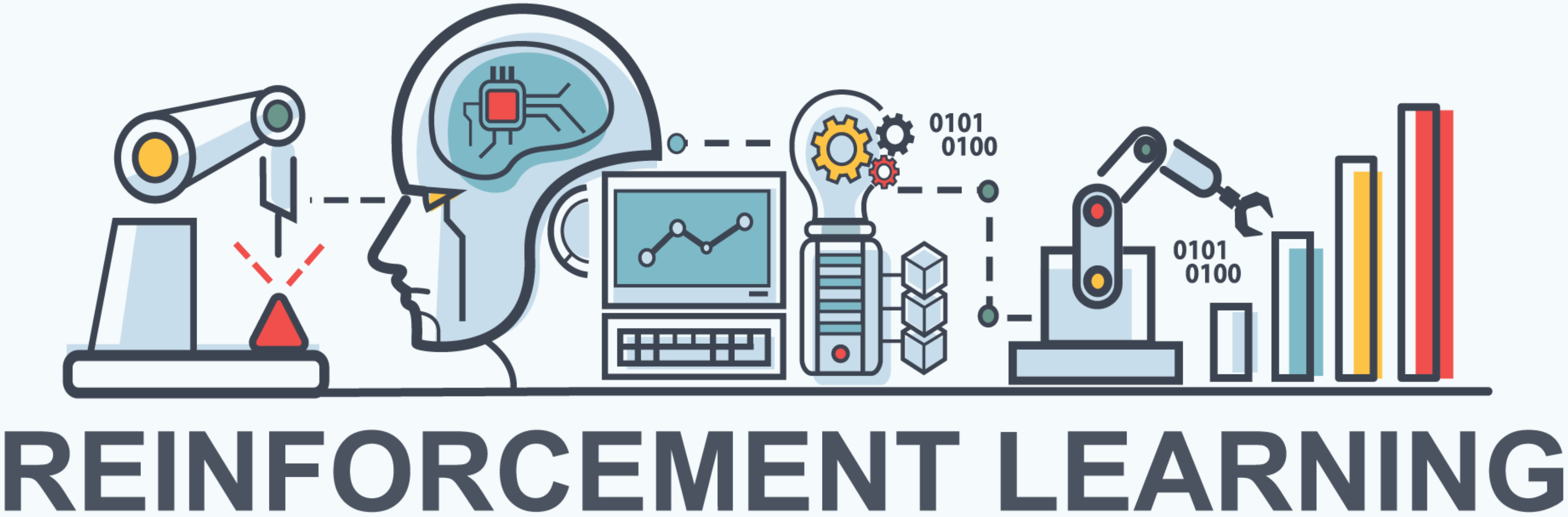


CP8319/CPS824
Lecture 13

Instructor: Nariman Farsad

* Some of the slides in this deck are adopted from courses offered David Silver, Emma Brunskill, and Sergey Levine.

Today’s Agenda

1. **Review of Previous Lectures**

2. Finish Value Function Approximation Policy Evaluation

3. Value Function Approximation for Control

Value Function Approximation

- So far we have represented value function by a lookup table
  - Every state s has an entry $v(s)$
  - Or every state-action pair s, a has an entry $Q(s, a)$

