An Introduction to Reinforcement Learning

Part II: Classical Reinforcement Learning

Christoph Belak

Technische Universität Berlin

Stochastic Numerics and Statistical Learning Workshop
KAUST, May 23, 2022

Preliminary Discussion

We want to solve the **Markov Decision Process**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big].$$

We want to solve the **Markov Decision Process**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big].$$

One possibility to approach this is via the **Dynamic Programming Principle**

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s' \in \mathbb{S}} \mathcal{V}(s') p(s'|s,a) \Big].$$

We want to solve the **Markov Decision Process**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

One possibility to approach this is via the **Dynamic Programming Principle**

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s' \in \mathbb{S}} \mathcal{V}(s') p(s'|s,a) \Big].$$

Challenges:

- The DPP might be quite costly to solve

We want to solve the **Markov Decision Process**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

One possibility to approach this is via the **Dynamic Programming Principle**

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s' \in \mathbb{S}} \mathcal{V}(s') p(s'|s,a) \Big].$$

Challenges:

- The DPP might be quite costly to solve
- We may not know the true transition kernel p

We want to solve the **Markov Decision Process**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

One possibility to approach this is via the **Dynamic Programming Principle**

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s' \in \mathbb{S}} \mathcal{V}(s') p(s'|s,a) \Big].$$

Challenges:

- The DPP might be quite costly to solve
- We may not know the true transition kernel p

But maybe:

- We have a way to simulate the state process without knowing p

We want to solve the **Markov Decision Process**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

One possibility to approach this is via the **Dynamic Programming Principle**

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s' \in \mathbb{S}} \mathcal{V}(s') p(s'|s,a) \Big].$$

Challenges:

- The DPP might be quite costly to solve
- We may not know the true transition kernel p

But maybe:

- We have a way to simulate the state process without knowing p
- Or we at least have a lot of data available

Before moving on, we need some notation. Recall the (optimal) **value function**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big].$$

Before moving on, we need some notation. Recall the (optimal) **value function**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

Given a policy π, we define the **value function of π** as

$$\mathcal{V}_\pi(s) = \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

Before moving on, we need some notation. Recall the (optimal) **value function**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

Given a policy π, we define the **value function of π** as

$$\mathcal{V}_\pi(s) = \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big].$$

Moreover, a useful tool for MDPs is the value function of π conditioned on the first action being fixed. Formally, we refer to

$$Q_\pi(s,a) \triangleq \mathbb{E}_{s,a} \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big) \Big]$$

as the **Q-function of π**

Before moving on, we need some notation. Recall the (optimal) **value function**

$$\mathcal{V}(s) = \sup_{\pi} \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big].$$

Given a policy π, we define the **value function of π** as

$$\mathcal{V}_{\pi}(s) = \mathbb{E}_s \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big].$$

Moreover, a useful tool for MDPs is the value function of π conditioned on the first action being fixed. Formally, we refer to

$$Q_{\pi}(s,a) \triangleq \mathbb{E}_{s,a} \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big]$$

as the **Q-function of π** and

$$Q(s,a) \triangleq \sup_{\pi} \mathbb{E}_{s,a} \Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi}, A_n^{\pi}\big) \Big]$$

as the (optimal) **Q-function**.

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_{\pi}(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_{\pi}(s,a).$$

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_{\pi}(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_{\pi}(s,a).$$

Moreover, it holds that

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} Q(s,a) \qquad \text{and} \qquad \mathcal{V}_{\pi}(s) = \sum_{a \in \mathbb{A}(s)} \pi(a|s) Q_{\pi}(s,a).$$

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_{\pi}(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_{\pi}(s,a).$$

Moreover, it holds that

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} Q(s,a) \qquad \text{and} \qquad \mathcal{V}_{\pi}(s) = \sum_{a \in \mathbb{A}(s)} \pi(a|s) Q_{\pi}(s,a).$$

Conclusions:

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_{\pi}(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_{\pi}(s,a).$$

Moreover, it holds that

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} Q(s,a) \qquad \text{and} \qquad \mathcal{V}_{\pi}(s) = \sum_{a \in \mathbb{A}(s)} \pi(a|s) Q_{\pi}(s,a).$$

Conclusions:

- Knowing Q, we can compute $\mathcal{V}$ and an optimal policy

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_\pi(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_\pi(s,a).$$

