A Brief Introduction to Reinforcement Learning and Markov Decision Processes

Eirik Fagerhaug
Norwegian University of Science and Technology

Slides Adapted from/based on:

- Slides from Erlend Coates
- Slides from Hanna Hajishirzi
- Slides from Deepmind (Hado von Hasselt)
- Grokking DRL (Miguel Morales)
- AI: A Modern Approach (Russel and Norvig)

Reading Material

Russel and Norvig;

- Chapter 16.2.1 (not including convergence proof)
- Chapter 16.4.0 (only introduction)
- Chapter 23.2.3
- Chapter 23.3

- Recap
- Solving MDPs
- Temporal-difference Q Learning

A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$
- A transition model $P(s'|s,a)$
- A reward function $R(s,a,s')$
- A start state S_0

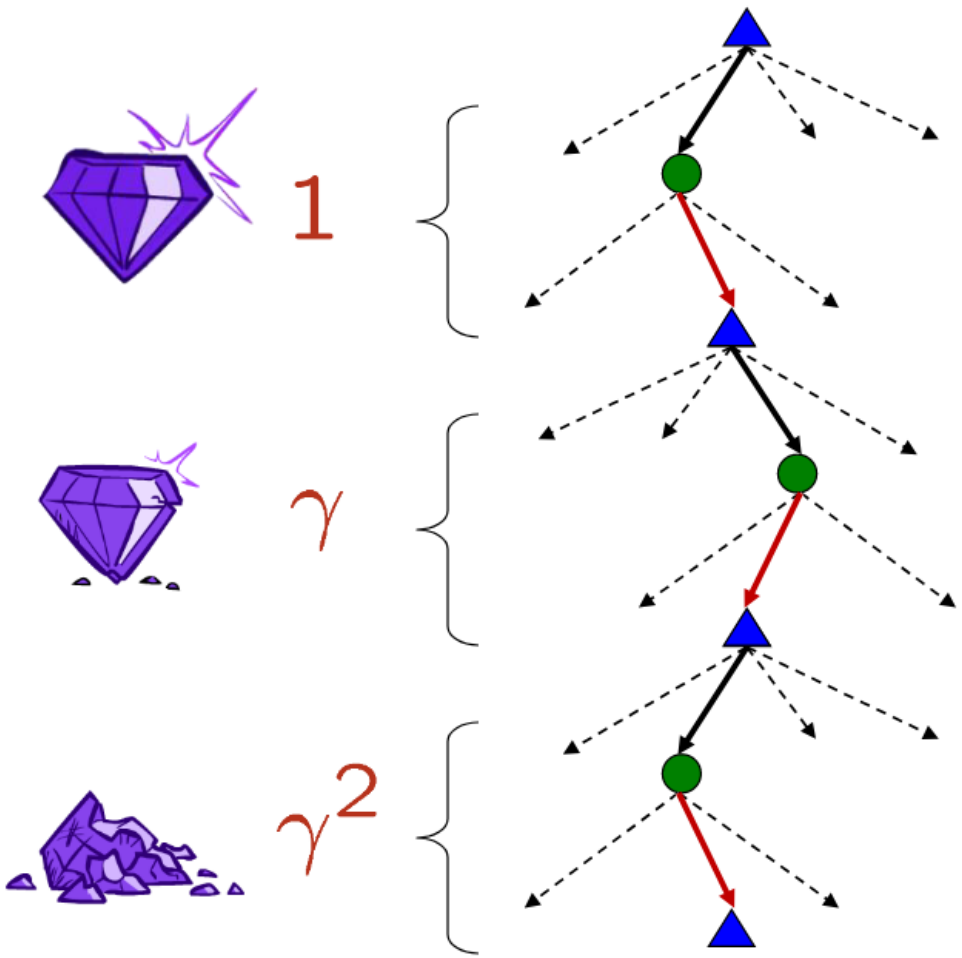
Sometimes

- A discount factor γ
- The horizon H

MDP's can be considered non-deterministic search problems

Discounting

- We introduce a discount factor $\gamma \in [0, 1]$
- The discount factor trades off the importance of immediate vs. long-term reward
- For each timestep, we multiply the discount once



Example, $\gamma = 0.5$:

$$U_h([1, 2, 3]) = 1 * \gamma^0 + 2 * \gamma^1 + 3 * \gamma^2$$

$$U_h([1, 2, 3]) = 1 * 1 + 2 * 0.5 + 3 * 0.25$$

$$U_h([1, 2, 3]) < U_h([3, 2, 1])$$

Utility Function

The utility of a state is the expected reward for the next step plus the discounted utility of the subsequent state, assuming that the agent chooses the optimal action.

Utility Function:

$$U^{\pi^*}(s) = U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

This is also called a Bellman Equation, after Richard E. Bellman

Optimal policy:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

Q-Function

(Action-utility function)

Q-Function w/r to the utility function:

$$U(s) = \max_a Q(s, a)$$

Optimal policy w/r the Q-function:

$$\pi^*(s) = \arg\max_a Q(s, a)$$

The Q-function as a Bellman Equation:

$$Q(s, a) = \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')]$$

Q-Function

(Action-utility function)

The Q-function as a Bellman Equation:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

Q-Function

(Action-utility function)

The Q-function as a Bellman Equation:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

The action-utility function is the expected utility of taking a given action in a given state.

The state is a unique and self-contained configuration of the environment.

An action is what the agent does to affect the environment.

Q-Function

(Action-utility function)

The Q-function as a Bellman Equation:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

- The transition model describes the outcome of each action in each state.
- For stochastic environments; the probability of reaching state s' if action a is done in state s.

