Reinforcement Learning

August 2, 2019

MMCi Applied Data Science
Block 5, Lecture 1

Matthew Engelhard


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last – a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO


NETFLIX
ALPHAGO

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') \max_{a' \in A} Q^{\pi}(s',a')$$

$$Q(s_t, a_t; \theta^{new}) = Q(s_t, a_t; \theta^{old} + \alpha \cdot [r_t + \gamma \cdot \max_{a'} Q\left(s'_{t+1}, a'; \theta^{old}\right) - Q\left(s_t, a_t; \theta^{old}\right)] \nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}})$$

$$\Delta\theta = \alpha \cdot [r_t + \gamma \cdot \max_{a'} Q\left(s'_{t+1}, a'; \theta^{old}\right) - Q\left(s_t, a_t; \theta^{old}\right)] \nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}}$$

$$Q(s_t, a_t; \theta^{old} + \Delta\theta) \approx Q\left(s_t, a_t; \theta^{old}\right) + \alpha \cdot \left[r_t + \gamma \cdot \max_{a'} Q\left(s'_{t+1}, a'; \theta^{old}\right) - Q\left(s_t, a_t; \theta^{old}\right)\right] ||\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}}||_2^2$$

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') \max_{a' \in A} Q^{\pi}(s',a')$$

MATHEMATICS!!

$$Q(s_t, a_t; \theta^{new}) = Q(s_t, a_t; \theta^{old} + \alpha \cdot [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old}) - Q(s_t, a_t; \theta^{old})]\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}})$$

$$\Delta\theta = \alpha \cdot [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old}) - Q(s_t, a_t; \theta^{old})]\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}}$$

$$Q(s_t, a_t; \theta^{old} + \Delta\theta) \approx Q(s_t, a_t; \theta^{old}) + \alpha \cdot \left[r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old}) - Q(s_t, a_t; \theta^{old})\right] ||\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}}||_2^2$$

Sequential Decision-Making

An agent

takes actions

based on the state
of a system

to maximize reward

Sequential Decision-Making

An agent

takes actions

based on the state of a system

to maximize reward

Sequential Decision-Making

An agent

takes actions

based on the state of a system

to maximize reward

https://www.techradar.com/news/fords-robot-postman-will-deliver-packages-to-your-front-door

Sequential Medical Decision-Making

An agent

takes actions

based on the state
of a system

to maximize reward

- Consider a clinician seeking to treat patients effectively while minimizing costs
- The health of each patient is observed as a set of clinical variables
- Assume the clinician has a fixed set of actions he or she can perform
- Each action may (or may not) change the health state of the patient.
- The clinician seeks a policy that achieves the best outcomes at the lowest average cost
- Could apply to particular types of patients (e.g., diabetics) or in particular settings within a health system (e.g., ICU)

Sequential Medical Decision-Making

An agent

takes actions

based on the state
of a system

to maximize reward

- Consider a clinician seeking to treat patients effectively while minimizing costs
- The health of each patient is observed as a set of clinical variables
- Assume the clinician has a fixed set of actions he or she can perform
- Each action may (or may not) change the health state of the patient.
- The clinician seeks a policy that achieves the best outcomes at the lowest average cost
- Could apply to particular types of patients (e.g., diabetics) or in particular settings within a health system (e.g., ICU)

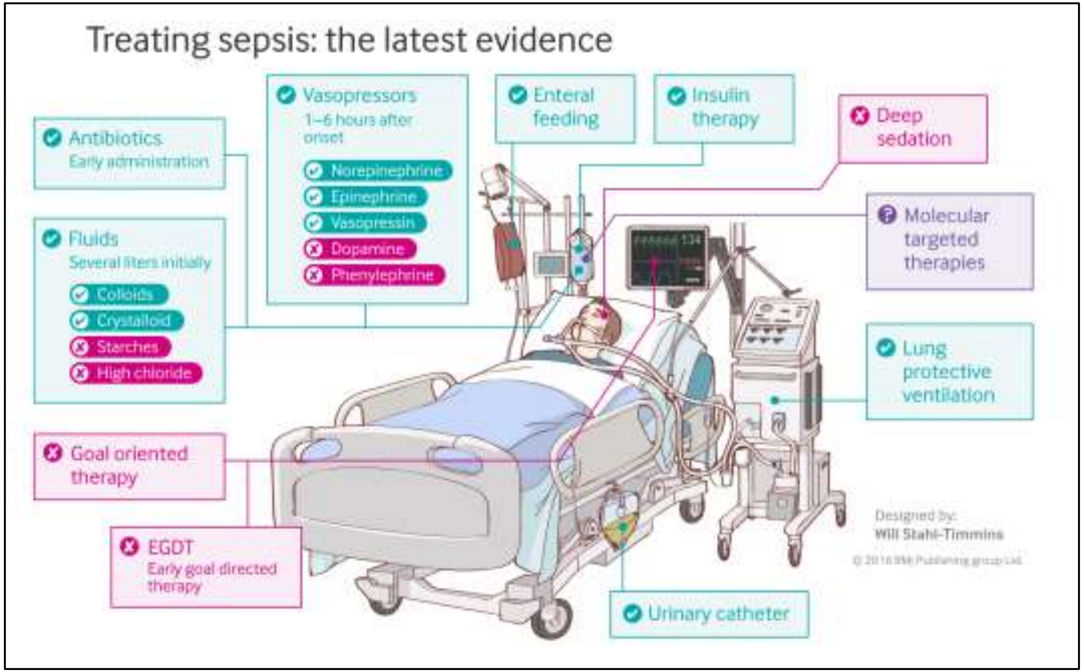
Sequential Medical Decision-Making

An agent

takes actions

based on the state of a system

to maximize reward



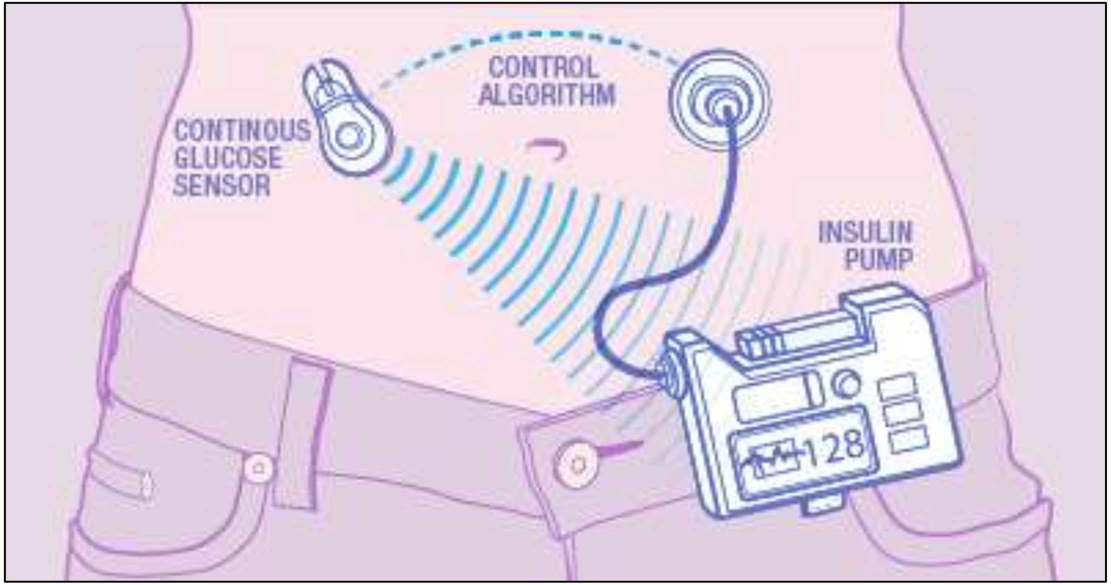
Sequential Medical Decision-Making

An agent

takes actions

based on the state
of a system

to maximize reward



Sequential Medical Decision-Making

- There is randomness in how a patient in state s will respond to a given action, because the information in s yields an incomplete view of patient health
- Let $P(s, a, s')$ represent the *probability* that when a patient is in state s and the MD takes action a, the patient state will change to s'

