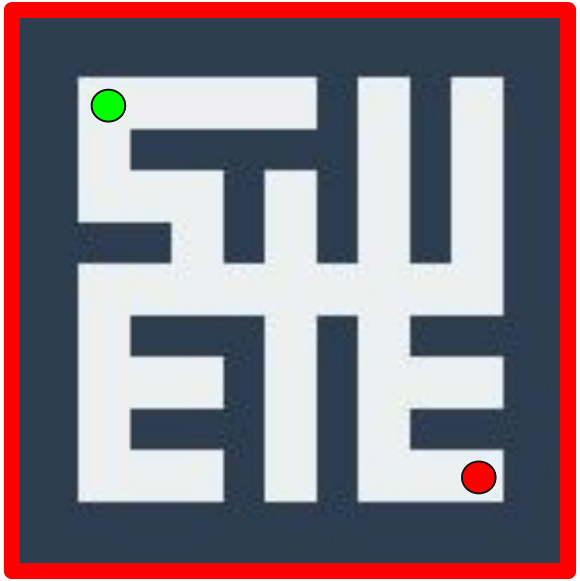
Cereal Box Hero

Comparing Model-Free Reinforcement Learning Approaches in Maze Solving

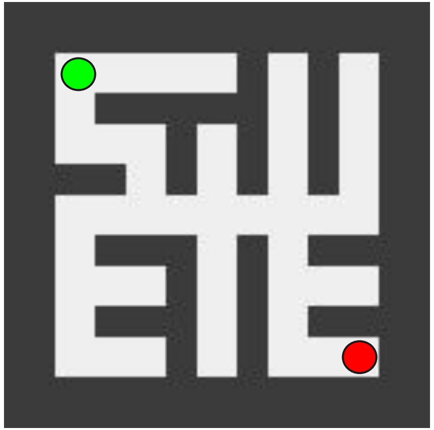
Goal: Solve a 9x9 Maze

9x9 maze images generated using Kruskal's algorithm



Maze Analysis and Representation

- Apply grayscale
- Maze represented as a reward matrix (Walls: -100, Path: -1, End: 10000)



```
[-100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100]
[-100   -1   -1   -1   -1   -1 -100   -1 -100   -1 -100]
[-100   -1 -100 -100 -100 -100 -100   -1 -100   -1 -100]
[-100   -1   -1   -1 -100   -1 -100   -1 -100   -1 -100]
[-100 -100 -100   -1 -100   -1 -100   -1 -100   -1 -100]
[-100   -1   -1   -1   -1   -1   -1   -1   -1   -1 -100]
[-100   -1 -100 -100 -100   -1 -100   -1 -100 -100 -100]
[-100   -1   -1   -1 -100   -1 -100   -1   -1   -1 -100]
[-100   -1 -100 -100 -100   -1 -100   -1 -100 -100 -100]
[    -100        -1        -1        -1      -100        -1      -100
  1000000      -100]
[-100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100]
```

Reinforcement Learning

Markov Decision Process

State: s

Action: $A(s)$

Policy: $\pi(s, a) = Pr(a = a | s = s)$

Reward: $R(s),\ R(s, a),\ R(s, a, s')$

Value Function

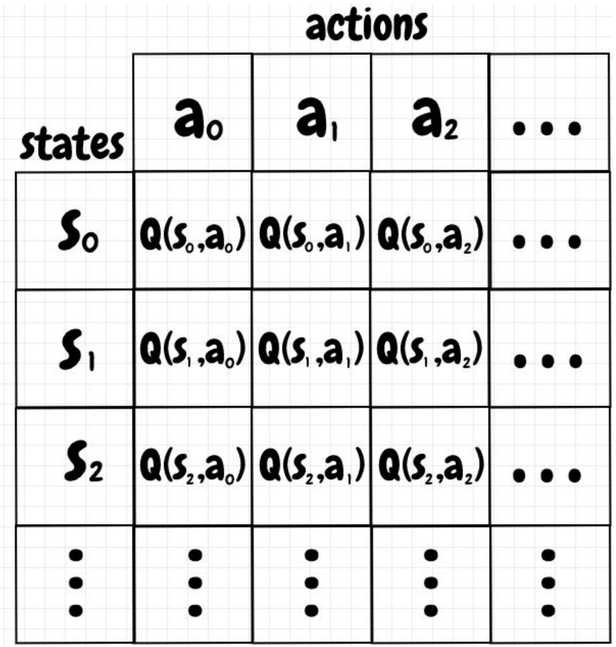
- Used to optimize policy to increase reward size or frequency

$$V_{\pi}(s) = \left(\sum_{t=0}^{n} \gamma^{t} r_{t} | s_{0} = s\right)$$

Discount rate Reward

Model-Free Reinforcement Learning

- No pre-built model of environment
- Measures and stores “quality” of a state/action pair based on a value function
- Better rewards → higher q-values
- Often based on either Temporal Difference (TD) or Gradient-Based Learning methods

