# Reinforcement Learning

Amelie Cameron

# Markov Decision Process

Value Iteration Agent takes an MDP on construction which is defined by:

- S: a finite set of states
- A: a finite set of actions
- T: a transition function T(s, a, s')

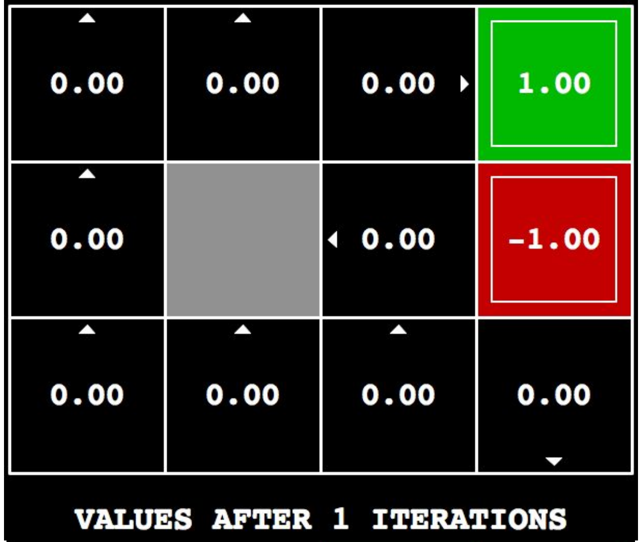
  Probability that a from s leads to s' i.e., P(s'| s, a)
- R: a reward function R(s, a, s')
- $\gamma$: a discount factor, value between 0 and 1

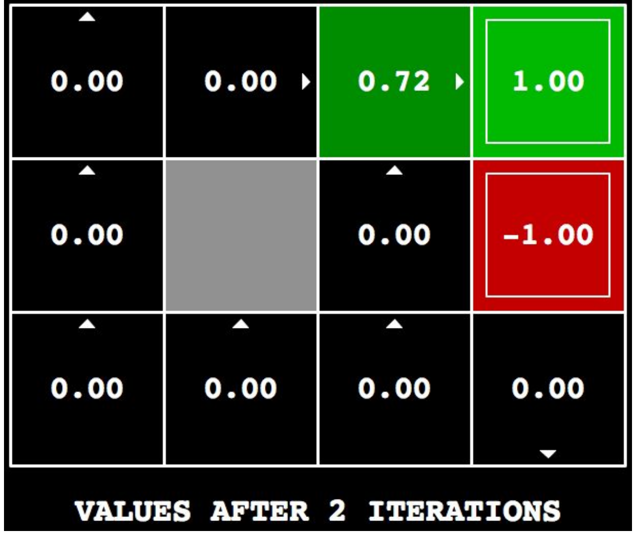
# Value Iteration

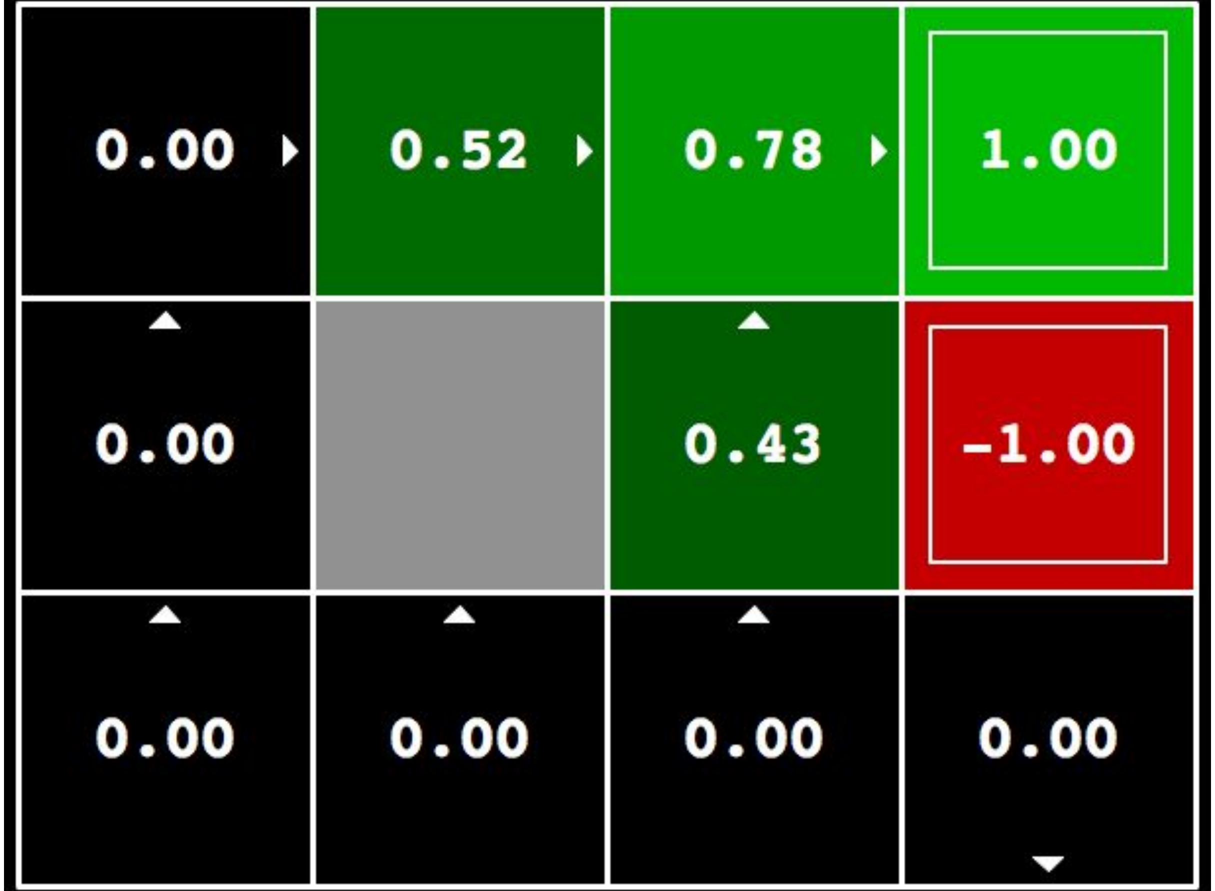
- $V(s) = \left(R(s) + \gamma \max_{a} \sum_{s'} T(s, a, s')\, V(s')\right)$

