Reinforcement Learning
Lecture 4

Georg Martius

Distributed Intelligence / Autonomous Learning Group, Uni Tübingen, Germany

November 5, 2024

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Overview

- Last time: unknown MDP
  - how to estimate the value function
  - no model (model-free)
- Today: Solve an unknown MDP
  - optimize the value function
  - obtain optimal policy
  - also without model
- Next time: Bandits – a special case of RL problems with a single state and exploration
  (given by Claire Vernade)

Model-free Control

Plan for today

1. Q-function
2. Monte-Carlo Control
3. Temporal Difference Learning
4. On-Policy vs. Off-Policy
5. Off-Policy Learning (Q-learning)

Where to apply model-free control?

- MDP model is unknown, but experience can be sampled
  Examples:
  - Robot control
  - Tokamak plasma control
  - Protein Folding
- MDP model is known, but is too big to use, except by samples
  Examples:
  - Game of Go
  - Elevator optimization
  - Optimizing decision making problems in simulation

What stops us from optimizing the value function?

We know how to estimate the value function: Monte-Carlo or TD
How to derive the (greedy) policy? (deterministic notation)

$$\pi'(s) = \underset{a \in \mathcal{A}}{\arg\max}\, \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

➡ greedy policy improvement over $V(s)$ requires the MDP or a model of it

What to do? Learn an action-value function!

Action-value function (reminder)

$q_\pi(s, a)$: value of performing action a in state s and then following the policy
$Q(s, a)$ is an estimate of $q_\pi(s, a)$

$$\pi'(s) = \underset{a \in \mathcal{A}}{\arg\max}\, Q(s, a)$$

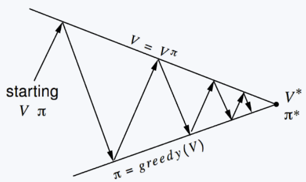
➡ greedy policy improvement over $Q(s, a)$ is model-free

Policy Iteration (reminder)

Policy Iteration

Given a policy π

1. Policy Evaluation: Evaluate the policy π
   ➡ estimate v_π
2. Policy Improvement: $\pi' > \pi$
   e.g. Improve the policy by acting greedily with respect to v_π requires model
3. iterate until value function does not change



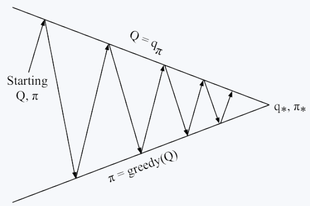
- Policy iteration always converges to π^*

Policy Iteration with Action-Value Function

Policy Iteration with Action-Value Function

Given a policy π

1. Policy Evaluation: Evaluate the policy π
   ➡ estimate action-value function q_π
2. Policy Improvement: $\pi' > \pi$
   e.g. Improve the policy by acting greedily with respect to q_π ✓
3. iterate until action-value function does not change



Greedy action selection

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

Example: There are two doors in front of you:

- You open the left door and get reward 0
  $V(\text{left}) = 0$
- You open the right door and get reward $+1$
  $V(\text{right}) = +1$
- You open the right door and get reward $+3$
  $V(\text{right}) = +2$
- You open the right door and get reward $+2$
  $V(\text{right}) = +2$

Are you sure you've chosen the best door?

Exploration is needed!

ϵ-Greedy exploration

Simplest idea for ensuring continual exploration

- All m actions are tried with non-zero probability
- With probability $1-\epsilon$ choose the greedy action
- With probability ϵ choose an action at random

$$\pi(a \mid s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{if } a^* = \arg\max_{a \in \mathcal{A}} Q(s,a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$$

Does ϵ-Greedy policy harm improvement? NO

i.e.: does Policy Iteration still converge?

Theorem: ϵ-Greedy Policy Improvement

For any ϵ-greedy policy π,
the ϵ-greedy policy π' with respect to q_π
is an improvement, i.e. $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a \mid s) q_\pi(s,a) \\ &= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\epsilon) \max_{a \in \mathcal{A}} q_\pi(s,a) \\ &\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a \mid s) - \epsilon/m}{1-\epsilon} q_\pi(s,a) \\ &= \sum_{a \in \mathcal{A}} \pi(a \mid s) q_\pi(s,a) = v_\pi(s) \end{aligned}$$

