Reinforcement Learning

Lecture 1

Lawrence Carin
Duke University

- Consider an MD, seeking to treat patients in an effective manner, while minimizing costs
- The state of health for each patient is observed in terms of a set of clinical variables
- Assume there are a set of actions the MD can perform
- Each action will impact the health state of the patient (possibly no change in the state)
- Seek a policy that achieves the best outcomes at the lowest average price
- Could apply to particular types of patients (e.g., diabetics) or in particular settings within a health system (e.g., operating room)

- Assume the set of observations defining the patient **state** represented by vector s
- Assume the set of **actions** that may be taken is $A = \{a_1, \dots, a_m\}$
- There is **randomness** in how a patient in state s will respond to a given action
- Let $P(s, a, s')$ represent the *probability* that when a patient is in state s and the MD takes action a, the patients state will change to s'
- Let $r(s, a, s')$ represent the "**reward**" associated with the MD taking action a for a patient in state s, and then the patient transiting to new state of health s'
- The reward $r(s, a, s')$ may reflect the *immediate* impact for the patient of the change $s \rightarrow s'$ and also the cost of action a

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

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

- Goal: Develop a **policy** that defines for the MD the optimal action a to take when presented by a patient in state s; the policy may define *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

Big Challenge

- We typically do not know $P(s, a, s')$
- How can we learn a policy without this?
- We can just experience/try things, keep a record of outcomes, and adapt and adjust
- Reinforce actions for particular patient states that are rewarding
- Discourage actions that are expensive and yield poor outcomes
- In many ways this is how medicine works, over time

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

Medicine/Health

- Reinforcement learning is the formalization of this challenge

- Addresses sequential decision making in an uncertain (stochastic) world

Medicine/Health

Monitoring/Maintenance of Factory

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

Medicine/Health

Monitoring/Maintenance of Factory

Investing

- 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

- Consider diabetes control

- Assume that the patient **state** s is defined by the **minimum and maximum glucose concentration from previous day**

- The **action** a may be the **rate of continuous insulin supply** and the **bolus dose**

Illustrative Example

- Consider diabetes control

- Assume that the patient **state** s is defined by the **minimum and maximum glucose concentration from previous day**

- The **action** a may be the **rate of continuous insulin supply** and the **bolus dose**

- $r(s, a, s')$ may be defined to reward (punish) glucose levels that are desirable (undesirable)

- Assume that we may specify $r(s, a, s')$

Solution Setup

- Assume that the patient **state** s is defined by the **minimum and maximum glucose concentration from previous day**
    - Discretize the continuous range of the state values into n bins

- The **action** a may be the **rate of continuous insulin supply** and the **bolus dose**
    - Discretize the continuous range of the action values into m bins

Solution Setup

- Assume that the patient **state** s is defined by the **minimum and maximum glucose concentration from previous day**
  - Discretize the continuous range of the state values into n bins
- The **action** a may be the **rate of continuous insulin supply** and the **bolus dose**
  - Discretize the continuous range of the action values into m bins
- The Q function $Q(s, a)$ is an $n \times m$ matrix, denoting the value of taking action a when in state s

Simple, Intuitive Solution

- Set initial values of $n \times m$ matrix $Q(s, a)$ based on prior available medical knowledge, or set at random

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

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

Simple, Intuitive Solution

- Set initial values of $n \times m$ matrix $Q(s, a)$ based on prior available medical knowledge, or set at random

- 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 $Q^{old}(s, a)$

Simple, Intuitive Solution

- Set initial values of $n \times m$ matrix $Q(s,a)$ based on prior available medical knowledge, or set at random

- 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:

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

Simple, Intuitive Solution

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

- 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 the TD is positive, the inferred value of taking action a in state s is increased; if TD negative, the value of taking action a in state s is diminished

Simple, Intuitive Solution

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

- If the reward $r(s,a,s')$ is large (small) then $Q^{new}(s,a)$ is typically increased (diminished)

Simple, Intuitive Solution

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

- ❑ If the reward $r(s,a,s')$ is large (small) then $Q^{new}(s,a)$ is typically increased (diminished)
- ❑ Problem: This 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'

Example: Problem With Myopic Policy

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

- Assume state of patient s corresponds to severe poor health
- A particular action a may have probable positive immediate reward $r(s,a,s')$
- However, there may be serious long-term complications (e.g., loss of opportunity to have children)
- A policy that only accounts for the immediate reward would not account for long-term, later consequences

Extension

- Recall our simple solution

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

Extension

- Recall our simple solution

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

- Consider the extension

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

Extension

- ❑ Recall our simple solution

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

- ❑ Consider the extension

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

Extension

- Recall our simple solution

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

- Consider the extension

Discount Factor Between 0 and 1

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

Extension

- Recall our simple solution

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

- How about extending it as

$$Q^{new}(s,a) \leftarrow Q^{old}(s,a) + \alpha \cdot \underbrace{[\, r(s,a,s') + \gamma \cdot \max_{a'} Q^{old}(s',a') - Q^{old}\,(s,a)]}_{\text{Non-Myopic Temporal Difference (TD)}}$$

Q Learning

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

- Based on simple logic and intuition, we have derived an algorithm for a system/MD to learn based on experience

- This is actually a widely employed method for reinforcement learning, called Q Learning

Q Learning

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

- Based on simple logic and intuition, we have derived an algorithm for a system/MD to learn based on experience

- This is actually a widely employed method for reinforcement learning, called Q Learning

