# Model-based Prediction and Control

Oleksii Hrinchuk

This lecture is strongly based on David Silver's UCL course on reinforcement learning, lecture 3

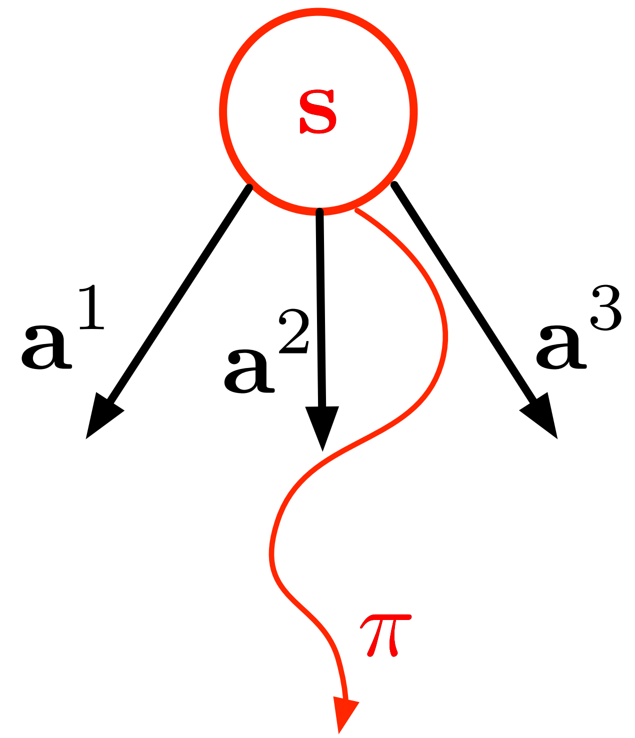
# Recap

- Value function

$$V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \ldots\right]$$

- Optimal value function

$$V^{*}(\mathbf{s}_t) = \max_{\pi} V^{\pi}(\mathbf{s}_t), \quad \forall \mathbf{s}_t \in \mathcal{S}$$



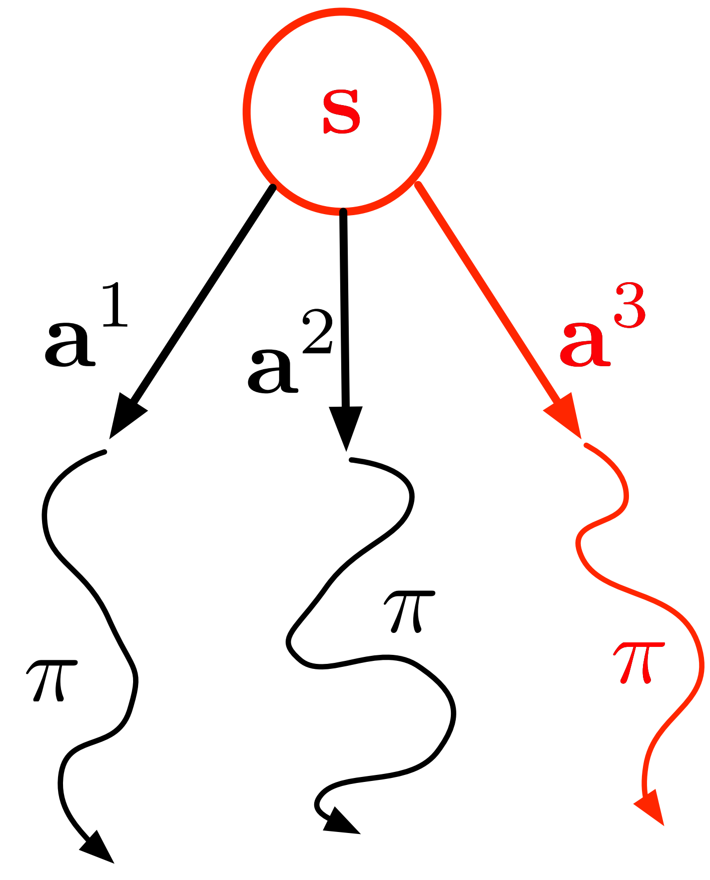
# Recap

- Q-function (state-action value function)

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \ldots\right]$$

- Optimal Q-function

$$Q^{*}(\mathbf{s}_t, \mathbf{a}_t) = \max_{\pi} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t), \quad \forall (\mathbf{s}_t, \mathbf{a}_t)$$



# Recap

- Connection between $V^\pi$ and $Q^\pi$

$$V^\pi(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V^\pi(\mathbf{s}_{t+1})$$

- Bellman expectation equation

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \mathbf{a}_{t+1}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

$$V^\pi(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t} \left[ r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V^\pi(\mathbf{s}_{t+1}) \right]$$

