CS 591: Data Analytics - Theory and Applications – Spring 2017

Project

Gautam Bhat, U02246123, gautam@bu.edu

1 Probability, Information Theory

1.1 Isaac Newton Helps Samuel Pepys

Intuitively, I believe it is more likely to get 1 six in 6 rolls than the other two cases.

Verifying the intuition:

Probability of getting no sixes in 6 rolls $= (\frac{5}{6})^6 = 0.3349$

Probability of getting at least one six in 6 rolls $= 1 - (\frac{5}{6})^6 = \mathbf{0.6651}$

Probability of getting no sixes in 12 rolls $= (\frac{5}{6})^{12}$

Probability of getting exactly one six in 12 rolls $= \binom{12}{1}\frac{1}{6}(\frac{5}{6})^{11}$

Probability of getting at least 2 sixes in 12 rolls $= 1 - (\frac{5}{6})^{12} - \binom{12}{1}\frac{1}{6}(\frac{5}{6})^{11} = \mathbf{0.619}$

Probability of getting no sixes in 18 rolls $= (\frac{5}{6})^{18}$

Probability of getting exactly one six in 18 rolls $= \binom{18}{1}\frac{1}{6}(\frac{5}{6})^{17}$

Probability of getting exactly two sixes in 18 rolls $= \binom{18}{2}(\frac{1}{6})^2(\frac{5}{6})^{16}$

Probability of getting at least 3 sixes in 18 rolls $= 1 - (\frac{5}{6})^{18} - \binom{18}{1}\frac{1}{6}(\frac{5}{6})^{17} - \binom{18}{2}(\frac{1}{6})^2(\frac{5}{6})^{16} = \mathbf{0.597}$

Generalising the above to : **Getting at least n sixes on 6n rolls :**

$$Pr[at\ least\ n\ sixes\ on\ 6n\ rolls] = 1 - \sum_{i=0}^{n-1} \binom{6n}{i} (\frac{1}{6})^i (\frac{5}{6})^{6n-i}$$

1.2 Sensitive Questions

For the **1st** case, the possible events are: BB, GG, BG, GB

Therefore, knowing at least one is a boy, the possible events are: BB, GB, BG

Out of this, the possible events of the second child being a girl are: GB, BG

$$\Rightarrow Pr[\text{One Girl} \mid \text{One Boy}] \;=\; \mathbf{\frac{2}{3}}$$

For the **2nd** case, we see one of the kids, and it is a boy. However, we do not know whether it is the first child or the second child. Therefore, the possible events are: $B_1B_2, B_2B_1, G_1G_2, G_2G_1, B_1G_2, B_2G_1$

Therefore, in this case, using similar logic on the set of events,

$$Pr[\text{One Girl} \mid \text{One Boy}] \;=\; \mathbf{\frac{1}{2}}$$

1.3 Bayes' Rule

D1 = Event that prize is behind Door 1
D2 = Event that prize is behind Door 2
D3 = Event that prize is behind Door 3

H1 = Event that host opens Door 1
H2 = Event that host opens Door 2
H3 = Event that host opens Door 3

$Pr[D1] = Pr[D2] = Pr[D3] = 1/3$

The contestant chose **Door 1** and the host opened **Door 3**.

$Pr[H1] = 0, \;\; Pr[H2] = \frac{1}{2}, \;\; Pr[H3] = \frac{1}{2}$

$Pr[H3|D1] = \frac{1}{2}, \;\; Pr[H3|D2] = 1, \;\; Pr[H3|D3] = 0$

Now, we are left with the possibility of the prize being behind Door 1 or Door 2, since the host opened Door 3.

$$Pr[D1|H3] = \frac{Pr[H3|D1] * Pr[D1]}{Pr[H3]} \;=\; \frac{(1/2) * (1/3)}{(1/2)} \;=\; 1/3$$

$$Pr[D2|H3] = \frac{Pr[H3|D2] * Pr[D2]}{Pr[H3]} = \frac{1 * (1/3)}{(1/2)} \;=\; 2/3$$

Therefore, **(b)** The contestant should switch to door 2.

1.4 Boston Globe

$C \rightarrow$ Event of developing premenopausal breast cancer

$A \rightarrow$ Event that teenage years were physically active

$S \rightarrow$ Event that teenage years were sedentary

Note that $A = 1 - S$

$Pr[C|A] = (1 - 0.23)Pr[C|S] = 0.77Pr[C|S]$

The problem with this deduction is that, correlation does not necessary imply causation.

