THREE TEMPORAL DIFFERENCE LEARNING ALGORITHMS

Scott O'Hara
Metrowest Developers Machine Learning Group
7/8/2020

REFERENCES

Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

- https://www.coursera.org/learn/sample-based-learning-methods/

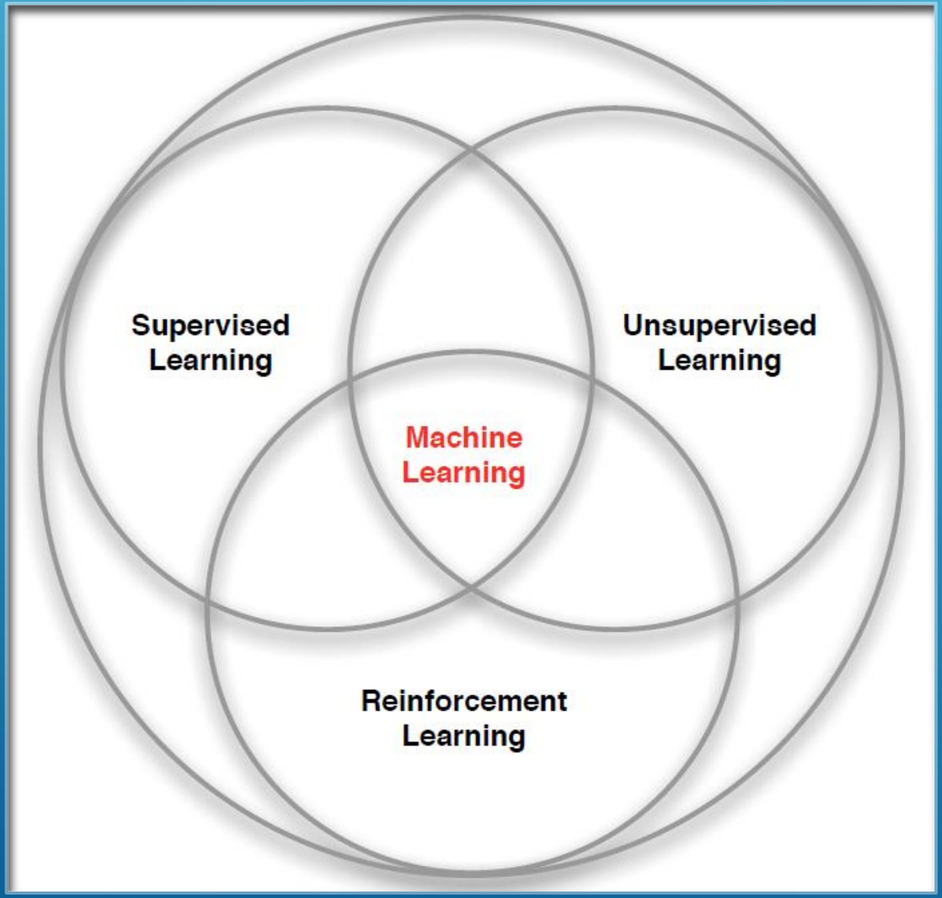
Reinforcement learning: An Introduction *R. S. Sutton and A. G. Barto, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.*

Reinforcement Learning *David Silver - University College London / Google DeepMind, 2015.*

- *https://www.davidsilver.uk/teaching/*

WHAT IS REINFORCEMENT LEARNING?

"Reinforcement learning is a kind **of *unsupervised supervised learning*"**
– *Rich Sutton*



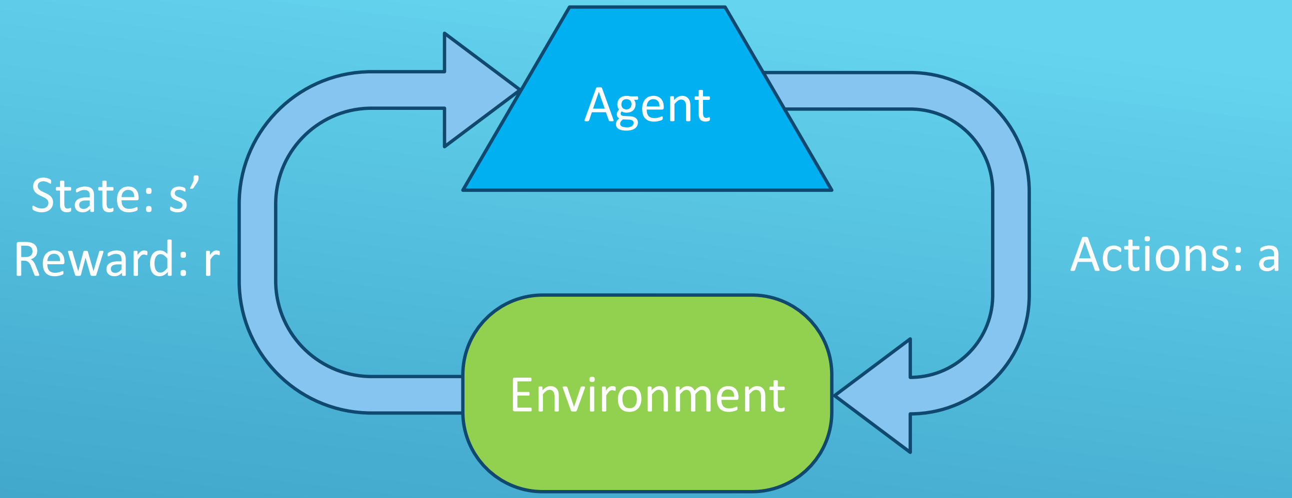
Supervised Learning – Learn a function from labeled data that maps input attributes to an output.

Unsupervised Learning – Find classes, patterns or generalizations in unlabeled data.

Reinforcement Learning –An agent learns to maximize rewards while acting in an uncertain environment.

Credit: image borrowed from lecture slides David Silver, DeepMind, "Introduction to Reinforcement Learning

THE REINFORCEMENT LEARNING PROBLEM



- Agent must learn to act to maximize expected rewards.
- Agent knows the current state s, takes an action ***a***, receives a reward ***r*** and observes the next state ***s’***.

$$S_0, A_0, R_0, S_1, A_1, R_2, S_2, A_2, R_2, \dots, S_n, A_n, R_n, S_T$$

- Agent has **no access** to the reward model ***r(s,a,s’)*** or the transition model ***T(s,a,s’)***.