Moreover, it holds that

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} Q(s,a) \qquad \text{and} \qquad \mathcal{V}_\pi(s) = \sum_{a \in \mathbb{A}(s)} \pi(a|s) Q_\pi(s,a).$$

Conclusions:

- Knowing Q, we can compute $\mathcal{V}$ and an optimal policy
- To compute $\mathcal{V}$ (resp. Q) via $\mathcal{V}_\pi$ (resp. Q_π), we need to perform two things

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_\pi(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_\pi(s,a).$$

Moreover, it holds that

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} Q(s,a) \qquad \text{and} \qquad \mathcal{V}_\pi(s) = \sum_{a \in \mathbb{A}(s)} \pi(a|s) Q_\pi(s,a).$$

Conclusions:

- Knowing Q, we can compute $\mathcal{V}$ and an optimal policy
- To compute $\mathcal{V}$ (resp. Q) via $\mathcal{V}_\pi$ (resp. Q_π), we need to perform two things
  - **Prediction**: Compute $\mathcal{V}_\pi$ (resp. Q_π) for a given policy π

Obviously, we have

$$\mathcal{V}(s) = \sup_{\pi} \mathcal{V}_\pi(s) \qquad \text{and} \qquad Q(s,a) = \sup_{\pi} Q_\pi(s,a).$$

Moreover, it holds that

$$\mathcal{V}(s) = \max_{a \in \mathbb{A}(s)} Q(s,a) \qquad \text{and} \qquad \mathcal{V}_\pi(s) = \sum_{a \in \mathbb{A}(s)} \pi(a|s) Q_\pi(s,a).$$

Conclusions:

- Knowing Q, we can compute $\mathcal{V}$ and an optimal policy
- To compute $\mathcal{V}$ (resp. Q) via $\mathcal{V}_\pi$ (resp. Q_π), we need to perform two things
  - **Prediction**: Compute $\mathcal{V}_\pi$ (resp. Q_π) for a given policy π
  - **Control**: Find an optimal policy π^*

The Prediction Problem

First naive attempt at the prediction problem: **Monte Carlo**

$$\mathcal{V}_\pi(s) \approx v_\pi^M(s) \triangleq \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m})$$

First naive attempt at the prediction problem: **Monte Carlo**

$$\mathcal{V}_\pi(s) \approx v_\pi^M(s) \triangleq \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m})$$

The Monte Carlo estimate can be computed iteratively:

$$v_\pi^{M+1}(s) = \frac{1}{M+1} \sum_{m=1}^{M+1} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m})$$

First naive attempt at the prediction problem: **Monte Carlo**

$$\mathcal{V}_\pi(s) \approx v_\pi^M(s) \triangleq \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m})$$

The Monte Carlo estimate can be computed iteratively:

$$\begin{aligned} v_\pi^{M+1}(s) &= \frac{1}{M+1} \sum_{m=1}^{M+1} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m}) \\ &= \frac{M}{M+1} v_\pi^M(s) + \frac{1}{M+1} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,M+1}, A_n^{\pi,M+1}) \end{aligned}$$

First naive attempt at the prediction problem: **Monte Carlo**

$$\mathcal{V}_\pi(s) \approx v_\pi^M(s) \triangleq \frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m})$$

The Monte Carlo estimate can be computed iteratively:

$$\begin{aligned} v_\pi^{M+1}(s) &= \frac{1}{M+1} \sum_{m=1}^{M+1} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,m}, A_n^{\pi,m}) \\ &= \frac{M}{M+1} v_\pi^M(s) + \frac{1}{M+1} \sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,M+1}, A_n^{\pi,M+1}) \\ &= v_\pi^M(s) + \alpha_{M+1} \Big[\sum_{n=0}^{\infty} \beta^n r(S_n^{\pi,M+1}, A_n^{\pi,M+1}) - v_\pi^M(s) \Big] \end{aligned}$$

with $\alpha_{M+1} \triangleq 1/(M+1)$.

