

Function Approximation

Control using VFA

Marius Lindauer



Winter Term 2021

Control using Value Function Approximation

- ▶ Use value function approximation to represent state-action values $\hat{Q}^{\pi}(s, a; \vec{w}) \approx Q^{\pi}$
- ▶ Interleave
 - ▶ Approximate policy evaluation using value function approximation
 - ▶ Perform ϵ -greedy policy improvement
- ▶ Can be unstable. Generally involves intersection of the following:
 - ▶ Function approximation
 - ▶ Bootstrapping
 - ▶ Off-policy learning

Action-Value Function Approximation with an Oracle

- ▶ $\hat{Q}^{\pi}(s, a; \vec{w}) \approx Q^{\pi}$
- ▶ Minimize the mean-squared error between the true action-value function $Q^{\pi}(s, a)$ and the approximate action-value function:

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

- ▶ Use stochastic gradient descent to find a local minimum

$$\begin{aligned} -\frac{1}{2} \nabla_{\vec{w}} J(\vec{w}) &= \mathbb{E} \left[(Q^{\pi}(s, a) - \hat{Q}^{\pi}(s, a; \vec{w})) \nabla_{\vec{w}} \hat{Q}^{\pi}(s, a; \vec{w}) \right] \\ \Delta \vec{w} &= -\frac{1}{2} \alpha \nabla_{\vec{w}} J(\vec{w}) \end{aligned}$$

- ▶ Stochastic gradient descent (SGD) samples the gradient

Linear State Action Value Function Approximation with an Oracle

- Use features to represent both the state and action

$$\vec{x}(s, a) = \begin{pmatrix} \vec{x}_1(s, a) \\ \vec{x}_2(s, a) \\ \dots \\ \vec{x}_n(s, a) \end{pmatrix}$$

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

$$\hat{Q}(s, a; \vec{w}) = \vec{x}(s, a)^T \vec{w} = \sum_{j=1}^n x_j(s, a) w_j$$

- Stochastic gradient descent update

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

Incremental Model-Free Control Approaches

- ▶ Similar to policy evaluation, true state-action value function for a state is unknown and so substitute a target value
- ▶ In Monte Carlo methods, use a return G_t as a substitute target

$$\Delta \vec{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \vec{w})) \nabla_{\vec{w}} \hat{Q}(s_t, a_t; \vec{w})$$

- ▶ For SARSA instead use a TD target $r + \gamma \hat{Q}(s', a'; \vec{w})$ which leverages the current function approximations value

$$\Delta \vec{w} = \alpha(r + \gamma \hat{Q}(s', a'; \vec{w}) - \hat{Q}(s, a; \vec{w})) \nabla_{\vec{w}} \hat{Q}(s, a; \vec{w})$$

- ▶ For Q-learning instead use a TD target $r + \gamma \max_{a'} \hat{Q}(s', a'; \vec{w})$ which leverages the max of the current function approximations value

$$\Delta \vec{w} = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \vec{w}) - \hat{Q}(s, a; \vec{w})) \nabla_{\vec{w}} \hat{Q}(s, a; \vec{w})$$