Q-Function

(Action-utility function)

The Q-function as a Bellman Equation:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

Reward function: Reward received from the transition $(s,a) \rightarrow a'$

Q-Function

(Action-utility function)

The Q-function as a Bellman Equation:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

The expected utility from state s'

Q-Function

(Action-utility function)

The Q-function as a Bellman Equation:

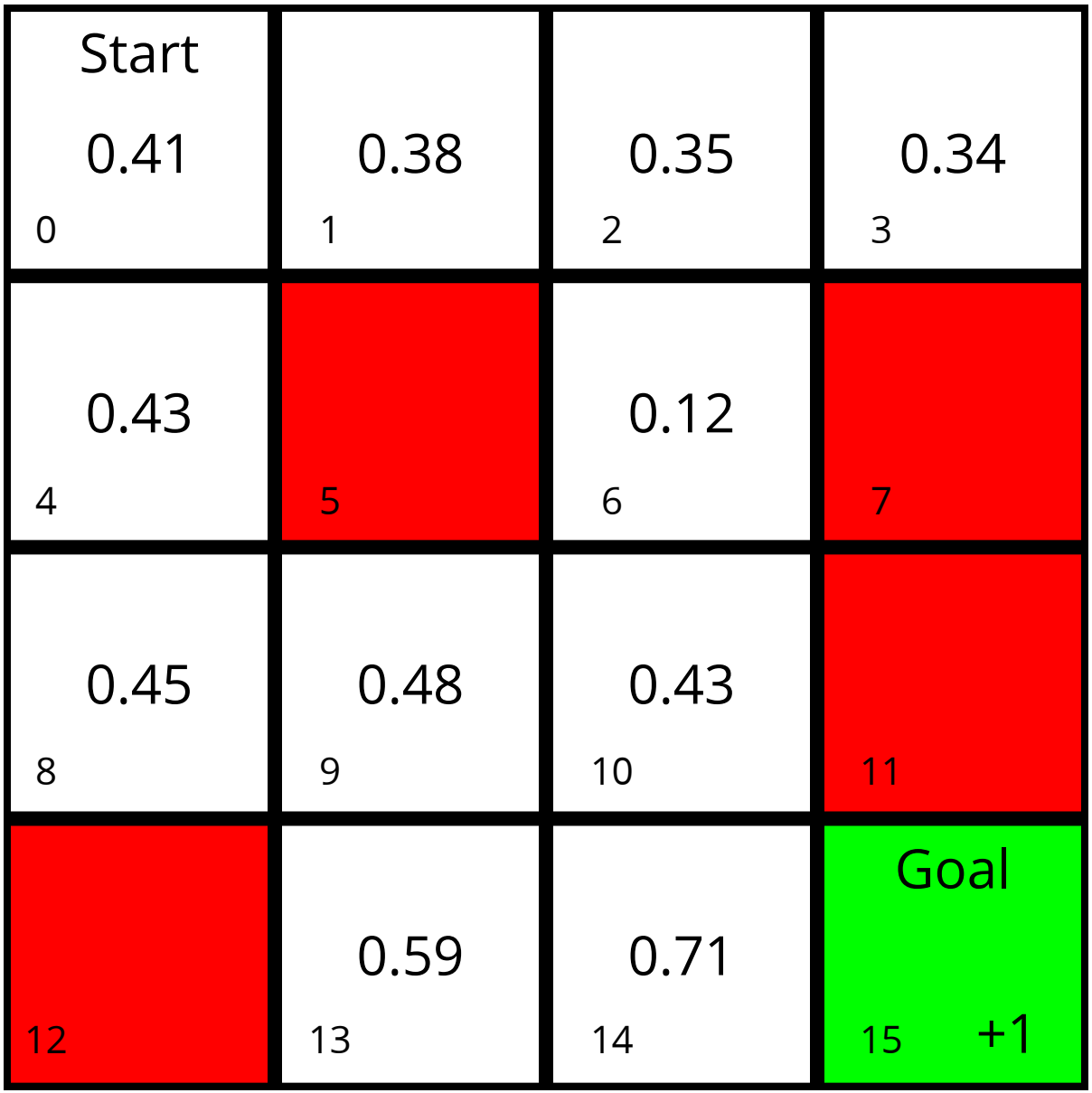
$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

Discount factor

- Recap

→ • Solving MDPs (non-RL methods)

- Temporal-difference Q Learning


Start
0.41
0
0.38
1
0.35
2
0.34
3
0.43
4
5
0.12
6
7
0.45
8
0.48
9
0.43
10
11
12
0.59
13
0.71
14
Goal
15
+1

