CSCI S-89C Deep Reinforcement Learning

Harvard Summer School

Dmitry Kurochkin

Summer 2020
Lecture 8

Contents

1. Quiz Review
   - Extra Credit: Quizzes 1-6
   - Quiz 7
2. Approximate Solution Methods (Continued)
   - Q-Learning
     - Q-Learning with Approximation
     - Double Q-Learning with Approximation
     - Deep Q-Network
   - Eligibility Traces
     - λ-return Algorithm
     - λ-return Algorithm with Approximation
     - TD(λ) with Approximation: Eligibility Traces
     - SARSA(λ) with Approximation: Eligibility Traces
3. Reinforcement Learning as Supervised Learning Algorithms
   - Prediction Problem via Supervised Learning

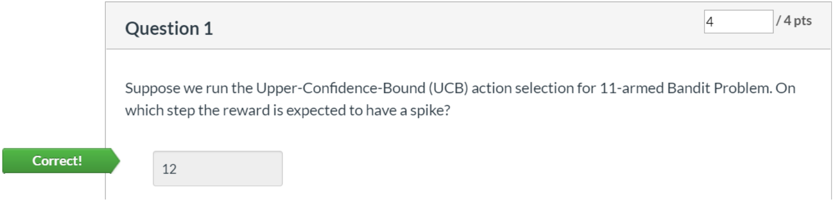
Contents

1 Quiz Review
- Extra Credit: Quizzes 1-6
- Quiz 7

2 Approximate Solution Methods (Continued)
- Q-Learning
  - Q-Learning with Approximation
  - Double Q-Learning with Approximation
  - Deep Q-Network
- Eligibility Traces
  - λ-return Algorithm
  - λ-return Algorithm with Approximation
  - TD(λ) with Approximation: Eligibility Traces
  - SARSA(λ) with Approximation: Eligibility Traces

3 Reinforcement Learning as Supervised Learning Algorithms
- Prediction Problem via Supervised Learning

Extra Credit: Quizzes 1-6



Extra Credit: Quizzes 1-6

Question 2

4 / 4 pts

The joint distribution $P\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$ of the environment's state S_t and the reward R_t, conditional on the previous state $S_{t-1} = s$ and the agent's action $A_{t-1} = a$, is denoted by $p(s', r|s, a)$.

The marginal distribution of the next state S_t under the same condition, $S_{t-1} = s$ and $A_{t-1} = a$, is then

(A) $\sum_{s'} p(s', r|s, a)$

(B) $\sum_{r} p(s', r|s, a)$

(C) $\sum_{s',r} p(s', r|s, a)$

(D) None of (A), (B), (C)

Please select:

Correct!

- B
- C
- A
- D

Extra Credit: Quizzes 1-6

Question 3 4 / 4 pts

One can derive an optimal policy from the state-value function $v_*(s)$ if the transition probabilities $p(s', r|s, a)$ are known.

Correct!

- (•) True
- ( ) False

Extra Credit: Quizzes 1-6

Question 4

4 / 4 pts

Let $q_\pi(s, a)$ denote the action-value function that corresponds to an ε-soft policy $\pi(a|s)$ with $\varepsilon > 0$.

Then the ε-greedy policy $\pi'(a|s)$ with respect to q_π can only improve the policy π, that is, $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$.

Correct!

- (•) True
- ( ) False

Extra Credit: Quizzes 1-6

Question 5

4 / 4 pts

The environment has three states: s_A, s_B, and s_C. In each state there are two actions, a_1 and a_2, available.

Suppose we want to estimate $q_\pi(s, a)$ using the 1-step TD learning, where π is some policy.

We generate the sequence under this policy π and observe:

$s_A, a_1, R_1 = 5, s_B, a_2, \ldots$

If $\alpha = 0.1$, $\gamma = 0.9$, and initial values are

$Q(s_A, a_1) = 1, Q(s_A, a_2) = 2,$

$Q(s_B, a_1) = 3, Q(s_B, a_2) = 4,$

$Q(s_C, a_1) = 5, Q(s_C, a_2) = 6,$

what is $Q(s_A, a_1)$ after its first update according to the 1-step TD method?

