ECON 607 Assignment 9

Andrew Girgis

November 27, 2023

Contents

1 Numerical Integration **3**

1.1 Lottery Game 3

1.2 Inverse Mills Ratio 3

1.3 Computing the expected payoff of the final round 3

1.4 Computing the expected payoff of the first and second rounds . . 4

1.5 Why are the expected payoffs decreasing? 5

1.6 Monte Carlo simulation 6

2 Code **7**

1 Numerical Integration

1.1 Lottery Game

In this assignment we will be analyzing the following lottery game. In the lottery, we have up to three opportunities to draw a random cash prize from the normal distribution with mean $\mu = 1.5$ and variance $\sigma^2 = 1.5^2$. We will denote the first random draw by X_1. If we accept X_1 as our cash prize, the game is over and our payoff is X_1. If we reject X_1, then we continue to round two and draw a second cash prize amount denoted by X_2. Similarly, if we accept X_2 as your cash prize, the game is over and your payoff is equal to X_2. Finally if we reject X_2, then we continue to the third round and draw a final cash prize amount denoted by X_3. If we accept X_3 as your cash prize, the game is over and our payoff is equal to X_3. If we reject X_3, the game is over and our payoff is equal to zero. It is important to note that in this lottery it is possible to receive a negative cash prize where we would have to pay that amount, whereas a positive cash prize is the amount paid to us. The challenge of this lottery is to decide at each stage whether to accept the current prize or continue to the next round. We will base our decisions off of the expected payoffs of each round.

1.2 Inverse Mills Ratio

The inverse Mills ratio is the ratio of the probability density function to the complementary cumulative distribution function of a distribution. Its use is often motivated by the following property of the truncated normal distribution: If X is a random variable having a normal distribution with mean μ and variance σ^2, then

$$\mathbb{E}[X|X > x] = \mu + \sigma \frac{\phi(\frac{x-\mu}{\sigma})}{1 - \Phi(\frac{x-\mu}{\sigma})} \tag{1}$$

where

- x is a constant
- ϕ denotes the standard normal density function
- Φ is the standard normal cumulative distribution function

The fraction

$$\frac{\phi(\frac{x-\mu}{\sigma})}{1 - \Phi(\frac{x-\mu}{\sigma})} \tag{2}$$

is the inverse Mills ratio.

1.3 Computing the expected payoff of the final round

Lets denote the expected payoff of round n as V_n. The expected payoff of round 3 is denoted V_3. Since we know that the third round is the final round of the lottery game we know that the expected value is the max of the payoff of the lottery(x_3) and 0 since if the final lottery ends negative we would just reject the final lottery leaving us with a payoff of 0. To compute the expected payoff of round 3(V_3):

$$V_3 = \mathbb{E}[\max(x_3, 0)]$$
$$\implies \int_{-\infty}^{\infty} \max(x_3, 0)\, dF(x_3)$$
$$\implies \int_{-\infty}^{0} 0\, dF(x_3) + \int_{0}^{\infty} x_3\, dF(x_3)$$

Since

$$\int_{-\infty}^{0} 0\, dF(x_3) = 0$$

We are left with

$$\implies \int_{0}^{\infty} x_3\, dF(x_3)$$
$$\implies (1 - F(0)) \int_{0}^{\infty} x_3 \frac{f(x_3}{1 - F(0)}\, dF(x_3)$$
$$\implies \mathbb{E}[X|X_3 > 0]$$

From equation 1 we know that

$$\implies (1 - F(0))[\mu + \sigma \frac{\phi(\frac{0-\mu}{\sigma})}{1 - \Phi(\frac{0-\mu}{\sigma})}]$$

Therefore

$$\implies V_3 = (1 - F(0))[\mu + \sigma \frac{\phi(\frac{0-\mu}{\sigma})}{1 - \Phi(\frac{0-\mu}{\sigma})}] \tag{3}$$

Using this semi-closed form expression in our Matlab code we approximate an expected payoff from round three(V_3) of 1.6250 $\rightarrow$ \$1.63.