MARKOV DECISION PROCESSES

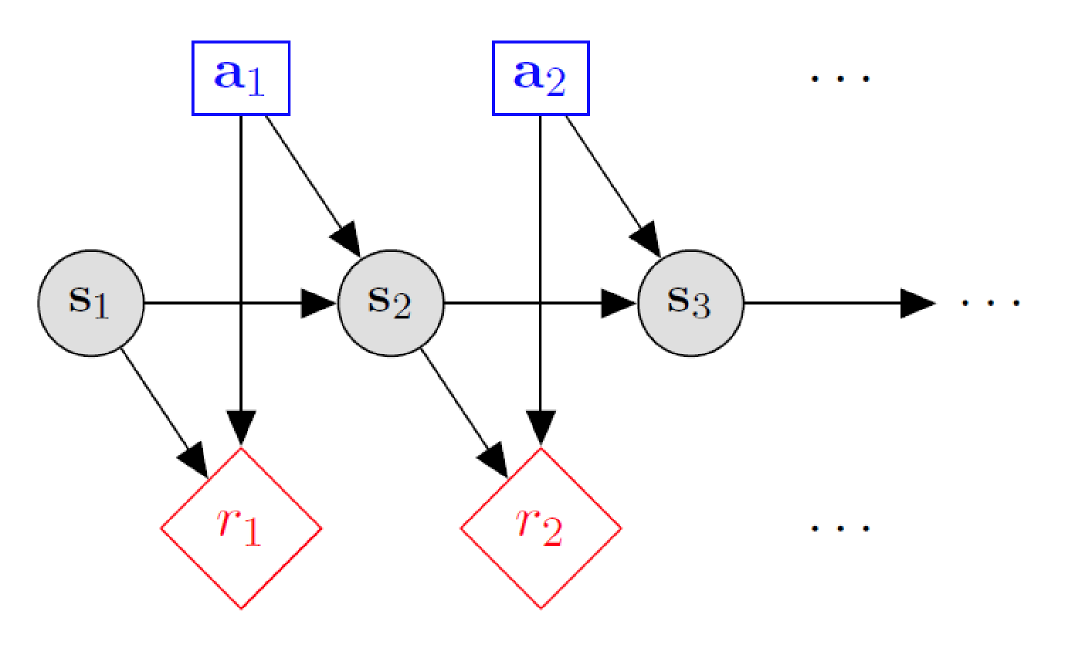
- **States:** $s_1, \dots, s_n$
- **Actions:** $a_1, \dots, a_m$
- **Reward <u>model</u>:**

  $$\mathrm{R}(s, a, s') \in R$$

- **Transition <u>model</u>:**

  $$T(s, a, s') = \mathrm{P}(s'|s, a)$$

- **Discount factor:** $\gamma \in [0, 1]$



<https://en.wikipedia.org/w/index.php?title=Markov_decision_process&oldid=855934986> [accessed 11 September 2018]

QUANTITIES TO OPTIMIZE

- The value (utility) of a **state s:**

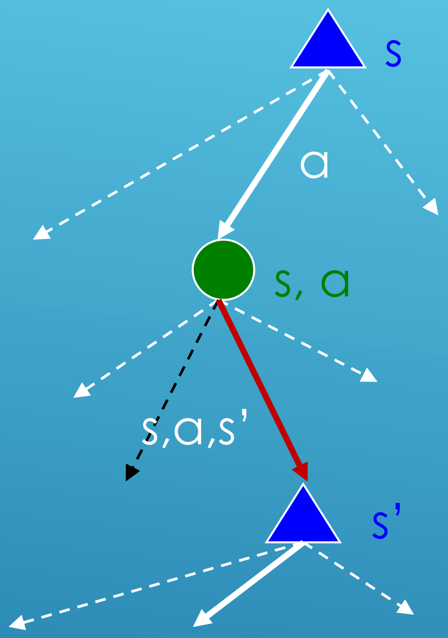
  V(s) = expected utility starting in s and acting optimally thereafter.

- The value (utility) of a **q-state (s,a):**

  Q(s,a) = expected utility when taking action a from state s and acting optimally thereafter.

- The policy π:

  π(a|s) = probability of action a from state s

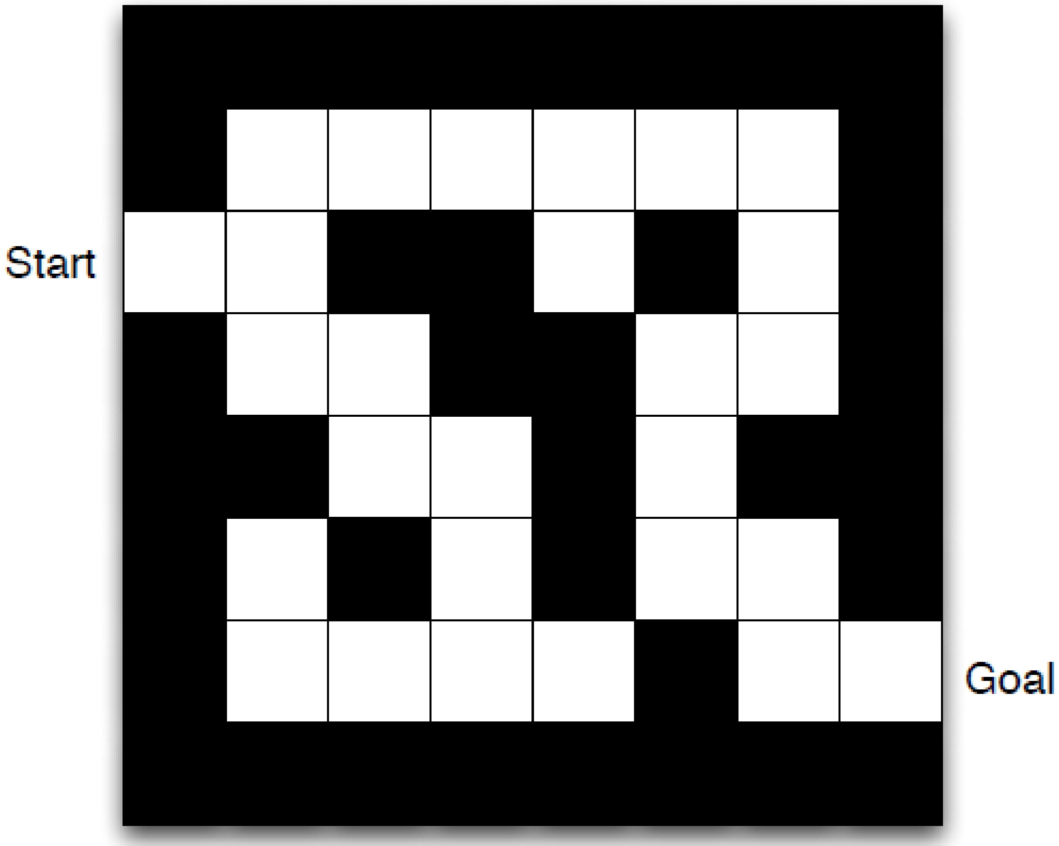


s is a *state*

(s, a) is a *q-state*

(s,a,s') is a *transition*

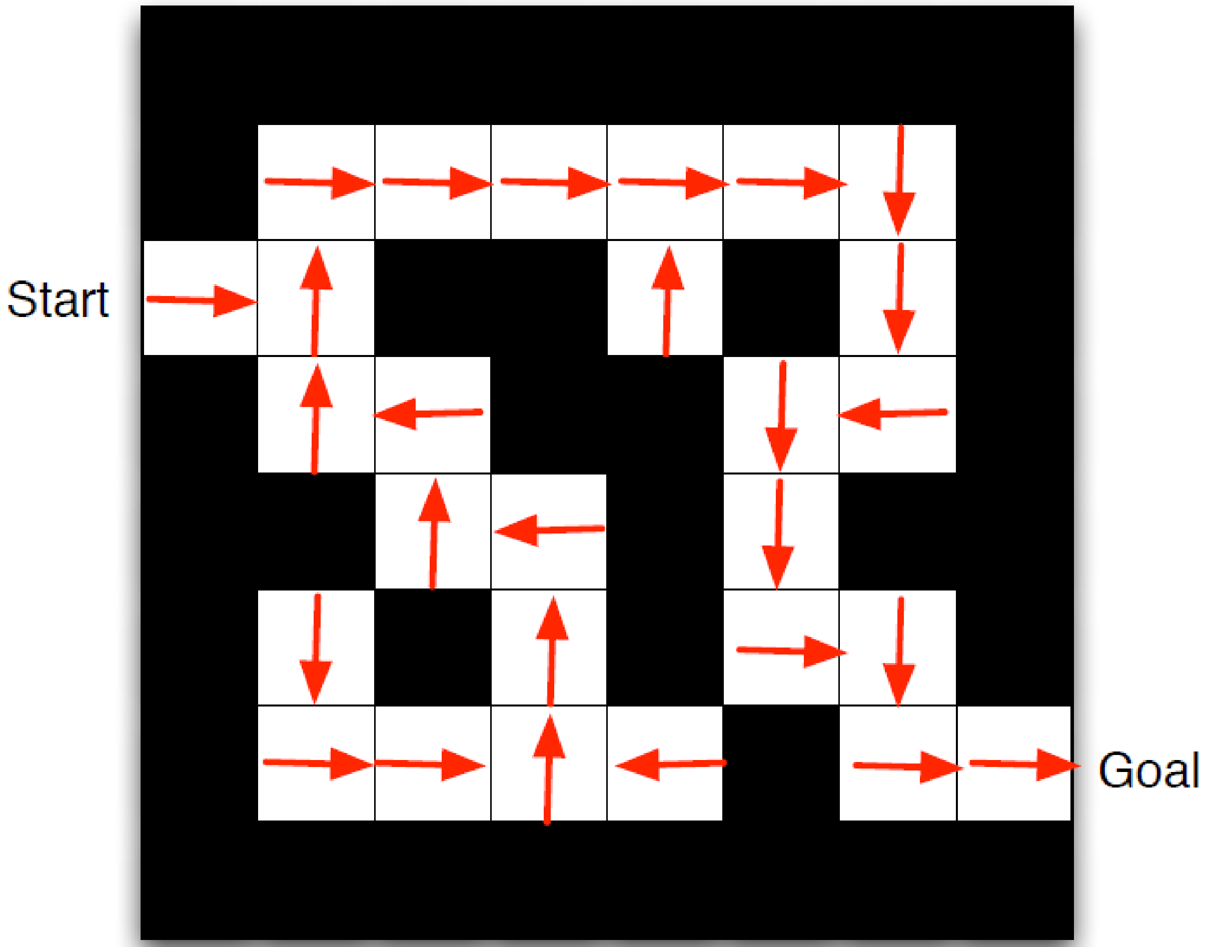
Maze Example



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Credit: Reinforcement Learning, *David Silver, University College London / Google DeepMind, 2015.*

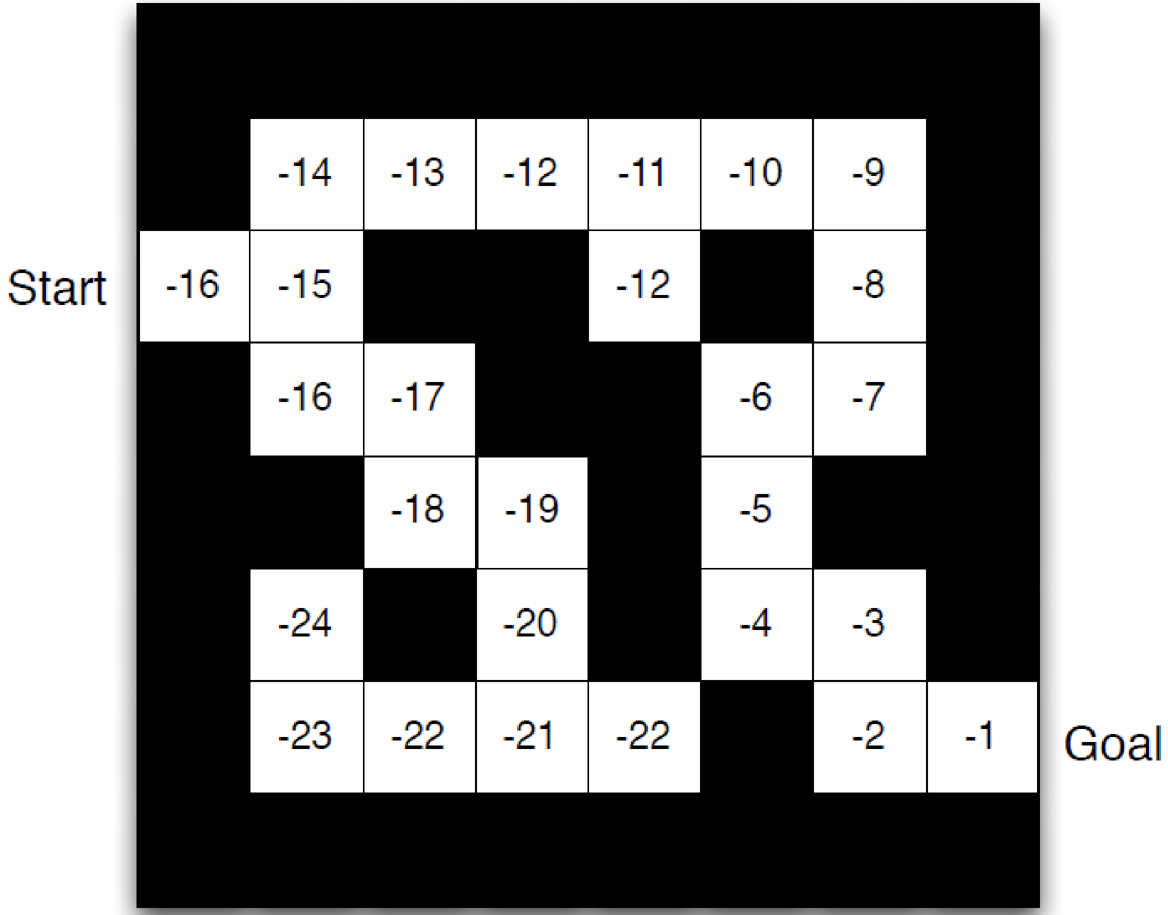
Maze Example: **Policy**



- Arrows represent policy $\pi(s)$ for each state s

Credit: Reinforcement Learning, *David Silver, University College London / Google DeepMind, 2015.*

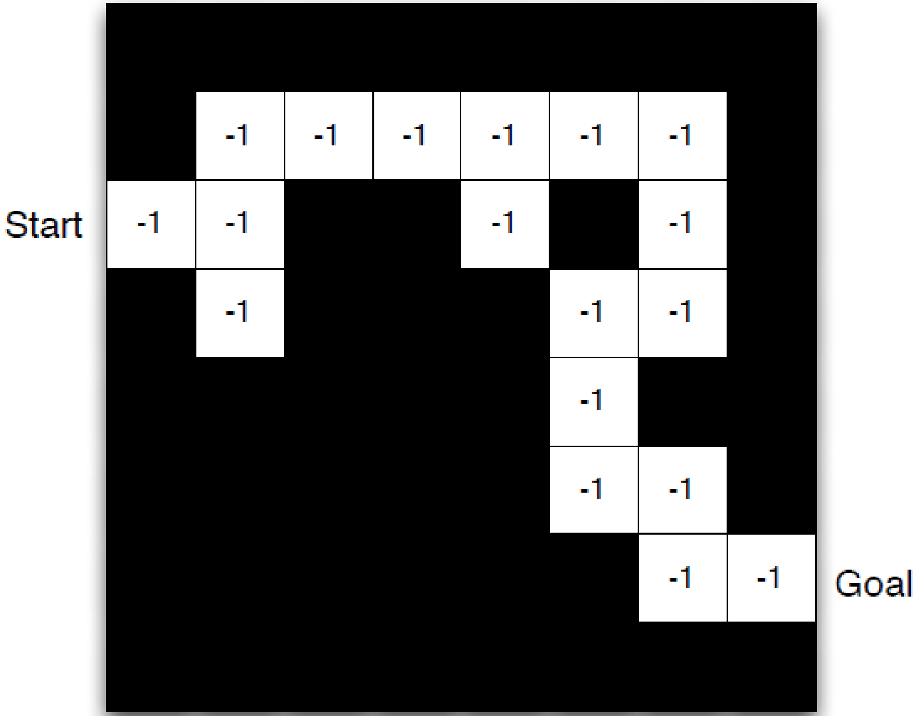
Maze Example: **Value Function**



- Numbers represent value $v_\pi(s)$ of each state s

Credit: Reinforcement Learning, *David Silver, University College London / Google DeepMind, 2015.*

Maze Example: **Model**



- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

- Grid layout represents transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (same for all a)

Credit: Reinforcement Learning, *David Silver, University College London / Google DeepMind, 2015.*

WHAT IS TEMPORAL DIFFERENCE (TD) LEARNING?

