Acknowledgements

This material includes content adapted from instructional resources made available by David Silver as part of his Reinforcement Learning course at UCL under CC-BY-NC 4.0.

Refer to https://www.davidsilver.uk/teaching/ for full details.

MONASH University
Information Technology

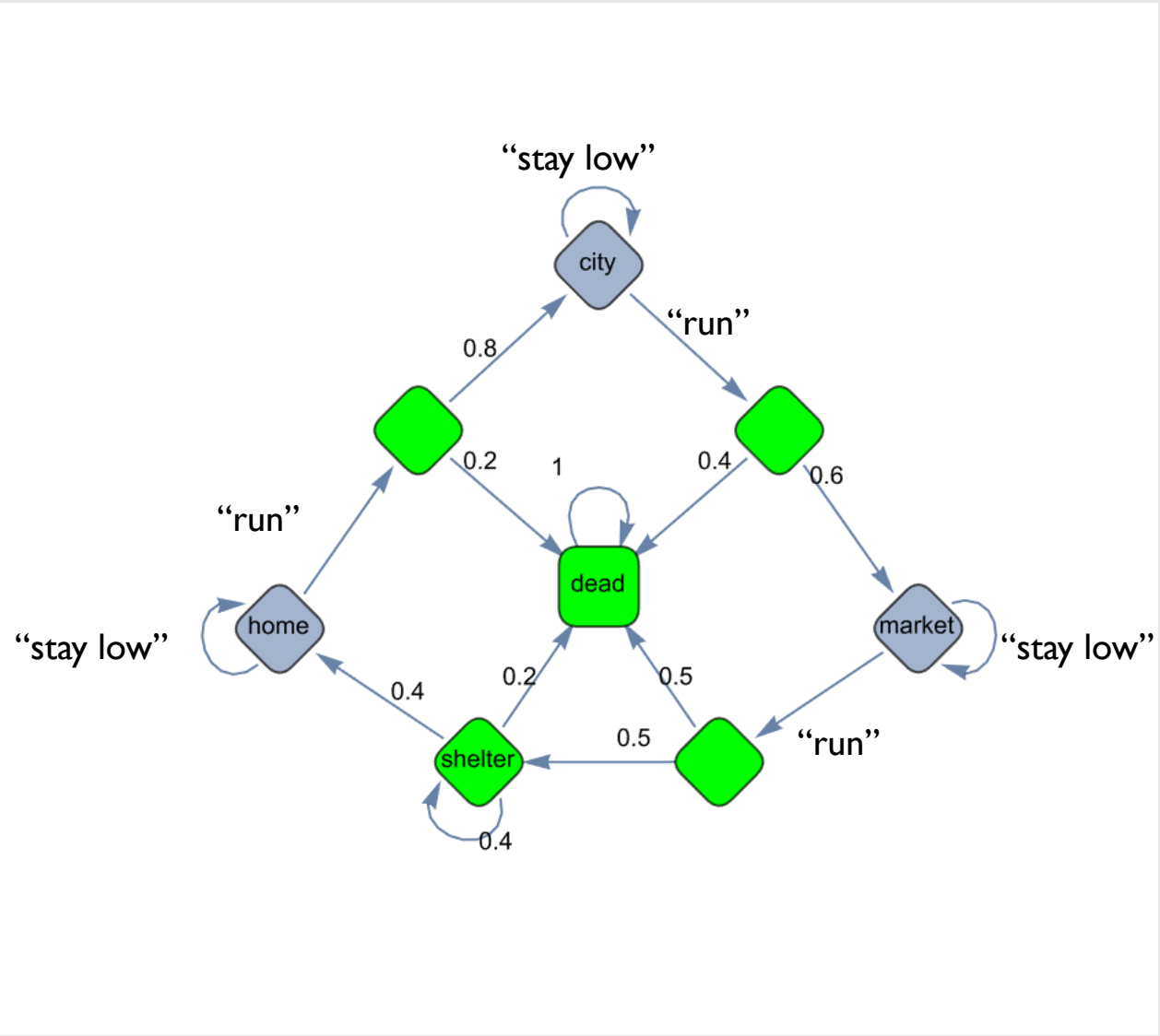
FIT5226

Multi-agent System & Collective Behaviour

Wk 3: MDPs, Bellman Equation, Policy and Value Iteration

Bernd Meyer, July 2024

Example: The 7 lives of cats (MDP)
Should I stay or should I go?



