Dynamic Ticket Pricing with Reinforcement Learning

Sammy Kao, Reed Bodley, Javier Gonzalez

Tufts COMP 138, Reinforcement Learning

October 29, 2025

Part 1: Introduction

Dynamic pricing models are essential in modern markets. Practically every company in the world uses pricing strategies that change with the change in demand. The main challenge with building these models is determining when and how much to adjust prices to balance revenue maximization with selling the entirety of your inventory. Strategies like last minute discounts or sales fail to adapt optimally to uncertain, time-dependent demand.

This project proposes an autonomous reinforcement learning agent that learns to set ticket prices dynamically for live events. The system models ticket pricing as a sequential decision-making problem, where each pricing decision influences future demand and total revenue. The RL agent observes conditions in the market such as time remaining until the event, current inventory, and recent demand rate, and outputs a new ticket price.

The motivation stems from the real-world difficulty of pricing under uncertainty. Human pricing managers often rely on historical biases and experience, and cannot possibly update prices at the precision and frequency required for optimal results. By learning directly from data, an RL approach can discover strategies that traditional methods overlook, improving both long-term revenue and the probability of selling out all inventory.

Our project will implement, train, and evaluate an RL-based pricing agent using a simulated environment derived from historical/synthetic ticket-sales data. The system will be compared to baseline strategies like fixed pricing and time based linear price.. We expect that the learned policy through the RL agent will exhibit improved adaptability in volatile market conditions and yield higher cumulative rewards, demonstrating the advantages of data-driven, self-optimizing pricing.

Part 2: Background and Related Work

Dynamic pricing has been widely explored in operations research through revenue management and demand models. These models assume a known demand curve and use optimization methods to compute the best prices. However, they often require precise prior knowledge of customer behavior and are sensitive to modeling errors.

Reinforcement Learning offers an alternative by framing pricing as an interaction problem rather than an optimization one. An RL agent learns directly from experience: by taking pricing actions, observing the results of the sales with that price, and adjusting its policy through reward-based feedback. This approach does not rely on an explicit analytical demand model and can adapt to changing market conditions.

This project draws directly on concepts from Sutton and Barto (2018), including the Markov Decision Process, temporal difference learning, and policy improvement through Q-learning. These foundations will guide both our environment modeling and algorithmic design.

Our work also connects to model-based reinforcement learning, where an agent explicitly learns a model of the environment's transitions (here, the change of ticket demand in response to pricing decisions) to enable planning and lookahead. While our initial implementation will employ model-free approaches like Q-learning or Deep Q-Networks, which learn policies directly from experience without an explicit model, future iterations may contain model-based components to help the agent's planning capability and efficiency by leveraging predictive models of market behavior.

Part 3: Environment Platform

3.1 Data-Driven Simulation Environment

The environment will be a custom-built, data-driven simulator based on historical ticket sales data from SeatData.io. Instead of relying on artificial demand curves, the environment will model market dynamics using real pricing and sales patterns extracted from the dataset.

Each simulation episode represents a full ticket sales cycle. At each time step, the agent selects a price, and the simulator predicts ticket sales using patterns derived from the SeatData.io data.

State variables:

$$s = [\text{time\_remaining}, \text{inventory\_remaining}, \text{price\_level}, \text{demand\_rate}]$$

Action space:

$$a \in \{-\Delta p, 0, +\Delta p\}$$

The next state is determined by updating time, reducing inventory, and recalculating demand rate. Each episode ends when tickets sell out or the event date arrives.

3.2 Market Model and Assumptions

$$q_t = \alpha_t \cdot e^{(-\beta p_t)} + \varepsilon_t$$

Variables: q_t (tickets sold), α_t (demand intensity), β (price elasticity), p_t (price), ε_t (noise).

Assumptions: Elastic demand, temporal variation, inventory constraint, and stability penalty for large price jumps.

Part 4: AI Agent

4.1 Algorithm Choice (Tabular Q-learning → DQN Extension)

The project will use a two-stage approach.

Stage 1: Tabular Q-learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Stage 2: Deep Q-Network (DQN)

$$L(\theta) = [r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta)]^2$$

The DQN will have two hidden layers (64 neurons each, ReLU activation) with replay buffer and target network.