# Recap

- Connection between $V^*$ and $Q^*$

$$V^*(\mathbf{s}_t) = \max_{\mathbf{a}_t \in \mathcal{A}} Q^*(\mathbf{s}_t, \mathbf{a}_t)$$

$$Q^*(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V^*(\mathbf{s}_{t+1})$$

- Bellman optimality equation

$$Q^*(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} \max_{\mathbf{a}_{t+1} \in \mathcal{A}} Q^*(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

$$V^*(\mathbf{s}_t) = \max_{\mathbf{a}_t \in \mathcal{A}} \left[ r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V^*(\mathbf{s}_{t+1}) \right]$$

# Model-based prediction and control

- Model-based
  Transition model $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ are available

- Prediction
  Find $V^\pi$ for a given policy $\pi$ in a given MDP $\mathcal{M}$

- Control
  Find the optimal policy $\pi^*$ in a given MDP $\mathcal{M}$

- … also known as Dynamic Programming

# Iterative policy evaluation

- Problem: how to find $V^\pi$ given the policy $\pi$?
  (how good is the given policy $\pi$?)

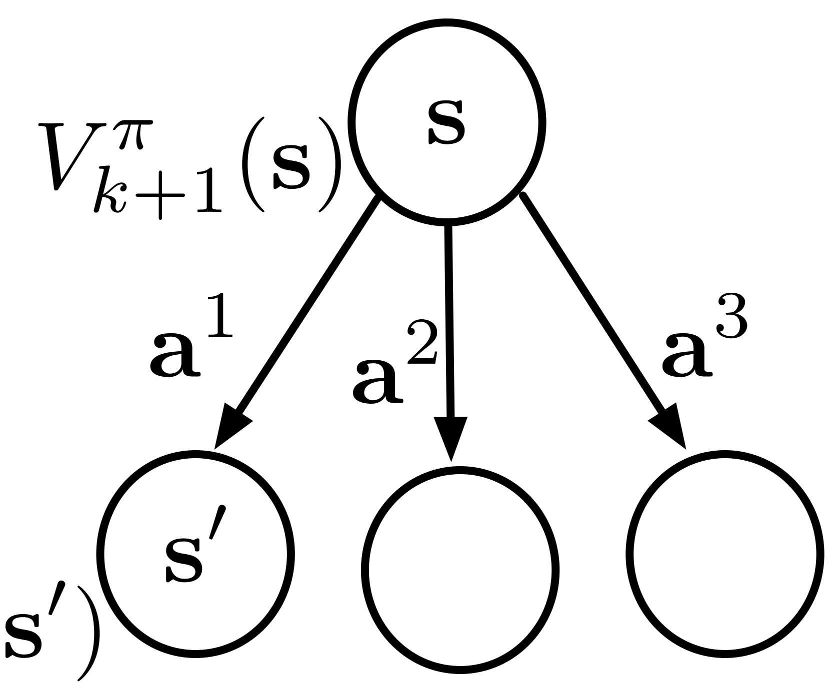
- Solution: use Bellman expectation backup.

$$V_{k+1}^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V_k^{\pi}(\mathbf{s}_{t+1})\right]$$

$$\boxed{V_0^\pi \to V_1^\pi \to \cdots \to V^\pi}$$

converging sequence

initial guess (e.g. zeros)

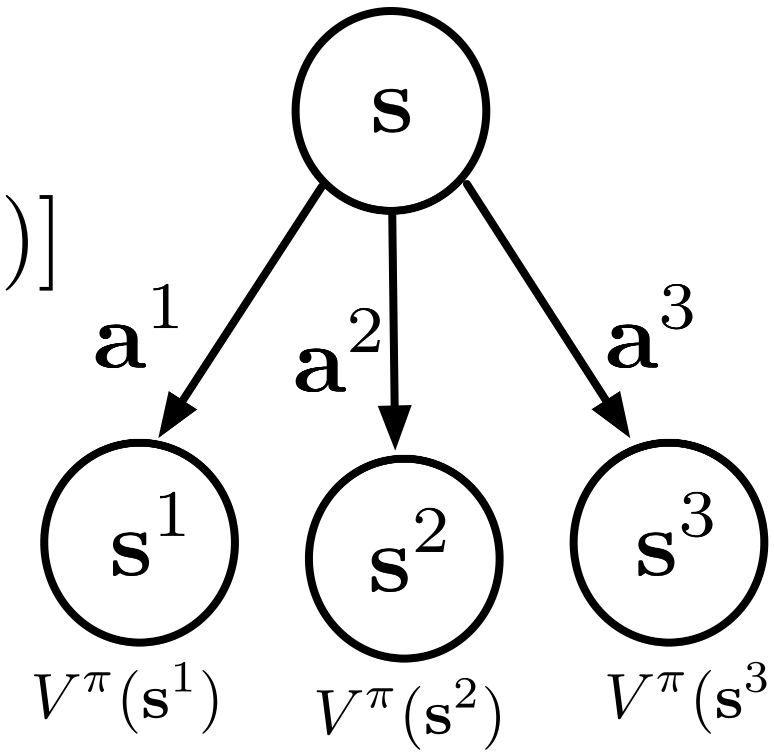


# Greedy policy improvement

- Assume we estimated $V^{\pi}$ for some policy $\pi$. Can we come up with a better policy? Yes!

