



# 10-601 Introduction to Machine Learning

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

# Reinforcement Learning: Q-Learning + Deep RL

Matt Gormley  
Lecture 17  
Mar. 29, 2021

# Reminders

- **Homework 5: Neural Networks**
  - **Out: Thu, Mar. 18**
  - **Due: Mon, Mar. 29 at 11:59pm**
- **Homework 6: Deep RL**
  - **Out: Mon, Mar. 29**
  - **Due: Wed, Apr. 07 at 11:59pm**

# Q-LEARNING

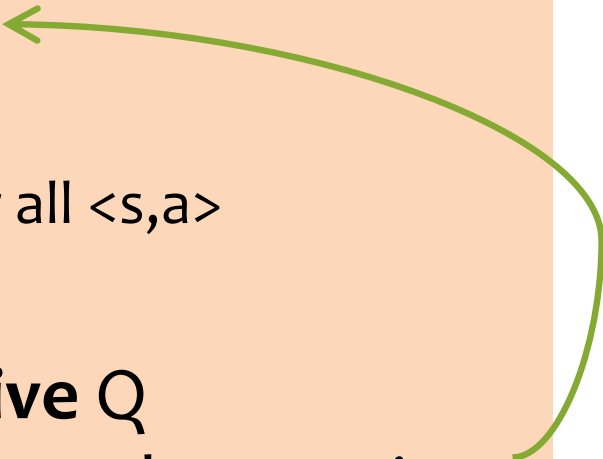
# Q-Learning

## *Whiteboard*

- Q-Learning Algorithm
  - Case 1: Deterministic Environment
  - Case 2: Nondeterministic Environment
- $\epsilon$ -greedy Strategy

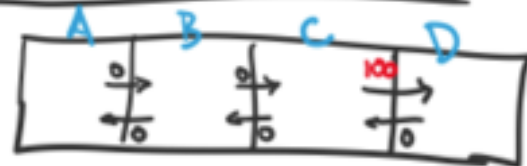
# Q-Learning

## Remarks

- **Q converges to  $Q^*$**  with probability 1.0, assuming...
    1. each  $\langle s, a \rangle$  is visited infinitely often
    2.  $0 \leq \gamma < 1$
    3. rewards are bounded  $|R(s,a)| < \beta$ , for all  $\langle s,a \rangle$
    4. initial Q values are finite
  - Q-Learning is **exploration insensitive Q**  
 $\Rightarrow$  any state visitation strategy will work assuming
  - May take **many iterations** to converge in practice
- 

# Reordering Experiences

Ex: Easiest Maze Ever!



arrows we  $R(s_A)$

$$\gamma = 0.9$$

$$S = \{A, B, C, D\}$$

$$A = \{E, W\}$$

$$Q(s_A) = 0 \text{ at start}$$

① suppose we visit

$i$	$s$	$a$	$r$	$s'$
-----	-----	-----	-----	------

1	A	E	0	B
---	---	---	---	---

$$Q(A, E) = 0$$

2	B	E	0	C
---	---	---	---	---

$$Q(B, E) = 0$$

3	C	E	100	D
---	---	---	-----	---

$$Q(C, E) = 100$$

② suppose we visit in reverse order

$i$	$s$	$a$	$r$	$s'$
-----	-----	-----	-----	------

1	C	E	100	D
---	---	---	-----	---

$$Q(C, E) = 100$$

2	B	E	0	C
---	---	---	---	---

$$Q(B, E) = 90$$

3	A	E	0	B
---	---	---	---	---

$$Q(A, E) = 81$$

# Designing State Spaces

**Q:** Do we have to retrain our RL agent every time we change our state space?

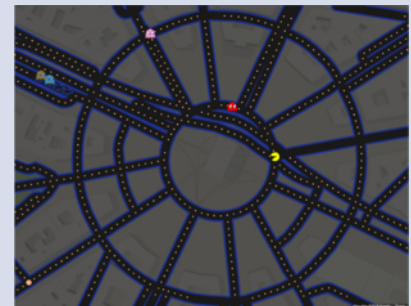
**A:** Yes. But whether your state space changes from one setting to another is determined by your design of the state representation.

