# Network Friendly Recommendations Project

Kalamarakis Theodoros: 2018030022
Toganidis Nikos: 2018030085
July 2023

# Understanding the Recommendation System Environment

- **States :**
  There are K states in total, where state *i* represents the user watching video *i.*

# Understanding the Recommendation System Environment

- **States :**

  There are K states in total, where state *i* represents the user watching video *i.*

- **Actions :**

  Each actions in this environment is a tuple of two distinct states (videos), excluding the previously watched state.

  Thus, the size of action set should be $\binom{K}{2} = K^2 - K = O(K^2)$

# Understanding the Recommendation System Environment

- **States :**
  There are K states in total, where state *i* represents the user watching video *i.*
- **Actions :**
  Each actions in this environment is a tuple of two distinct states (videos), excluding the previously watched state.
  Thus, the size of action set should be $\binom{K}{2} = K^2 - K = O(K^2)$
- **Cost and Reward :**
  If a video is cached, its cost is 0. If it is not cached, the cost is 1. → $Reward_i = 1 - Cost_i$

# Understanding the Recommendation System Environment

- **States :**

  There are K states in total, where state *i* represents the user watching video *i.*
- **Actions :**

  Each actions in this environment is a tuple of two distinct states (videos), excluding the previously watched state.

  Thus, the size of action set should be $\binom{K}{2} = K^2 - K = O(K^2)$
- **Cost and Reward :**

  If a video is cached, its cost is 0. If it is not cached, the cost is 1. → $Reward_i = 1 - Cost_i$
- **Transition probability :**

  The transition probability $P(i \to j)$:

  If all recommendations are relevant

  If video is present in the recommendation batch then

  $P(i \to j) = a/N + (1 - a)/K$

  If video is NOT present in the recommendation batch then

  $P(i \to j) = (1 - a)/K$

  If at least one video in the recommendation batch is irrelevant

  $P(i \to j) = 1/K$

# Understanding the Recommendation System Environment

- **States :**
  There are K states in total, where state *i* represents the user watching video *i.*
- **Actions :**
  Each actions in this environment is a tuple of two distinct states (videos), excluding the previously watched state.
  Thus, the size of action set should be $\binom{K}{2} = K^2 - K = O(K^2)$
- **Cost and Reward :**
  If a video is cached, its cost is 0. If it is not cached, the cost is 1. $\rightarrow Reward_i = 1 - Cost_i$
- **Transition probability :**
  The transition probability $P(i \rightarrow j)$:
  If all recommendations are relevant
  If video is present in the recommendation batch then
  $P(i \rightarrow j) = a/N + (1-a)/K$
  If video is NOT present in the recommendation batch then
  $P(i \rightarrow j) = (1-a)/K$
  If at least one video in the recommendation batch is irrelevant
  $P(i \rightarrow j) = 1/K$
- **Key Parameters:**

$$\gamma = 1 - q, \quad (\epsilon greedy)\ \epsilon = \frac{1}{t^{\frac{1}{3}}}\left(\#actions \cdot \log(t)\right)^{\frac{1}{3}},\ a = 0.01\ \ (learning\ ratio)$$

# Optimal Policy Verification through Policy Iteration

Toy example:

$$U = \begin{pmatrix} 0 & 0.8 & 0.6 & 0.3 \\ 0.8 & 0 & 0.7 & 0.2 \\ 0.3 & 0.1 & 0 & 0.2 \\ 0.6 & 0.4 & 0.2 & 0 \end{pmatrix}$$

$$Cost = [\ 1, 0, 1, 0\ ]$$

$$a = 0.8, \qquad q = 0.2, \quad u_{min} \quad = 0.2$$

# Optimal Policy Verification through Policy Iteration

Toy example:

$$U = \begin{pmatrix} 0 & 0.8 & 0.6 & 0.3 \\ 0.8 & 0 & 0.7 & 0.2 \\ 0.3 & 0.1 & 0 & 0.2 \\ 0.6 & 0.4 & 0.2 & 0 \end{pmatrix}$$

$$Cost = [\ 1, 0, 1, 0\ ]$$

$$a = 0.8, \qquad q = 0.2, \quad u_{min} \quad = 0.2$$

The optimal policy given by our algorithm is:

