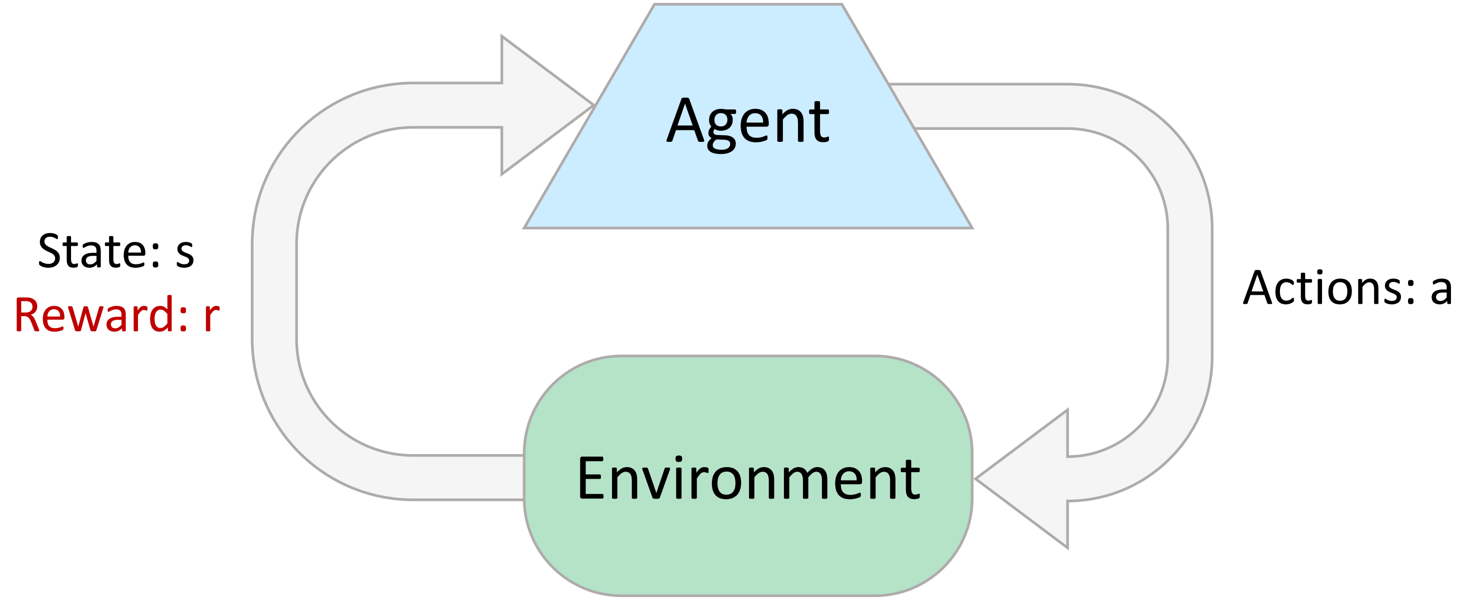
Reinforcement learning

Reinforcement learning

- Basic idea:
  - ✓ Receive feedback in the form of rewards
  - ✓ Agent's utility is defined by the reward function
  - ✓ Must (learn to) act so as to maximize expected rewards
  - ✓ All learning is based on observed samples of outcomes!

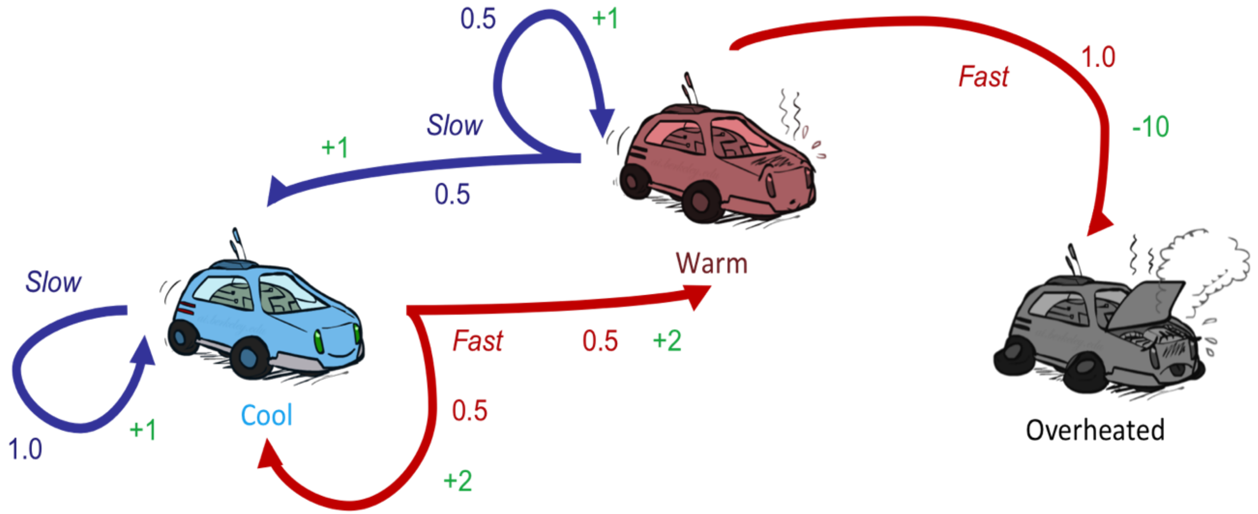


Reinforcement learning

- Examples:
  - ✓ How a 2 year old child walk?
  - ✓ How we recognize colors?
  - ✓ How to deal with situations we haven’t faced or encountered before?

MDP: what we have seen?

- A Markov decision process (MDP) has:
  - ✓ A set of states s $\in$ S
  - ✓ A set of actions (per state) A
  - ✓ A model T(s,a,s')
  - ✓ A reward function R(s,a,s')
- A policy π(s)



Reinforcement learning

- Still assume a Markov decision process (MDP):
  - ✓ A set of states $s \in S$
  - ✓ A set of actions (per state) A
  - ✓ A model $T(s,a,s')$
  - ✓ A reward function $R(s,a,s')$
- Still looking for a policy $\pi(s)$
- New twist: don't know T or R
  - ✓ I.e. we don't know which states are good or what the actions do
  - ✓ Must actually try out actions and states to learn

Learning

- Online: Reinforcement

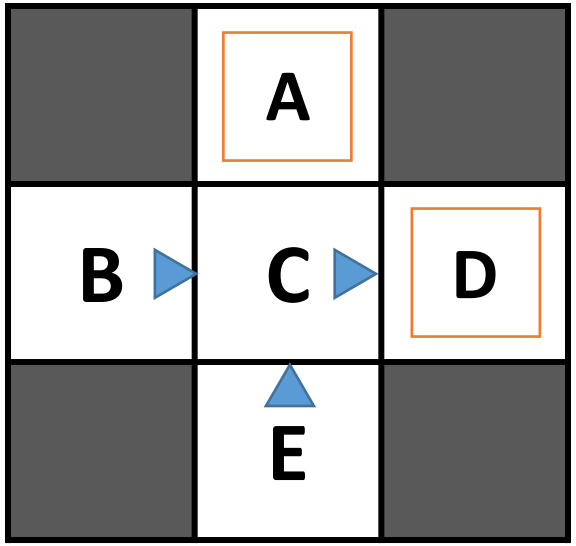
- Offline: MDPs

Model-Based Learning

- Model-Based Idea:
  - ✓ Learn an approximate model based on experiences
  - ✓ Solve for values as if the learned model were correct
- Step 1: Learn empirical MDP model
  - ✓ Count outcomes s’ for each s, a
  - ✓ Normalize to give an estimate of $\widehat{T}(s,a,s')$
  - ✓ Discover each $\widehat{R}(s,a,s')$ when we experience (s, a, s’)
- Step 2: Solve the learned MDP
  - ✓ For example, use value iteration, as before

Model-Based Learning

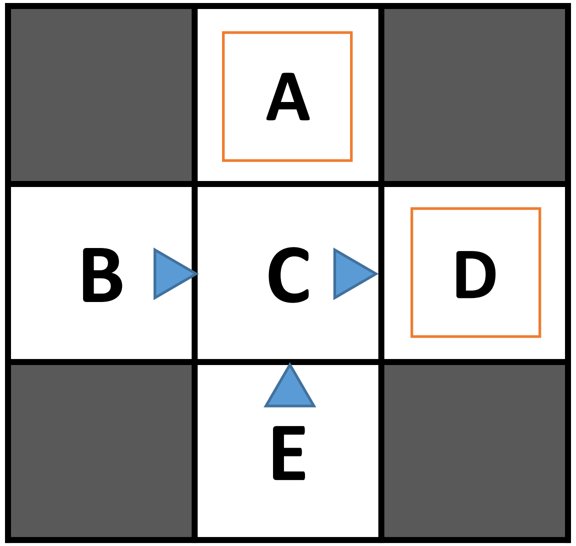
Input Policy π



Assume: $\gamma = 1$

Model-Based Learning

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Ep 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Ep 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Ep 3