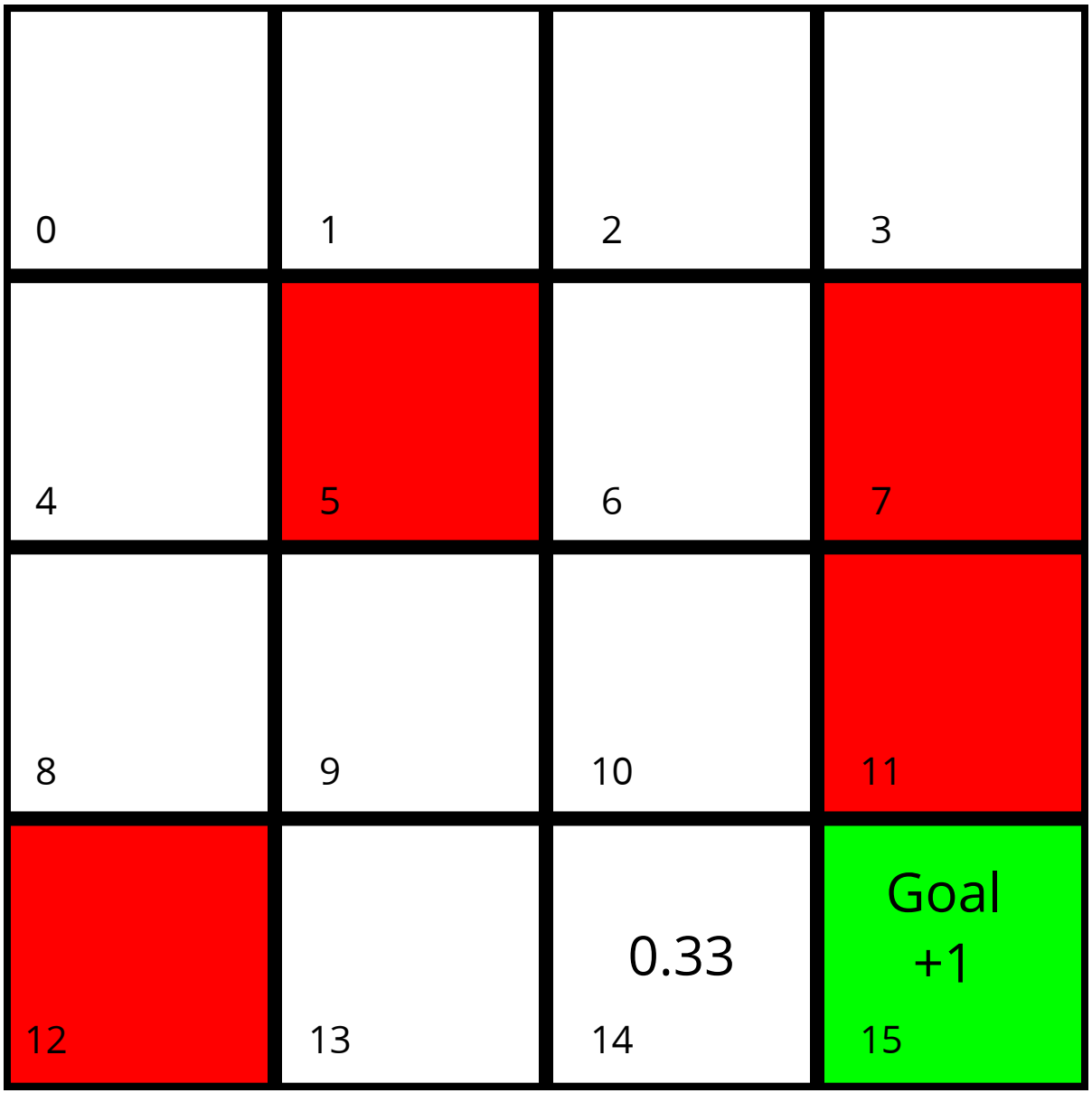
Bellman Update

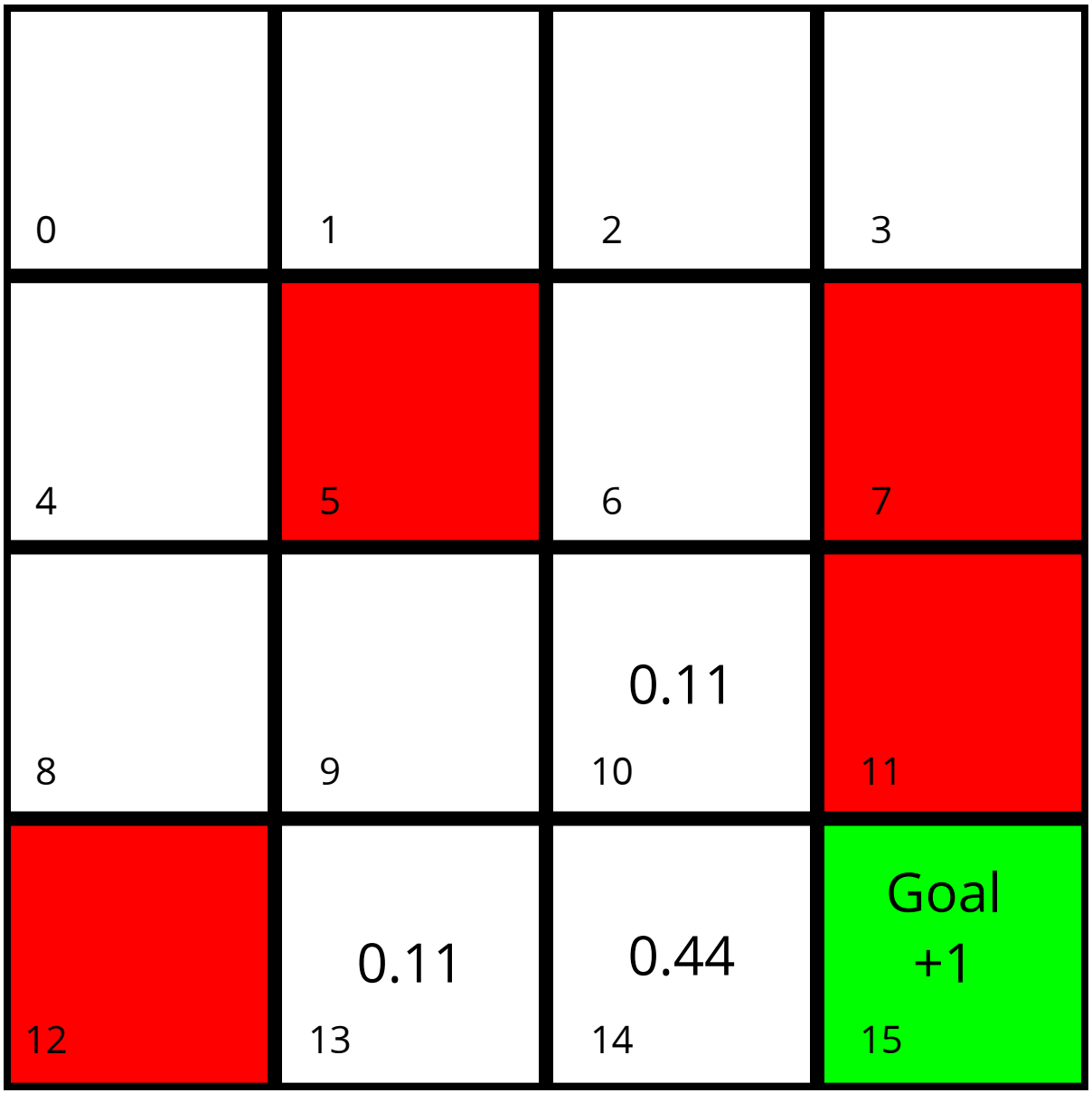
Utility Function / Bellman equation:

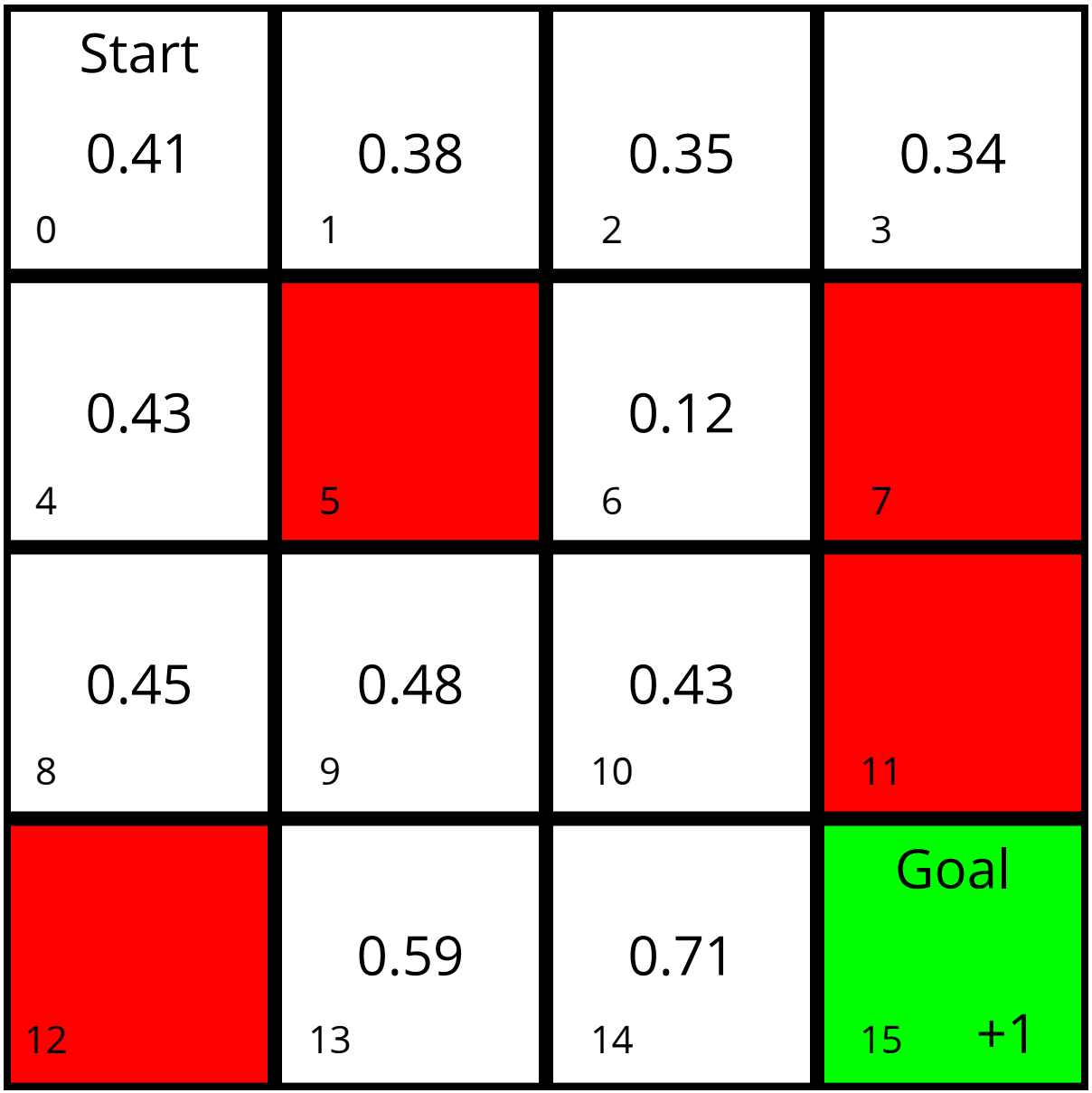
$$U^{\pi^*}(s) = U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

Bellman Update:

$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U_i(s')]$$


0
1
2
3
4
5
6
7
8
9
10
11
12
13
0.33
14
Goal
+1
15


0
1
2
3
4
5
6
7
8
9
0.11
10
11
12
0.11
13
0.44
14
Goal
+1
15


Start
0.41
0
0.38
1
0.35
2
0.34
3
0.43
4
5
0.12
6
7
0.45
8
0.48
9
0.43
10
11
12
0.59
13
0.71
14
Goal
15
+1