Reminder: Policy

A policy defines the agent's behaviour

It is a map from state to action, e.g.

- Deterministic policy: $a = \pi(s)$

- Stochastic policy:

$$\pi(a \mid s) = P[\text{Action} = a \mid \text{ state} = s]$$

Reminder: Model

A model captures the environment

In our context, this means a model (*P*, *R*)

- *P* predicts the next state

$$P^a_{s,s'} = P[s_{t+1} = s' \mid s_t = s, a_t = a]$$

- *R* the next reward

$$R^a_s = \mathbb{E}[R_{t+1} \mid s_t = s, a_t = a]$$

From MDP back to MRP

For any MDP $M = (S, A, P, R, \gamma)$,

a policy π and a start state S_1 together induce

- a Markov Chain (S, P^π)
- an associated state sequence $S_1, S_2, \ldots, S_n$

Reward R and discount factor γ then induce

- a Markov Reward Process $(S, P^\pi, R^\pi, \gamma)$
- a state-reward sequence $S_1, R_2, S_2, \ldots, R_n, S_n$

with $P^\pi_{s,s'} = \sum_a \pi(a \mid s) P^a_{s,s'}$ and $R^\pi_s = \sum_a \pi(a \mid s) R^a_s$

Value Functions

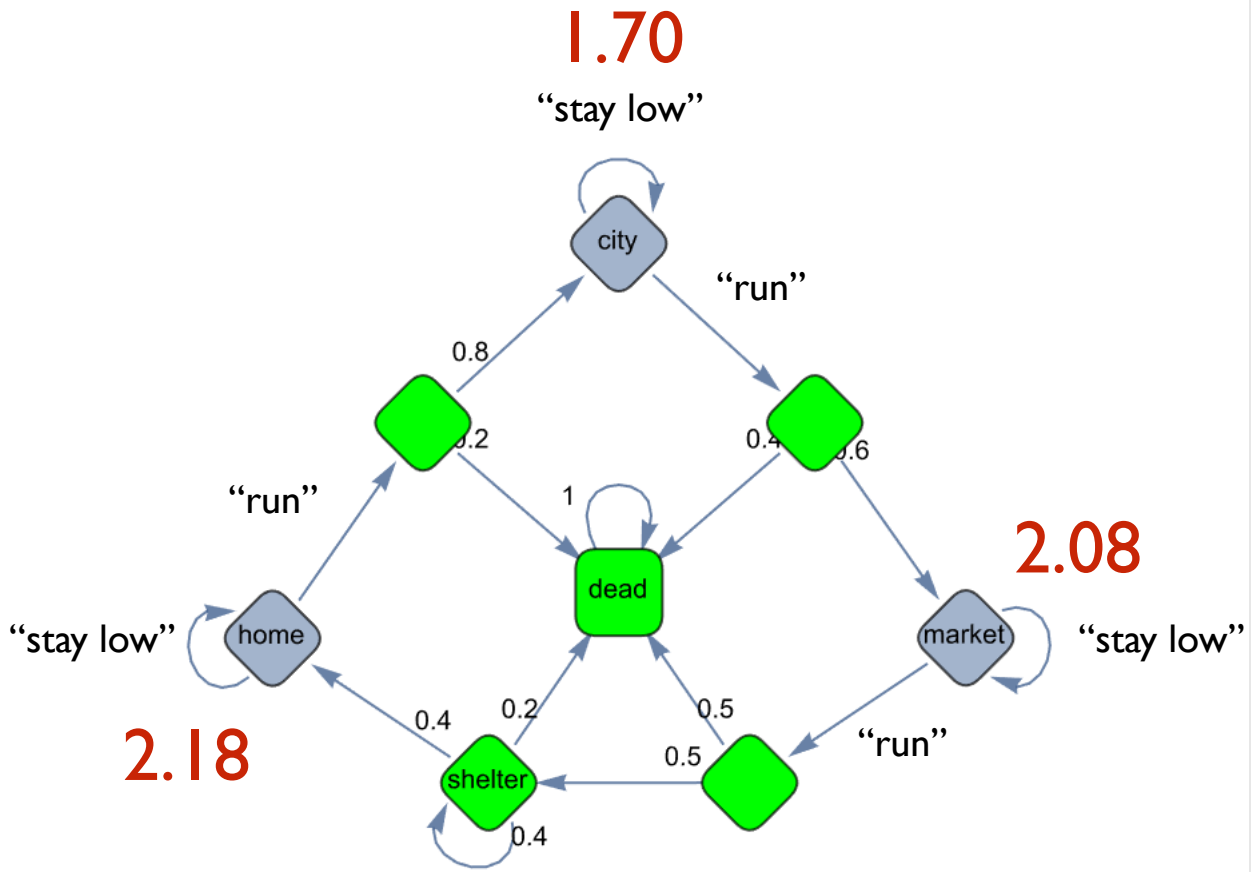
The state-value function for an MDP $M = (S, A, P, R, \gamma)$, and policy π gives the expected return from a state following that policy