- TD-Learning is a kind of *prediction learning* that takes advantage of the temporal structure of learning to predict.

- In prediction learning:

  - You make a prediction about what will happen next.

  - You wait to see what happens.

  - You learn by comparing what happens to what you predicted.

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

WHAT IS TEMPORAL DIFFERENCE (TD) LEARNING?

- TD-Learning is one of the most fundamental ideas in reinforcement learning.

- From *Reinforcement Learning: An Introduction*: "If one had to identify one idea as central and novel to reinforce learning, it would be temporal difference learning." (page 119, Chapter 6)

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Updating from a Prediction

$$V(S_t) \leftarrow V(S_t) + \alpha \left[\underbrace{\boxed{R_{t+1} + \gamma V(S_{t+1})}}_{\text{The Future}} - \underbrace{V(S_t)}_{\text{The Present}} \right]$$

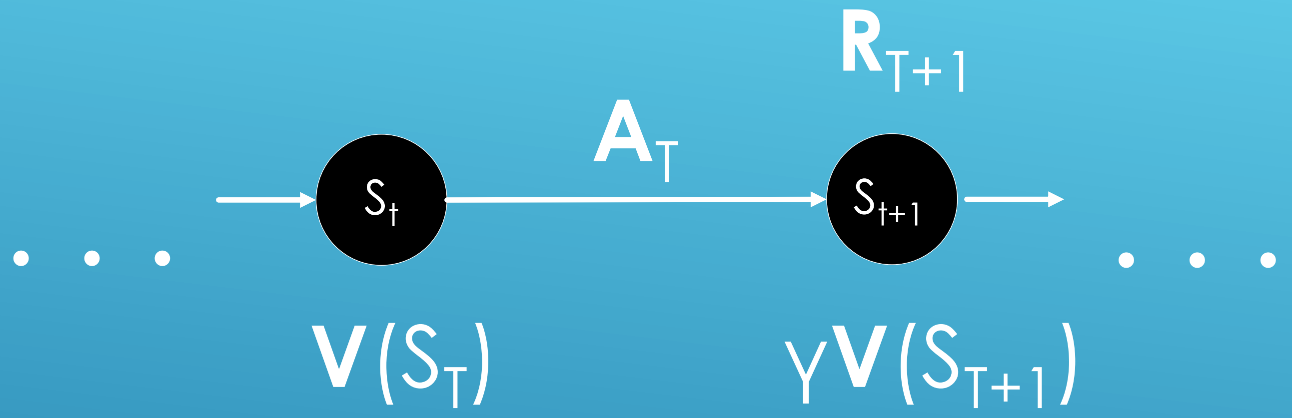
The Future
(The TD-target)

The Present
(The TD-prediction)

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots, S_t, A_t, R_{t+1}, S_{t+1}$$

Updating from a Prediction



$$V(S_t) \leftarrow V(S_t) + \alpha \left[\boxed{R_{t+1} + \gamma V(S_{t+1})} - V(S_t) \right]$$

TD Learning: Value Prediction

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

PREDICTION VS CONTROL

- Unfortunately, the TD Value Estimation algorithm will only allow you to **predict** the value you will get from being in a given state.

- BUT, unless you have a model of the environment, it does not allow you to determine a policy to **control** the agent to maximize the value in the environment.

Update Rule:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[\boxed{R_{t+1} + \gamma V(S_{t+1})} - V(S_t) \right]$$

Bellman Eq:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

PREDICTION VS CONTROL (2)

- Fortunately, if you learn the state-action value function **Q**, you can both **predict** the value of an action at a given state **AND** you can **control** the agent to maximize the value in the environment.

Update Rule:

$$\boldsymbol{Q_k(s,a) \leftarrow Q_k(s,a) + \alpha[r + \gamma Q_k(s',a') - Q_k(s,a)]}$$

Bellman Eq:

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

3 UPDATE RULES, 3 ALGORITHMS

Sarsa:

$$Q_k(s,a) \leftarrow Q_k(s,a) + \alpha[r + \gamma Q_k(s',a') - Q_k(s,a)]$$

Q-Learning:

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

Expected Sarsa:

$$Q_k(s,a) \leftarrow Q_k(s,a) + \alpha\left[r + \gamma \sum_{a'} \pi(a'|s') Q_k(s',a') - Q_k(s,a)\right]$$

CLASSIFYING RL ALGORITHMS

- **Prediction:**
  - Predict the value of a state or a state-action pair.
  - How does the algorithm compute the ***value functions*** **V** and **Q**?

- **Control:**
  - How does algorithm decide what to do next?
  - How is the agent's ***policy*** created and optimized?
  - Exploitation vs. exploration strategies.

- **Planning:**
  - Does the algorithm use a ***model*** of the environment?
  - How is the model created and updated?
  - How is the model exploited?

ON-POLICY / OFF-POLICY MODEL-BASED / MODEL-FREE

- **Control:**
  - **On-policy** – learn policy based on the one you are following.
  - **Off-policy** – learn policy different from the one you are following.
- **Planning:**
  - **Model-based** – use a model of the environment for prediction and control.
  - **Model-free** – learn value function or policy directly without a model i.e., the <u>transition function</u> and the <u>reward function</u>. Sarsa, Q-learning and Expected Sarsa are all model-free.

SARSA – On-policy TD Control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

Q-learning – Off-policy TD Control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

