Lab 3 - A Survey of Elementary Distributions through the lens of Bitcoin

Due: Saturday, April 8 2017, 11:59 pm

1 What to submit (MANDATORY)

1. PDF file ((lab3_fisrtname_lastname.pdf))including plots and a snapshot of the code used to answer the questions.
    - Names of the collaborators.
    - Number of late days for this assignment.
    - Number of late days so far.
    - References used
2. Python script (lab3_Bitcoin_mine.py) with the code used.

Failing to meet any of the above requirements will cause a decrease of your grade.

2 Bitcoin

In this Lab, we are going to study some basic properties of elementary distributions. As a motivating example, we are going to examine a simplified version of the crypto currency Bitcoin.

2.1 What is Bitcoin?

Bitcoin a digital currency initially introduced by "Satoshi Nakamoto" in early 2008. While Bitcoin is incredibly fascinating in regards to philosophy and implementation, this lab will focus on how Bitcoin is *mined.*

Bitcoin is similar to objects with "intrinsic" value which are obtained through physical labor such as gold. While there are many reasons why gold has become socially accepted as an object instilled with intrinsic value, it can be partially attributed to its scarcity and traditional difficulty of acquisition. For starters, if we were to acquire all of the gold in the world, and melt it down, it would fit in about 3.5 Olympic regulation sized pools. Second, the process of mining gold is costly and physically difficult. You need to smack a rock with a pickax until you strike gold. Further, even if you mine for gold your whole life, there is no guarantee that you will ever actually mine any!

In order to simulate this level of exertion, Bitcoin bases its algorithm for mining off of a *proof of work* scheme, where instead of physically exerting oneself as in the case of gold mining, Bitcoin miners compute something that requires a considerable investment in the way of hardware, power, and time. The actual proof of work scheme involves finding an input to the SHA-256 hash function that has n leading zeros. (So, the miner needs to find some x such that $y = \text{SHA256}(x)$ where the first n bits of y are zero.) As a remark, there are restrictions on the input.

Bitcoin makes use of a data structure called the Blockchain, which acts as a public ledger of all transactions. You can imagine it as a linked list of blocks. Each block contains a set of transactions. In order to mine a new Bitcoin, a miner needs to first propose a new block B_j (which includes a set of transactions to be confirmed on the blockchain; we omit details but you can learn more about this online!). Then, the miner needs to *confirm* this block B_j by finding a nonce r_j such that $y = \text{SHA256}(B_{j-1}||B_j||r_j)$ and the first n bits of y are zero. B_{j-1} is the previous block in the blockchain. The nonce (a value that never repeats, usually a value that is chosen at random) r_j is used to confirm that the block B_j comes immediately after block B_{j-1} on the blockchain. Miners compete to find a valid nonce r_j; the first miner to find and publish such a nonce r_j to the other miners is rewarded with bitcoin(s).

3 README

For questions where you are asked to plot a histogram, unless otherwise specified, please submit 3 graphs corresponding to $\rho = 1, 3, 6$. Failure to include graphs will result in a lower grade.

4 Bernoulli Distribution

We begin the discussion of distributions with one of the most elementary and important: the Bernoulli distribution. Many problems in probability theory involve independently repeating random experiments and observing whether or not a specific event occurs. That is, Bernoulli random variables arise naturally in probability theory as indicators of events. As a motivating example, suppose out of all possible outcomes Ω, we define $A \subset \Omega$ to be a success and $\Omega \setminus A$ to be a failure . Such an experiment is modeled by a random variable X with Bernoulli distribution written $X \sim \mathit{Bernouli}(p)$ with outcomes

$$X = \begin{cases} 1 & \text{if } A \text{ occurs,} \\ 0 & A \text{ does not occur.} \end{cases}$$

$$\begin{aligned} &\Pr[X = 1] = p \\ &\Pr[X = 0] = 1 - p \\ &\Pr[X = k] = 0 \qquad \forall k \notin \{0, 1\} \end{aligned}$$

Repeated trials of a random experiment are called Bernoulli trials if the following conditions are met.

- The trials are independent of each other.
- The result of each trial is either a success or a failure
- The probability of success remains unchanged from trial to trial

Recall that independence of random variables X, Y is defined as

$$\Pr[X|Y) = \Pr[X]$$

and similarly,

$$\Pr[Y|X] = \Pr[Y]$$

Intuitively, this means that observing the outcome of one event in no way influences the outcome of future outcomes. For example, if I flip a fair coin 100 times and get all heads, I am still equally as likely to get heads or tails on the 101st flip. Each coin flip can be thought of as an independent Bernoulli trial with probability of success $p = 1/2$. More generally, the most elementary example of a Bernoulli random variable, is flipping a p-biased coin, and having the random variable take on 1 (success) if the coin comes up heads and 0 (failure) otherwise.

