CS 747, Autumn 2020: Week 10, Lecture 1

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2020

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)

Multi-step Returns

- For illustration consider prediction—estimating V^{π}.

Multi-step Returns

- For illustration consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_{\top}.}$$

Multi-step Returns

- For illustration consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

Multi-step Returns

- For illustration consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

- With First-visit Monte Carlo, our update would be

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 \cdot 2 + \gamma^4 \cdot 1 - V^{\text{old}}(s_2)\}.$$

Multi-step Returns

- For illustration consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

- With First-visit Monte Carlo, our update would be

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 \cdot 2 + \gamma^4 \cdot 1 - V^{\text{old}}(s_2)\}.$$

- Can we make this update instead?

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 V^{old}(s_3) - V^{\text{old}}(s_2)\}.$$

Multi-step Returns

- For illustration consider prediction—estimating V^{π}.
- Suppose we generate this episode.

$$\boxed{s_2, 2, s_3, 1, s_3, 1, s_3, 2, s_2, 1, s_\top.}$$

- With TD(0), our first update would be:

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma V^{\text{old}}(s_3) - V^{\text{old}}(s_2)\}.$$

- With First-visit Monte Carlo, our update would be

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 \cdot 2 + \gamma^4 \cdot 1 - V^{\text{old}}(s_2)\}.$$

- Can we make this update instead?

$$V^{\text{new}}(s_2) \leftarrow V^{\text{old}}(s_2) + \alpha\{2 + \gamma \cdot 1 + \gamma^2 V^{old}(s_3) - V^{\text{old}}(s_2)\}.$$

Yes. It uses a 2-step return as target.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the *n*-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

- n-step TD makes updates of the form

$$\boxed{V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.}$$

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.
- n-step TD makes updates of the form

$$\boxed{V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.}$$

- For each $n \geq 1$, we have $\lim_{t\to\infty} V^t = V^\pi$.

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

- n-step TD makes updates of the form

$$\boxed{V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.}$$

- For each $n \geq 1$, we have $\lim_{t \to \infty} V^t = V^\pi$.
- What is the effect of n on bootstrapping?

n-step Returns

- Trajectory: $s^0, r^0, s^1, r^1, \ldots$.
- For $t \geq 0, n \geq 1$, the n-step return $G_{t:t+n}$ is

$$G_{t:t+n} \stackrel{\text{def}}{=} r^t + \gamma r^{t+1} + \gamma^2 r^{t+2} + \cdots + \gamma^{n-1} r^{t+n-1} + \gamma^n V^{t+n-1}(s^{t+n}).$$

- Convention: on episodic tasks, if a terminal state is encountered at $t + n'$ for $1 \leq n' < n$, take $G_{t:t+n} = G_{t:t+n'}$.

- n-step TD makes updates of the form

$$\boxed{V^{t+n}(s^t) \leftarrow V^{t+n-1}(s^t) + \alpha\{G_{t:t+n} - V^{t+n-1}(s^t)\}.}$$

- For each $n \geq 1$, we have $\lim_{t\to\infty} V^t = V^\pi$.
- What is the effect of n on bootstrapping?

  Small n means more bootstrapping.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes. $G_{t:t+1}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$$\frac{G_{t:t+1} + G_{t:t+2}}{2}.$$

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

$\frac{G_{t:t+1} - 2G_{t:t+2} + 4G_{t:t+3}}{3}$.

Combining Returns

- Consider updating the estimate of s^t at step $t + 3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\frac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\frac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

$\frac{G_{t:t+1} - 2G_{t:t+2} + 4G_{t:t+3}}{3}$. No.

Combining Returns

- Consider updating the estimate of s^t at step $t+3$ using

$$V^{t+3}(s^t) \leftarrow V^{t+2}(s^t) + \alpha\{\text{Target} - V^{t+2}(s^t)\}.$$

- Can we use this as our target?

$G_{t:t+3}$. Yes.

$G_{t:t+1}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2}}{2}$. Yes.

$\dfrac{2G_{t:t+1} + 3G_{t:t+2} + G_{t:t+3}}{6}$. Yes.

$\dfrac{G_{t:t+1} + G_{t:t+2} + 3G_{t:t+3}}{4}$. No.

$\dfrac{G_{t:t+1} - 2G_{t:t+2} + 4G_{t:t+3}}{3}$. No.

- Can use any convex combination of the applicable G's.

