(k^2-6))-\frac{1}{6}(k^2-1)\}.$$

We should definitely stop if $V(k+1)+\frac{5}{6}((k+1)^2-1-(k^2-6))-\frac{1}{6}(k^2-1)<0$. It's easy to show that if $\frac{5}{6}((k+1)^2-1-(k^2-6))-\frac{1}{6}(k^2-1)<0$, then $\frac{5}{6}((k'+1)^2-1-(k'^2-6))-\frac{1}{6}(k'^2-1)<0$ for all $k'>k$. So we should definitely stop if $\frac{5}{6}((k+1)^2-1-(k^2-6))-\frac{1}{6}(k^2-1)<0$. This gives as an upper bound on k, which is 13.

So we have implement at DP to solve this problem, with boundary condition that we should stop when the current number is at 163.

(If this was an interview and I'm not allowed to write code, I can also give a lower bound on k, which solves $\frac{5}{6}((k+1)^2-6-(k^2-1))-\frac{1}{6}(k^2-6)>0$, and suggests that we should continue at $k\le 8$. This is not a perfect answer, though. Because we still don't know what to do for $k=9,\dots,12$.)

□

5. Given a stick, if randomly cut into 3 pieces, whats the average size of the smallest, of the middle-sized, and of the largest pieces?

Answer. Suppose the length of the stick is 1. I answer a generalized question with n pieces, each with length l_i, and cut by $c_{(1)}<c_{(2)}\le c_{(n-1)}$.

For any cut such that $l_{(1)}=x$, consider another cut such that we remove x from all pieces (and the smallest piece disappears), and add xn as the right-most piece. This gives another cut with n pieces.

On the other hand, given a cut such that the right-most pieces has length xn, it can be mapped back to n possible cuts (depends on where you insert the smallest piece) by removing the right-most piece, and add x to each of the other pieces (and create one piece with size x and insert it anywhere).

Therefore there's a n-to-1 mapping between $l_{(1)}=x$ and $l_n=xn$.

The density of $l_n=xn$ is simply

$$f(l_n=xn)=f(c_{n-1}=1-xn)=(n-1)(1-xn)^{(n-2)}.$$

Therefore the density of $l_{(1)}$ is given by

$$f(l_{(1)} = x) = nf(l_n = xn) = n(n-1)(1-xn)^{(n-2)}.$$

So expectation is given by

$$\begin{aligned} E[l_{(1)}] &= \int_0^{1/n} n(n-1)(1-xn)^{(n-2)} x dx \\ &= \left[-\frac{1}{n^2}(1-xn)^n - (1-xn)^{n-1} x \right]_0^{1/n} \\ &= \frac{1}{n^2}. \end{aligned}$$

Using tower property of expectations, we have

$$\begin{aligned} E[l_{(2)}] =& E[E[l_{(2)} - l_{(1)}|l_{(1)}]] + E[l_{(1)}] \\ =& E[\frac{1-nl_{(1)}}{(n-1)^2}] + \frac{1}{n^2} \\ =& \frac{1}{n}[\frac{1}{n-1} + \frac{1}{n}]. \end{aligned}$$

Suppose $E[l_{(k)}] = \frac{1}{n}[\frac{1}{n-k+1} + \ldots + \frac{1}{n}]$, we have for $k+1$

$$\begin{aligned} E[l_{(k+1)}] =& E[E[l_{(k+1)} - l_{(1)}|l_{(1)}]] + E[l_{(1)}] \\ =& E[\frac{1-nl_{(1)}}{n-1}[\frac{1}{n-k} + \ldots + \frac{1}{n-1}]] + \frac{1}{n^2} \\ =& \frac{1}{n}[\frac{1}{n-k} + \ldots + \frac{1}{n-1}] + \frac{1}{n^2} \\ =& \frac{1}{n}[\frac{1}{n-k} + \ldots + \frac{1}{n}] \end{aligned}$$

Thus by induction, the general formula is

$$E[l_{(k)}] = \frac{1}{n}[\frac{1}{n-k+1} + \ldots + \frac{1}{n}].$$

When $n = 3$, we have $E[l_{(1)}] = \frac{1}{9}, E[l_{(2)}] = \frac{5}{18}, E[l_{(3)}] = \frac{11}{18}$.

□

6. At a party, N people throw their hats (all hats are different) into the center of room. The hats are mixed up and each people randomly selects one. Let Y be the number of people who select their own hats. Now ask

(1) what is the expectation of Y ?

(2) what is the variance of Y ?