First naive attempt at the prediction problem: **Monte Carlo**

$$\mathcal{V}_\pi(s) \approx v_\pi^M(s) \triangleq \frac{1}{M}\sum_{m=1}^{M}\sum_{n=0}^{\infty}\beta^n r(S_n^{\pi,m}, A_n^{\pi,m})$$

The Monte Carlo estimate can be computed iteratively:

$$\begin{aligned} v_\pi^{M+1}(s) &= \frac{1}{M+1}\sum_{m=1}^{M+1}\sum_{n=0}^{\infty}\beta^n r(S_n^{\pi,m}, A_n^{\pi,m}) \\ &= \frac{M}{M+1}v_\pi^M(s) + \frac{1}{M+1}\sum_{n=0}^{\infty}\beta^n r(S_n^{\pi,M+1}, A_n^{\pi,M+1}) \\ &= v_\pi^M(s) + \alpha_{M+1}\Big[\sum_{n=0}^{\infty}\beta^n r(S_n^{\pi,M+1}, A_n^{\pi,M+1}) - v_\pi^M(s)\Big] \end{aligned}$$

with $\alpha_{M+1} \triangleq 1/(M+1)$. We write this more compactly as

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[\sum_{n=0}^{\infty}\beta^n r(S_n, A_n) - v_\pi(s)\Big].$$

Summing up: For naive Monte Carlo, we estimate

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big)\Big]$$

by a rule of the form

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[\sum_{n=0}^{\infty} \beta^n r(S_n, A_n) - v_\pi(s)\Big].$$

Summing up: For naive Monte Carlo, we estimate

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big)\Big]$$

by a rule of the form

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[\sum_{n=0}^{\infty} \beta^n r(S_n, A_n) - v_\pi(s)\Big].$$

However, $\mathcal{V}_\pi$ satisfies a version of the DPP, namely

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[r\big(S_0^\pi, A_0^\pi\big) + \beta\mathcal{V}_\pi(S_1^\pi)\Big],$$

Summing up: For naive Monte Carlo, we estimate

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big)\Big]$$

by a rule of the form

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[\sum_{n=0}^{\infty} \beta^n r(S_n, A_n) - v_\pi(s)\Big].$$

However, $\mathcal{V}_\pi$ satisfies a version of the DPP, namely

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[r\big(S_0^\pi, A_0^\pi\big) + \beta\mathcal{V}_\pi(S_1^\pi)\Big],$$

which suggests the alternative update rule

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[r\big(s, A_0\big) + \beta v_\pi(S_1) - v_\pi(s)\Big].$$

Summing up: For naive Monte Carlo, we estimate

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^\pi, A_n^\pi\big)\Big]$$

by a rule of the form

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[\sum_{n=0}^{\infty} \beta^n r(S_n, A_n) - v_\pi(s)\Big].$$

However, $\mathcal{V}_\pi$ satisfies a version of the DPP, namely

$$\mathcal{V}_\pi(s) = \mathbb{E}_s\Big[r\big(S_0^\pi, A_0^\pi\big) + \beta\mathcal{V}_\pi(S_1^\pi)\Big],$$

which suggests the alternative update rule

$$v_\pi(s) \leftarrow v_\pi(s) + \alpha\Big[r\big(s, A_0\big) + \beta v_\pi(S_1) - v_\pi(s)\Big].$$

This is referred to as (one-step) **Temporal Difference Prediction**.

In fact, if we use the temporal difference update rule, nothing stops us from using one sample path to update the estimate for all visited states, that is

$$v_\pi(S_n) \leftarrow v_\pi(S_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta v_\pi(S_{n+1}) - v_\pi(S_n)\Big].$$

In fact, if we use the temporal difference update rule, nothing stops us from using one sample path to update the estimate for all visited states, that is

$$v_\pi(S_n) \leftarrow v_\pi(S_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta v_\pi(S_{n+1}) - v_\pi(S_n)\Big].$$

There is also a version for estimating the Q-function of π via

$$q_\pi(S_n, A_n) \leftarrow q_\pi(S_n, A_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta q_\pi(S_{n+1}, A_{n+1}) - q_\pi(S_n, A_n)\Big]$$

called the **SARSA rule**.

In fact, if we use the temporal difference update rule, nothing stops us from using one sample path to update the estimate for all visited states, that is

$$v_\pi(S_n) \leftarrow v_\pi(S_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta v_\pi(S_{n+1}) - v_\pi(S_n)\Big].$$

There is also a version for estimating the Q-function of π via

$$q_\pi(S_n, A_n) \leftarrow q_\pi(\boldsymbol{S}_n, \boldsymbol{A}_n) + \alpha\Big[\boldsymbol{r}\big(S_n, A_n\big) + \beta q_\pi(\boldsymbol{S}_{n+1}, \boldsymbol{A}_{n+1}) - q_\pi(S_n, A_n)\Big]$$

called the **SARSA rule**.