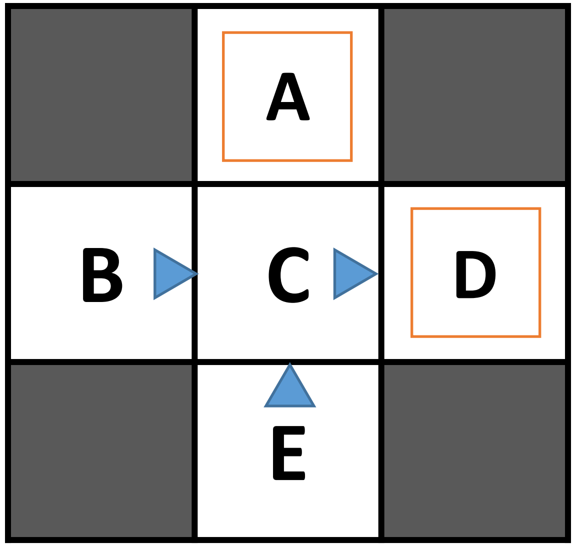
E, north, C, -1
C, east, D, -1
D, exit, x, +10

Ep 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Model-Based Learning

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Ep 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Ep 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Ep 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Ep 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$$\hat{T}(s, a, s')$$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
…

$$\hat{R}(s, a, s')$$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
…

Model-Free Learning

Example: Expected Age

Goal: Compute expected age of 3ENC6-10 students

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Model-Free Learning: Passive Reinforcement Learning

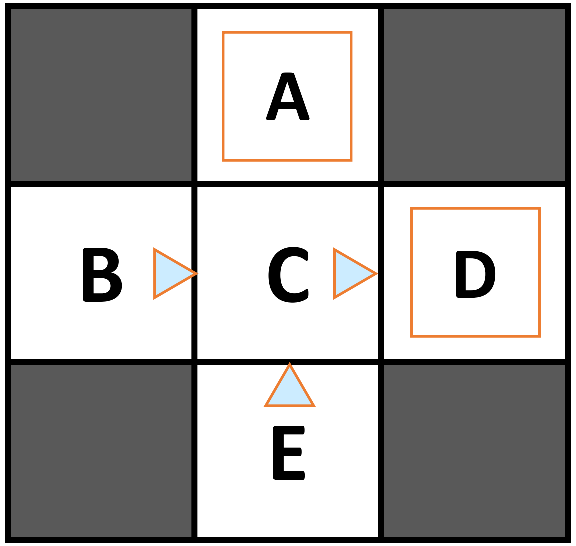
- Simplified task: policy evaluation
  - ✓ Input: a fixed policy $\pi(s)$
  - ✓ You don't know the transitions T(s,a,s')
  - ✓ You don't know the rewards R(s,a,s')
  - ✓ Goal: learn the state values
- In this case:
  - ✓ Learner is "along for the ride"
  - ✓ No choice about what actions to take
  - ✓ Just execute the policy and learn from experience
  - ✓ This is NOT offline planning! You actually take actions in the world.

Model-Free Learning: Passive Reinforcement Learning: Direct Evaluation

- Goal: Compute values for each state under π
- Idea: Average together observed sample values
  - ✓ Act according to π
  - ✓ Every time you visit a state, write down what the sum of discounted rewards turned out to be
  - ✓ Average those samples
- This is called direct evaluation.

Model-Free Learning: Passive Reinforcement Learning: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Exp 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Exp 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

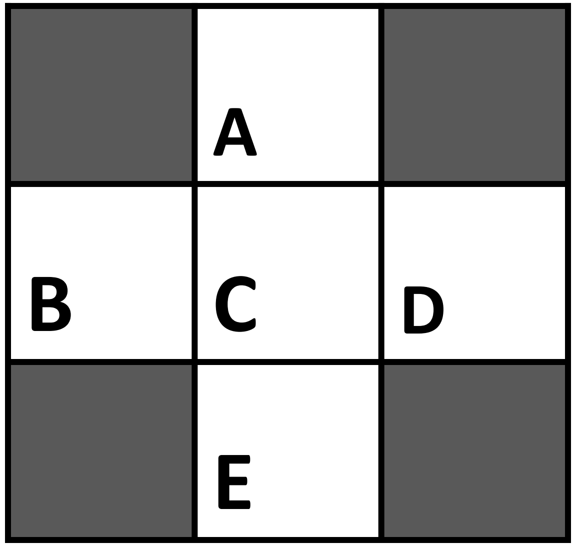
Exp 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Exp 4

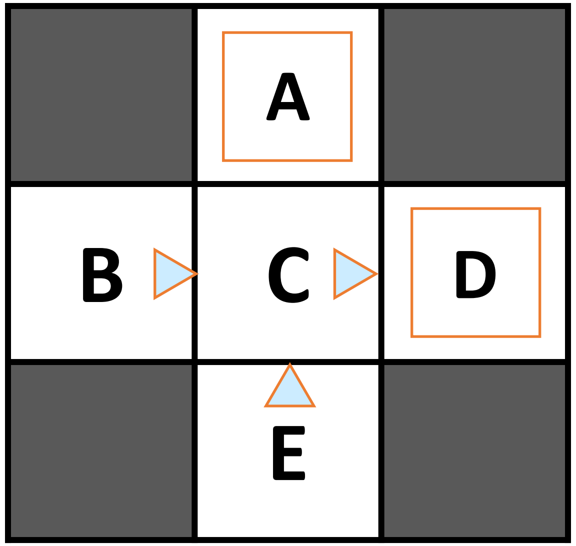
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



Model-Free Learning: Passive Reinforcement Learning: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Exp 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Exp 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

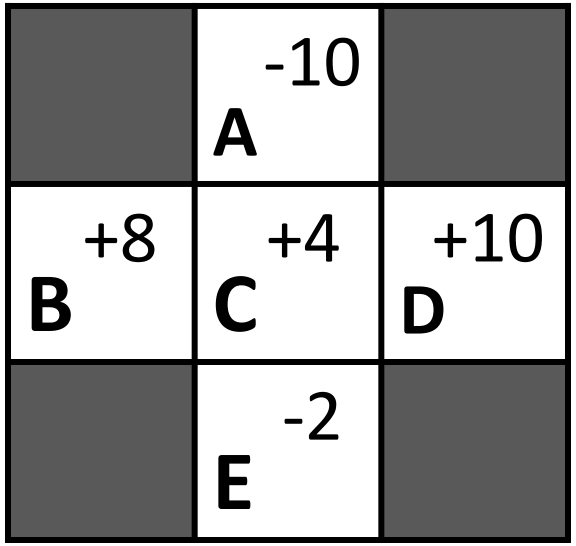
Exp 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Exp 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

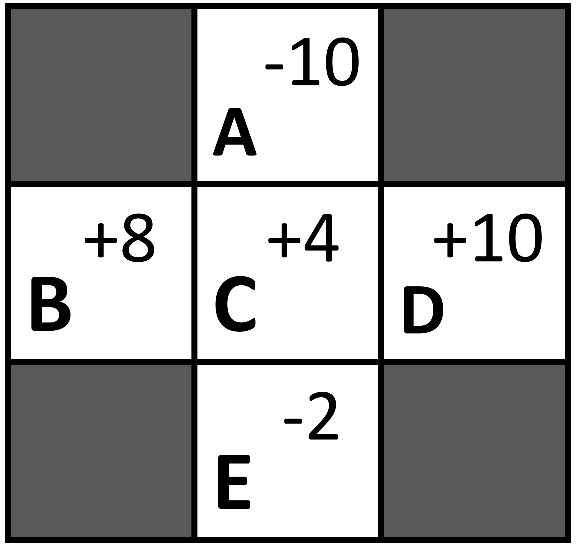
Output Values



Problems with Direct Evaluation

- What's good about direct evaluation?
  - It's easy to understand
  - It doesn't require any knowledge of T, R
  - It eventually computes the correct average values, using just sample transitions
- What bad about it?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn
- **How to make this work?**

Output Values

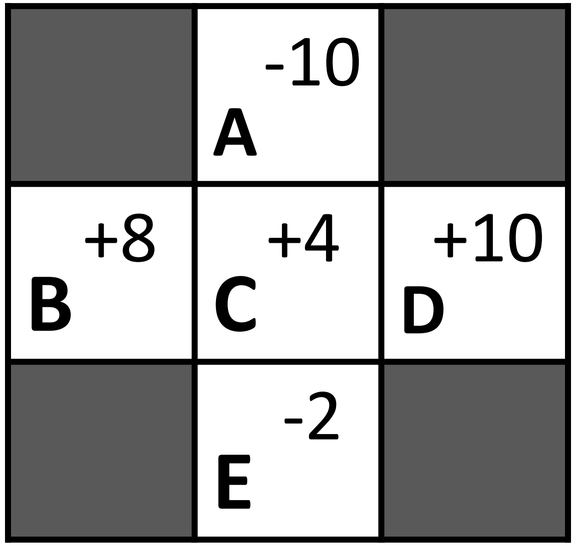


If B and E both go to C under this policy, how can their values be different?

Problems with Direct Evaluation

- What's good about direct evaluation?
  - It's easy to understand
  - It doesn't require any knowledge of T, R
  - It eventually computes the correct average values, using just sample transitions
- What bad about it?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn
- **How to make this work?**
  - **Collect huge large data.**

Output Values



If B and E both go to C under this policy, how can their values be different?

Why Not Use Policy Evaluation?

- Can we use policy evaluation to find values?

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

Why Not Use Policy Evaluation?

- Can we use policy evaluation to find values?
    - ✓ No, because we do not have T, R.
- Still can we find values?
    - Yes, but some modification is needed.

Sample-Based Policy Evaluation

- Idea: Take samples of outcomes s' (by doing the action!) and average

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

$$sample_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$sample_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

$$\ldots$$

$$sample_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$

Temporal difference learning

- We can learn from every experience.