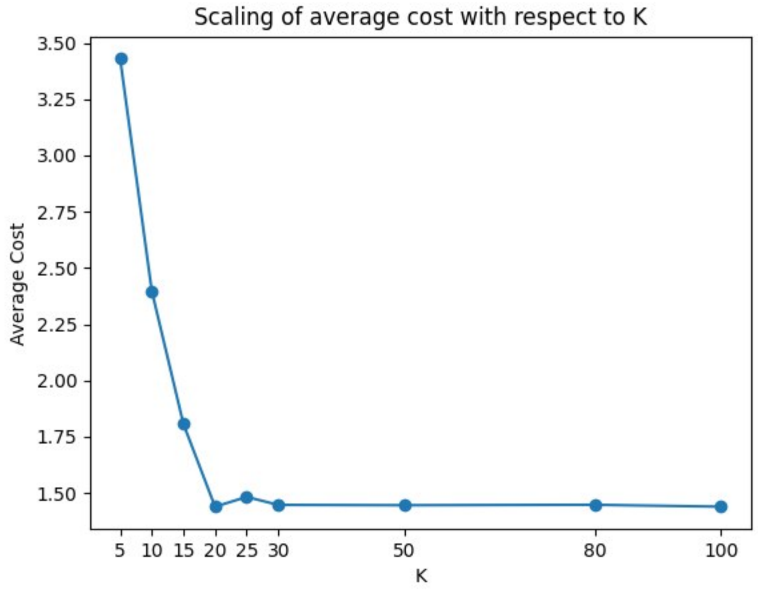
$$\pi[0] = (1, 3), \qquad \pi[1] = (0, 3), \qquad \pi[2] = (0, 3), \qquad \pi[3] = (0,1)$$

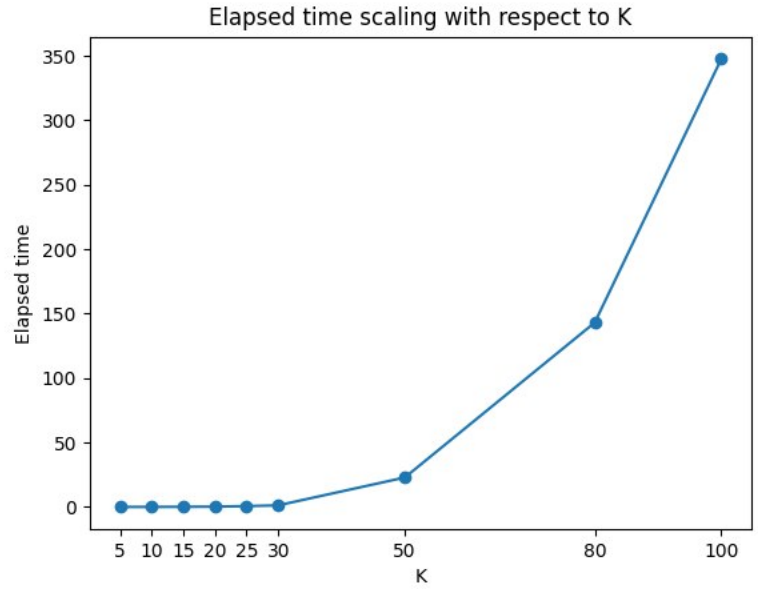
# Solution Complexity and Scaling

Analyzing the impact of increasing the size of the video catalog (K) on the algorithm's average cost and elapsed time

# Solution Complexity and Scaling

Analyzing the impact of increasing the size of the video catalog (K) on the algorithm's average cost and elapsed time