- $v_\pi(s) = \mathbb{E}_\pi[G_t \mid \ S_t = s]$
- with state sequence $S_1, S_2, \ldots, S_n$

the action-state-value function gives the expected return from a state taking action a and then following that policy

- $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid \ S_t = s, A_t = a]$

Example: The 7 lives of cats (state-value)



$v_\pi(s)$ for $\pi(a \mid s) = 0.5,\ \gamma = .95$

Reminder: Value

The value of a state is the *expected return* from this state

$$v(s) = \mathbb{E}[G_t \,|\, S_t = s]$$

Reminder: the value function of an agent predicts the future reward in a state given a policy (but we not yet have a policy our transitions are pre-determined)

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \;\; \gamma^\infty R_{t+\infty} \mid \;\; S_t = s]$$

Bellman Equation (Expectation)

The state-value function can recursively be decomposed

- immediate reward and
- discounted future reward

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \ \gamma^\infty R_{t+\infty} \mid \ S_t = s]$$

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid \ S_t = s]$$

Bellman Equation (Expectation)

Both value functions can recursively be decomposed in the same way, into

- immediate reward and
- discounted future reward

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

$$q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) \mid S_t = s, A_t = a]$$

Direct Solution for Bellman Equation

The Bellman equation can be compactly written via the induced MRP in matrix form as

$$v_\pi = R^\pi + \gamma P^\pi v_\pi$$

with closed-form solution

$$v_\pi = (\mathbf{1} - \gamma P^\pi)^{-1} R^\pi$$

Iterative Policy Evaluation

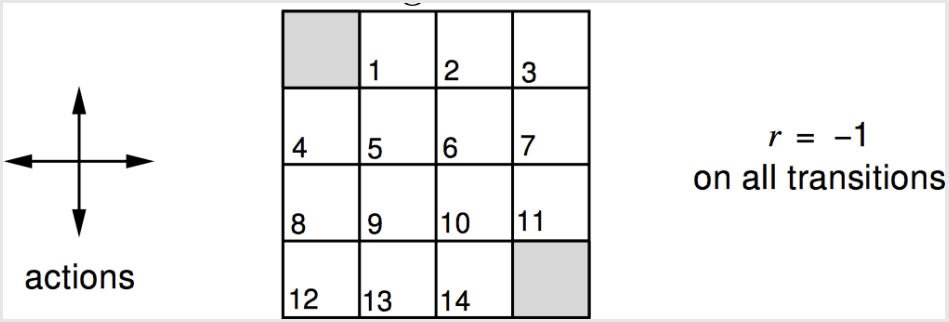
- Problem: evaluate a given policy π
- Solution: iterative application of Bellman expectation
- $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n = v_\pi$

- Update $v_{k+1}(s)$ from $v_k(s')$ where s' is a successor state of s according to Bellman Expectation equation
  - *note: subscript k is the iteration of the evaluation, not the step number*