The λ-return

- A particular convex combination is the λ-return, $\lambda \in [0, 1]$:

$$\boxed{G_t^{\lambda} \stackrel{\text{def}}{=} (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_{t:T}}$$

where $s^T = s_\top$ (otherwise $T = \infty$).

The λ-return

- A particular convex combination is the λ-return, $\lambda \in [0, 1]$:

$$\boxed{G_t^\lambda \stackrel{\text{def}}{=} (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_{t:T}}$$

  where $s^T = s_\top$ (otherwise $T = \infty$).
- Observe that $G_t^0 = G_{t:t+1}$, yielding full bootstrapping.
- Observe that $G_t^1 = G_{t:\infty}$, a Monte Carlo estimate.
- In general, λ controls the amount of bootstrapping.

The λ-return

- A particular convex combination is the λ-return, $\lambda \in [0, 1]$:

$$\boxed{G_t^{\lambda} \stackrel{\text{def}}{=} (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_{t:T}}$$

  where $s^T = s_\top$ (otherwise $T = \infty$).
- Observe that $G_t^0 = G_{t:t+1}$, yielding full bootstrapping.
- Observe that $G_t^1 = G_{t:\infty}$, a Monte Carlo estimate.
- In general, λ controls the amount of bootstrapping.

- If $\lambda > 0$, transition (s^t, r^t, s^{t+1}) contributes to the update of every previously-visited state: that is, $s^0, s^1, s^2, \ldots, s^t$.
- The amount of contribution falls of geometrically.
- Updating with the λ-return as target can be implemented elegantly by keeping track of the "eligibility" of each previous state to be updated.

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)

TD(λ) algorithm

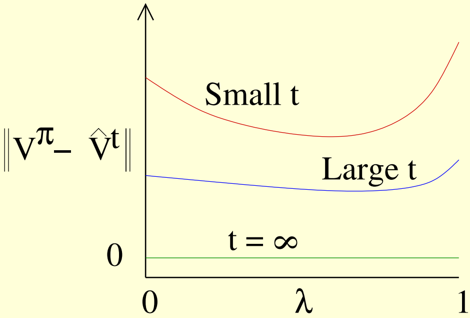
- Maintains an eligibility trace $z : S \to \mathbb{R}$.
- Implementation often called the backward view.

TD(λ) algorithm

- Maintains an eligibility trace $z : S \to \mathbb{R}$.
- Implementation often called the backward view.

Initialise $V : S \to \mathbb{R}$ arbitrarily.
Repeat for each episode:
  Set $z \to \mathbf{0}$.//Eligibility trace vector.
  Assume the agent is born in state s.
  Repeat for each step of episode:
    Take action a; obtain reward r, next state s'.
    $\delta \leftarrow r + \gamma V(s') - V(s)$.
    $z(s) \leftarrow z(s) + 1$.
    For all s:
      $V(s) \leftarrow V(s) + \alpha\delta z(s)$.
      $z(s) \leftarrow \gamma\lambda z(s)$.
    $s \leftarrow s'$.

Effect of λ



- Lower λ: more bootstrapping, more bias (less variance).
- Higher λ: more dependence on empirical rewards, more variance (less bias).
- For finite t, error is usually lowest for intermediate λ value.

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)

Half Field Offense

Half Field Offense

- Decision-making restricted to offense player with ball.
- Based on state, choose among DRIBBLE, PASS, SHOOT.

Half Field Offense

- Decision-making restricted to offense player with ball.
- Based on state, choose among DRIBBLE, PASS, SHOOT.
- How many states are there?

Half Field Offense

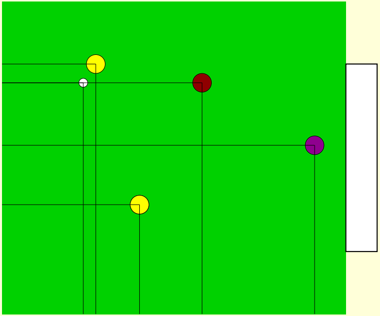
- Decision-making restricted to offense player with ball.
- Based on state, choose among DRIBBLE, PASS, SHOOT.
- How many states are there? An infinite number!

Half Field Offense

- Decision-making restricted to offense player with ball.
- Based on state, choose among DRIBBLE, PASS, SHOOT.
- How many states are there? An infinite number!
- What to do?