- This formula can be re-written as:

$$V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n}\sum_i sample_i$$

Sample of V(s): $$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

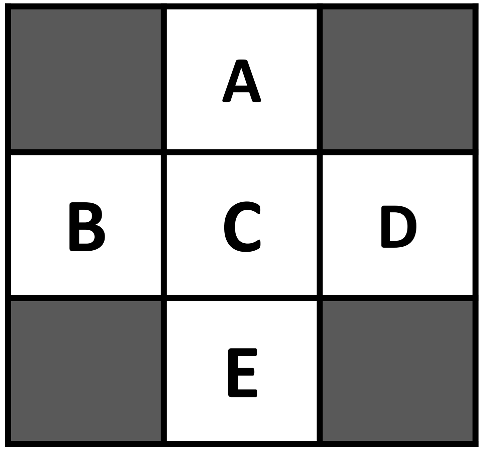
Update to V(s): $$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$$

Same update: $$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$$

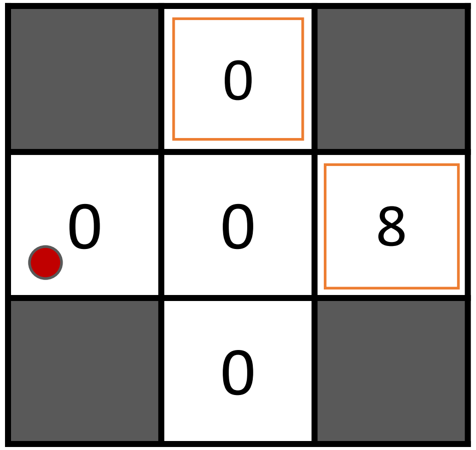
- Decreasing learning rate (alpha) can give converging averages

Example: Temporal Difference Learning

States



B, east, C, -2

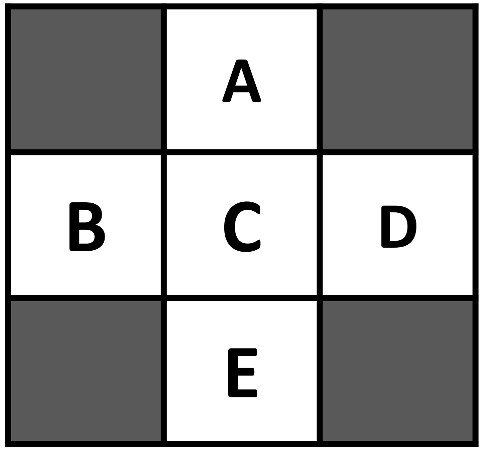


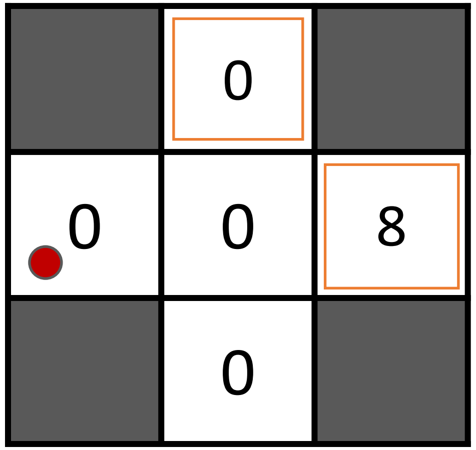
Assume: $\gamma = 1$, $\alpha = 1/2$

- Find the **sample value** of **red** dot, and update **v-value**.

Example: Temporal Difference Learning

States





B, east, C, -2

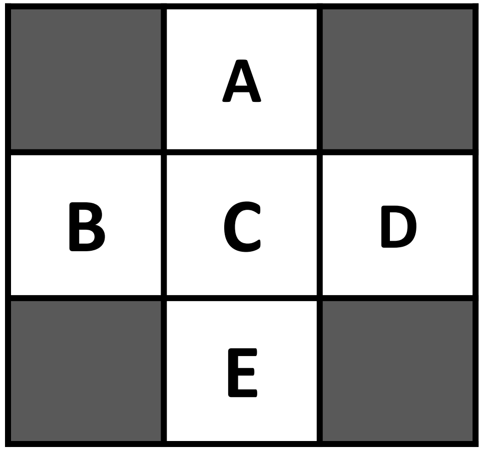
$$\text{Sample} = R + \gamma V$$
$$= -2 + 1 \cdot 0$$
$$= -2$$
$$V = (1-\alpha)V + \alpha(\text{sample})$$

Assume: $\gamma = 1$, $\alpha = 1/2$

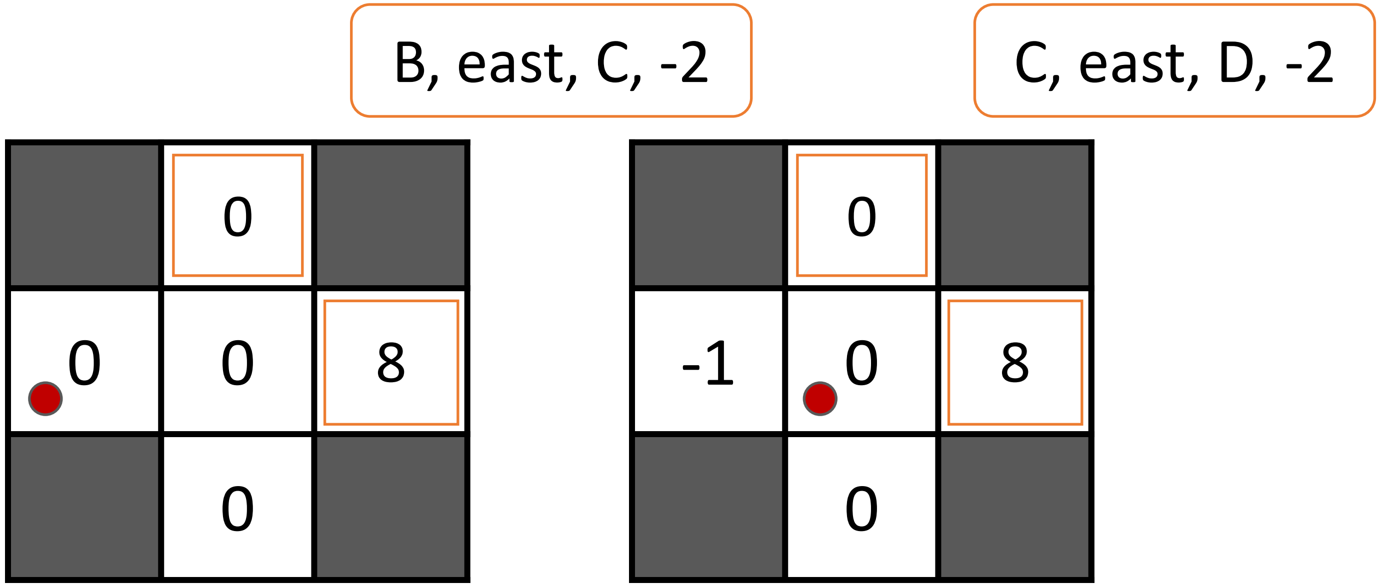
- Find the **sample value** of red dot, and update **v-value**.

