Smooth Tabular Reinforcement Learning for Dynamic Pricing

Adam Page[1] James Grant[1] Chris Kirkbride[1] Anna-Lena Sachs[1]
Jeremy Bradley[2] Alexander Green[2] Jordan Palmer[2]
[1]STOR-i Lancaster University, [2]Datasparq

1 Problem

We want to find an optimal pricing strategy for pre-bookable car parks.

Figure 1: We can draw from hotel literature to help us price car parks (Klein et al. 2020).

This involves the dynamic pricing of a perishable good with limited inventory. We make four assumptions:

- Customers have a random maximum price they are willing to pay.
- Customers have a lead time between booking online and arriving.
- There is a limited capacity of spaces we can offer.
- We can only set one price per day.

2 Reinforcement Learning

Reinforcement learning (RL) is a tool which involves an agent taking sequential actions within an environment and receiving reward signals to learn an optimal policy. The agent must balance exploring uncertain actions and exploiting the current best action whilst learning.

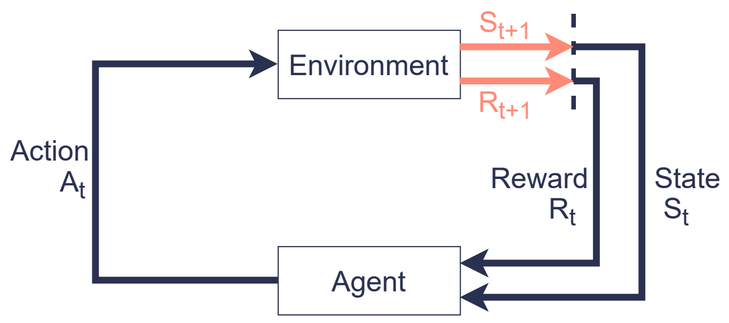


Figure 2: Agent-Environment interactions (Sutton & Barto 2018)

The reward, $R(S, A)$, is the revenue generated for a set price, A, and usage level, S. Action value functions, $Q(S, A)$, store the expected total reward starting from the state-action pair. An ϵ-greedy exploration policy is used. Choosing a random price with probability ϵ, otherwise setting $A = \arg\max_{a^*} Q(S, a^*)$.

3 Q-Learning

The Q-learning agent is a member of a family of algorithms known as temporal difference learners. Every time step the Q-learner updates its Q-value towards the newest observation of the environment.

Figure 3: Back up diagram of the Q-learning algorithm

Introduced by Watkins (1989), each update step consists of the following update rule:

$$Q(S, A) \leftarrow (1 - \alpha)Q(S, A) + \alpha\left(R + \gamma \max_a Q(S', a)\right)$$

- $\alpha :=$ Learning rate/step size.
- $\gamma :=$ Discount factor for look ahead value.
- $S' :=$ New state after taking action A.

4 Smoothness in Pricing

It is likely that similar prices yield similar revenues. The reward function can be shown to be Lipschitz continuous:

$$|R(S, a_1) - R(S, a_2)| \leq L_R\, d_{\mathcal{A}}(a_1, a_2)\,,$$

where $d_{\mathcal{A}}$ is a distance metric on the action space.

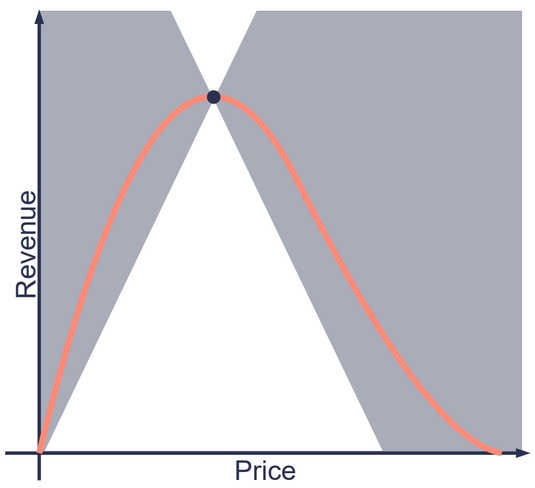


Figure 4: Lipschitz continuity of the revenue function

This shows that the gradient of our reward function is bounded. So there is no sudden jump in revenue when the price changes.