- Based on the learned matrix $Q(s,a)$ the policy that is typically employed is

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

Q Learning

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

Q Learning

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

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

Q Learning

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

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

Q Learning

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

Q Learning

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

Q Learning

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

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

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

Previous Estimate of Value of Action a_t when in state s_t

Updated Estimate of Value of Action a_t when in state s_t After Observing r_t

Q Learning

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

- Don't need to actually perform the action defined by $\operatorname{argmax}_a Q^{old}(s_{t+1}, a)$

Q Learning

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

- Don't need to actually perform the action defined by $\underset{a}{\operatorname{argmax}}\, Q^{old}(s_{t+1}, a)$

- Next action a_{t+1} may be based on $\underset{a_{t+1}}{\operatorname{argmax}}\, Q^{new}(s_{t+1}, a_{t+1})$, we then observe immediate reward r_{t+1} and new state s_{t+2}

$$\ldots\ s_t \quad a_t \quad r_t \quad s_{t+1} \quad a_{t+1} \quad r_{t+1} \quad s_{t+2}\ \ldots$$

Q Learning

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

- Don't need to actually perform the action defined by $\underset{a}{\operatorname{argmax}}\, Q^{old}(s_{t+1}, a)$

- Next action a_{t+1} may be based on $\underset{a_{t+1}}{\operatorname{argmax}}\, Q^{new}(s_{t+1}, a_{t+1})$, we then observe immediate reward r_{t+1} and new state s_{t+2}

$$\dots\ s_t \quad a_t \quad r_t \quad s_{t+1} \quad a_{t+1} \quad r_{t+1} \quad s_{t+2}\ \dots$$

- Sequentially and continually update the Q function $Q^{new}(s_t, a_t)$, which is an $n \times m$ matrix

Learning Rate

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

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

- ❑ Convergence guaranteed after sufficient experience, if learning rate diminishes with time
- ❑ In practice one may set $\alpha = 0.1$
- ❑ This online Q-learning setup allows one to learn an optimal policy based directly on experience

Q Learning vs. SARSA

- **Q-learning**, from previous slides:

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

Assumes access to s_t a_t r_t s_{t+1} (SARS); we don't implement a_{t+1} to update the Q functions

Q Learning vs. SARSA

- **Q-learning**, from previous slides:

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

Assumes access to $s_t\ a_t\ r_t\ s_{t+1}$ (SARS); we don't implement a_{t+1} to update the Q functions

- **SARSA** based learning considers the slightly modified

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

Assumes access to $s_t\ a_t\ \ r_t\ \ s_{t+1}\ a_{t+1}$ (SARSA); a_{t+1} may be specified by some chosen mechanism

ϵ-Greedy: Exploration and Exploitation

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

- Assume we observe s_t a_t r_t s_{t+1}

ϵ-Greedy: Exploration and Exploitation

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

- Assume we observe $s_t \;\; a_t \;\; r_t \;\; s_{t+1}$

- Which next action a_{t+1} should we take within Q learning or SARSA?

ϵ-Greedy: Exploration and Exploitation

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

- Assume we observe $s_t \;\; a_t \;\; r_t \;\; s_{t+1}$

- Which next action a_{t+1} should we take within Q learning or SARSA?

- ϵ-Greedy (small ϵ):

  - With probability ϵ, choose a_{t+1} at random
  - With probability $1-\epsilon$, next action $a_{t+1} = \underset{a}{\operatorname{argmax}} Q^{old}(s_{t+1}, a)$

ϵ-Greedy: Exploration and Exploitation

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

- Assume we observe $s_t \;\; a_t \;\; r_t \;\; s_{t+1}$

- Which next action a_{t+1} should we take within Q learning or SARSA?

- ϵ-Greedy (small ϵ):

  - With probability ϵ, choose a_{t+1} at random
  - With probability $1-\epsilon$, next action $a_{t+1} = \underset{a}{\operatorname{argmax}}\, Q^{old}(s_{t+1}, a)$

Allows exploration with probability ϵ

Reinforcement Learning

Lecture 2

Lawrence Carin
Duke University

Limitations of Tabular Q Learning

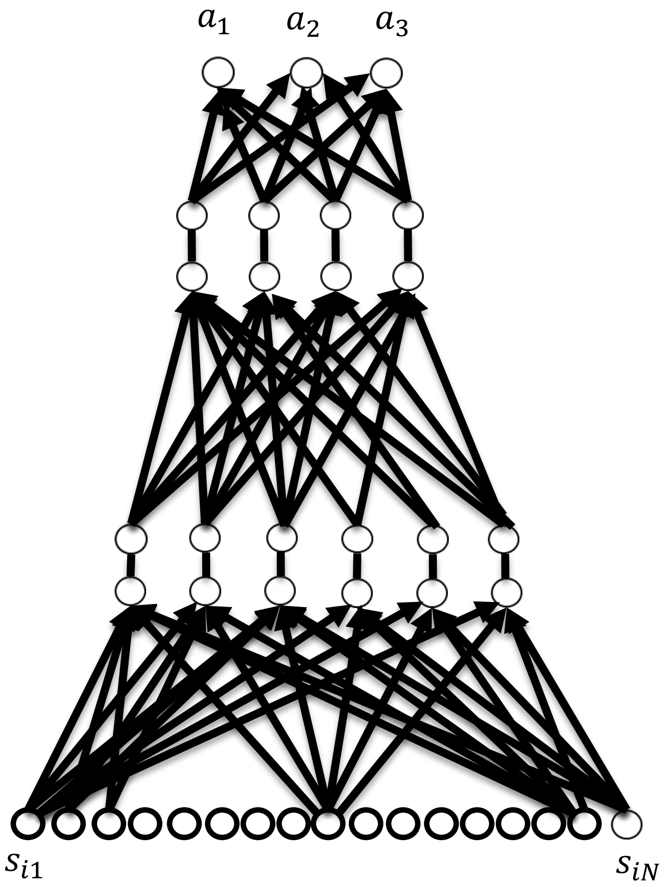
$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)\cdot Q^{old}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a Q^{old}(s_{t+1}, a)]$$

