Data Mining 21/22 Homework 1

Marco Calamo

October 2021

All work is tracked on Github at `https://github.com/IlKaiser/DM22_HW`

1 Exercise 1

1.1 Probability space definition

Let $S = \{h, d, c, s\}$ be the set of card suits. Let $R = \{A, 2, \ldots, 10, J, Q, K\}$ be the set of card ranks. Hence, $C = S \times R$ is the set of cards contained in a Poker deck. Let Ω be the probability space of this process, where every element has probability $\frac{1}{52}$ of being drawn. Therefore,

$$\begin{aligned}\Omega &= Sym(C) \\ &= \text{symmetric group of C} \\ &= \text{collection of all permutations of the elements in C}\end{aligned}$$

By definition of symmetric group, we have that

$$|\Omega| = |C|! = (|S| \cdot |R|)! = 52!$$

Hence we have:

$$\Pr(\omega) = \frac{1}{|\Omega|} = \frac{1}{52!} \qquad \forall\ \omega \in \Omega$$

1.2 Events probabilities

(a) Let $A \subset \Omega$ be the event related to drawing at least 1 ace in the first 2 cards. Note that we can write $P(A)$ as the inverse of not drawing any ace in the first two cards:

$$P(A) = 1 - P(NoAceIn2) = 1 - [\frac{48}{52} \cdot \frac{47}{51}] \tag{1}$$

Since there are 48 cards at the beginning that are not aces, and going downwards since no aces are drawn.
And finally:

$$P(A) \simeq 0.149 \tag{2}$$

(b) Let $B \subset \Omega$ be the event related to drawing at least 1 ace in the first 5 cards. Therefor in a similar way to equality 1,

$$P(B) = 1 - P(NoAceIn5) \simeq 0.341 \tag{3}$$

(c) Let $C \subset \Omega$ be the event related to drawing a "couple" in the first 2 cards. Therefore,

$$P(C) = \frac{13 \cdot \binom{4}{2} \cdot 2! \cdot 50!}{52!} \simeq 0.058$$

since there are:

- 13 different ranks.
- $\binom{4}{2}$ ways to choose 2 cards among those with the same rank.
- 2! permutations for each of the first 2 cards combination.
- 50! permutations of the remaining cards.

(d) Let $D \subset \Omega$ be the event related to drawing only Diamonds in the first 5 cards. Therefore,

$$P(D) = \frac{\binom{13}{5} \cdot 5! \cdot 47!}{52!} \simeq 0.0005$$

since there are:

- $\binom{13}{5}$ ways to choose 5 cards among the Diamonds.
- 5! permutations for each of the first 5 cards combination.
- 47! permutations of the remaining cards.

(e) Let $E \subset \Omega$ be the event related to drawing a "full house" in the first 5 cards. Therefore,

$$P(E) = \frac{13 \cdot 12 \cdot \binom{4}{2} \cdot \binom{4}{3} \cdot 5! \cdot 47!}{52!} \simeq 0.0014 \tag{4}$$

since there are:

- $13 \cdot 12$ different dispositions of 2 (out of 13) ranks.
- $\binom{4}{2}$ ways to choose the suits of the pair.
- $\binom{4}{3}$ ways to choose the suits of the tris.
- 5! permutations for each of the first 5 cards combination.
- 47! permutations of the remaining cards.

1.3 Event simulation in Python

I developed a simple program that simulates extraction from a card deck using *pydealer* library and counting the number of requested sequences appeared per every draw.
For obtaining optimal results about 1×10^6 iterations are necessary (abut three minutes of computing). At the end of computation a graph with *matplotlib* shows how the various probability tend to reach the exact (limit) value after multiple iterations.
For more details see *deck.py*

2 Exercise 2

2.1 Probability Space

Given the set $G = \{b, g\}$ for the possible gender of the two kids (b for boy and g for girl), and given the set $D = \{mo, tu, we, th, fr, sa, su\}$ for the day of the week, the probability space can be defined as :

$$\Omega = \{G \times G \times D \times D\} \tag{5}$$

and the size of Ω is given by:

$$|\Omega| = |G| \times |G| \times |D| \times |D| = 4 \cdot 49 = 196 \tag{6}$$

2.2 Kid is a Girl, given a Girl

We can use the conditional probability definition to obtain:

$$P(G|G) = \frac{P(G \wedge G)}{P(G)} = \frac{49/196}{98/196} = \frac{1/4}{1/2} = \frac{1}{2} \tag{7}$$

Because intuitively the case when both the kids are girl is $\frac{1}{4}$, -two times $\frac{1}{2}$-, and the case when a kid is a girl is just $\frac{1}{2}$, as stated in exercise text.

2.3 Girl given Girl on Sunday

Also in this case we can compute the probability with the conditional probability:

$$p(G|G \wedge SU) = \frac{P(G \wedge G \wedge SU)}{P(G \wedge SU)} = \frac{13/196}{27/196} \simeq 0.48 \tag{8}$$

Contrarily to what it may appear at first glance, the probability of the event is less than one half.