  where
  - $\gamma = 0.9$
  - reward = 0 for non-terminal states
    1 or -1 for terminal states
  - T(s,a,s') represents probability that action a from state s leads to s' P(s'| s, a)
- Update values based on the best next state
- Account for noise (probability of moving in an unintended direction = 0.2)

$$V([2,2]) = [R([2,2]) + 0.9 * P([2,2], \text{right}, s') * V(s')]$$
$$= (0.0 + 0.9 * 0.8 * 1.0)$$
$$= 0.72$$

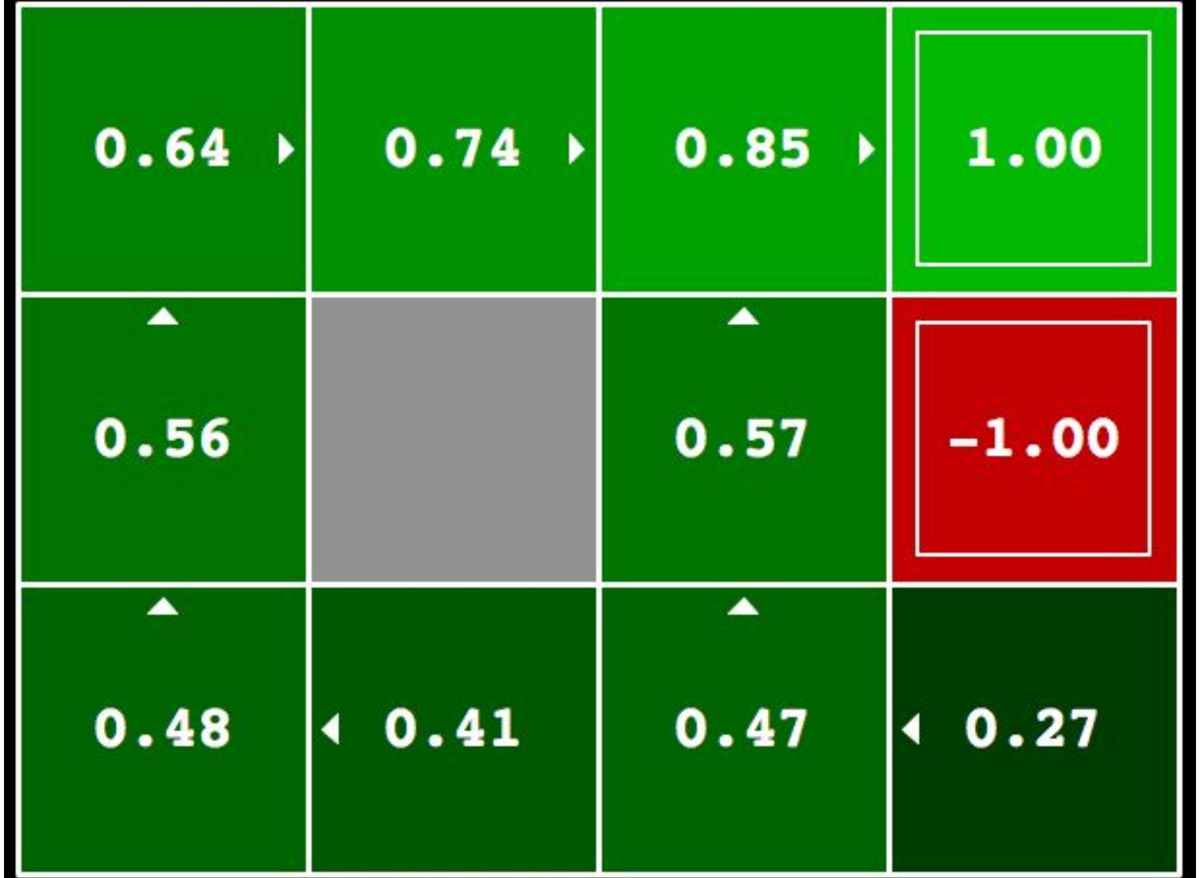





VALUES AFTER 3 ITERATIONS


0.51
0.72
0.84
1.00
0.27
0.55
-1.00
0.00
0.22
0.37
0.13
VALUES AFTER 5 ITERATIONS




VALUES AFTER 10 ITERATIONS

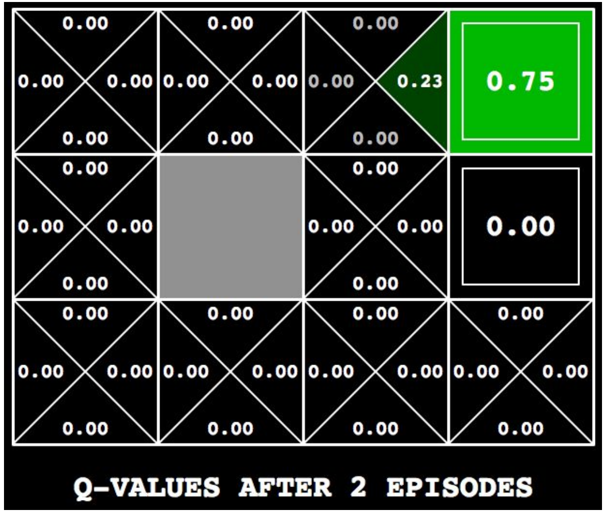

0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

# Q Learning

- Determines the sum of expected future rewards when the agent performs the action a in the state s, continuing to act optimally.

- The discount factor differentiates the rewards far away from the actual state, i.e. higher value to the closest rewards. The function Q defines the sum of the discounted future rewards.

