Politecnico di Torino

Reinforcement Learning for Airline Tickets Pricing

Numerical Optimization and Stochastic Optimization for Large Scale Problems

Giovanbattista Tarantino 338137,
Massimiliano Carli 337728,
Samuele Vanini 318684

1 Objective and Problem Description

This report provides a comprehensive overview of the methodologies, techniques, and decisions employed in the development of a Reinforcement Learning algorithm designed to optimize ticket pricing and inventory management. The task to solve is inspired by a real-world scenario, where a flight company needs to dynamically set the price of a particular flight taking into account the tickets and the time remaining.
The project's objective is therefore the creation of an algorithm that autonomously determines ticket prices to maximize revenue while effectively managing inventory over a finite time horizon.
The problem's domain is limited to a single company trying to sell its tickets. It does not consider any other flight companies or the comparison of prices between them.
Customers arrives according to a Poisson process and each has a perceived value for the flight Θ, which does not depend on the time, and follows a beta distribution (Beta(α, β)). The parameters for the Poisson process, beta distribution, and price range have been left open for selection by the user.

2 Problem Formalization and Assumptions made

To clearly define the boundaries of the project, several assumptions have been made. First, the price range is fixed with designated maximum and minimum values, subdivided into N bins. To simulate the arrival of customers we decided to interpret the Poisson process as a counting process. This gave us the possibility of simulating it using an Exponential distribution ($\exp(\lambda)$) where the inter-arrival time between consecutive customers is sampled from said distribution. These parameters can be specified by the user prior to runtime.
We can then model the problem as a Markov Decision Process; specifically, we defined:

- A set of actions A consisting of setting the price from the available bins.
- A set of states S formed as the tuple (tickets left to sell, price fixed).
- A performance metric R(s,a) that provides an estimate of the possible revenue considering how many customers try to buy a ticket in a day.
- A transaction matrix T that describes the probability of moving from one state to another given a specific action.

Additionally, to determine the importance to be in a certain state at a certain time, we defined the Value Table $V_t(s)$, in which every element $V_t(s_j)$ represent the value of the state s_j at time t.

3 Methodology

We employed dynamic programming techniques to implement the value iteration algorithm for a finite time horizon as shown in Algorithm 1. Given the formalization and assumptions

Algorithm 1 Value iteration for finite time horizon

1: **for** $t \leftarrow horizon$ **to** 0 **do**
2: **for all** s **do**
3: $V_t(s) = opt_{a \in A(s)} f(s,a) + \gamma \sum_{s' \in S} \pi_{s,a,s'} \times V_{t-1}(s')$
4: **end for**
5: **end for**

previously described, the Value Table, and both the Rewards and the Transition Probabilities have been implemented using NumPy [1] matrices to speed up the computational time required to converge to an optimal policy. More specifically, all three matrices (Value Table, Rewards, and Transition Probabilities) are initialized before the algorithm runs the following way:
For each price bin the probability of selling a ticket is given by:

$$p = S_{\alpha,\beta}(b) = 1 - CDF_{\alpha,\beta}(b)$$

where $S_{\alpha,\beta}$ is the survival function for the Beta distribution; the probability of selling k tickets at price b when on average n customers try to buy is computed as:

$$p_{b,k} = \binom{n}{k} \times p^k \times (1-p)^{(n-k)}$$

$p_{b,k}$ is computed for each k (normalizing the results to have sum equal to one) in every combination of tickets left and fixed price bin. This results are then used to populate the transaction matrix, where for each state (tickets left, price bin) we have the probability of moving to a state with the same price bin but different number of tickets.
For the reward matrix the construction is a bit different. Since being in a specific price bin and moving to another one does not influence the single customer's purchase probability, we only store two dimensions: one for the tickets left and one for the price bin in which we want to sell the tickets. Each entry $R[s,b]$ is calculated using the formula:

$$\sum_{k=0}^{s} price \times k \times p_{b,k} \forall s \in S$$

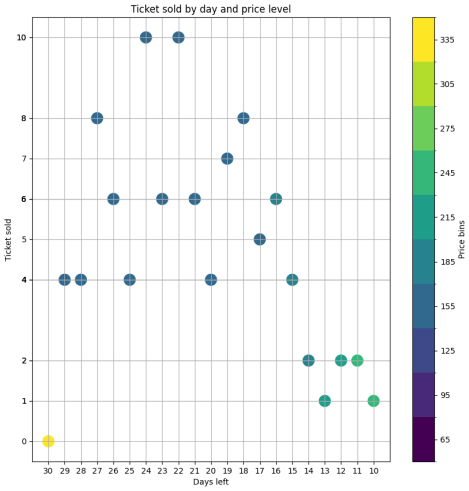
Finally, the Value Table is initialized with all zeros except for the day of the flight; in this case, for each combination of price bin and tickets left, the entry is initialized as follows to represent the cost of unsold flight tickets to the company:

$$n_{tickets} \times c$$

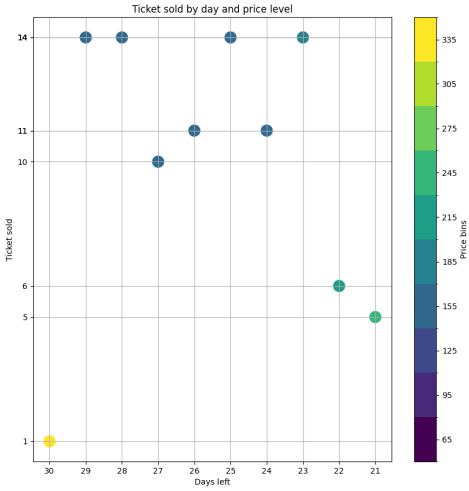
where $n_{tickets}$ is the number of tickets left unsold and c is the fixed cost of a seat.

4 Experiments

The following two plots (1a, 1b) show how the algorithm behaves with fixed parameters (total number of tickets = 100, $\alpha = 2, \beta = 2$, days to sell the tickets = 30) and a different number of expected customers each day.
Both figures show how the optimal policy extracted from the Value Table behaves. To obtain them, we generate a customer trace (i.e. a list of all the customers with the corresponding arrival day and perceived ticket's value Θ) simulated using the same hyperparameters used during training.



(a) Strategy with 10 customers per day sold all tickets, total revenue: 16310$



(b) Strategy with 20 customers per day sold all tickets, total revenue: 16910$

The results obtained confirm our expectations; the algorithm at the start maintains a high price since many days are left to sell tickets. The closer we get to the horizon end, the lower the price is set to sell as many tickets as possible. We saw a change of behavior when all the parameters are left untouched except for the average number of customers expected per day. Increasing this value, the algorithm recognizes a greater chance of selling more tickets at a more profitable price and therefore sets the fixed selling value higher.

5 Conclusions

We are satisfied by the results obtained in the current setting. The policy constructed by the agent is coherent with our expectation and follows a quite intuitive direction. We believe that the work could possibly be expanded by adding a shift in distribution of the perceived value based on other factors like the time remaining, the tickets left and the overall supply in the whole market.

References

[1] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.