CS 747, Autumn 2022: Lecture 17

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2022

Reinforcement Learning

1. Generalisation and function approximation
2. Linear function approximation
3. 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

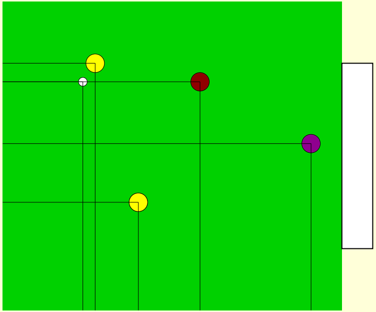
- Decision-making restricted to offense player with ball.
- Based on state, choose among DRIBBLE, PASS, SHOOT.
- How many states are there? An infinite number!

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?

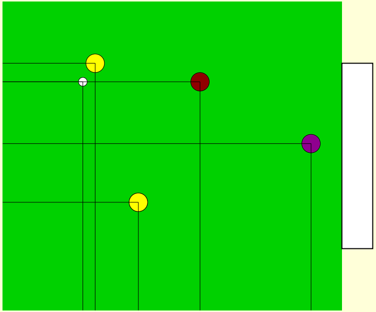
Features

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



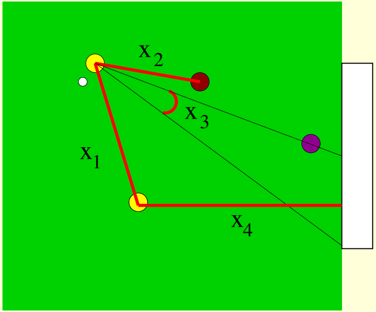
Features

- 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

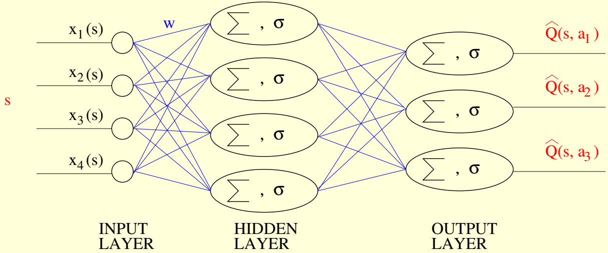
- State s is defined by positions and velocities of players, ball.
- Velocities might not be important for decision making.
- Position coordinates might not generalise well.
- Define features $x : S \to \mathbb{R}$. Idea is that states with similar features will have similar consequences of actions,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

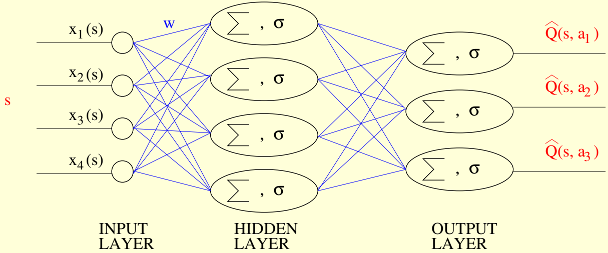
Compact Representation of $\hat{Q}$

- 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. Generalisation and function approximation
2. Linear function approximation
3. 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	$2w_1 + 3w_2$

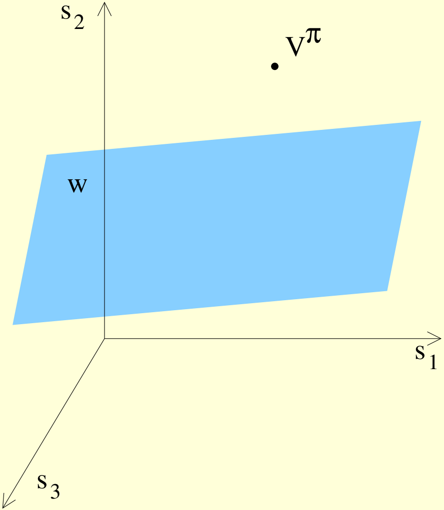
- Observe that for all $w \in \mathbb{R}^2$, $\hat{V}(w, s_2) = \frac{3\hat{V}(w,s_1)+\hat{V}(w,s_3)}{2}$.
- In general, $\hat{V}$ cannot be made equal to V^π.
- Which w provides the best approximation?
- A common choice is

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

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

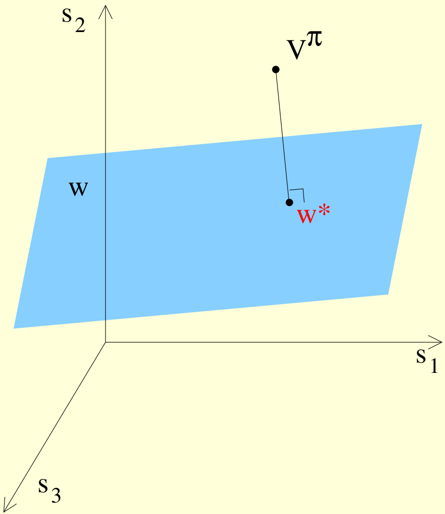
where $\mu^\pi : S \to [0, 1]$ is the stationary distribution of π.