Therefore: $v_{\pi'}(s) \geq v_\pi(s)$

Monte-Carlo Action-value function estimation

1. trajectory using π: $\tau = \{S_1, A_1, R_2, ..., S_T\} \sim \pi$
2. For each state S_t and action A_t in τ:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

3. iterate

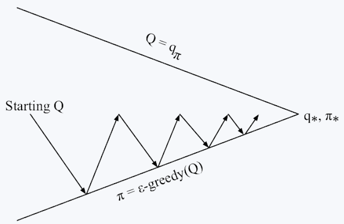
We have the ingredients

- Action-value estimation
- Policy: ϵ-greedy

Monte-Carlo Control / Policy Iteration

Every episode we iterate:

1. Policy evaluation: Monte-Carlo action-value estimation $Q \approx q_\pi$
2. Policy improvement: ϵ-greedy policy improvement



Convergence?

What do you think is needed?

1. nothing, it converges always ✗
2. it does not converge ✗
3. a lot of iterations ✓
4. every state, action pair is visited infinitely often ✓
5. a greedy policy ✗
6. policy needs to become greedy in the limit ✓
7. policy needs to keep exploring ✓

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \to \infty} \pi_k(a \mid s) = [\![a = \arg\max_{a' \in \mathcal{A}} Q_k(s, a')]\!]$$

Convergence?

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \to \infty} \pi_k(a \mid s) = [\![a = \arg\max_{a' \in \mathcal{A}} Q_k(s, a')]\!]$$

- Adapt exploration rate as $\epsilon \leftarrow \frac{1}{k}$
  - ➭ converges to greedy policy
- infinite exploration because $\sum_{k=1}^{\infty} \frac{1}{k} = \infty$

Theorem

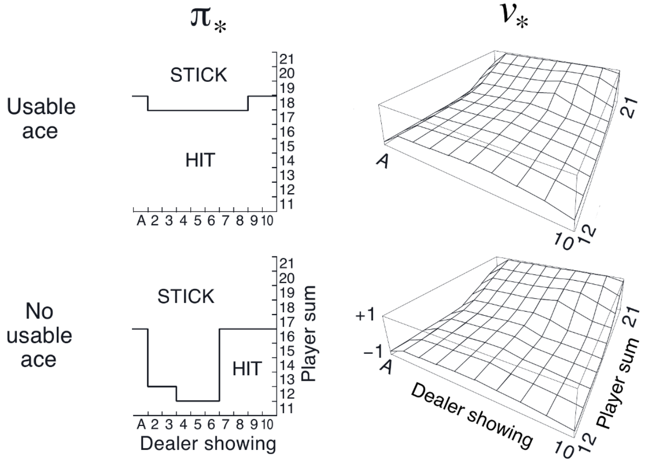
GLIE Monte-Carlo Control converges to the optimal action-value function, $Q(s, a) \to q^*(s, a)$

Example: Blackjack

- Goal: get as close as possible to 21 points but not above
- Counting: Face-cards: 10, Ace: 1 or 11, other: cards their value
- start with two cards, dealer has one card open

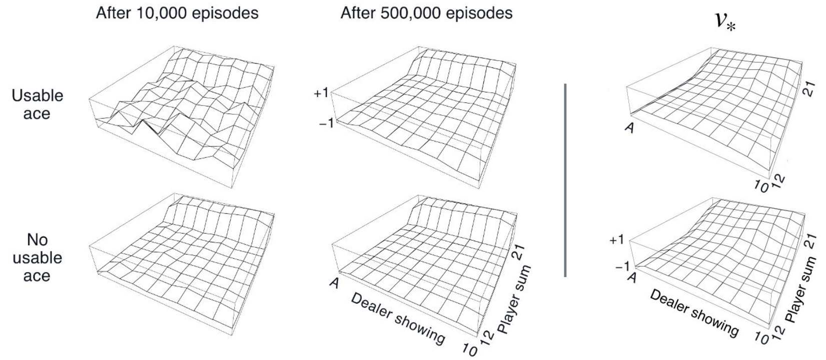
Example: Monte-Carlo Control in Blackjack

- Dealer: stick if sum of cards ≥ 17, otherwise twist



Compare: Optimal Value vs Policy Evaluation

- Policy: stick if sum of cards ≥ 20, otherwise twist (suboptimal policy)
- Dealer: stick if sum of cards ≥ 17, otherwise twist



Break

Using Temporal-difference (TD) learning

What are the advantages of TD over MC?