In fact, if we use the temporal difference update rule, nothing stops us from using one sample path to update the estimate for all visited states, that is

$$v_\pi(S_n) \leftarrow v_\pi(S_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta v_\pi(S_{n+1}) - v_\pi(S_n)\Big].$$

There is also a version for estimating the Q-function of π via

$$q_\pi(S_n, A_n) \leftarrow q_\pi(S_n, A_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta q_\pi(S_{n+1}, A_{n+1}) - q_\pi(S_n, A_n)\Big]$$

called the **SARSA rule**. Regarding convergence, we make a crucial observation.

In fact, if we use the temporal difference update rule, nothing stops us from using one sample path to update the estimate for all visited states, that is

$$v_\pi(S_n) \leftarrow v_\pi(S_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta v_\pi(S_{n+1}) - v_\pi(S_n)\Big].$$

There is also a version for estimating the Q-function of π via

$$q_\pi(S_n, A_n) \leftarrow q_\pi(S_n, A_n) + \alpha\Big[r\big(S_n, A_n\big) + \beta q_\pi(S_{n+1}, A_{n+1}) - q_\pi(S_n, A_n)\Big]$$

called the **SARSA rule**. Regarding convergence, we make a crucial observation.

Exploration Requirement

The estimate q_π can only converge if $(S, A) = (S^\pi, A^\pi)$ visits each state-action pair (s, a) infinitely many times.

The Control Problem

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

Then

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big)$$

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

Then

$$\begin{aligned}\mathcal{V}_\pi(s) &\leq Q_\pi\big(s, \pi'(s)\big) \\ &= \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta \mathcal{V}_\pi\big(S_1^{\pi'}\big)\Big]\end{aligned}$$

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

Then

$$\begin{aligned}
\mathcal{V}_\pi(s) &\leq Q_\pi\big(s, \pi'(s)\big) \\
&= \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta \mathcal{V}_\pi\big(S_1^{\pi'}\big)\Big] \\
&\leq \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta Q_\pi\big(S_1^{\pi'}, \pi'(S_1^{\pi'})\big)\Big]
\end{aligned}$$

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

Then

$$\begin{aligned}
\mathcal{V}_\pi(s) &\leq Q_\pi\big(s, \pi'(s)\big) \\
&= \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta \mathcal{V}_\pi\big(S_1^{\pi'}\big)\Big] \\
&\leq \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta Q_\pi\big(S_1^{\pi'}, \pi'(S_1^{\pi'})\big)\Big] \\
&\leq \ldots \leq \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi'}, A_n^{\pi'}\big)\Big]
\end{aligned}$$

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

Then

$$\begin{aligned}
\mathcal{V}_\pi(s) &\leq Q_\pi\big(s, \pi'(s)\big) \\
&= \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta \mathcal{V}_\pi\big(S_1^{\pi'}\big)\Big] \\
&\leq \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta Q_\pi\big(S_1^{\pi'}, \pi'(S_1^{\pi'})\big)\Big] \\
&\leq \ldots \leq \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi'}, A_n^{\pi'}\big)\Big] \\
&= \mathcal{V}_{\pi'}(s).
\end{aligned}$$

Thus π' is better than π.

The main idea behind the control problem is **policy improvement**. Let π, π' be two policies with

$$\mathcal{V}_\pi(s) \leq Q_\pi\big(s, \pi'(s)\big), \qquad s \in \mathbb{S}.$$

Then

$$\begin{aligned}
\mathcal{V}_\pi(s) &\leq Q_\pi\big(s, \pi'(s)\big) \\
&= \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta \mathcal{V}_\pi\big(S_1^{\pi'}\big)\Big] \\
&\leq \mathbb{E}_s\Big[r\big(S_0^{\pi'}, A_0^{\pi'}\big) + \beta Q_\pi\big(S_1^{\pi'}, \pi'(S_1^{\pi'})\big)\Big] \\
&\leq \ldots \leq \mathbb{E}_s\Big[\sum_{n=0}^{\infty} \beta^n r\big(S_n^{\pi'}, A_n^{\pi'}\big)\Big] \\
&= \mathcal{V}_{\pi'}(s).
\end{aligned}$$