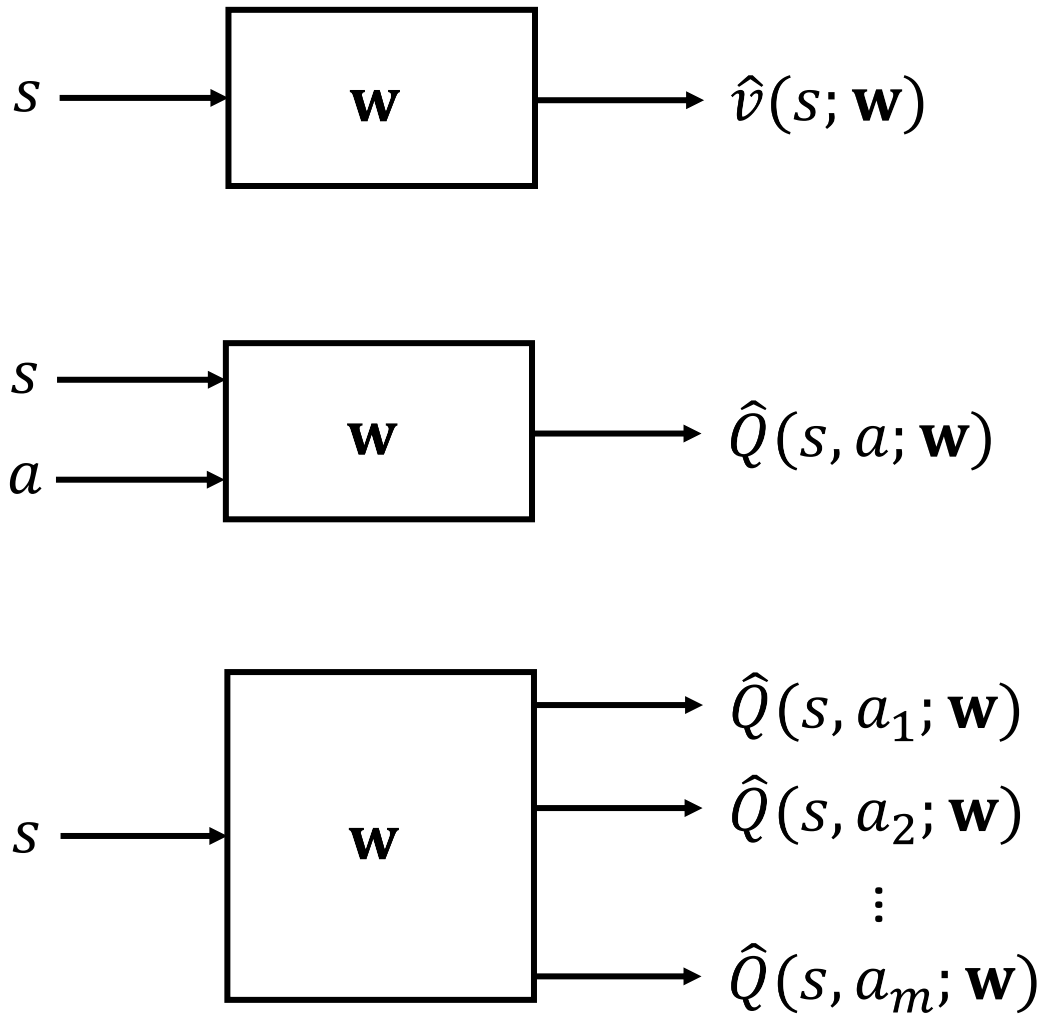
- Problem with large MDPs:
  - There are too many states and/or actions to store in memory
  - It is too slow to learn the value of each state individually

- Solution for large MDPs:
  - Estimate value function with *function approximation*

$$v^{\pi}(s) \approx \hat{v}(s; \mathbf{w})$$
$$\text{or } Q^{\pi} \approx \hat{Q}(s, a; \mathbf{w})$$

  - *Generalize* from seen states to unseen states
  - Update parameter $\mathbf{w}$ using MC or TD learning

Types of Value Function Approximation



Value Function Approx. By Stochastic Gradient Descent

- Goal: Find the parameter vector $\mathbf{w}$ that minimizes the loss between a true value function $v^{\pi}(s)$ and its approximation $\hat{v}(s;\mathbf{w})$ as represented with a particular function/model class parameterized by $\mathbf{w}$.
- Generally, use mean squared error and define the loss as

$$J(\mathbf{w}) = \mathbb{E}_{\pi}\left[\left(v^{\pi}(s) - \hat{v}(s;\mathbf{w})\right)^2\right]$$

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{1}{2}\alpha\nabla_{\mathbf{w}}J(\mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\mathbb{E}_{\pi}\left[\left(v^{\pi}(s) - \hat{v}(s;\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{v}(s;\mathbf{w})\right]$$

- Stochastic gradient descent (SGD) *samples* the gradient:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(v^{\pi}(s) - \hat{v}(s;\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{v}(s;\mathbf{w})$$