Hint: use the following n-step on-policy TD updates:

$Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:(t+n)} - Q_{t+n-1}(S_t, A_t)\right]$

with $n = 1$.

Correct!

1.76

Extra Credit: Quizzes 1-6

Question 5 4 / 4 pts

The environment has three states: s_A, s_B, and s_C. In each state there are two actions, a_1 and a_2, available.

Suppose we want to estimate $q_\pi(s, a)$ using the 1-step TD learning, where π is some policy.

We generate the sequence under this policy π and observe:

$s_A, a_1, R_1 = 5, s_B, a_2, \ldots$

If $\alpha = 0.1$, $\gamma = 0.9$, and initial values are

$Q(s_A, a_1) = 1, Q(s_A, a_2) = 2$,

$Q(s_B, a_1) = 3, Q(s_B, a_2) = 4$,

$Q(s_C, a_1) = 5, Q(s_C, a_2) = 6$,

what is $Q(s_A, a_1)$ after its first update according to the 1-step TD method?

$$
\begin{aligned}
Q_1(S_0, A_0) &= Q_0(S_0, A_0) + \alpha\Big[\overbrace{R_1 + \gamma Q_0(S_1, A_1)}^{G_{0:1}} - Q_0(S_0, A_0)\Big] \\
&= Q_0(s_A, a_1) + \alpha\left[R_1 + \gamma Q_0(s_B, a_2) - Q_0(s_A, a_1)\right] \\
&= 1 + 0.1\left[5 + 0.9 \cdot 4 - 1\right] \\
&= 1.76
\end{aligned}
$$

Contents

1. Quiz Review
   - Extra Credit: Quizzes 1-6
   - Quiz 7
2. Approximate Solution Methods (Continued)
   - Q-Learning
     - Q-Learning with Approximation
     - Double Q-Learning with Approximation
     - Deep Q-Network
   - Eligibility Traces
     - λ-return Algorithm
     - λ-return Algorithm with Approximation
     - TD(λ) with Approximation: Eligibility Traces
     - SARSA(λ) with Approximation: Eligibility Traces
3. Reinforcement Learning as Supervised Learning Algorithms
   - Prediction Problem via Supervised Learning

Quiz 7

Question 1

4 / 4 pts

Suppose we have a MDP with $\mathcal{S} = \{s_A, s_B, s_C\}$.

The state-value function is being approximated by

$v_\pi(s) \approx \hat{v}(s, \mathbf{w}) \doteq w_1 \cdot 1_{(s=s_A)} + w_2 \cdot 1_{(s=s_B)} + w_3 \cdot 1_{(s=s_C)}$,

where $\mathbf{w} = (w_1, w_2, w_3)^T$ are *weights.*

If at time t we have weights estimated to be $\mathbf{w}_t = (2.3, 1.4, 0.8)^T$, what is the estimate of $v_\pi(s_C)$?

Correct!

0.8

Quiz 7

Question 2

4 / 4 pts

Suppose we have a MDP with $\mathcal{S} = \{s_A, s_B, s_C\}$.

The state-value function is being approximated by

$v_\pi(s) \approx \hat{v}(s, \mathbf{w}) \doteq w_1 \cdot \mathbf{1}_{(s=s_A)} + w_2 \cdot \mathbf{1}_{(s=s_B)} + w_3 \cdot \mathbf{1}_{(s=s_C)}$,

where $\mathbf{w} = (w_1, w_2, w_3)^T$ are *weights*.

If at time t we have weights estimated to be $\mathbf{w}_t = (2.3, 1.4, 0.8)^T$, what is $\nabla \hat{v}_\pi(s_C, \mathbf{w_t})$?

Please notice that ∇ denotes the gradient with respect to the weights.

- 0.8
- 0
- Correct! (0, 0, 1)
- 1
- (0, 0, 0.8)

Quiz 7

Question 3

4 / 4 pts

Suppose that the environment has four states {stage k} with k=1, 2, 3, 4 (k is a *feature* of the state):

s_A ={stage 1}, s_B ={stage 2}, s_C ={stage 3}, and s_D ={stage 4}.