It is important to keep in mind why it is even worth studying distributions, aside from the fact that they are interesting in their own right. In particular, we, as computer scientists often wish model the outcome of uncertain events. Lets look at a single attempt at mining a Bitcoin. To model this process, we make a few assumptions.

First, lets assume that outputs of the hash function are indistinguishible from what is called a *Random Oracle*. That is, on input $x \in \{0, 1\}^*$, the output value SHA256(x) is uniformly chosen from $\{0, 1\}^{256}$, which is the same as if you flipped 256 fair independent coins and recorded the output. Further, if you hash the same x, you get the same value both times. Hence, it behaves like an oracle that on a request randomly chooses an output, and then remembers it for future requests.

Second, assume that each miner randomly chooses a value for the nonce r_j, hashes it along with B_j and B_{j_1}, and checks if the output has ρ leading zeros. If it doesn't, it chooses a fresh random r_j and tries again. As a remark, it is not this simple in reality.

Third, lets assume that the required number of leading zeros is denoted $\rho \in \mathbb{N}$. Note that in real life, this parameter is chosen by the Bitcoin community based on the number of active miners and the total computing power available to the miners in order to curb the introduction of new blocks.

Under these assumptions, we can model a single attempt as a Bernoulli experiment with parameter p where p is the probability of successfully mining a Bitcoin.

As mentioned earlier, the parameter ρ which is the number of leading zeroes is NOT always the same. For a given $\rho \in \{0, 1, 2, ..., 256\}$, we want to be able to write down the probability of finding a nonce r that verifies for a block B_{j-1}. I will do the first 1 for you– if $\rho = 0$, we assume this means we don't care how many leading zeroes there are, and the probability of success is 1.

1. Recall that we assume SHA256 is indistinguishable from a random oracle. Under this assumption, let p be the probability that a randomly chosen nonce r_j confirms that some block B_j comes after block B_{j-1}. Determine p in terms of ρ. Hint– recall that a when a random oracle (on $\{0,1\}^{256}$ is queried on a new input, its response is uniformly chosen from $\{0,1\}^{256}$ and becomes fixed for that input.
2. Let X be a random variable that for a single trial takes on 1 if a coin is mined and 0 otherwise. Explain why a X is a Bernoulli random variable. Write a python function function `Bernoulli_mine` simulates X in terms of ρ.
3. Lets prove some basic properties of a Bernoulli random variable. Let X be a Bernoulli random variable with probability of success p. Prove that the distribution on X is a valid probability distribution. For this lab, when I say prove that a certain discrete RV has a valid distribution, you need only show

$$\sum_{x \in Range(X)} Pr[X = x] = 1$$

   . The remaining properties are either tedious or painfully obvious.
4. Now, lets try actually mining Bitcoin. To accomplish this, we will be using SHA256 from python's hashlib library and the code below to create generate nonces. Write a function called `hash_mine` that takes as input the old block, the proposed block and a nonce. The function should either return [False,''] which indicates the mine was unsuccessful, or [True, the_nonce] indicating the a successful mine.

```
import hashlib, binascii
from numpy.random import randint
def hash_mine(b_old, b_new, rho):
        pass
 #generate a random nonce
 length=8 #defult
    nonce= ''.join([str(randint(0, 9)) for i in range(length)])
```

5. For this problem, lets assume that block B_{j-1} has value `wubba lubba` (note the space) and block B_j has value `dub dub`. Let X be a random variable that takes on value 1 if we successfully found a nonce, and 0 otherwise Write a function called `bernoulli_hist`(n,ρ) that does the following:
   - Simulates the PMF of X where the number of leading zeroes needed for a success is ρ.
   - Plot a histogram of frequencies of success and failures. Scale the y-axis so that we see frequencies like we did with the function dieRoll() from Lab 1.
   - Under the assumption that each mine can be modeled by a Bernoulli random variable, plot the theoretical PMF on top of the histogram. Note that there are only two points to plot, so a good way to ensure both graphs are visible is to use a scatter plot for the theoretical PMF and use a different color.