- 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

Represent Q Function as a 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

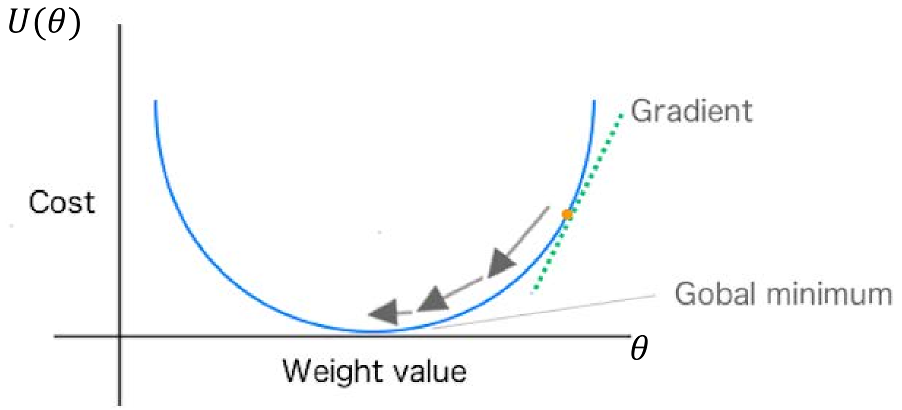
$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a; \theta)$$

Deep Q Learning (DQN)

- If we update θ like in Q learning, with every new observed $s_t \;\; a_t \;\; r_t \;\; s'_{t+1}$, at each step we seek to minimize

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\,]^2$$

- To move θ such that $U(\theta; s_t, a_t)$ is made smaller, we update model parameters as



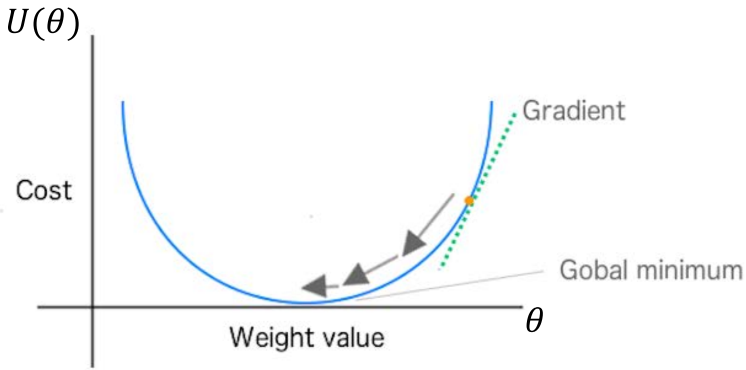
Gradient Descent

- If we update θ like in Q learning, with every new observed $s_t \;\; a_t \;\; r_t \;\; s'_{t+1}$, at each step we seek to minimize

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$

- To move θ such that $U(\theta; s_t, a_t)$ is made smaller, we update model parameters as

$$\theta^{new} \leftarrow \theta^{old} - \alpha \cdot \nabla_\theta U(\theta; s_t, a_t)|_{\theta=\theta^{old}}$$



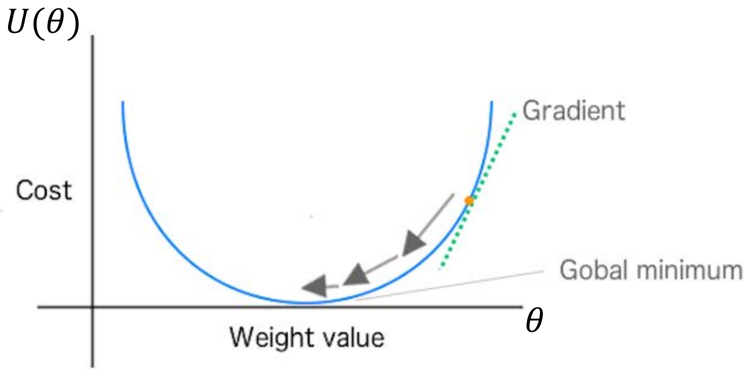
Gradient Descent

- If we update θ like in Q learning, with every new observed $s_t \ a_t \ r_t \ s'_{t+1}$, at each step we seek to minimize

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$

- To move θ such that $U(\theta; s_t, a_t)$ is made smaller, we update model parameters as

$$\theta^{new} \leftarrow \theta^{old} - \alpha \cdot \underbrace{\nabla_\theta U(\theta; s_t, a_t)|_{\theta=\theta^{old}}}_{\text{Multi-dimensional “slope”}}$$



Gradient Descent

- If we update θ like in Q learning, with every new observed $s_t \;\; a_t \;\; r_t \;\; s'_{t+1}$, at each step we seek to minimize

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$

- To move θ such that $U(\theta; s_t, a_t)$ is made smaller, we update model parameters as

$$\theta^{new} \leftarrow \theta^{old} - \alpha \cdot \nabla_\theta U(\theta; s_t, a_t)|_{\theta=\theta^{old}}$$