# Solution Complexity and Scaling

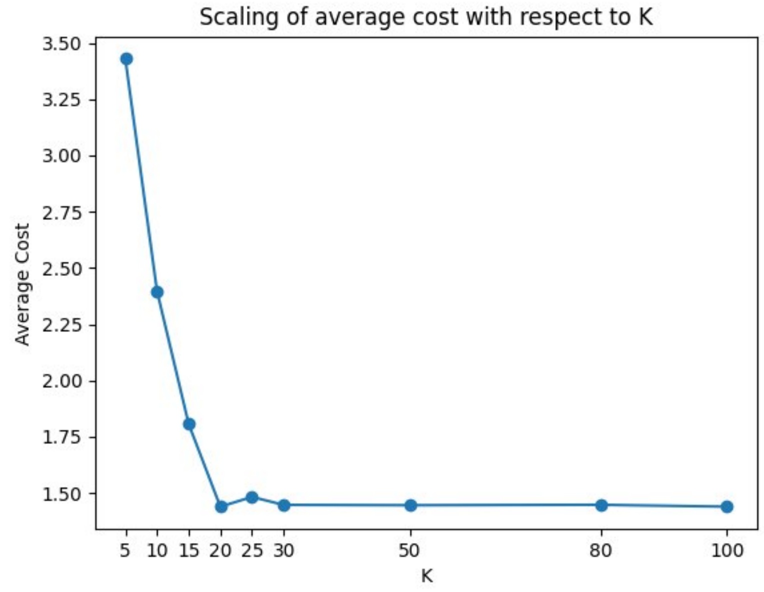
Analyzing the impact of increasing the size of the video catalog (K) on the algorithm's average cost and elapsed time

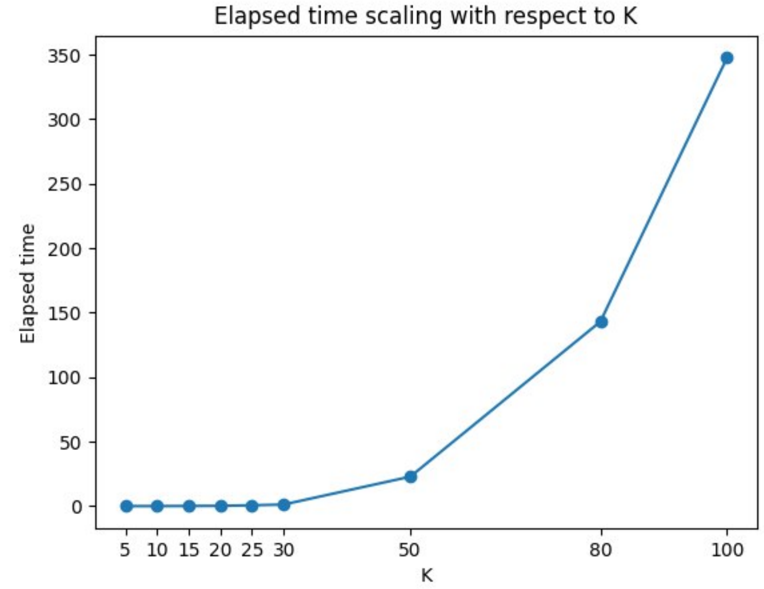
**Evidence of Complexity Scaling:**

- Average Cost: As *K* increases, the expected average cost decreases due to more cached items being available for recommendation, resulting in lower cost.

