Before:
1. \( V^*(s) = \max_{\pi} V^\pi(s) \)
2. Goals & Motifs.

Pop Quiz: Discussion

\( V^\pi(s) = \) value of a state (single state)
\( V^\pi(S_0) = \) value of a state-value distribution (\( S_0 \) is r.v.)

\[ \text{r.v. over distribution of start states} \]
This would be \( V(S_0) \)

**Policy Evaluation**

Given a policy \( \pi \), find \( V^\pi \).

Assume \( P, R, \gamma \) are known. It's essentially a planning problem.

**Method:**
1. Solve system of equations produced by the Bellman equations.
   \[ V^\pi(s) = \sum_a \pi(a | s) \left( R(s, a, s') + \gamma V^\pi(s') \right) \]
   \( \text{Bellman eqn.} \)
2. **Dynamic Programming**

**Dynamic Programming**

Sequence of approximations of \( V^\pi \).

\[ \hat{V}_0^\pi, \hat{V}_1^\pi, \hat{V}_2^\pi, \ldots \]

- Initial guess of value function for every state
- \( \hat{V} \) is a vector
- Chosen arbitrarily except for \( \hat{V}_0^\pi(s) = 0 \) for terminal states (\( S_0 \)). Terminal state
\[
\hat{V}_\pi^\pi(s) = \sum_a \pi(s,a) \sum_{s'} p(s,a,s') [R(s,a,s') + \gamma \hat{V}_\pi^\pi(s')] 
\]

Properties:

1. \( \hat{V}_\pi^\pi = V^\pi \) is a fixed point. \( f(x) = x \).
2. \( \hat{V}_\pi^\pi \to V^\pi \) as \( k \to \infty \) for finite MDPs with bounded rewards.
3. A full pass over the state space is called a "full backup." A single state update is called a "backup."

Example:

\[
\begin{array}{cccc}
\downarrow & \downarrow & \downarrow & \downarrow \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\leftarrow & \leftarrow & \leftarrow & \leftarrow \\
\end{array}
\]

\( R_t = -1 \) always
\( \pi \) is shown
\( V = 1 \)

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
\uparrow & \uparrow & \uparrow & \uparrow \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
-1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 \\
\end{array}
\]

\( V(s) = \frac{1 + \pi(s,a) = 0}{\downarrow} \)
\( \downarrow \to \frac{1 = 0}{\downarrow} \) Reward at \( k+1 \)

This value never changes as terminal state

\[
\begin{array}{cccc}
-2 & -2 & -2 & -2 \\
-2 & -2 & -2 & -2 \\
-2 & -2 & -2 & -1 \\
-2 & -2 & -1 & 0 \\
\end{array}
\]
\[ V_3^\pi = \begin{bmatrix}
-3 & -3 & -3 & -3 \\
-3 & -3 & -3 & -3 \\
-3 & -3 & -2 & -3 \\
-3 & -2 & -1 & 0
\end{bmatrix} \]

\[ V_4^\pi = \begin{bmatrix}
-4 & -4 & -4 & -5 \\
-4 & -4 & -3 & -3 \\
-4 & -3 & -2 & -3 \\
-3 & -2 & -1 & 0
\end{bmatrix} \]

\[ V_5^\pi = \begin{bmatrix}
-5 & -5 & -4 & -3 \\
-5 & -4 & -3 & -2 \\
-4 & -3 & -2 & -3 \\
-3 & -2 & -1 & 0
\end{bmatrix} \]

\[ V_6^\pi = \begin{bmatrix}
-5 & -5 & -4 & -3 \\
-5 & -4 & -3 & -2 \\
-4 & -3 & -2 & -3 \\
-3 & -2 & -1 & 0
\end{bmatrix} \]

now it converges.

In-place implementation

- When update $V_x^\pi(s)$ store it back in the same table rather than
- Update states in any order.
- Asynchronous updates
  - update a state many times before updating another state.
- Guaranteed to converge to $V^\pi$ if no states are starved of updating.
  - Order doesn't change final convergence but it will change.

Policy Improvement

If we have estimated $q^\pi$, how can we improve $\pi$
Policy Improvement Theorem:

Let \( \pi \) and \( \pi' \) be deterministic policies such that for all \( s \in S \):

\[
q^\pi(s, \pi'(s)) \geq V^\pi(s) = q^\pi(s, \pi(s))
\]

Then, \( \pi' \geq \pi \)

\( \forall s \in S \), \( V^\pi'(s) \geq V^\pi(s) \)

Lecture 10 starts here:

\( P_{\text{ProB}}: \)

\[
V^\pi(s) \leq q^\pi(s, \pi'(s))
\]

\[
= \mathbb{E} \left[ R_t + Y R_{t+1} + Y^2 R_{t+2} + Y^3 R_{t+3} \ldots \mid S_t = s, a_t = \right. \]

\( \text{from } \pi \)

\( \left. \text{from } \pi' \right] \]

\( a_t = \text{achieve } \)

\[
= \mathbb{E} \left[ R_t + Y \mathbb{E} \left[ q^\pi(s_{t+1}, \pi'(s_{t+1})) \mid S_t = s, \pi' \right] \right]
\]

\( \pi' \) influences \( s_{t+1} \) is a.s.