The state-value function is being approximated by the following linear function (linear in weights):

$v_\pi(s) \approx \hat{v}(s, \mathbf{w}) \doteq w_1 + w_2 \cdot k + w_3 \cdot k^2$,

where $\mathbf{w} = (w_1, w_2, w_3)^T$ are weights and k *corresponds to the state* s.

For example, $v_\pi(s_C) \approx \hat{v}(s_C, \mathbf{w}) \doteq w_1 + w_2 \cdot 3 + w_3 \cdot 3^2$.

If we want to run the stochastic-gradient descent (SGD) constant-α MC for estimating v_π,

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[\underbrace{G_t}_{\approx v_\pi(S_t)} - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t),$$

and the current state and weights are $S_t = s_B$ and $\mathbf{w}_t = (1.4, 3.2, 2.8)^T$, respectively, what gradient $\nabla \hat{v}(S_t, \mathbf{w}_t)$ should we use?

Please select:

- (0, 3.2, 11.2)
- (1.4, 6.4, 11.2)
- Correct! (1, 2, 4)
- (1, 4, 16)

Quiz 7

Question 4

4 / 4 pts

Suppose that the environment has four states {stage k} with k=1, 2, 3, 4:

s_A ={stage 1}, s_B ={stage 2}, s_C ={stage 3}, and s_D ={stage 4}.

The state-value function is being approximated by the following linear function (linear in weights):

$v_\pi(s) \approx \hat{v}(s, \mathbf{w}) \doteq w_1 + w_2 \cdot k + w_3 \cdot k^2$,

where $\mathbf{w} = (w_1, w_2, w_3)^T$ are *weights* and k *corresponds to the state* s.

We generate the sequence under a policy π and observe:

$s_A, a_1, R_1 = 6, s_B, \ldots$

If $\alpha = 0.1$, $\gamma = 0.9$, and initial values (i.e. values at time $t = 0$) for weights are $\mathbf{w}_0 = (1.4, 3.2, 2.8)^T$, what is $\mathbf{w}_1$ after the first update (i.e. weights at $t = 1$) according to the semi-gradient 1-step TD method:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[\underbrace{R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t)}_{\approx v_\pi(S_t)} - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t)?$$

Please select:

- (2.97, 6.34, 9.08)
- (1.926, 3.727, 3.326)
- (2.198, 5.024, 4.396)
- Correct! (2.97, 4.77, 4.37)

Quiz 7

Question 5

4 / 4 pts

The function approximation methods in reinforcement learning assume that the approximating functions are linear in weights. For example, $\hat{q}(s, a, \mathbf{w})$ in the approximation $q_\pi(s, a) \approx \hat{q}(s, a, \mathbf{w})$ is linear in $\mathbf{w} = (w_1, w_2, \ldots, w_d)$.

- True
- False

Correct!

Contents

1 Quiz Review
- Extra Credit: Quizzes 1-6
- Quiz 7

2 Approximate Solution Methods (Continued)
- Q-Learning
  - Q-Learning with Approximation
  - Double Q-Learning with Approximation
  - Deep Q-Network
- Eligibility Traces
  - λ-return Algorithm
  - λ-return Algorithm with Approximation
  - TD(λ) with Approximation: Eligibility Traces
  - SARSA(λ) with Approximation: Eligibility Traces

3 Reinforcement Learning as Supervised Learning Algorithms
- Prediction Problem via Supervised Learning

Q-Learning with Approximation

Recall tabular *Q-learning:*

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right],$$

the algorithm which will converge to $q_*(s,a)$ as long as all pairs (s,a) continue to be updated, i.e. all state-action pairs (S_t, A_t) continue to be visited.

Notice: we follow some behavioral policy, say b, to generate data but pick a greedy action

$$\text{argmax}_a Q(S_{t+1}, a)$$

for estimating Q (compare with SARSA),
i.e. no importance-sampling ratio is needed!