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

Gradient Descent

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

Gradient Descent

$$\theta^{new} \leftarrow \theta^{old} + \alpha \cdot \left[r_t + \gamma \cdot \max_{a'} Q\left(s'_{t+1}, a'; \theta^{old}\right) - Q(s_t, a_t; \theta^{old}) \right] \underbrace{\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}}}_{\text{Modification to account for neural network parameters } \theta}$$

Gradient Descent

$$\theta^{new} \leftarrow \theta^{old} + \alpha \cdot \underbrace{\left[r_t + \gamma \cdot \max_{a'} Q\left(s'_{t+1}, a'; \theta^{old}\right) - Q(s_t, a_t; \theta^{old})\right]}_{\text{Looks a lot like the prior Q learning}} \underbrace{\nabla_\theta Q(s_t, a_t; \theta)|_{\theta=\theta^{old}}}_{\text{Modification to account for neural network parameters } \theta}$$

- ❑ One may show rigorously that this is directly related to the prior sequential Q learning
- ❑ Details not very difficult (first-order Taylor series expansion), but beyond the scope of our discussion (**see Appendix**)

Revisit What Deep Q Network Seeks to Learn

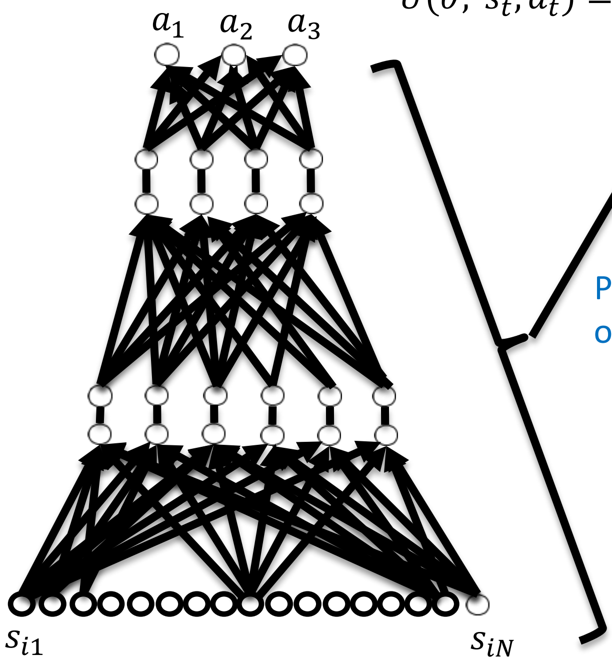
- Seek parameters θ that minimize prediction error

$$U(\theta;\ s_t, a_t) = \frac{1}{2}\left[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ \right]^2$$

Revisit What Deep Q Network Seeks to Learn

- Seek parameters θ that minimize prediction error

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$



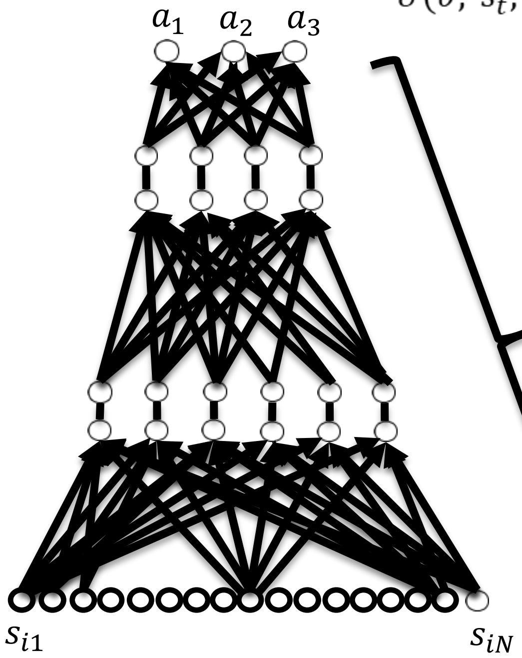
Predictive model for value of state-action pair (s_t, a_t)

Revisit What Deep Q Network Seeks to Learn

- Seek parameters θ that minimize prediction error

$$U(\theta;\, s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \overbrace{\gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})}^{\text{Discounted future rewards}}]\,]^2$$

Next immediate reward



Predictive model for value of state-action pair (s_t, a_t)

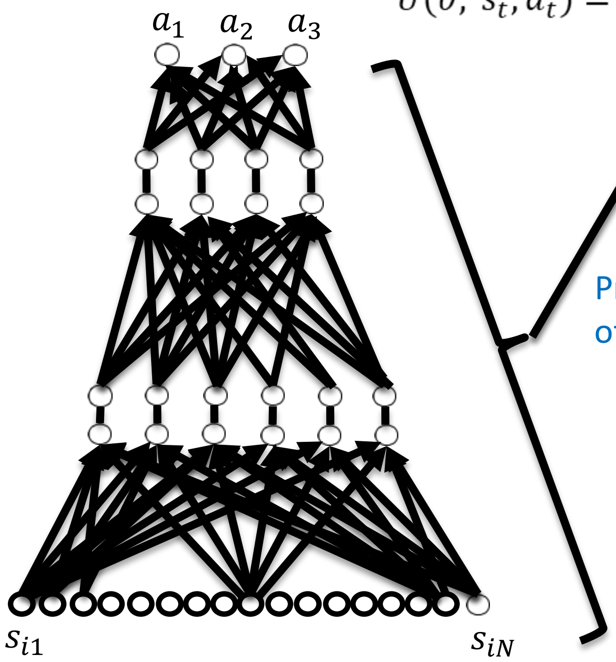
Revisit What Deep Q Network Seeks to Learn