Thus π' is better than π. Conversely, given π, we can improve it by choosing

$$\pi'(s) \in \arg\max_{a \in \mathbb{A}(s)} Q_\pi(s, a).$$

Given an initial policy π_0, we construct a sequence of policies $\{\pi_k\}_{k\in\mathbb{N}_0}$ by choosing

$$\pi_{k+1}(s) \in \arg\max_{a\in\mathbb{A}(s)} Q_{\pi_k}(s,a)$$

Given an initial policy π_0, we construct a sequence of policies $\{\pi_k\}_{k\in\mathbb{N}_0}$ by choosing

$$\begin{aligned}\pi_{k+1}(s) &\in \arg\max_{a\in\mathbb{A}(s)} Q_{\pi_k}(s,a)\\ &= \arg\max_{a\in\mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big],\end{aligned}$$

Given an initial policy π_0, we construct a sequence of policies $\{\pi_k\}_{k\in\mathbb{N}_0}$ by choosing

$$\begin{aligned}\pi_{k+1}(s) &\in \arg\max_{a\in\mathbb{A}(s)} Q_{\pi_k}(s,a)\\ &= \arg\max_{a\in\mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big],\end{aligned}$$

which is improving in the sense that

$$\mathcal{V}_{\pi_0} \leq \mathcal{V}_{\pi_1} \leq \mathcal{V}_{\pi_2} \leq \dots$$

Given an initial policy π_0, we construct a sequence of policies $\{\pi_k\}_{k\in\mathbb{N}_0}$ by choosing

$$\begin{aligned}\pi_{k+1}(s) &\in \arg\max_{a\in\mathbb{A}(s)} Q_{\pi_k}(s,a)\\ &= \arg\max_{a\in\mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big],\end{aligned}$$

which is improving in the sense that

$$\mathcal{V}_{\pi_0} \leq \mathcal{V}_{\pi_1} \leq \mathcal{V}_{\pi_2} \leq \dots$$

Eventually, we will reach an iterate with $\mathcal{V}_{\pi_k} = \mathcal{V}_{\pi_{k+1}}$

Given an initial policy π_0, we construct a sequence of policies $\{\pi_k\}_{k\in\mathbb{N}_0}$ by choosing

$$\begin{aligned}\pi_{k+1}(s) &\in \arg\max_{a\in\mathbb{A}(s)} Q_{\pi_k}(s,a)\\ &= \arg\max_{a\in\mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big],\end{aligned}$$

which is improving in the sense that

$$\mathcal{V}_{\pi_0} \leq \mathcal{V}_{\pi_1} \leq \mathcal{V}_{\pi_2} \leq \dots$$

Eventually, we will reach an iterate with $\mathcal{V}_{\pi_k} = \mathcal{V}_{\pi_{k+1}}$, implying

$$\mathcal{V}_{\pi_k}(s) = \mathcal{V}_{\pi_{k+1}}(s) = \max_{a\in\mathbb{A}(s)} \Big[r(s,a) + \beta \sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big].$$

Given an initial policy π_0, we construct a sequence of policies $\{\pi_k\}_{k\in\mathbb{N}_0}$ by choosing

$$\begin{aligned}\pi_{k+1}(s) &\in \arg\max_{a\in\mathbb{A}(s)} Q_{\pi_k}(s,a)\\ &= \arg\max_{a\in\mathbb{A}(s)}\Big[r(s,a)+\beta\sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big],\end{aligned}$$

which is improving in the sense that

$$\mathcal{V}_{\pi_0}\leq\mathcal{V}_{\pi_1}\leq\mathcal{V}_{\pi_2}\leq\dots$$

Eventually, we will reach an iterate with $\mathcal{V}_{\pi_k}=\mathcal{V}_{\pi_{k+1}}$, implying

$$\mathcal{V}_{\pi_k}(s)=\mathcal{V}_{\pi_{k+1}}(s)=\max_{a\in\mathbb{A}(s)}\Big[r(s,a)+\beta\sum_{s'\in\mathbb{S}} p(s'|s,a)\mathcal{V}_{\pi_k}(s')\Big].$$

But then $\mathcal{V}_{\pi_k}$ solves the DPP, implying that $\mathcal{V}=\mathcal{V}_{\pi_k}$ and $\pi^*=\pi_k$ is **optimal** by the verification theorem.

Putting Everything Together

At this point, it is just a matter of combining prediction and control.

At this point, it is just a matter of combining prediction and control... right?

At this point, it is just a matter of combining prediction and control... right?

Exploration vs. Exploitation

Prediction requires that all states and actions are continued to be visited, which (typically) fails if we choose the greedy action according to the policy improvement procedure.

At this point, it is just a matter of combining prediction and control... right?