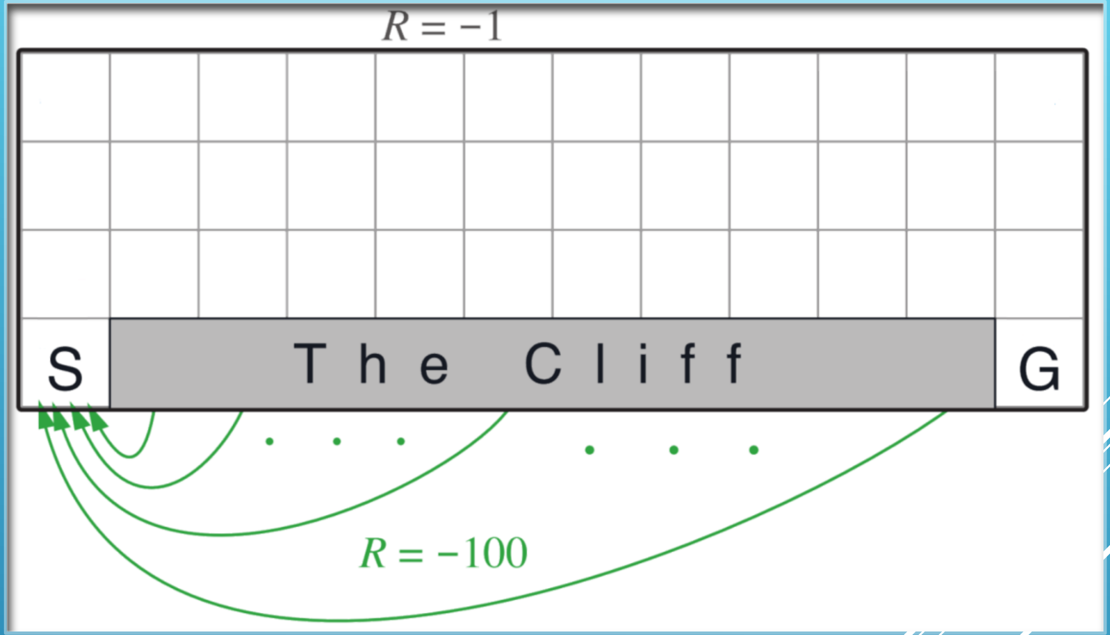
Expected SARSA – On-policy TD Control

Same as Q-Learning, but substitute expected state-action value for the max state-action value.

$$\boldsymbol{Q_k(s,a) \leftarrow Q_k(s,a) + \alpha\left[r + \gamma\sum_{a'}\pi(a'|s')Q_k(s',a') - Q_k(s,a)\right]}$$

RL ENVIRONMENT: CLIFFWORLD

- **States: <x, y>** locations
- **Actions:** move **north, south, east** or **west.**
- **Reward model:**
  - 0 if robot moves to the goal state G where episode finishes.
  - -100 if robot moves to the cliff.
  - -1 for every other move.
- **Transition model:**
  - Robot moves deterministically in the chosen direction: **north, south, east** or **west.**
  - Robot stays put if it moves into a wall.
  - Robot transitions to the start state if it moves onto the cliff. (NOTE: episode does not finish.)



- **Discount factor: γ = 1.0**

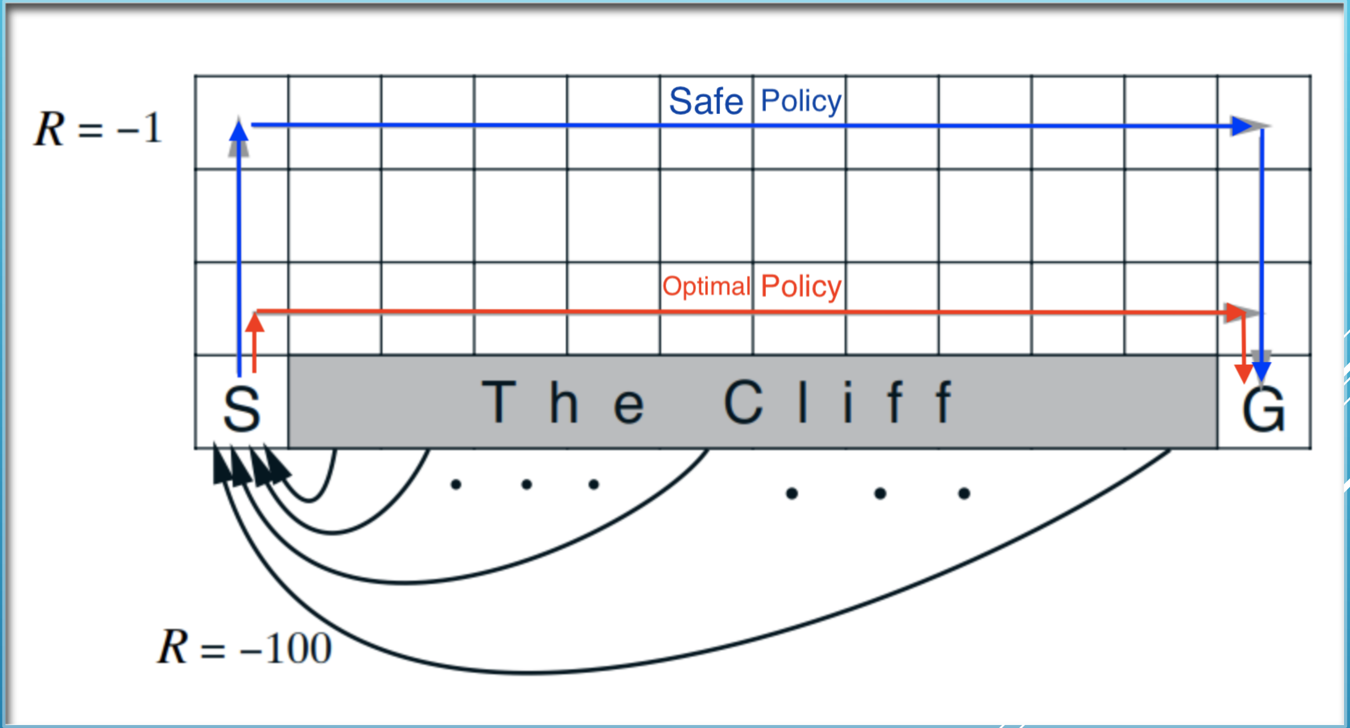
Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Q-LEARNING: OPTIMAL VS. "SAFE" POLICIES

- Q-learning will learn the optimal policy.

- However, q-learning must stop exploring and change to complete exploitation mode to take advantage of this.

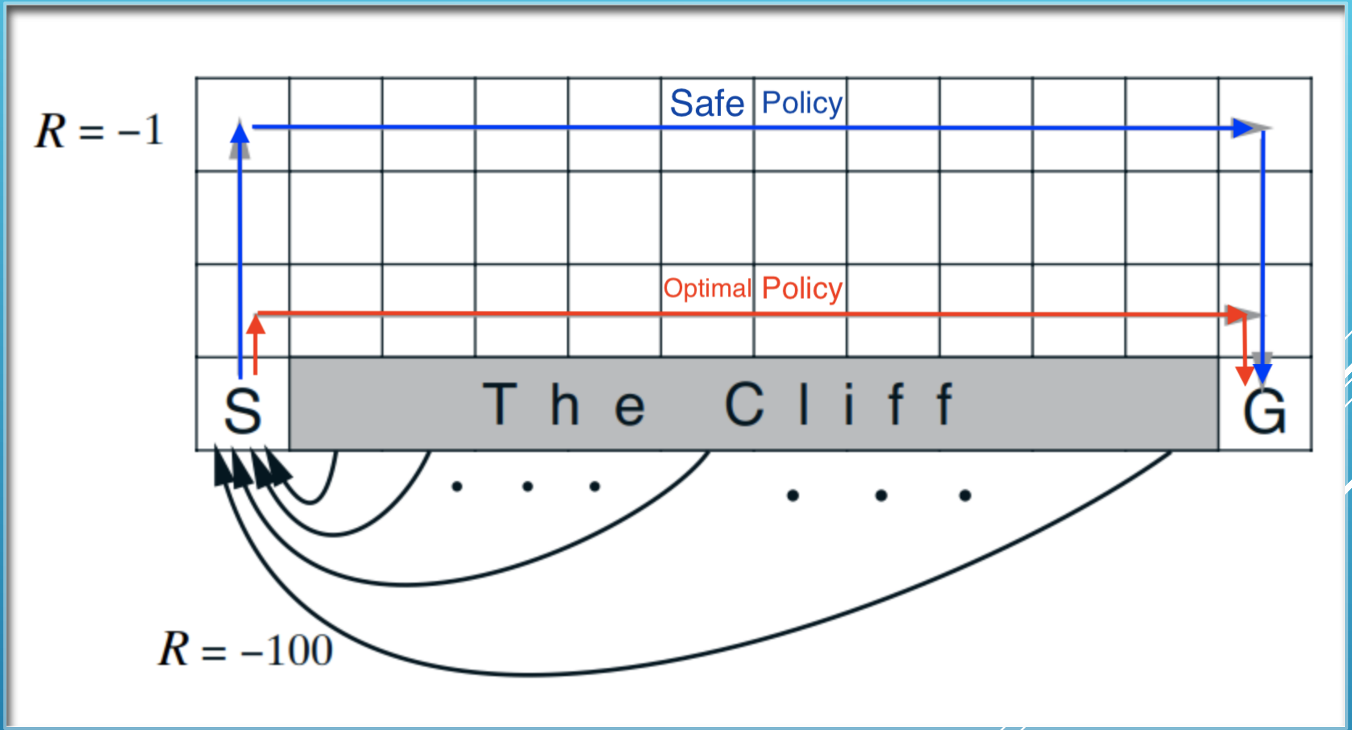
- If q-learning continues to explore (off-policy), it will often get bad results since exploration will lead it to step over the cliff.



Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

SARSA: OPTIMAL VS. "SAFE" POLICIES

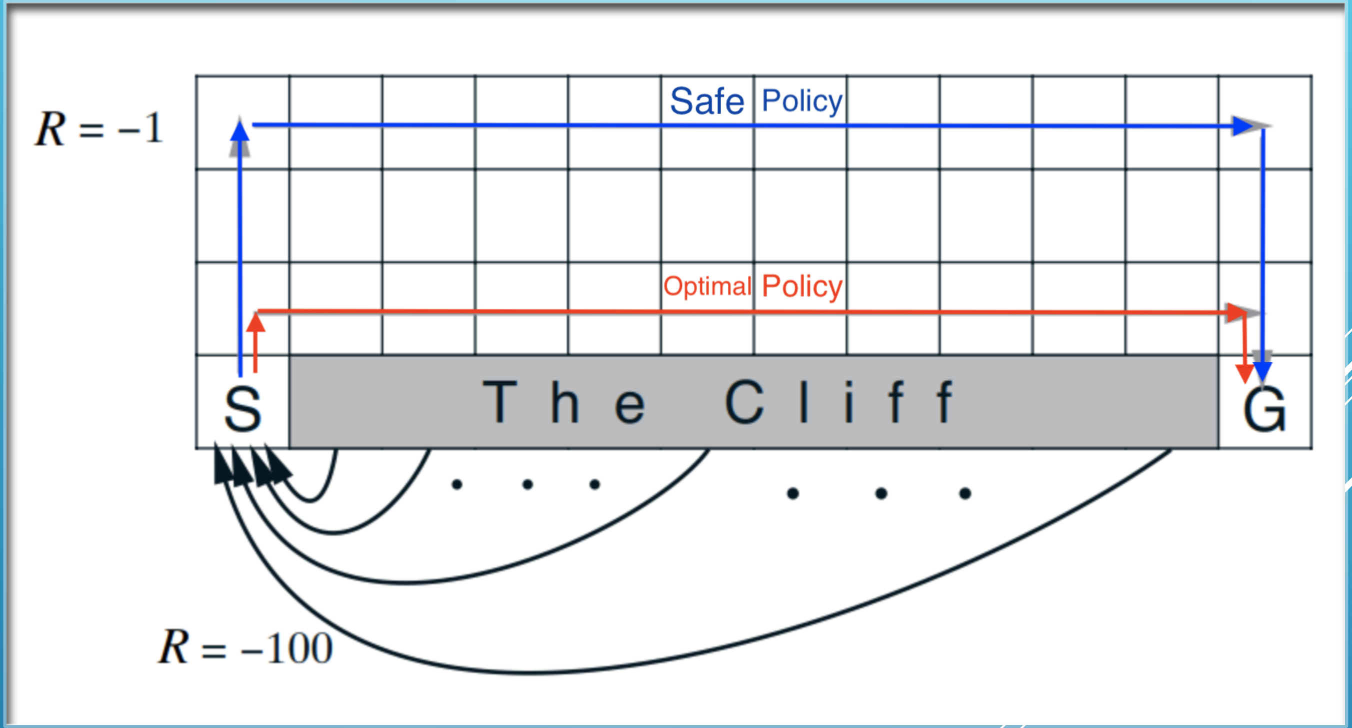
- SARSA will learn the safe policy since it learns the policy it actually does.
- However, SARSA learns slowly since it doesn't take full advantage of the knowledge it has of state-action values.



Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

EXPECTED SARSA: OPTIMAL VS. "SAFE" POLICIES

- Is it possible to learn an optimal policy that allows the agent to continue exploring?
- Yes! Expected SARSA, which is an on-policy algorithm can take into consideration the probability that an exploration action will take it over the cliff.
- Downside: It is more expensive to compute this policy.



Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

CHOOSING AN ACTION: EXPLORATION VS EXPLOITATION

- How should an agent choose an action? An obvious answer is simply to follow the current policy. However, this is often not the best way to improve your model.

- **Exploit:** use your current model to maximize the expected utility now.

- **Explore:** choose an action that will help you improve your model.

EXPECTED SARSA WITH E-GREEDY METHOD

- n – number of actions
- m – number of max actions
- With probability 1 - ϵ:
  select the action with the maximum value.