- Expected update is equal to full gradient update

Feature Vectors

- The state is represented by a *feature vector*

$$\mathbf{x}(s) = \begin{pmatrix} x_1(s) \\ \vdots \\ x_n(s) \end{pmatrix}$$

- For example:
  - Distance of a robot from "landmarks"
  - Trends in the stock market
  - Piece and pawn configurations in chess

Linear Value Function Approximation With Oracle

- Represent a value function (or state-action value function) for a particular policy with a weighted linear combination of features:

$$\hat{v}(s;\mathbf{w}) = \sum_{j=0}^{n} x_j(s) w_j = \mathbf{x}(s)^T \mathbf{w}$$

- Objective function is:

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[\left(v^\pi(s) - \hat{v}(s;\mathbf{w})\right)^2\right]$$

- Update rule is:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(v^\pi(s) - \hat{v}(s;\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{v}(s;\mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(v^\pi(s) - \hat{v}(s;\mathbf{w})\right)\mathbf{x}(s)$$

MC Linear VFA for Policy Evaluation

Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$

loop

  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,L_k})$ given π

  for $t = 1, \ldots, L_k$ **do**

    if First visit to (s) in episode k **then**

      $G_t(s) = \sum_{j=t}^{L_k} r_{k,j}$

      Update weights:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(G_t(s) - \mathbf{x}(s)^T \mathbf{w})\mathbf{x}(s)$$

    end if

  end for

  $k = k + 1$

end loop

Today's Agenda

1. Review of Previous Lectures

2. **Finish Value Function Approximation Policy Evaluation**

3. Value Function Approximation for Control

Recall: Temporal Difference Learning w/ Lookup Table

- Uses bootstrapping and sampling to approximate $v^{\pi}(s)$
- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $v^{\pi}(s_t)$ toward estimated return $r_t + \gamma v^{\pi}(s_{t+1})$

$$v^{\pi}(s_t) = v^{\pi}(s_t) + \alpha([r_t + \gamma v^{\pi}(s_{t+1})] - v^{\pi}(s_t))$$

- $r_t + \gamma v^{\pi}(s_{t+1})$ is called the ***TD target***
- $\delta_t = r_t + \gamma v^{\pi}(s_{t+1}) - v^{\pi}(s_t)$ is called the ***TD error***
- a biased estimate of the true value $v^{\pi}(s)$
- Represent value for each state with a separate table entry

Temporal Difference (TD(0)) Learning with VFA

- In value function approximation, the target is $r + \gamma \hat{v}^{\pi}(s'; \mathbf{w})$, a biased and approximated estimate of the true value $v^{\pi}(s)$

- Can reduce doing TD(0) learning with value function approximation to supervised learning on a set of data pairs:

$$\langle s_1, r_1 + \gamma \hat{v}^{\pi}(s_2; \mathbf{w}) \rangle, \langle s_2, r_2 + \gamma \hat{v}^{\pi}(s_3; \mathbf{w}) \rangle, \ldots$$

- Find weights to minimize mean squared error:

$$J(\mathbf{w}) = \mathbb{E}_{\pi}\left[\left(r_j + \gamma \hat{v}^{\pi}(s_{j+1}; \mathbf{w}) - \hat{v}(s_j; \mathbf{w})\right)^2\right]$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(r_j + \gamma \mathbf{x}(s_{j+1})^T \mathbf{w} - \mathbf{x}(s_j)^T \mathbf{w}\right) \mathbf{x}(s_j)$$

- Therefore, we are suing 3 forms of approximation, what are they?

TD(0) Linear VFA for Policy Evaluation

1: Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$
2: **loop**
3: Sample tuple (s_k, a_k, r_k, s_{k+1}) given π
4: Update weights:

$$\boldsymbol{w} = \boldsymbol{w} + \alpha(r + \gamma \boldsymbol{x}(s')^T \boldsymbol{w} - \boldsymbol{x}(s)^T \boldsymbol{w})\boldsymbol{x}(s)$$