There is a possibility that, that the two events A (or S) and C are conditionally independent. Since the set of study participants just 'happened' to exercise or not exercise during their teenage years, it doesn't rule out the existence of a third random variable that A, S, and C are separately connected to.

If the same ratio of conditional probabilities ($\frac{P[C|A]}{P[C|S]} \approx 0.77$) can be achieved by taking into account the random variable (if any) to which the outcomes of A(or S) and C are actually connected, then this causation can be more plausible.

Otherwise, more data needs to be collected on a large, random set of women who are explicitly asked to exercise in their teenage years, and a random set of women who are explicitly asked to more sedentary. That way, the conditional dependence of A(or S) and C can be overlooked in this study.

2 Hashing

2.1 2-wise independent hash function

$h(x) = ax + b \ mod \ p$

Let us take two elements, x_1 and x_2.

$h(x_1) = ax_1 + b \ mod \ p \quad = v_1$
$h(x_2) = ax_2 + b \ mod \ p \quad = v_1$

Since p is prime, the probability of $h(x_1)$ hashing to v_1 is $\frac{1}{p}$

Similarly, the probability of $h(x_2)$ hashing to v_2 is $\frac{1}{p}$

Also, in terms of the values x_1, x_2, v_1, v_2,

$a = \dfrac{(v_1 v_2)}{(x_1 x_2)}\ mod\ p$

$b = \dfrac{(v_2 x_1 - v_1 x_2)}{(x_1 - x_1)}\ mod\ p$

Since a and b are selected uniformly at random from $[p]$, that means the probability of the both of them taking on these values,

$Pr[a = \dfrac{(v_1 v_2)}{(x_1 x_2)}\ mod\ p, b = \dfrac{(v_2 x_1 - v_1 x_2)}{(x_1 - x_2)}\ mod\ p]\ =\ (\dfrac{1}{p})^2$

Note that this is the same as:

$Pr[h(x_1) = v_1,\ h(x_2) = v_2]\ =\ (\frac{1}{p})^2$

Also,

$Pr[h(x_1) = v_1] = \frac{1}{p}$

$Pr[h(x_2) = v_2] = \frac{1}{p}$

$\Rightarrow Pr[h(x_1) = v_1]Pr[h(x_2) = v_2]\ =\ (\frac{1}{p})^2\ =\ Pr[h(x_1) = v_1,\ h(x_2) = v_2]$

Hence, the hash function is pairwise independent.

2.2 Searching for patterns in fMRI

1. A naive approach would involve checking each $r \times r$ sub-slice of the slice and compare each element in that sub-slice with the corresponding element in the pattern. If all match, a positive match is returned.

   The pre-processing involved in this algorithm is only the steps needed to convert the data into a proper 2d array, and is usually language specific.

   While checking each sub-slice, the matching time would be in the order of r^2.

   The time-complexity of this is about $O((n - r + 1)^2 r^2)$.

   The space complexity of this, apart from the size of the slice (n^2) and pattern(r^2), is $O(1)$.

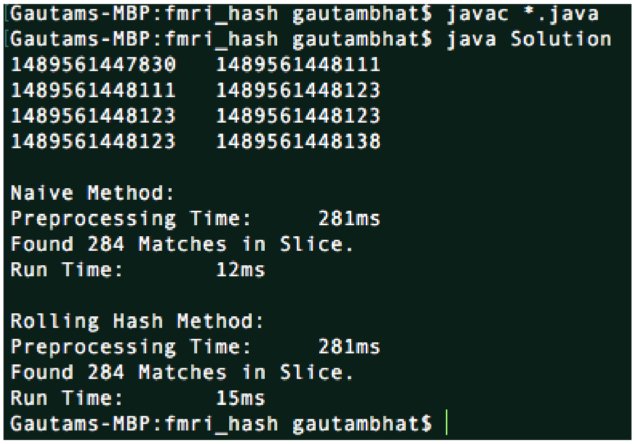
2. A more efficient method (for large data) would be to use a rolling hash-based scheme, similar to Rabin-Karp algorithm. We would compute the hash of the first sub-slice, and calculate each subsequent hash by modifying the current hash value based on the clever trick described in the Rabin-Karp algorithm. These hash values for each sub-slice would be compared with the hash of the pattern, to look for a match.

   The pre-processing, apart from the same language-specific programming, would involve calculating the starting hash for the slice, and hash of the pattern. This would be in the order of $O(r^2)$.

