CASE STUDY 048
[Python]
Proving the Secretary Problem using Python

Difficulty Level: 3 of 3

The secretary problem is a famous problem that demonstrates a scenario involving the **optimal stopping theory**. The problem has been studied extensively in the fields of applied probability, statistics, and decision theory. It is also known as the **marriage problem**, the **sultan's dowry problem**, the **fussy suitor problem**, the **googol game**, and the **best choice problem**.

The basic form of the problem is the following: imagine an administrator who wants to hire the best secretary out of n rankable applicants for a position. The applicants are interviewed one by one in random order. A decision about each particular applicant is to be made immediately after the interview. Once rejected, an applicant cannot be recalled. During the interview, the administrator can rank the applicant among all applicants interviewed so far but is unaware of the quality of yet unseen applicants. The question is about the optimal strategy (stopping rule) to maximize the probability of selecting the best applicant. The difficulty is that the decision must be made immediately.

The problem has an elegant solution, proving that the optimal win probability is always at least **1/e**. The optimal stopping rule prescribes always rejecting the first **1/e %** applicants that are interviewed at the look stage, and then stopping at the first applicant who is better than every applicant interviewed so far at the commit stage. One reason why the secretary problem has received so much attention is that the optimal policy for the problem (the stopping rule) is simple and selects the single best candidate about 37% of the time, irrespective of whether there are 100 or 100 million applicants.

e *is the base of the natural logarithm*

Although there are many variations, the basic problem can be stated as follows:

- There is a single position to fill.
- There are n applicants for the position, and the value of n is known.
- The applicants, if seen altogether, can be ranked from best to worst unambiguously.
- The applicants are interviewed sequentially in random order, with each order being equally likely.
- Immediately after an interview, the interviewed applicant is either accepted or rejected, and the decision is irrevocable.
- The decision to accept or reject an applicant can be based only on the relative ranks of the applicants interviewed so far.
- The objective of the general solution is to have the highest probability of selecting the best applicant of the whole group. This is the same as maximizing the expected payoff, with payoff defined to be one for the best applicant and zero otherwise.

To prove that this solution is optimal, we need to build a trial that should be executed 1,000 times and count the number of successes.

1) Define the sample size and the number of trials as 1,000.

2) Build a function called get_optimal_stop() that receives the sample size and the stop value and return an integer representing the position of the optimal stop. The stop value parameter should have the default value of 1/e.

3) Build a function called create_pool() to simulate a pool of applicants. The function should receive the sample size and return an array containing the relative rank indicating the skill from 0 to sample size -1 in random order. Suppose that the sample size is 10, the resulting array could be:

index	0	1	2	3	4	5	6	7	8	9
skill	7	2	1	9	8	3	4	0	6	5

4) Build a function called look_stage() that receives the array containing the pool of applicants and the optimal stop, and returns the best applicant of the beginning of the array until the optimal stop position.

5) Build a function called commit_stage() that receives the array containing the pool of applicants, the optimal stop and the best applicant of the look stage, and returns the first applicant that is better than the threshold defined in the look stage from the optimal stop until the end of the array.

6) Build a function called best_of_all() that receives the array containing the pool of applicants and return the best of all;

7) Build a function called one_round() with no parameters, that calls create_pool(), look_stage(), commit_stage(), best_of_all(). If the commit_stage() returns the same value of best_of_all(), the strategy was successfully executed and should return 1. Otherwise, returns 0.

8) Define the optimal stop using just the sample_size parameter (using the default stop_value), and run the one_round() function for the number of trials, counting the success cases and print the result.

9) Call the optimal_stop() passing the stop_value as 0.10 and run the experiment again.

10) Call the optimal_stop() passing the stop_value as 0.90 and run the experiment again.

Good luck!

Difficulty note: this is a difficult assignment. Do not be surprised that there will be lots of nuances we have not covered off in the courses. But just like in the Real Life – there will be things training has not prepared you for and you will need to do research to find how to solve the problems at hand. If you get stuck, check the clues file.