Two examples:

- State Space A:  $\langle x, y \rangle$  position on map  
e.g.  $s_t = \langle 74, 152 \rangle$
- State Space B: window of pixel colors  
centered at current Pac Man location

e.g.  $s_t =$

0	1	0
0	0	0
1	1	1

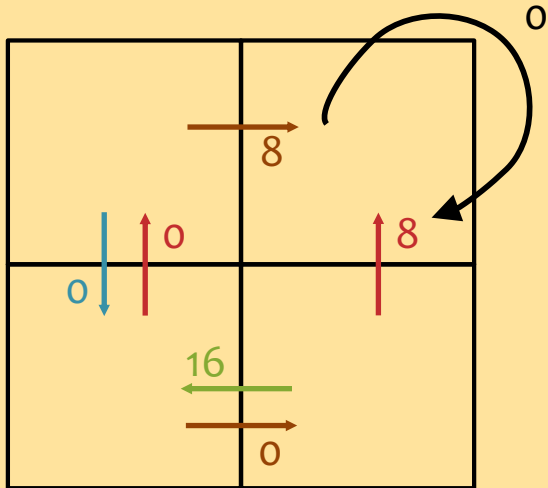


# Q-Learning

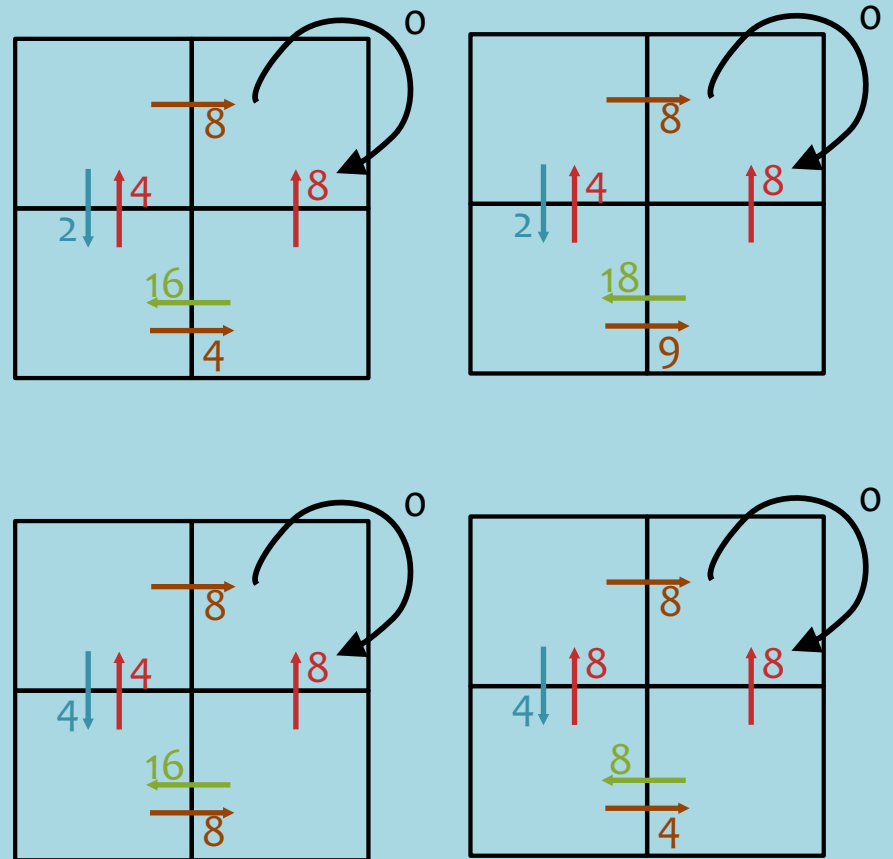
## Question:

For the  $R(s,a)$  values shown on the arrows below, which are the corresponding  $Q^*(s,a)$  values?

Assume discount factor = 0.5.