- evaluate for all states, at each iteration

- Convergence to v_π can be proven

Example: Iterative Policy Evaluation (Computing the Value Function)



- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states 1, ..., 14
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is −1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(s \mid \_) = \pi(n \mid \_) = \pi(w \mid \_) = \pi(e \mid \_) = 0.25$$

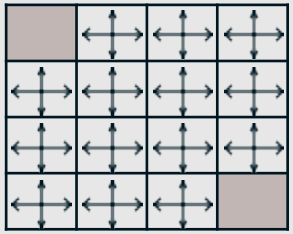
Example: Iterative Policy Evaluation

v_k for the Random Policy

Greedy Policy w.r.t. v_k

$k = 0$

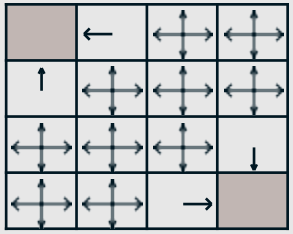
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

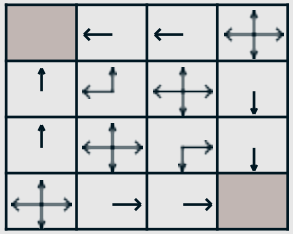
$k = 1$

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



$k = 2$

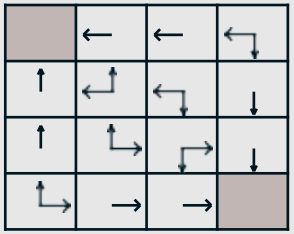
0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



Example: Iterative Policy Evaluation

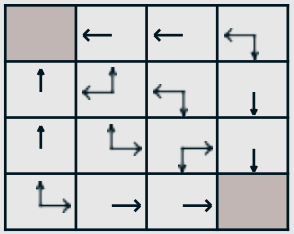
$k = 3$

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



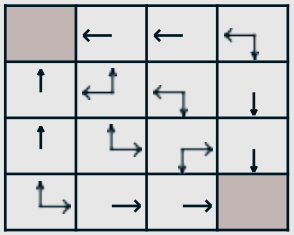
$k = 10$

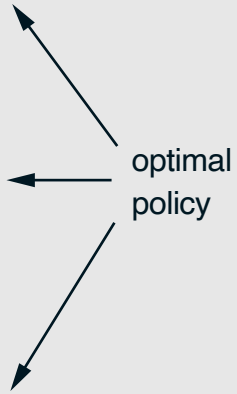
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



$k = \infty$

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





Optimal Value Function

To "solve" an MDP we must find the optimal policy. This amounts to optimising the value function over all possible policies