Example: Temporal Difference Learning

States



Observed Transitions

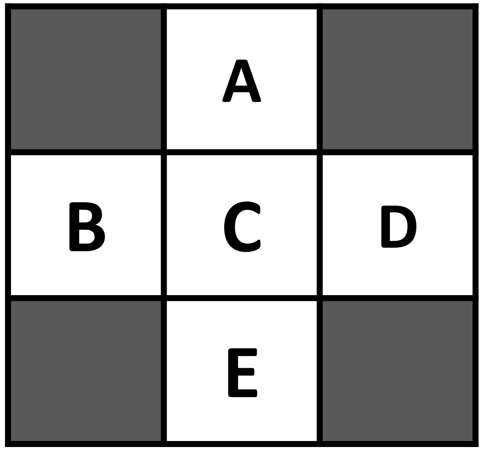


Assume: γ = 1, α = 1/2

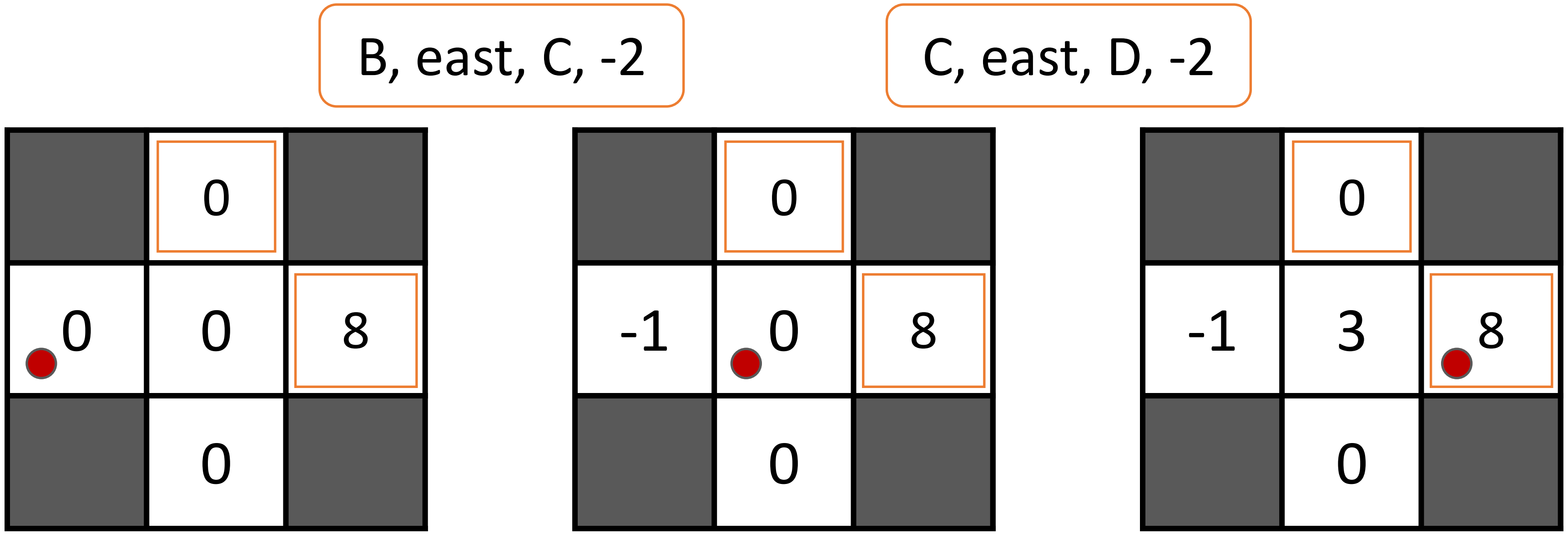
- Find the **sample value** of **red** dot, and update **v-value**.

Example: Temporal Difference Learning

States



Observed Transitions



Assume: $\gamma = 1$, $\alpha = 1/2$

Problems with TD Value Learning

- We can only evaluate the policy, but can not update it.

Problems with TD Value Learning

- We can only evaluate the policy, but can not update it.
- However, if we want to turn values into a (new) policy, we're sunk:

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

$$Q(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V(s')\right]$$

- Solution: learn Q-values

Active Reinforcement Learning

- Full reinforcement learning: optimal policies (like value iteration)
  - ✓ You don't know the transitions T(s,a,s')
  - ✓ You don't know the rewards R(s,a,s')
  - ✓ You choose the actions now
  - ✓ Goal: learn the optimal policy / values
- In this case:
  - ✓ Learner makes choices!
  - ✓ Fundamental tradeoff: exploration vs. exploitation (will see this in some time)
  - ✓ This is NOT offline planning! You actually take actions in the world and find out what happens

Q-Learning: recap

- Value iteration: find successive (depth-limited) values
  - Start with $V_0(s) = 0$, which we know is right
  - Given V_k, calculate the depth k+1 values for all states:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

- But Q-values are more useful, so compute them instead
  - Start with $Q_0(s,a) = 0$, which we know is right
  - Given Q_k, calculate the depth k+1 q-values for all q-states:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right]$$

Q-Learning

- Q-Learning: sample-based Q-value iteration

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

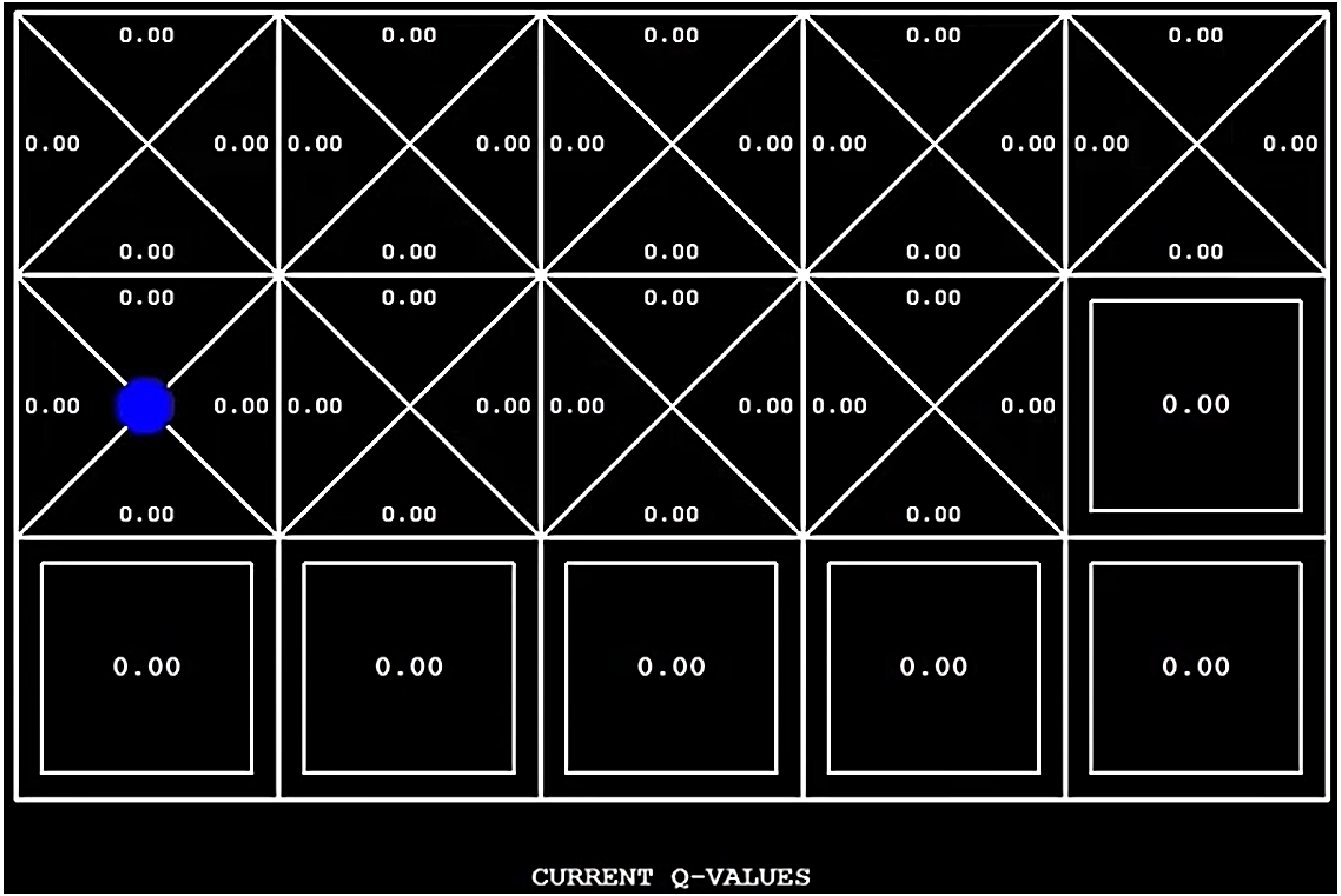
- Learn Q(s,a) values as you go
  - Receive a sample (s,a,s',r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[sample\right]$$