An agent

takes actions → A set $\{a^1, \ldots, a^m\}$ of possible actions

based on the state of a system → A vector $\boldsymbol{s}$ of measurements

to maximize reward → A function $r(s, a, s')$ that quantifies the impact of transitioning from s to s' as well as the cost associated with action a

- The clinician interacts with the patient through a series of actions, patient state changes, and rewards/costs

$$\ldots\ s_{t-1}\quad a_{t-1}\quad r_{t-1}\quad s_t\quad a_t\quad r_t\quad s_{t+1}\ \ldots$$

- Goal: Develop a **policy** that specifies the optimal action a to take when the patient is in state s. This policy might define the *standard of care*

- The optimal policy will maximize the average reward over time, with reward accounting for patient outcome and costs

- The policy should be non-myopic, in that it thinks ahead, to the long-run impact of actions

- The policy will typically weight impacts in the near-term more highly than what happens in the long run

The Challenge

- Unfortunately, we typically do not know $P(s, a, s')$. In other words, we don't know ahead of time how our actions will affect the state
- So, **how can we learn a policy**?
- We can just experience/try things, keep a record of outcomes, and adapt and adjust
- <u>Reinforce</u> actions for particular patient states that are rewarding
- <u>Discourage</u> actions that are expensive and yield poor outcomes
- In many ways this is how medicine works (over a long period of time)

- Reinforcement learning is the formalization of this challenge
- Addresses sequential decision making in an uncertain (stochastic) world

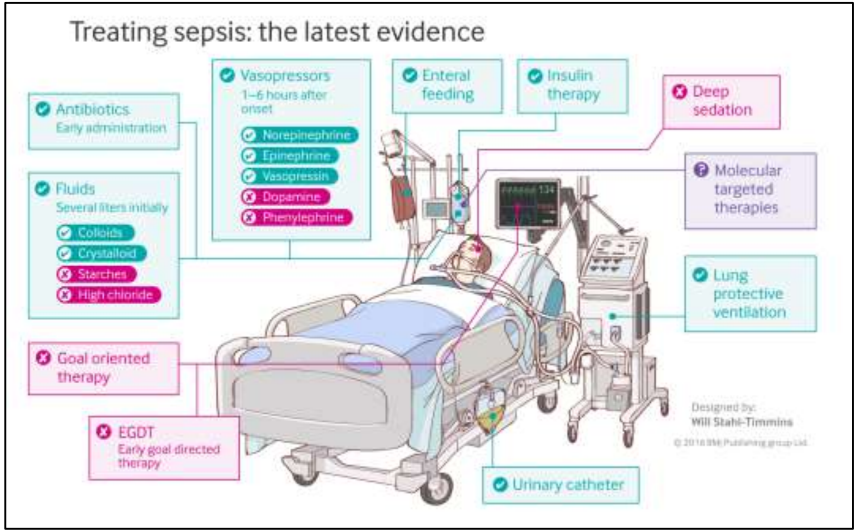
Medicine/Health

Monitoring/Maintenance of Factory

Investing

Illustrative Example: Sepsis Management in the ICU

- The patient **state** s is defined by physiologic measures available at the bedside; as well as patient characteristics
  - Discretize the continuous range of the state values into n distinct bins



- The **action** a specifies the **amount of fluid** and **amount of vasopressor** given to the patient
  - Discretize the possible actions into m distinct bins

- $r(s, a, s')$ is specified by the clinician and/or health system to reward (punish) patient states that are desirable (undesirable)

Illustrative Example: Sepsis Management in the ICU

- The patient **state** s is defined by physiologic measures available at the bedside; as well as patient characteristics
  - Discretize the continuous range of the state values into n distinct bins
- The **action** a specifies the **amount of fluid** and **amount of vasopressor** given to the patient
  - Discretize the possible actions into m distinct bins
- $r(s, a, s')$ is specified by the clinician and/or health system to reward (punish) patient states that are desirable (undesirable)

The Q function $Q(s, a)$ is an $n \times m$ matrix that specifies the value of taking action a when in state s

	a^1	a^2	…	a^m
s^1				
s^2		$Q(s^2, a^2)$		
s^3				
s^4				
s^5				
…				
s^n				

Illustrative Example: Sepsis Management in the ICU

- ❑ Set initial values of $Q(s, a)$ based on prior medical knowledge, at random, or to a default value

The Q function $Q(s, a)$ is an $n \times m$ matrix that specifies the value of taking action a when in state s

	a^1	a^2	…	a^m
s^1	0	0		0
s^2	0	0		0
s^3	0	0		0
s^4	0	0		0
s^5	0	0		0
…				0
s^n	0	0		0

Illustrative Example: Sepsis Management in the ICU

- Set initial values of $Q(s,a)$ based on prior medical knowledge, at random, or to a default value

- After initializing $Q(s,a)$, take an action a when patient is in particular state s, and then observe the new state s'

$$(s,a) \to s', \qquad r(s,a,s')$$

- Assume that the prior/old value for the Q function when taking action a in state s is $Q^{old}(s,a)$

- We may consider the update rule:

$$Q^{new}(s,a) \leftarrow (1-\alpha) \cdot Q^{old}(s,a) + \alpha \cdot r(s,a,s')$$

where $\alpha \in (0,1]$

The Q function $Q(s,a)$ is an $n \times m$ matrix that specifies the value of taking action a when in state s

	a^1	a^2	…	a^m
s^1	0	0		0
s^2	0	0		0
s^3	0	0		0
s^4	0	0		0
s^5	0	0		0
…				0
s^n	0	0		0

Illustrative Example: Sepsis Management in the ICU

- Set initial values of $Q(s,a)$ based on prior medical knowledge, at random, or to a default value

- After initializing $Q(s,a)$, take an action a when patient is in particular state s, and then observe the new state s'

$$(s,a) \to s', \qquad r(s,a,s')$$

- Assume that the prior/old value for the Q function when taking action a in state s is $Q^{old}(s,a)$

- We may consider the update rule:

$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot r(s,a,s')$$

where $\alpha \in (0,1]$

The Q function $Q(s,a)$ is an $n \times m$ matrix that specifies the value of taking action a when in state s

	a^1	a^2	…	a^m
s^1	0	0		0
s^2	0	0		0
s^3	0	0		0
s^4	0	0		0
s^5	0	0		0
…				0
s^n	0	0		0

Illustrative Example: Sepsis Management in the ICU

- Set initial values of $Q(s,a)$ based on prior medical knowledge, at random, or to a default value

- After initializing $Q(s,a)$, take an action a when patient is in particular state s, and then observe the new state s'

$$(s,a) \rightarrow s', \qquad r(s,a,s')$$

- Assume that the prior/old value for the Q function when taking action a in state s is $Q^{old}(s,a)$

- We may consider the update rule:

$$Q^{new}(s,a) \leftarrow (1-\alpha) \cdot Q^{old}(s,a) + \alpha \cdot r(s,a,s')$$

where $\alpha \in (0,1]$

The Q function $Q(s,a)$ is an $n \times m$ matrix that specifies the value of taking action a when in state s

	a^1	a^2	…	a^m
s^1	0	0		0
s^2	0	0		0
s^3	0	$\alpha \cdot r(s^3, a^2, s')$		0
s^4	0	0		0
s^5	0	0		0
…				0
s^n	0	0		0