5: $k = k + 1$
6: **end loop**

TD(0) Linear VFA for Policy Evaluation Example

Recall MC update: $\alpha(G_t - \mathbf{x}(s)^T\mathbf{w})\mathbf{x}(s)$

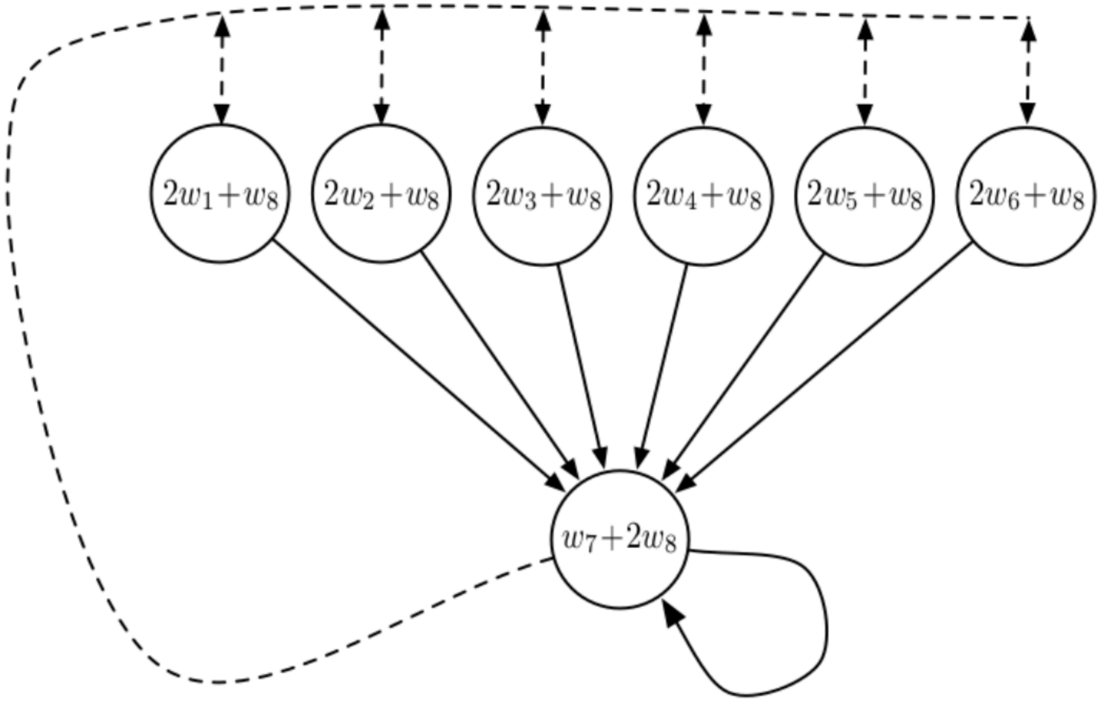
Two actions:

- a_1 goes to 7 (solid line)
- a_2 goes to 1-6 with 1/6 probability (dashed line)

Observe $(s_1, a_1, 0, s_7)$

Assume $\mathbf{w}_0 = [1,1,1,1,1,1,1,1], \alpha = 0.5, \gamma = 0.9$

What is $\mathbf{w}_1$ after the first SGD update?