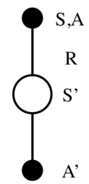
- Lower variance
- online
- incomplete sequences
- exploits Markov property

Natural idea: use TD instead of MC

Simplest TD Q update:

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma Q(S', A') - Q(S, A)\right)$$

S A R S A



SARSA

1. Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
2. Repeat (for each episode):
   - 2.1 $S \leftarrow$ from environment
   - 2.2 $A \sim \pi_\epsilon^Q(S)$ (ϵ -greedy w.r.t. Q)
   - 2.3 Repeat (for each step of episode):
     - 2.3.1 Take action A, observe R, S'
     - 2.3.2 $A' \sim \pi_\epsilon^Q(S')$ (ϵ -greedy w.r.t. Q)
     - 2.3.3 $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma Q(S', A') - Q(S, A)\right]$
     - 2.3.4 $S \leftarrow S'; A \leftarrow A'$
   - 2.4 until S is terminal

Convergence of Sarsa

Theorem

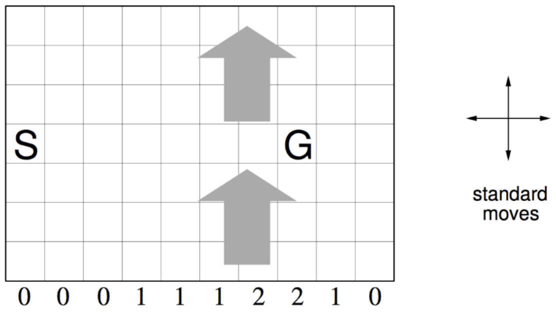
Sarsa converges to the optimal action-value function, $Q(s, a) \to q^*(s, a)$, under the following conditions:

- GLIE sequence of policies $\pi_t(a \mid s)$ (decay of exploration rate)
- Robbins-Monro sequence of step-sizes α_t

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

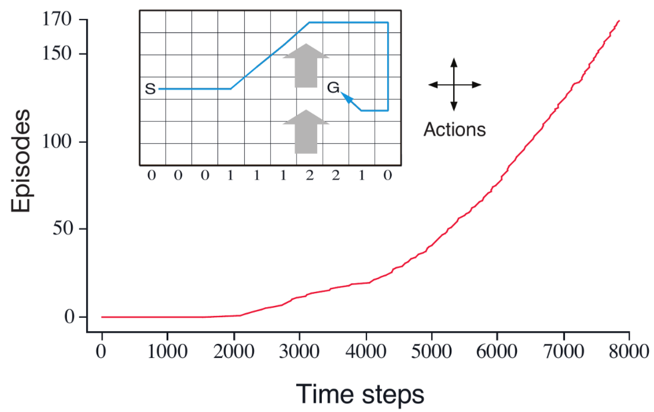
Windy Gridworld Example



Reward: $R = -1$ per timestep until reaching goal

Undiscounted: $\gamma = 1$

SARSA on the Windy Gridworld Example



SARSA(λ) Algorithm

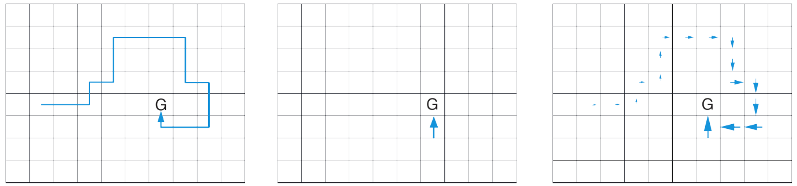
Naturally we can combine n-step TD updates with SARSA ➧ SARSA(λ)
Use eligibility traces (backward view):

SARSA(λ)