$$V^{\pi}(\mathbf{s}) = \sum_{\mathbf{a} \in \mathcal{A}} \pi(\mathbf{a}|\mathbf{s}) \left[ r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} V^{\pi}(\mathbf{s}') \right]$$

$$\leq \max_{\mathbf{a} \in \mathcal{A}} \underbrace{\left[ r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} V^{\pi}(\mathbf{s}') \right]}_{Q^{\pi}(\mathbf{s}, \mathbf{a})}$$



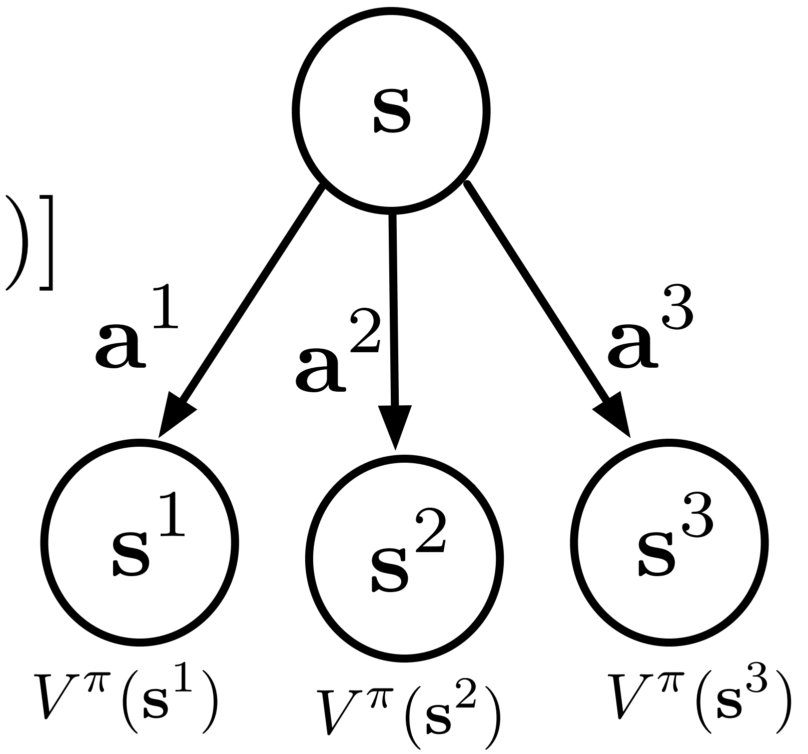

$$\hat{\pi}(\mathbf{a}|\mathbf{s}) = \begin{cases} 1, & \mathbf{a} = \arg\max_{\mathbf{a} \in \mathcal{A}} \left[ r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} V^{\pi}(\mathbf{s}') \right] \\ 0, & \text{otherwise} \end{cases}$$

# Greedy policy improvement

- Assume we estimated $V^\pi$ for some policy $\pi$. Can we come up with a better policy? Yes!

$$V^\pi(\mathbf{s}) = \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a}|\mathbf{s}) \left[r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} V^\pi(\mathbf{s}')\right]$$

$$\leq \max_{\mathbf{a}\in\mathcal{A}} \underbrace{\left[r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'} V^\pi(\mathbf{s}')\right]}_{Q^\pi(\mathbf{s},\mathbf{a})}$$