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

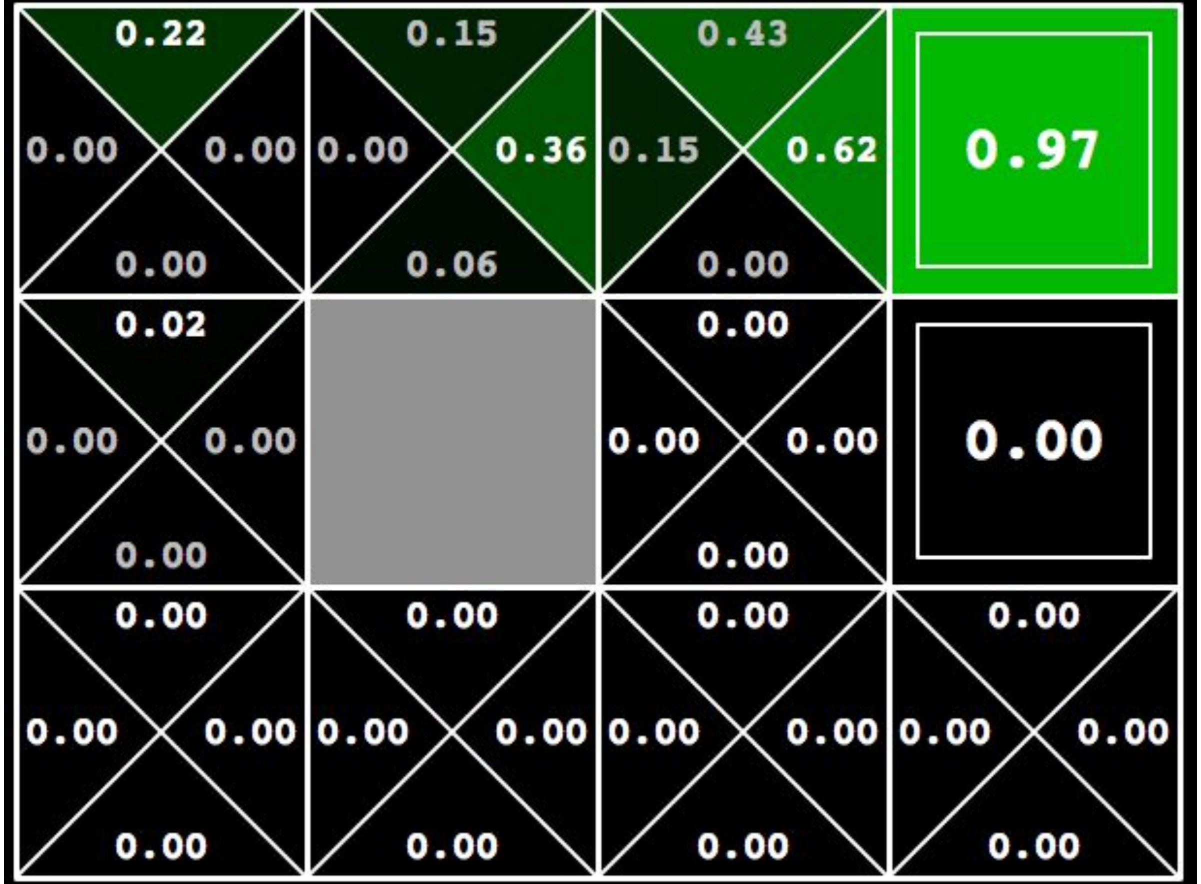

# Q-Learning

- Q value is initialized arbitrarily
- For each iteration, the agent selects an action