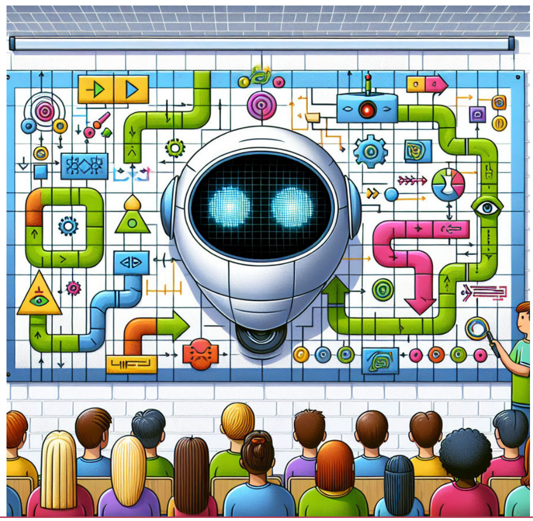
1. Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
2. Repeat (for each episode):
   - 2.1 $E(s, a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
   - 2.2 $S \leftarrow$ from environment
   - 2.3 $A \sim \pi_\epsilon^Q(S)$ (ϵ -greedy w.r.t. Q)
   - 2.4 Repeat (for each step of episode):
     - 2.4.1 Take action A, observe R, S'
     - 2.4.2 $A' \sim \pi_\epsilon^Q(S')$ (ϵ -greedy w.r.t. Q)
     - 2.4.3 $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
     - 2.4.4 $E(S, A) \leftarrow E(S, A) + 1$
     - 2.4.5 For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
       $Q(s, a) \leftarrow Q(s, a) + \alpha\delta E(s, a)$
       $E(s, a) \leftarrow \gamma\lambda E(s, a)$
     - 2.4.6 $S'; A \leftarrow A'$
   - 2.5 until S is terminal

SARSA(λ) update on the Windy Gridworld Example

Path taken — Action values increased by one-step Sarsa — Action values increased by Sarsa(λ) with λ=0.9



Break



On-policy vs. Off-Policy

On-policy

- "Learn while doing the job"
- Learn about policy π from experience sampled with π

Examples:

- MC Control/Policy Iteration
- SARSA

Off-policy

- "Look over someone's shoulder"
- Learn about policy π from experience sampled with μ

Examples: (discussed next)

- Importance Sampling MC
- Q-Learning

Why is off-policy learning a good idea?

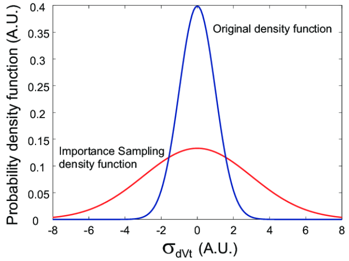
- can learn from demonstrations
- can re-use all experience (data from policies $\pi_1, \pi_2, \ldots \pi_{t-1}$)
- learn about optimal policy while following exploratory policy
- learn about multiple policies while following one policy

Importance Sampling

We get samples from current policy, but would like to compute the expectation with respect to a different policy.

Importance Sampling

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X)\\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(X)\\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]\end{aligned}$$



➪ estimate expectation of a different distribution

Importance Sampling for Off-Policy Monte-Carlo

Have: return G_t from policy μ

Importance Sampling: reweight according to similarity between policies

- Multiply importance sampling corrections along whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t \mid S_t)}{\mu(A_t \mid S_t)}\frac{\pi(A_{t+1} \mid S_{t+1})}{\mu(A_{t+1} \mid S_{t+1})}\cdots\frac{\pi(A_T \mid S_T)}{\mu(A_T \mid S_T)}G_t$$

- Update value towards corrected return

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^{\pi/\mu} - V(S_t)\right)$$

- Warning: cannot use if μ is zero when π is non-zero
- Importance sampling can dramatically increase variance

Similarly possible for TD: weight TD-target by $\frac{\pi}{\mu}$

Off-policy without Importance Sampling

Off-policy learning of action-values $Q(s, a)$

- $q_\pi(s, a)$: value of doing action a in state s and then following π
- we want to approximate $q_\pi(s, a)$,
  but get data from policy μ: $A_t \sim \mu(\cdot \mid S_t)$
- No problem!
  Current action A_t comes from exploration policy μ, but then we follow π.

$$A' \sim \pi(\cdot \mid S_{t+1})$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma\underbrace{Q(S_{t+1}, A')}_{\approx v_\pi(S_{t+1})} - Q(S_t, A_t)\right)$$

➪ Q estimates $q_\pi(s, a)$

Q-Learning

Let both exploration- and target policy improve!