Exploration vs. Exploitation

Prediction requires that all states and actions are continued to be visited, which (typically) fails if we choose the greedy action according to the policy improvement procedure.

The way out is to choose **ϵ-greedy actions**, that is, choose the greedy action with probability $1 - \epsilon$ and a random action with probability ϵ.

SARSA (On-Policy Temporal Difference Control)

SARSA (On-Policy Temporal Difference Control)

▷ Initialize q arbitrarily

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)
- ▷ Choose action A_0 using ϵ-greedy policy derived from q

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)
- ▷ Choose action A_0 using ϵ-greedy policy derived from q
- ▷ Loop over $n = 0, 1, 2, \ldots$

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)
- ▷ Choose action A_0 using ϵ-greedy policy derived from q
- ▷ Loop over $n = 0, 1, 2, \ldots$
  - ▷ Take action A_n and observe $r(S_n, A_n)$ and S_{n+1}

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)
- ▷ Choose action A_0 using ϵ-greedy policy derived from q
- ▷ Loop over $n = 0, 1, 2, \ldots$
  - ▷ Take action A_n and observe $r(S_n, A_n)$ and S_{n+1}
  - ▷ Choose action A_{n+1} using ϵ-greedy policy derived from q

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)
- ▷ Choose action A_0 using ϵ-greedy policy derived from q
- ▷ Loop over $n = 0, 1, 2, \ldots$
  - ▷ Take action A_n and observe $r(S_n, A_n)$ and S_{n+1}
  - ▷ Choose action A_{n+1} using ϵ-greedy policy derived from q
  - ▷ Update q according to

$$q(S_n, A_n) \leftarrow q(S_n, A_n) + \alpha \Big[r(S_n, A_n) + \beta q(S_{n+1}, A_{n+1}) - q(S_n, A_n) \Big]$$

SARSA (On-Policy Temporal Difference Control)

- ▷ Initialize q arbitrarily
- ▷ Initialize S_0 (e.g. randomly)
- ▷ Choose action A_0 using ϵ-greedy policy derived from q
- ▷ Loop over $n = 0, 1, 2, \ldots$
  - ▷ Take action A_n and observe $r(S_n, A_n)$ and S_{n+1}
  - ▷ Choose action A_{n+1} using ϵ-greedy policy derived from q
  - ▷ Update q according to

$$q(S_n, A_n) \leftarrow q(S_n, A_n) + \alpha \Big[r(S_n, A_n) + \beta q(S_{n+1}, A_{n+1}) - q(S_n, A_n) \Big]$$

Regarding convergence, we have to choose α and ϵ adaptively such that $\epsilon_n \downarrow 0$ and the learning rate satisfies the **Robbins-Monro condition**

$$\sum_{n=0}^{\infty} \alpha_n = \infty \qquad \text{and} \qquad \sum_{n=0}^{\infty} \alpha_n^2 < \infty.$$

There is also an alternative version in which we work with two policies:

There is also an alternative version in which we work with two policies:

- A **behavioral policy** π_b used to generate the MDP

There is also an alternative version in which we work with two policies:

- A **behavioral policy** π_b used to generate the MDP
- A **control policy** π to update q to approximate an optimal control

There is also an alternative version in which we work with two policies:

- A **behavioral policy** π_b used to generate the MDP
- A **control policy** π to update q to approximate an optimal control