After computing the hash, the matching time would be $O(1)$. The time complexity of this algorithm would be about $O(n - r + 1^2 r)$, assuming recalculating the hash value takes about $O(r)$ time.

The space complexity, apart from the pattern and slice, is $O(1)$.

3. The implementation for the above algorithms is provided in **U02246123_fmri_hash/Solution.java**. Below is the screenshot of the run times and matches.

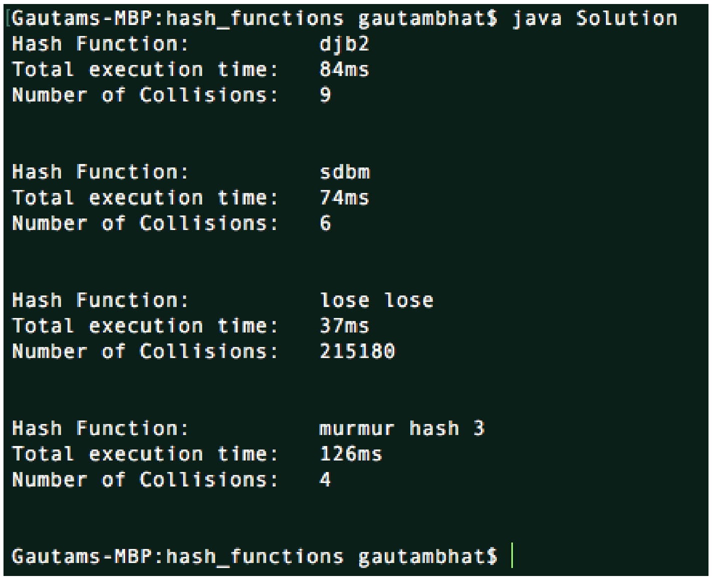


```
Gautams-MBP:fmri_hash gautambhat$ javac *.java
Gautams-MBP:fmri_hash gautambhat$ java Solution
1489561447830   1489561448111
1489561448111   1489561448123
1489561448123   1489561448123
1489561448123   1489561448138

Naive Method:
Preprocessing Time:     281ms
Found 284 Matches in Slice.
Run Time:       12ms

Rolling Hash Method:
Preprocessing Time:     281ms
Found 284 Matches in Slice.
Run Time:       15ms
Gautams-MBP:fmri_hash gautambhat$ 
```

The run time of the naive method is better possibly because the size of r is too small to really make the rolling hash method more efficient.

2.3 Hashing strings

The implementation for the hashing algorithms can be found in **U02246123_hash_functions/Solution.java**. The number of collisions, and run times, are reported in the screenshot below.



```
Gautams-MBP:hash_functions gautambhat$ java Solution
Hash Function:          djb2
Total execution time:   84ms
Number of Collisions:   9


Hash Function:          sdbm
Total execution time:   74ms
Number of Collisions:   6


Hash Function:          lose lose
Total execution time:   37ms
Number of Collisions:   215180


Hash Function:          murmur hash 3
Total execution time:   126ms
Number of Collisions:   4


Gautams-MBP:hash_functions gautambhat$ 
```

7 Individual Project

7.1 Milestone

For the individual project, the area I want to focus on is **Machine Learning**.

As discussed with the instructor, I am interested in implementing reinforcement learning using neural networks, in a variety of applications. As a good starting point, I want to start with RL on combinatorial games, of which Tic-Tac-Toe is a great example.

The problem I would be focusing on would be train an AI to play a betting-based version of Tic-Tac-Toe, in which both players start out with equal amounts of money, and have to play a higher bet in order to make the next move on the board.

The learning would be achieved by the AI playing games against itself.

Before moving on to the betting aspect, I want to focus make the traditional version of the game work for, after which I can add a layer of complexity with the betting aspect.

I have not committed to using a particular language or framework, but unless something changes, I would be choosing TensorFlow and Python to implement the project. If animation for the game board is required, it would include some Javascript as well.

I am/will be refering to the following links to help me move forward in the project.

- Implementation of Reinforcement Learning Algorithms. Python, OpenAI Gym, Tensorflow. Exercises and Solutions to accompany Sutton's Book and David Silver's course.
- A toolkit for developing and comparing reinforcement learning algorithms.
- Reinforcement Learning: An Introduction (Course Materials)
- Reinforcement Learning: An Introduction (Book)
- Playing Atari with Deep Reinforcement Learning
- Combinatorial Games: Tic-Tac-Toe Theory

Any further guidance and feedback would be appreciated.