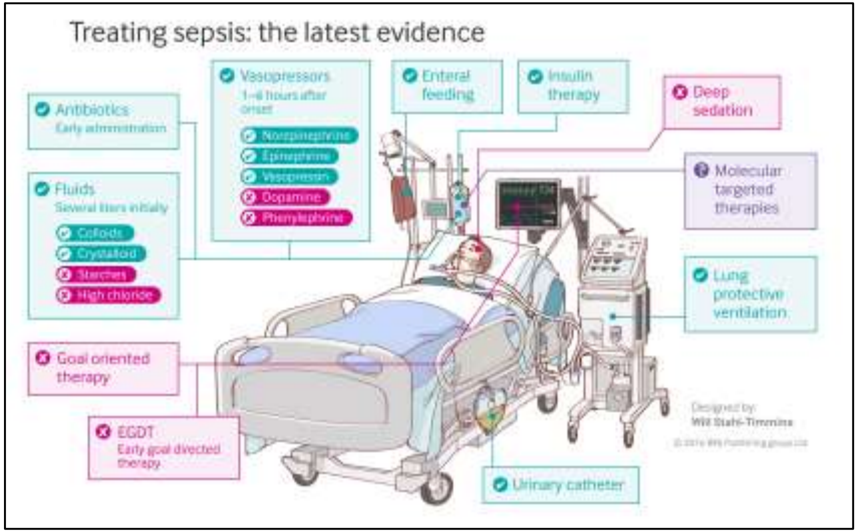
Simple, Intuitive Solution

$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot\ r(s,a,s'),\ \ \text{with } \alpha \in (0,1]$$

- α is called the **learning rate**, and controls the relative balance between our old estimate $Q^{old}(s,a)$ and new information provided by $r(s,a,s')$

- If α is large, we quickly replace our old estimate with new information

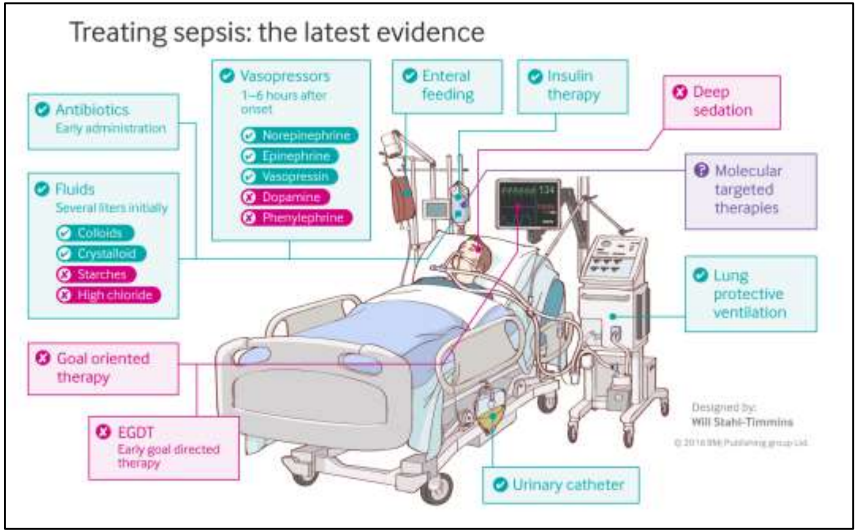
- However, due to possible randomness, we typically want to be conservative in updating $Q^{old}(s,a)$



Problem: We're only looking one step ahead!

$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot \ r(s,a,s'), \ \text{ with } \alpha \in (0,1]$$

- ❑ This simple solution only accounts for the *immediate* reward $r(s,a,s')$
- ❑ Doesn't account for what may happen subsequently once the patient state changes to s'
- ❑ What if our decisions look good in the short-term, but ultimately cause the patient's condition to deteriorate?
- ❑ This rule leads to a *myopic* policy



Let's augment our reward to consider what happens next

$$Q^{new}(s,a) \leftarrow (1-\alpha) \cdot Q^{old}(s,a) + \alpha \cdot \ r(s,a,s'), \quad \text{with } \alpha \in (0,1]$$

- Question: How much reward can we expect to earn going forward now that we're in state s'?
- Answer: $\max_{a'} Q^{old}(s',a')$
- Our effective reward is how much we earn now, plus how much we expect to earn going forward if we choose the best actions available to us.
- So, let's consider the following modification to our update rule:

$$Q^{new}(s,a) \leftarrow (1-\alpha) \cdot Q^{old}(s,a) + \alpha \cdot [r(s,a,s') + \gamma \cdot \max_{a'} Q^{old}(s',a')]$$

- The parameter γ controls how much we value immediate reward vs expected future reward

Let's augment our reward to consider what happens next

$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot \ r(s,a,s'), \quad \text{with } \alpha \in (0,1]$$

- Question: How much reward can we expect to earn going forward now that we're in state s'?
- Answer: $\max_{a'} Q^{old}(s',a')$
- Our effective reward is how much we earn now, plus how much we expect to earn going forward if we choose the best actions available to us.
- So, let's consider the following modification to our update rule:

$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot [r(s,a,s') + \gamma \cdot \max_{a'} Q^{old}(s',a')]$$

- The parameter γ is called the **discount factor**, and controls how much we value immediate reward versus expected future reward

Non-Myopic Update Rule

$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot [\underbrace{r(s,a,s')}_{\text{Immediate Reward}} + \underbrace{\gamma}_{\text{Discount Factor Between 0 and 1}} \cdot \underbrace{\max_{a'} Q^{old}(s',a')}_{\text{Expected Future Rewards Based on Optimal Policy}}]$$

- We have derived an algorithm for a clinician (or system) to learn based on experience
- This is **Q Learning**, a widely employed method for reinforcement learning
- Based on the learned matrix $Q(s,a)$, we typically employ the following policy:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Q Learning

Learning rate, $\alpha \in (0,1]$

Immediate Reward

Discount Factor Between 0 and 1

$$\underbrace{Q^{new}(s,a)}_{\text{Updated Estimate of Value of Action } a \text{ when in state } s} \leftarrow (1-\alpha)\cdot \underbrace{Q^{old}(s,a)}_{\text{Previous Estimate of Value of Action } a \text{ when in state } s} + \alpha \cdot [\overbrace{r(s,a,s')}^{\text{Immediate Reward}} + \gamma \cdot \underbrace{\max_{a'} Q^{old}(s',a')}_{\text{Expected Future Rewards Based on Optimal Policy}}]$$

Updated Estimate of Value of Action a when in state s

Previous Estimate of Value of Action a when in state s

Expected Future Rewards Based on Optimal Policy

- Note: while we're learning, we won't always follow the action defined by $\underset{a}{\operatorname{argmax}} Q^{old}(s,a)$
- If we did, we would never learn more about alternative actions
- Most of the time, we'd like to advantage of actions we know are rewarding. But sometimes, we should explore alternative actions to see whether they might be better. This tradeoff is known as *exploration versus exploitation.*

