UC San Diego
JACOBS SCHOOL OF ENGINEERING

ECE/SIOC 228 Machine Learning for Physical Applications
Lecture 19: Deep Reinforcement Learning IV

Yuanyuan Shi

Assistant Professor, ECE

University of California, San Diego

Reference & Acknowledgement: David Silver's Course on RL https://www.davidsilver.uk/teaching/
Sergey Levine, Deep Reinforcement Learning https://rail.eecs.berkeley.edu/deeprlcourse/

Announcement

Course Evaluation

UC San Diego

- Fill the feedback form: https://academicaffairs.ucsd.edu/Modules/Evals/
- Your feedbacks are really important for us

Poster Session

UC San Diego

- **Time:** June 2nd 2:00pm – 4:00pm. Please arrive **at least 20 minutes** before 2pm to setup the posters, and the event starts right at 2pm.
- **Location:** Booker Conference Rooms (2512) and OSD/Multipurpose room (2509) in Jacobs Hall.
- **Grading and Presentation:** There will be 3 groups of judges, each group has two raters. Each poster will be at least evaluated by 2 raters, and the final score will be calculated as the average of all ratings. When each judge team comes, the team can make a 5 minutes (maximum) presentation, potentially followed by Q&As.
- **Poster size:** A common poster size fits the provided board is 36'' x 24''.
- **Other information**: https://canvas.ucsd.edu/courses/36302/discussion_topics/479036
- **If you're sick, please follow the campus COVID-19 guideline and stay at home.**
- **Checkout other groups' projects and posters, and have fun!**

Final Report and Github Repo

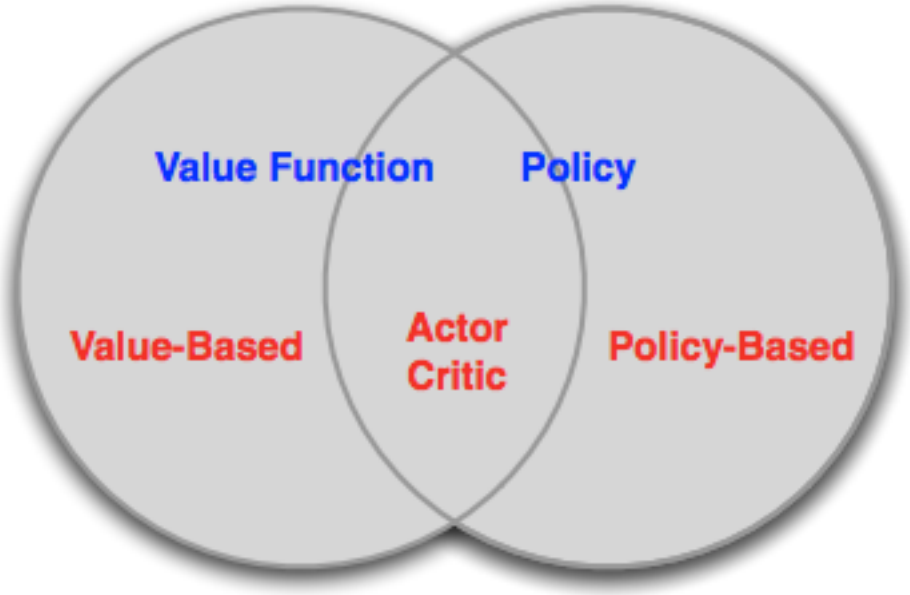
UC San Diego

- Import Dates:
  - **June 3rd , Friday 11:59pm**: poster due
  - **June 9th , Thursday, 11:59pm**: final report and code due.
  - Please include the link to your project Github Repo as the last sentence in abstract. Make sure the link works
- **Individual Contribution Section:** At the end of final report (after the conclusion section and before the reference part), please include a paragraph describing each team member's contribution for this project (coding, discussion, proposal/milestone/final report writing, poster session, etc)
- We will post more guidelines on the final report and Github repo later this week
- The final project reports and posters will be made available on the course website as past years

