Reinforcement Learning for Revenue Management and Dynamic Pricing

Supervised by Adam Page

Keelan Adams

Durham University

30/08/24

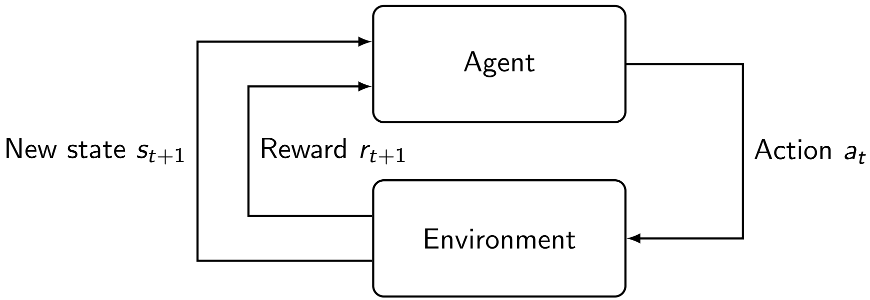
Table of Contents

The Problem

- We are a hotel trying to find an **optimal pricing policy** to maximise our **revenue** for a particular night
- Depends on how many rooms we have already sold
- Only information we have is observing what happens when we set a chosen price

This idea can be applied to any **perishable good**

What is Reinforcement Learning?



- **States** - rooms sold so far
- **Actions** - prices
- **Rewards** - revenue

The function $Q(S, A)$ **estimates** the **expected return** taking action A from state S under a given **policy**

ϵ-Greedy Policy

We choose the value of the **exploration parameter** ϵ.

ϵ-Greedy Action Selection

With probability ϵ:

Explore - Pick a random action

With probability $1 - \epsilon$:

Exploit - Pick the action maximising Q (for the state we're in)

Q-Learning

So how do we update $Q(S, A)$ based on what we've seen?

$$Q(S_{t+1}, A_{t+1}) \leftarrow \underbrace{(1-\alpha)Q(S_t, A_t)}_{\text{old estimate}} + \underbrace{\alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) \right)}_{\text{new observation}}$$

Weighted by **learning rate** α

Table of Contents

Modelling Customers - No States

- Each customer has a random willingness to pay W, which decreases with respect to price
- N customers per episode
- Assume prices (actions a) range from £1 - £100

Modelling Customers - No States

- Each customer has a random willingness to pay W, which decreases with respect to price
- N customers per episode
- Assume prices (actions a) range from £1 - £100

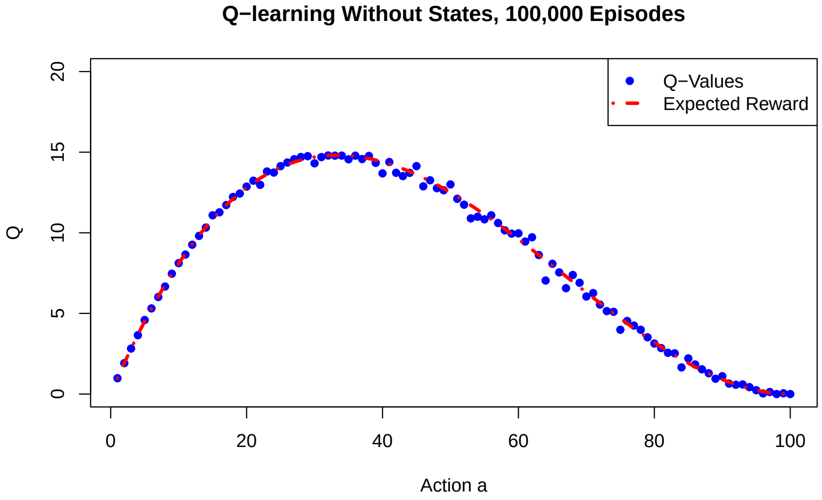
Reward for each customer...

$$r_i(a) = \begin{cases} a & \text{if} \quad W > a \\ 0 & \text{if} \quad W < a \end{cases}$$

Total reward...

$$R(a) = \frac{\sum_{i=1}^{N} r_i(a)}{N}$$

Results - No States



100 rooms. Now not every customer who tries to book will get a room.