a, observes a reward R, and reaches a new state s' and Q value is updated

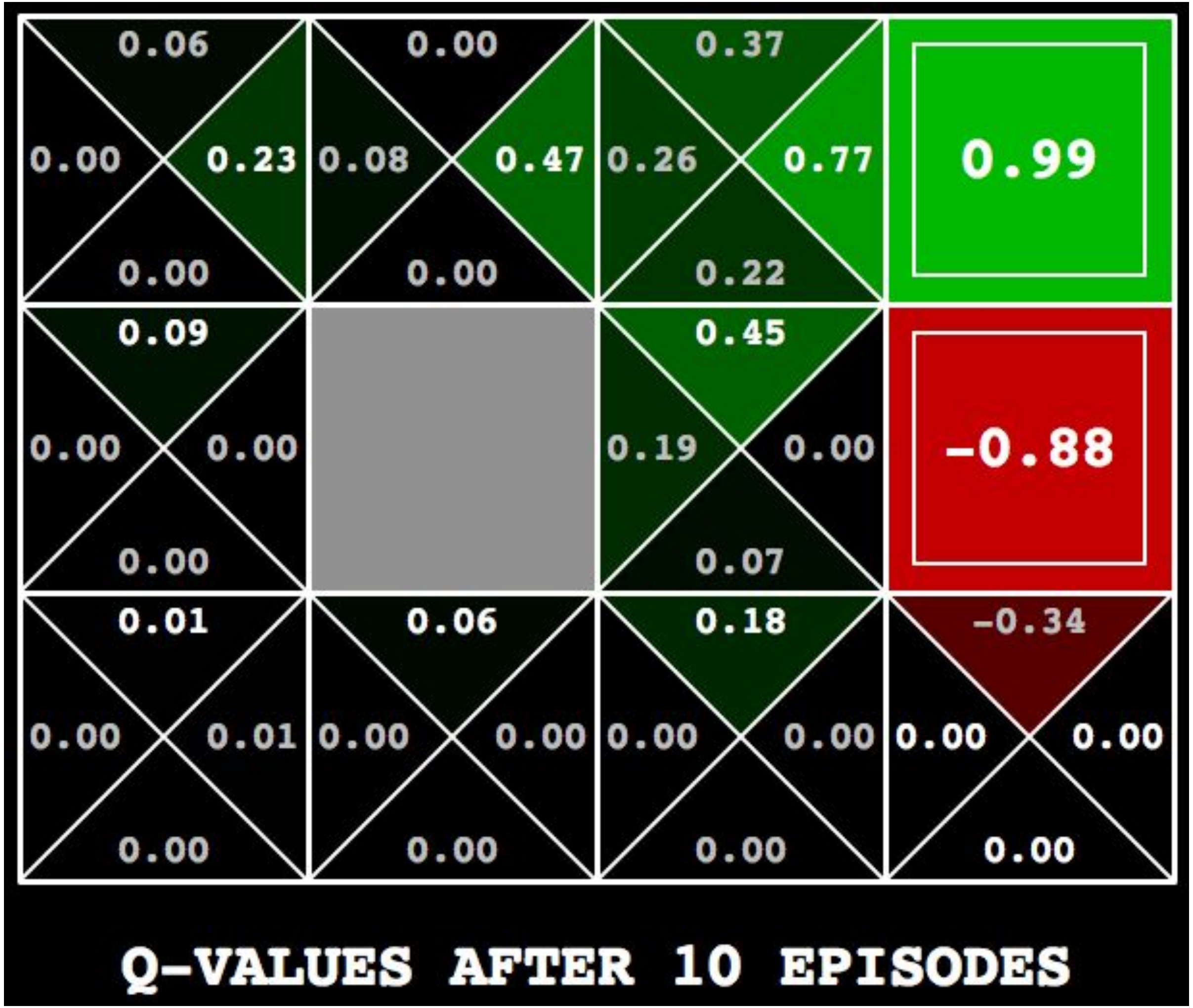
- $\alpha$ represents learning rate ( $0 < \alpha <= 1$)
  In qLearningAgents, alpha is 0.5