Q-Learning: example



+ reward = 10

- reward = -100

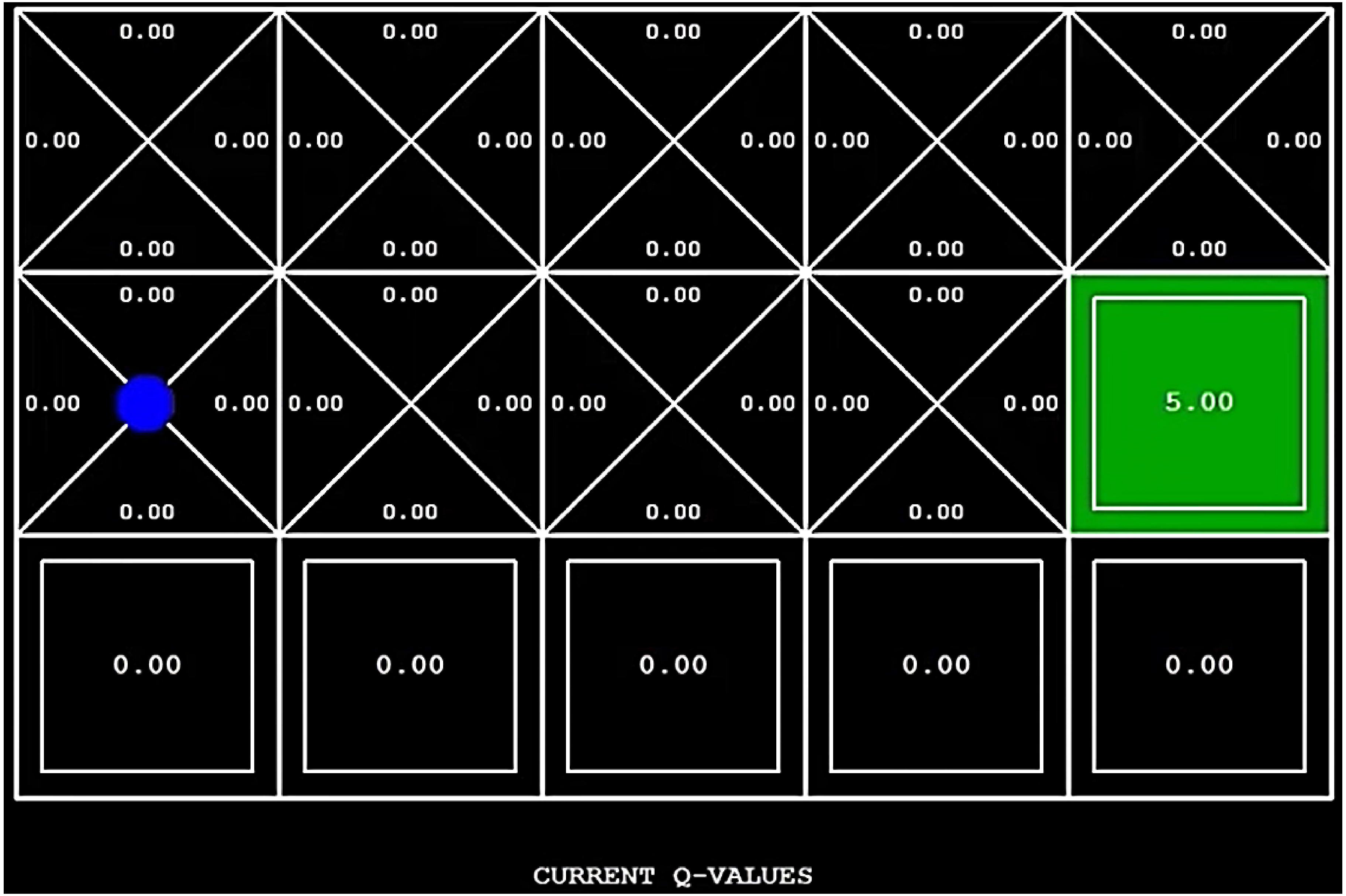
Alpha = 0.5

Discounting = 1

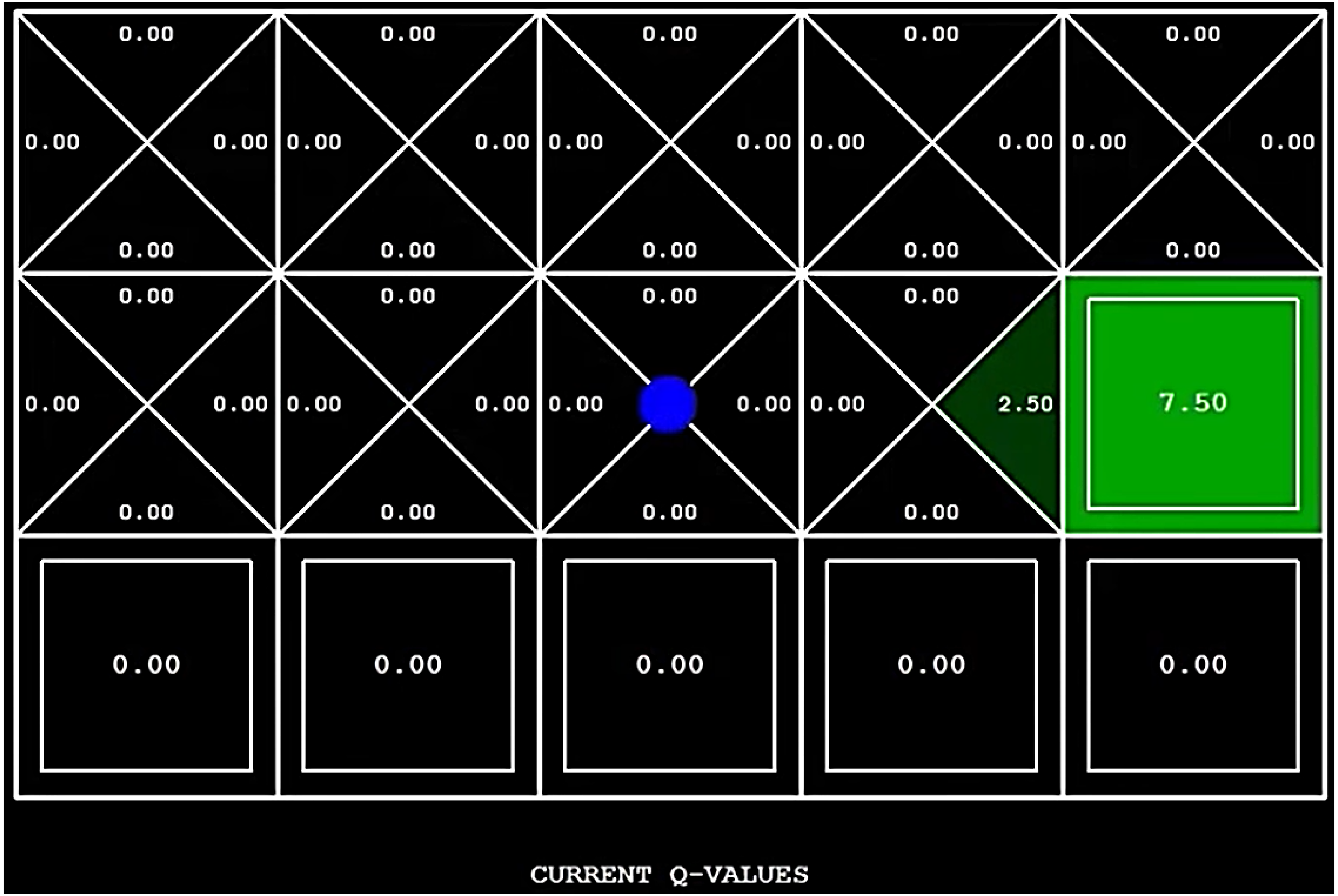
Sample = ?

Q-value = ?