## Answer:





# DEEP RL EXAMPLES

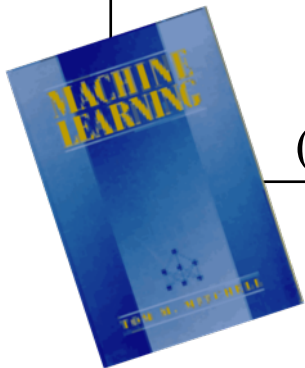
# TD Gammon → Alpha Go

## Learning to beat the masters at board games

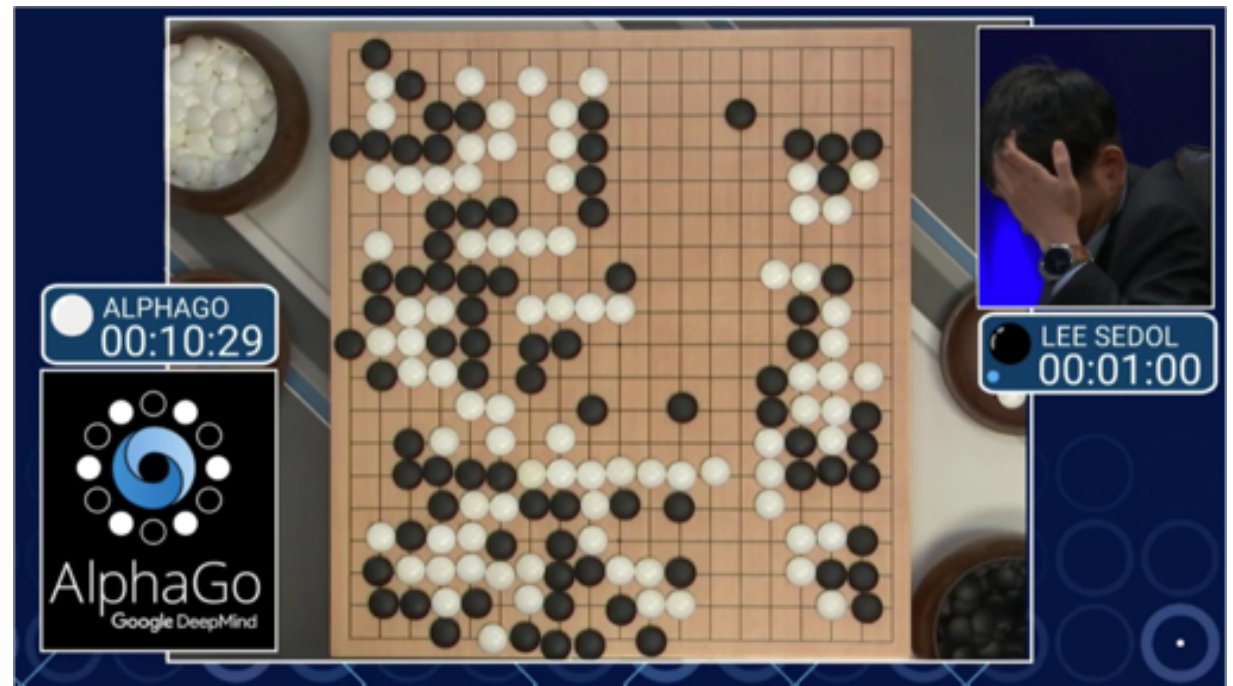
THEN

“...the world’s top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself...”

(Mitchell, 1997)