\( S_{t+1} \) is a.s.

we do care about how we got here.

\[
\leq \mathbb{E} \left[ R_t + Y q^\pi(s_{t+1}, \pi'(s_{t+1})) \mid S_t = s, \pi' \right]
\]

\[
= \mathbb{E} \left[ R_t + Y \mathbb{E} \left[ R_{t+1} + Y V^\pi(s_{t+2}) \mid S_t = s, \pi' \right] \right]
\]

Why no \( s_{t+1} ? \)

Because \( S_{t+1} \)

The last \( q \) of \( s_t \).
\[
\mathbb{E} \left[ R_t + \gamma R_{t+1} + \gamma^2 V^\pi(S_{t+2}) \mid S_t = s, \pi' \right] \\
\leq \mathbb{E} \left[ R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 V^\pi(S_{t+3}) \mid S_t = s, \pi' \right] \\
\leq \mathbb{E} \left[ \right] \\
\leq V^\pi
\]

**Policy Improvement**

- For all states, select an action \( a \) that maximises \( q^\pi(s, a) \)
- This policy is called "greedy with respect to \( q^\pi \)"
  \[
  \pi'(s) \in \arg\max_a \sum_{s'} P(s, a, s') \left[ R(s, a, s') + \gamma V^\pi(s') \right]
  \]

- What if \( \pi' = \pi \)? (fixed pt.)
  \[
  S_0, V^\pi(s) = \max_a \sum_{s'} P(s, a, s') \left[ R(s, a, s') + \gamma V^\pi(s') \right]
  \]

**Policy Iterations:**

\[
\pi_0 \xrightarrow{\text{eval}} V^{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{eval}} V^{\pi_1} \xrightarrow{\text{improve}} \pi_2 \xrightarrow{\text{eval}} \ldots
\]

- Will terminate after finite \( \pi \) iterations.
- Exists an optimal policy that is deterministic.

Problem is evaluation?
- The hard way is to run a policy fully & evaluate.
  - But too expensive.
  - So, we will take some backups & approximate \( V^\pi \).
    - \( \pi \) not all

**Value Iteration**

- One full backup during policy evaluation in policy iteration.
\( V_0 : \text{arbitrarily} \)

\( \pi_0 : \text{arbitrarily (but deterministic)} \)

\[ V_1 : \forall s \quad V_1(s) = \sum_{s'} P(s, \pi_0(s), s') R(s', \pi_0(s), s') + Y V_0(s') \]

No sum over actions as \( \pi_0 \) is deterministic.

\[ \pi_1 : \forall s \quad \pi_1(s) \in \arg\max_a \sum_{s'} P(s, a, s') \left[ R(s, a, s') + Y V_1(s') \right] \]

\[ V_2 : \forall s \quad V_2(s) = \sum_{s'} P(s, \pi_1(s), s') \left[ R(s, \pi_1(s), s') + Y V_2(s') \right] \]

\[ \pi_2 : \forall s \quad \pi_2(s) \in \arg\max_a \sum_{s'} P(s, a, s') \left[ R(s, a, s') + Y V_2(s') \right] \]

\[ V_{K+1}(s) = \max_a \sum_{s'} P(s, a, s') \left[ R(s, a, s') + Y V_K(s') \right] \]

This looks like the Bellman optimality equation. We can say that we have taken the BDE.

**Properties:**

1) \( V_K \to V^* \)

Asynchronous: You update some state 3 times before updating some other state.

Value function converges. [How? Wait for proof.]

---

**Bellman Operator**

[To prove Value Iteration converges]

View state value functions as vectors in \( \mathbb{R}^{1 \times 1} \):

\[ V_K = \begin{bmatrix} V_K(s_1) \\ V_K(s_2) \\ \vdots \\ V_K(s_{M}) \end{bmatrix} \]
Bellman Operator: \( \mathbb{R}^{|S|} \rightarrow \mathbb{R}^{|S|} \)

\[ T(V_K) \subseteq V_{K+1} \]

\[
(T_{V_K}) (s) = V_{K+1} (s) = \max_a \sum_{s'} p(s,a,s') \left[ r(s,a,s') + \gamma V_K (s') \right]
\]

This whole thing is a vector. 5th entry of vector.

**Contraction Mapping**

\( f : \mathcal{X} \rightarrow \mathcal{X} \) is a contraction mapping if there exists a \( \lambda \in (0,1) \) such that \( \forall x \in \mathcal{X} \) and \( \forall y \in \mathcal{X} \)

\[
\delta (f(x), f(y)) \leq \lambda \delta (x,y) \text{ where } \delta \text{ is a distance function.}
\]

\[
\begin{aligned}
\text{Distance to self } &= 0 \\
\lambda \text{ inequality } &= \lambda > 0
\end{aligned}
\]

**Banach fixed point theorem**

If \( f \) is a contraction mapping on a non-empty complete normed vector space,
then \( f \) has a unique fixed pt. and the sequence defined by \( X_{k+1} = f(X_k) \)

**Bellman Operator is Contraction Mapping**

\[

f \leftarrow T \quad X \leftarrow \mathbb{R}^{[1]} \quad d(v,v') = \max |V(s) - V'(s)| \\
\|TV - TV'\| \leq \alpha \|V - V'\|

\]

where

\[

\|TV - TV'\| = \max_s \left| (TV - TV') (s) \right| \\
= \max_s \left| TV (s) - TV' (s) \right| \\
= \max_s \left| \sum_{s'} P(s,a,s') \left[ R(s,a,s') + \gamma V(s') \right] - \sum_{s'} P(s,a,s') \left[ R(s,a,s') + \gamma V(s') \right] \right| \\
= \max_s \max_a \left| \sum_{s'} P(s,a,s') \left[ R(s,a,s') + \gamma V(s') \right] - \sum_{s'} P(s,a,s') \left[ R(s,a,s') + \gamma V(s') \right] \right|

\]

Claim: \( \max_x |f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)| \)

\[

\forall x \quad f(x) - g(x) \leq |f(x) - g(x)| \\
f(x) \leq |f(x) - g(x)| + g(x) \\
\max_x f(x) \leq \max_x |f(x) - g(x)| + g(x) \\
\leq \max_x \left( |f(x) - g(x)| + \max_x g(x) \right)

\]

\[

\max_x |f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|

\]