1.4 Computing the expected payoff of the first and second rounds

Using the same method used to compute the second round's expected payoff

$$V_2 = \mathbb{E}[\max(x_2, V_3)]$$
$$\implies \int_{-\infty}^{\infty} \max(x_2, V_3)\, dF(x_2)$$
$$\implies \int_{-\infty}^{V_3} V_3\, dF(x_2) + \int_{V_3}^{\infty} x_2\, dF(x_2) V_3$$

After integration we are left with

$$\implies V_2 = (1 - F(V_3))[\mu + \sigma \frac{\phi(\frac{V_3-\mu}{\sigma})}{1 - \Phi(\frac{V_3-\mu}{\sigma})}]V_3 \tag{4}$$

and similarly for V_1

$$V_1 = \mathbb{E}[\max(x_1, V_2)]$$
$$\implies \int_{-\infty}^{\infty} \max(x_1, V_2)\, dF(x_1)$$
$$\implies \int_{-\infty}^{V_2} V_2\, dF(x_1) + \int_{V_2}^{\infty} x_1\, dF(x_1) V_2$$

After integration we are left with

$$\implies V_1 = (1 - F(V_2))[\mu + \sigma \frac{\phi(\frac{V_2-\mu}{\sigma})}{1 - \Phi(\frac{V_2-\mu}{\sigma})}]V_2 \tag{5}$$

Therefore after using equation 4 and equation 5 for computation in matlab we approximate an expected payoff from round $2(V_2)$ of $2.1630 \rightarrow \$2.16$ and an expected payoff from round $1(V_1)$ of $2.4874 \rightarrow \$2.49$

1.5 Why are the expected payoffs decreasing?

In this lottery game, as described in Section 1.1, the decision-making process spans three rounds, offering choices between accepting the current cash prize or advancing to the next round. The primary aim is to maximize the expected payoff.

In the initial round (Round 1), the potential for up to three rounds initially boosts the expected payoff. If X_1 is accepted, the payoff is X_1. We establish a decision rule: if $X_1 > V_2$ where V_2 is the expected payoff of round 2, then we accept the payoff; if the lottery payment of round one is less than the expected payoff of round $2(V_2)$, we reject the payoff. This results in an expected payoff where $V_1 > V_2$ since V_2 is our minimum threshold in round 1. If X_1 is declined and we move to Round 2, here there's a chance for up to two rounds. Here accepting X_2 yields a payoff of X_2. In Round 2 we establish a decision rule based on the value of X_2 compared to the expected payoff of Round 3 (V_3), using V_3 as the minimum threshold. Similarly to round 1, this establishes $V_2 > V_3$ since V_3 is our minimum threshold in round 2. If X_2 is rejected, in the final Round, with only one chance remaining, the decision involves choosing between X_3 and 0, with 0 as the minimum threshold. Opting for the maximum between X_3 and 0 calculates the expected payoff at the start of Round 3 (V_3).

The reasoning emphasizes the consistent use of the expected value of the next round as the minimum threshold. With more potential rounds in the earlier stages, the expected payoffs decrease as we progress through the rounds, resulting in $V_1 > V_2 > V_3$.

1.6 Monte Carlo simulation

Using V_3 from equation 3 as our optimal accept/reject threshold in round 2 and V_2 from equation 4 as our optimal accept/reject threshold in round 1 we use a Monte Carlo simulation to estimate the expected payoff, V_1. After computing the Monte Carlo simulation under the optimal threshold strategy, we approximate an average expected payoff of $2.4602 \rightarrow \$2.46$.

After conducting the above Monte Carlo simulation 500 times we obtain an average of the average expected payoff of $2.4839 \rightarrow \$2.49$ with a standard deviation(σ) of 0.0388. Here we can see after running the lottery 500,000 times and taking the average payoff under the optimal strategy we can see that the average expected payoff is incredibly close to our expected payoff of round $1(V_1)$.

