EE 303 Lab 2: The Bernoulli Experiment and the Distributions it Generates

1 Introduction

In this lab we illuminate four discrete distributions: the Bernoulli, the binomial, the geometric, and the Pascal. We use measurements in the Bernoulli experiment to generate these distributions. In the first section, we show how to randomly generate zeros and ones in a Bernoulli experiment. Next, we use the randomly generated zeros and ones to randomly generate realizations of binomial, geometric, and Pascal rvs. Then, histograms are used to compare experimental results with theoretical results.

2 Basic MATLAB Commands

Four basic MATLAB functions that you will need to know for this lab are `sum`, `plot`, `hold on`, and `for`.

1. The function `sum` can be applied to a vector, a matrix or a multidimensional array. For the simple case where the input is a vector, the result is a number that is a sum of the elements of that vector. For a matrix, we have to specify a second input in order to define how the summation should be calculated. To understand this, define a random matrix in MATLAB

```
>> A = [2 3 6;7 12 3;1 0 6]
A =

    2     3     6
    7    12     3
    1     0     6
```

If you want to find the sum of values of each column, either type `sum(A)` or `sum(A,1)`. This will show the following result in MATLAB command prompt

```
>> sum(A)

ans =

    10    15    15
```

The result is a row vector that contains the sum of the elements in each column of matrix `A`. Now type `sum(A,2)`

```
>> sum(A,2)

ans =

   11
   22
   7
```

The result is a column vector where each element is the sum of the elements in a row of matrix `A`.

2. The function `plot` plots curves. This function can be used in different ways. Here, we introduce a very simple method to use this function. Assume you want to plot a function $f(x)$ for different values of x. For example, you know that values of $f(x)$ for $x = [1, 2.3, 3, 4.5]$ are $[1, 5.29, 9, 20.25]$. To plot $f(x)$, gather the values of x in a vector and call it X. So type

```
>> X=[1 2.3 3 4.5]

X =

    1.0000    2.3000    3.0000    4.5000
```

Now put the values of $f(x)$ in a vector and call it Y. Type

```
>> Y=[1 5.29 9 20.25]

Y =

1.0000    5.2900    9.0000   20.2500
```

Note that if X is a row vector, Y should also be a row vector and if X is a column vector, so should be Y. Now use the function `plot` in the following way

```
>> plot(X,Y)
```

The result is a figure that MATLAB generates in a window. If you want to use the same figure to draw a plot for another function that for the same values of x takes values different than $f(x)$, you can use the function `hold on`. For example, if you have another function that takes values $[6, 7, 5.5, 4]$ for the same values of the vector X and you want to plot this function in the same figure, you can type

```
>> G=[6 7 5.5 4]

G =

6.0000    7.0000    5.5000    4.0000

>> hold on
>> plot(X,G)
```

You can see that the same figure has been used for the second plot, resulting in a plot of f and g vs. x.

3. The function `for` generates a decision loop. The structure of this function is exactly like the `if` function. Usually, this function is used in the following form:

```
>> for i=1:N
MATLAB commands
end
```

This code generates a loop with the counter i that starts from 1, increases by 1 at each step, and ends at N. For each i the MATLAB commands that are inside the `for` loop are executed. This procedure continues until i reaches N.

3 Bernoulli Experiments in MATLAB

A Bernoulli trial is an experiment where the outcome is one of two possible outcomes, namely "success" or "failure". For simplicity, we denote these two outcomes as one and zero, respectively. The probability of getting one is p, and the probability of getting zero is $(1-p)$. We denote the Bernoulli random variable as U and write its pmf as

$$p_U(u) = (1-p)(1-u) + pu \qquad u = 0, 1. \tag{1}$$

A sequence of coin flips is a sequence of Bernoulli trials, provided the flips are "independent," meaning the probability of the sequence $U_1, U_2, \ldots, U_n$ is a sequence of independent rvs U_i for trial i. Typically, $p = 0.5$. A "bent coin" on the other hand may have a better chance of landing on one side than another. In this case, $0 \leq p \leq 1$ is a value other than 0.5.