TD(0) Linear VFA for Policy Evaluation Example

Recall MC update: $\alpha(G_t - \mathbf{x}(s)^T \mathbf{w})\mathbf{x}(s)$

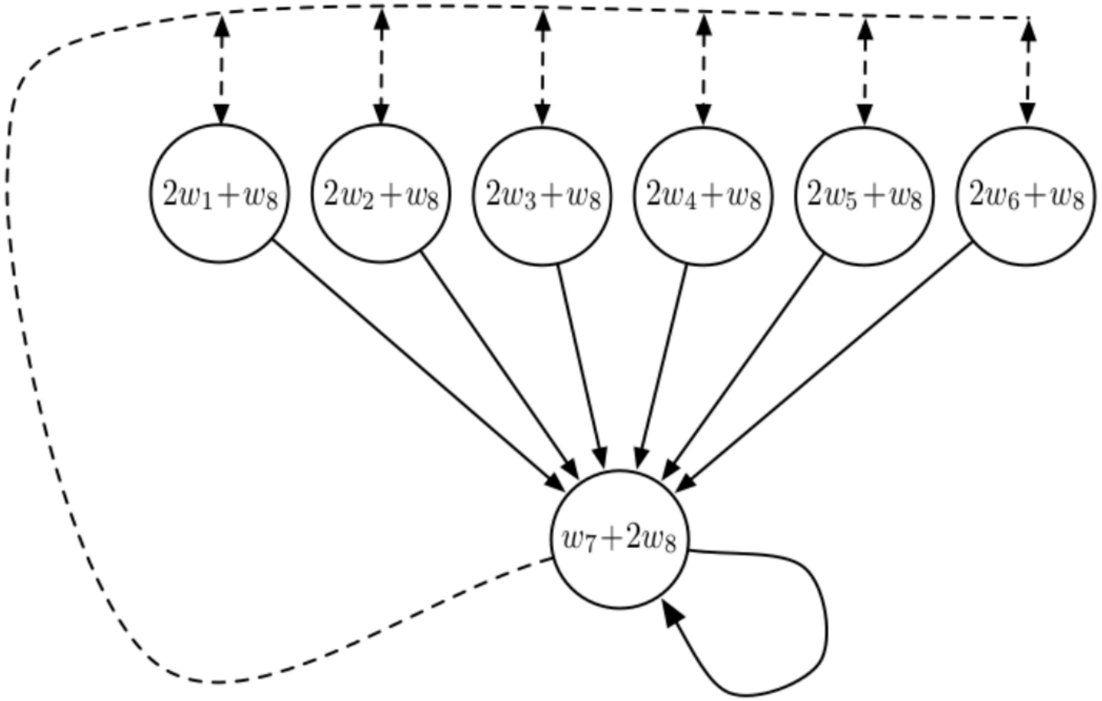
Two actions:

- a_1 goes to 7 (solid line)
- a_2 goes to 1-6 with 1/6 probability (dashed line)

Observe $(s_1, a_1, 0, s_7)$

Assume $\mathbf{w}_0 = [1,1,1,1,1,1,1,1], \alpha = 0.5, \gamma$ =0.9

What is $\mathbf{w}_1$ after the first SGD update?



Convergence Rates for Linear Value Function Approximation for Policy Evaluation

- Does TD or MC converge faster to a fixed point?
- Not (to my knowledge) definitively understood
- Practically TD learning often converges faster to its fixed value function approximation point

Today’s Agenda

1. Review of Previous Lectures

2. Finish Value Function Approximation Policy Evaluation

3. **Value Function Approximation for Control**

Control using Value Function Approximation

- Use value function approximation to represent state-action values $\hat{Q}^{\pi}(s, a; \mathbf{w}) \approx Q^{\pi}(s, a)$

- Control using value function approximation involves repeating these steps:
  - Approximate policy evaluation using value function approximation of $\hat{Q}^{\pi}(s, a; \mathbf{w})$
  - Perform ϵ-greedy policy improvement

- Can be unstable. Involves intersection of the following:
  - Function approximation
  - Bootstrapping
  - Sampling
  - Off-policy learning (which can cause problems as we will see)

State Action VFA with an Oracle

- Approximate the action-value function:

$$\hat{Q}^{\pi}(s, a; \mathbf{w}) \approx Q^{\pi}(s, a)$$

- Minimise mean-squared error between approximate action-value function $\hat{Q}^{\pi}(s, a; \mathbf{w})$ and the true action-value function $Q^{\pi}(s, a)$

$$J(\mathbf{w}) = \mathbb{E}_{\pi}\left[\left(Q^{\pi}(s, a) - \hat{Q}(s, a; \mathbf{w})\right)^2\right]$$

- Use stochastic gradient descent to find a local minimum with the update rule

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left(Q^{\pi}(s, a) - \hat{Q}(s, a; \mathbf{w})\right)\nabla_{\mathbf{w}}\hat{Q}(s, a; \mathbf{w})$$

Linear State Action VFA with an Oracle

- Use features to represent both the state and action:

$$\mathbf{x}(s,a) = \begin{pmatrix} x_1(s,a) \\ \vdots \\ x_n(s,a) \end{pmatrix}$$

- Represent state-action value function with a weighted linear combination of features

$$\hat{Q}(s,a;\mathbf{w}) = \sum_{j=0}^{n} x_j(s,a) w_j = \mathbf{x}(s,a)^T \mathbf{w}$$

