EECS 70 Discrete Mathematics and Probability Theory
Fall 2014 Anant Sahai

Optional Virtual Lab 14

1. Optional Random Variables and Distributions Lab

(d) Given an experiment with two possible outcomes, S and F, with probability p for S and $1-p$ for F, the geometric random variable is the number of experiments run until the first outcome of S, i.e. the number of trials required to reach the first success. The number includes the last experiment with the S outcome. This type of R.V. was seen in the coupon collector's problem, where the successful event was collecting a coupon that you didn't have.

Its PMF and CMF are given by:

$$f(k;p) = p(1-p)^{k-1}$$
$$F(k;p) = 1-(1-p)^{k}$$

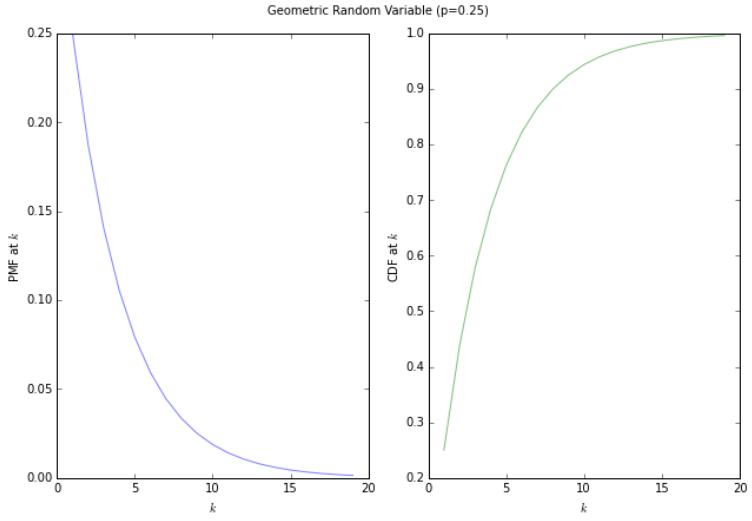
, respectively.

Related Values:

$$E[X] = \frac{1}{p}$$
$$Var(X) = \frac{1-p}{p^2}$$

Plot the PMF and CMF of a geometric random variable with parameter $p = 0.25$. What do you observe about the two curves?

Solution: The two curves look like the opposite of each other. The PMF decreases as k increases, whereas the CMF increases with k.



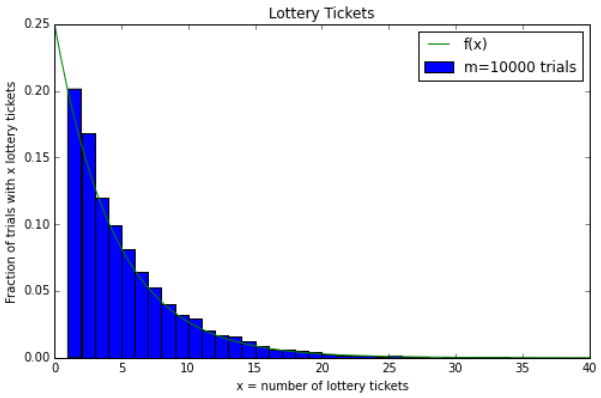
(e) Suppose that every day, Alice buys a lottery ticket, and will only stop buying lottery tickets when she wins. Let p be the probability that on any given day, Alice wins the lottery. Let X represent the total number of lottery tickets Alice buys.

Simulate $m = 10,000$ trials of Alice buying lottery tickets until she gets a winner. Use $p = 0.2$, and plot a histogram with the number of lottery tickets on the x-axis and the fraction of trials with each of these outcomes on the y-axis. Use a bin width of 1.

Next, overlay $f(x) = \mathbb{P}(X = x) = (1-p)^{x-1}p$, which is the probability of getting $(x-1)$ non-winning tickets, then a winning ticket. What is the average number of lottery tickets Alice has to buy?

Hint: Implement the functions `lottery_trial`, which computes the number of tickets Alice has to buy to win the lottery, and `lottery_pmf`, which computes the probability of buying x lottery tickets (i.e. $f(x)$).

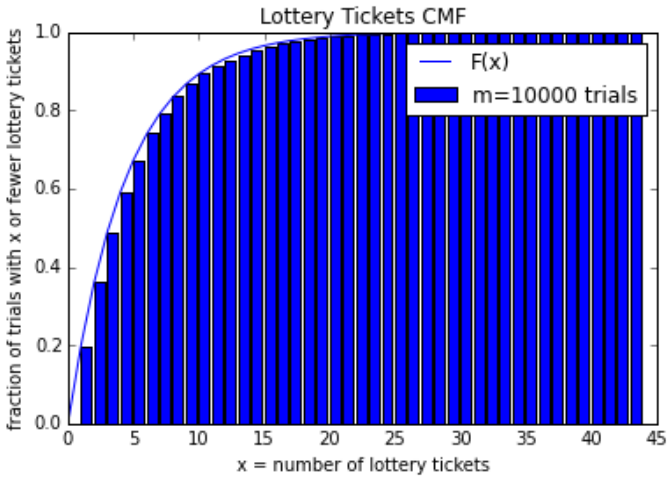
Solution: Based on the simulation result, on average Alice has to buy $4.8 - 5.2$ tickets. This number fluctuates around 5, which we know is the true expected value for a geometric random variable with parameter $p = 0.2$.