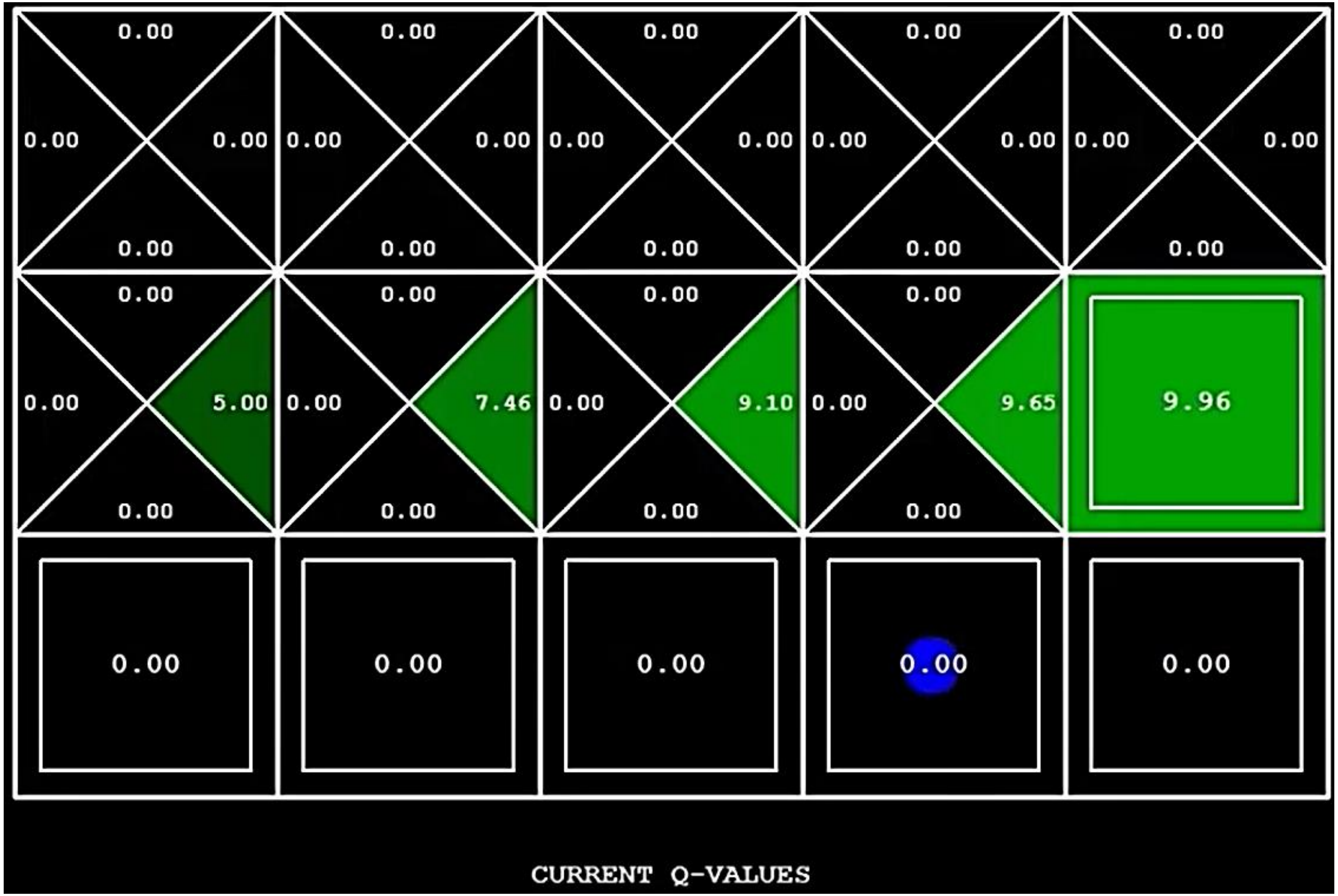
Q-Learning: example



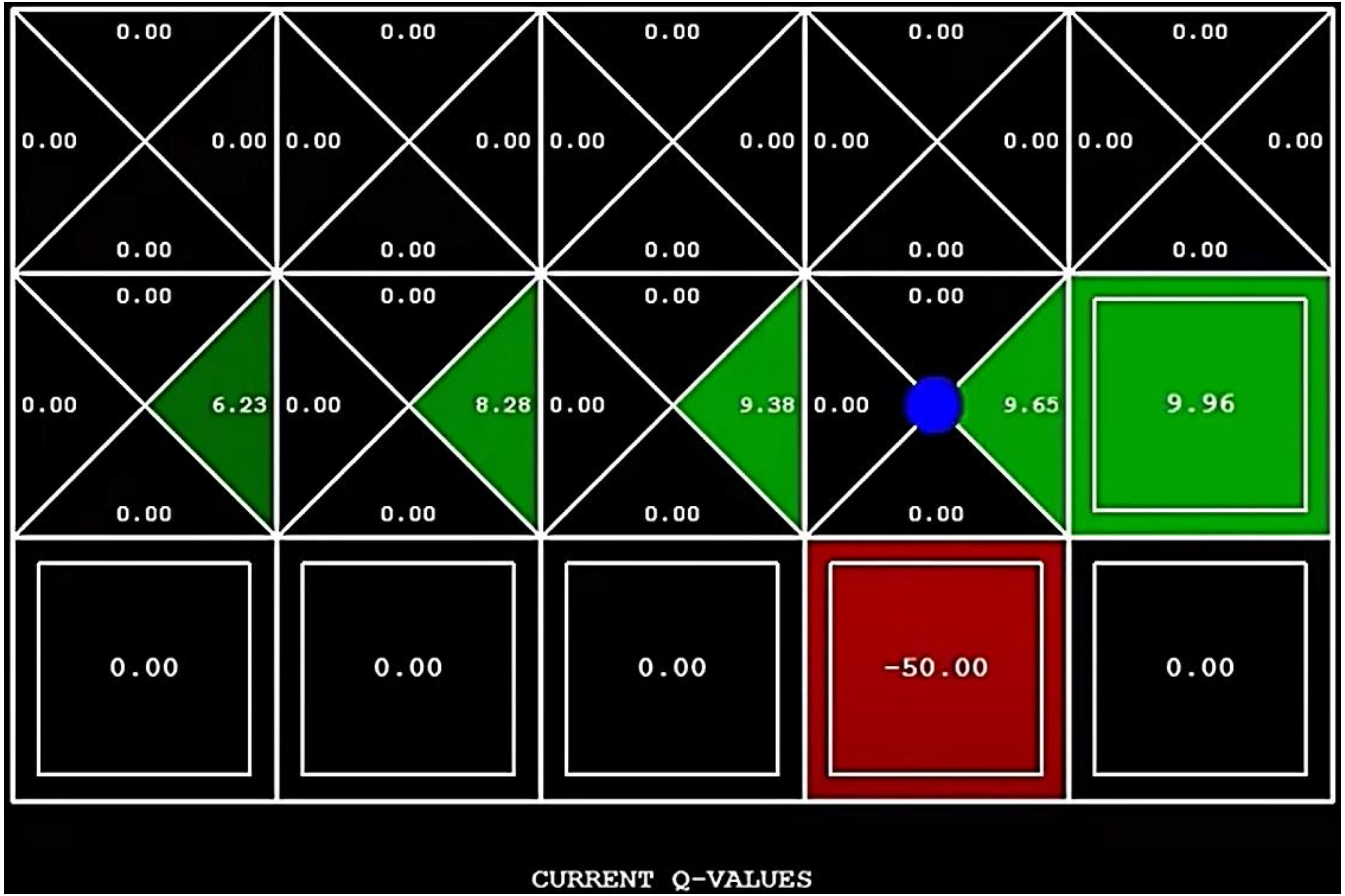
Q-Learning: example



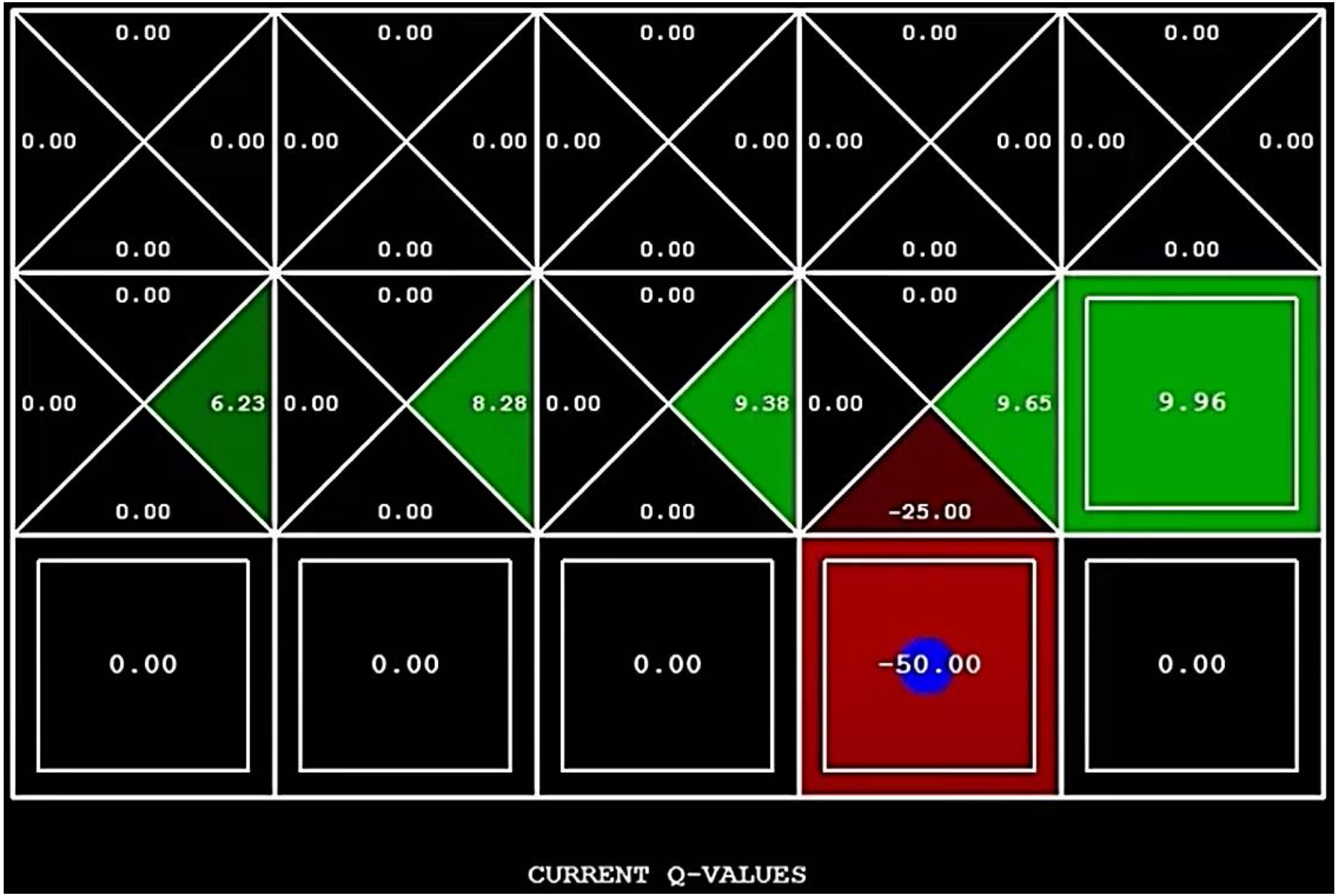
Q-Learning: example



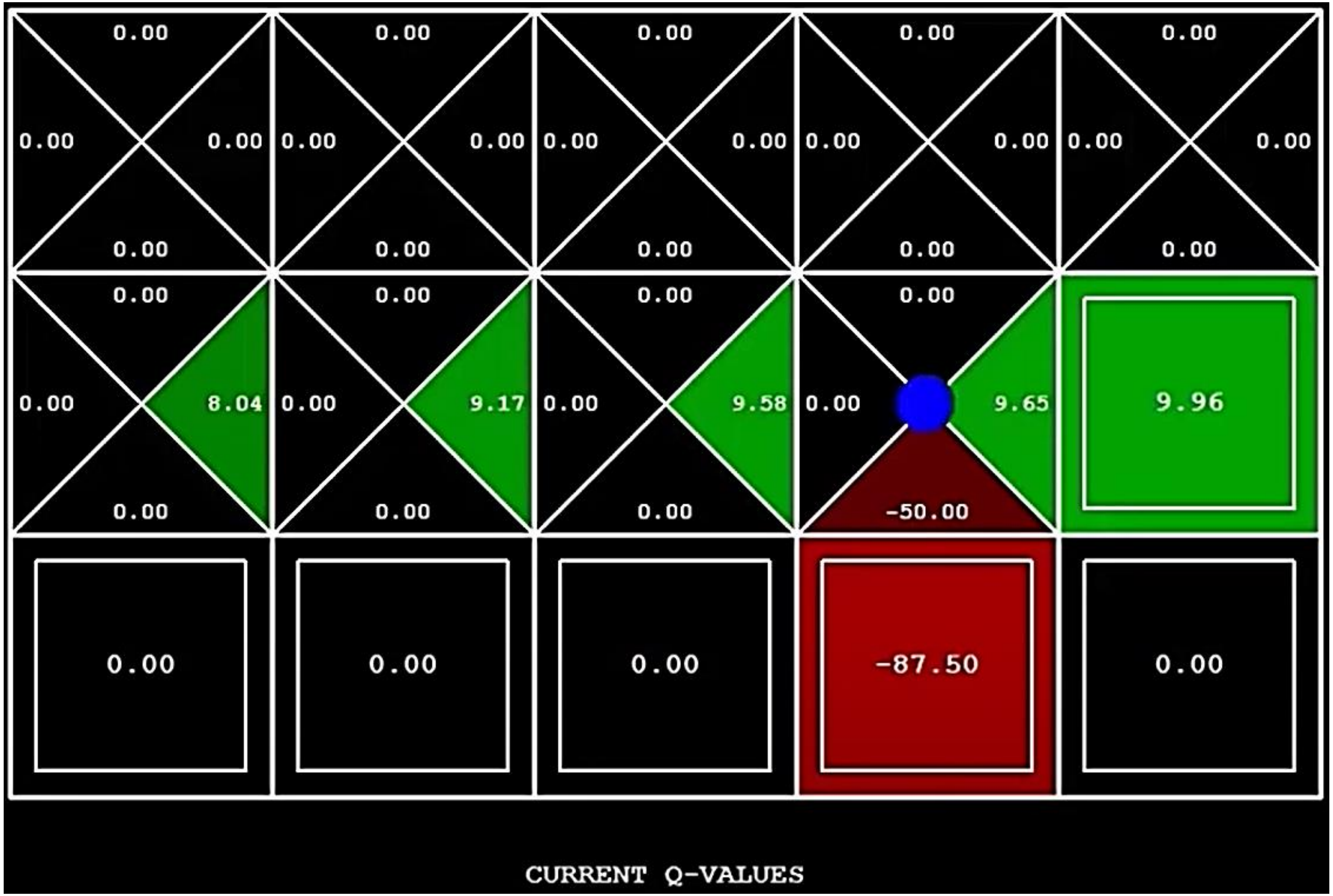
Q-Learning: example



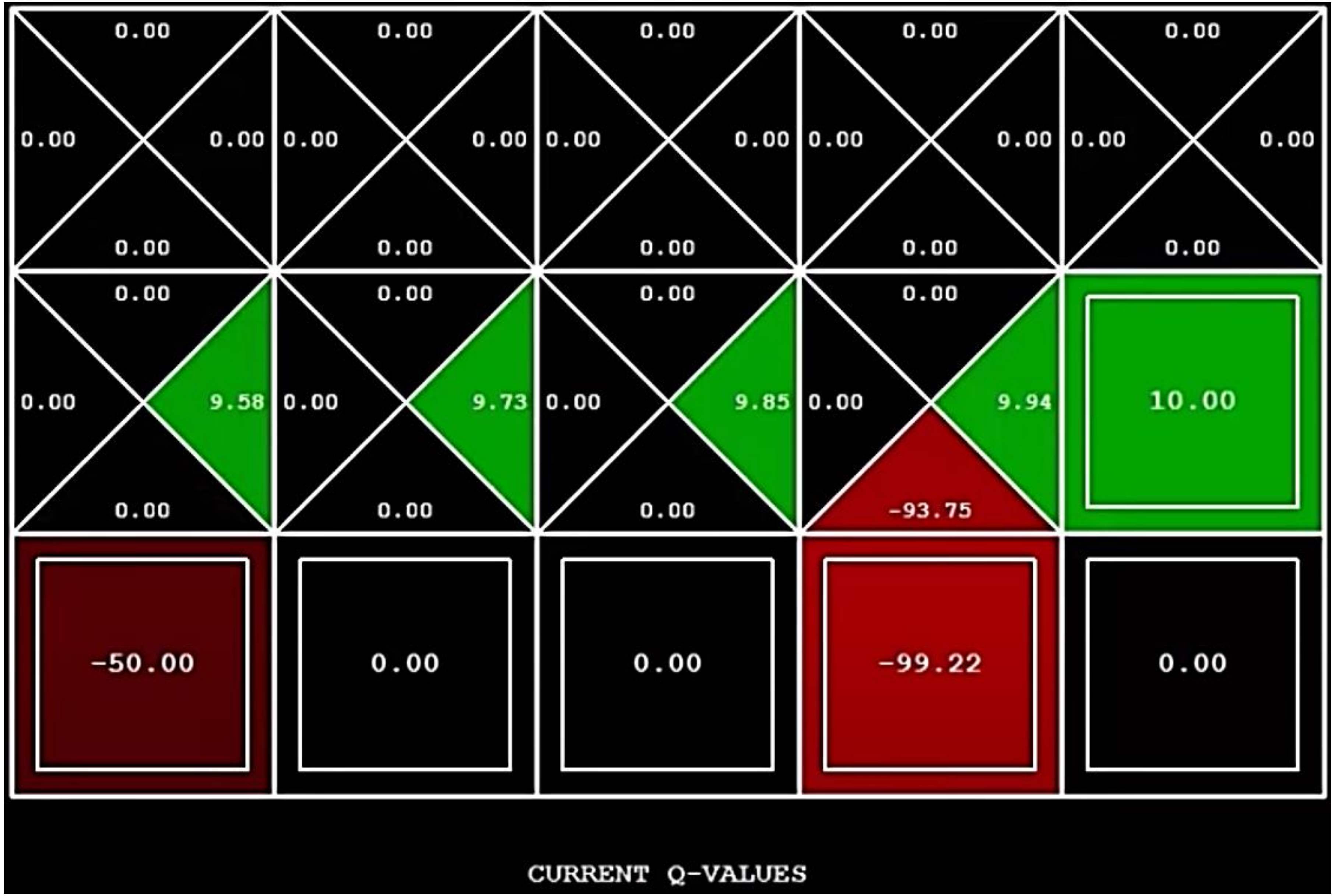
Q-Learning: example