Disadvantages:

- Takes a long time to converge (especially for large state-spaces)
- Requires full environment observability

- Requires transparent transition model
- Requires transparent reward function

Disadvantages:

- Takes a long time to converge (especially for large state-spaces)
- Requires full environment observability

- Requires transparent transition model
- Requires transparent reward function

Can be solved with similar methods

Disadvantages:

- Takes a long time to converge (especially for large state-spaces)
- Requires full environment observability

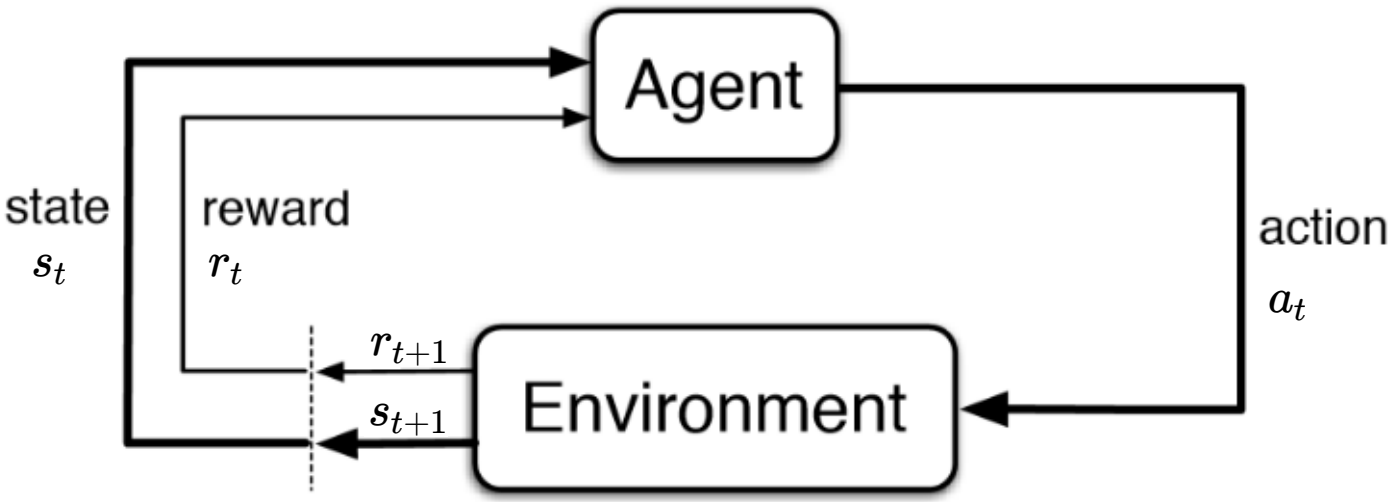
+

- Requires transparent transition model
- Requires transparent reward function

Can be solved by interacting with the environment: Reinforcement Learning

- Recap
- Solving MDPs (non-RL methods)
- Temporal-difference Q Learning

Agent-Environment interaction



- At each step *t* the agent:
  - Receives state s_t (and reward r_t)
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_{t+1} (and reward r_{t+1})