Q-Learning with Approximation

The Q-learning,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right],$$

can also be used with approximation:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t).$$

Doble Q-Learning with Approximation

Similarly, the Double Q-learning algorithm in which we keep two estimates, Q_1 and Q_2, of $q_*(s,a)$ and update only one of them (each case has probability 0.5 of being selected) at a time:

$$\begin{cases} Q_1(S_t,A_t) \leftarrow Q_1(S_t,A_t) + \alpha\left[R_{t+1} + \gamma Q_2(S_{t+1}, \operatorname{argmax}_a Q_1(S_{t+1},a)) - Q_1(S_t,A_t)\right], \\ Q_2(S_t,A_t) \leftarrow Q_2(S_t,A_t) + \alpha\left[R_{t+1} + \gamma Q_1(S_{t+1}, \operatorname{argmax}_a Q_2(S_{t+1},a)) - Q_2(S_t,A_t)\right], \end{cases}$$

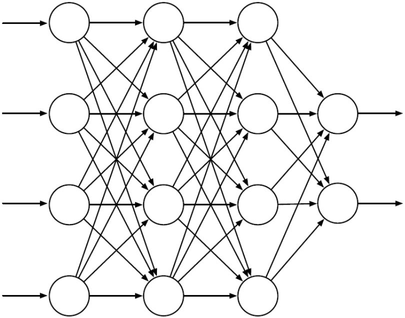
can also be used with approximation as follows:

$$\begin{cases} \mathbf{w}^A \leftarrow \mathbf{w}^A + \alpha\Big[R_{t+1} + & \gamma\hat{q}(S_{t+1}, \operatorname{argmax}_a \hat{q}(S_{t+1}, a, \mathbf{w}^A), \mathbf{w}^B) \\ & -\hat{q}(S_t, A_t, \mathbf{w}^A)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}^A), \\ \mathbf{w}^B \leftarrow \mathbf{w}^B + \alpha\Big[R_{t+1} + & \gamma\hat{q}(S_{t+1}, \operatorname{argmax}_a \hat{q}(S_{t+1}, a, \mathbf{w}^B), \mathbf{w}^A) \\ & -\hat{q}(S_t, A_t, \mathbf{w}^B)\Big]\nabla\hat{q}(S_t, A_t, \mathbf{w}^B), \end{cases}$$

where we keep two vectors of weights, $\mathbf{w}^A$ and $\mathbf{w}^B$, and update only one of them (choosing randomly) on each time step.

Deep Q-Network

Alternatively, $Q(s, a)$ in the Q-learning algorithm can be approximated via Deep Neural Networks (NNs) resulting in *Deep Q-Network* algorithm. In addition, similarly to Double Q-learning, one can keep two NNs and update only one of them at a time.



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

1. Quiz Review
   - Extra Credit: Quizzes 1-6
   - Quiz 7
2. Approximate Solution Methods (Continued)
   - Q-Learning
     - Q-Learning with Approximation
     - Double Q-Learning with Approximation
     - Deep Q-Network
   - Eligibility Traces
     - λ-return Algorithm
     - λ-return Algorithm with Approximation
     - TD(λ) with Approximation: Eligibility Traces
     - SARSA(λ) with Approximation: Eligibility Traces
3. Reinforcement Learning as Supervised Learning Algorithms
   - Prediction Problem via Supervised Learning

λ-return Algorithm

Recall the tabular n-step TD:

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}),$$

where $V_t(s)$ denotes the estimate of $v_\pi(s)$ at time t
(if $t+n \geq T$, the convention is that all missing terms are zeros).