$$v_\star(s) = \max_\pi \; v_\pi(s)$$

with

$$v_\star(s) = \max_a \; q_\star(s, a)$$

$$q_\star(s, a) = \max_\pi \; q(s, a)$$

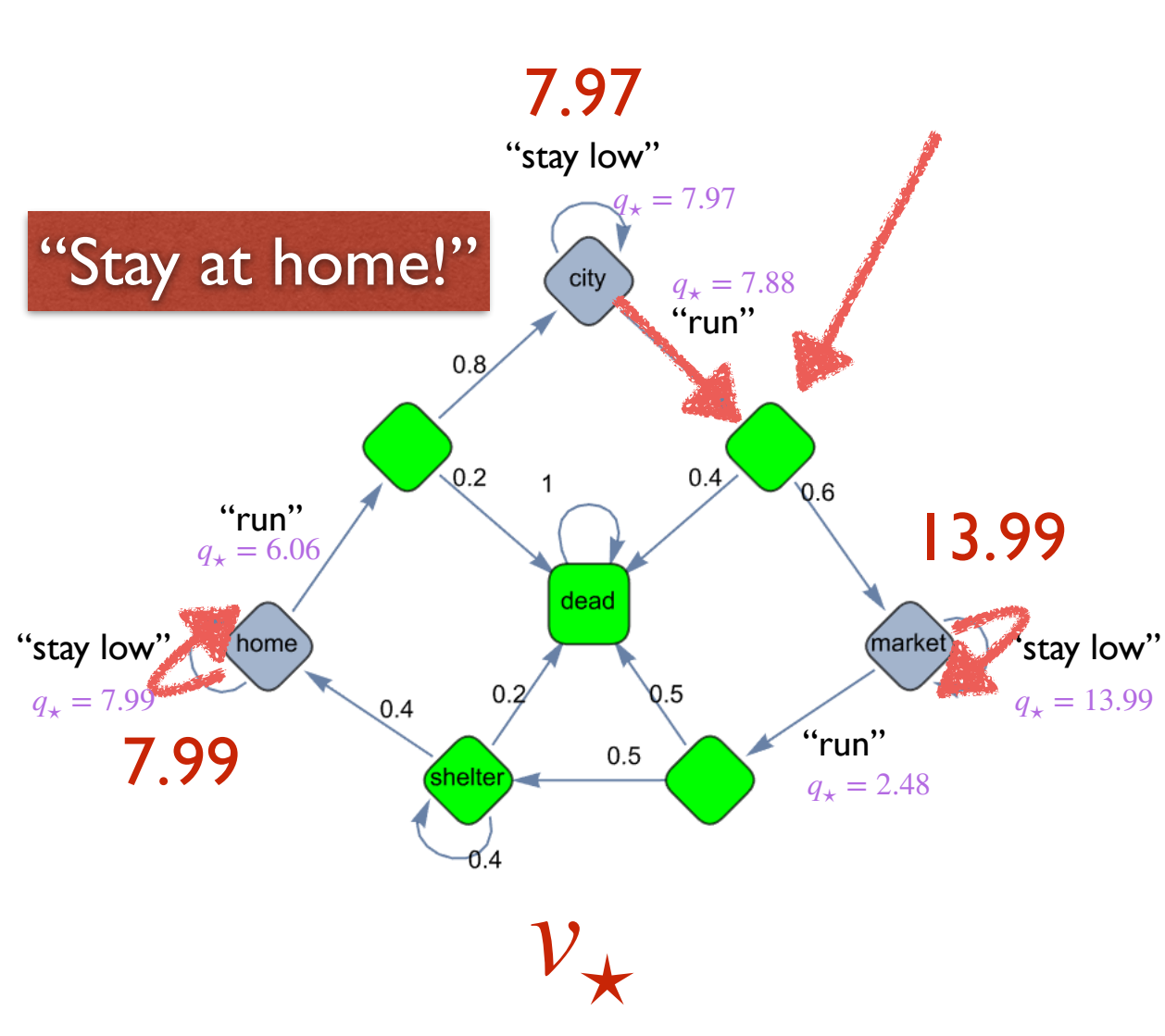
Computing the Optimal Policy

The optimal policy can be derived from $q_\star$ by acting greedily

$$\pi_\star(a \mid s) = \begin{cases} 1 \text{ if } a = \underset{a \in A}{\text{argmax}}\ q_\star(s, a) \\ 0 \text{ otherwise} \end{cases}$$