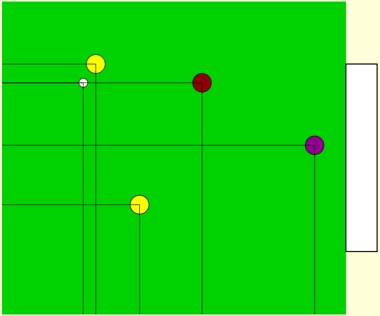
Features

- State s is defined by positions and velocities of players, ball.



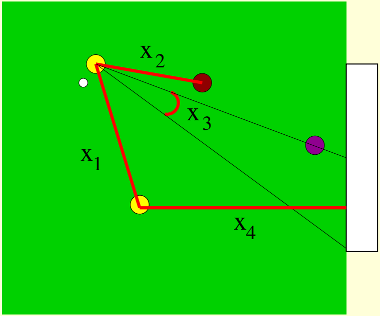
Features

- State s is defined by positions and velocities of players, ball.
- Velocities might not be important for decision making.
- Position coordinates might not generalise well.



Features

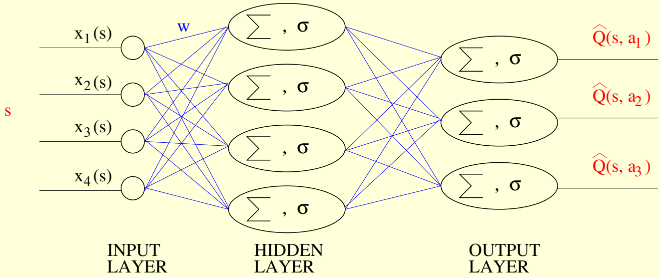
- State s is defined by positions and velocities of players, ball.
- Velocities might not be important for decision making.
- Position coordinates might not generalise well.
- Define features $x : S \rightarrow \mathbb{R}$. Idea is that states with similar features will have similar consequences of actions, values.



- $x_1(s)$: Distance to teammate.
- $x_2(s)$: Distance to nearest opponent.
- $x_3(s)$: Largest open angle to goal.
- $x_4(s)$: Distance of teammate to goal.

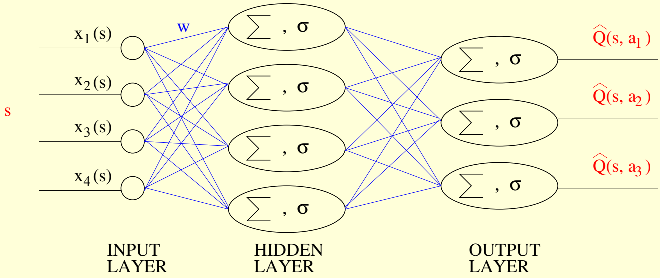
Compact Representation of $\hat{Q}$

- Illustration of $\hat{Q}$ approximated using a neural network.
- Input: (features of) state. One output for each action.
- Similar states will have similar Q-values.
- Can we learn weights w so that $\hat{Q}(s, a) \approx Q^{\star}(s, a)$?



Compact Representation of $\hat{Q}$

- Illustration of $\hat{Q}$ approximated using a neural network.
- Input: (features of) state. One output for each action.
- Similar states will have similar Q-values.
- Can we learn weights w so that $\hat{Q}(s, a) \approx Q^{\star}(s, a)$?



- Might not be able to represent $Q^{\star}$!
- Unlike supervised learning, convergence not obvious!
- Even if convergent, might induce sub-optimal behaviour!

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)

Prediction with a Linear Architecture

- Suppose we are to evaluate π on MDP (S, A, T, R, γ).
- Say we choose to approximate V^{π} by $\hat{V}$: for $s \in S$,

$$\hat{V}(w, s) = w \cdot x(s), \text{ where}$$

$x : S \to \mathbb{R}^d$ is a d-dimensional feature vector, and
$w \in \mathbb{R}^d$ is the weight/coefficient vector.

Prediction with a Linear Architecture

- Suppose we are to evaluate π on MDP (S, A, T, R, γ).
- Say we choose to approximate V^π by $\hat{V}$: for $s \in S$,

$$\hat{V}(w, s) = w \cdot x(s), \text{ where}$$

$x : S \to \mathbb{R}^d$ is a d-dimensional feature vector, and
$w \in \mathbb{R}^d$ is the weight/coefficient vector.