- Seek parameters θ that minimize prediction error

Discounted future rewards

$$U(\theta;\, s_t, a_t) = \frac{1}{2}\left[Q(s_t, a_t; \theta) - [r_t + \overbrace{\gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]}^{\text{Discounted future rewards}}\right]^2$$

Next immediate reward



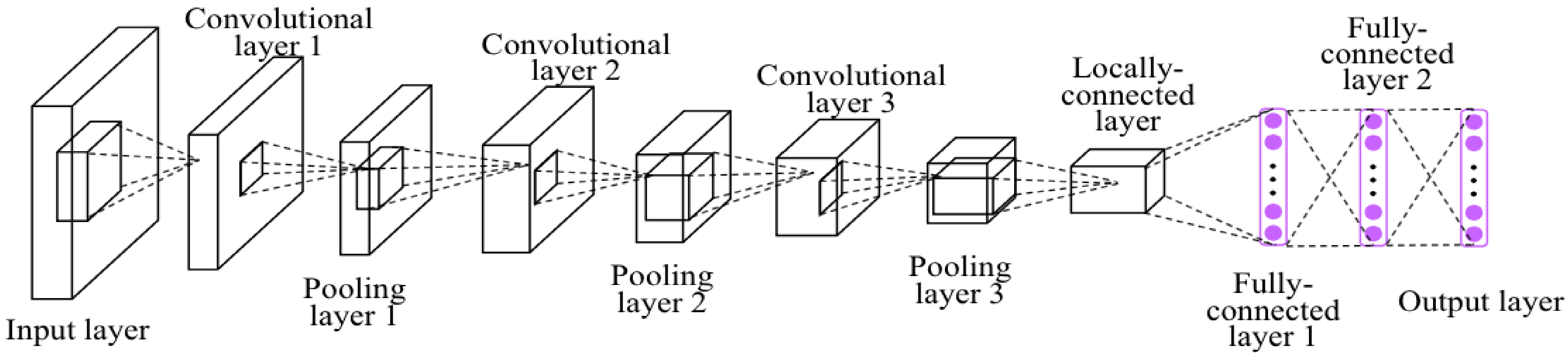
Predictive model for value of state-action pair (s_t, a_t)

