Matlab assignment

Deadline: Wednesday, January 3, 2024, 23.59 (CET).

1 Optimal stopping problem

a) **[5 points]** To answer question a. Firstly I determined the amount of boxes by taking the leinght of provided pdfs and made and made an array to store threshold values for each box. Then I calculated the threshold of second to last box as the expected value of the last box, and the last box is left at 0 because the player will always pick last box. Then as instructed I used backward recursion starting from n-2 box because the last box and second to last box already have caculated values. Then I basically just took the given function of calculating thresholds in assignment and thought backward what i needed to calculate the expected value. I made a function of which I need to take integral to get the expected value so I could get the threshold. Then I integrate it for each box which gives me the value of each threshold. Then after the loop is done I calculate the expected value of payoff by taking the integral from the function with respect to the first box threshold and first box pdf.

b) **[4 points]** First, i made variables for all of the given parameters. Then i made functions for each given pdf, I had problems with half-normal pdf but I coded it without taking a built-in function for it, I took the definition from matlab website so it should be correct. Then I made an array to store pdfs for each box , and assigned each pdf to the correct box, I had some problems doing this at first, but I found a solution, the loop basically makes it such that pdfs to the boxes are assigned cyclically, with the Half-normal, Gamma, and Uniform distributions assigned to every third box starting from the first, fourth and seventh box respectively. So this would work for any amount of boxes. And in the end, when the array of pdfs is made I just call the function that I made in A.

c) **[5 points]** Given a generated vector of realised values and a threshold vector, I've created a function that selects the first box which exceeds the threshold value for any given probability. If such a value is never reached, then the last box is picked.

d) **[2 point]** Using preset probability distribution functions as the ones defined in b), with an input vector of the specified parameters, I created a function that selects a set number of realizations from each by iterating through the vector of pdf's, where each outputted column represents realisations from a specific distribution.

e) **[3 points]** First, I've created a matrix of augmented thresholds by taking a 41-dimensional vector of alpha, ranging from 0 to 2 in increments of 1/40. This vector is multiplied with the pre-calculated threshold vector to obtain the augmented threshold matrix. In this matrix, each column corresponds to the pre-prepared threshold vector multiplied by a respective value of alpha. Column one is the threshold vector multiplied by 0, column two is multiplied by 1/40, column three by 2/40, and so on.

Having generated a million realizations for each of the distributions, I apply the Choose-Box function to the generated random picks along with each modified threshold vector

(each column of augmented_threshold_matrix). The resulting values are stored in Matrix_of_vistar.

A column-wise mean is then calculated to produce the average payoff for each particular choice of alpha. Finally, I plot the average results according to alpha as requested. The average payoff appears to increase with alpha until around 3.8, after which it drops to a constant level. This looks suspicious, and I suspect that it has to do with the fact that I have not used the anonymous functions done by my colleague in point b (due to my unabashed lack of Matlab ability at that level). Normally, I would've expected the average payoff to reach a maximum point at 1 and then decrease.

f) **[1 point]** First function from a) is called to get the optimal expected payoff, then we look at part e) to see what the simulated average optimal payoff for when α=1, then the absolute difference between the two are computed and displayed

g) **[2 points]** First the number of entities is got by checking the size of realization matrix V then 3 empty vectors are created to speed up the computations. Then in a loop with runs trough all the d entries thresholds are found, then the entry which satisfies this threshold is found. After that we compute the payoff of this entry by getting its realized value from the matrix V, after that we get the entries utility by subtracting costs. Lastly the mean value of the vector is taken to get the average utility.

h) **[3 points]** First the objective function to be minimized by fminsearch is defined. Since our goal is to maximise the average utility we use minimise -g with respect to beta. Then initial guess for beta is created which consists of ones, as they are non-negative. After that we find the optimal beta using fminsearch, the objective function and the initial guess. Lastly we call the function g using the optimal beta we just got, threshold vector,cost vector and the realized payoff matrix.

i) **[2 points]** First a realization matrix is created which consists of 10000 Unif(0,15) entries. Then an array which consists of 25 Unif(0,15) pds is created. Finally the cost vector as defined in the rules of the question is created. We use the pdf vector to get the threshold vector using the function created in a), then the function from h) is called using the the threshold vector from above, cost vector and realization vector to obtain the optimal B and the resulting utility.

j) **[3 points]** In order to create a CDF of the k-th order statistics with inputs x, probability distribution *F*, sample size *n* and *k*th order statistic. First we create an array with different coefficients of the N chooses K function (the coefficients of the Pascal's triangle). Having the coefficients, we apply the formula of the cumulative distribution function. Later, we create a median function called *mediian*, which returns the actual (or a very close approximation) median of the certain distribution function using the *fzero* function so that *cdf=0.5* and a mean of the distribution function as the initial guess.

k) **[1 point]** Having tested the function from Question J on a Halfnormal distribution with μ = 1 and σ = 11 for k = 66 and n = 100, we can conclude that the its median is 11.3765.

ℓ) **[6 points]** In part ℓ, 2 simulation functions are created. The first one *simulate* generates a random number of simulations (*nsims*) times n, which is the sample size. The second

simulation function called *kbest* inputs the probability distribution F, number of simulations and a sample size. The function gets an approximate k-th order statistics that gives the highest payoff for a certain number of simulations. Within the function, first a vector of 0s is created, which later is updated my the mean of the medians returned from the simulations. In order to make the calculations quicker, a threshold is defined for every iteration apart from the last one as it converges to the actual median. In the end, a mean payoff is calculated over the iterations and select k, which yields the highest.
For the selection of distributions, we decided to pick distributions with different skeweness; a standard normal, which is symmetric, a *Gamma (2,1)*, which is right skewed and a *beta(20,2)*, which is left skewed.
Couldn't get a optimal k-th order statistic for the 3 distributions (error in the code)

2 Who did what?

Rudolfs Jansons (2080485) a,b
Andrei Agapie (2075694) c,d,e
Dans Lismanis (2080683) f,g,h
Nikodem Baehr (2076515) j,k,l