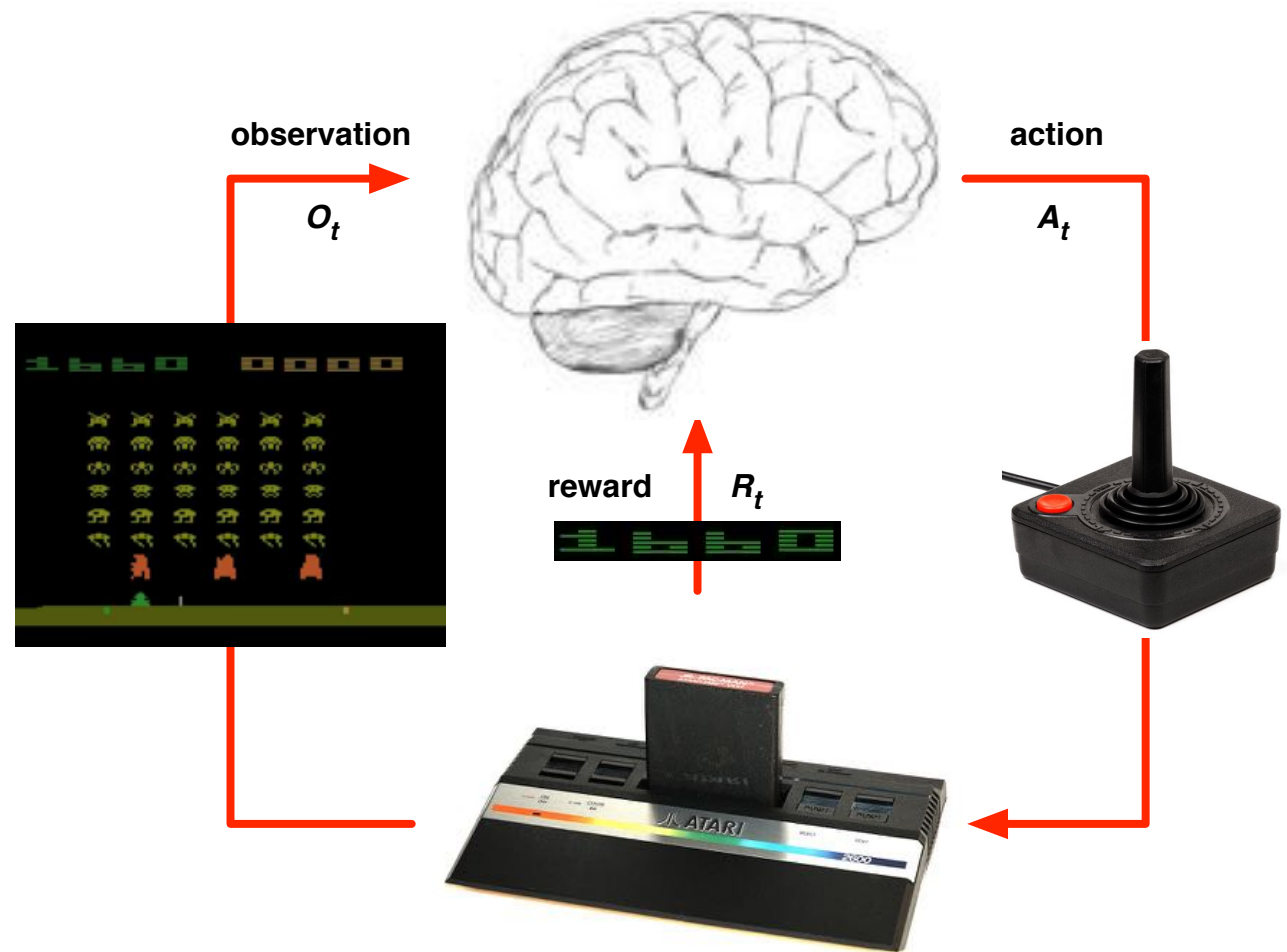


NOW



# Playing Atari with Deep RL

- Setup: RL system observes the pixels on the screen
- It receives rewards as the game score
- Actions decide how to move the joystick / buttons



# Playing Atari with Deep RL



Figure 1: Screen shots from five Atari 2600 Games: (Left-to-right) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

## Videos:

- Atari Breakout:

<https://www.youtube.com/watch?v=V1eYniJoRnk>

- Space Invaders:

<https://www.youtube.com/watch?v=ePvoFs9cGgU>

# Playing Atari with Deep RL



Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

	<b>B. Rider</b>	<b>Breakout</b>	<b>Enduro</b>	<b>Pong</b>	<b>Q*bert</b>	<b>Seaquest</b>	<b>S. Invaders</b>
<b>Random</b>	354	1.2	0	-20.4	157	110	179
<b>Sarsa [3]</b>	996	5.2	129	-19	614	665	271
<b>Contingency [4]</b>	1743	6	159	-17	960	723	268
<b>DQN</b>	<b>4092</b>	<b>168</b>	<b>470</b>	<b>20</b>	<b>1952</b>	<b>1705</b>	<b>581</b>
<b>Human</b>	7456	31	368	-3	18900	28010	3690
<b>HNeat Best [8]</b>	3616	52	106	19	1800	920	<b>1720</b>
<b>HNeat Pixel [8]</b>	1332	4	91	-16	1325	800	1145
<b>DQN Best</b>	<b>5184</b>	<b>225</b>	<b>661</b>	<b>21</b>	<b>4500</b>	<b>1740</b>	1075

Table 1: The upper table compares average total reward for various learning methods by running an  $\epsilon$ -greedy policy with  $\epsilon = 0.05$  for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an  $\epsilon$ -greedy policy with  $\epsilon = 0.05$ .