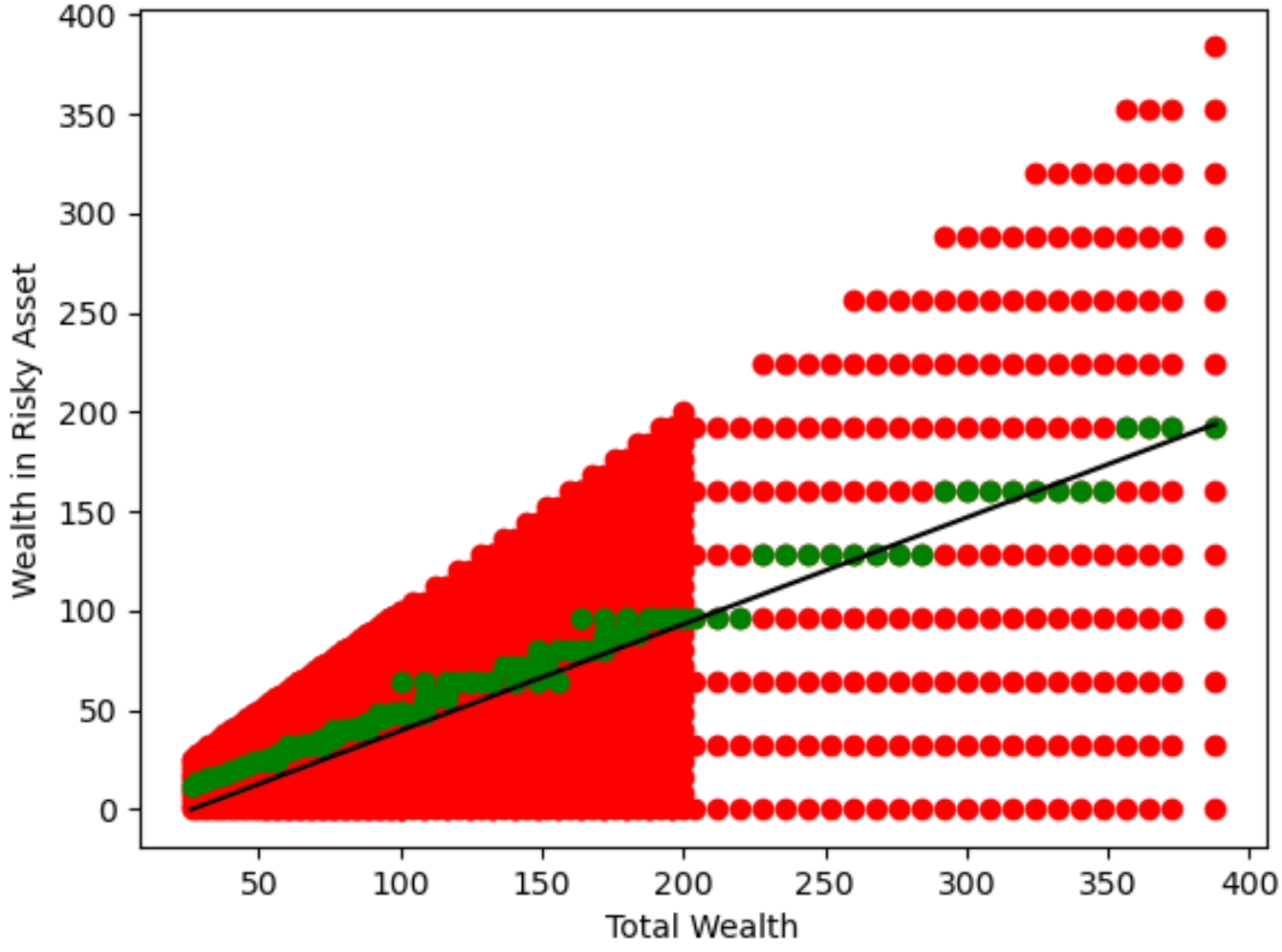
Q-Learning (Off-Policy Temporal Difference Control)

- Initialize q arbitrarily
- Initialize S_0 (e.g. randomly)
- Choose action A_0 using ϵ-greedy policy derived from q
- Loop over $n = 0, 1, 2, \ldots$
  - Take action A_n and observe $r(S_n, A_n)$ and S_{n+1}
  - Choose action A_{n+1} using ϵ-greedy policy derived from q
  - Update q according to