Supervised Learning

Power of Deep Q Networks for RL With Images

- Seek parameters θ that minimize prediction error

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$

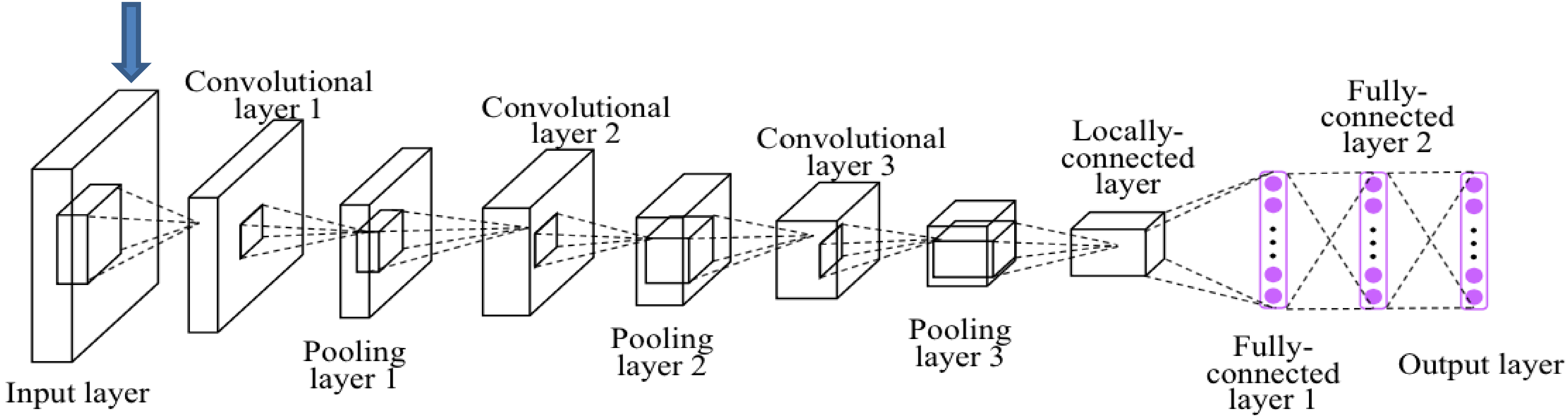


Power of Deep Q Networks for RL With Images

- Seek parameters θ that minimize prediction error

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$

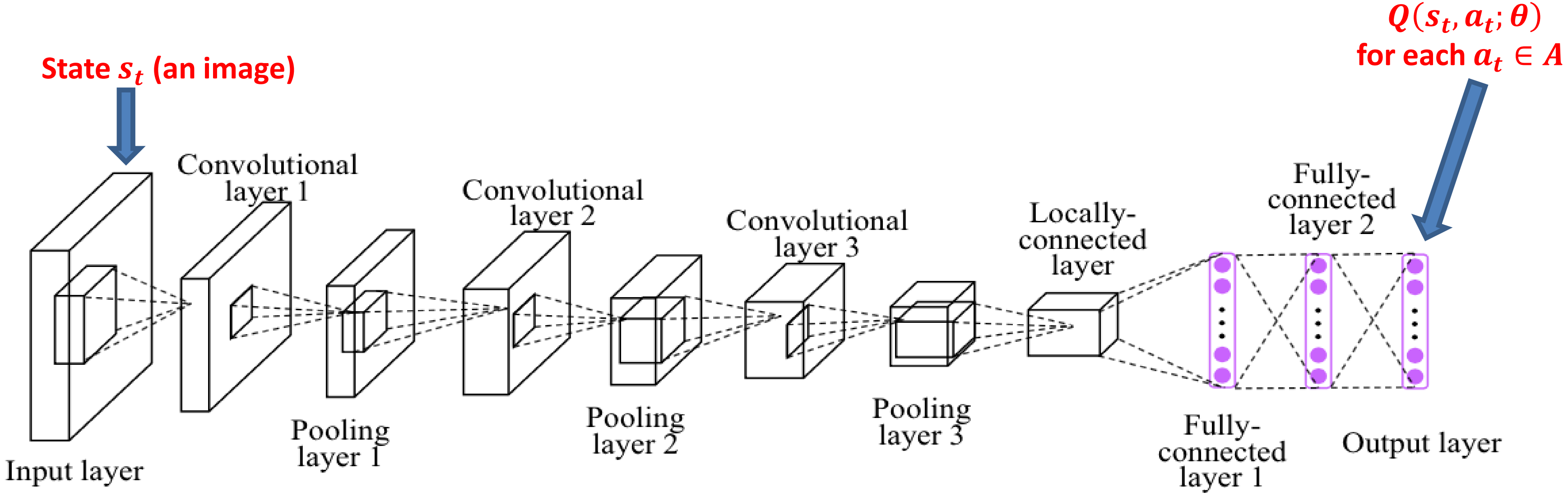
State s_t (an image)



Power of Deep Q Networks for RL With Images

- Seek parameters θ that minimize prediction error

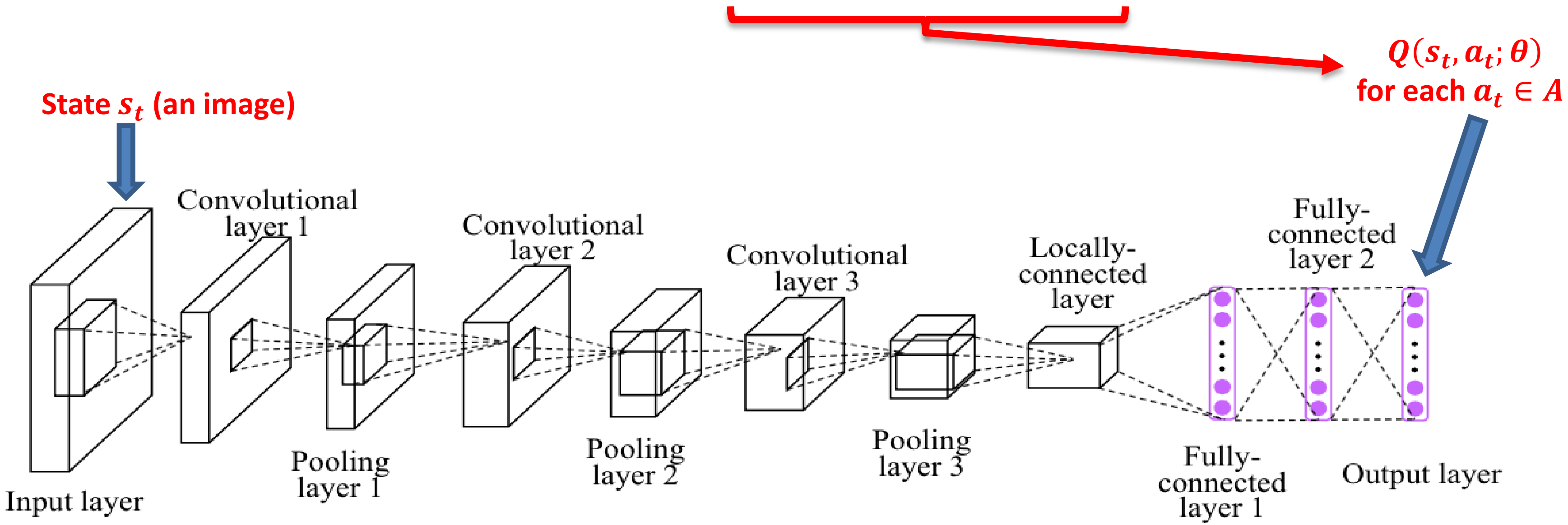
$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\,]^2$$



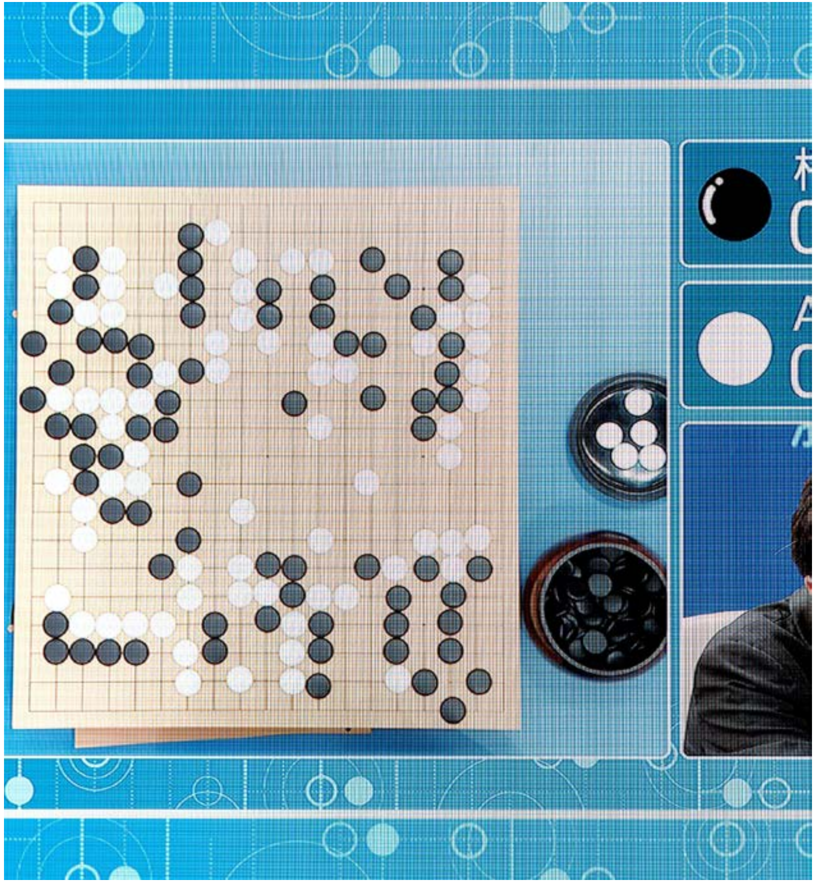
Power of Deep Q Networks for RL With Images