Q Learning

Learning rate, $\alpha \in (0,1]$

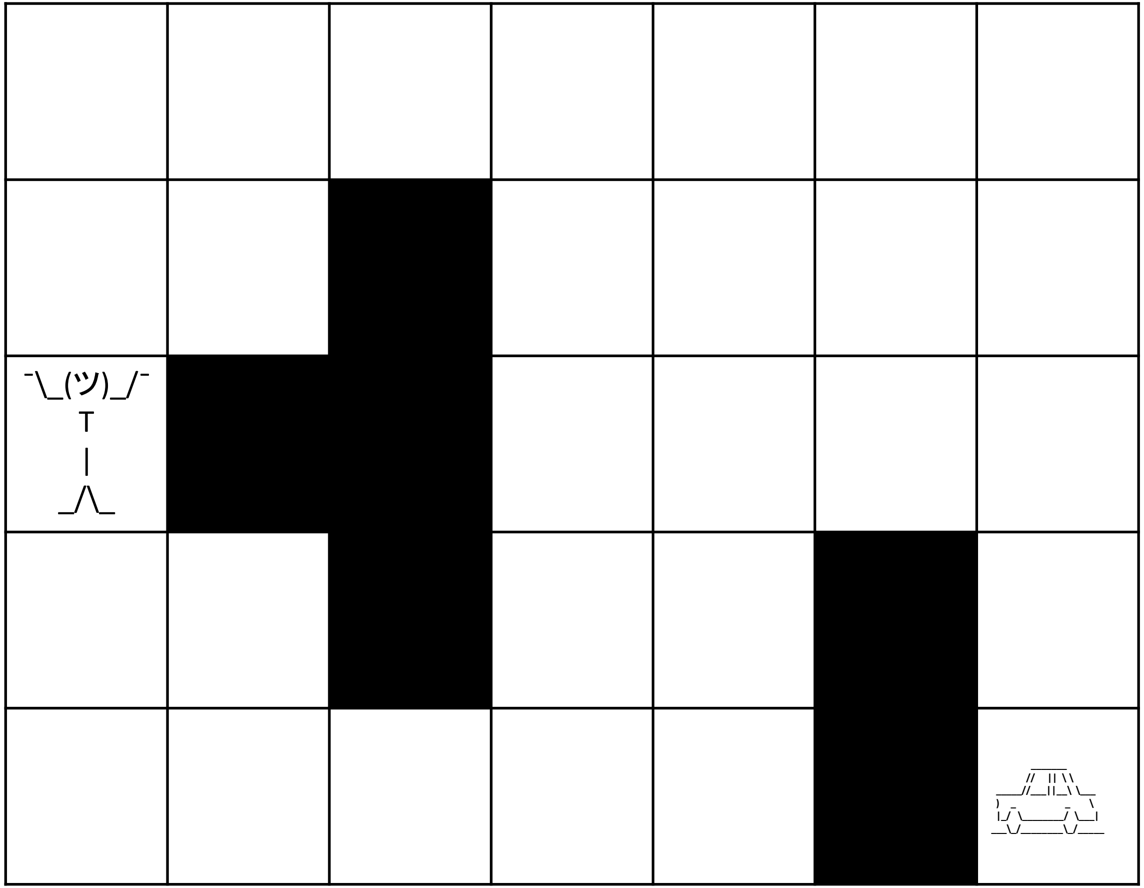
Immediate Reward

Discount Factor Between 0 and 1

$$\underbrace{Q^{new}(s,a)}_{\text{Updated Estimate of Value of Action } a \text{ when in state } s} \leftarrow (1-\alpha)\cdot \underbrace{Q^{old}(s,a)}_{\text{Previous Estimate of Value of Action } a \text{ when in state } s} + \alpha \cdot [r(s,a,s') + \gamma \cdot \underbrace{\max_{a'} Q^{old}(s',a')}_{\text{Expected Future Rewards Based on Optimal Policy}}]$$

The canonical RL example…

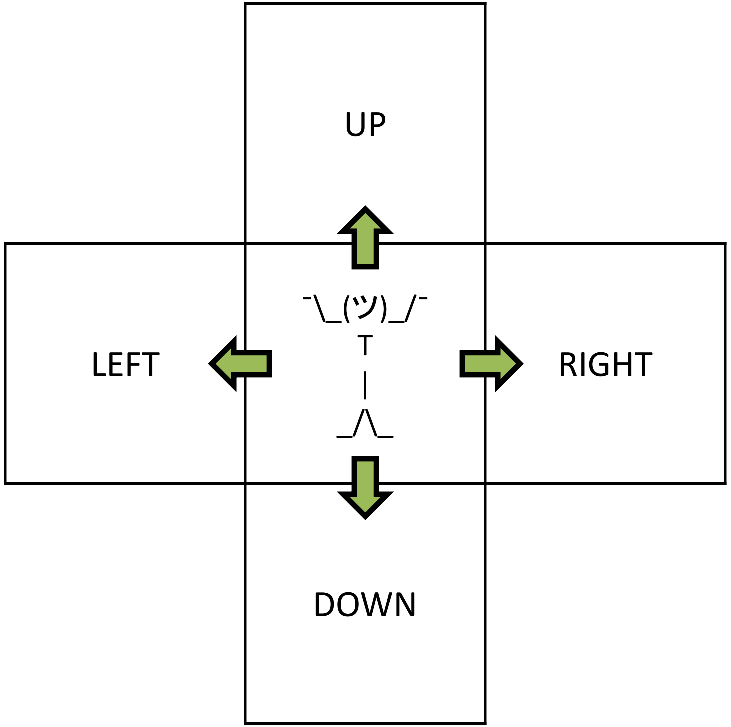
GRIDWORLD



Gridworld: **States**

The states are the different squares of gridworld.

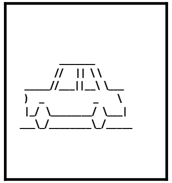
Gridworld: **Actions**



The actions are:

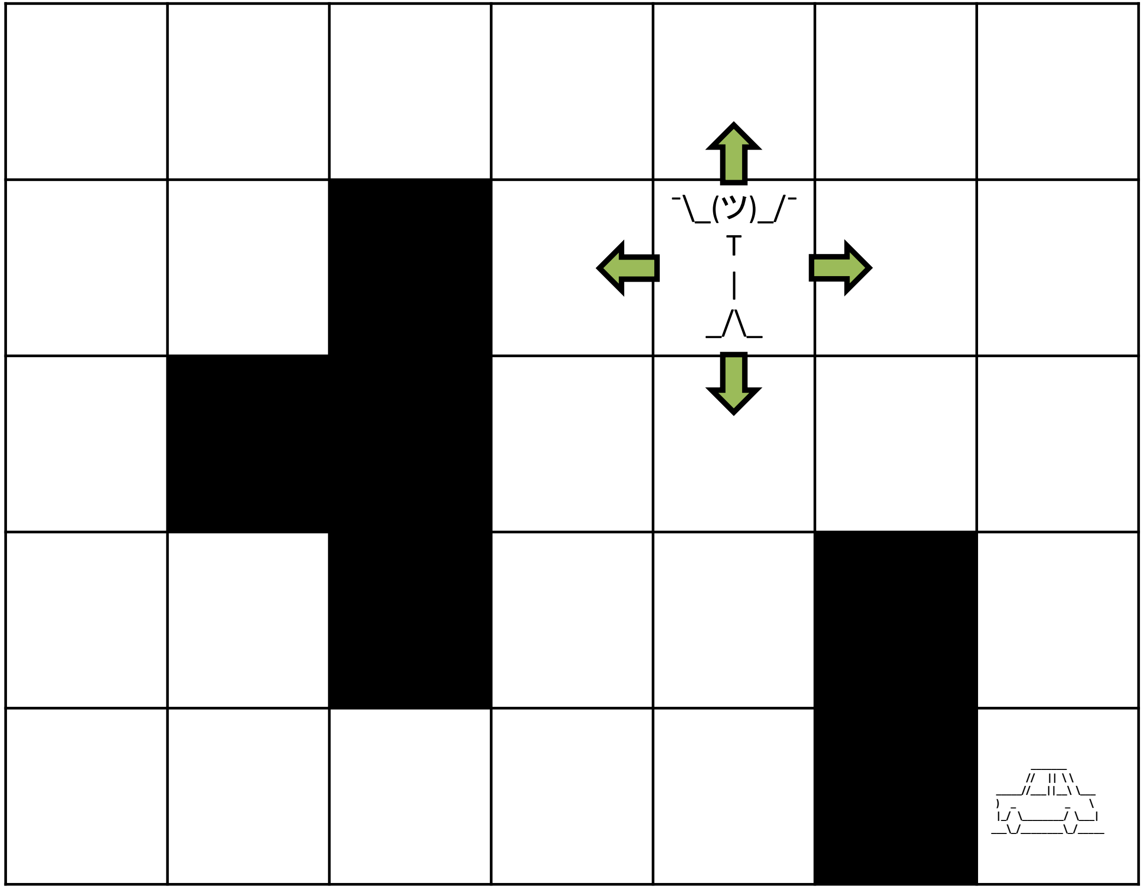
- Move left
- Move right
- Move up
- Move down