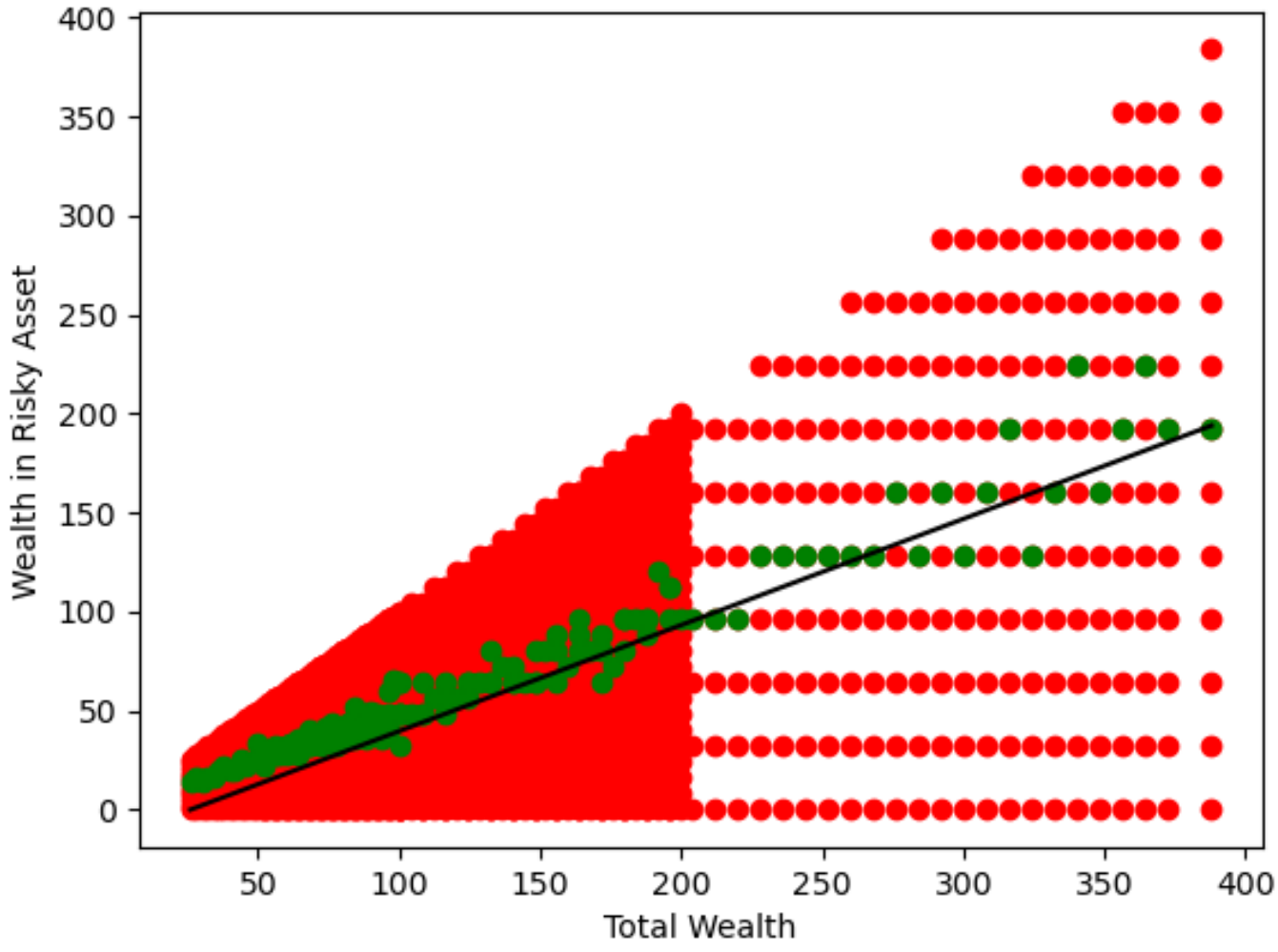
$$q(S_n, A_n) \leftarrow q(S_n, A_n) + \alpha \Big[r(S_n, A_n) + \beta \max_{a \in \mathbb{A}(s)} q(S_{n+1}, a) - q(S_n, A_n) \Big]$$

Numerical Experiments

Optimal investment with indivisible assets solved with **Dynamic Programming**



Optimal investment with indivisible assets solved with **Q-Learning**



Observations:

Observations:

- Dynamic Programming is significantly faster

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data
- If we increase the number of states/actions, both algorithms become inefficient

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data
- If we increase the number of states/actions, both algorithms become inefficient
- There are ways to improve Q-Learning

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data
- If we increase the number of states/actions, both algorithms become inefficient
- There are ways to improve Q-Learning
- We need to be very careful with the choice of α

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data
- If we increase the number of states/actions, both algorithms become inefficient
- There are ways to improve Q-Learning
- We need to be very careful with the choice of α
- Q-Learning does not require us to know the exact dynamics

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data
- If we increase the number of states/actions, both algorithms become inefficient
- There are ways to improve Q-Learning
- We need to be very careful with the choice of α
- Q-Learning does not require us to know the exact dynamics
- Where do we get all the data from?

Observations:

- Dynamic Programming is significantly faster
- Q-Learning requires a lot of data
- If we increase the number of states/actions, both algorithms become inefficient
- There are ways to improve Q-Learning
- We need to be very careful with the choice of α
- Q-Learning does not require us to know the exact dynamics
- Where do we get all the data from?
- If we do not know the dynamics, how can we check convergence?