The <u>n-step TD</u> updates are

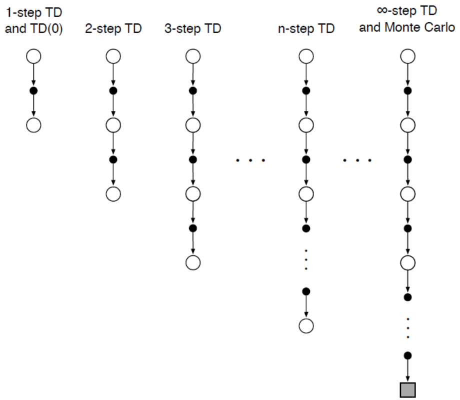
$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[G_{t:(t+n)} - V_{t+n-1}(S_t) \right]$$

λ-return Algorithm

To summarize, the tabular n-step TD:

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$$

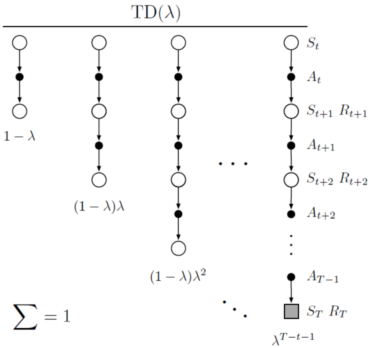
$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[G_{t:(t+n)} - V_{t+n-1}(S_t) \right]$$



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

λ-return Algorithm

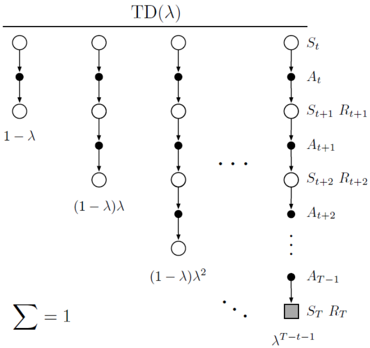
What n should we pick?



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

λ-return Algorithm

What n should we pick?



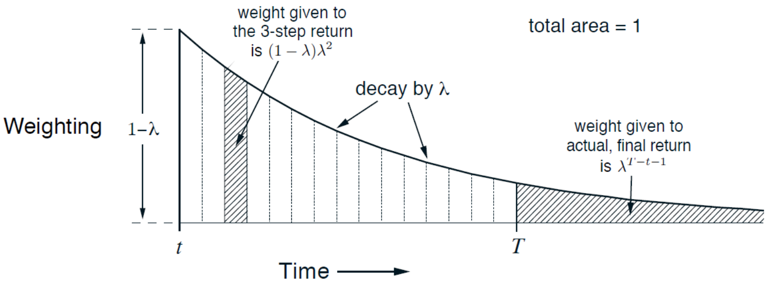
Let's use a weighted sum: $G_t^\lambda \doteq \sum_{n=1}^{T-t-1} (1-\lambda)\lambda^{n-1} G_{t:(t+n)} + \lambda^{T-t-1} G_t$.

Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

λ-return Algorithm

The λ-*return* is

$$G_t^\lambda \doteq \sum_{n=1}^{T-t-1} (1-\lambda)\lambda^{n-1} G_{t:(t+n)} + \lambda^{T-t-1} G_t.$$



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

λ-return Algorithm

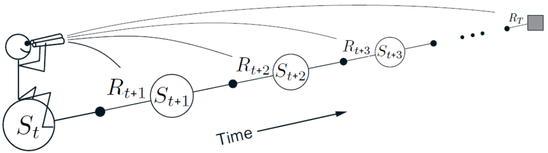
The λ-*return* algorithm is

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \left[G_t^\lambda - V_{t+n-1}(S_t) \right],$$

where

$$G_t^\lambda \doteq \sum_{n=1}^{T-t-1} (1-\lambda)\lambda^{n-1} G_{t:(t+n)} + \lambda^{T-t-1} G_t \quad \text{with } \lambda \in [0,1],$$

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}).$$



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

λ-return Algorithm with Approximation

Similarly, in the case of approximation $v_\pi(s) \approx \hat{v}(s, \mathbf{w})$,
the λ-return algorithm is