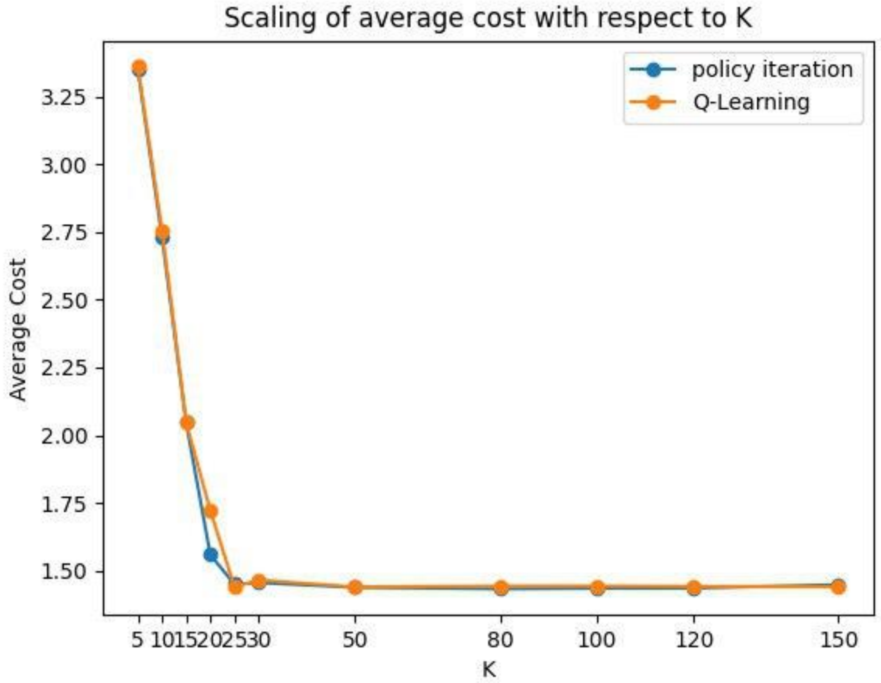
- Elapsed Time: With the size of the action set escalating at $O(K^2)$ , there is a proportional increase in elapsed time with respect to K.

**Key Takeaway:**
As the scenario becomes larger (i.e., the video catalog expands), Policy Iteration remains effective but requires more computational resources and time.



# Optimal Policy Verification through Q-Learning

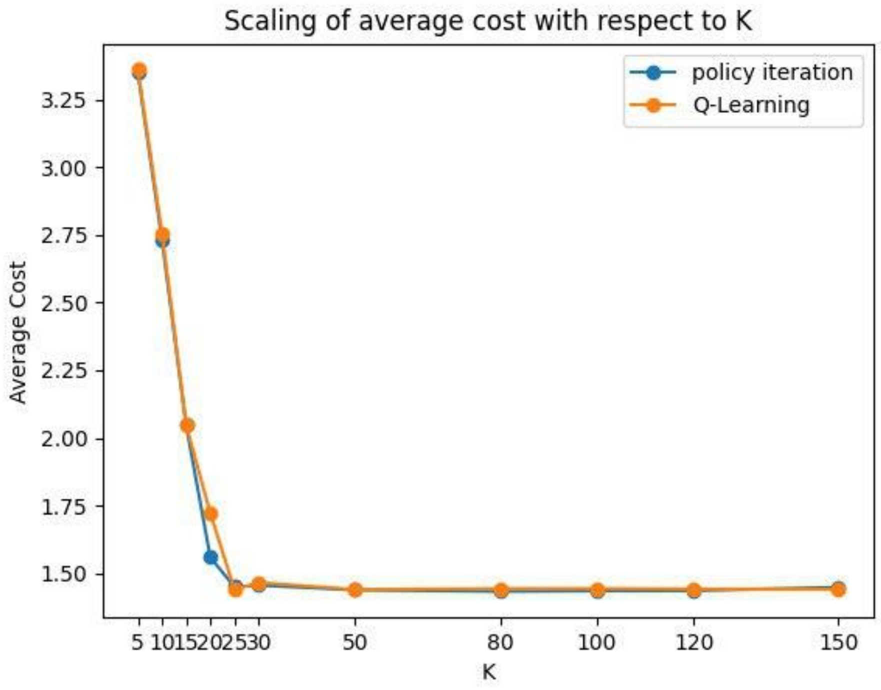



Comparison between Q-Learning and Policy Iteration with parameters $a = 0.8, q = 0.2, u_{min} = 0.2, C = 0.2K$

# Optimal Policy Verification through Q-Learning

**Evidence of Optimality:**

1. **Average Cost Comparison**: The Q-Learning algorithm line coincides with the Policy Iteration line, indicating that they both achieve a similar average cost, hence showing the optimal policy effectiveness of Q-Learning.