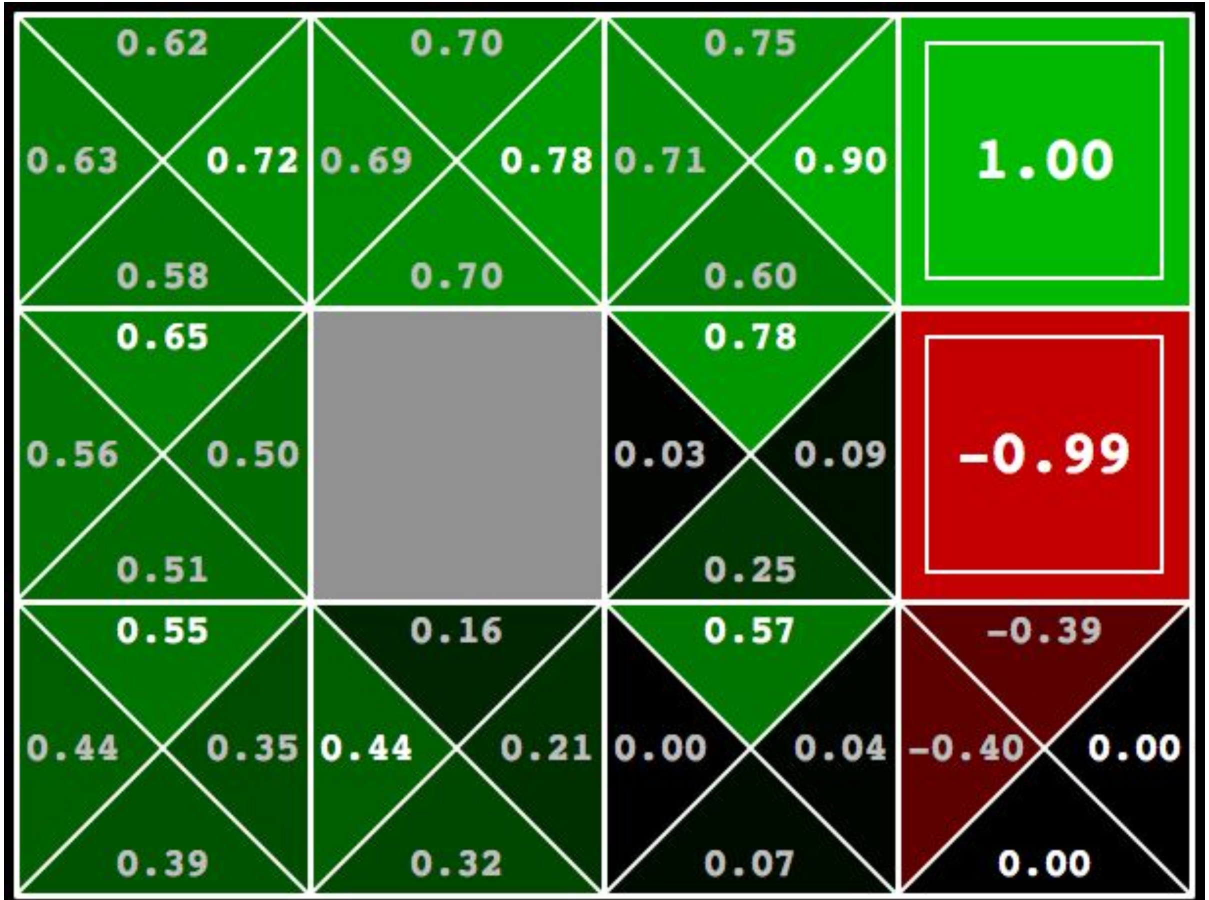
- $Q(s,a) \leftarrow (1-\alpha) * Q(s,a) + \alpha\,[R(s) + \gamma * \max_{a} Q(s', a)]$