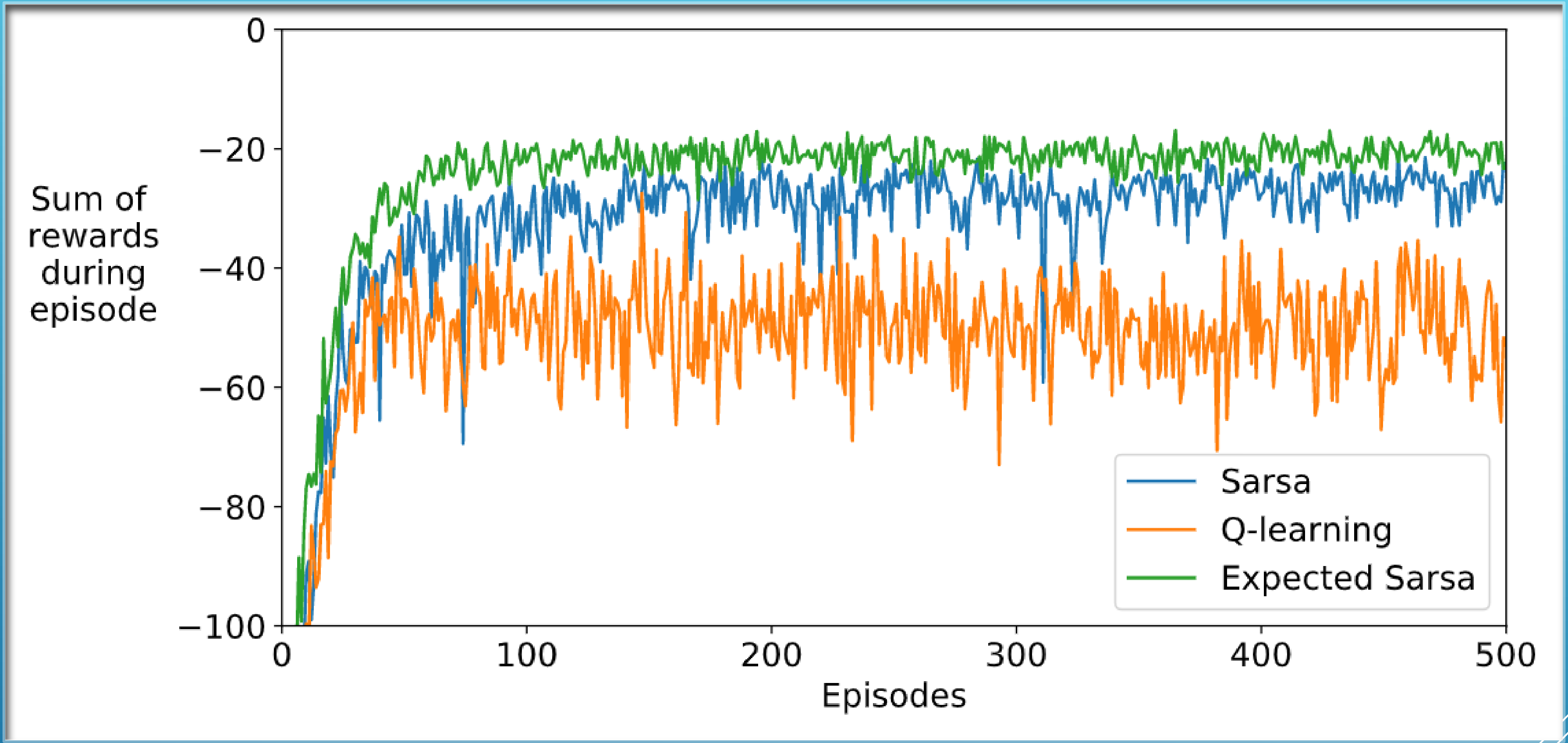
$$A_t = argmax\ Q_t(a)$$

$$P(a_{max}) = \frac{(1-\epsilon)}{m} + \frac{\epsilon}{n}$$

- With probability ϵ:
  randomly select an action from all the actions with equal probability.

$$P(a_{other}) = \frac{\epsilon}{n}$$

SARSA VS. Q-LEARNING VS. EXPECTED SARSA

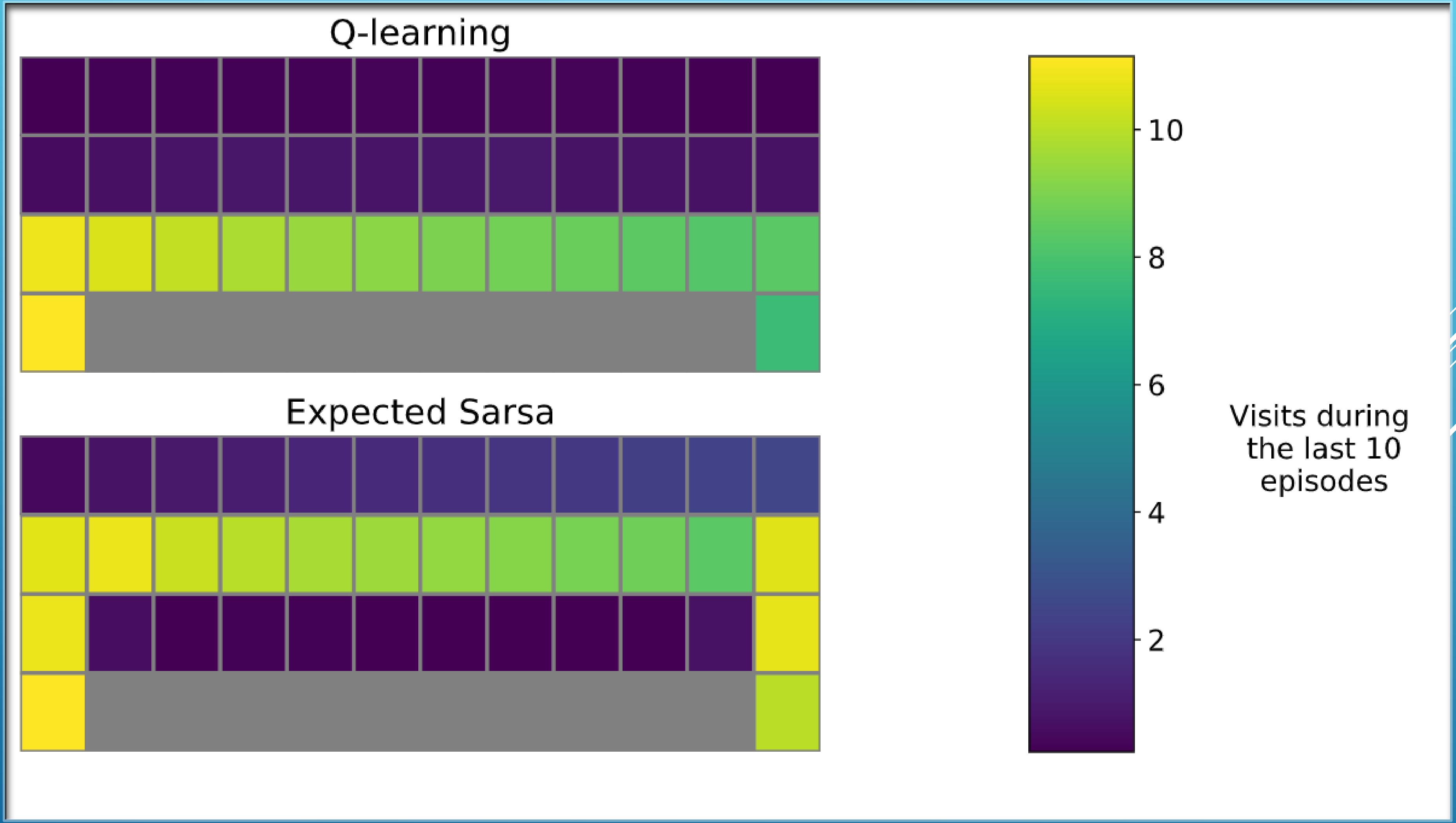


- 100 runs / 500 episodes per run.
- Average the sum of rewards for each episode over 100 runs.

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Q-LEARNING VS. EXPECTED SARSA