Gridworld: **Rewards**

- 1 for reaching 
- 0 otherwise

The goal is to reach the car. So our reward is:

$$r(s, a, s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$



GOAL:

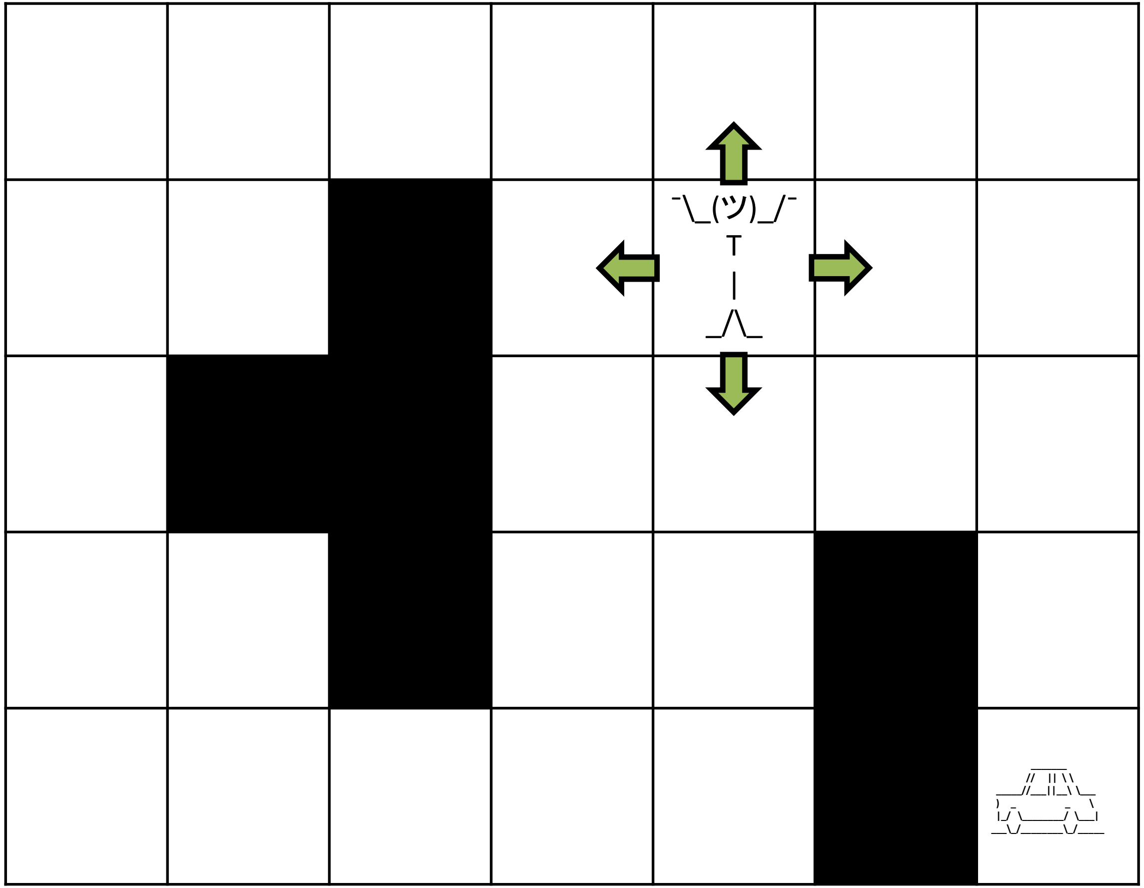
Learn a <u>policy</u> $\pi : S \to A$

that **maximizes expected reward over time**

HOW?

Learn the value $Q(s, a)$ of action *a* when in state *s*

$$Q(\text{current square}, down)$$



GOAL:

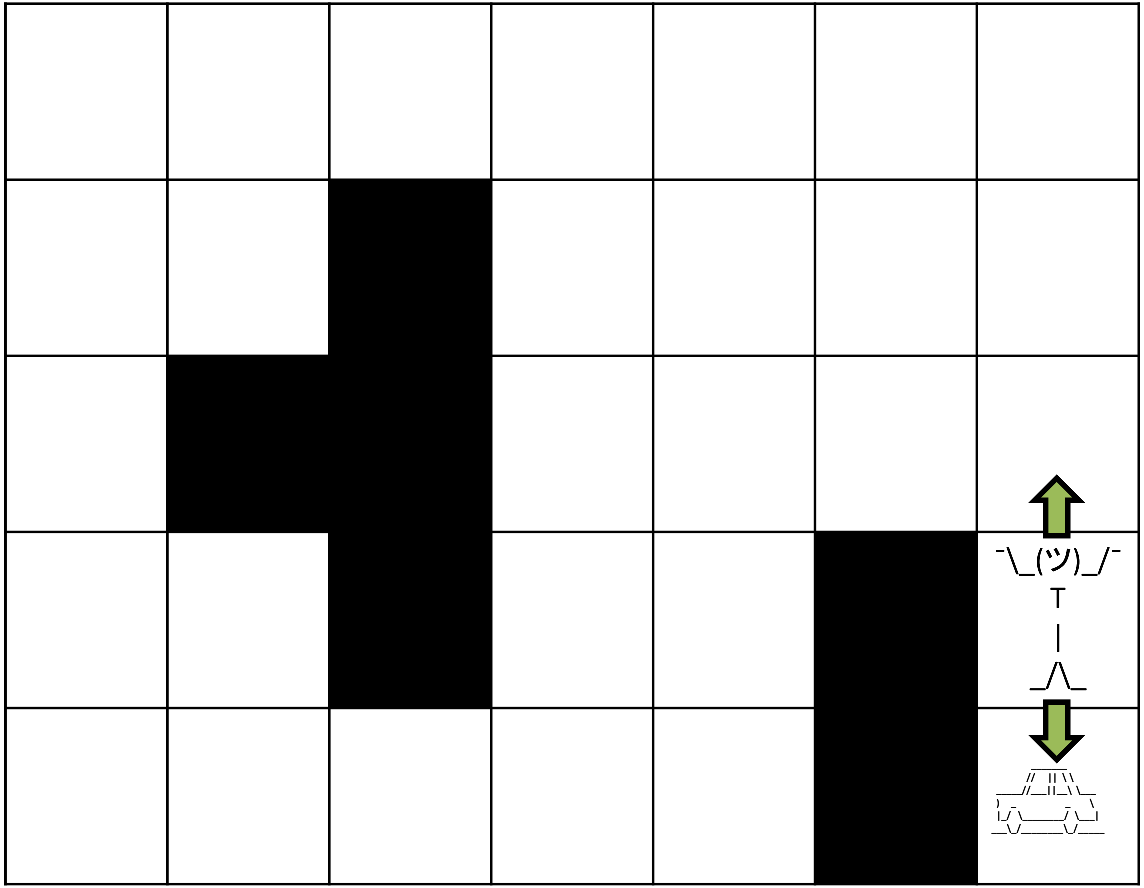
Learn a <u>policy</u> $\pi: S \rightarrow A$

that **gets us to the car as quickly as possible**

HOW?