$$= (1 - 0.5) * 0 + 0.5( 0 + 0.9 * 0.5)$$
$$= 0.23$$



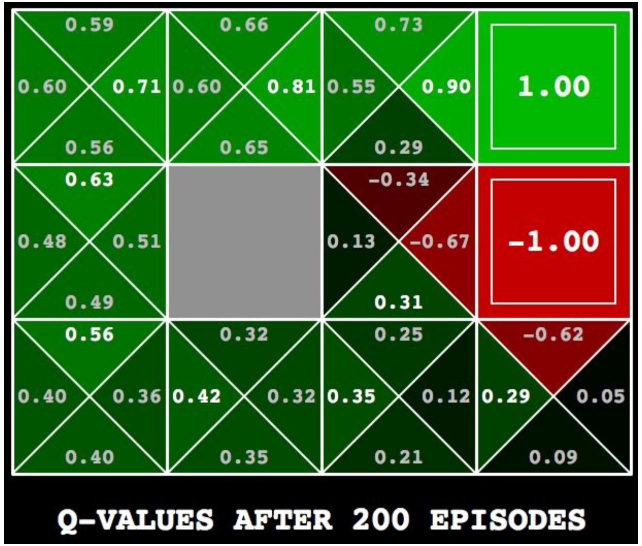

**Q-VALUES AFTER 5 EPISODES**

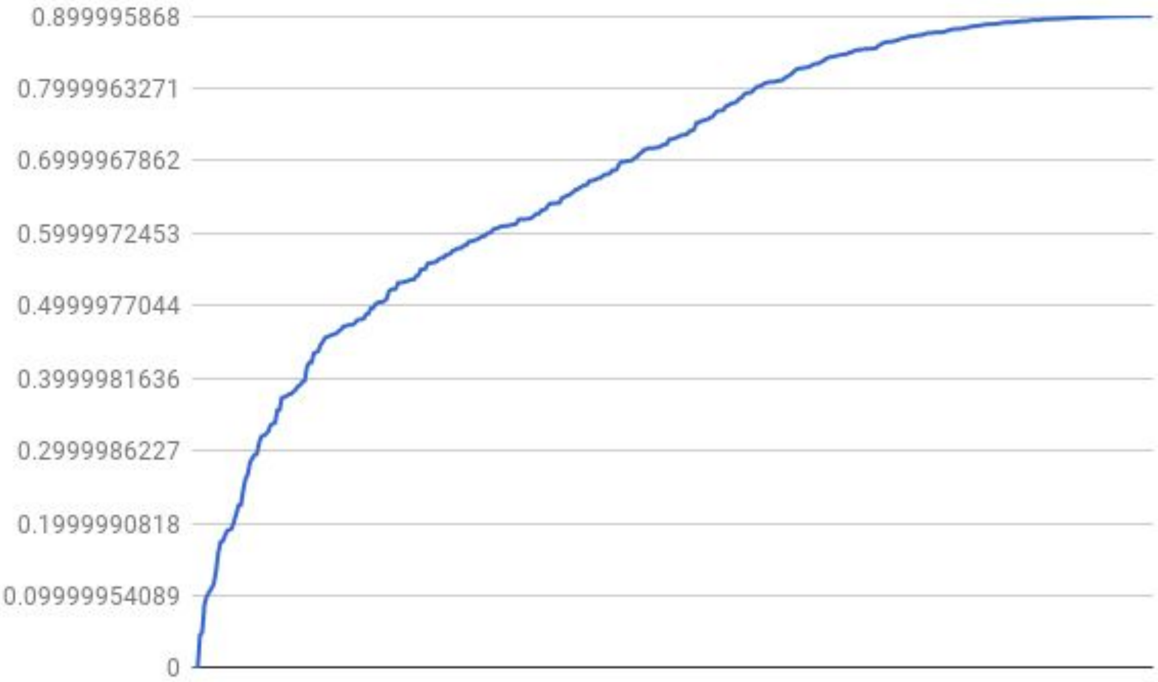

0.06
0.00
0.23
0.00
0.00
0.08
0.47
0.00
0.37
0.26
0.77
0.22
0.99
0.09
0.00
0.00
0.00
0.45
0.19
0.00
0.07
-0.88
0.01
0.00
0.01
0.00
0.06
0.00
0.00
0.00
0.18
0.00
0.00
0.00
-0.34
0.00
0.00
0.00
Q-VALUES AFTER 10 EPISODES