2 Code

```
clear

% Set our mean, variance and standard deviation
mu = 1.5;
var = 1.5^2;
std_dev = sqrt(var);

% Compute the CDF at 0
cdf_0 = normcdf(0, mu, std_dev);

% Compute the pdf and cdf at x = 0
pdf_x0 = normpdf((0-mu)/std_dev);
cdf_x0 = normcdf((0-mu)/std_dev);

% Compute the expected payoff in round 3 using the inverse mills ratio
v_3 = (1-cdf_0) * (mu + std_dev * (pdf_x0/(1-cdf_x0)));


% Compute the pdf and cdf at x = v_3
pdf_xv3 = normpdf((v_3-mu)/std_dev);
cdf_xv3 = normcdf((v_3-mu)/std_dev);

% Compute the expected payoff in round 2 using expected payoff of round 3
% and the inverse mills ratio
v_2 = (1-cdf_xv3) * (mu + std_dev * (pdf_xv3/(1-cdf_xv3)))+(cdf_xv3)*v_3;

% Compute the pdf and cdf at x = v_2
pdf_xv2 = normpdf((v_2-mu)/std_dev);
cdf_xv2 = normcdf((v_2-mu)/std_dev);

% Compute the expected payoff in round 1 using expected payoff of round 2
% and the inverse mills ratio
v_1 = (1-cdf_xv2) * (mu + std_dev * (pdf_xv2/(1-cdf_xv2)))+(cdf_xv2)*v_2;

fprintf('Expected payoffs from lottery:\n');
fprintf('Expected payoff from round 3:  %.4f\n', v_3)
fprintf('Expected payoff from round 2:  %.4f\n', v_2)
fprintf('Expected payoff from round 1:  %.4f\n', v_1)

% Set N for number of loops/simulations
N = 1000;
```

```
% Initialize one-dimensional array as a (Nx1) matrix of 0s
% to store optimal payoff
opt_pay = zeros(N, 1);

for i = 1:N
    % Create a (1x3) matrix of normal randomly generated numbers with a
    % mean of mu and standard deviation of std_dev
    r = normrnd(mu, std_dev, [1, 3]);

    % Check if random numbers meet the accept threshold
    if r(1) > v_2
        % Store the first lottery in the opt_pay array if larger then
        % threshold
        opt_pay(i) = r(1);
    else
        if r(2) > v_3
            % Store the second lottery in the opt_pay array if larger than
            % threshold
            opt_pay(i) = r(2);
        else
            if r(3) >= 0
                % Store the third lottery in the opt_pay array if larger
                % than 0
                opt_pay(i) = r(3);
            else
                % If we dont accept any of the lotteries then we
                % necessarily rejected all therefore store a 0
                opt_pay(i) = 0;
            end
        end
    end
end

% Display the average opt_pay
disp('Optimal Payoff Average:');
disp(mean(opt_pay));

% Set N_2 for number of loops/simulations of the loops/simulations
N_2 = 500;

% Initialize an empty list/array to store optimal payoff means
opt_pay_mean = [];

for j = 1:N_2
% Initialize one-dimensional array as a (Nx1) matrix of 0s
% to store optimal payoff
opt_pay = zeros(N, 1);

    for i = 1:N
        % Create a (1x3) matrix of normal randomly generated numbers with a
        % mean of mu and standard deviation of std_dev
        r = normrnd(mu, std_dev, [1, 3]);

        % Check if random numbers meet the accept threshold
        if r(1) > v_2
            % Store the first lottery in the opt_pay array if larger then
            % threshold
            opt_pay(i) = r(1);
        else
            if r(2) > v_3
                % Store the second lottery in the opt_pay array if larger than
                % threshold
                opt_pay(i) = r(2);
            else
                if r(3) >= 0
                    % Store the third lottery in the opt_pay array if larger
                    % than 0
                    opt_pay(i) = r(3);
                else
                    % If we dont accept any of the lotteries then we
                    % necessarily rejected all therefore store a 0
                    opt_pay(i) = 0;
                end
            end
        end
    end
    % Calculate and store the mean for the current iteration
    opt_pay_mean(j) = mean(opt_pay);
end

disp('Mean of 500 Optimal payoff averages:');
disp(mean(opt_pay_mean));
disp('Standard deviation of 500 Optimal payoff averages:');
disp(std(opt_pay_mean));
```