# Deep Q-Learning

**Question:** *What if our state space  $S$  is too large to represent with a table?*

**Examples:**

- $s_t$  = pixels of a video game
- $s_t$  = continuous values of a sensors in a manufacturing robot
- $s_t$  = sensor output from a self-driving car

**Answer:** Use a parametric function to approximate the table entries

**Key Idea:**

1. Use a neural network  $Q(s,a; \theta)$  to approximate  $Q^*(s,a)$
2. Learn the parameters  $\theta$  via SGD with training examples  $\langle s_t, a_t, r_t, s_{t+1} \rangle$

# Deep Q-Learning

## *Whiteboard*

- Strawman loss function (i.e. what we cannot compute)
- Approximating the Q function with a neural network
- Approximating the Q function with a linear model
- Deep Q-Learning
- function approximators  
( $\langle \text{state}, \text{action}_i \rangle \rightarrow \text{q-value}$   
**vs.**  
state  $\rightarrow$  all action q-values)

# Experience Replay

- **Problems** with online updates for Deep Q-learning:
  - not i.i.d. as SGD would assume
  - quickly forget rare experiences that might later be useful to learn from
- **Uniform Experience Replay** (Lin, 1992):
  - Keep a *replay memory*  $D = \{e_1, e_2, \dots, e_N\}$  of  $N$  most recent experiences  $e_t = \langle s_t, a_t, r_t, s_{t+1} \rangle$
  - Alternate two steps:
    1. Repeat  $T$  times: randomly sample  $e_i$  from  $D$  and apply a Q-Learning update to  $e_i$
    2. Agent selects an action using epsilon greedy policy to receive new experience that is added to  $D$
- **Prioritized Experience Replay** (Schaul et al, 2016)
  - similar to Uniform ER, but sample so as to prioritize experiences with high error



# Alpha Go

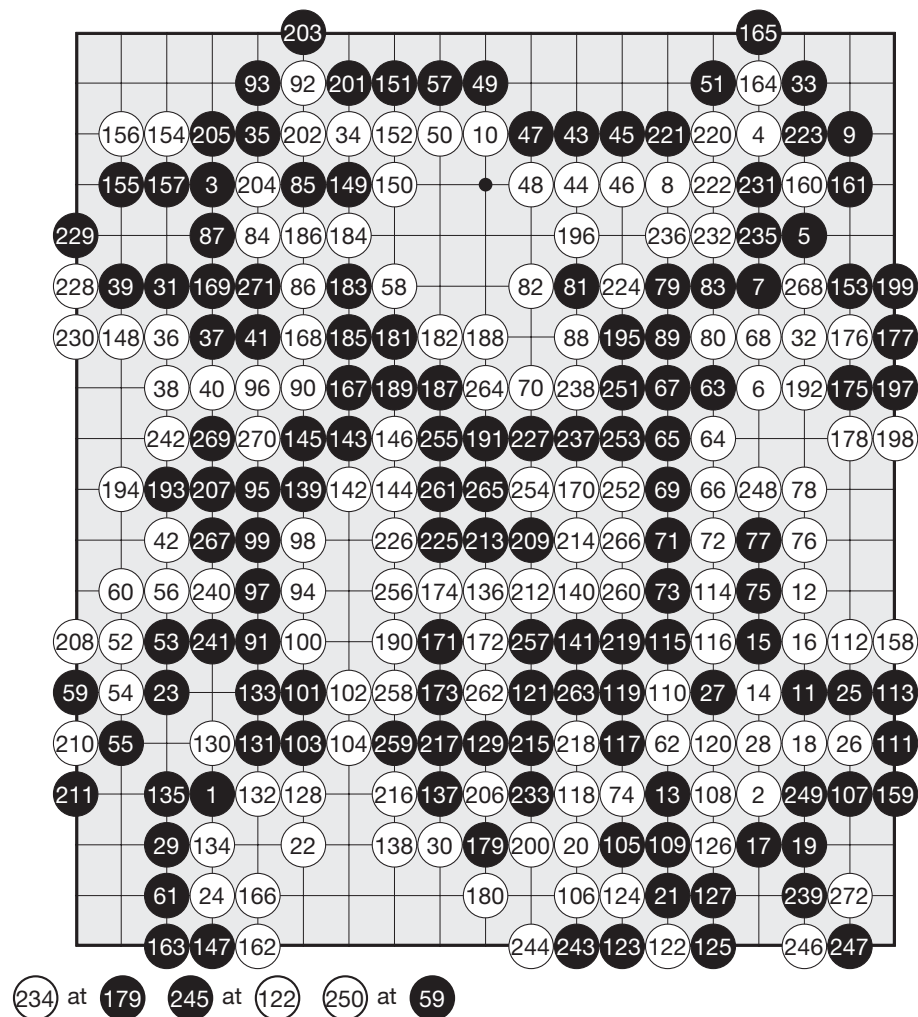
## Game of Go (圍棋)