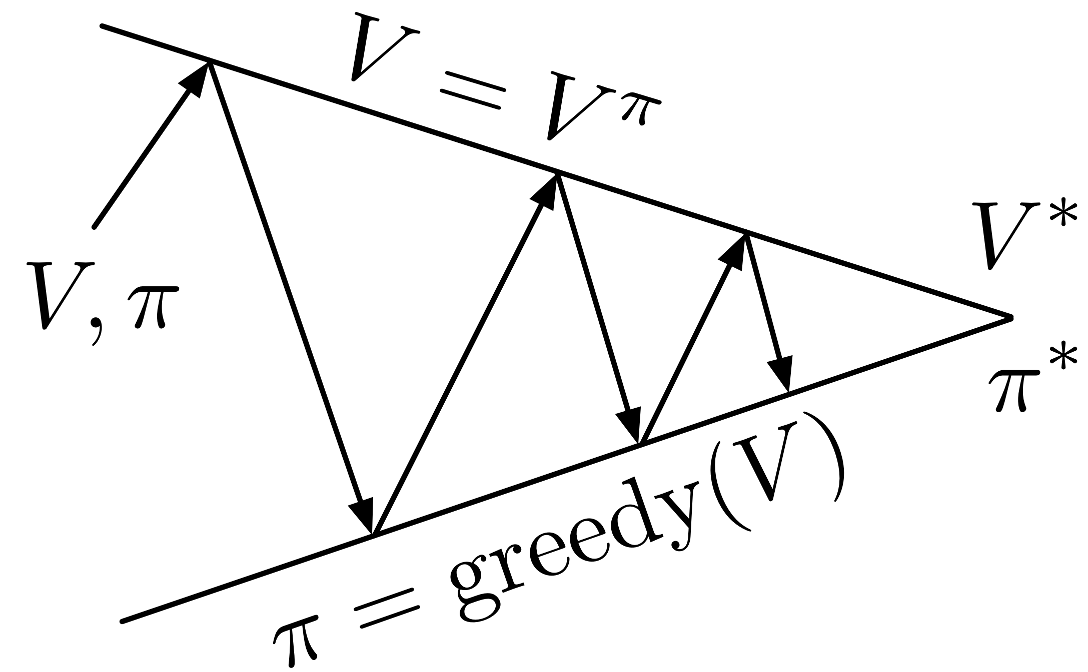



$$\hat{\pi}(\mathbf{a}|\mathbf{s}) = \begin{cases} 1, & \mathbf{a} = \arg\max_{\mathbf{a}\in\mathcal{A}} Q^\pi(\mathbf{s},\mathbf{a}) \\ 0, & \text{otherwise} \end{cases}$$

# Policy iteration

- Policy evaluation $\pi \rightarrow V^{\pi}$
  Iterative policy evaluation

- Policy improvement $\pi, V^{\pi} \rightarrow \hat{\pi}$
  Greedy policy improvement

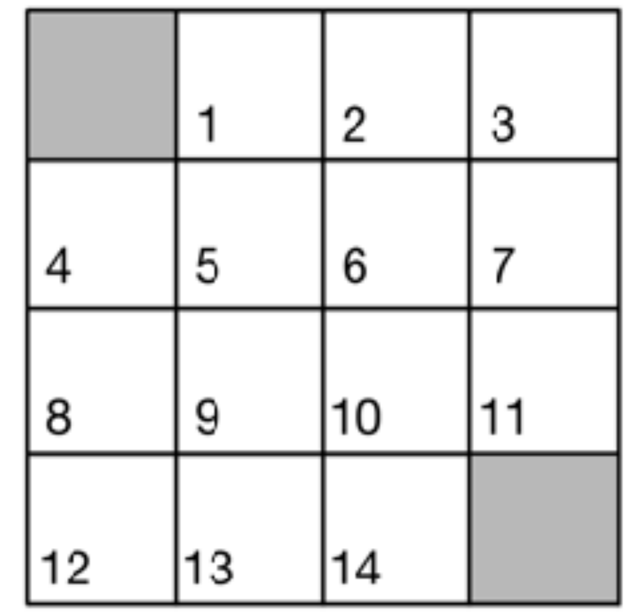



$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \cdots \rightarrow \pi^*$$

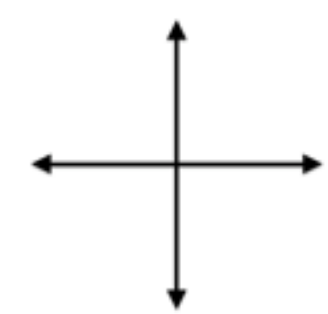
initial guess (e.g. random policy)

# Example: Small Gridworld

- Nonterminal states 1, …, 14
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Agent follows uniform random policy




$r = -1$
on all transitions



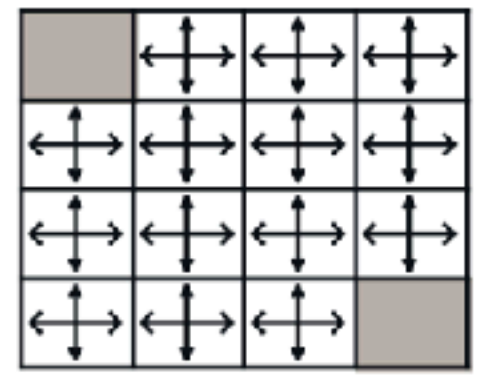

Source: David Silver’s UCL course on reinforcement learning, lecture 3

# Example: Small Gridworld

$V_k^{\pi}$ for the random policy | greedy policy w.r.t. $V_k^{\pi}$

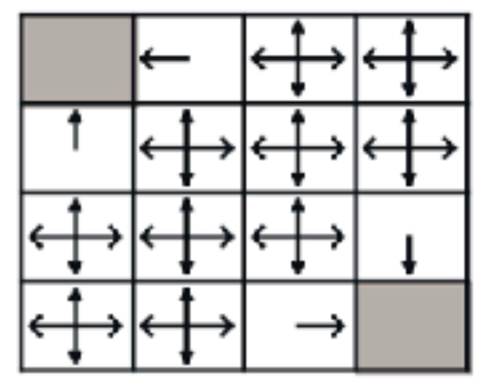
$k = 0$

| | | | |
|---|---|---|---|
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 |