Geometric View



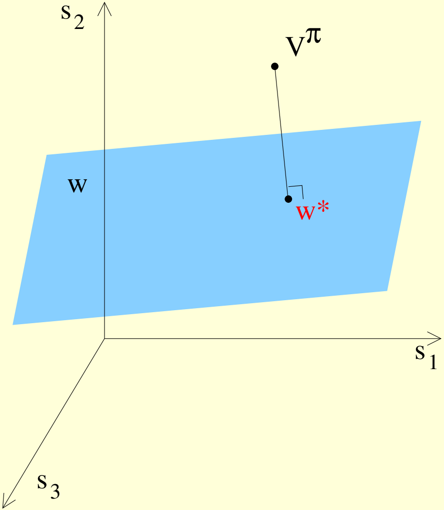
(μ^{π}-scaling not explicitly shown.)

Geometric View



(μ^{π}-scaling not explicitly shown.)

Geometric View



(μ^π-scaling not explicitly shown.)

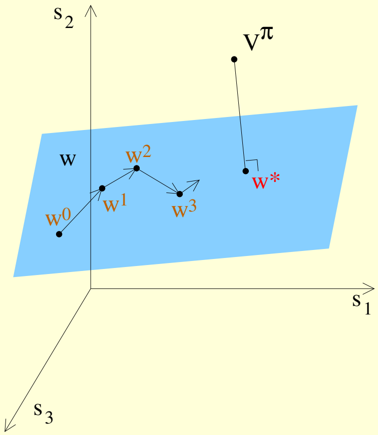
How to find $w^\star$?

Reinforcement Learning

1. Generalisation and function approximation
2. Linear function approximation
3. Linear TD(λ)

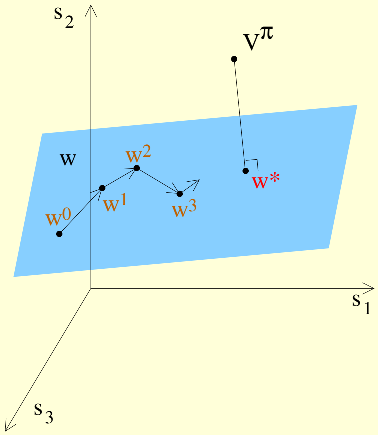
Gradient Descent

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



Gradient Descent

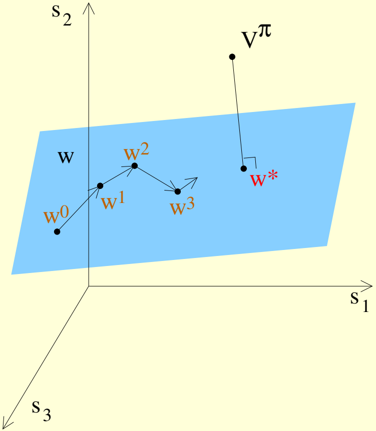
- Iteratively take steps in the w space in the direction minimising $MSVE(w)$.



- 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

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

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

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

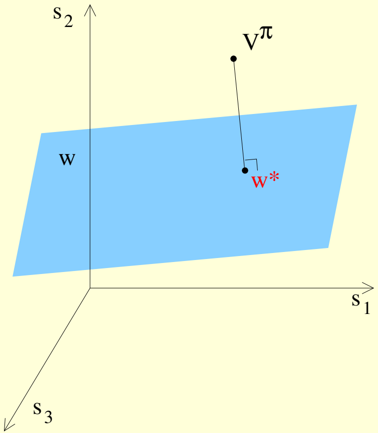
- Maintains an eligibility trace $z \in \mathbb{R}^d$.
- Recall that $\hat{V}(w, s) = w \cdot x(s)$, hence $\nabla_w \hat{V}(w, s) = x(s)$.

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).

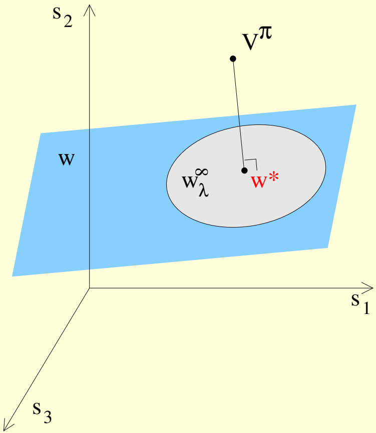
Convergence of Linear TD(λ)

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



Convergence of Linear TD(λ)

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

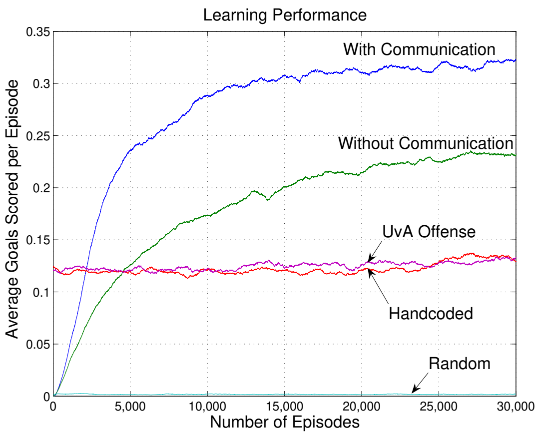


Control with Linear Function Approximation

- 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. Generalisation and function approximation
2. Linear function approximation
3. Linear TD(λ)