- Seek parameters θ that minimize prediction error

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\ ]^2$$




Google DeepMind
Challenge Match
8 - 15 March 2016
ALPHAGO
01:59:54
LEE SEDOL
01:59:02
AlphaGo



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 minimize the functional

$$U(\theta;\ s_t, a_t) = \frac{1}{2}[Q(s_t, a_t; \theta) - [r_t + \gamma \cdot \max_{a'} Q(s'_{t+1}, a'; \theta^{old})]\,]^2$$

- 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

Appendix 1

Introduction to Markov Decision Process (MDP)

Markov Decision Process (MDP)

- A set of **states** $S = (s_1, \ldots, s_n)$
  We observe the state of the system/environment/subject

Markov Decision Process (MDP)

- A set of **states** $S = (s_1, \dots, s_n)$
  We observe the state of the system/environment/subject

- A set of **actions** $A = (a_1, \dots, a_m)$
  We may select an action to take, given that the system is in an observed state s

Markov Decision Process (MDP)

- A set of **states** $S = (s_1, \dots, s_n)$
  We observe the state of the system/environment/subject

- A set of **actions** $A = (a_1, \dots, a_m)$
  We may select an action to take, given that the system is in an observed state s

- **State-transition probabilities** $P(s, a, s')$
  Defines the probability of transitioning $s \to s'$ upon taking action a when in state s; $\sum_{s' \in S} P(s, a, s') = 1$

Markov Decision Process (MDP)

- A set of **states** $S = (s_1, \dots, s_n)$
  We observe the state of the system/environment/subject

- A set of **actions** $A = (a_1, \dots, a_m)$
  We may select an action to take, given that the system is in an observed state s

- **State-transition probabilities** $P(s, a, s')$
  Defines the probability of transitioning $s \to s'$ upon taking action a when in state s'; $\sum_{s' \in S} P(s, a, s') = 1$

- Set of **rewards** $r(s, a, s')$ reflecting the *immediate* reward of taking action a in state s and transiting to state s'

Markov Decision Process (MDP)

- A set of **states** $S = (s_1, \dots, s_n)$
  We observe the state of the system/environment/subject

- A set of **actions** $A = (a_1, \dots, a_m)$
  We may select an action to take, given that the system is in an observed state s

- **State-transition probabilities** $P(s, a, s')$
  Defines the probability of transitioning $s \rightarrow s'$ upon taking action a when in state s'; $\sum_{s' \in S} P(s, a, s') = 1$

- Set of **rewards** $r(s, a, s')$ reflecting the *immediate* reward of taking action a in state s and transiting to state s'

- **Discount factor** $\gamma \in [0,1)$
  Reflects the degree to which future rewards are discounted relative to immediate rewards

Markov Decision Process (MDP)

Set of states

State-action-state transition probabilities

$$S, A, P, R, \gamma$$

Discount factor

Set of actions

Rewards

Policy

- A policy π specifies which action $a \in A$ is taken when in state $s \in S$

- The policy may be stochastic, with $\pi(a; s)$ reflecting the *probability* of action a when in state s; $\sum_{i=1}^{m} \pi(a_i; s) = 1$

Policy

- A policy π specifies which action $a \in A$ is taken when in state $s \in S$

- The policy may be stochastic, with $\pi(a; s)$ reflecting the *probability* of action a when in state s; $\sum_{i=1}^{m} \pi(a_i; s) = 1$

- Consider the sequence $(s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{N-1}, a_{N-1}, r_{N-1}, s_N, a_N, r_N)$

  - The total reward for this sequence is $r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^N r_N$
  - This total reward is a *random variable,* because the sequence of states visited is random

- We wish to learn the policy π to maximize the <u>expected</u> reward of the "agent" over a sequence of states, actions and rewards

Expected (Average) Reward of a Policy

- The state-action value function $Q^{\pi}(s,a)$ of any policy π indicates the expected (average) discounted total reward when taking action a in state s and following the optimal policy thereafter

$$Q^{\pi}(s,a) = E_{a_t \sim \pi; s_t \sim P}\left(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right)$$

- We may re-express $Q^{\pi}(s,a)$ as

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

- Immediate *expected* reward $R(s,a) = \sum_{s'} P(s,a,s') r(s,a,s')$

Deterministic Policy

- Assumed the policy selects the single most rewarding action when in state s

$$\pi(s) = \arg\max_{a \in A} Q^{\pi}(s, a)$$

- Bellman's equation

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

- Suggests an iterative solution to learn the optimal deterministic policy

Policy Iteration

- Initialize the Q functions with $Q_0(s,a) = R(s,a)$

- Iterate as

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

- May be shown that for a large number of steps this iterative solution converges to $Q^{\pi}(s,a)$ for the optimal policy $\pi(a;s)$