Comparison between Q-Learning and Policy Iteration with parameters $a = 0.8, q = 0.2, u_{min} = 0.2, C = 0.2K$

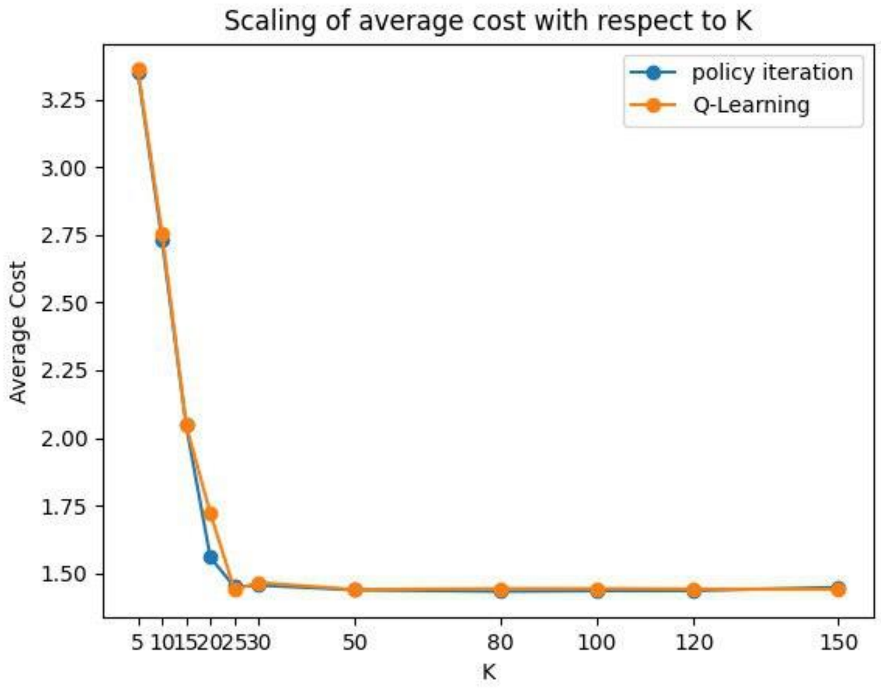
# Optimal Policy Verification through Q-Learning

**Evidence of Optimality:**

1. **Average Cost Comparison**: The Q-Learning algorithm line coincides with the Policy Iteration line, indicating that they both achieve a similar average cost, hence showing the optimal policy effectiveness of Q-Learning.

2. **Policy Comparison:** Both algorithms may not find the exact same policies due to multiple equivalent policies, but both converge to a policy that minimizes the cost, indicating Q-Learning's ability to find an optimal policy.

**Key Takeaway:**
Q-Learning effectively identifies the optimal policy and performs as well as Policy Iteration in minimizing the cost.