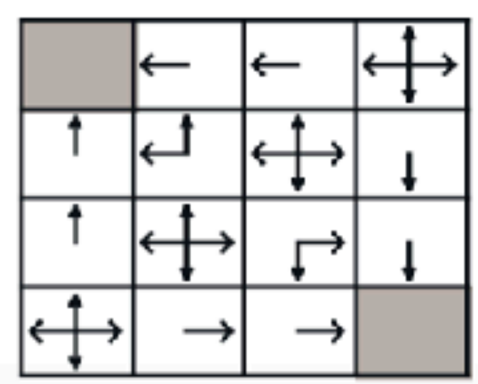

$k = 1$

| | | | |
|---|---|---|---|
| 0.0 | -1.0 | -1.0 | -1.0 |
| -1.0 | -1.0 | -1.0 | -1.0 |
| -1.0 | -1.0 | -1.0 | -1.0 |
| -1.0 | -1.0 | -1.0 | 0.0 |



$k = 2$

| | | | |
|---|---|---|---|
| 0.0 | -1.7 | -2.0 | -2.0 |
| -1.7 | -2.0 | -2.0 | -2.0 |
| -2.0 | -2.0 | -2.0 | -1.7 |
| -2.0 | -2.0 | -1.7 | 0.0 |



Source: David Silver’s UCL course on reinforcement learning, lecture 3

# Example: Small Gridworld

$V_k^\pi$ for the random policy

$k = 3$

| | | | |
|---|---|---|---|
| 0.0 | -2.4 | -2.9 | -3.0 |
| -2.4 | -2.9 | -3.0 | -2.9 |
| -2.9 | -3.0 | -2.9 | -2.4 |
| -3.0 | -2.9 | -2.4 | 0.0 |