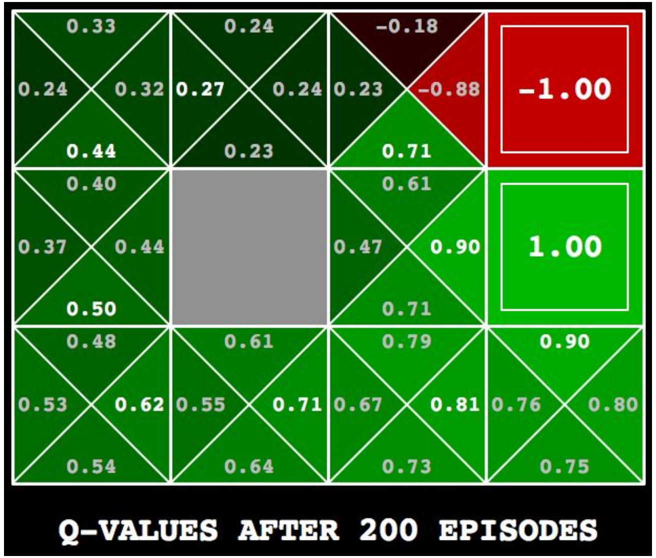

Q-VALUES AFTER 100 EPISODES

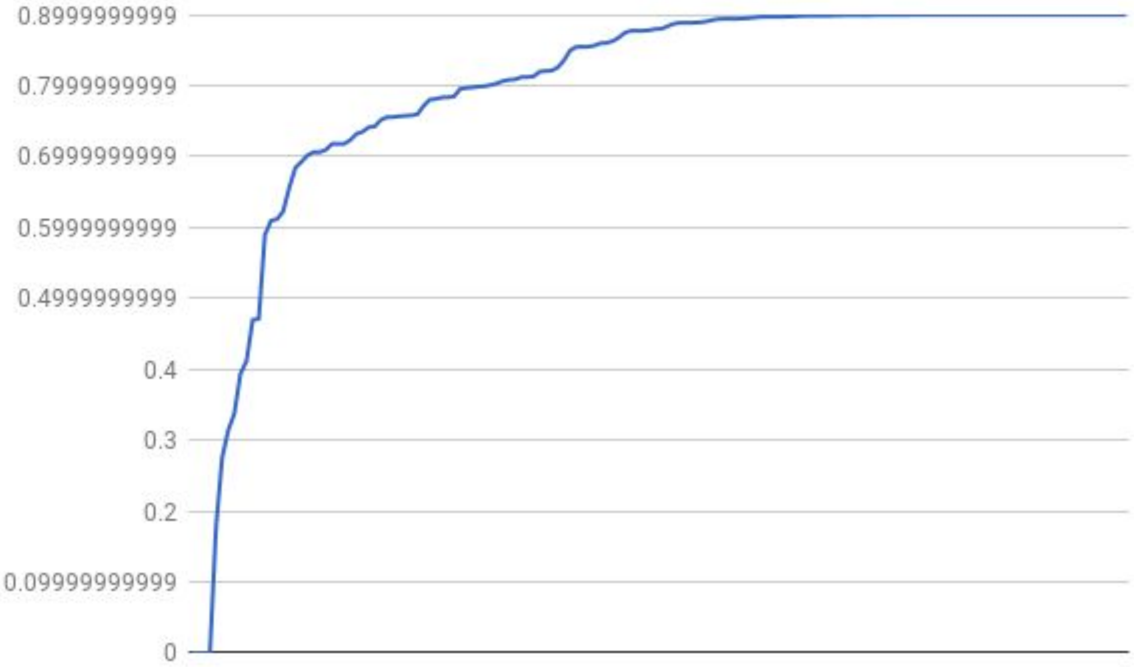
# Further Q Value Testing



# Further Q Value Testing



# Application of Reinforcement Learning

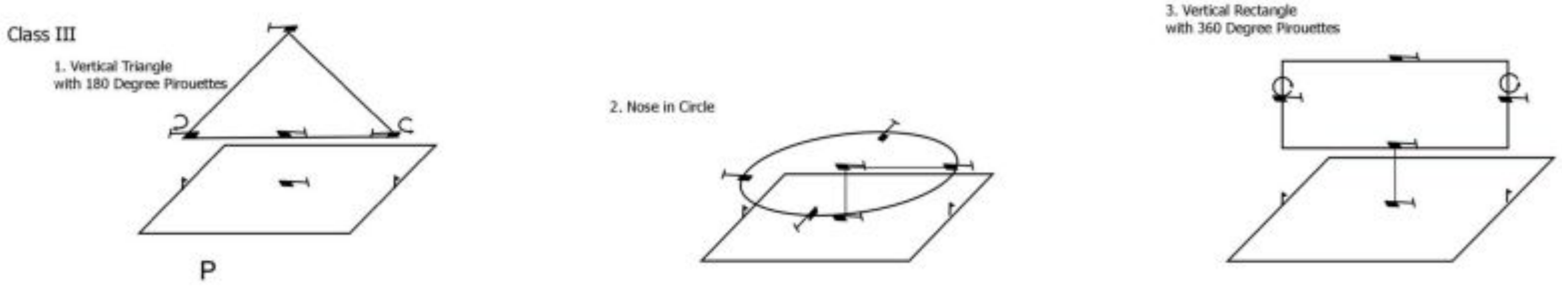
- Autonomous helicopter
- Yamaha R-50 helicopter (approximately 3.6m long)
- An onboard navigation computer runs a Kalman filter which integrates the sensor information from the GPS, Inertial Navigation System, and a digital compass
- Reports the estimates of the helicopter's position (x, y, z), orientation, velocity, and angular velocities to the ground station.
- Trained helicopter to fly in place and to perform learned maneuvers
- Used a Markov Decision Process called PEGASUS