- Usually $d \ll |S|$.
- Illustration with $|S| = 3, d = 2$. Take $w = (w_1, w_2)$.

s	$V^\pi(s)$	$x_1(s)$	$x_2(s)$	$\hat{V}(w, s)$
s_1	7	2	-1	$2w_1 - w_2$
s_2	2	4	0	$4w_1$
s_3	-4	2	3	$2w_1 + 3w_2$

The Best Approximation

s	$V^\pi(s)$	$x_1(s)$	$x_2(s)$	$\hat{V}(w, s)$
s_1	7	2	-1	$2w_1 - w_2$
s_2	2	4	0	$4w_1$
s_3	-4	2	3	$2w_1 + 3w_2$

- Observe that for all $w \in \mathbb{R}^2$, $\hat{V}(w, s_2) = \frac{3\hat{V}(w,s_1)+\hat{V}(w,s_3)}{2}$.
- In general, $\hat{V}$ cannot be made equal to V^π.

The Best Approximation

s	$V^\pi(s)$	$x_1(s)$	$x_2(s)$	$\hat{V}(w, s)$
s_1	7	2	-1	$2w_1 - w_2$
s_2	2	4	0	$4w_1$
s_3	-4	2	3	$2w_1 + 3w_2$

- Observe that for all $w \in \mathbb{R}^2$, $\hat{V}(w, s_2) = \frac{3\hat{V}(w,s_1)+\hat{V}(w,s_3)}{2}$.
- In general, $\hat{V}$ cannot be made equal to V^π.
- Which w provides the best approximation?

The Best Approximation

s	$V^{\pi}(s)$	$x_1(s)$	$x_2(s)$	$\hat{V}(w, s)$
s_1	7	2	-1	$2w_1 - w_2$
s_2	2	4	0	$4w_1$
s_3	-4	2	3	$2w_1 + 3w_2$

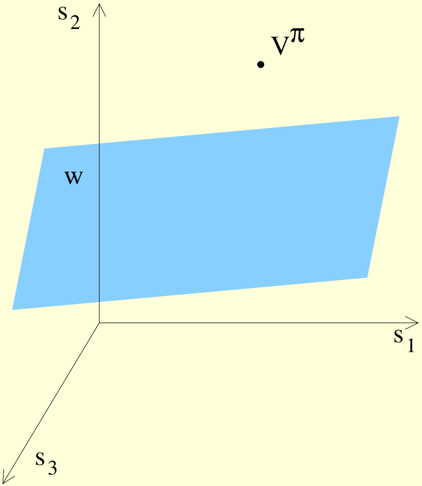
- Observe that for all $w \in \mathbb{R}^2$, $\hat{V}(w, s_2) = \frac{3\hat{V}(w,s_1)+\hat{V}(w,s_3)}{2}$.
- In general, $\hat{V}$ cannot be made equal to V^{π}.
- Which w provides the best approximation?
- A common choice is

$$w^{\star} = \operatorname*{argmin}_{w \in \mathbb{R}^d} MSVE(w),$$

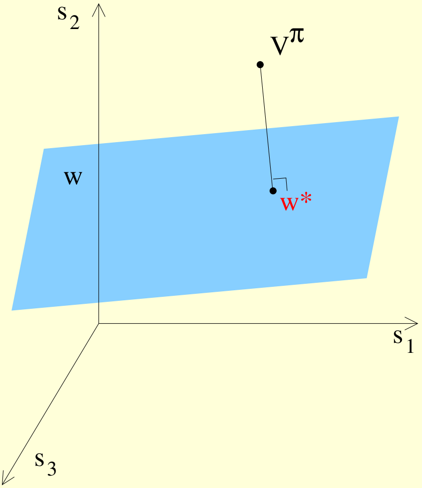
$$MSVE(w) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{s \in S} \mu^{\pi}(s)\{V^{\pi}(s) - \hat{V}(w, s)\}^2,$$

where $\mu^{\pi} : S \to [0, 1]$ is the stationary distribution of π.