Recap: Reinforcement Learning Algorithms

UC San Diego

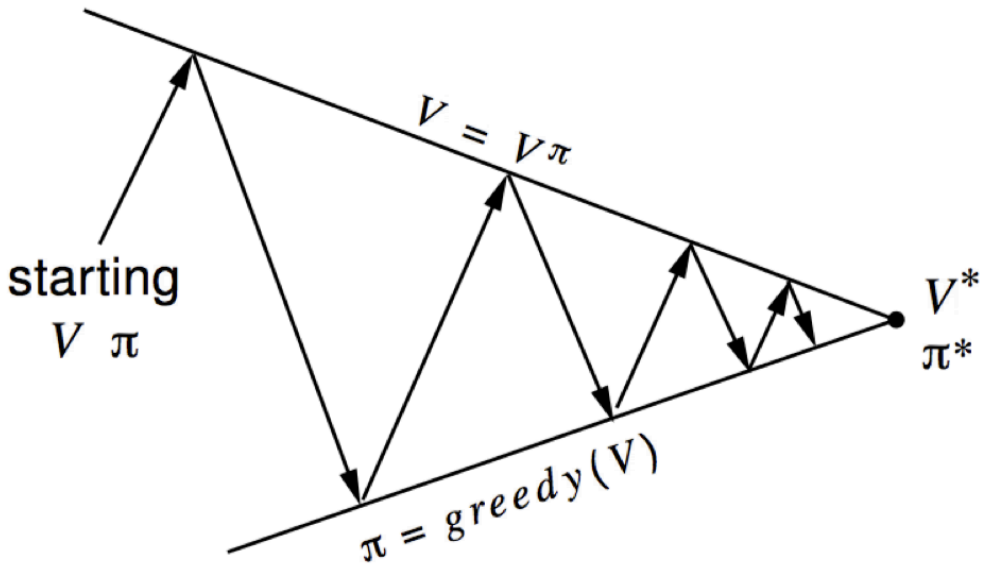
- Policy Based Reinforcement Learning
  - Learn a policy
  - No value function / not necessarily have a value function
- Actor-Critic
  - Learn value function
  - Learn policy
- **Value Based Reinforcement Learning**
  - Learn value function
  - Policy is implicit (e.g., greedy)



Value-based Reinforcement Learning

Model-based Policy Iteration (in MDP)

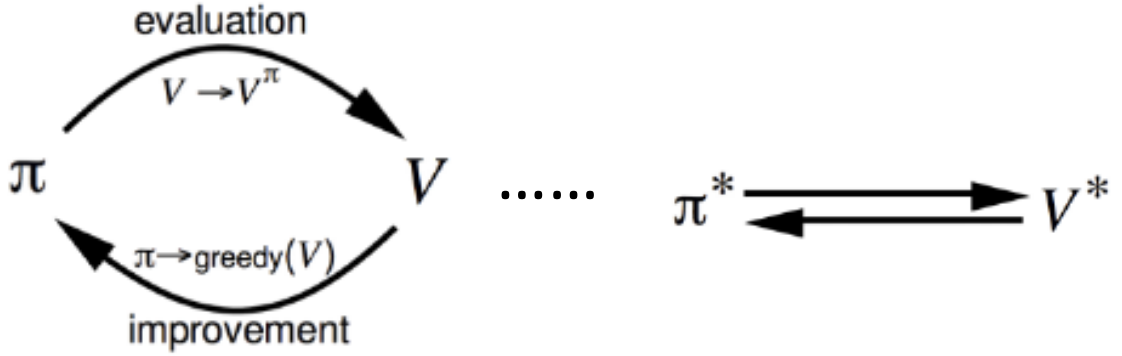
UC San Diego



Policy evaluation Estimate v_π
Iterative policy evaluation

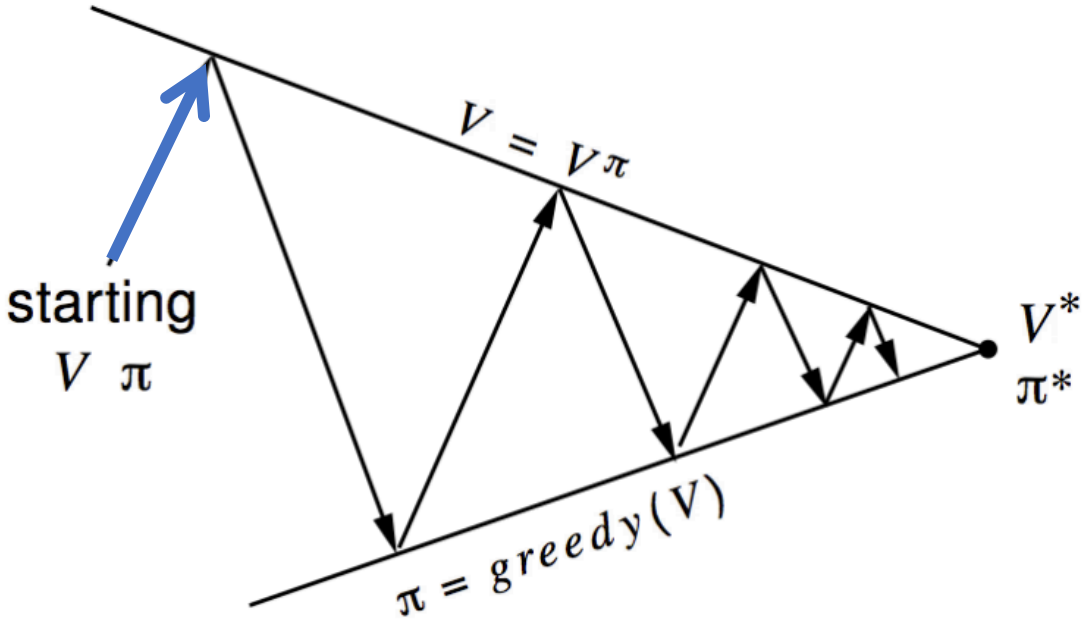
Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement

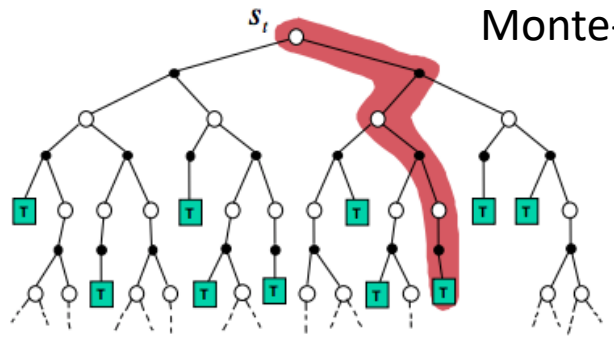
- Convergence property of policy iteration: $\pi \rightarrow \pi^*$
- Proof involves showing that each policy evaluation is a contraction and policy must improve each step, or be optimal policy (policy improvement theorem)

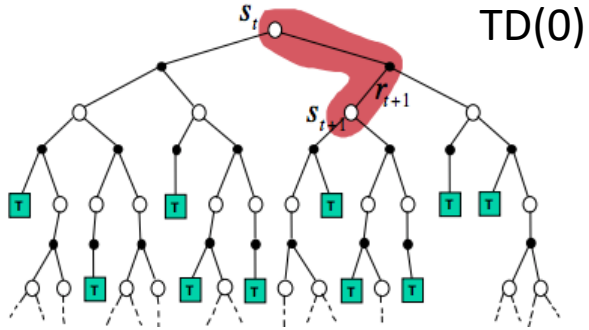


Model Free Policy Evaluation: $v_\pi(s)$

UC San Diego







Policy evaluation: Given a policy π, find out v_π

- Monte-Carlo policy evaluation $v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha\big(G_t - v_\pi(s_t)\big)$
- Temporal difference policy evaluation $v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha\big(r(s_t, a_t) + \gamma v_\pi(s_{t+1}) - v_\pi(s_t)\big)$

Model-Free Policy Optimization via Q-function

UC San Diego

- Greedy policy improvement over $v_\pi(s)$ requires model of the MDP

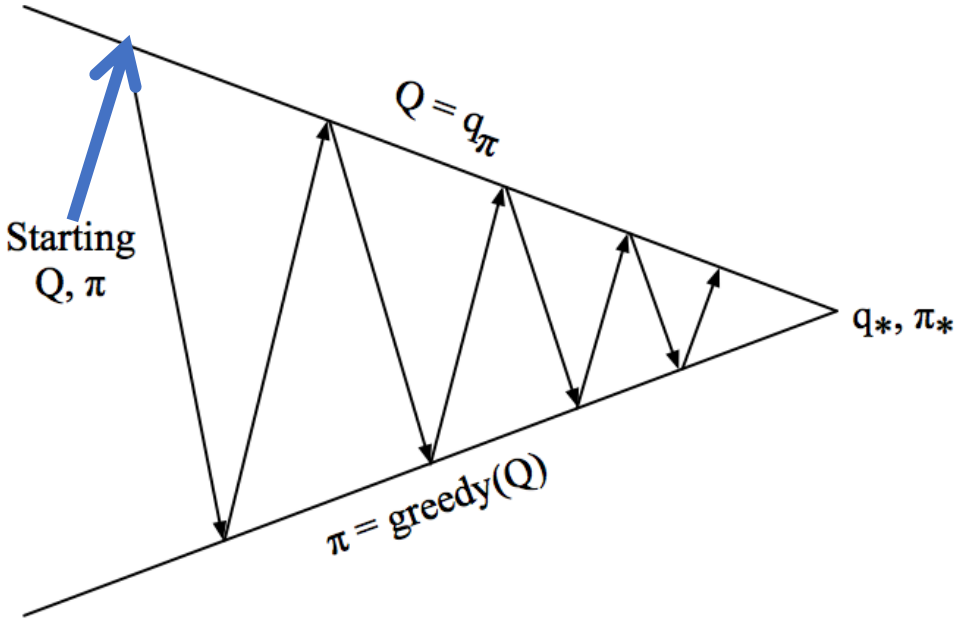
$$\pi'(s) = \underset{a \in \mathcal{A}}{argmax} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