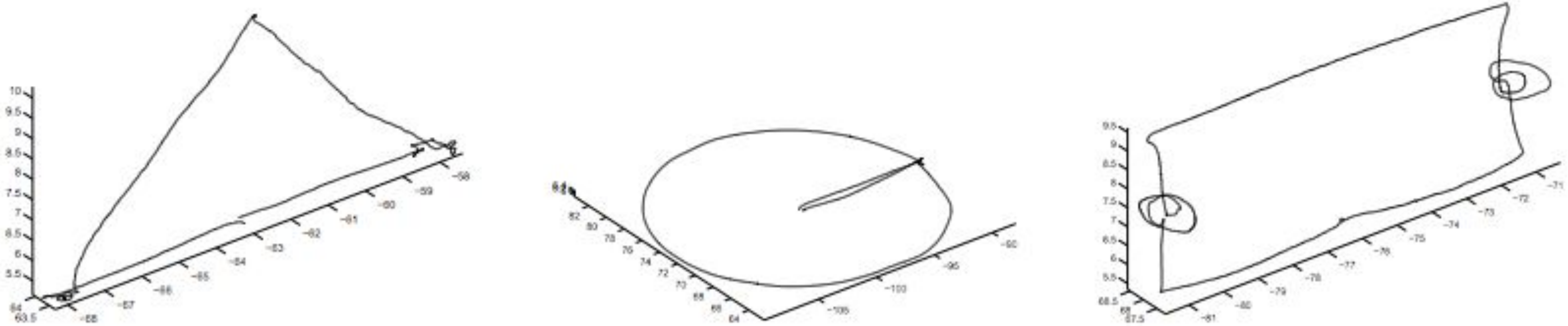
https://people.eecs.berkeley.edu/~jordan/paper/ng-etal03.pdf

By Andrew Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry

# Application of Reinforcement Learning

Maneuver diagrams



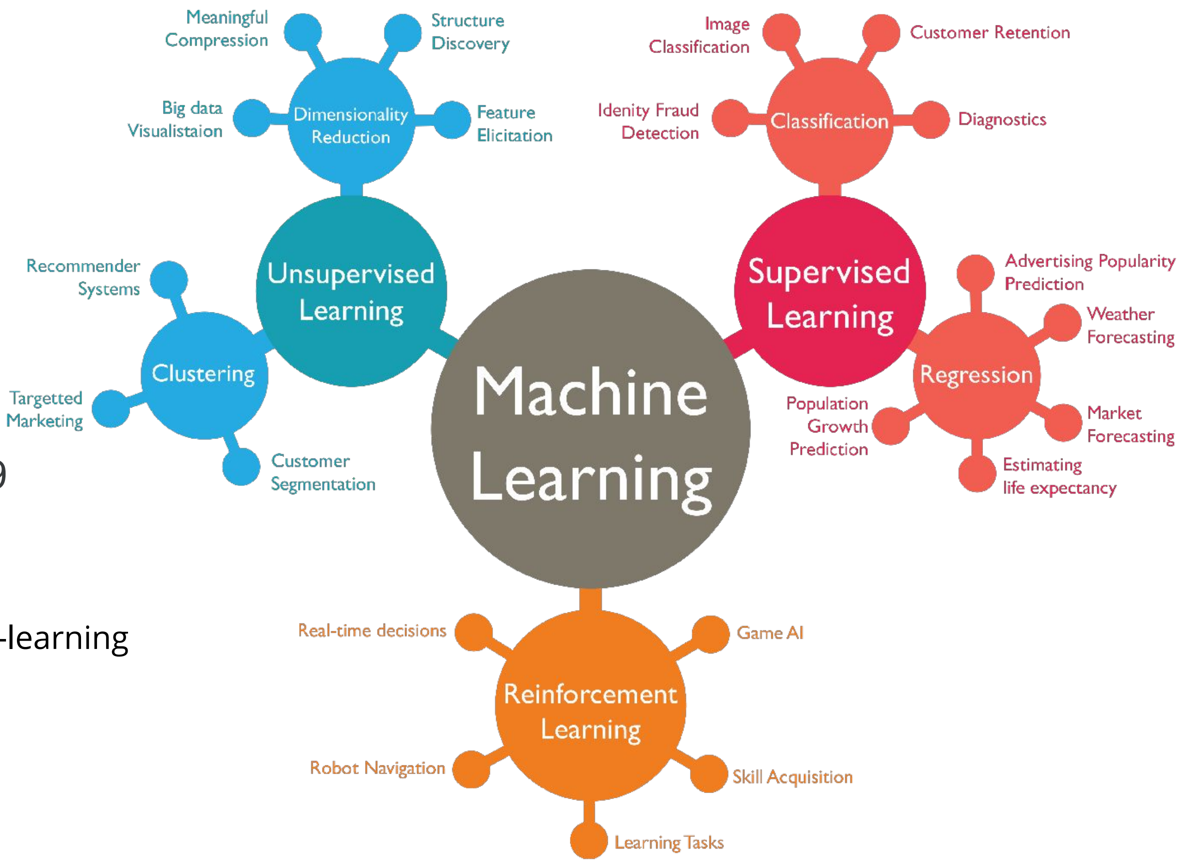



Actual trajectories flown using learned controller.

# Future Work

- Andrew Ng,
  Co-founder, Coursera
  Adjunct Professor, Stanford
- Free 11 week ML Course
- New Session May 28- Aug 19
- Enrollment starts May 19th

https://www.coursera.org/learn/machine-learning



# Q & A