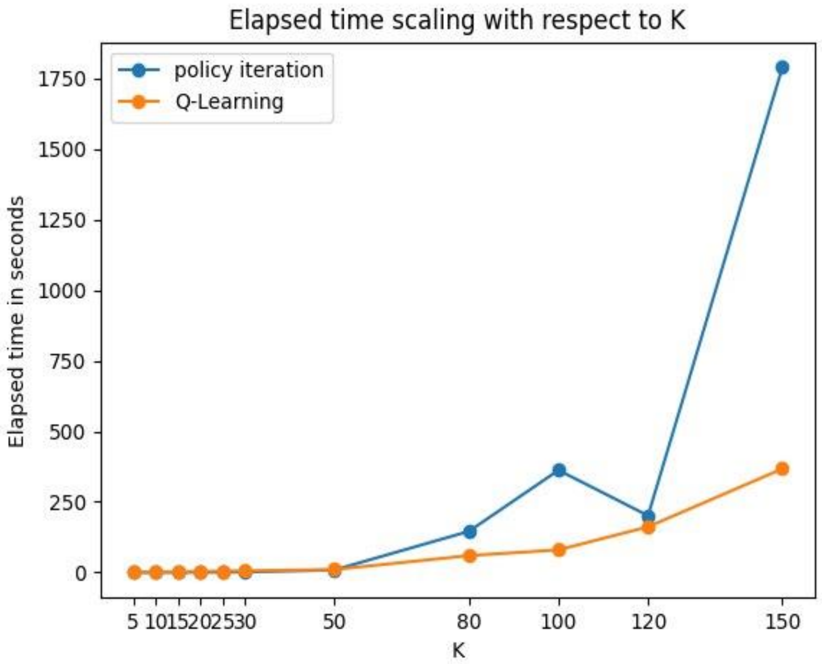



Comparison between Q-Learning and Policy Iteration with parameters $a = 0.8, q = 0.2, u_{min} = 0.2, C = 0.2K$

# Solution Complexity and Scalability in Q-Learning

**Elapsed Time**:
We have set Q-Learning to perform *2000K* iterations
Thus, elapsed time scales in $O(K)$, which is faster than the $O(K^2)$ scaling of Policy Iteration.
For instance, at K = 150, Q-Learning significantly outperforms Policy Iteration, which requires half an hour to converge.

**Key Takeaway:**
Q-Learning not only achieves an optimal policy but also scales better than Policy Iteration, demonstrating higher efficiency and scalability as the video catalog expands.




Comparison between Q-Learning and Policy Iteration with parameters
$a = 0.8, q = 0.2, u_{min} = 0.2, C = 0.2K$

# Interpretation of the results

1. Why both algorithms converge to an average cost of *1.44* as $K \to \infty$ ?
2. Is *1.44* the optimal average cost for $a = 0.8,\ q = 0.2,\ u_{min} = 0.2,\ C = 0.2K\ ,\ K\ \to \infty$ ?

# Interpretation of the results

1. Why both algorithms converge to an average cost of *1.44* as $K \to \infty$ ?
2. Is *1.44* the optimal average cost for $a = 0.8,\ q = 0.2,\ u_{min} = 0.2,\ C = 0.2K\ ,\ K\ \to \infty$ ?

We have derived a theoretical formula that calculates the average cost per session, assuming that we follow an optimal policy. We consider *a* and *q* to be variables and we set *C=0.2K* and $K \to \infty$

$$E[S] = 0.8 + 0.8\left(\frac{1}{q} - 1\right)(1 - a) \qquad (1)$$

if we set $a = 0.8$ and $\mathrm{q} = 0.2\ \ \to$ $\qquad E[S] = 0.8 + 0.8(5 - 1)(1 - 0.8) = 0.8 + 0.64 = 1.44$ !!!!!

# Interpretation of the results

1. Why both algorithms converge to an average cost of *1.44* as $K \to \infty$ ?
2. Is *1.44* the optimal average cost for $a = 0.8,\ q = 0.2,\ u_{min} = 0.2,\ C = 0.2K\ ,\ K\ \to \infty$ ?

We have derived a theoretical formula that calculates the average cost per session, assuming that we follow an optimal policy. We consider *a* and *q* to be variables and we set *C=0.2K* and $K \to \infty$

$$E[S] = 0.8 + 0.8\left(\frac{1}{q} - 1\right)(1 - a) \qquad (1)$$

if we set $a = 0.8$ and $\mathrm{q} = 0.2$ $\rightarrow$ $E[S] = 0.8 + 0.8(5 - 1)(1 - 0.8) = 0.8 + 0.64 = 1.44$ !!!!!