To accomplish this, generate a random nonce and append it to $B_{j-1}||B_j$, hash it using SHA256 and count the number of leading zeroes. If the number of leading zeroes is greater than or equal to ρ, this is a success (1) and a failure otherwise (0). In order to get a good estimate for the PMF, repeat this process 100,000 times and plot the resulting histogram. **Submit your code.**

5 Geometric

As I am sure many of you will recognize at this point, our current model is very limited. It is only capable of modeling 1 attempt at mining. In reality, we will likely have to try billions of nonces before we are ever successful. A natural way to extend this model is to allow for a sequence of mining attempts. In other words, we are moving away from a singular Bernoulli trial to *some* number of Bernoulli trials. How many you ask? In this section, lets assume we will continue to mine until we successfully find a verifying nonce. We make the following assumptions:

- ρ stays constant during this series
- As a result, the probability of success remains constant
- Each trial is independent.

A good example to keep in mind is the experiment from Lab 1 of repeatedly flipping a p biased coin until we observe a heads. This example gives us an (almost) complete description of the Geometric distribution.

Let $\{X_i\}$ be collection of independent trials of a Bernoulli experiment where i is the trial number and p is the probability of success(written $X_i \sim Bernoulli(p)$). Let

$$T = \min\{i : X_t = 1\}$$

In English, T is the index where we first observe a success. From lab1, we have

$$p_X(k) = \Pr[X_T = k] = p(1-p)^{k-1} \qquad \forall k \in \mathbb{N}$$

X has a geometric distribution, written $X \sim G(p)$.
As a remark, there is an equivalent definition that is defined in terms of the number of failures in the string. Notice that this definition has range $\{0, 1, 2, 3...\} = \{0\} \cup \mathbb{N}$. We will use the convention that the first Bernoulli trial occurs at $t = 1$ and X takes on the index of the first success.

Now lets prove some basic results about geometric random variables.

6. Prove that $X \sim G(p)$ is a probability distribution. (Hint: use the Geometric Sum formula)
7. Prove that the expected value of the geometric random variable is the expected value $E[X] = \frac{1}{p}$. (Hint, try taking the derivative of $\sum_{i=1}^{\infty}(1-p)^i$. If you are skeptical about differentiating an infinite sum, Google Uniform Convergence. Alternatively, you could use the law of total expectation)
8. Prove that for any $s, t \in \mathbb{N}$ and $X_T \sim G(p)$,

$$\Pr[X_T \geq s + t | X_T \geq t] = \Pr[X_T \geq s]$$

   Or in English, if you observed t failures in a string of Bernoulli Trials, the probability of (for the same Geometric random variable) observing a success on the $s + t$th index for some s is the same as the probability of observing a success on the sth time. This is commonly called the *Lack of Memory Property.*
9. For this problem, lets assume that block B_{j-1} has value `'wubba lubba '` (note the space) and block B_j has value `dub dub`. Complete the python function `geometrix_hash_mine` that takes as input an integer ρ, the block $B = B_{j-1}||B_j$ and finds a nonce r such that SHA256($B_{j-1}||B_j||r$) has at least ρ leading zeros. This function should return [num_trials, r] where num_trials is the number of trials it took to successfully find a verifying nonce r.

10. Let X be a random variable that denotes the first index at which a verifying nonce was found. In this question you will be estimating the PMF of X. Write a function called `geometric_hist`(b_old, b_new, ρ) that takes as input the number of leading zeroes ρ and $B = B_{j-1}||B_j$ (old block and new block)and plots the histogram of 100,000 calls to `geometric_hash_mine`. Note you only need the first element of the output for each call (IE don't plot the nonce, only the number of tries it took successfully find a verifying nonce). On top of the histogram, plot the actual PMF of X based on ρ. Explain why the Geometric distribution is a good model for this experiment.

6 Binomial

Now suppose we conduct a finite sequence of independent Bernoulli trials. Often, we are interested in counting the number of successes. As a motivating example, suppose you flip 3 fair coins, and are interested in the probability that exactly one coin comes up heads. We first write down the space of possible outcomes where T denotes tails and H denotes heads.

$$\Omega = \{(HHH), (HHT), (HTH), (THH), (TTH), (THT), (HTT), (TTT)\}$$

We next identify the set A of all trials with exactly one heads

$$A \subset \Omega = \{(TTH), (THT)(HTT)\}$$

In the case where the coin is fair, the probability that exactly one of the trials is heads is

$$\frac{|A|}{|\Omega|} = \frac{3}{8}$$

A random variable X that counts the number of successes in a sequence of independent Bernoulli trials is called a Binomial random variable written $X \sim B(n,p)$ where n is the number of trials and p is the probability of success.