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \left[G_t^\lambda - \hat{v}(S_t, \mathbf{w}_t) \right] \nabla \hat{v}(S_t, \mathbf{w}_t),$$

where

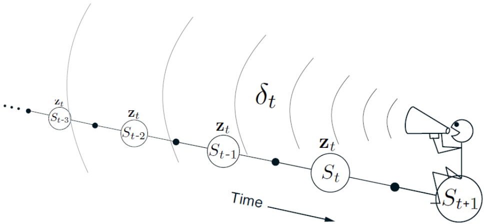
$$G_t^\lambda \doteq \sum_{n=1}^{T-t-1} (1-\lambda)\lambda^{n-1} G_{t:(t+n)} + \lambda^{T-t-1} G_t \quad \text{with } \lambda \in [0,1],$$

$$G_{t:(t+n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1})$$

TD(λ) with Approximation: Eligibility Traces

Let $\mathbf{z}_t \in \mathbb{R}^d$ with $\mathbf{z}_{-1} \doteq 0$.
The TD(λ), where $\lambda \in [0, 1]$ is *trace-decay* parameter, algorithm with approximation is

$$\mathbf{z}_t \doteq \gamma\lambda\mathbf{z}_{t-1} + \nabla\hat{v}(S_t, \mathbf{w}_t)$$
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \underbrace{\left[R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)\right]}_{\doteq \delta_t} \mathbf{z}_t$$



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

TD(λ) with Approximation: Eligibility Traces

TD(λ) algorithm:

Semi-gradient TD(λ) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

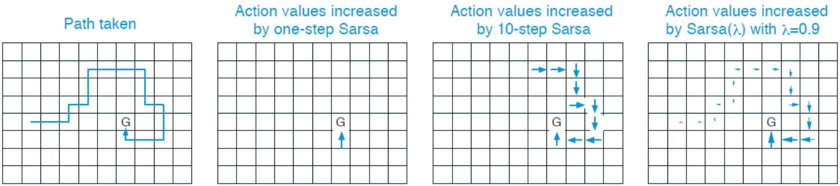
Loop for each episode:
    Initialize S
    $\mathbf{z} \leftarrow \mathbf{0}$ (a d-dimensional vector)
    Loop for each step of episode:
    | Choose $A \sim \pi(\cdot|S)$
    | Take action A, observe R, S'
    | $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \nabla\hat{v}(S,\mathbf{w})$
    | $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
    | $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
    | $S \leftarrow S'$
    until S' is terminal

Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

SARSA(λ) with Approximation: Eligibility Traces

Let $\mathbf{z}_t \in \mathbb{R}^d$ with $\mathbf{z}_{-1} \doteq 0$.
The SARSA(λ), where $\lambda \in [0,1]$ is *trace-decay* parameter, algorithm with approximation is

$$\mathbf{z}_t \doteq \gamma\lambda\mathbf{z}_{t-1} + \nabla\hat{q}(S_t, A_t, \mathbf{w}_t)$$
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \underbrace{\left[R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)\right]}_{\doteq \delta_t} \mathbf{z}_t$$



Source: *Reinforcement Learning: An Introduction* by R. Sutton and A. Barto

Contents

1 Quiz Review
- Extra Credit: Quizzes 1-6
- Quiz 7

2 Approximate Solution Methods (Continued)
- Q-Learning
  - Q-Learning with Approximation
  - Double Q-Learning with Approximation
  - Deep Q-Network
- Eligibility Traces
  - λ-return Algorithm
  - λ-return Algorithm with Approximation
  - TD(λ) with Approximation: Eligibility Traces
  - SARSA(λ) with Approximation: Eligibility Traces

3 Reinforcement Learning as Supervised Learning Algorithms
- Prediction Problem via Supervised Learning

Supervised Learning for Estimating $v_\pi(s)$

"training data":

$$\langle S_1, v_\pi(S_1)\rangle, \langle S_2, v_\pi(S_2)\rangle, \langle S_3, v_\pi(S_3)\rangle \ldots, \langle S_T, v_\pi(S_T)\rangle.$$

We can approximate $v_\pi(s)$ with

- G_t (MC)
- $R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t)$ (1-step TD)
- $G_{t:(t+n)}$ (n-step TD)
- etc.

-think of these as noisy "measurements" of $v_\pi(s)$.