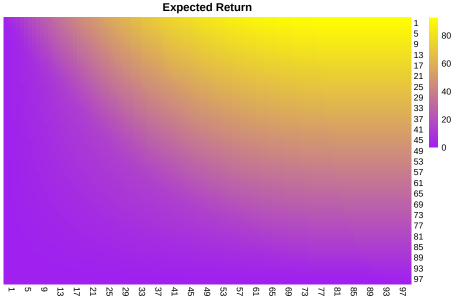


Figure: Expected **return** for taking action *a* (x-axis) starting from state s (y-axis)

Adding States to the Model

100 rooms. Now not every customer who tries to book will get a room.

Figure: Expected **return** for taking action *a* (x-axis) starting from state s (y-axis)

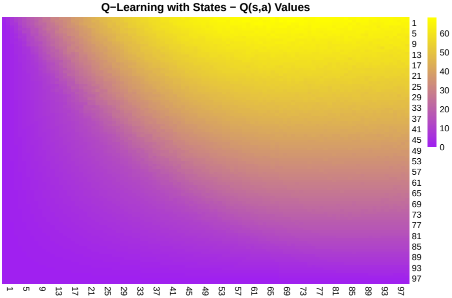


Figure: Estimates of $Q(s, a)$ with 1,000,000 episodes - taking action a (x-axis) starting in state s (y-axis)

Tiered Product

Now assume there are multiple tiers of room.
Different customers will have different ways of selecting their preferred tier.

For this toy example, there are 2 tiers.

Tier	Possible Prices
1	£1 - £10
2	£11 - £20

Actions: $\mathbf{a} = (a_1, a_2)$; $a_1 =$ Tier 1 price; $a_2 =$ Tier 2 price

Now looking at $Q(a_1, a_2)$

Ignore states here... difficult to present as 4 dimensional!

Tiered: Customer 1 - Max Buying

Each customer has their fixed W, goes for most expensive room within their budget

$$r_i(a_1, a_2) = \begin{cases} a_2 & \text{if} \quad W \geq a_2 \\ a_1 & \text{if} \quad a_1 \leq W < a_2 \\ 0 & \text{if} \quad W < a_1 \end{cases}$$

Tiered: Customer 1 - Max Buying

Each customer has their fixed W, goes for most expensive room within their budget

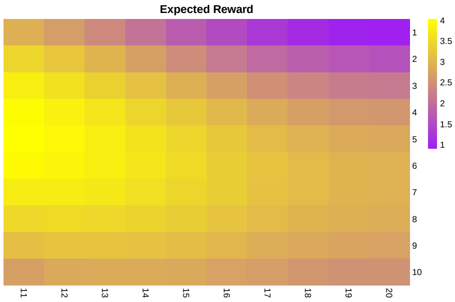


Figure: Expected reward for taking action a_1 (y-axis) and a_2 (x-axis)

Tiered: Customer 1 - Max Buying

Each customer has their fixed W, goes for most expensive room within their budget

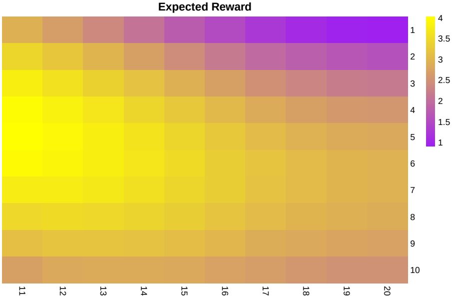


Figure: Expected reward for taking action a_1 (y-axis) and a_2 (x-axis)

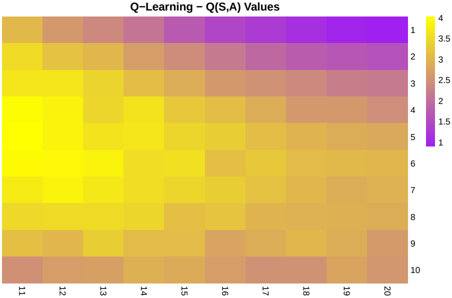


Figure: Estimates of $Q(a_1, a_2)$ with 100,000 episodes - taking action a_1 (y-axis) and a_2 (x-axis)

Tiered: Customer 2 - Desperate

Customer arrives in the middle of the night willing to pay anything for the room they want

Tiered: Customer 2 - Desperate

Customer arrives in the middle of the night willing to pay anything for the room they want