- **Greedy policy improvement over $Q_\pi(s, a)$ is model-free**

$$\pi'(s) = \underset{a \in \mathcal{A}}{argmax}\, Q_\pi(s, a)$$

Model-Free Policy Evaluation: $Q_\pi(s, a)$

UC San Diego



Policy evaluation: Given a policy π, find out Q_π

- Monte-Carlo: $Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha\big(G_t - Q_\pi(s_t, a_t)\big)$
- Temporal difference (TD(0)): $Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha\big(r(s_t, a_t) + \gamma Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)\big)$

Policy improvement: Greedy policy improvement?

Example of Greedy Action Selection

UC San Diego

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

There are two doors in front of you.

- You open the left door and get reward 0 Q(left) = 0
- You open the right door and get reward +1 Q(right) = 1
- You open the right door and get reward +3 Q(right) = 2
- You open the right door and get reward +2 Q(right) = +2

Are you sure you've chosen the best door?

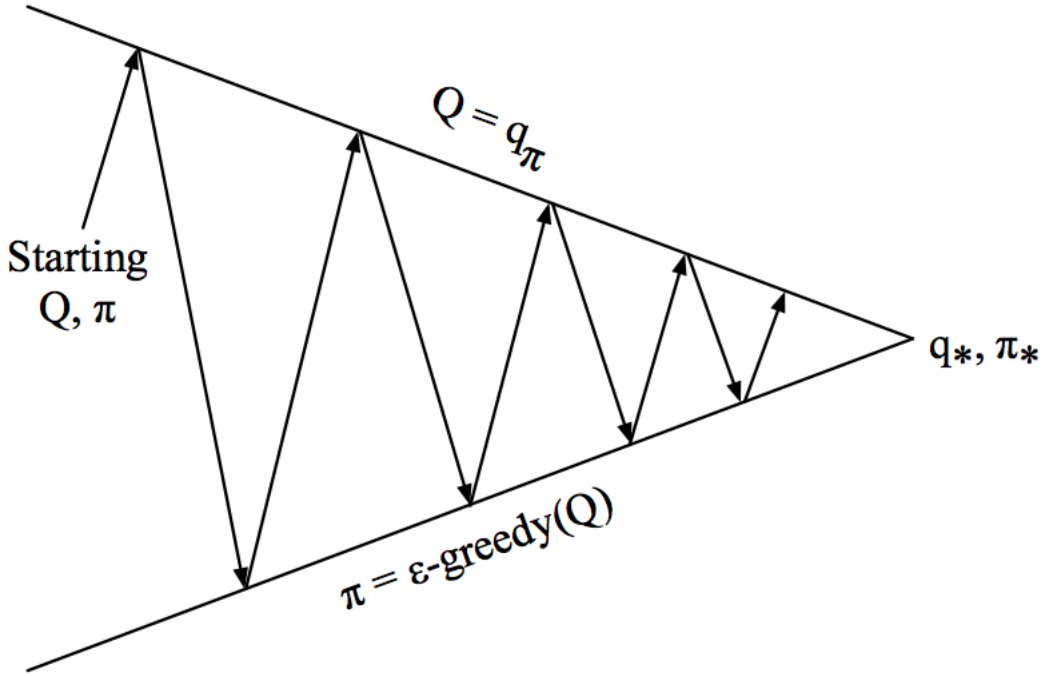
ϵ-Greedy Exploration

UC San Diego

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability
- With probability $1 - \varepsilon$, choose the greedy action
- With probability ε, choose an action at random

Model-free Policy Iteration

UC San Diego



Policy evaluation: Given a policy π, find out Q_π (can use MC or TD)

Policy improvement: ϵ-greedy policy improvement

Sarsa: On-Policy Action-Value Function Learning

UC San Diego

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$

Repeat (for each episode):

  Initialize S

  Choose A from S using policy derived from Q (e.g., ε-greedy)

  Repeat (for each step of episode):

    Take action A, observe R, S'

    Choose A' from S' using policy derived from Q (e.g., ε-greedy)

    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$

    $S \leftarrow S'$; $A \leftarrow A'$;

  until S is terminal

Q-Learning: Off-Policy Action-Value Function Learning

UC San Diego

- The **behavior policy** is ϵ-greedy, with respect to current $Q(s, a)$

- The TD target for Q learning is:

- Corresponds to a **target policy** is greedy, with respect to current $Q(s, a)$

Q-Learning: Off-Policy Action-Value Function Learning

UC San Diego

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$;
  until S is terminal

On-Policy vs Off-Policy

UC San Diego

- **On-policy Learning**: the behavior policy used to generate samples is the same as the target policy
  - For example, in Sarsa, both the behavior and target policy of agent are ϵ-greedy

- **Off-policy Learning:** the behavior and the target policy are different. The target policy is learned independently of the actions chosen for exploring the environment (i.e., behavior policy).
  - For example, in Q-learning, the behavior policy is ϵ-greedy while the target policy is a greedy action policy (such that it directly estimates Q^*)

Deep Q-Learning

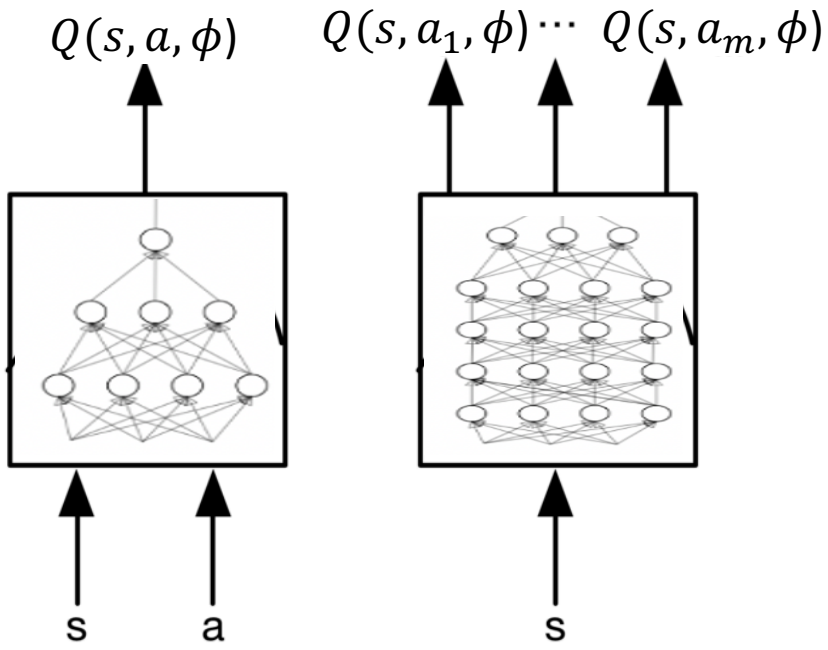
Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." Nature 518, no. 7540 (2015): 529-533.

Q Networks

UC San Diego

- Represent Q function by a neural network (Q-network), with weights ϕ

$$\widehat{Q_\phi}(s,a) \approx Q_\pi(s,a)$$



Deep Q-Learning

UC San Diego

- Define optimization objective function: MSE between Q-target and Q-network prediction

Set TD target: $\boldsymbol{y \leftarrow r(s,a) + \gamma \max_{a'} Q(s', a', \phi)}$

$$L(\phi) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}',\boldsymbol{r})}\left[\frac{1}{2}\left(\boldsymbol{y} - Q(s,a,\phi)\right)^2\right]$$

- Gradient of Q-learning:

$$\frac{\partial L(\phi)}{\partial \phi} = -\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}',\boldsymbol{r})}\left[\left(\boldsymbol{y} - Q(s,a,\phi)\right)\frac{\partial Q(s,a,\phi)}{\partial \phi}\right]$$

DQN

UC San Diego

- Naive Q-Learning with neural network is not stable (easily diverge in training)
  - Data is sequentially collected, non i.i.d.
  - Target value oscillates with consecutive changes to Q-values
- DQN provides a stable solution to deep value-based RL
- Use **experience replay**
  - Learn from data collected by all past policies → reduce correlations in data
- Freeze **target Q-network**
  - Avoid oscillations
  - Break correlations between Q-network and target

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." Nature 518, no. 7540 (2015): 529-533.