Practical Limitations of MDP Policy Learning

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

- Assumes we have access to a model of the environment, given by $P(s,a,s')$
- Typically we do not have these model parameters
- Possible strategy: Estimate $P(s,a,s')$ based on experience with environment
- Challenge: May require a lot of such experience to get good estimates

Appendix 2

Alternative Derivation of Q Learning

Learn the Policy as we Experience Environment

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

Learn the Policy as we Experience Environment

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

- Based on the definition of $R(s,a)$, this is equivalent to

$$Q^{new}(s,a) \leftarrow \sum_{s' \in S} P(s,a,s') \left[r(s,a,s') + \gamma \max_{a' \in A} Q^{old}(s',a')\right]$$

Learn the Policy as we Experience Environment

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

- Based on the definition of $R(s,a)$ this is equivalent to

$$Q^{new}(s,a) \leftarrow \sum_{s' \in S} P(s,a,s') \left[r(s,a,s') + \gamma \max_{a' \in A} Q^{old}(s',a') \right]$$

- By the definition of probability, as $K \to \infty$, this is equivalent to

$$Q^{new}(s,a) \leftarrow \frac{1}{T} \sum_{t=1}^{T} \left[r(s,a,s'_t) + \gamma \max_{a'} Q^{old}(s'_t,a') \right], \qquad \text{with each} \qquad s'_t \sim P(s,a,s')$$

Learning from Experience

$$Q^{new}(s,a) \leftarrow \frac{1}{T}\sum_{t=1}^{T} \left[\, r(s,a,s_t') + \gamma \max_{a'} Q^{old}(s_t',a')\right], \qquad \text{with each} \qquad s_t' \sim P(s,a,s')$$

- We don't know the underlying probability distribution $P(s,a,s')$
- But we can sample from it, from experience; <u>don't need to know or estimate $P(s,a,s')$</u>
- When in state s, try an action $a \in A$, and see what happens*, i.e.*, see what reward r is manifested
- Model learns to value, or *reinforce*, actions in a given state that are expected to be valuable

Naïve Q Learning

$$Q^{new}(s,a) \leftarrow \frac{1}{T}\sum_{t=1}^{T} \left[r(s,a,s'_t) + \gamma \max_{a'} Q^{old}(s'_t,a') \right], \qquad \text{with each} \qquad s'_t \sim P(s,a,s')$$

- Suggests an update rule we implement "on the fly", as we experience the environment

$$Q^{new}(s,a) \leftarrow Q^{old}(s,a) + \left[r_t + \gamma \max_{a'} Q^{old}(s'_t,a') \right], \qquad \text{with } s'_t \sim P(s,a,s')$$

Naïve Q Learning

$$Q^{new}(s,a) \leftarrow \frac{1}{T}\sum_{t=1}^{T} [\, r(s,a,s'_t) + \gamma \max_{a'} Q^{old}(s'_t,a')] \,, \qquad \text{with each} \qquad s'_t \sim P(s,a,s')$$

- Suggests an update rule we implement “on the fly”, as we experience the environment

$$Q^{new}(s,a) \leftarrow Q^{old}(s,a) + [r_t + \gamma \max_{a'} Q^{old}(s'_t,a')] \;, \qquad \text{with } \; s'_t \sim P(s,a,s')$$

- The $1/T$ factor doesn’t affect policy (max operation), and therefore we ignore it for simplicity

Q-Learning

- The simple/naïve setup is

$$Q^{new}(s,a) \leftarrow Q^{old}(s,a) + [r_t + \gamma \max_{a'} Q^{old}(s'_t, a')]$$

- Introduce a "learning rate" $\alpha \in [0,1]$ and modify the update rule as

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

- This is called Q learning, and it allows one to learn a policy on the life, adaptively, as the environment is experienced

Appendix 3

Connecting Deep Q Learning to Conventional Q Learning

DQN Parameter Update Equation

- Recall that in deep Q learning the update equation of the model parameters is:

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

- After implementing the parameter update, the Q function may be expressed as

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

Taylor Series Expansion

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

- Define

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

from which

$$Q(s, a; \theta^{new}) = Q(s, a; \theta^{old} + \Delta\theta)$$

- First-order Taylor-series expansion, for small $\Delta\theta$:

$$Q(s, a; \theta^{old} + \Delta\theta) \approx Q\left(s, a; \theta^{old}\right) + \Delta\theta^T \nabla_\theta Q(s, a; \theta)|_{\theta=\theta^{old}}$$

Taylor Series Expansion

- By the definition of $\Delta\theta$

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

- Define $\alpha' = \alpha \cdot ||\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)|_{\theta=\theta^{old}}||_2^2$, yielding approximately

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

Connecting DQN to Q Learning

- The update rule for the model parameters in deep learning is

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

- When viewed from the perspective of the Q function this yields

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

$$\alpha' = \alpha \cdot ||\nabla_\theta \mathrm{Q}(s_t, a_t; \theta)\big|_{\theta=\theta^{old}}||_2^2$$

- Hence, the update rule in deep Q learning for the model parameters corresponds exactly to the update rule in conventional Q learning

Connecting DQN to Q Learning

- When doing updates in deep Q learning we do NOT actually implement

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

because now the Q function is not simply a table, as in original Q learning

- We implement

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

meaning we update the functional model $Q(s, a; \theta)$ for the Q function