In MATLAB, the uniformly (pseudo) random number generator, namely `rand`, may be used to simulate a Bernoulli experiment. To simulate a "bent coin" with $p = 0.3$ being flipped 10 times, you could use the following code.

```
>> p = 0.3;
>> rand(1,10) < p

ans =

     0     0     0     0     1     0     1     1     0     0
```

The function `rand` returns values uniformly distributed between 0 and 1. By using the logical operator "less than", MATLAB assigns values less than 0.3 as a success (one) and values greater than or equal to 0.3 as a failure (zero).

The above example is one Bernoulli experiment with 10 trials. In general, we want to do M experiments with N trials in each experiment. For example to run $M = 5$ experiments consisting of $N = 7$ trials per experiment with $p = 0.4$, you would type the following in MATLAB

```
>> M = 5;
>> N = 7;
>> p = 0.4;
>> B = rand(M,N) < p


B =

     0     1     1     0     0     0     1
     0     0     1     0     1     0     0
     0     0     0     1     0     0     1
     1     0     0     0     1     0     0
     0     0     0     1     0     0     0
```

Here, each row is a different experiment (sequence of Bernoulli trials). Notice that none of the rows has exactly four ones. Is this surprising to you? Since you will need to generate such matrices in the next sections, we have already built a function called "make_Bernoulli_matrix." This function takes values of M, N and p as inputs and returns an $(M \times N)$ matrix. In order to generate a matrix like above, you have to type the following in MATLAB

```
>> B = make_Bernoulli_matrix(5,7,.4)

B =

     0     0     0     1     1     1     0
     0     0     0     1     1     0     0
     1     0     0     0     0     0     0
     0     0     0     1     1     0     1
     0     1     0     1     1     0     0
```

4 Distributions Generated From The Bernoulli Experiment

Various distributions may be traced to measurements in the Bernoulli experiment. Examples include the binomial, the geometric, and the Pascal. These distributions are related to the Bernoulli experiment as follows. The Bernoulli random variable is 1 or 0, depending on the value of a single binary transmission. The binomial rv counts the number of binary ones in N transmissions. The geometric counts the number of transmissions until the first 1 and the Pascal distribution counts the number of transmissions to the r-th one. In this section, you will run experiments to generate (random) realizations for each of these rvs. Then, you will compare histograms for these (random) realizations with pmfs for the exact distributions.

4.1 Binomial Distribution

A binomial experiment is run by summing the results of N Bernoulli trials, namely $X = U_1 + U_2 + \ldots + U_N$, where U_i is an i.i.d Bernoulli rv. The probability mass function for this is

$$p_X(x) = \begin{pmatrix} N \\ x \end{pmatrix} p^x (1-p)^{N-x} \qquad x = 0, 1, \ldots, N. \tag{2}$$

To run a binomial experiment, you can generate an $(M \times N)$ matrix of Bernoulli rvs and sum the rows. Each of the M sums will be the number of ones (between 0 and N) for a given experiment. The estimated probability of observing x ones ($0 \leq x \leq N$) is the number of times that x ones appear in these M experiments, divided by M.

Let us start by generating a random (8×10) Bernoulli matrix with probability $p = .35$ using the `make_Bernoulli_matrix` function. Type the following in MATLAB

```
>> A = make_Bernoulli_matrix(8,10,.35)

A =

     0     1     0     0     0     0     0     1     0     0
     1     1     0     0     0     0     0     0     1     0
     0     1     0     1     0     0     1     1     0     0
```

```
     0     0     0     1     0     0     1     0     0     0
     0     1     0     1     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     1
     0     1     0     0     0     1     1     0     1     0
     0     0     0     1     0     1     0     1     0     1
```

Now use the homebrew function `bino_rv` to compute realizations of a binomial rv:

```
>> b = bino_rv(A)


b =

     2
     3
     4
     2
     3
     2
     4
     4
```

Open the function `bino_rv` to see that `bino_rv`= $sum(A, 2)$, where $sum(A, 2)$ sums elements in each row. As you can see, each element in the vector `b` has a value between 0 and $N = 10$ which represents the number of ones in that experiment. It is important to understand that the result generated here contains values for different trials of a binomial experiment with parameters $p = 0.35$ and $N = 10$. In other words, if you were given a binomial distribution with different parameters p and N and were asked to generate binomial realizations for $M = 8$ trials, then your results would have looked different from the vector `b` shown above.