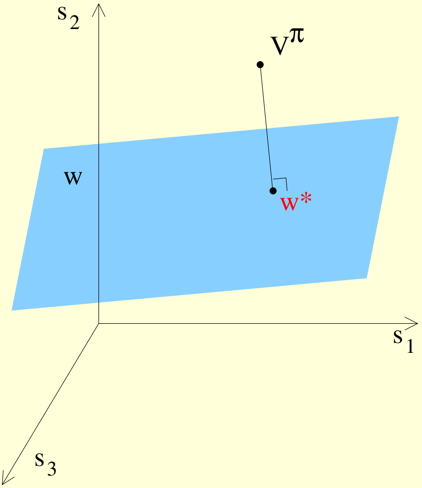
Geometric View



Geometric View

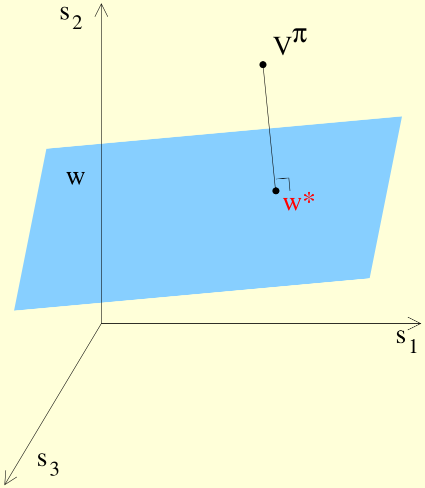


Geometric View



(Scaling based on μ^π not explicitly shown.)

Geometric View



(Scaling based on μ^{π} not explicitly shown.)

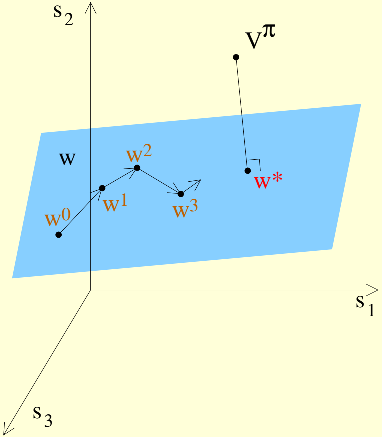
How to find $w^{\star}$?

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)

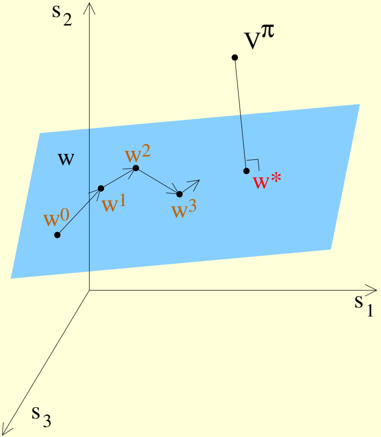
Gradient Descent

- Iteratively take steps in the w space in the direction minimising $MSVE(w)$.



Gradient Descent

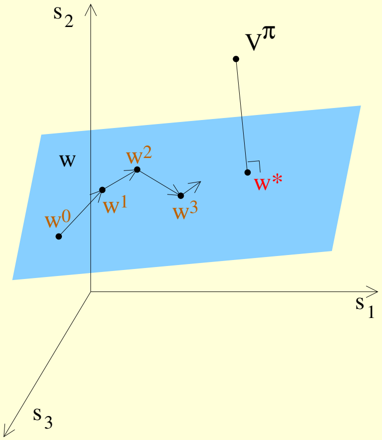
- Iteratively take steps in the w space in the direction minimising $MSVE(w)$.



- Feasible here?

Gradient Descent

- Iteratively take steps in the w space in the direction minimising $MSVE(w)$.



- Feasible here? Sort of.

Gradient Descent

- Initialise $w^0 \in \mathbb{R}^d$ arbitrarily. For $t \geq 0$ update as

$$w^{t+1} \leftarrow w^t - \alpha_{t+1} \nabla_w \left(\frac{1}{2} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \}^2 \right)$$

$$= w^t + \alpha_{t+1} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \} \nabla_w \hat{V}(w^t, s).$$

Gradient Descent

- Initialise $w^0 \in \mathbb{R}^d$ arbitrarily. For $t \geq 0$ update as

$$w^{t+1} \leftarrow w^t - \alpha_{t+1} \nabla_w \left(\frac{1}{2} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \}^2 \right)$$
$$= w^t + \alpha_{t+1} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \} \nabla_w \hat{V}(w^t, s).$$

- But we don't know $\mu^\pi(s)$, $V^\pi(s)$ for all $s \in S$. We're learning, remember?