Temporal Difference Q-Learning

TODO

Bellman Update

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Bellman Update

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

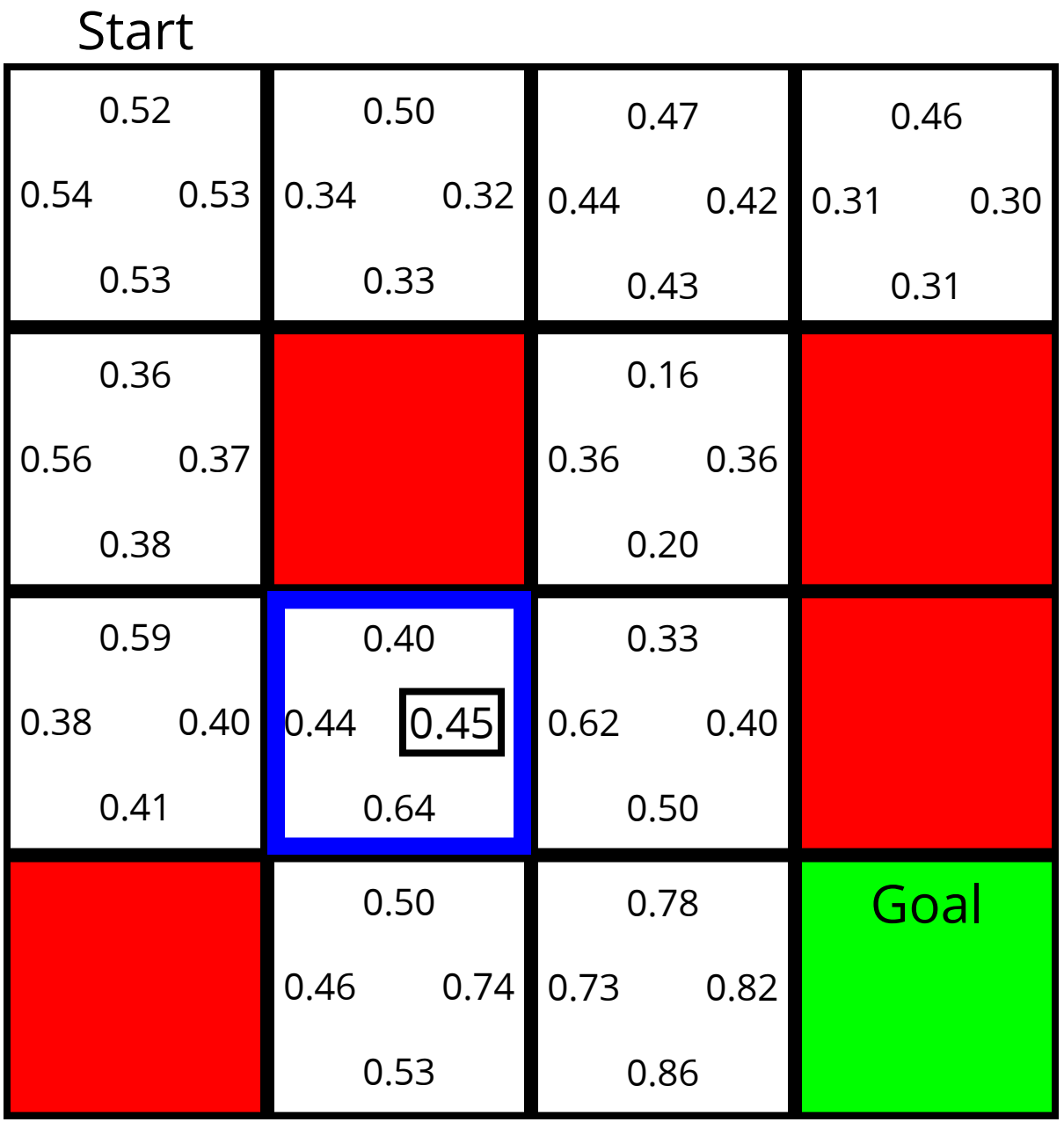
Temporal-difference / Error-signal

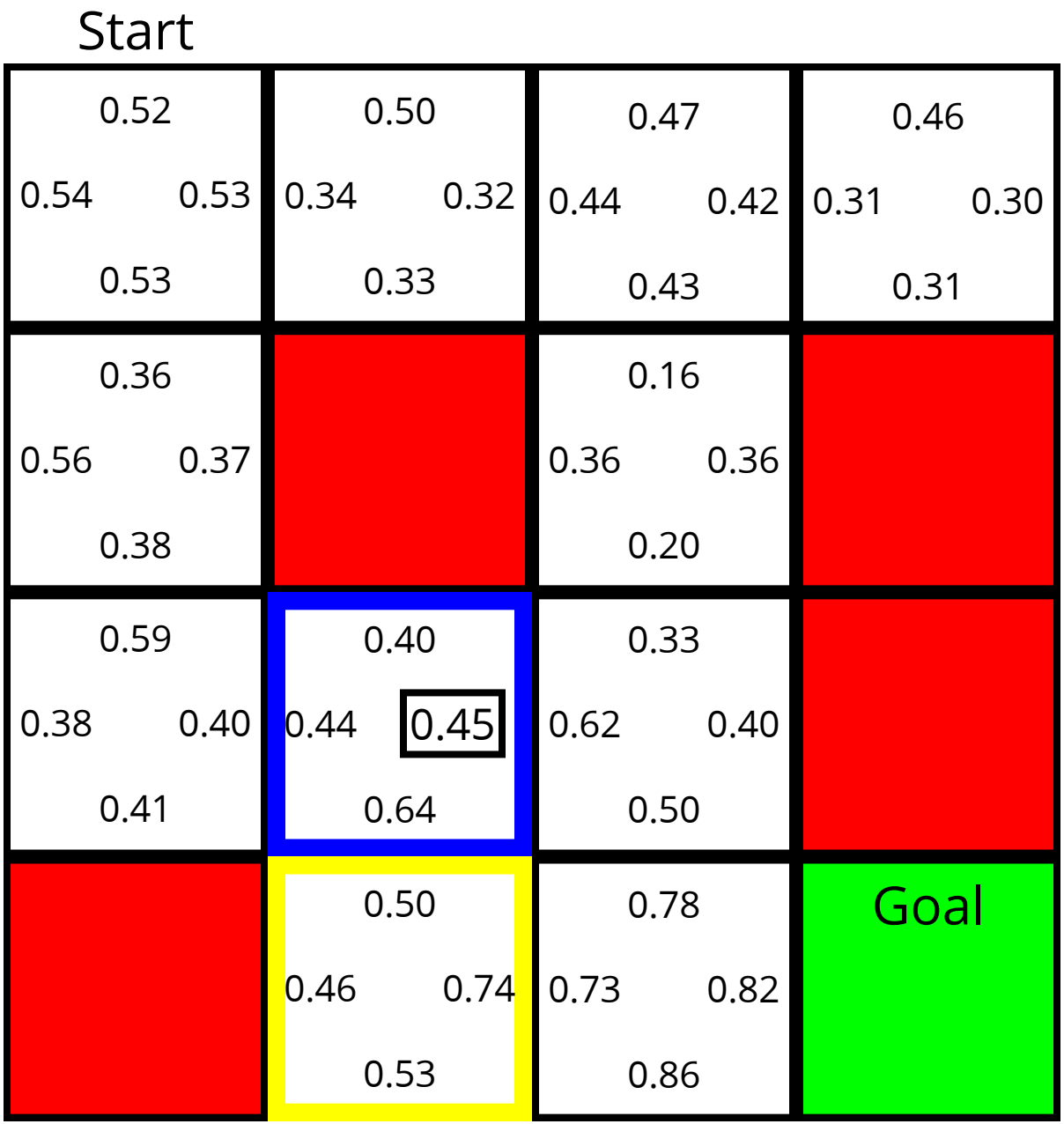
Bellman Update

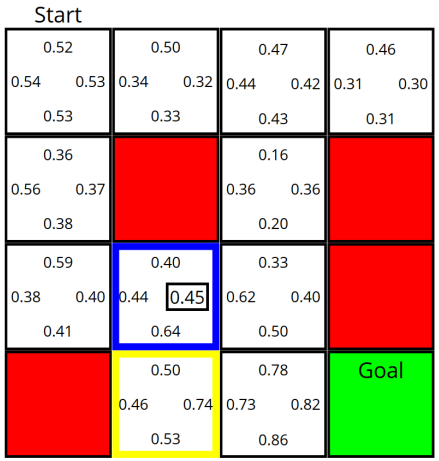
$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Start