Learn the value $Q(s, a)$ of action *a* when in state *s*

$$Q(\text{current square}, down)$$



GOAL:

Learn a <u>policy</u> $\pi: S \rightarrow A$

that **gets us to the car as quickly as possible**

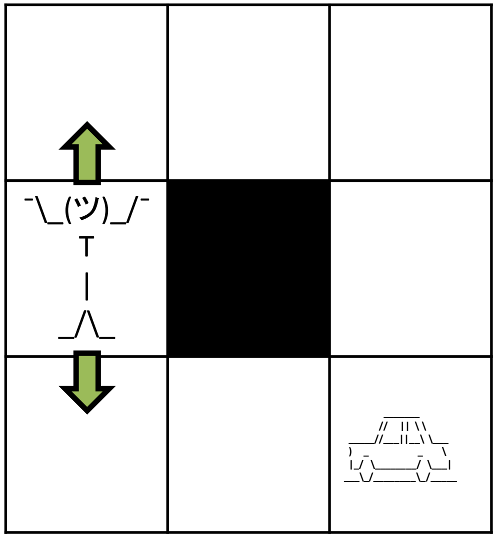
HOW?

Learn the value $Q(s, a)$ of action *a* when in state *s*

$$Q(\text{current square}, down)$$

Gridworld Example

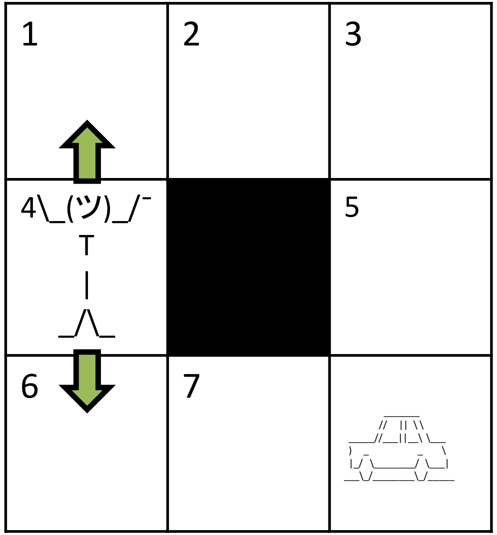
$$Q^{new}(s,a) \leftarrow (1-\alpha)\cdot Q^{old}(s,a) + \alpha \cdot [\, r(s,a,s') + \gamma \cdot \max_{a'} Q^{old}(s',a')\,]$$



Gridworld Example

$$Q^{new}(s,a) \leftarrow (1-1)\cdot Q^{old}(s,a) + 1\cdot[\, r(s,a,s') + .8\cdot \max_{a'} Q^{old}(s',a')\,]$$

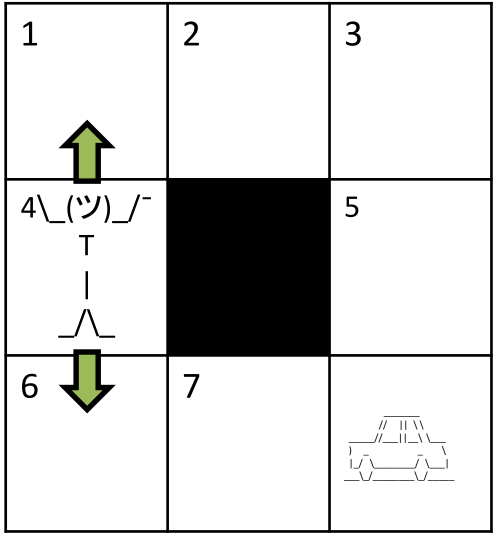
$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$



Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$



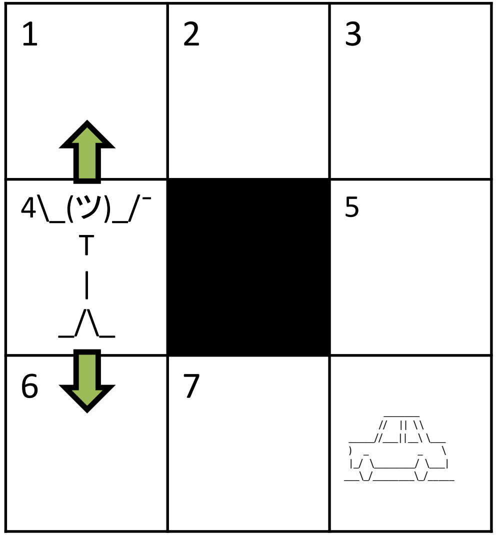
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1				
Square 2				
Square 3				
Square 4				
Square 5				
Square 6				
Square 7				

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

ITERATION 0



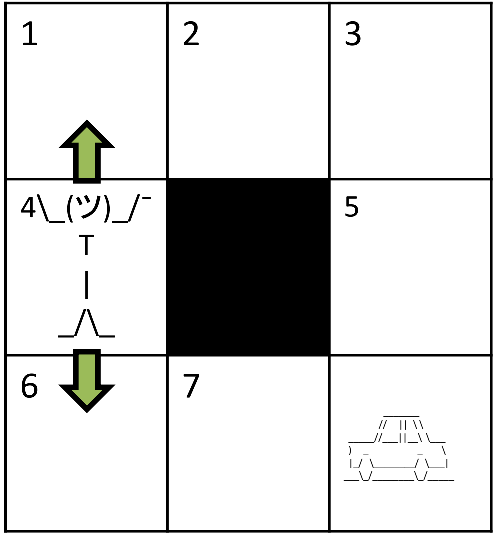
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		0		0
Square 2	0	0		
Square 3	0			0
Square 4			0	0
Square 5			0	0
Square 6		0	0	
Square 7	0	0		

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

ITERATION 1



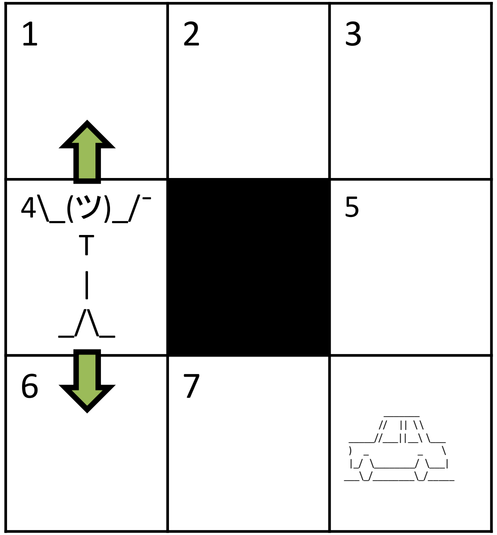
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		0		0
Square 2	0	0		
Square 3	0			0
Square 4			0	0
Square 5			0	1
Square 6		0	0	
Square 7	0	1		

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

ITERATION 2



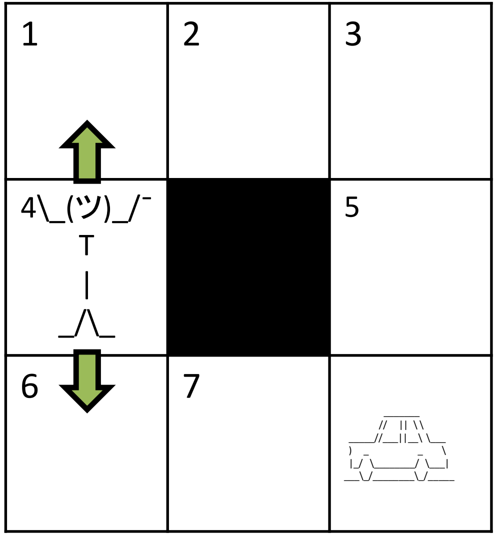
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		0		0
Square 2	0	0		
Square 3	0			.8
Square 4			0	0
Square 5			0	1
Square 6		.8	0	
Square 7	0	1		