4.2 Binomial Histogram

The histogram is a plot of estimated probabilities. You may think of it as an estimated pmf, estimated from experimental realizations. A histogram is a stem plot of estimated probability for X vs. x. The estimated probability of observing x is:

$$\text{estimated probability of } x = \frac{\text{number of occurrences of } x}{\text{number of trials}}.$$

Consider the vector `b` that we generated in the previous section. We know that elements in this vector can only take values between 0 and 10 since they represent the number of ones in a sequence of 0's and 1's of the length 10. The value for each bar of the histogram is computed by counting the number of times that you observe value x in the vector `b`, and dividing it by the number of trials, i.e., which is the length of the vector `b`. This fraction is the height of the stem located at x. For example, the number of times that you observe the value 2 in the vector `b` is 3. So 3 is the number of occurrences of the number 2. The estimated probability of observing the number 2 is 3/8. So the height of the stem at 2 is 3/8.

We have written a function called the "bino_histo" that accepts a Bernoulli matrix as an input and plots a histogram based on the binomial realizations generated from this matrix. To run this function, generate a Bernoulli matrix and call it A. Now type the following command

```
>> bino_histo(A)
```

Assignment

1. Generate `bino_rv` and `bino_histo` for $M = 100$, $N = 30$, $p = 0.5$, and for $M = 1000$, $N = 30$, $p = 0.5$. Explain what you see. Repeat the experiment for different values of p. Could you use your histograms to estimate p if you did not know it?

2. Assume $M = 50$, $N = 100$ and $p = .73$. Run `make_Bernoulli_matrix` and generate a histogram. Write MATLAB code to plot the pmf of a binomial distribution with parameters (N, p) on the same histogram plot. Now repeat the experiment but this time assume $M = 1000$, $N = 100$ and $p = .73$. What do you conclude from these results?

3. Open the `bino_histo` M-file. Describe what each line does.

4.3 The Geometric Distribution

The geometric distribution counts the number of transmissions until the first one is observed. To generate realizations of a geometric rv, generate a Bernoulli matrix. Then find the first one in each row and write down the index. The following function of `A=make_Bernoulli_matrix` does it

```
>> g=geo_rv(A)
```

Open the `geo_rv(A)` M-file to see that the result is a vector g that contains realizations of a geometric random variable. We might call this the geometric experiment.

You can generate histograms for realizations of the geometric rv. MATLAB function `geo_histo(A)` accepts a Bernoulli matrix as input and plots the histogram of the geometric realizations. So to use this function for the Bernoulli matrix A, you have to type

```
>> geo_histo(A)
```

Assignment

4. Generate geometric `geo_rv(A)` and `g=geo_histo(A)` for the Bernoulli matrices generated in assignments 1 and 2. Explain your results.

4.4 The Pascal Distribution

The Pascal distribution counts transmissions until the r-th one is observed.

Assignment

5. Write MATLAB code `Pascal_rv(A)` that accepts a Bernoulli matrix as input, finds the r-th one in each row, and outputs a vector containing the index of the r-th one in each row. If the r-th one is not found in N trials, return $N + 1$.

6. Write MATLAB code for `Pascal_histo(A,r)` that accepts a Bernoulli matrix as input and draws the Pascal histogram generated from this matrix. Then plot the pmf of a Pascal distribution on the same figure with the same parameters that you have used to generate the Bernoulli matrix. What do you conclude?

5 Conclusion

The Bernoulli experiment is the core experiment for generating the binomial, geometric, and Pascal random variables. In this lab we showed how we can use a simple

Bernoulli experiment to generate realizations of these rvs. The histograms computed from experiments were used to estimate pmfs. From these histograms, the (possibly unknown) values of p can be estimated.

A reasonable extension of the idea would be to design a binomial random process as

$$B[n] = \sum_{k=0}^{\infty} u[n - g_k]$$

where $\{u[n]\}$ is the discrete-time unit step sequence and $\{g_k\}_1^\infty$ is a sequence of independent geometric rvs:

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

(g_k: The geometric time between the $(k-1)$st 1 in a sequence of Bernoulli trials and the kth 1.)

Then the probability that $B[N]$, the number of ones in N trials is $\leq r$, is:

$$P\left[B[N] \leq r\right] = P\left[g_1 + \ldots + g_r > N\right] = P\left[\text{Pascal}(r) > N\right].$$

The probability mass function is:

$$P_{B[N]}[r] = \binom{n}{k} p^r (1-p)^{N-r},$$

which is binomial.