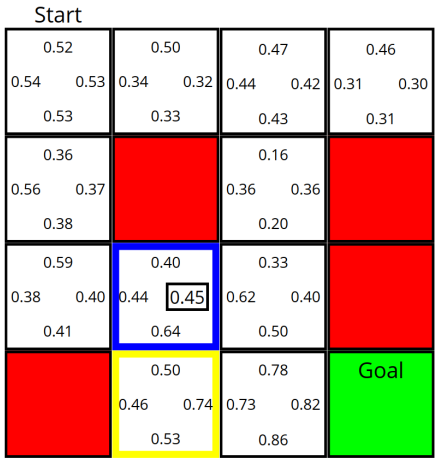
0.52 0.54 0.53 0.53	0.50 0.34 0.32 0.33	0.47 0.44 0.42 0.43	0.46 0.31 0.30 0.31
0.36 0.56 0.37 0.38		0.16 0.36 0.36 0.20	
0.59 0.38 0.40 0.41	0.40 0.44 0.45 0.64	0.33 0.62 0.40 0.50	
	0.50 0.46 0.74 0.53	0.78 0.73 0.82 0.86	Goal


Start
0.52
0.54 0.53
0.53
0.50
0.34 0.32
0.33
0.47
0.44 0.42
0.43
0.46
0.31 0.30
0.31
0.36
0.56 0.37
0.38
0.16
0.36 0.36
0.20
0.59
0.38 0.40
0.41
0.40
0.44 0.45
0.64
0.33
0.62 0.40
0.50
0.50
0.46 0.74
0.53
0.78
0.73 0.82
0.86
Goal


Start
0.52
0.54
0.53
0.53
0.50
0.34
0.32
0.33
0.47
0.44
0.42
0.43
0.46
0.31
0.30
0.31
0.36
0.56
0.37
0.38
0.16
0.36
0.36
0.20
0.59
0.38
0.40
0.41
0.40
0.44
0.45
0.64
0.33
0.62
0.40
0.50
0.50
0.46
0.74
0.53
0.78
0.73
0.82
0.86
Goal

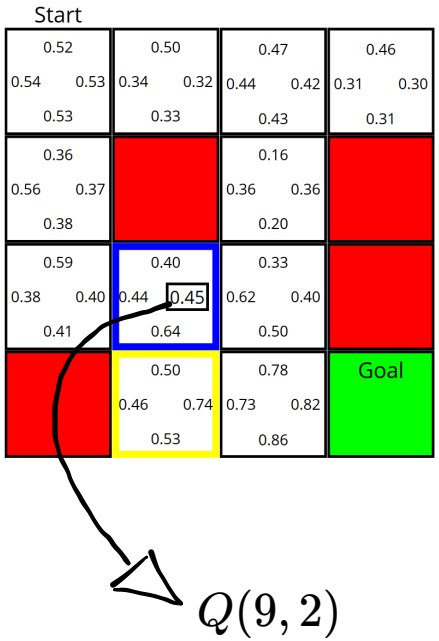


$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$



$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

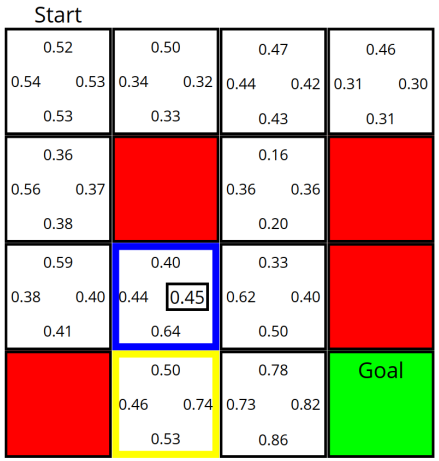
$$Q(9,2) \leftarrow Q(9,2) + \alpha[R(9,2,13) + \gamma \max_{a'} Q(13,a') - Q(9,2)]$$



$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$$Q(9,2) \leftarrow Q(9,2) + \alpha[R(9,2,13) + \gamma \max_{a'} Q(13,a') - Q(9,2)]$$

$$Q(9,2) = 0.45$$

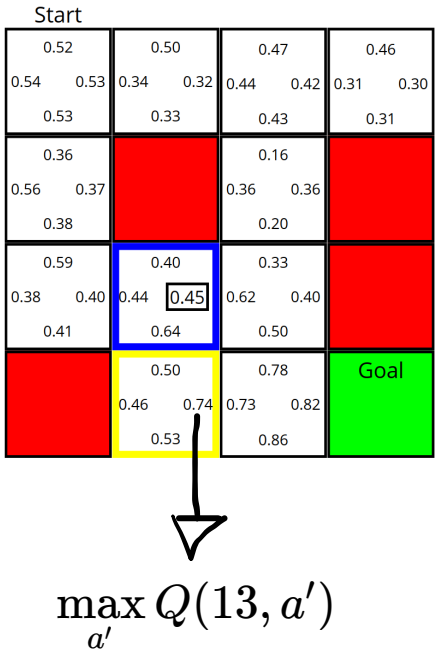


$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$$Q(9,2) \leftarrow Q(9,2) + \alpha[R(9,2,13) + \gamma \max_{a'} Q(13,a') - Q(9,2)]$$