(f) The cumulative mass function (cmf) for X, i.e. the number of tickets Alice has to buy until she wins the lottery for the first time, is given as follows: $F(x) = \mathbb{P}(X \leq x)$. Use your histogram from the previous part to plot an "empirical CMF" as a bar chart (i.e. for each number of tickets x, plot the fraction of trials where Alice bought x or fewer tickets). Then overlay $F(x)$.

Hint: Implement the function `lottery_cmf`, which computes the probability of buying x or fewer lottery tickets.

Solution:



(g) In part (c) of VL 11, we went directly to one of the most powerful bounds we have, the Chernoff bound. Let's now look at a much simpler bound – the Chebyshev's inequality. For coin tosses, this inequality says

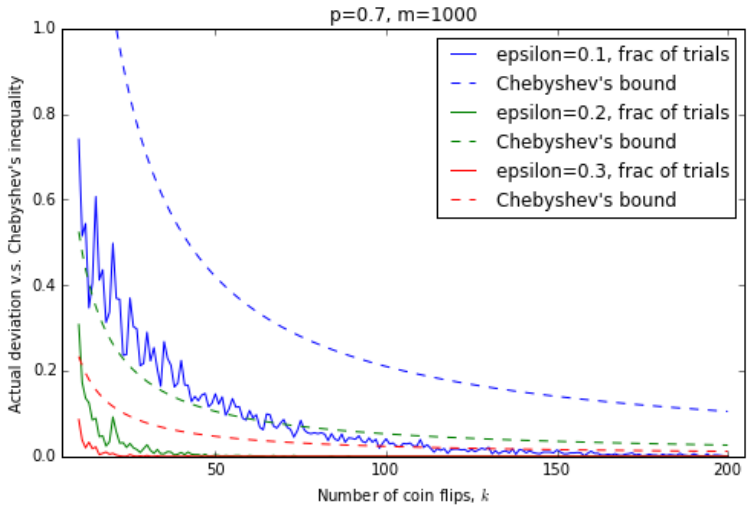
$$P(|S_k - kp| \geq \varepsilon k) \leq \frac{p(1-p)}{k\varepsilon^2}$$

, where S_k is the number of heads observed in k tosses of a p-biased coin.

Notice here that Chebyshev's inequality looks at two-sided deviations. We count both when S_k is much bigger than kp and when it is much smaller than kp. This is a big difference from the Chernoff's bound.

For $\varepsilon = 0.1, 0.2, 0.3$, plot and compare the actual frequency of such deviations to what Chebyshev's inequality estimates for $m = 1000$ trials of $k = [10, 200]$ tosses of a $p = 0.7$-biased coin. What do you observe?

Solution: We notice that the distance between the actual trial fraction frequency and the theoretical line is quite large, compared to the Chernoff's bound plot in VL 11. This is because Chebyshev's bound is a very loose inequality.



(h) Natural Language Processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. A solid understanding of probability is absolutely essential in doing research and solving NLP problems.

In this question, we will solve a simple NLP problem using conditional probability and the chain rule. According to Wikipedia, "a bigram is every sequence of two adjacent elements in a string of tokens, which are typically letters, syllables, or words; they are n-grams for $n = 2$". For example, we have the following document:

Peter Piper picked a peck of pickled pepper.
Where's the pickled pepper that Peter Piper pickled?

Given a word, e.g. "Peter", we want to find the most likely word that will follow it. For example, the conditional probability of "Piper" given "Peter" is:

$$P(Piper|Peter) = \frac{|Peter, Piper|}{|Peter|} = \frac{2}{2} = 1$$

(the $|S|$ notation shows the cardinality, or size of the set S. In this context, it is the frequency of the words (or pair of consecutive words) in the original document.)

However, the conditional probability of "Piper" given "a" is:

$$P(Piper|a) = \frac{|a, Piper|}{|a|} = \frac{0}{1} = 0$$

Using conditional probabilities thus captures the fact that the likelihood of "Piper" varies by preceding context: it is more likely after "Peter" than after "a". In a bigram model, the context is the immediately preceding word, which is often known as the Markov property:

$$P(w_1 w_2 \ldots w_i) = P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_i|w_{i-1})$$

Given the formula above, your task is to estimate the probability that a phrase appears in a document. We will give you a simplified corpus where every word has been lower-cased and punctuation are all removed, based on an 1865 novel written by Lewis Carroll, *Alice's Adventures in Wonderland*[1].

Solution: See *lab14sol.pdf*. There are two different solutions to this problem, and the first one (using `count` on strings) is technically wrong. For example, `"cathy's cat is cute".count("cat")` returns 2, where we really want 1 as the answer. However, since the original test cases given in the skeleton were made with this approach in mind, we will accept both solutions for full credit.

For the correct solution, you will need to use the `string.split()` method, then use `count` on a list for the exact count. You can also use regular expression, which is more elegant and works efficiently for large corpus since no intermediate lists (unlike `split`) are created.

(i) Question 9, part (j)

[1] `http://www.gutenberg.org/ebooks/11`