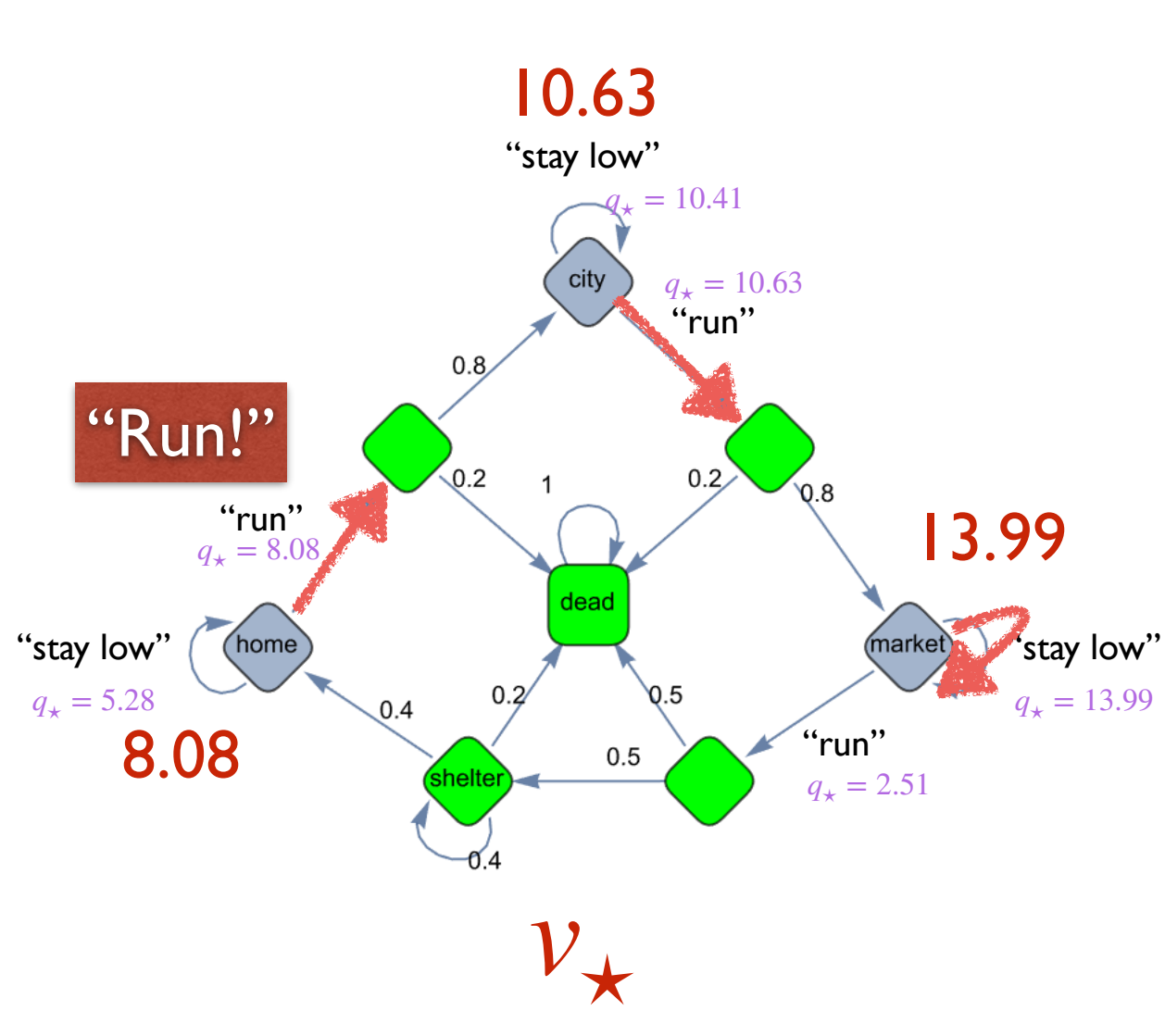
- if we know $q_\star$ we know the optimal policy directly
- there is always a deterministic optimal policy!

Optimal Policy for Cats



$\gamma = 0.95$

Optimal Policy for Cats



$\gamma = 0.95$

Reminder: Bellman Equation (Expectation)

Both value functions (for expectation) can recursively be decomposed in the same way, into

- immediate reward and
- discounted future reward

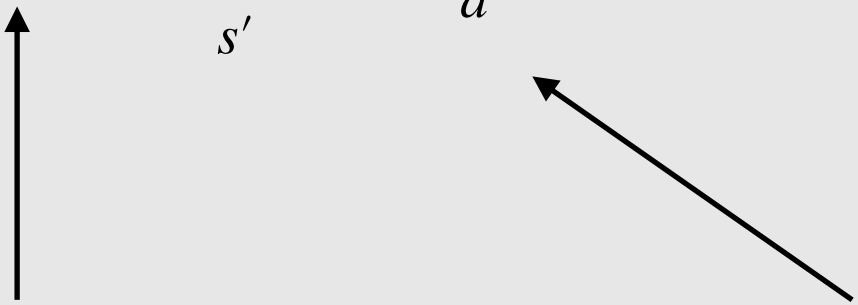
$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

$$q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) \mid S_t = s, A_t = a]$$