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

ITERATION 3



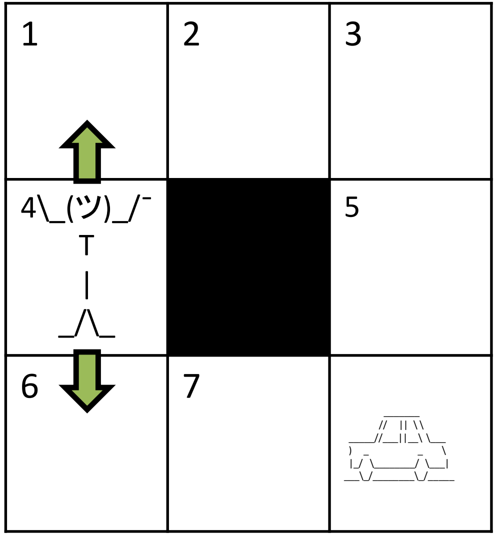
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		0		0
Square 2	0	.8^2		
Square 3	0			.8
Square 4			0	.8^2
Square 5			.8^2	1
Square 6		.8	0	
Square 7	.8^2	1		

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

ITERATION 4



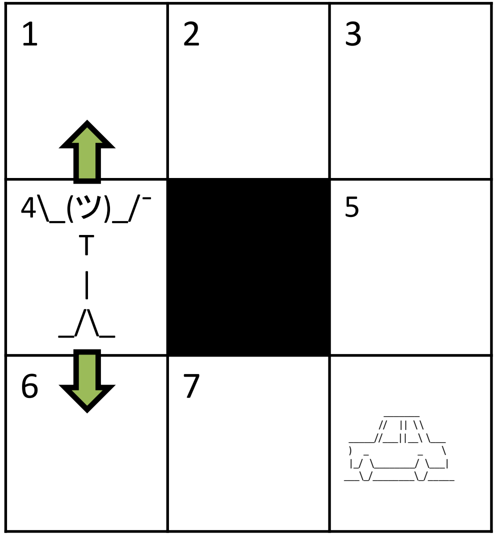
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		.8^3		.8^3
Square 2	0	.8^2		
Square 3	.8^3			.8
Square 4			0	.8^2
Square 5			.8^2	1
Square 6		.8	.8^3	
Square 7	.8^2	1		

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

ITERATION 5



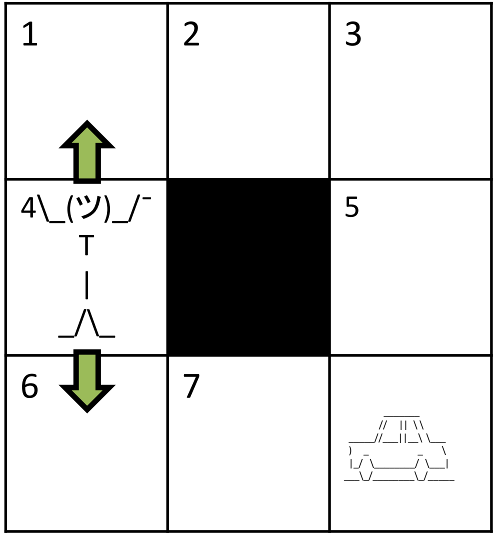
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		.8^3		.8^3
Square 2	.8^4	.8^2		
Square 3	.8^3			.8
Square 4			.8^4	.8^2
Square 5			.8^2	1
Square 6		.8	.8^3	
Square 7	.8^2	1		

Gridworld Example

$$Q^{new}(s,a) \leftarrow 1 \cdot [\, r(s,a,s') + .8 \cdot \max_{a'} Q^{old}(s',a') \,]$$

$$r(s,a,s') = \begin{cases} 1, & s' = \text{car} \\ 0, & \text{otherwise} \end{cases}$$

$Q(s,a) = \gamma^N$, where N is the fewest possible moves required to get to the car after taking action a from state s



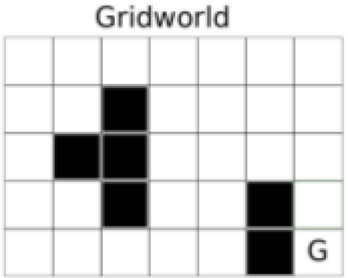
$Q(s,a)$	LEFT	RIGHT	UP	DOWN
Square 1		.8^3		.8^3
Square 2	.8^4	.8^2		
Square 3	.8^3			.8
Square 4			.8^4	.8^2
Square 5			.8^2	1
Square 6		.8	.8^3	
Square 7	.8^2	1		

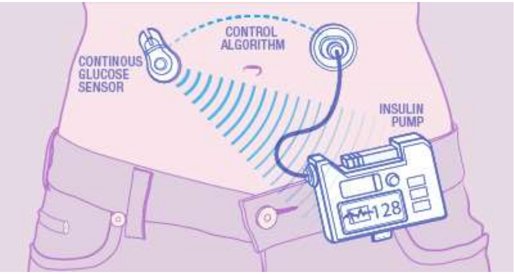
Q-Learning: Represent Q Function as a Look-Up Table

Learning rate, $\alpha \in [0,1]$

Discount factor, $\gamma \in [0,1]$

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q^{old}(s_t, a_t)}_{\substack{\text{Previous Estimate of Value of} \\ \text{Action } a_t \text{ when in state } s_t}} + \alpha \cdot \underbrace{[r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)]}_{\substack{\text{Updated Estimate of Value of} \\ \text{Action } a_t \text{ when in state } s_t \\ \text{After Observing } r_t}}$$





Limitations of Tabular Q Learning

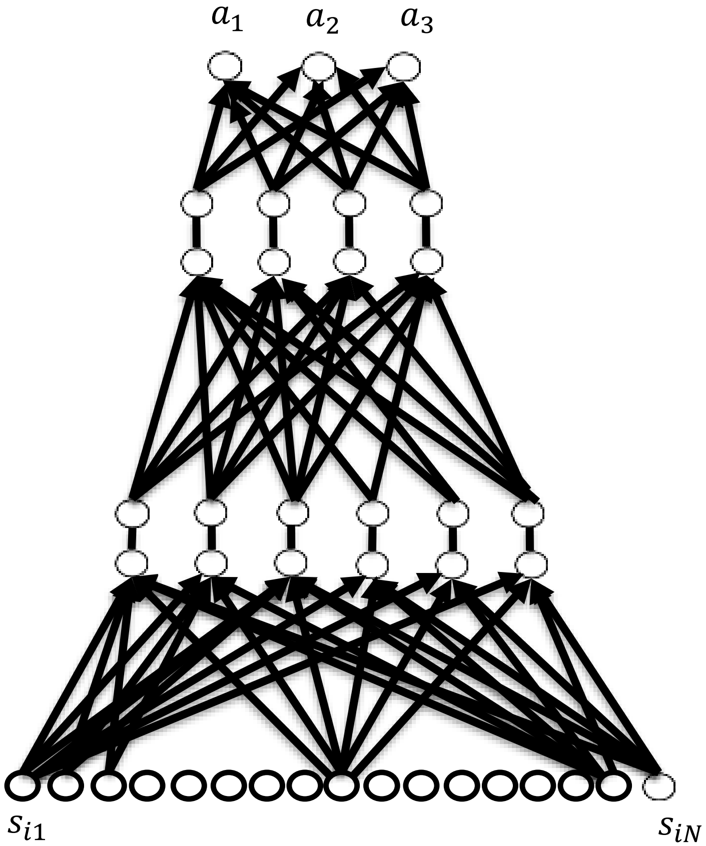
$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot Q^{old}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)]$$

❑ Simple update rule, but …..