Gradient Descent

- Initialise $w^0 \in \mathbb{R}^d$ arbitrarily. For $t \geq 0$ update as

$$w^{t+1} \leftarrow w^t - \alpha_{t+1} \nabla_w \left(\frac{1}{2} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \}^2 \right)$$
$$= w^t + \alpha_{t+1} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \} \nabla_w \hat{V}(w^t, s).$$

- But we don't know $\mu^\pi(s)$, $V^\pi(s)$ for all $s \in S$. We're learning, remember?
- Luckily, stochastic gradient descent allows us to update as

$$w^{t+1} \leftarrow w^t + \alpha_{t+1} \{ V^\pi(s^t) - \hat{V}(w^t, s^t) \} \nabla_w \hat{V}(w^t, s^t)$$

since $s^t \sim \mu^\pi$ anyway (as $t \to \infty$).

Gradient Descent

- Initialise $w^0 \in \mathbb{R}^d$ arbitrarily. For $t \geq 0$ update as

$$w^{t+1} \leftarrow w^t - \alpha_{t+1} \nabla_w \left(\frac{1}{2} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \}^2 \right)$$
$$= w^t + \alpha_{t+1} \sum_{s \in S} \mu^\pi(s) \{ V^\pi(s) - \hat{V}(w^t, s) \} \nabla_w \hat{V}(w^t, s).$$

- But we don't know $\mu^\pi(s)$, $V^\pi(s)$ for all $s \in S$. We're learning, remember?
- Luckily, stochastic gradient descent allows us to update as

$$w^{t+1} \leftarrow w^t + \alpha_{t+1} \{ V^\pi(s^t) - \hat{V}(w^t, s^t) \} \nabla_w \hat{V}(w^t, s^t)$$

  since $s^t \sim \mu^\pi$ anyway (as $t \to \infty$).
- But still, we don't know $V^\pi(s^t)$! What to do?

Gradient Descent

- Although we cannot perform update

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{V^\pi(s^t) - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

we can do

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_{t:\infty} - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

since $\mathbb{E}[G_{t:\infty}] = V^\pi(s^t)$.

Gradient Descent

- Although we cannot perform update

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{V^\pi(s^t) - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

we can do

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_{t:\infty} - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

since $\mathbb{E}[G_{t:\infty}] = V^\pi(s^t)$.

- In practice, we also do

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_t^\lambda - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

for $\lambda < 1$, even if $\mathbb{E}[G_t^\lambda] \neq V^\pi(s^t)$ in general.

Gradient Descent

- Although we cannot perform update

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{V^\pi(s^t) - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

  we can do

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_{t:\infty} - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

  since $\mathbb{E}[G_{t:\infty}] = V^\pi(s^t)$.
- In practice, we also do

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_t^\lambda - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$

  for $\lambda < 1$, even if $\mathbb{E}[G_t^\lambda] \neq V^\pi(s^t)$ in general. For example, Linear TD(0) performs the update

$$\boxed{w^{t+1} \leftarrow w^t + \alpha_{t+1}\{r^t + \gamma w^t \cdot x(s^{t+1}) - w^t \cdot x(s^t)\}x(s^t).}$$

Gradient Descent

- Although we cannot perform update
$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{V^\pi(s^t) - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$
we can do
$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_{t:\infty} - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$
since $\mathbb{E}[G_{t:\infty}] = V^\pi(s^t)$.
- In practice, we also do
$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{G_t^\lambda - \hat{V}(w^t, s^t)\}\nabla_w \hat{V}(w^t, s^t),$$
for $\lambda < 1$, even if $\mathbb{E}[G_t^\lambda] \neq V^\pi(s^t)$ in general. For example, Linear TD(0) performs the update
$$\boxed{w^{t+1} \leftarrow w^t + \alpha_{t+1}\{r^t + \gamma w^t \cdot x(s^{t+1}) - w^t \cdot x(s^t)\}x(s^t).}$$
- For $\lambda < 1$, the process is not true gradient descent. But it still converges with linear function approximation.

Linear TD(λ) algorithm

- Maintains an eligibility trace $z \in \mathbb{R}^d$.
- Recall that $\hat{V}(w, s) = w \cdot x(s)$, hence $\nabla_w \hat{V}(w, s) = x(s)$.

Linear TD(λ) algorithm

- Maintains an eligibility trace $z \in \mathbb{R}^d$.
- Recall that $\hat{V}(w, s) = w \cdot x(s)$, hence $\nabla_W \hat{V}(w, s) = x(s)$.