Now, the picking hats game rule is extended. For each hats pick round, the people choosing their own hats quit the game, while others (those picked wrong hats) put their selected hats back in the center of room, mix them up, and then reselect. Also, suppose that this game continues until each individual has his own hat. Suppose N individuals initially join the game, let $R(N)$ be the number of rounds that are run and $S(N)$ be the total number of selections made by the these N individuals, $(N > 1)$.

(3) Find the expectation of R(N).

(4) Find the expectation of S(N).

(5) Find the expected number of false selections made by one of the N people.

Answer. (1) $E[Y] = \sum_i E[\text{person i selects her own hat}] = \sum_i 1/N = 1$.

(2) Let I_i be the indicator function that person i selects her own hat.

$$Var[Y] = Var[\sum_i I_i] = \sum_i Var(I_i) + \sum_{i \neq j} Cov(I_i, I_j).$$

Here,

$$\begin{aligned} Var(I_i) =& \frac{N-1}{N}(\frac{1}{N})^2 + \frac{1}{N}(1-\frac{1}{N})^2 = \frac{N-1}{N^2} \\ Cov(I_i, I_j) =& \frac{1}{N(N-1)}(1-\frac{1}{N})^2 + \frac{2}{N}\frac{N-2}{N-1}(1-\frac{1}{N})(-\frac{1}{N}) \\ & + (1 - \frac{1}{N(N-1)} - \frac{2}{N}\frac{N-2}{N-1})(-\frac{1}{N})^2 \\ =& \frac{1}{(N-1)N^2} \end{aligned}$$

Therefore

$$Var[Y] = N\frac{N-1}{N^2} + N(N-1)\frac{1}{(N-1)N^2} = 1.$$

(3) We use induction to prove that $R(N) = N$.
First, when $N = 2$, we have

$$R(2) = \frac{1}{2} + \frac{1}{2}(1 + R(2)) \implies R(2) = 2.$$

Now suppose $R(i) = i$ for all $i < k$, let A_1 to denote the number of people quit the game after round 1. Then

$$\begin{aligned}
R(k) =&(1 - P(A_1 > 0))(1 + R(k)) + P(A_1 > 0)E[E[1 - R(k - A_1)|A_1 > 0]|A_1 > 0] \\
=&(1 - P(A_1 > 0))(1 + R(k)) + P(A_1 > 0)(1 - k + E[A_1|A_1 > 0]) \\
=&(1 - P(A_1 > 0))(1 + R(k)) + P(A_1 > 0)(1 - k + \frac{1}{P(A_1 > 0)}) \\
=&(1 - P(A_1 > 0))R(k) + kP(A_1 > 0) \\
\implies & R(k) = k.
\end{aligned}$$

Thus by induction $R(N) = N$.

(4) Using induction similar to (3) we can show $S(N) = \frac{N(N+2)}{2}$.

(5) $\frac{S(N)}{N} - 1 = \frac{N}{2}$.

□

7. Consider linear regression of Y on features $X1, X2$: Model1-$(Y, X1)$, $R^2 = 0.1$; Model2-$(Y, X2)$, $R^2 = 0.2$; Model3-$(Y, X1, X2)$, calculate the range of R^2 of Model3.

Answer. We know that for single regression, $R^2 = \rho(X, Y)$, where $\rho(X, Y)$ is correlation. For regression with two factors, we have

$$R^2 = \frac{\rho(X1, Y)^2 + \rho(X2, Y)^2 - 2\rho(X1, Y)\rho(X2, Y)\rho(X1, X2)}{1 - \rho(X1, X2)}.$$

So we only need to find the range of $\rho(X1, X2)$ given $\rho(X1, Y)$ and $\rho(X2, Y)$.

Using the fact that $\rho(x, y)$ is the cosine between vectors x and y, we can calculate a range for $\rho(X1, X2)$ and thus bound R^2. □

8. Given a function for a fair coin, write a function for a biased coin that returns heads with probability $\frac{1}{n}$ (n is a param).

Answer. Let $0.x_1x_2x_3\ldots$ be a binary expansion for $1/n$. Toss the fair coin and denote $H = 1, T = 0$ and compare result t_i with x_i for $i = 1, 2, \ldots$. Return H if $x_i > t_i$ and return T if $x_i < t_i$. Otherwise, continue. □

9. 10 islands with 9 bridges. The bridges are either strong or weak (half half). Weak bridge falls if stepped on and the man is drifted to the 1st island, then all the bridges are miraculously fixed. To arrive the 10th island, how many bridges on average the man has to cross?