3 Exercise 3

3.1 The strategy with win probability $> \frac{1}{2}$

Before demanding to Andrea or Aris which number he chose, you have to choose a number r at random using any continuous probability distribution of your choice. We will call the revealed number - by Andrea or Aris - x. Then you decide who has chosen the lowest page number with this criteria:
If $x > r$ you decide that the person that did not tell you his number has the lower page, otherwise the other one.
In this way there are three cases (where x and y are the two page numbers chosen by Andrea and Aris):

1. $r < x \wedge r < y$
   In this case, you guess "smaller" and win the game if $x > y$ because variables x and y were assigned to the hidden numbers uniformly at random, $P(x > y) = 1/2$. Thus, in this case you win with probability $\frac{1}{2}$

2. $r > x \wedge r > y$
   By a symmetric argument to the first case, you guess "larger" and win with probability $\frac{1}{2}$

3. $x < r < y$ or $y < r < x$
   In this case (represented by two combinations) you win with probability 1 (always).

Case 3 occurs with a finite non-zero probability ϵ. Averaging over all the cases, your chance of winning is $\frac{(1+\epsilon)}{2}$, which is strictly greater than half.

4 Exercise 4

4.1 Probability Space

Given the model Erdős-Rényi $G_{n,p}$, where n is the number of nodes, an edge is present with probability $p \in [0, 1]$, and $N = \binom{n}{2}$ the number of all possible edges, the probability space can be defined as:

$$\Omega = 2^N = 2^{\binom{n}{2}} \tag{9}$$

4.2 Probability of each element

From paragraph 4.1 for every $G \in \Omega$, we have:

$$P(G) = p^m (1-p)^{N-m} \tag{10}$$

where m is the number of edges in G.

4.3 Probability of one triangle

The probability that there is exactly one triangle in a graph from G_{np} (with no other edge), from 4.2, is:

$$P(Triangle) = \binom{n}{3} p^3 \tag{11}$$

where $\binom{n}{3}$ are the possible ways we can have a triangle from the edges.

4.4 Probability of one line

Line graph is composed by n-1 edges and 2 nodes with degree 1 (head and tail) and n-2 nodes with degree 2. and from 4.3 we can have:

$$P(Line) = (2n-4)p(p-1)^{2n-4} \tag{12}$$

Where:

- $2n-4 = 2(n-2)$ are the possible nodes to be present for every node
- $(2n-4)p(p-1)^{2n-4}$ is the probability of having a line graph of degree 1

This follows from 4.2 this expression represent the chance to have a line graph.

4.5 Expected number of edges

The expected number of edges is:

$$\mathbb{E}[Edges] = \sum_{i=1}^{\binom{n}{2}} X_i = p\binom{n}{2} \tag{13}$$

Since we can say that the number of edges follows the binomial distribution, where X_i is a vector representing a graph (Bernoulli distribution).

4.6 Expected number of 3-star

We can observe that when a 3-star is present we have one node of degree (number of edges) 3. We can also notice that the the degree of a single node is defined by a binomial distribution as well. We have:

$$P(degree = 3) = \binom{n-1}{3} p^3 (1-p)^{n-1-3} \tag{14}$$

Hence:

$$\mathbb{E}[degree = 3] = nP(degree = 3) \tag{15}$$

4.7 Expected number of k-star

From 4.6 we have

$$P(degree = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k} \tag{16}$$

and:

$$\mathbb{E}[degree = k] = nP(degree = k) \tag{17}$$

5 Exercise 5

I created a script that detects if the file *beers.txt* is present it the current folder, if not present the script will proceed with the download. Then the file will be processed with a single pipeline of commands:

1. *cut -f 1* for selecting only the first column of the document
2. *sort* for getting all equal names close to each other
3. *uniq -c* for counting equal consecutive lines
4. *sort -g -k1 -r* for sorting from the numbers of occurrences
5. *head -n 10* for displaying only the top 10

For more details see *sorter.sh*.

6 Exercise 6

Using python it is possible to perform the same kind of operations described in previous sections in an easier way: first I count the times an item appears in the document while in the mean time I update the counter relative to that item of the total score until that moment. When the scanning of the document ends, i filter out all item that appeared less than 100 times and i compute the average score for the remaining items.
In the program it is also included a check for the bash script that outputs the 10 most frequent items (before the filtering).
For more details see *sorter.py*

7 Exercise 7

Using *Request* and *Beautiful Soup* libraries I build a simple web crawler that starts from the base link of the web-design job offerings of Kijiji and while saving all jobs data looks for hyperlinks that has not visited yet. A new link is explored by calling recursion on the function.

An attempt of filtering out ads content was made: whether an announce with special yellow banner (sponsored content) is encountered, the item is added to the announcement list with full description (otherwise only the description displayed on the page will be considered, i.e. first 150 characters of the full description).
At the end all results are written in a tab separated file, as requested.
For more details see *crawler.py*