Bellman **Optimality** Equation for $q_\star$

the Bellman optimality equation for $q_\star$ can be recursively decomposed just like the Bellman expectation equation

$$q_\star(s,a) = R_s^a + \gamma \sum_{s'} P_{s,s'}^a \max_{a'} q_\star(s',a')$$



expected immediate reward in state s for action a

best expected future reward in state s for action a

Bellman **Optimality** Equation for $v_\star$

Likewise, the Bellman optimality equation for $v_\star$ can be recursively decomposed in the same way

$$v_\star(s) = \max_a \left(R_s^a + \gamma \sum_{s'} P_{s,s'}^a v_\star(s') \right)$$

Solving the Bellman Optimality Equation

The Bellman Optimality Equation

- is non-linear
- does not admit a closed form solution in general
- needs to be solved iteratively using
  - Policy Iteration
  - Value Iteration
  - SARSA
  - Q-Learning
  - …

Policy Iteration

Given a policy π

1. Evaluate the policy π (as shown before)

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \mid S_t = s]$$

2. Improve the policy by acting greedily with respect to v_π

$$\pi^t = \text{greedy}(v_\pi)$$

3. repeat

This process of policy improvement always converges to $\pi_\star$
but may require many iterations of improvement.

Policy Iteration

- Does the policy evaluation need to fully converge to v_π?
- Could we stop earlier? e.g. fixed number of iterations

- If fixed number of iterations, can we stop after the first one already ($k = 1$)?

- We can! This method is known as value iteration

Deterministic Value Iteration

- If we know the solution $v_\star(s')$ to subsequent states s' then solution $v_\star(s)$ can be found by one-step lookahead

$$v_\star(s) = \max_{a \in A} \left(R_s^a + \gamma \sum_{s' \in S} P_{s,s'}^a v_\star(s') \right)$$

- The idea of value iteration is to apply these updates iteratively
- Intuition: start with final rewards and work backwards
- Still works with loopy, stochastic MDPs

Value Iteration

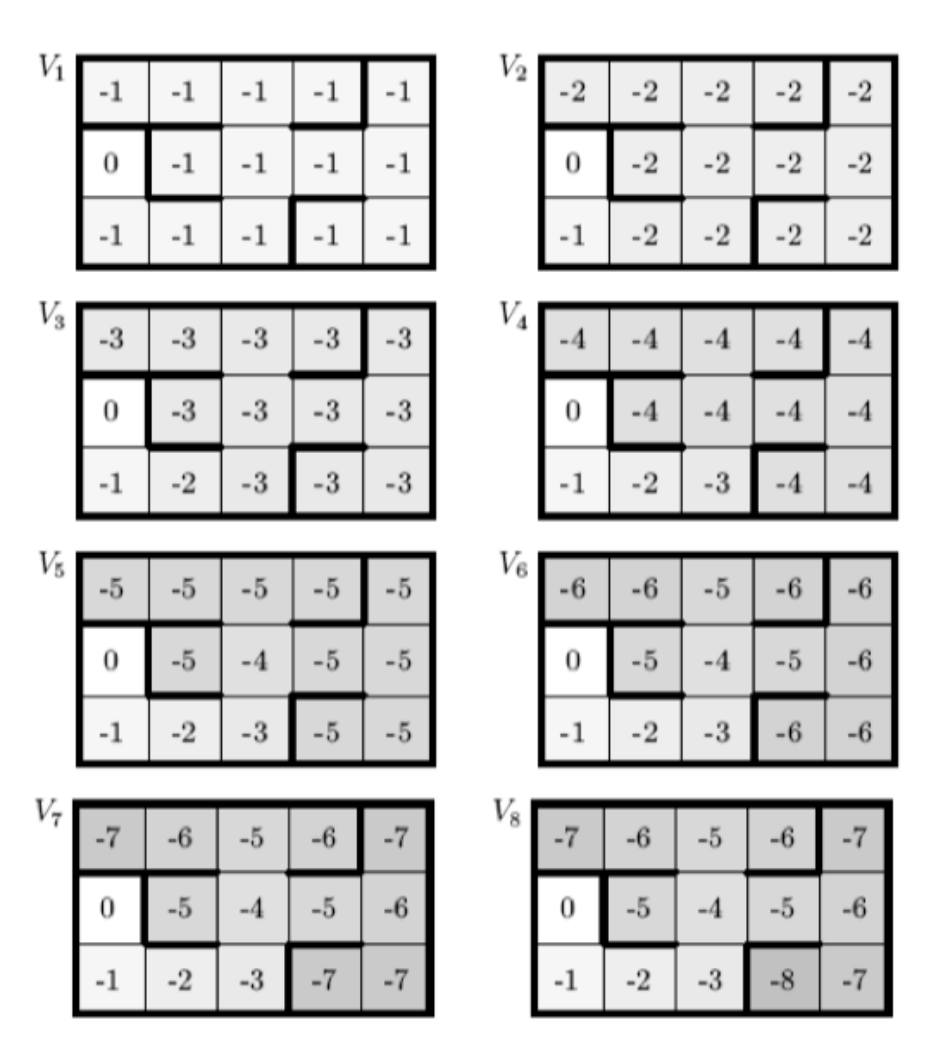
- Problem: find optimal value function $v_\star$
- Solution: iterative application of Bellman optimality equation
- $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n = v_\star$