Answer. After the man falls and the bridges are fixed, he will cross exactly 9 bridges. The problem comes down to "how many bridges on average the man crossed before he falls", which is equivalent to "the expected location of the first weak bridge". This is a geometric distribution which we know the mean is $1/p = 2$ ($p = 1/2$ because strong and weak are half-half).

($1/p$ is only an approximation as the geometric distribution is truncated at $n = 9$. But with the small probability of having $n \geq 9$, $1/p$ should be very close to the true solution).

Thus answer is $9 + 2 = 11$. □

10. Explain the following code: const int* const fun(const int* const& p) const;

Answer. I give up this question for now since I only use Python. □

11. How do you implement delete operation in a single-linked list

Answer. Go through the list looking for target value. Keep track of the previous node and the current node. If the target value is found in head, set head to the next node. If the target is found in the middle of the chain, connect the previous node to the next node. □

12. Implement the interface for matrix class in C++.

Answer. I give up this question for now since I only use Python. □

13. Can the constructor of a class be virtual? How to realize a similar function as a virtual constructor?

Answer. I give up this question for now since I only use Python. □

14. Is it okay for a non-virtual function of the base class to call a virtual function?

Answer. I give up this question for now since I only use Python. □

15. Given a string, return the longest palindrome subsequence.

Answer. Use dynamic programming. Let S be the string and use $P(i, j)$ to store the longest palindrome subsequence in $S[i : j + 1]$. When $i = j, P(i, j) = S[i]$. When $i + 1 = j, P(i, j) = S[i : i + 2]$ if $S[i] = S[i + 1]$, else $S[i]$.

When $i < j - 1$ we have the following DP

- If $S[i] = S[j]$, $P(i, j) = S[i] + P(i + 1, j - 1) + S[j]$.

- Otherwise, $P(i,j) = P(i+1,j)$ if $len(i+1,j) > len(i,j-1)$, else $P(i,j) = P(i,j-1)$.

□

16. How to inverse a string of sentence (without reverse the word) ?

    Answer. In python, use

    ' '.join(sentence.split(' ')[::-1]) □

17. Say you have an array for which the i-th element is the price of a given stock on day i. Design an algorithm to find the maximum profit. You may complete at most two transactions.

    Note: You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again).

    Answer. Use dynamic programming to solve this. Let $V(t,h,c)$ be the *future* payoff under optimal strategy at the end of time t when current position is $h \in \{1,0,-1\}$ and the number of transactions left is c.

    Boundary conditions can be easily found by $V(T,*,*) = 0$ and $V(t,h,0) = h*(p_T - p_t)$.

    And for non-boundary cases ($c > 0, t \leq T$) the DP is given by

$$V(t,h=0,c) = \max\{V(t+1,1,c-1), V(t+1,-1,c-1), V(t+1,0,c)\}$$
$$V(t,h=1,c) = \max\{(p_{t+1} - p_t) + V(t+1,1,c), V(t+1,0,c-1)\}$$
$$V(t,h=-1,c) = \max\{(-p_{t+1} + p_t) + V(t+1,-1,c), V(t+1,0,c-1)\}$$

    □

18. The book problem: There is a group of $N(2 \leq N \leq 1000)$ people which are numbered 1 through N, and everyone of them has not less than $[\frac{N+1}{2}]$ friends. A man with number 1 has the book, which others want to read. Write the program which finds a way of transferring the book so that it will visit every man only once, passing from the friend to the friend, and, at last, has come back to the owner.

    Note: if A is a friend of B then B is a friend of A.

    INPUT: First line of input contains number N. Next N lines contain information about friendships. (i+1)-th line of input contains a list of friends of i-th man.

    OUTPUT: If there is no solution then your program must output No solution. Else your program must output exactly N +1 number: this sequence should begin and should come to end by number 1, any two neighbors in it should be friends, and any two elements in it, except for the first and last, should not repeat.

Answer. This is a Hamiltonian path problem (http://www.roard.com/docs/cookbook/cbsu7.html). This problem is NP-complete. We can use the backtracking algorithm to solve it.

The idea is to start with the first node, gradually try to add to path a node such that (1) is not visited in the path yet (2) is a neighbor of the last node in the current path. When all nodes are visited, also check whether the last node is a neighbor of the first node. If so, return the path.

Any time when we fail to find the next node (or the last node is not a neighbor of the first node when you have visited all nodes), back track and remove the last node in the path, and try to replace it by another node. After all nodes are tested and failed in this position, back track again and try to replace the previous node.

□