Reinforcement Learning for Revenue Management and Dynamic Pricing

Keelan Adams [1] Supervisor: Adam Page [2]

[1]Durham University [2]STOR-i, Lancaster University

The Problem

We want to find an **optimal pricing policy** to maximise the revenue of a hotel.

- **States:** number of rooms sold so far (up to hotel capacity)
- **Actions:** prices we could set
- **Rewards:** money we receive from customers who book
- **Probabilities:** likelihood of customers booking at set price

The function $Q(S, A)$ estimates the expected return when taking action A starting from state S

Reinforcement Learning Methodology

We have to balance **exploration** - trying new actions to learn the values of $Q(S, A)$ across the action space, with **exploitation** - picking the action maximising $Q(S, A)$.

What action do we pick at each step? $\rightarrow$ ϵ-Greedy Policy

- Set exploration parameter ϵ
- **Explore** with probability ϵ and **exploit** with probability $1 - \epsilon$

How do we update our estimate $Q(S, A)$ at each step? $\rightarrow$ Q-Learning

Set learning rate α. We take a weighted average of our current estimate and the return based on what we've just observed:

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

No States - Infinite Number of Rooms

Each customer has a random willingness to pay W, the distribution of which decreases with respect to time. During each episode we observe 100 customers.

Assume we can set prices from £1 - £100. We set our price a, and the customer will book if $W \geq a$, so the reward for each customer is either a or 0.

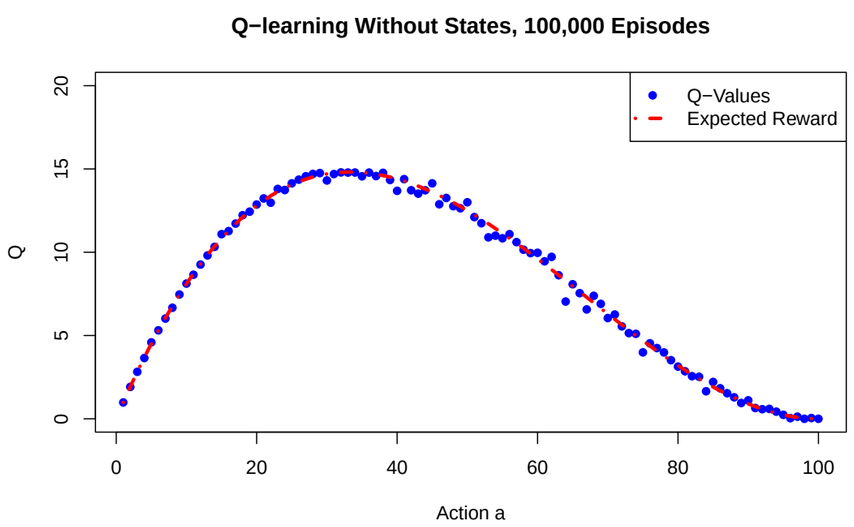

Figure 1. Results using a decaying ϵ-greedy algorithm

Adding States to the Model

We now have a limited number of rooms available to sell, which are offered on a first-come-first-served basis.

Figures 2 and 3 compare the expected return with the Q-learner estimates:

Figure 2. **Expected return** for taking action a (x-axis) starting from state s (y-axis)

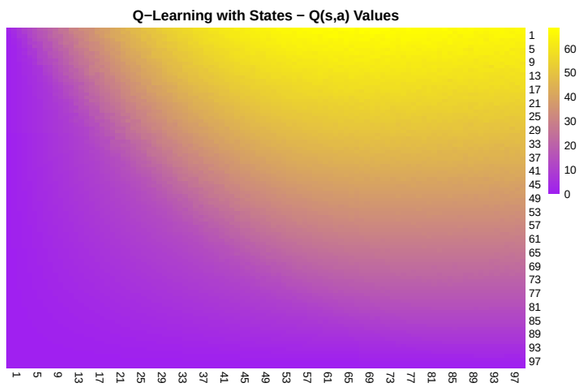

Figure 3. **Q-learning estimates** of Q(s,a) with 1,000,000 episodes - taking action a (x-axis) starting in state s (y-axis)

Tiered Product

Now suppose we have two different **tiers** of room available.

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

Actions: a $= (a_1, a_2)$ - a price for Tier 1 and a price for Tier 2

Different customers will have different preferences. Here we discuss two examples:

Max Buying Customer:

The customer books the most expensive room within their own willingness to pay W.