(1) Experience Replay

UC San Diego

- Break correlation in the data
- Procedure
  - Take an action a
  - Store sample (s, a, s', r) in memory
  - Sample randomly from memory
  - Optimize with the sampled mini-batch data

$$L(\phi) = \frac{1}{2}\mathbb{E}_{(s,a,s',r)}\left[\left(\underbrace{r(s,a) + \gamma \max_{a'} Q(s', a', \phi)}_{\text{target}} - Q(s, a, \phi)\right)^2\right] \approx \frac{1}{2B}\sum_{i=1}^{B}\left[\left(r_i + \gamma \max_{a'} Q(s_i', a', \phi) - Q(s_i, a_i, \phi)\right)^2\right]$$

(2) Fixed Q-targets

UC San Diego

- Freeze Q-target network to avoid oscillations
- Procedure
  - Compute Q-targets using old weights

$$r + \gamma \max_{a'} Q(s', a', \boldsymbol{\phi^-})$$

  - Optimize the **parameter in Q network** $Q(s, a, \phi)$ by taking one gradient descent step to minimize the following loss:

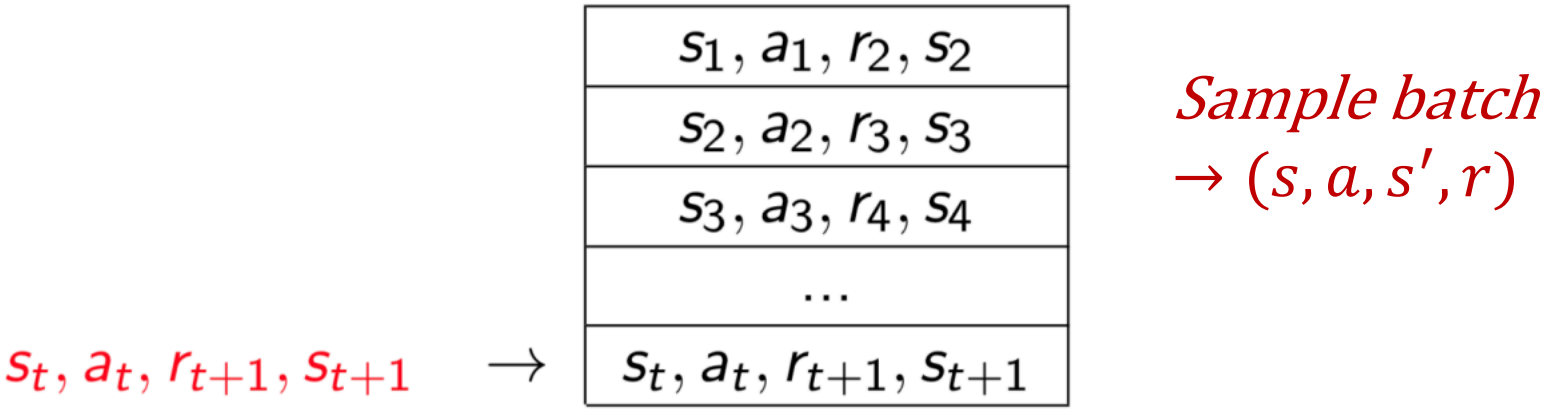
$$L(\phi) = \frac{1}{2B} \sum_{i=1}^{B} \left[\left(\boldsymbol{r_i + \gamma \max_{a'} Q(s_i', a', \phi^-)} - Q(s_i, a_i, \phi) \right)^2 \right]$$

  - Periodic update the target Q network weights: $\boldsymbol{\phi^- \leftarrow \phi}$

Deep Q-Networks

UC San Diego

- Sample data from replay buffer



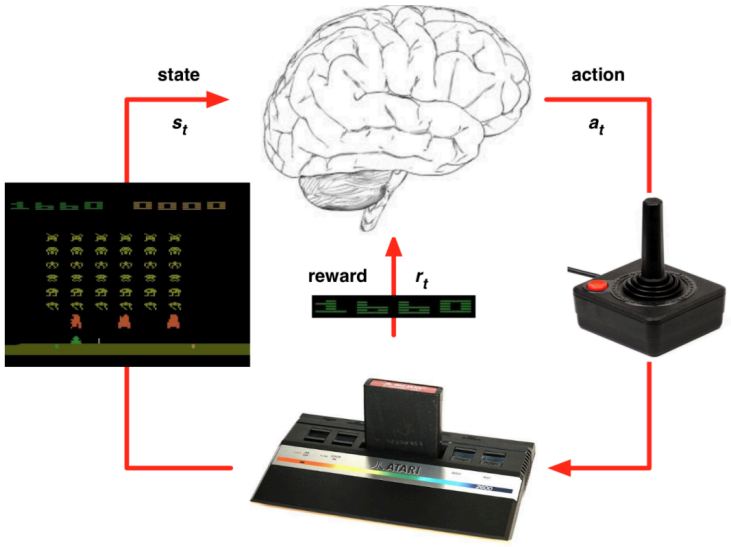
- Perform a gradient descent step on ϕ to minimize the following loss:

$$L(\phi) = \frac{1}{2B}\sum_{i=1}^{B}\left[\left(r_i + \gamma \max_{a'} Q(s_i', a', \phi^-) - Q(s_i, a_i, \phi)\right)^2\right]$$

$$\phi \leftarrow \phi - \alpha \nabla_\phi L$$

DQN in Atari

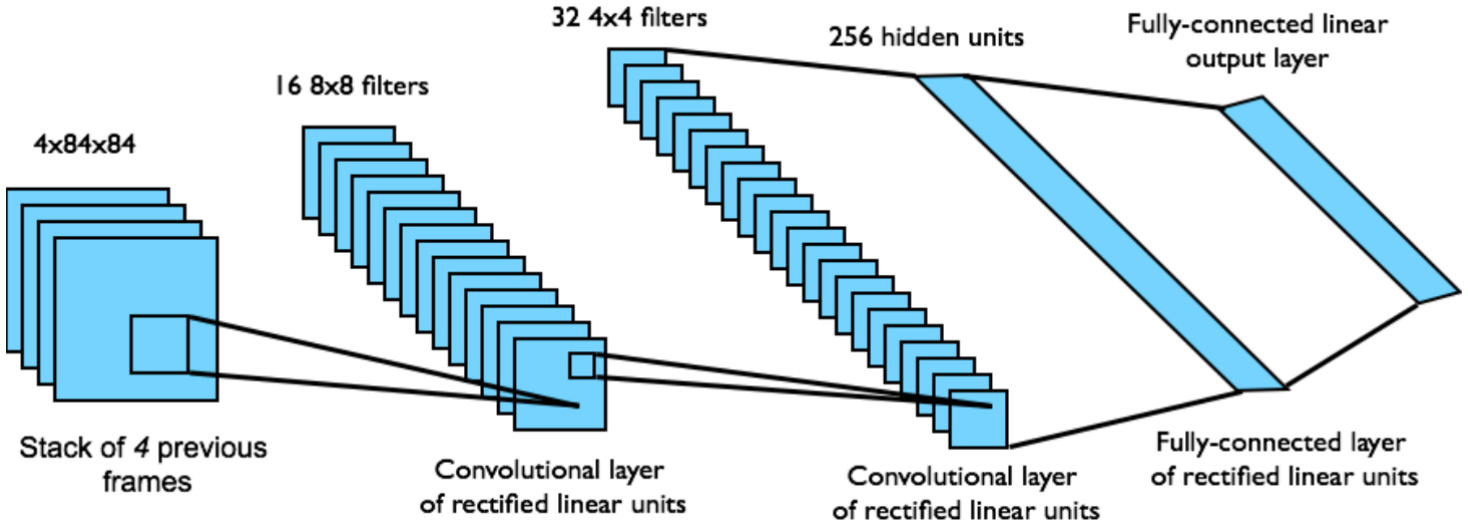
UC San Diego



DQN in Atari

UC San Diego

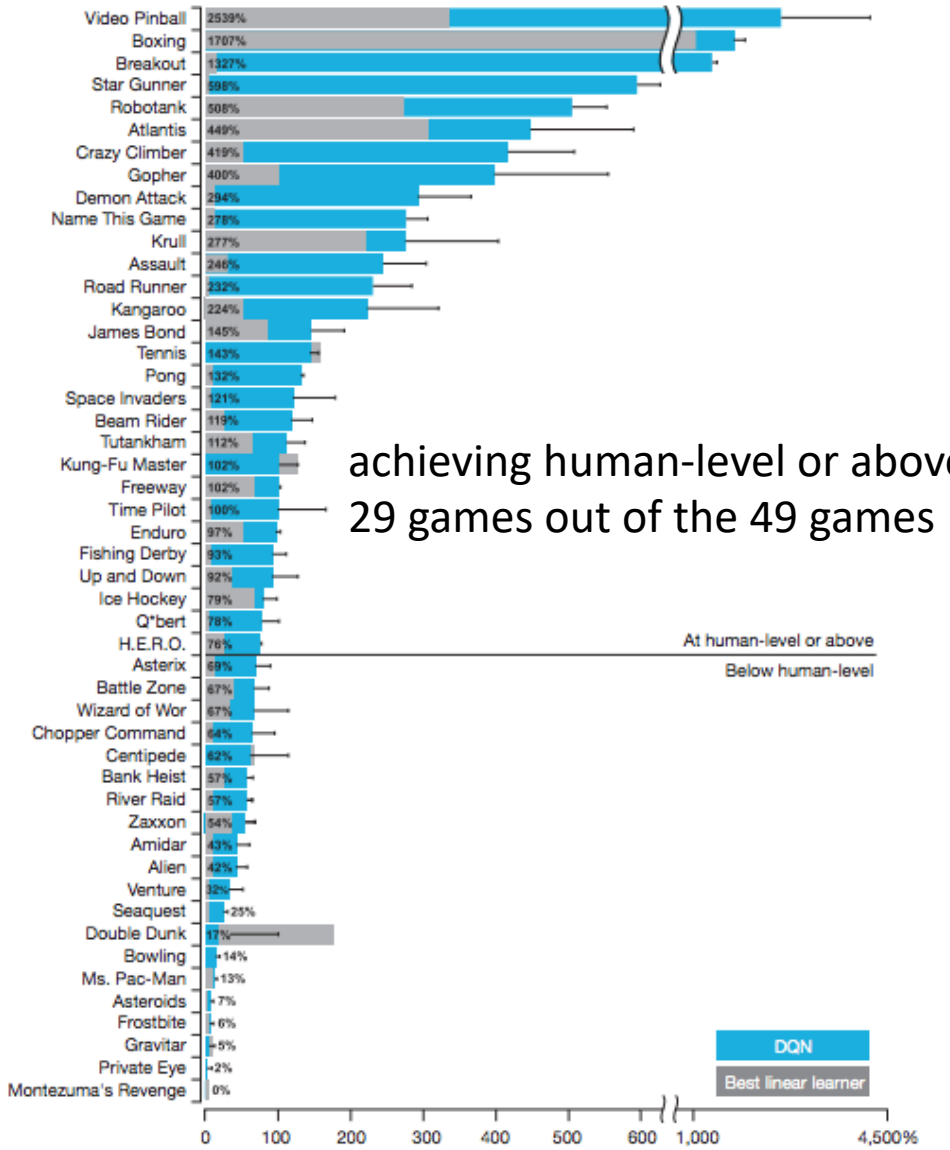
- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is defined as the change in score for that step



Network architecture and hyperparameters fixed across all games

Atari Results

UC San Diego

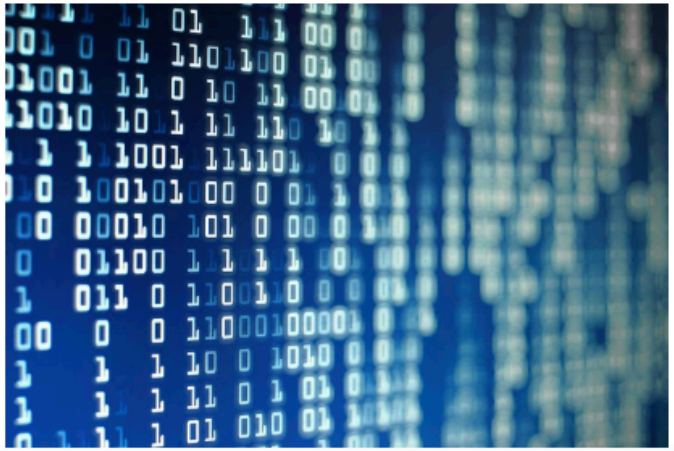


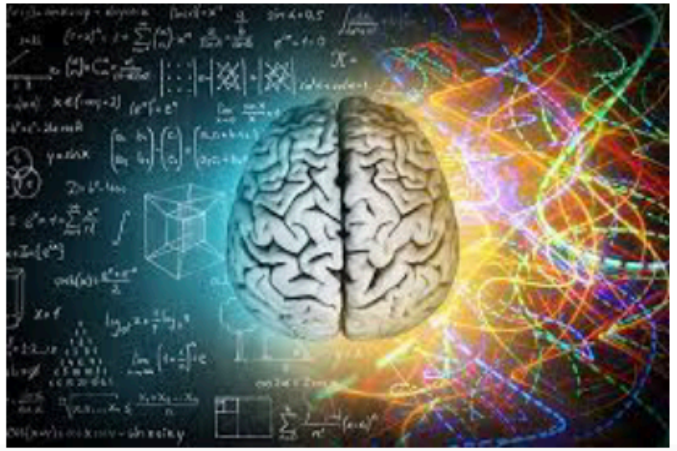
achieving human-level or above in 29 games out of the 49 games

Closing Remark

UC San Diego

- Research in deep learning skyrockets as people join and new discoveries are made. New methods and discoveries make significant contributions to supervised learning, unsupervised learning, reinforcement learning, etc.
- Success depends on: Data + Computation + Algorithm!
- You will learn these tools (and potentially innovate new tools) in this class and being able to apply them to some physical applications that excite you!





Everyone should be proud of yourself - you did an amazing job in learning!