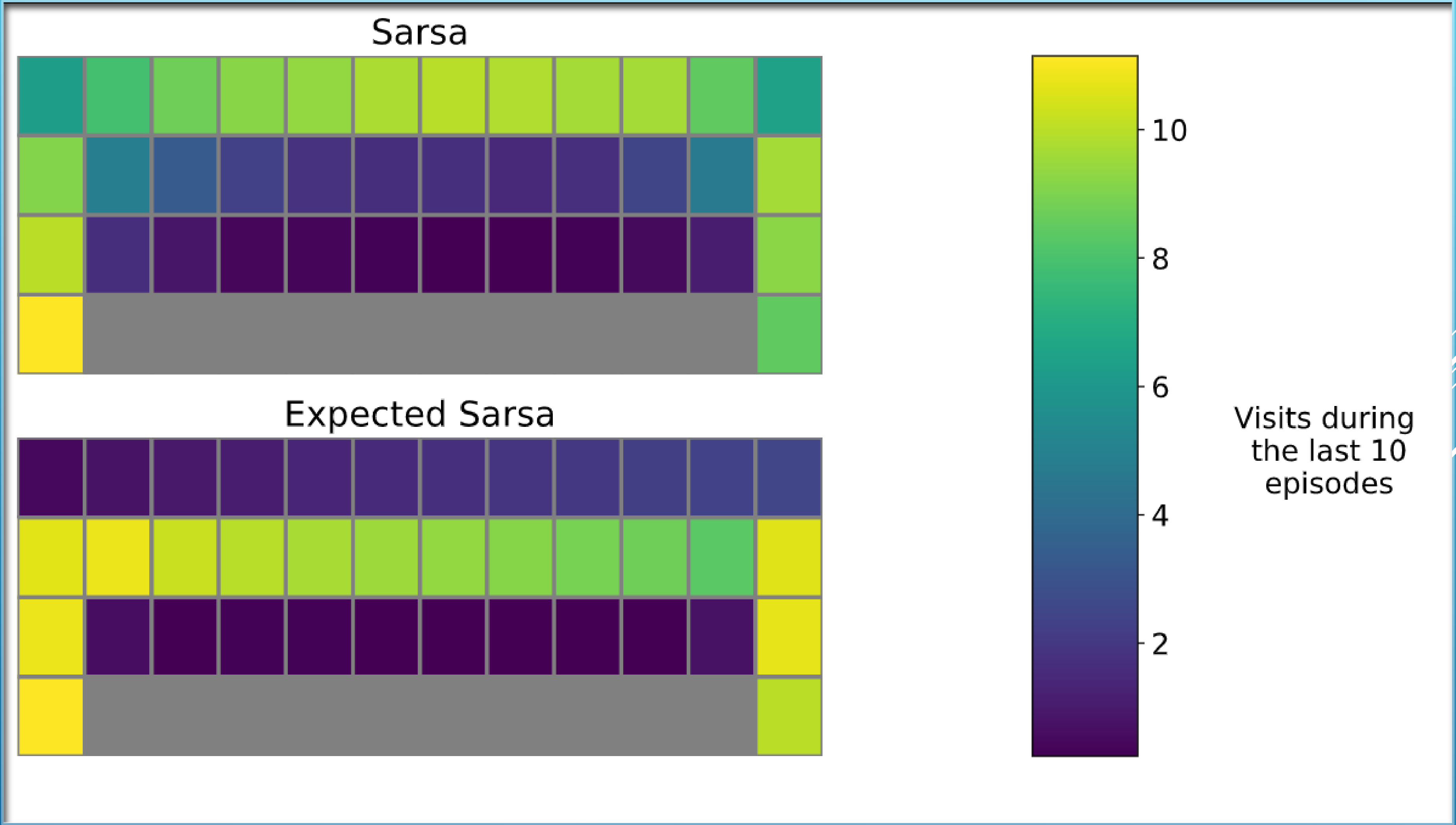
- 100 runs / 500 episodes per run.
- Average the number of visits to a state during the last 10 episodes over 100 runs.



Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

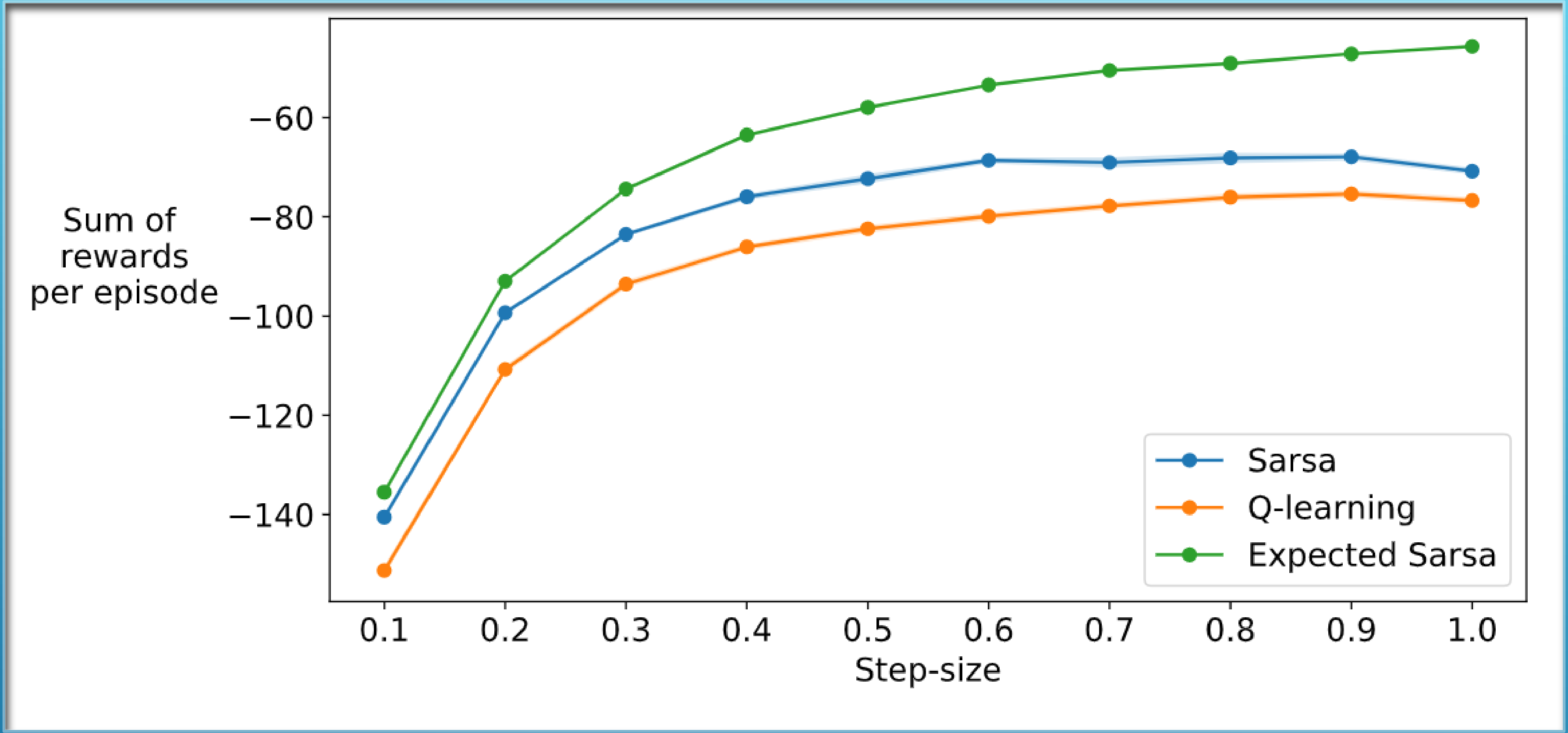
SARSA VS. EXPECTED SARSA

- 100 runs / 500 episodes per run.
- Average the number of visits to a state during the last 10 episodes over 100 runs.



Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

STEP-SIZE: SARSA VS. Q-LEARNING VS. EXPECTED SARSA



- 100 runs / 100 episodes per run.
- Average the sum of rewards for each episode over 100 runs for each step-size.

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

CONCLUSIONS

- Q-Learning will learn the optimal policy for an MDP but cannot fully exploit it unless it stops exploring.
- If q-learning continues to explore, the total value per episode will be sub-optimal.
- Expected Sarsa can find an optimal policy for a blend of exploitation and exploration.
- However, the computational overhead for Expected Sarsa is significant.
- The Sarsa algorithm can "play it safe" since it learns the policy it actually carries out.

THREE TEMPORAL DIFFERENCE LEARNING ALGORITHMS

Scott O'Hara
Metrowest Developers Machine Learning Group
7/8/2020