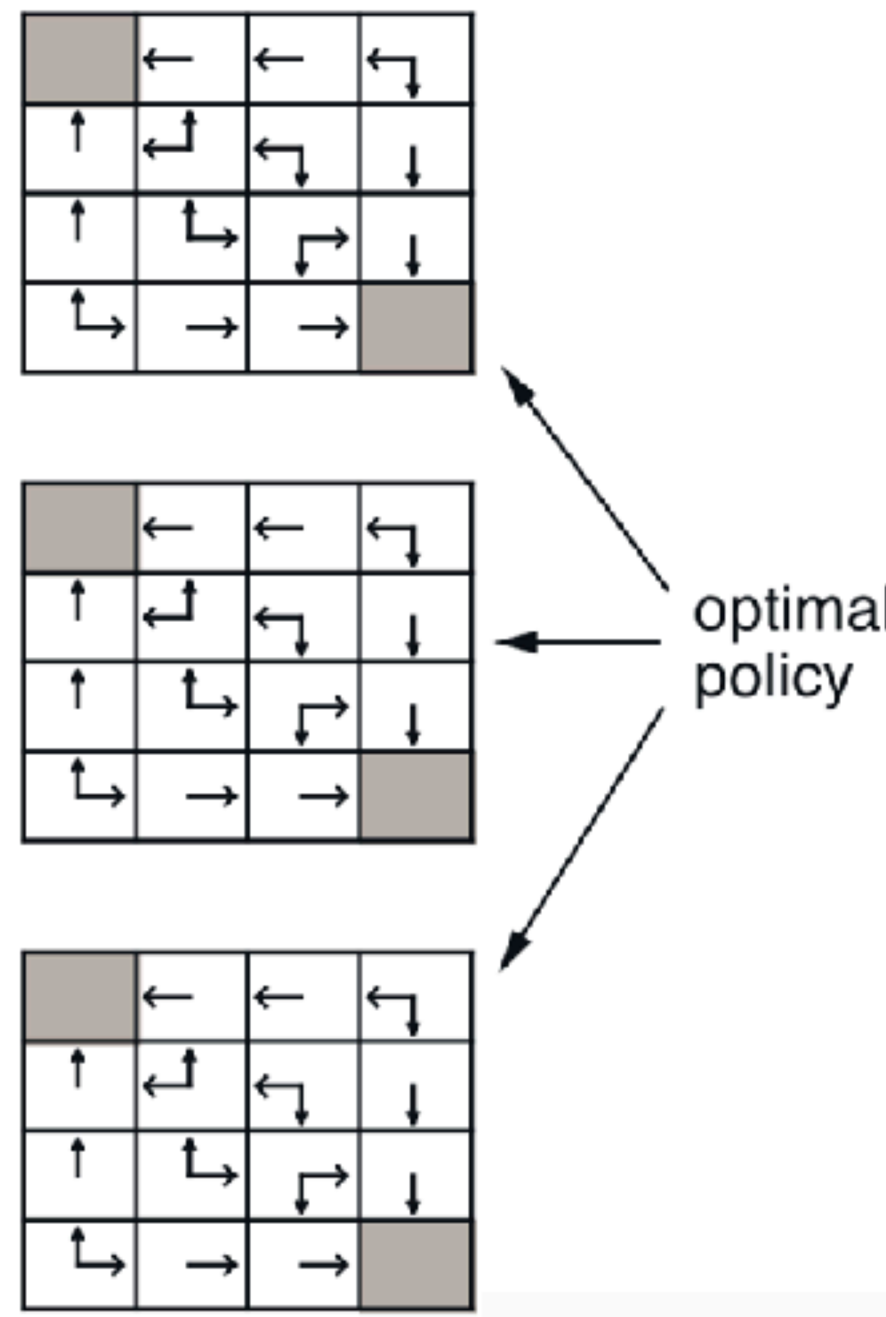
$k = 10$

| | | | |
|---|---|---|---|
| 0.0 | -6.1 | -8.4 | -9.0 |
| -6.1 | -7.7 | -8.4 | -8.4 |
| -8.4 | -8.4 | -7.7 | -6.1 |
| -9.0 | -8.4 | -6.1 | 0.0 |

$k = \infty$

| | | | |
|---|---|---|---|
| 0.0 | -14. | -20. | -22. |
| -14. | -18. | -20. | -20. |
| -20. | -20. | -18. | -14. |
| -22. | -20. | -14. | 0.0 |

greedy policy w.r.t. $V_k^\pi$

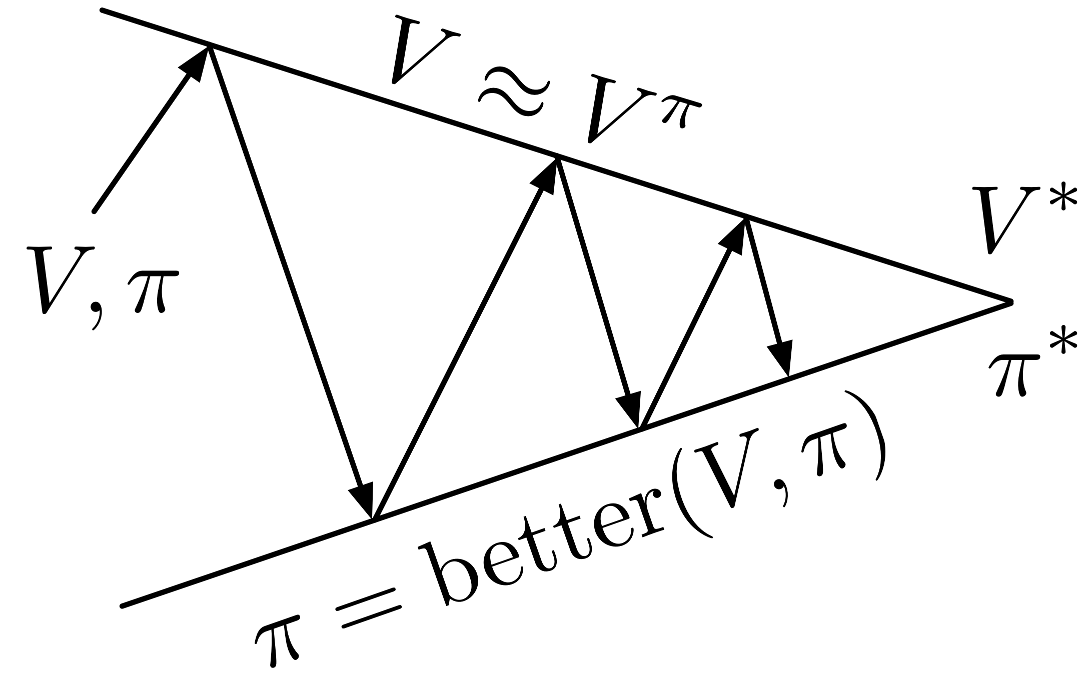



Source: David Silver's UCL course on reinforcement learning, lecture 3

# Generalized policy iteration

- Policy evaluation $\pi \to V^\pi$
  ANY policy evaluation algorithm

- Policy improvement $\pi, V^\pi \to \hat{\pi}$
  ANY policy improvement algorithm