In the case of the above example, the number of trials is 3, as we flip 3 coins and the probability of success is $p = \frac{1}{2}$ since the coin is fair.

Generalizing this for $X \sim B(n,p)$, we are interested in $\Pr[X = k]$ which is the probability that in a string of n Bernoulli trials, we get exactly k successes. If we observed k success, then that means there were $n-k$ failures as there were n trials total. Now we note that there are exactly $\binom{n}{k}$ ways of having k success and $n-k$ failures. Finally, since each trial is independent, we have

$$\Pr[B(n,p) = k] = \binom{n}{k} p^k (1-p)^{n-k} \qquad \forall k \in \{0, 1, ..., n\}$$

In English, suppose we flip a p biased coin n times. Then probability that exactly k of these coins land heads up is exactly the PMF of $X \sim B(n,p)$.

11. Write a function called binomial_mine(ρ, n) that simulates n attempts at finding a verifying nonce r according to a binomial model where the probability of mining a coin on any attempt is depends on ρ (again, the number of leading zeroes). This function should output the number of successes. Use bernoulli_mine(ρ) as subroutine in your function. **Submit your code .**

12. Write a function `binom_hash_mine(b_old,b_new,` ρ, n) which takes parameters $B = B_{j-1}, B_j$, n, ρ, and returns the number of coins successfully mined in n calls to `hash_mine` **Submit your code.**

13. Write a function called binom_hist(b_old,b_new, ρ, n)) which plots the result of 1,000 calls to binom_hash_mine. Scale the y-axis so we see probabilities like with the dieRoll() graph. Plot the theoretical PMF of the Binomial distribution with parameters n and p where p depends on ρ. Explain why the Binomial distribution is a good model for this experiment. **Submit your code and the plots you create by calling**

Notice that if we try to increase the number of data points from 10,000 to 100,000 this becomes incredibly slow. Further, we encounter overflow errors if we pick $n > 100$. (Alternatively, you could use `from scipy.stats import binom` which computes the PMF in such a way that it doesn't compute everything at once).

```
binomial_hist('wubba lubba ', 'dub dub', 1,100)
binomial_hist('wubba lubba ', 'dub dub', 2,100)
binomial_hist('wubba lubba ', 'dub dub', 3,100)
```

14. Prove that for $X \sim B(n,p)$, then $E[X] = np$. (Hint, we are counting the number of success of a sequence of n Bernoulli trials. Use the linearity of expectation.)

Recall that we are modeling a block being confirmed as a Bernoulli trial where a success is a new block being confirmed (aka as a bitcoin being mined). Further, since there are roughly $n = 3600 \times 2 \times 10^6 \times 10^{12}$ hashes per hour, we have n Bernoulli trials per hour each with parameter p (that you determined above). Let X be the random variable counting the number of new blocks confirmed in one hour. Then, X has a binomial distribution $B(n,p)$.

Notice that n is huge and p is tiny for large ρ.

7 Exponential Distribution

Lets revisit the experiment of flipping a coin until we get heads. As you know, the probability of getting a heads on the kth flip for a p biased coin is exactly $(1-p)^{k-1}p$. Now suppose for $n \in \mathbb{N}$, we flip the coin at times $\frac{1}{n}, \frac{2}{n}, \frac{3}{n} \ldots$. That is, instead of flipping a coin every minute, the time is quantized in units of duration $\frac{1}{n}$. Let X be the time of the first success. For $x \in \mathbb{R}, x > 0$. we set $k = \lfloor nx \rfloor$. Since $\frac{k}{n} < x < \frac{k+1}{n}$, it follows that $\{X > x\} = \{X > \frac{k}{n}\}$. Therefore,

$$\Pr[X > x] = \Pr[X > \frac{k}{n}] = (1-p)^k.$$

This is exactly the geometric distribution. From elementary calculus, we have $e^x \approx 1 + x$ for small x which in this case implies $x \approx \frac{k}{n} \implies$

$$\Pr[X > x] = (1-p)^k \approx e^{-pk} = e^{-np\frac{k}{n}} \approx e^{-npx}.$$

Therefore, it is natural to ask about random variable who's tail probability is

$$1 - e^{-\lambda x}.$$

In fact, this random variable would be the continuous analogue of the geometric distribution.