- 19x19 board
- Players alternately play black/white stones
- Goal is to fully encircle the largest region on the board
- Simple rules, but extremely complex game play

Game 1

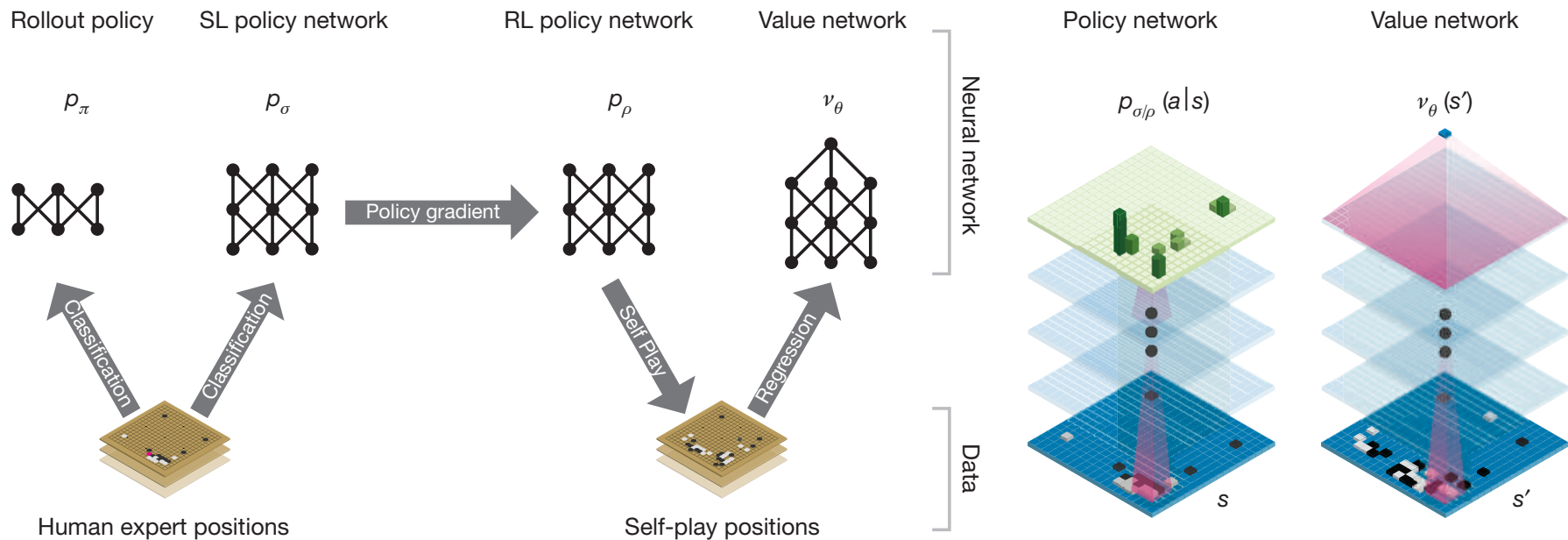
Fan Hui (Black), AlphaGo (White)