$$\pi_0 \to V^{\pi_0} \to \pi_1 \to V^{\pi_1} \to \cdots \to \pi^*$$

# Value iteration

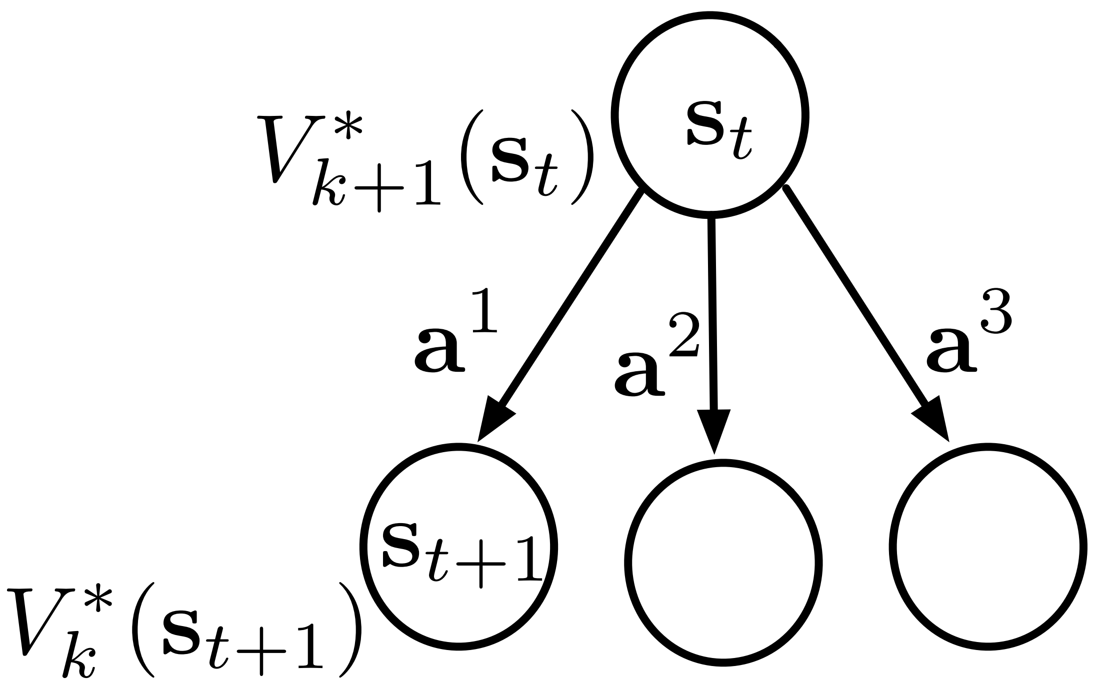
- Problem: how to find the optimal policy $\pi^*$?
- Solution: use Bellman optimality backup.

$$V_{k+1}^*(\mathbf{s}_t) = \max_{\mathbf{a}_t \in \mathcal{A}} \left[ r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V_k^*(\mathbf{s}_{t+1}) \right]$$

$$\boxed{V_0^* \to V_1^* \to \cdots \to V^*}$$

converging sequence

initial guess (e.g. zeros)



# Asynchronous updates

- In-Place Dynamic Programming
- Prioritized sweeping
- Real-time dynamic programming

# In-Place DP

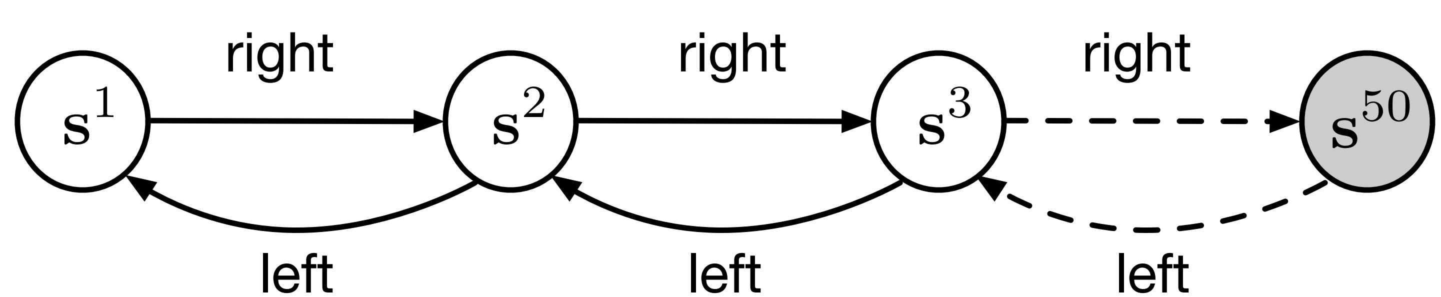
- Synchronous VI stores two copies of value function
  - for all $\mathbf{s}_t \in \mathcal{S}$

$$V_{k+1}(\mathbf{s}_t) \leftarrow \max_{\mathbf{a}\in\mathcal{A}} \left[\mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V_k(\mathbf{s}_{t+1})\right]$$

  - $k \leftarrow k+1$

- In-place VI stores one copy of value function
  - for all $\mathbf{s}_t \in \mathcal{S}$

$$V(\mathbf{s}_t) \leftarrow \max_{\mathbf{a}\in\mathcal{A}} \left[\mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V(\mathbf{s}_{t+1})\right]$$