5 Cross-Learning

Novel RL agent which updates multiple Q-values within the same step, leveraging the smoothness.

for *each learning episode* **do**
  Set price, A, dependent on $Q(S, \cdot)$;
  Observe reward R, and new state S';
  Take a Q-learning update step with α;
  for *all other actions,* a_i **do**
    Set $d = d_{\mathcal{A}}(A, a_i)$;
    Set $\widehat{\alpha} < \alpha$ dependent on α, d and L_R;
    Take Q-learning update on a_i with $\widehat{\alpha}$;
  end
  $S \leftarrow S'$;
end

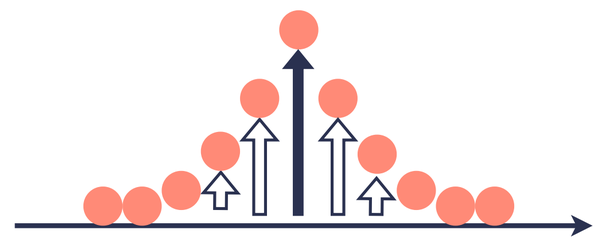
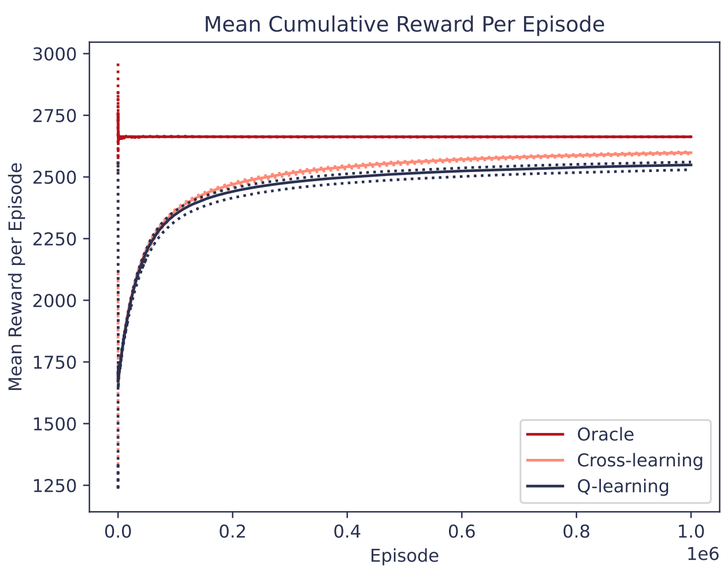


Figure 5: One update pulls up neighbouring Q-values

6 Results

Agents tested on a small instance: selling 100 car parking spaces over 10 time steps.
Both reinforcement learning algorithms perform close to optimal without knowledge of the demand.
Cross-learning outperforms Q-learning in terms of mean and median total revenue.



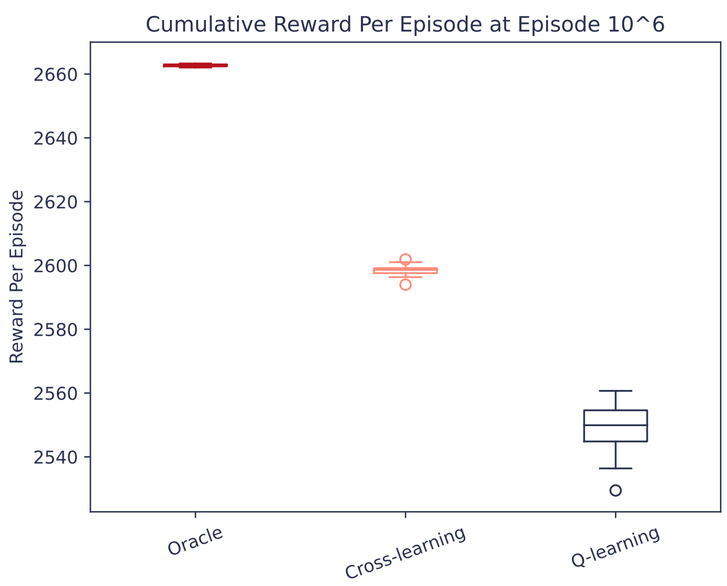


Figure 6: Agents tested on a pricing problem with Log-Uniform demand. Ran for 10^6 episodes with 30 macro-replications. The oracle is provided as a benchmark and is found through dynamic programming. **Left:** Mean cumulative reward per episode. Best and worst runs are shown in dotted lines of the respective algorithm. **Right:** Box plot showing cumulative reward per episode on the final episode, over the 30 macro-replications.

7 State Smoothness

Cross-learning has been found to outperform other tabular forms of RL, on a range of demand distributions (Selcuk & Avşar 2019). The algorithm requires little to no tuning to get these improvements.

A Lipschitz continuity bound can be made for the reward function with respect to states. A Lipschitz constant with respect to both states and actions can also be found. Both of these are weaker; this means more tuning is needed for a successful algorithm.

References

Klein, R., Koch, S., Steinhardt, C. & Strauss, A. K. (2020), 'A review of revenue management: Recent generalizations and advances in industry applications', *European Journal of Operational Research* .

Selcuk, A. M. & Avşar, Z. M. (2019), 'Dynamic pricing in airline revenue management', *Journal of Mathematical Analysis and Applications* .

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement Learning: An Introduction*, second edn, MIT Press.

Watkins, C. J. C. H. (1989), Learning From Delayed Rewards, PhD thesis, King's College, Cambridge United Kingdom.