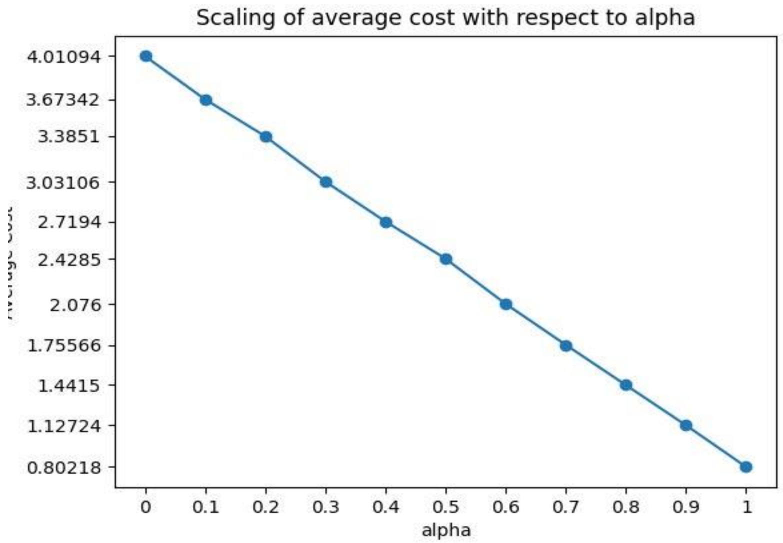
# Interpretation of the results

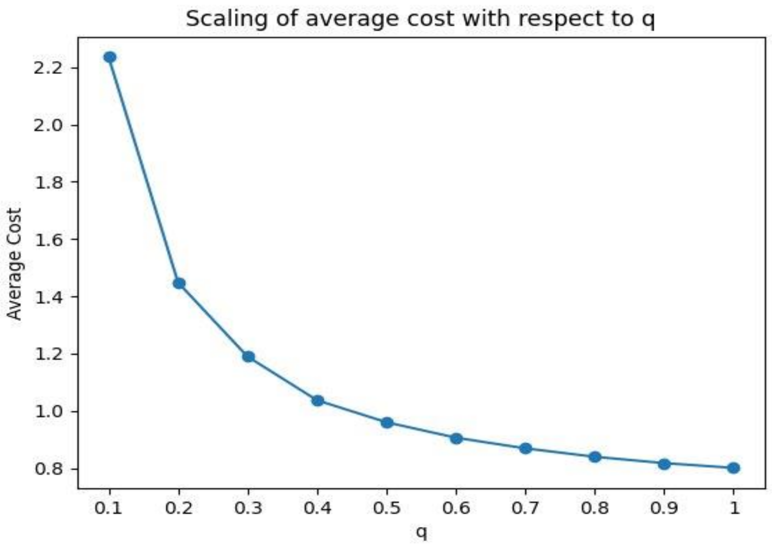
1. Why both algorithms converge to an average cost of *1.44* as $K \to \infty$ ?
2. Is *1.44* the optimal average cost for $a = 0.8,\ q = 0.2,\ u_{min} = 0.2,\ C = 0.2K\ ,\ K\ \to \infty$ ?

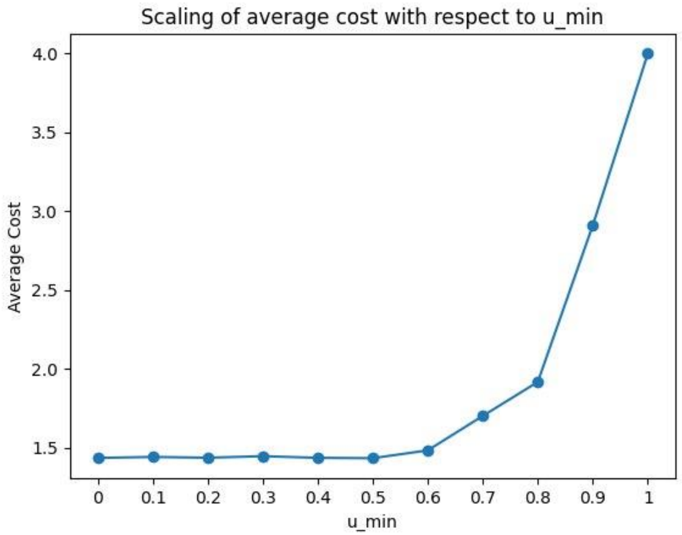
We have derived a theoretical formula that calculates the average cost per session, assuming that we follow an optimal policy. We consider *a* and *q* to be variables and we set *C=0.2K* and $K \to \infty$

$$E[S] = 0.8 + 0.8\left(\frac{1}{q} - 1\right)(1 - a) \qquad (1)$$

if we set $a = 0.8$ and q $= 0.2$ $\rightarrow$ $E[S] = 0.8 + 0.8(5 - 1)(1 - 0.8) = 0.8 + 0.64 = 1.44$ !!!!