# Prioritized sweeping

- Use magnitude of Bellman error to guide state selection

$$\delta(\mathbf{s}_t) = \left| \max_{\mathbf{a}_t \in \mathcal{A}} \left[ \mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V(\mathbf{s}_{t+1}) \right] - V(\mathbf{s}_t) \right|$$

- Backup the state with the largest remaining BE

- Update BE of affected states after each backup

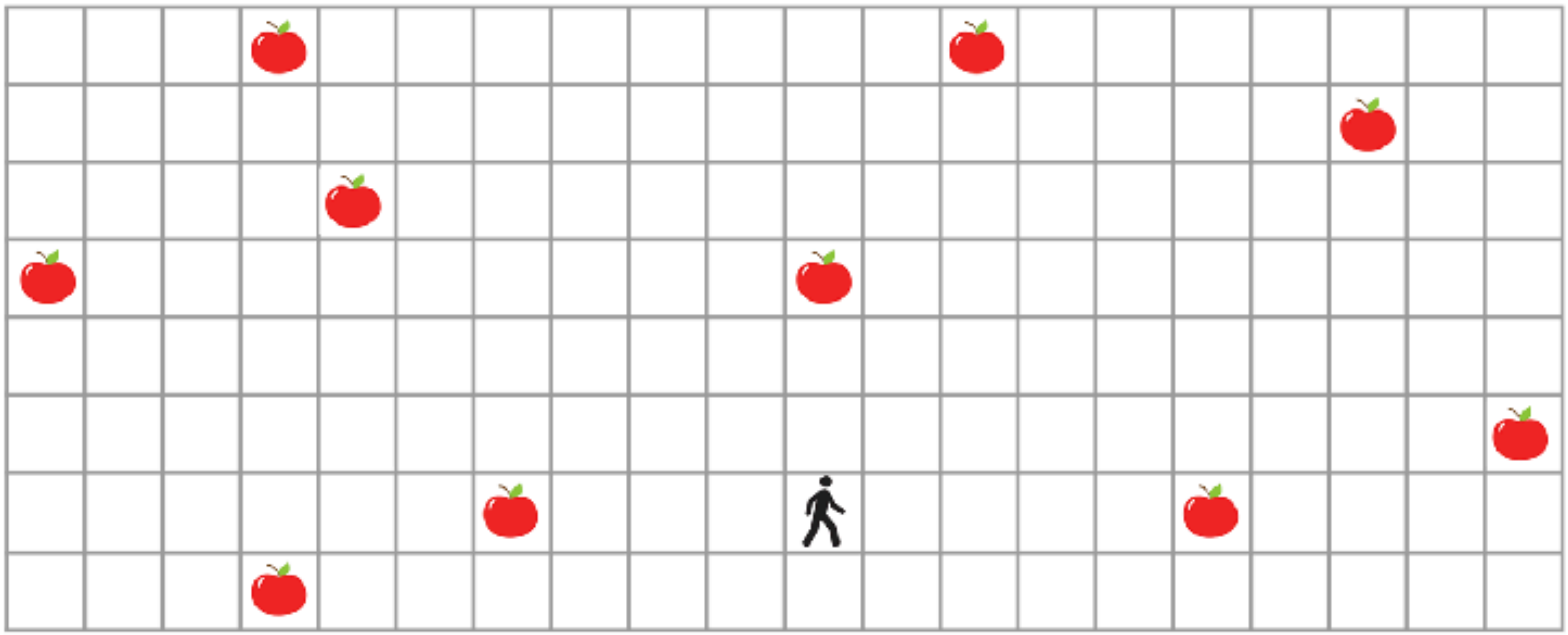
- Can be implemented efficiently by maintaining a priority queue

# Real-Time DP

- Use agent's experience to guide state selection

- After each time-step $t$ backup the state $\mathbf{s}_t$

$$V(\mathbf{s}_t) \leftarrow \max_{\mathbf{a} \in \mathcal{A}} \left[ \mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V(\mathbf{s}_{t+1}) \right]$$



# Applicability of DP

- Effective for medium-sized problems (millions of states) with known dynamics
- For large problems DP suffers Bellman's curse of dimensionality as number of states grows exponentially with number of state variables
- Even one full backup in state space can be too expensive (e.g. in backgammon $|\mathcal{S}| \approx 10^{20}$ )

# Approximate DP

- Approximate value function $V^*(\mathbf{s}) \approx \hat{V}^*(\mathbf{s}, \theta)$
- Estimate targets $y_t = \max_{\mathbf{a}_t \in \mathcal{A}} \left[\mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V^*(\mathbf{s}_{t+1}, \theta_k)\right]$
- Fit parameters of neural net to match targets $\{(\mathbf{s}_t, y_t)\}$