- Update $v_{k+1}(s)$ from $v_k(s')$ where s' is a successor state of s
- at each iteration evaluate for all states

- Convergence to $v_\star$ can be proven

- There is no explicit policy! (greedy policy over $v_\star$)

Example: Shortest Path



V_1

-1	-1	-1	-1	-1
0	-1	-1	-1	-1
-1	-1	-1	-1	-1

V_2

-2	-2	-2	-2	-2
0	-2	-2	-2	-2
-1	-2	-2	-2	-2

V_3

-3	-3	-3	-3	-3
0	-3	-3	-3	-3
-1	-2	-3	-3	-3

V_4

-4	-4	-4	-4	-4
0	-4	-4	-4	-4
-1	-2	-3	-4	-4

V_5

-5	-5	-5	-5	-5
0	-5	-4	-5	-5
-1	-2	-3	-5	-5

V_6

-6	-6	-5	-6	-6
0	-5	-4	-5	-6
-1	-2	-3	-6	-6

V_7

-7	-6	-5	-6	-7
0	-5	-4	-5	-6
-1	-2	-3	-7	-7

V_8

-7	-6	-5	-6	-7
0	-5	-4	-5	-6
-1	-2	-3	-8	-7

Policy versus Value Iteration

Policy Iteration	Value Iteration
Starts with a random policy	Starts with a random value function
Algorithm is more complex	Algorithm is simpler
Guaranteed to converge	Guaranteed to converge
Requires few iterations to converge	Requires more iterations to converge

Note that we need a model (P, R) for either

Principle of Optimality

What enabled us to use these techniques (iterative recursive decomposition) was the principle of optimality.

Any optimal policy can be subdivided into two components:

- an optimal first action $A*$
- an optimal policy from the successor state S

Theorem (Principle of Optimality)

A policy π achieves the optimal value from state s, $v_\pi(s) = v_\star(s)$ iff for any state s' reachable form s under π

π achieves the optimal value from s': $v_\pi(s') = v_\star(s')$

Summary: Dynamic Programming Algorithms

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

- Algorithms are based on state-value function $v_\pi(s)$ or $v_\star(s)$
  - Complexity $O(mn^2)$ per iteration, for *m* actions and *n* states
- Could do the same for action-value function $q_\pi(s,a)$ or $q_\star(s,a)$
  - Complexity $O(m^2n^2)$ per iteration

Take home lessons

- MDPs are the fundamental modelling framework for understanding RL
- MDP = MRP + actions; MRP = MC + rewards
- state-value functions describe how desirable a state is; state-action-value functions describe how good it is to take a particular action in a given state
- finding the best policy (behaviour) amounts to finding the policy that optimises the state values
- This is done on the basis of the Bellman equation
- The principle of optimality allows us to recursively decompose the Bellman optimality equation
- Normally, it needs to be solved iteratively by policy iteration or value iterations
- These methods require us to know a model (R, P) of the MDP