4.2 Reward Functions

Revenue-based:

$$R_t = p_t \cdot q_t$$

Balanced reward:

$$R_t = (p_t \cdot q_t) + \lambda_1 \cdot \text{SellOutBonus} - \lambda_2 \cdot |p_t - p_{t-1}|$$

4.3 Policy Representation and Learning Process

Tabular $\pi(s)$ uses discrete mappings; DQN $\pi(s) = \arg\max_a Q(s, a; \theta)$ uses gradient descent updates and replay sampling.

Part 5: Architecture and Evaluation

5.1 Architecture Components

- **Data Module:** Preprocess SeatData.io data.
- **Environment Module:** Simulates market demand and transitions.
- **Agent Module:** Contains Q-learning and DQN agents.
- **Training & Evaluation Module:** Handles experiment tracking and visualization.

5.2 Evaluation Metrics

Metrics: Total Revenue, Sell-through Rate, Price Stability, Convergence Speed, and Realism.

Baselines: Static Pricing and Heuristic Time Decay.

RL agents are expected to outperform baselines in both revenue and sell-through rate while maintaining stable pricing.

Part 6: Timeline and Responsibilities

Timeline:

- SeatData.io Pipeline (by Nov 8): The first and arguably most important step of building the dynamic ticket pricing agent is to ensure that we have a clear understanding of the data that is provided from SeatData.io, both in its form and its content. We will dedicate the first segment of our time to building the pipeline from SeatData.io to our project space, and then building out our environment for the experiment from historical prices, inventory, and sales etc. We can run an initial experiment to ensure that our data is retrievable and is the expected form. We should have a data structure containing the historical data from the time-period that we need before moving on to the next step.
- Tabular Q-Learning Implementation (Nov 8–20): Now that we have established our experimental environment based on the data coming from SeatData.io, we can begin to build our Tabular Q-Learning Agent based

on the experimental environment. We will build out our projected implementation of the agent to a simple degree based on the time-period of data that we have selected. The key deliverables will be setting up the Q-lookup table, implementing the epsilon-greedy exploration, and have a basic way to output the performance of the agent (either on the command line or basic visualization with graphs).

- Agent Optimization (Nov 20–30): This is the fun part, now that we have built out the basic functionality for our environment and the agent is able to use this environment to learn on the historical dataset, we can begin to mess with the parameters to see how we canto optimize the performance of our Q-Learning agent on the SeatData.io data. This can include things like changing the value of epsilon, and other parameters for the Agent, as well as running a deep dive into the trends in SeatData to see if there are any things that we can notice in the data that will help our agent perform better. We would also want to test on different time-periods of data to ensure that our agent can perform in different market conditions as opposed to the main set that we had been working with.
- Final Deliverables and Report (Dec 2–End of Semester): Now that we have optimized our agent across various time-periods of SeatData.io data, we will complete our report/presentation documenting our progress working on the project throughout the semester. It is important to note that we plan to keep good documentation throughout the time that we are working on the project, throughout all stages we want to provide good written insight into the thought process, challenges, and what ended up working for us. If we stick to keeping good documentation throughout the semester then it will make this step of the project much easier, since we will already have most of the content. This step can also include making a "pretty" visual output for our experiment, something that will look good and intuitive for a poster or presentation.

Individual Responsibilities:

- **Sammy:** Responsible for handling the SeatData.io data pipeline and building out the environment to have the parameters and data necessary for our experimental purposes.
- **Javier:** Mainly responsible for the Q-Learning agent algorithm and early stage implementations onto the environment. Essentially picking up from where Sammy left off with the SeatData.io pipeline.
- **Reed:** Tying together the work from both Sammy and Javier to optimize the Agent by running tests on various performance metrics during the experiment. Responsible for taking the optimally performing agent and displaying it's performance in a visually intuitive way.

All members will contribute to debugging, documentation, and presentation preparation.

Part 7: Conclusion and References

This project demonstrates that reinforcement learning can outperform traditional pricing strategies by adapting to real-time demand fluctuations. The RL agent will be trained using a realistic, data-driven simulation environment derived from SeatData.io, optimizing both revenue and sell-through efficiency. Future work could extend this model using continuous-action RL and model-based reinforcement learning for improved generalization and planning.

References

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 2nd Edition, 2018.

SeatData.io Dataset. Accessed 2025. `https://seatdata.io`