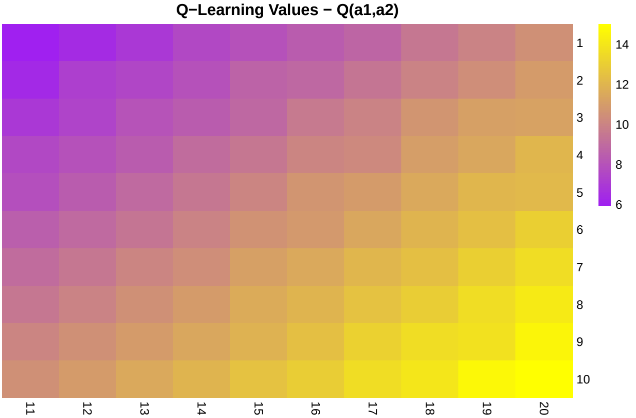


Figure: Estimates of $Q(a_1, a_2)$ with 100,000 episodes - taking action a_1 (y-axis) and a_2 (x-axis)

The Decoy Effect

You're at the cinema. Which is the best deal for popcorn?

Small	**Medium**	**Large**
£3	£6.50	£7

The Decoy Effect

You're at the cinema. Which is the best deal for popcorn?

Small	**Medium**	**Large**
£3	£6.50	£7

The medium option is a **decoy**
Can we get a Q-learner to pick up this pricing strategy?

Tiered: Customer 4 - Utility Maximisation

Customer sets willingness to pay for each tier:

$$\mathbf{W} = (W_1, W_2)$$

Customer wants to maximise (positive) difference:

$$W_i - a_i$$

Tiered: Customer 4 - Utility Maximisation

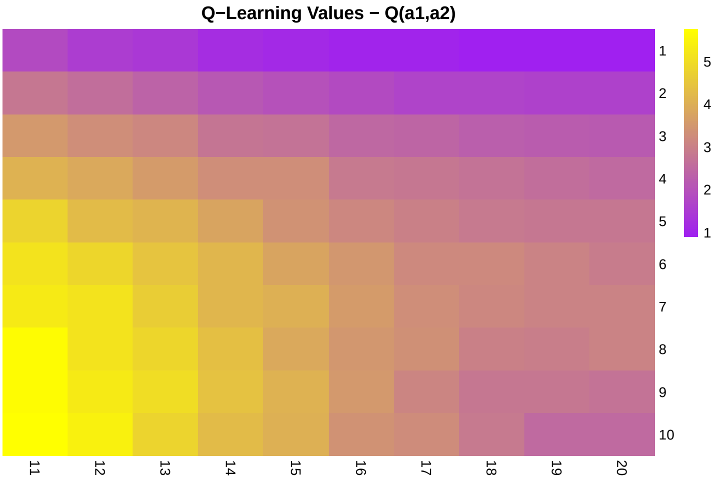


Figure: Estimates of $Q(a_1, a_2)$ with 100,000 episodes - taking action a_1 (y-axis) and a_2 (x-axis)

Tiered: Customer 4 - Utility Maximisation

Do we really need to do joint updates of $Q(a_1, a_2)$?
What if we just had separate updates $Q(a_1)$ and $Q(a_2)$?

Tiered: Customer 4 - Utility Maximisation

Do we really need to do joint updates of $Q(a_1, a_2)$?
What if we just had separate updates $Q(a_1)$ and $Q(a_2)$?

Update	Function	Tier 1	Tier 2
Joint	$Q(a_1, a_2)$	10	11
Separate	$Q(a_1), Q(a_2)$	6	16

Table: Optimal actions based on joint and separate Q-learning updates

Table of Contents

Function Approximation

Computing $Q(S, A)$ for all S, A is costly for large state/action spaces (or impossible for continuous!)

Instead we estimate as a value function:

$$\hat{q}(s, a, \mathbf{w}) \approx q(s, a)$$

We update the parameter $\mathbf{w}$, trying to minimise the mean-squared error between our approximate $\hat{q}$ and true q:

$$J(\mathbf{w}) = \mathbb{E}_{\mathbf{w}}\left[(q(s, a) - \hat{q}(s, \mathbf{w}))^2\right]$$

We adjust $J(\mathbf{w})$ in the direction of negative gradient each episode, to find the global minimum

Other Considerations

Alternatives to ϵ-greedy action selection?

Increasing Model Complexity:

- Booking in Advance
- Multiple Night Stays
- Competition/Locational Factors

Thank you for listening!