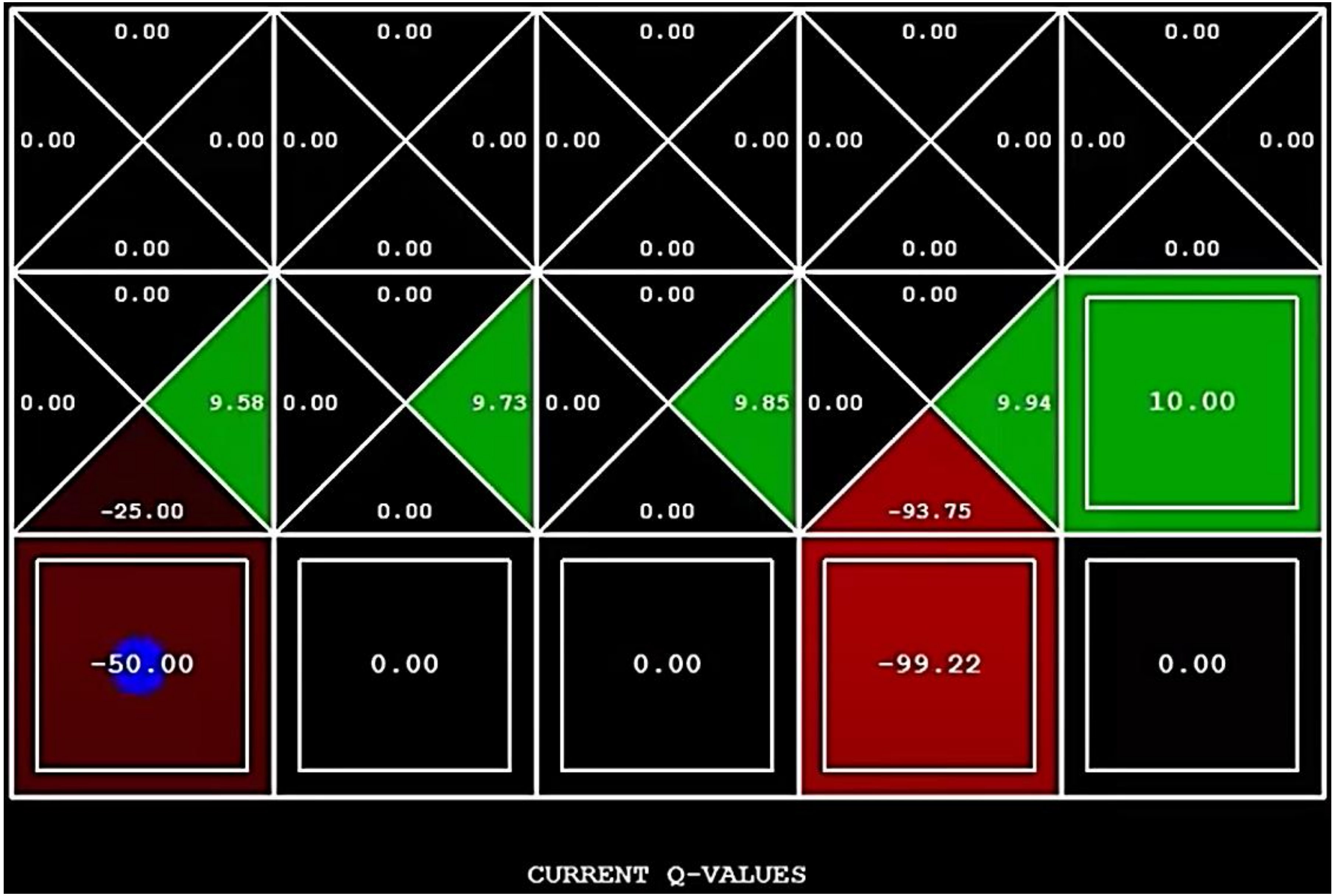
Q-Learning: example



Q-Learning: example



Q-Learning: example



Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you’re acting sub-optimally!
- This is called off-policy learning
- Limitations:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - … but not decrease it too quickly

Summary so far ...