- The Q function is stored as a table/matrix, which is impractical when considering a large number of states and actions

- No real capacity to generalize across sequence types, because the tabular Q function does not have a functional form

Deep Q Learning: Represent Q Function as a Deep Neural Network

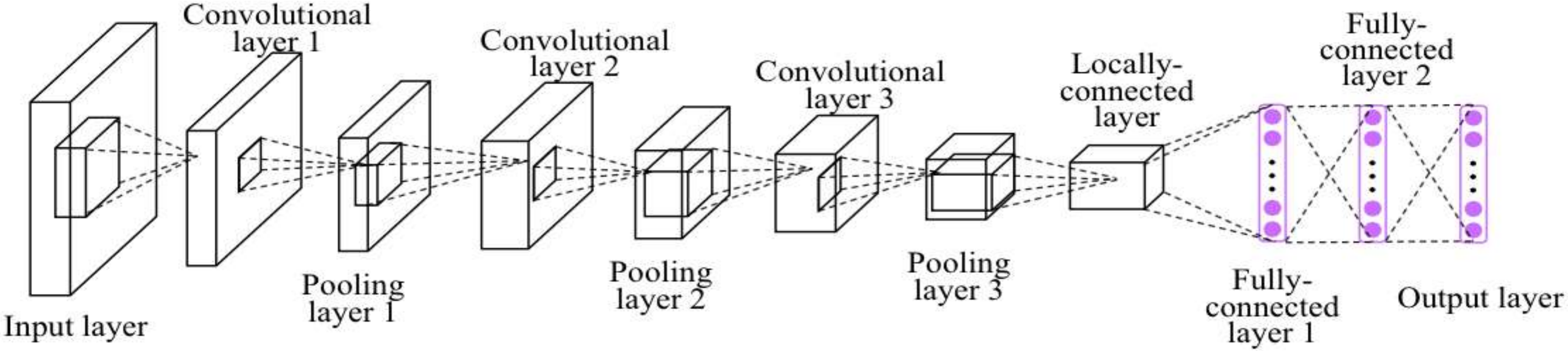


- The Q-function modeled via a neural network
- The state is input (e.g., an image with *N* pixels), and at the output we model the Q-function $Q(s, a)$ for each possible action
- Represent the neural network model as $Q(s, a; \theta)$ where θ represent the neural network parameters we wish to learn
- After we learn θ the policy is

$$\pi(s) = \operatorname*{argmax}_a Q(s, a; \theta)$$

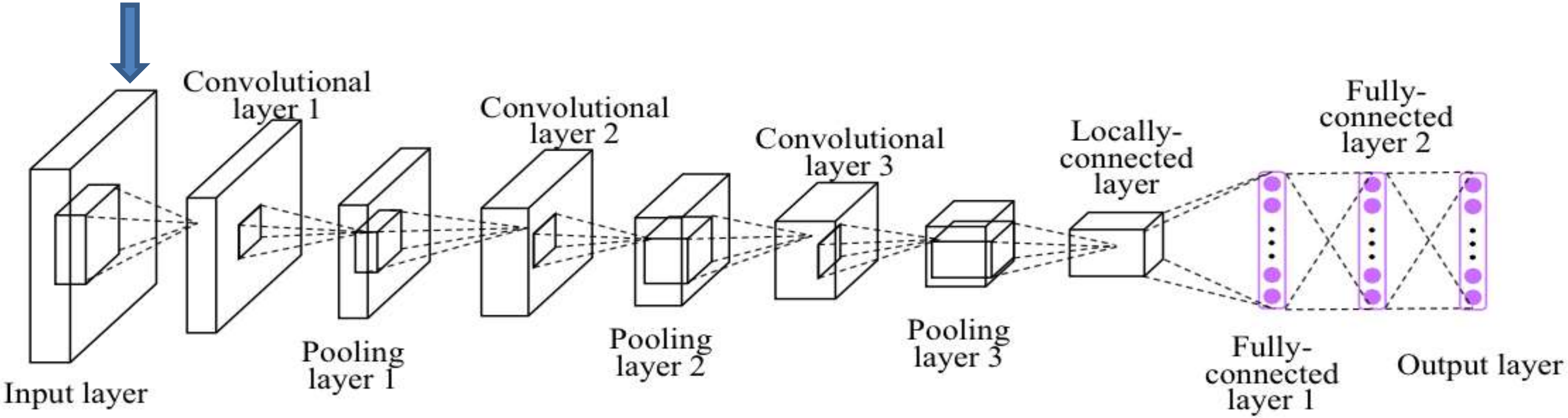
Power of Deep Q Networks for RL With Images

- Seek parameters θ that best approximate the Q function

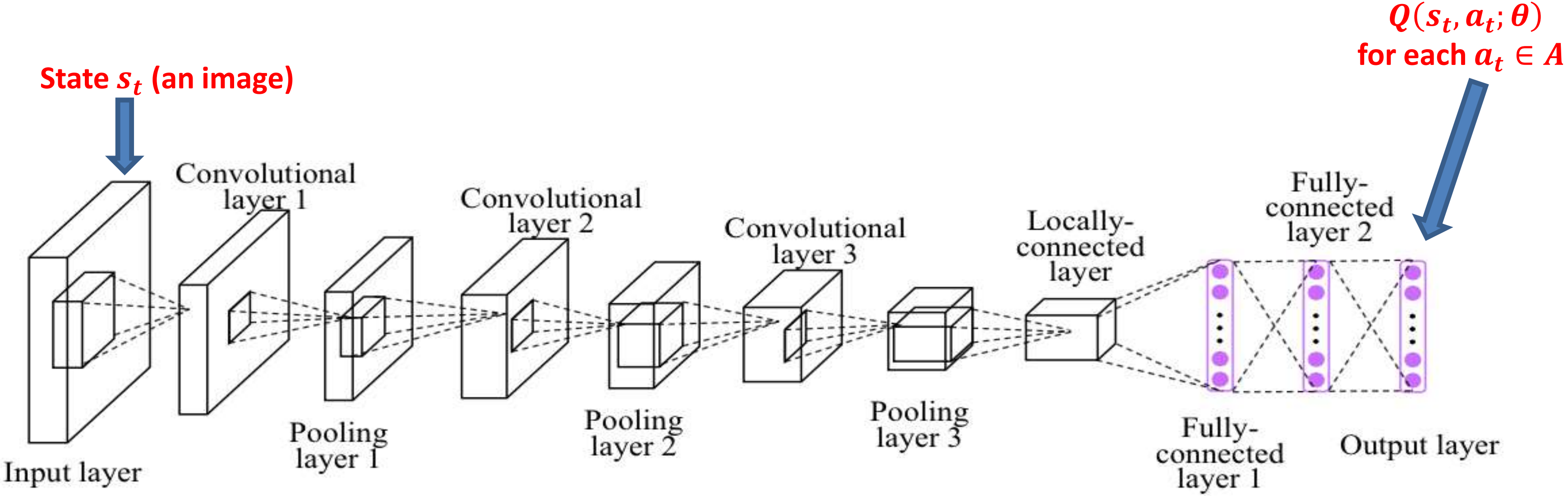


Power of Deep Q Networks for RL With Images

State s_t (an image)



Power of Deep Q Networks for RL With Images



Atari Games

Final Thoughts

- Tabular Q Learning employs the update rule

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot Q^{old}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot Q^{old}(s_{t+1}, a_{t+1})]$$

- Deep Q Learning uses a deep neural network with parameters θ to approximate the Q function

- Reinforcement learning has been studied for decades, and there are many other methods that we have not considered here, and are worth learning about for those interested

- There is also a rich theoretical foundation to RL, and therefore much is known about the fundamentals of these methods