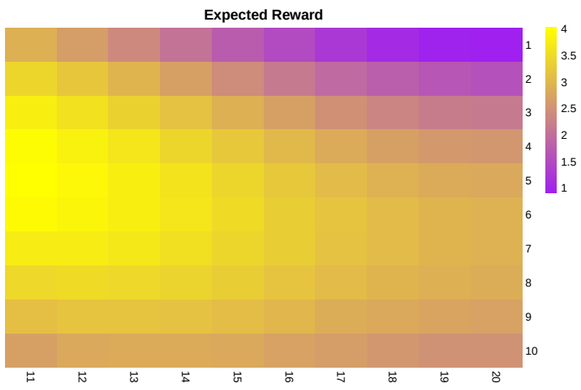

Figure 4. Expected reward when taking actions a_1 (y-axis) and a_2 (x-axis)

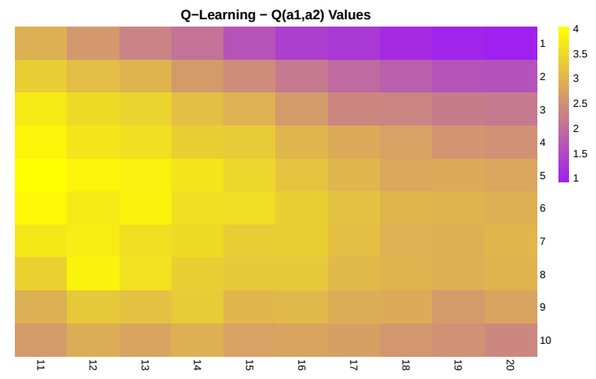

Figure 5. Estimates of $Q(a_1, a_2)$ with 100,000 episodes - taking action a_1 (y-axis) and a_2 (x-axis)

The Decoy Effect

Imagine you're at the cinema buying popcorn. Which would you buy?

Small	Medium	Large
£3	£6.50	£7

By pricing the medium close to the large, it is more likely that customers will trade up to buying the large. Here the medium option is called a **decoy**.
Can we apply this to our Q-learning setup?

Utility Maximisation Customer:

Each customer has a willingness to pay, W_i, for each tier, based on their preferences.

Then the customer wants to maximise $W_i - A_i$, so they get what they see as the best deal.

The heatmap shows our Q-learner has worked out how to use the decoy effect!

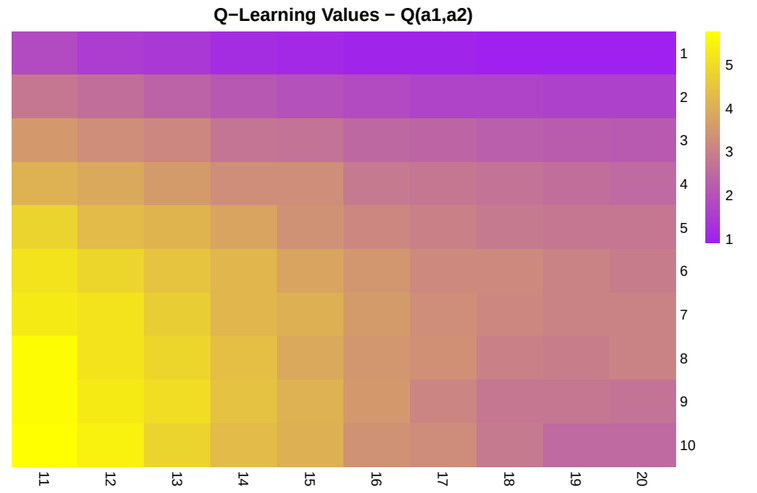

Figure 6. Estimates of $Q(a_1, a_2)$ with 100,000 simulations - taking action a_1 (y-axis) and a_2 (x-axis)

The optimal action here is to set Tier 1 at £10, Tier 2 at £11 - so nearly everyone would choose the more expensive tier.

Further Development

Function Approximation:

Q-learning is costly for large state/action spaces (or impossible for continuous!).

We can instead estimate $q(s, a)$ as a function $\hat{q}(s, a, \mathbf{w})$, where $\mathbf{w}$ is a parameter we change to minimise the mean squared error between our estimate $\hat{q}$ and true value q.

We minimise:

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

We do this by adjusting $J(\mathbf{w})$ in the direction of negative gradient each episode, in order to find the global minimum.

Other Considerations:

- Investigate alternative policies to ϵ-greedy action selection
- **Increasing Model Complexity** - we could consider additions such as booking in advance, multiple night stays, incorporating competition/locational factors

References

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

Bitran, G. R. Mondschein, S. V. (1995), 'An application of yield management to the hotel industry considering multiple day stays', Operations Research 43(3), 427–443.