Known MDP: Offline Solution

Goal	Technique
Compute V*, Q*, π*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

Unknown MDP: Model-Based

Goal	Technique
Compute V*, Q*, π*	VI/PI on approx. MDP
Evaluate a fixed policy π	PE on approx. MDP

Unknown MDP: Model-Free

Goal	Technique
Compute V*, Q*, π*	Q-learning
Evaluate a fixed policy π	Value Learning

Exploration vs. Exploitation

How to Explore?

- random actions (ϵ-greedy)
  - ✓ With (small) probability ϵ, act randomly
  - ✓ With (large) probability 1- ϵ, act on current policy
- Problems with random actions?
  - ✓ You do eventually explore the space, but keep thrashing around once learning is done
  - ✓ One solution: lower ϵ over time
  - ✓ Another solution: exploration functions

Exploration Functions

- When to explore?
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring
- Exploration function
  - Takes a value estimate u and a visit count n, and returns an optimistic utility, e.g. $f(u,n) = u + k/n$

Regular Q-Update: $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} Q(s',a')$

Modified Q-Update: $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

  - Note: this propagates the “bonus” back to states that lead to unknown states as well!

Approximate Q-Learning

- Why ?

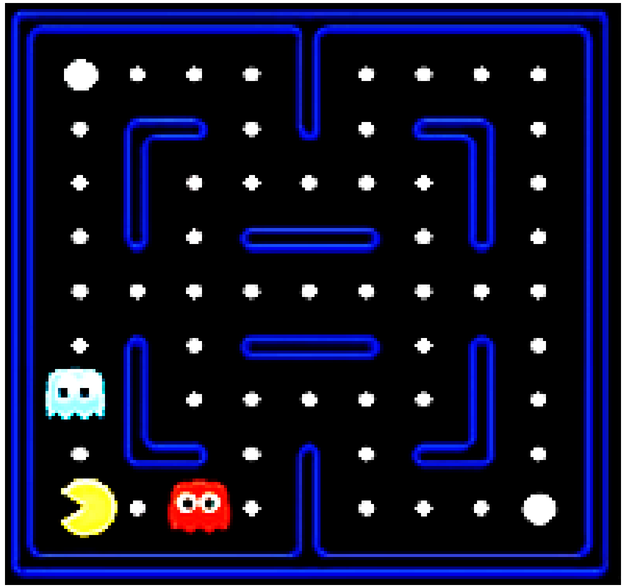
Generalizing Across States

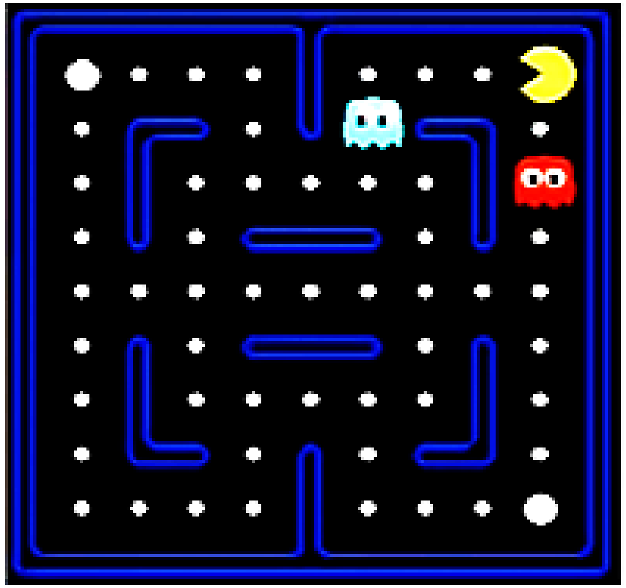
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
  - ✓ Too many states to visit them all in training
  - ✓ Too many states to hold the q-tables in memory
- Instead, we want to **generalize**:
  - ✓ Learn about some small number of training states from experience
  - ✓ Generalize that experience to new, similar situations
  - ✓ This is a fundamental idea in machine learning, and we'll see it over and over again

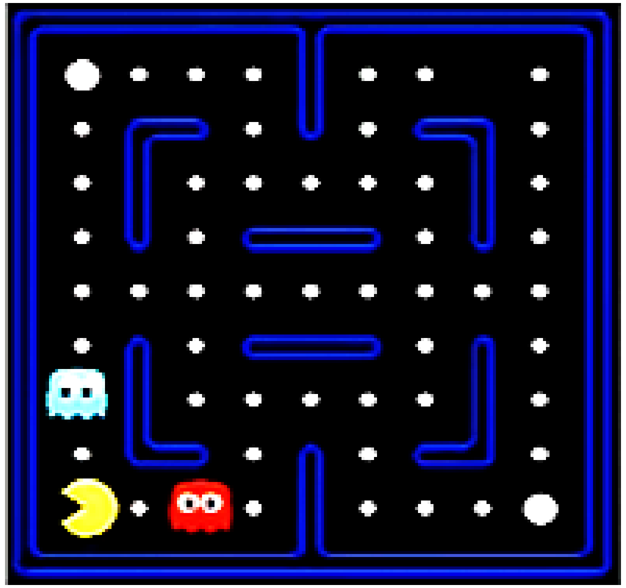
Generalizing Across States: example

- Lets say a person is having a glass of hot water just after the ice-cream or vice versa at Patiala.
- Is this experience bad?
- Lets say he repeats the same process in Delhi, Mumbai, etc.
- Is his experience change?
- What can be done?

Generalizing Across States: example







Feature-Based Representations

Solution: describe a state using a vector of features (properties)

- Features are functions from states to real numbers (often 0/1) that capture important properties of the state
- Example features:
  - ✓ Distance to closest ghost
  - ✓ Distance to closest dot
  - ✓ Number of ghosts
  - ✓ 1 / (dist to dot)^2
  - ✓ Is Pacman in a tunnel? (0/1)
  - ✓ …… etc.
- Can also describe a q-state (s, a) with features (e.g. action moves closer to food)

Approximate Q-Learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Q-learning with linear Q-functions:

$$\text{transition } = (s,a,r,s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s',a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}] \qquad \text{Exact Q's}$$

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a) \qquad \text{Approximate Q's}$$

- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on: dis-prefer all states with that state's features
- Formal justification: online least squares

Example: Q-Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

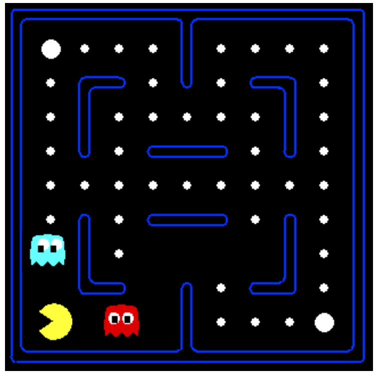
$f_{DOT}(s,a)$ Distance to closest DOT

$f_{GST}(s,a)$ Distance to closest ghost

Example: Q-Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s

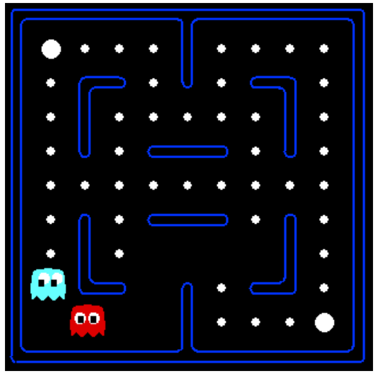


$f_{DOT}(s, \mathsf{NORTH}) = 0.5$

$f_{GST}(s, \mathsf{NORTH}) = 1.0$

$a = \mathrm{NORTH}$
$r = -500$

s'



$Q(s, \mathsf{NORTH}) = \ ?$

$r + \gamma \max_{a'} Q(s', a') = \ ?$

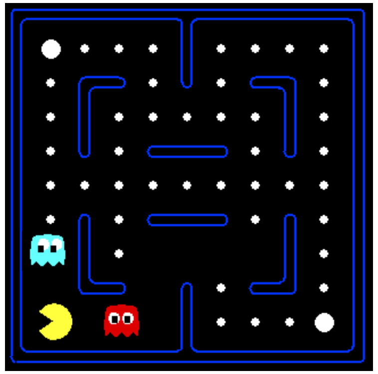
$Q(s', \cdot) = \ ?$

$\text{difference} = \ ?$

Example: Q-Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$

s



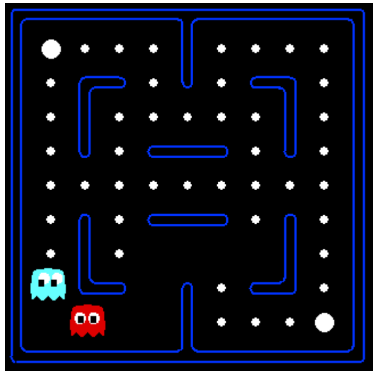
$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$a = \text{NORTH}$

$r = -500$

s'



$Q(s, \text{NORTH}) = +1$

$r + \gamma \max_{a'} Q(s', a') = -500 + 0$

$Q(s', \cdot) = 0$

difference $= -501$

$w_{DOT} \leftarrow 4.0 + \alpha \left[-501\right] 0.5$

$w_{GST} \leftarrow -1.0 + \alpha \left[-501\right] 1.0$

$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$