AlphaGo wins by 2.5 points



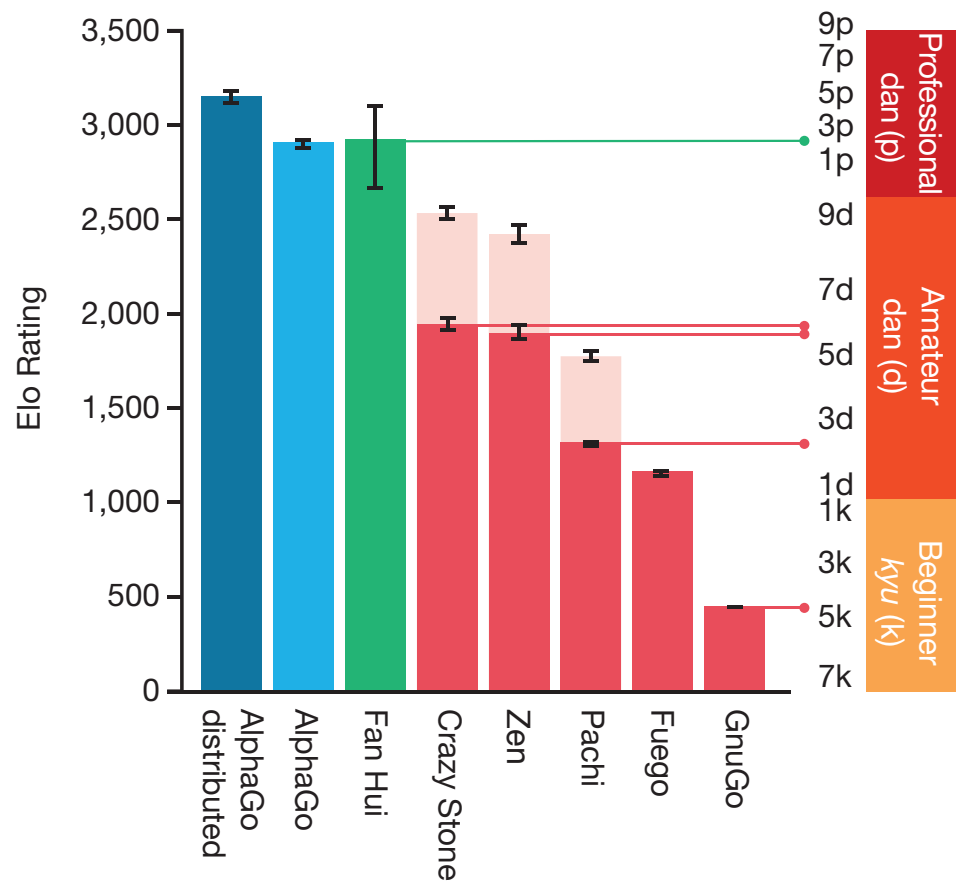
# Alpha Go

- State space is too large to represent explicitly since # of sequences of moves is  $O(b^d)$ 
  - Go:  $b=250$  and  $d=150$
  - Chess:  $b=35$  and  $d=80$
- Key idea:
  - Define a neural network to approximate the value function
  - Train by policy gradient



# Alpha Go

- Results of a tournament
- From Silver et al. (2016): “a 230 point gap corresponds to a 79% probability of winning”



# Learning Objectives

## Reinforcement Learning: Q-Learning

*You should be able to...*

1. Apply Q-Learning to a real-world environment
2. Implement Q-learning
3. Identify the conditions under which the Q-learning algorithm will converge to the true value function
4. Adapt Q-learning to Deep Q-learning by employing a neural network approximation to the Q function
5. Describe the connection between Deep Q-Learning and regression

# ML Big Picture

## Learning Paradigms:

*What data is available and when? What form of prediction?*

- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

## Theoretical Foundations:

*What principles guide learning?*

- probabilistic
- information theoretic
- evolutionary search
- ML as optimization

## Problem Formulation:

*What is the structure of our output prediction?*

boolean	Binary Classification
categorical	Multiclass Classification
ordinal	Ordinal Classification
real	Regression
ordering	Ranking
multiple discrete	Structured Prediction
multiple continuous	(e.g. dynamical systems)
both discrete & cont.	(e.g. mixed graphical models)

## Facets of Building ML Systems:

*How to build systems that are robust, efficient, adaptive, effective?*

1. Data prep
2. Model selection
3. Training (optimization / search)
4. Hyperparameter tuning on validation data
5. (Blind) Assessment on test data

## Big Ideas in ML:

*Which are the ideas driving development of the field?*

- inductive bias
- generalization / overfitting
- bias-variance decomposition
- generative vs. discriminative
- deep nets, graphical models
- PAC learning
- distant rewards

## Application Areas

*Key challenges?*

NLP, Speech, Computer Vision, Robotics, Medicine, Search

# Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \dots\}$
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \dots\}$