# Proof of formula (1)

Average cost per video

$$E[X] = 0P(C) + 1P(U) = P(U)$$

# Proof of formula (1)

Average cost per video

$$E[X] = 0P(C) + 1P(U) = P(U)$$

$$P(U) = P(U|A)P(A) + P(U|\bar{A})P(\bar{A})$$

Here, A denotes the event of choosing a video from the recommendation batch, so $P(A) = a$

# Proof of formula (1)

Average cost per video

$$E[X] = 0P(C) + 1P(U) = P(U)$$

$$P(U) = P(U|A)P(A) + P(U|\bar{A})P(\bar{A})$$

Here, A denotes the event of choosing a video from the recommendation batch, so $P(A) = a$

For $K \to \infty$ then $P(U|A) = 0$ Hence

$$E[X] = P(U) = P(U|\bar{A})P(\bar{A}) = 0.8(1 - a)$$

# Proof of formula (1)

Average cost per video

$$E[X] = 0P(C) + 1P(U) = P(U)$$

$$P(U) = P(U|A)P(A) + P(U|\bar{A})P(\bar{A})$$

Here, A denotes the event of choosing a video from the recommendation batch, so $P(A) = a$

For $K \rightarrow \infty$ then $P(U|A) = 0$ Hence

$$E[X] = P(U) = P(U|\bar{A})P(\bar{A}) = 0.8(1 - a)$$

If *n* is the number of videos per session, then $P(n) = (1 - q)^{n-1}q \rightarrow E[n] = \frac{1}{q}$

# Proof of formula (1)

Average cost per video

$$E[X] = 0P(C) + 1P(U) = P(U)$$

$$P(U) = P(U|A)P(A) + P(U|\bar{A})P(\bar{A})$$

Here, A denotes the event of choosing a video from the recommendation batch, so $P(A) = a$

For $K \to \infty$ then $P(U|A) = 0$ Hence

$$E[X] = P(U) = P(U|\bar{A})P(\bar{A}) = 0.8(1-a)$$

If *n* is the number of videos per session, then $P(n) = (1-q)^{n-1}q \quad \to E[n] = \frac{1}{q}$

The average cost per session (E[S]) is

$$E[S] = cost\ of\ the\ initial\ state + cost\ of\ the\ remaining\ of\ the\ session \Leftrightarrow$$

$$E[S] = 0.8 + (E[n]-1)E[X] = 0.8 + \left(\frac{1}{q} - 1\right)0.8(1-a)$$

# Proof of formula (1)

Average cost per video

$$E[X] = 0P(C) + 1P(U) = P(U)$$

$$P(U) = P(U|A)P(A) + P(U|\bar{A})P(\bar{A})$$

Here, A denotes the event of choosing a video from the recommendation batch, so $P(A) = a$

For $K \to \infty$ then $P(U|A) = 0$ Hence

$$E[X] = P(U) = P(U|\bar{A})P(\bar{A}) = 0.8(1-a)$$

If *n* is the number of videos per session, then $P(n) = (1-q)^{n-1}q \;\; \to E[n] = \frac{1}{q}$

The average cost per session (E[S]) is

$$E[S] = cost\ of\ the\ initial\ state + cost\ of\ the\ remaining\ of\ the\ session \Leftrightarrow$$

$$E[S] = 0.8 + (E[n]-1)E[X] = 0.8 + \left(\frac{1}{q} - 1\right)0.8(1-a)$$

General form $\left(E[S] = \left(1 - \frac{C}{\mathrm{K}}\right) + \left(1 - \frac{C}{\mathrm{K}}\right)\left(\frac{1}{q} - 1\right)(1-a)\right)$