Since the PDF is the derivative of the CDF, we end up with

$$f_X(x) = \frac{d}{dx}F_X(x) = \frac{d}{dx}(1 - e^{-\lambda x}) = \lambda e^{-\lambda x}$$

A continuous random variable X is an *Exponential* random variable if it has a pdf of the form

$$f_X(x) = \lambda e^{-\lambda x}.$$

with $\lambda > 0, x > 0$. Intuitively, exponential random variables are used to model independent interarrival times. Lets consider an example. Suppose people arrive at a hospital according to an exponential distribution with parameter $\lambda = 2$ for time measured in hours. Let X be the arrival time of a patient. If you were asked what is the probability of somebody arriving between $6:15$pm and $6:30$pm, we can compute it as follows. (To be more precise, we assume the previous arrival was at 6 : 00pm) First, we notice that the length of the interval is all that matters. Therefore, we are interested in

$$\Pr[1/4 < X < 1/2] = \int_{1/4}^{1/2} f_X(x)dx = \int_{1/4}^{1/2} 2e^{-2x}dx = e^{-2x}\Big|_{1/4}^{1/2}$$

15. Verify that an exponential distribution is indeed a probability distribution.
16. Let $X \sim Exp(\lambda)$. Compute $E(X)$ (Hint: a direct method would be to use integration by parts)

8 Poisson distribution.

It turns out that when a random variable X has a binomial distribution $B(n,p)$ where $n \to \infty$ and $p \to 0$ while $np = \lambda$ is constant, then we can approximate the distribution of that Binomial random variable with a **Poisson Distribution**.

If Y is a discrete Poisson random variable, then Y has range $Range(Y) = \{0, 1, 2, ..., \infty\}$. (This makes sense, since it approximates the Binomial distribution, which has range $\{0, 1, 2, ..., n\}$ when $n \to \infty$.) It's PDF is

$$\Pr[Y = k] = \frac{e^{-\lambda}\lambda^k}{k!}$$

As a motivating example, suppose we want to know approximately how many Bitcoins will arrive within a given time interval. That is, we want to understand the connection between independent arrival times and the number of arrivals within a given time interval. Let Y represent the number of new blocks mined in an hour, and X_n denote the amount of time measured in hours that passes until the nth block arrives.

17. Explain conceptually why $\{Y < n\}$ is the same event as $\{X_n > 1\}$ Since we do not know the exact distribution of Y, there is nothing to compute. Simply give an explanation for why this is at least plausible.

18. Lets return to the Binomial distribution. We can model a block being accepted as a Bernoulli trial where a success is a new block being accepted. Further, since there are roughly $2000000 * 1000000000000$ hashes a second, these are Bernoulli trials with $n >> 0$ and $p << 0$ (p is near 0, n is very large). Let X be the random variable counting the number of new blocks an hour, and let $\lambda = E(X)$. Prove that

$$\lim_{n\to\infty} \binom{n}{k} \left(\frac{\lambda}{n}\right)^k \left(1 - \frac{\lambda}{n}\right)^{n-k} = \frac{e^{-\lambda}\lambda^k}{k!}$$

    In particular, $X \sim Pois(\lambda)$ where the PMF is

$$f_X(k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

    (Hint, $\lim_{n\to\infty}(1 + \frac{x}{n})^n = e^x$)

19. Now we will show that the number of Bitcoins mined in a particular period of time is well approximated by a Poisson distribution.

    Get the file `real_ bitcoin.csv`. This file contains a list of 60 entries where the first column is the date and the second column is the number of Bitcions in circulation at the beginning of that day. Using python, open the csv file. For this question, you need to determine how "good of a fit" the Poisson distribution is to the process of mining Bitcoins.

    The data here shows the total number of Bitcoins in circulation each day. Notice that this number is monotonically increasing. This is because new bitcoins were mined each day. Recall that each time a block is confirmed, new bitcoins are created (mined). For this dataset, each mined block creates exactly 12.5 bitcoins.

Use this dataset to determine how many blocks were confirmed each day.

Then, plot a (scaled) histogram showing the number of blocks confirmed per day. Use 6 buckets.

This histogram should be well approximated by a Poisson distribution with some parameter λ. If you estimate λ by the average number of blocks confirmed per day, what is λ? Plot the pmf with parameter λ on top of the histogram. Be sure to scale the time interval. **Submit your code and plot.**

20. Finish the python program `poiss_hist`. Note that it should be identical to Binomial hist, with the only difference being the theoretical PMF plot. For what values of $rho \in \{1, 3, 6, 10\}$ does the Poisson adequately approximate the empirical results? **Justify your answer. Submit all plots.**
21. Does your header have all the requested information include your header?

9 END