Initialise $w \in \mathbb{R}^d$ arbitrarily.
Repeat for each episode:
    Set $z \to \mathbf{0}$.//Eligibility trace vector.
    Assume the agent is born in state s.
    Repeat for each step of episode:
        Take action a; obtain reward r, next state s'.
        $\delta \leftarrow r + \gamma \hat{V}(w, s') - \hat{V}(w, s)$.
        $z \leftarrow \gamma\lambda z + \nabla_w \hat{V}(w, s)$.
        $w \leftarrow w + \alpha\delta z$.
        $s \leftarrow s'$.

Linear TD(λ) algorithm

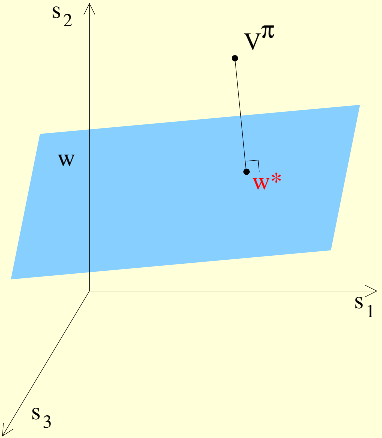
- Maintains an eligibility trace $z \in \mathbb{R}^d$.
- Recall that $\hat{V}(w, s) = w \cdot x(s)$, hence $\nabla_W \hat{V}(w, s) = x(s)$.

> Initialise $w \in \mathbb{R}^d$ arbitrarily.
> Repeat for each episode:
>   Set $z \to \mathbf{0}$.//Eligibility trace vector.
>   Assume the agent is born in state s.
>   Repeat for each step of episode:
>   Take action a; obtain reward r, next state s'.
> $\delta \leftarrow r + \gamma \hat{V}(w, s') - \hat{V}(w, s)$.
> $z \leftarrow \gamma \lambda z + \nabla_w \hat{V}(w, s)$.
> $w \leftarrow w + \alpha \delta z$.
> $s \leftarrow s'$.

- See Sutton and Barto (2018) for variations (accumulating, replacing, and dutch traces).

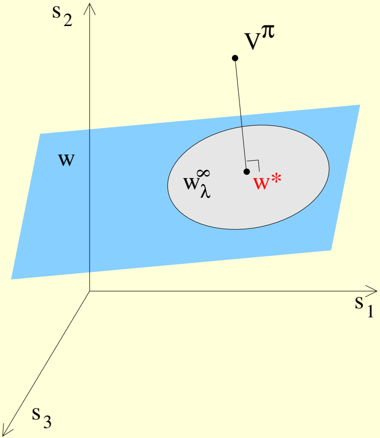
Convergence of Linear TD(λ)

$$\boxed{MSVE(w_\lambda^\infty) \leq \frac{1 - \gamma\lambda}{1 - \gamma} MSVE(w^\star).}$$



Convergence of Linear TD(λ)

$$MSVE(w_\lambda^\infty) \leq \frac{1 - \gamma\lambda}{1 - \gamma} MSVE(w^\star).$$

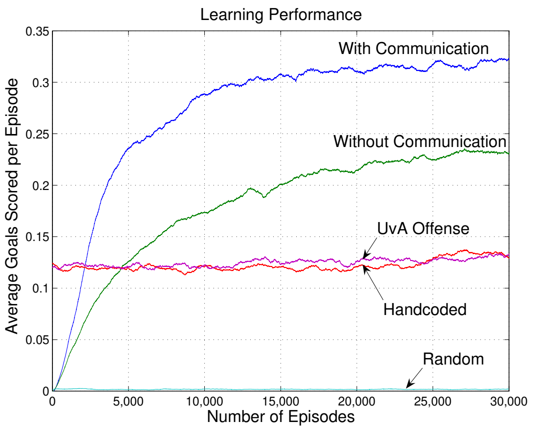


Control with Linear Function Approximation

- Linear function approximation is implemented in the control by approximating $Q(s, a) \approx w \cdot x(s, a)$.
- Linear Sarsa(λ) is a very popular algorithm.

RL on Half Field Offense

- Uses Linear Sarsa(0) with tile coding.



Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study. Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone. RoboCup 2006: Robot Soccer World Cup X, pp. 72–85, Springer, 2007.

Reinforcement Learning

1. Multi-step returns
2. TD(λ)
3. Generalisation and Function Approximation
4. Linear function approximation
5. Linear TD(λ)