- Use stochastic gradient descent to find a local minimum with the update rule

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(Q^{\pi}(s,a) - \hat{Q}(s,a;\mathbf{w}) \right) \mathbf{x}(s,a)$$

Model-Free Control Approaches Using VFA

- In practice, there is no oracle
- In Monte Carlo methods, use a return G_t from the episode as a substitute target

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(G_t - \hat{Q}(s, a; \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

- For SARSA use a TD target $r + \gamma \hat{Q}(s', a'; \mathbf{w})$ which leverages the current VFA

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(r + \gamma \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

- For Q-learning use a TD target $r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w})$ which leverages the max of the current VFA

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

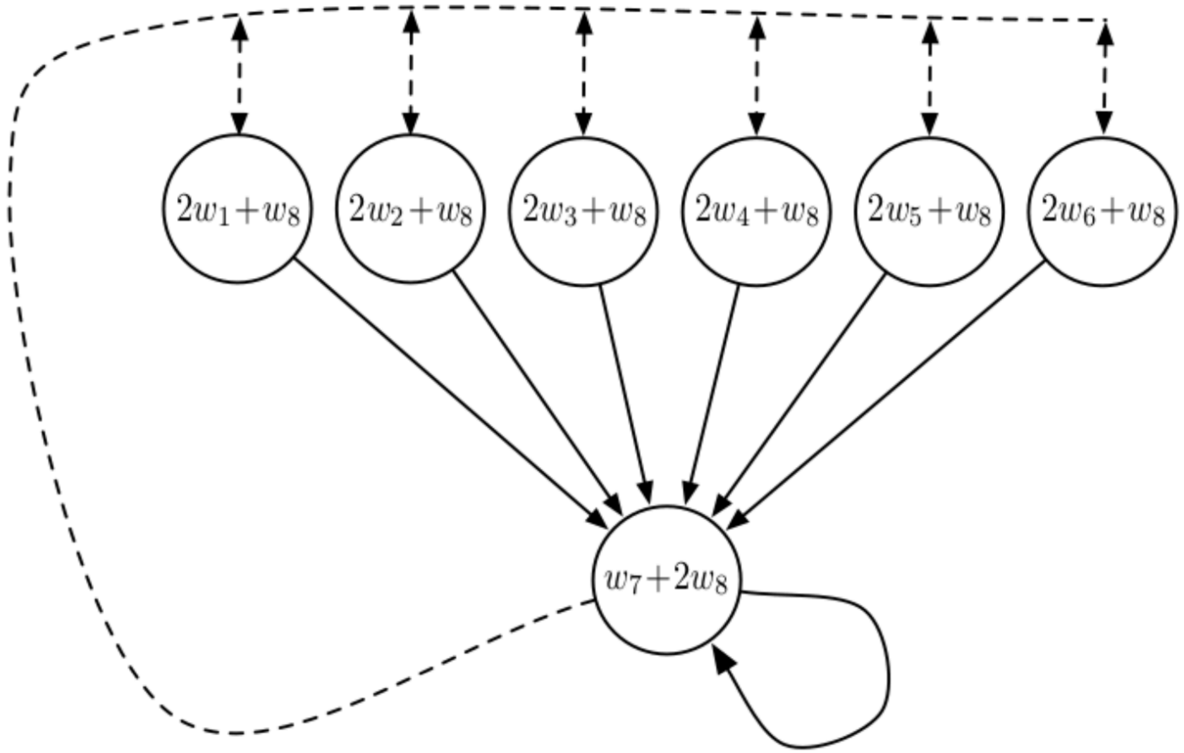
Convergence of TD Methods with VFA

- Informally, updates involve doing an (approximate) Bellman backup followed by best trying to fit underlying value function to a particular feature representation

- Bellman operators are contractions, but value function approximation fitting can be an expansion

- Convergence is not guaranteed

Challenges of Off Policy Control: Baird Example



$\pi(\text{solid}|\cdot) = 1$

$\mu(\text{dashed}|\cdot) = 6/7$

$\mu(\text{solid}|\cdot) = 1/7$

$\gamma = 0.99$

- Behavior policy and target policy are not identical
- Value can diverge

Convergence of Control Algorithms

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function