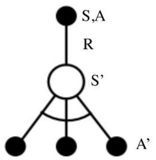
- The target policy π is greedy w.r.t. $Q(s, a)$ (approaching π^*)

$$\pi(S_{t+1}) = \arg\max_{a'} Q(S_{t+1}, a')$$

- The behavior policy μ is e.g. ϵ-greedy w.r.t. $Q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned}& R_{t+1} + \gamma Q(S_{t+1}, A') && A' \sim \pi(\cdot \mid S_{t+1})\\ &= R_{t+1} + \gamma Q\left(S_{t+1}, \arg\max_{a'} Q(S_{t+1}, a')\right)\\ &= R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a')\end{aligned}$$

Q-Learning Control update



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$$

Theorem

Q-Learning (Control) converges to the optimal action-value function, $Q(s,a) \to q^*(s,a)$

Q-Learning Algorithm

Q-Learning

1. Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
2. Repeat (for each episode):
   2.1 $S \leftarrow$ from environment
   2.2 Repeat (for each step of episode):
      2.2.1 $A \sim \pi_\epsilon^Q(S)$ (ϵ -greedy w.r.t. Q)
      2.2.2 Take action A, observe R, S'
      2.2.3 $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right]$
      2.2.4 $S \leftarrow S'$
   2.3 until S is terminal

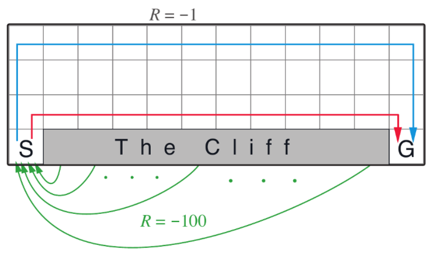
Q-Learning Demo

`python3 gridworld.py -a q -k 20 -n 0`
(You can do the same after completing the homeworks)

Let's look at it after 100 or 1000 iterations:
`python3 gridworld.py -a q -k 100 -n 0 -q`

What happens if we add noise?
`python3 gridworld.py -a q -k 1000 -n 0.2 -q`

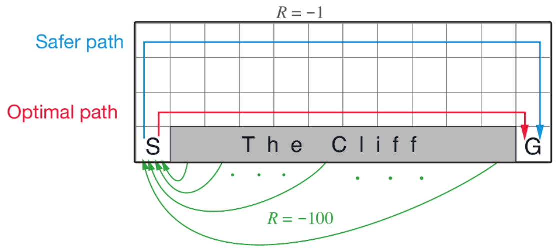
Cliff Walking Example

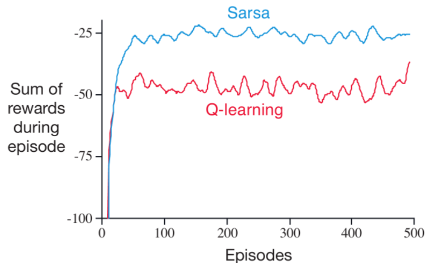


Fixed ϵ-greedy exploration policy. No noise in environment.

1. Which path will be learned by Q-Learning?
2. Which path will be learned by SARSA? (Poll)

Cliff Walking Example

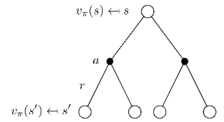
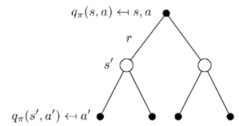
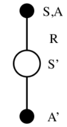
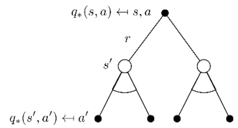
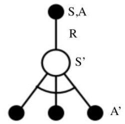




Q-learning learns the optimal policy, but shows lower online performance
(falls occationally off the cliff)

SARSA learns to cope with the ϵ-greedy policy

Relationship between DP and TD

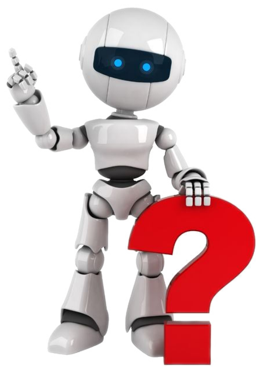
	Full Backup (DP)	Sample Backup (TD)
Bellman Expectation Equation for $v_\pi(s)$	 Iterative policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s,a)$	 Q-policy Iteration	 SARSA
Bellman Optimality Equation for $q^*i(s,a)$	 Q-Value Iteration	 Q-Learning

Relationship between DP and TD (cont)

Full Backup (DP)	Sample Backup (TD)
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}[R + \gamma V(S') \mid s]$	TD Learning $V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-Policy Iteration $Q(s,a) \leftarrow \mathbb{E}[R + \gamma Q(S',A') \mid s,a]$	SARSA $Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
Q-Value Iteration $Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	Q-Learning $Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$

where $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$

Questions?



[Image: globalrobots.com]