$$Q(9,2) = 0.45$$

$$R(9,2,13) = 0$$



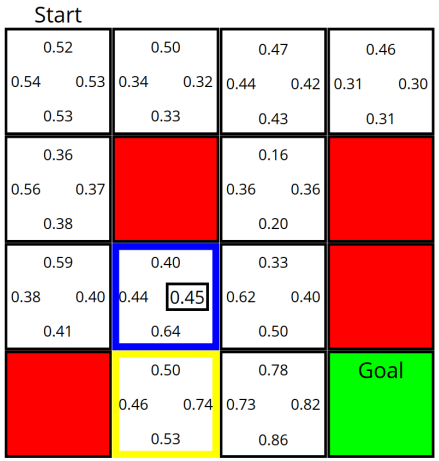
$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$$Q(9,2) \leftarrow Q(9,2) + \alpha[R(9,2,13) + \gamma \max_{a'} Q(13,a') - Q(9,2)]$$

$$Q(9,2) = 0.45$$

$$R(9,2,13) = 0$$

$$\gamma \max_{a'} Q(13,a') = \gamma 0.74$$



$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$$Q(9,2) \leftarrow Q(9,2) + \alpha[R(9,2,13) + \gamma \max_{a'} Q(13,a') - Q(9,2)]$$

$$Q(9,2) = 0.45$$

$$R(9,2,13) = 0$$

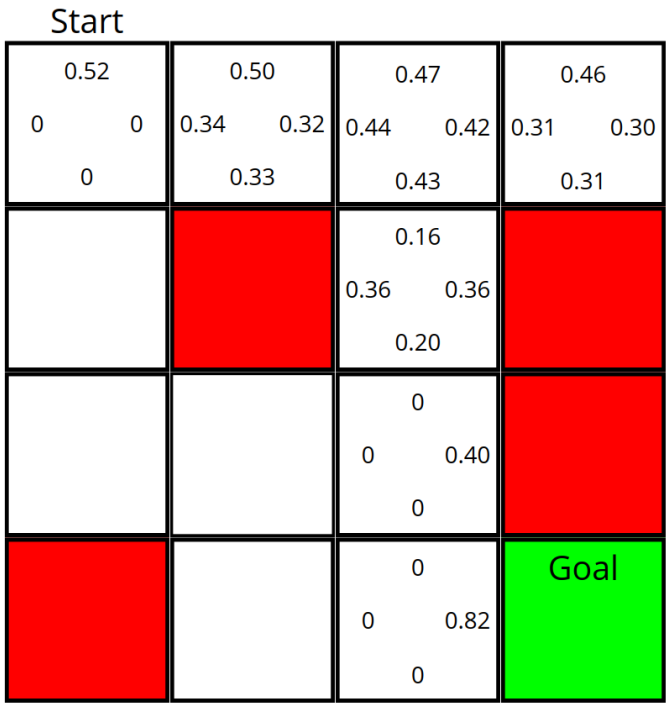
$$\gamma \max_{a'} Q(13,a') = \gamma 0.74$$

With: $\gamma = 0.98, \alpha = 0.5$:

$$Q(9,2) \leftarrow 0.45 + 0.5[0 + 0.98(0.74) - 0.45]$$

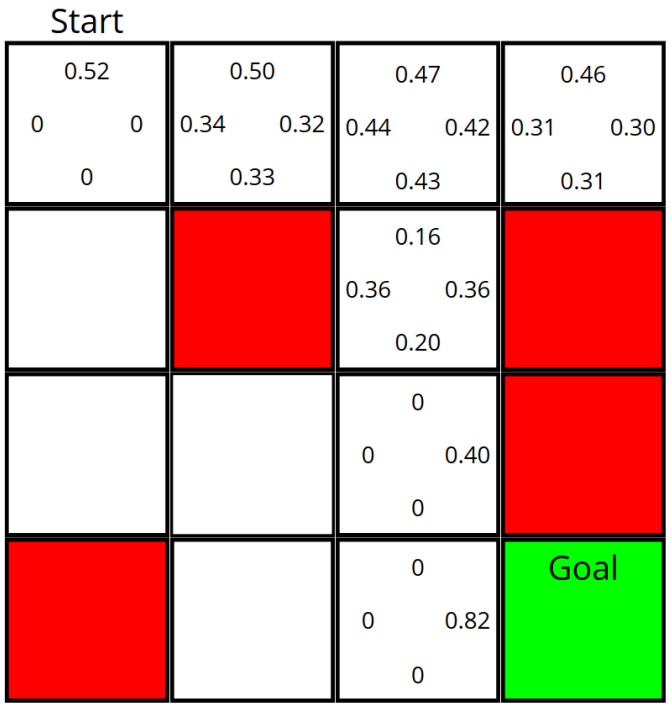
$$Q(9,2) \leftarrow 0.5876$$

Exploration vs Exploitation



- Explore to avoid non-optimal policies (think getting stuck at a local-maximum)
- Exploit to focus on "promising" states.
- Ideally; start with high exploration, end with high exploitation.

Exploration vs Exploitation



- Explore to avoid non-optimal policies (think getting stuck at a local-maximum)
- Exploit to focus on "promising" states.
- Ideally; start with high exploration, end with high exploitation.

Epsilon-greedy policy:

$$a \leftarrow \begin{cases} \arg\max_a Q(s,a), & \text{with probability } 1-\epsilon \\ a \sim Uniform(\{a_1 ... a_k\}), & \text{otherwise} \end{cases}$$

- Recap
- Solving MDPs (non-RL methods)
- Temporal-difference Q Learning

Questions?