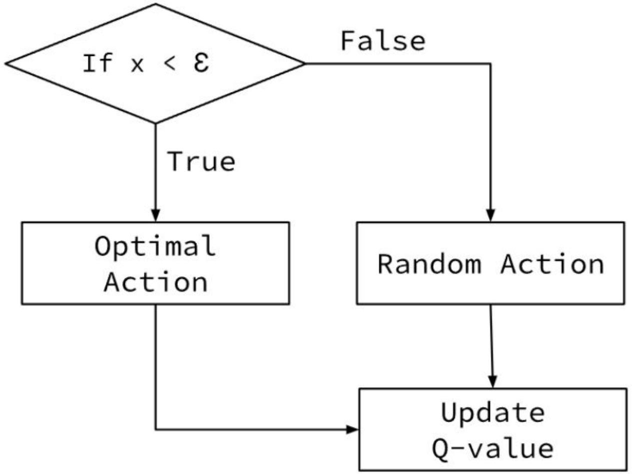


	actions			
states	a_0	a_1	a_2	...
s_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$	...
s_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$	...
s_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$	...
⋮	⋮	⋮	⋮	⋮

Model-Free Reinforcement Learning

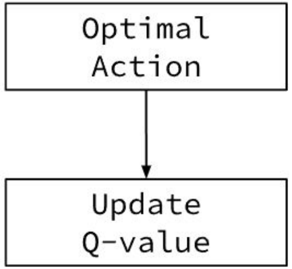
Q-Learning

- “Quality” Learning
- Can use ON, OFF, or HYBRID policy using an Epsilon-greedy function



SARSA

- “State Action Reward State Action”
- Always ON policy



Both methods based on TD-learning...

Temporal Difference Learning (TD-Learning) Function:

$$V^{new}(s_k) = V^{old}(s_k) + \alpha\,\overline{[r_k + \gamma V^{old}(s_{k+1}) - V^{old}(s_k)]}$$

Learning rate

Q-Learning Function:

$$Q_2(s,a) = Q_1(s,a) + \alpha\left[(r_k + \gamma \max_a Q_1(s_{t+1},a) - Q_1(s,a)\right]$$

Q-Learning Function:

$$Q_2(s, a) = Q_1(s, a) + \alpha\left[(r_k + \gamma \max_a Q_1(s_{t+1}, a) - Q_1(s, a)\right]$$

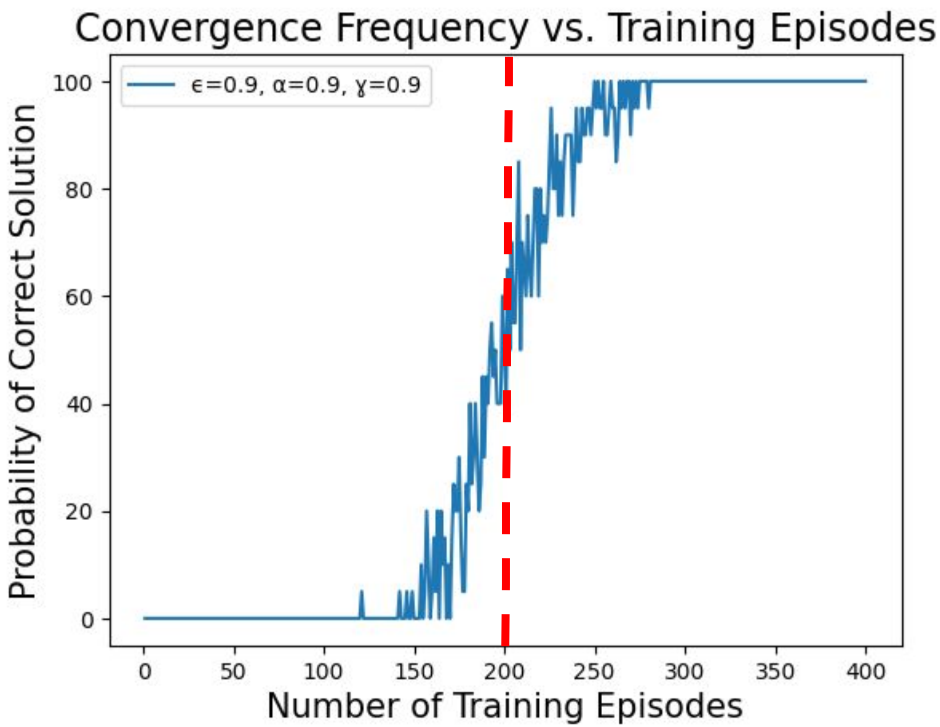
SARSA Function:

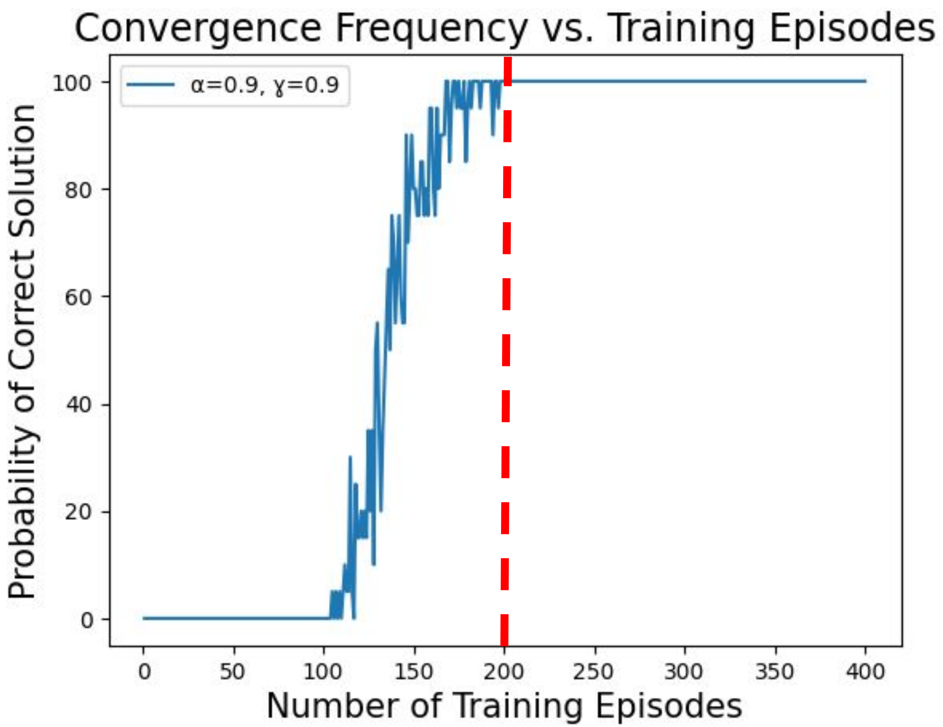
$$Q_2(s, a) = Q_1(s, a) + \alpha\left[(r_k + \gamma Q_1(s_{t+1}, a_{t+1}) - Q_1(s, a)\right]$$

Training Phase

1) Start at random location/state on the path
2) Move up, down, right, or left, depending on largest Q-value for given location
    a) **<u>Q-Learning</u>**: 10% of the time, a random action is chosen
    b) **<u>SARSA</u>**: Optimal action is always chosen
3) Update last Q-value depending on whether or not the move was on path
4) Repeat 1-3 if last action was on path

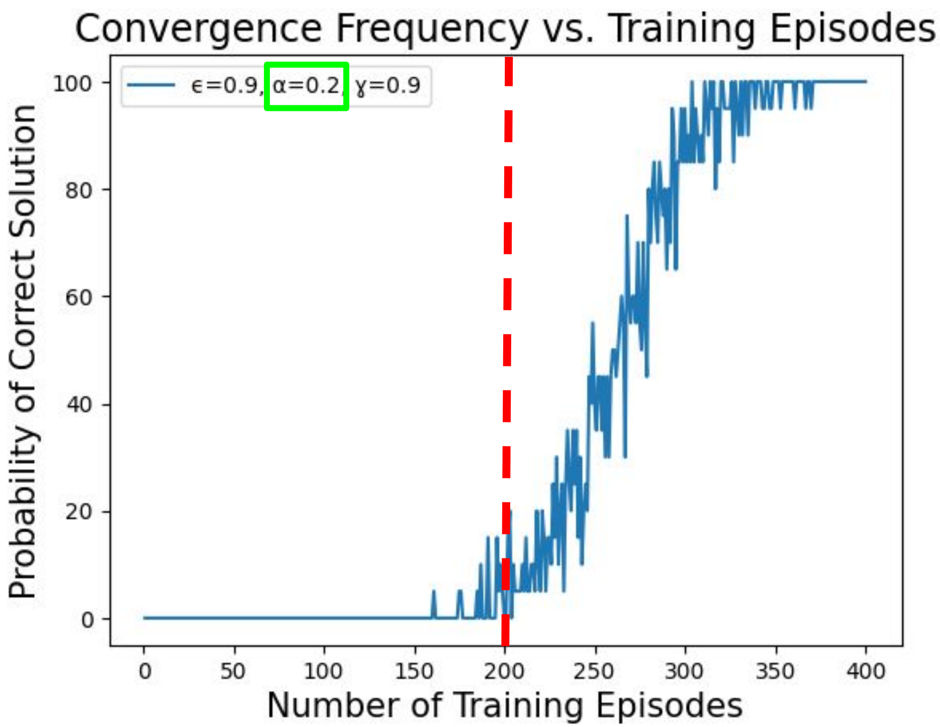
Continue until Q-tables converge upon an optimal solution!



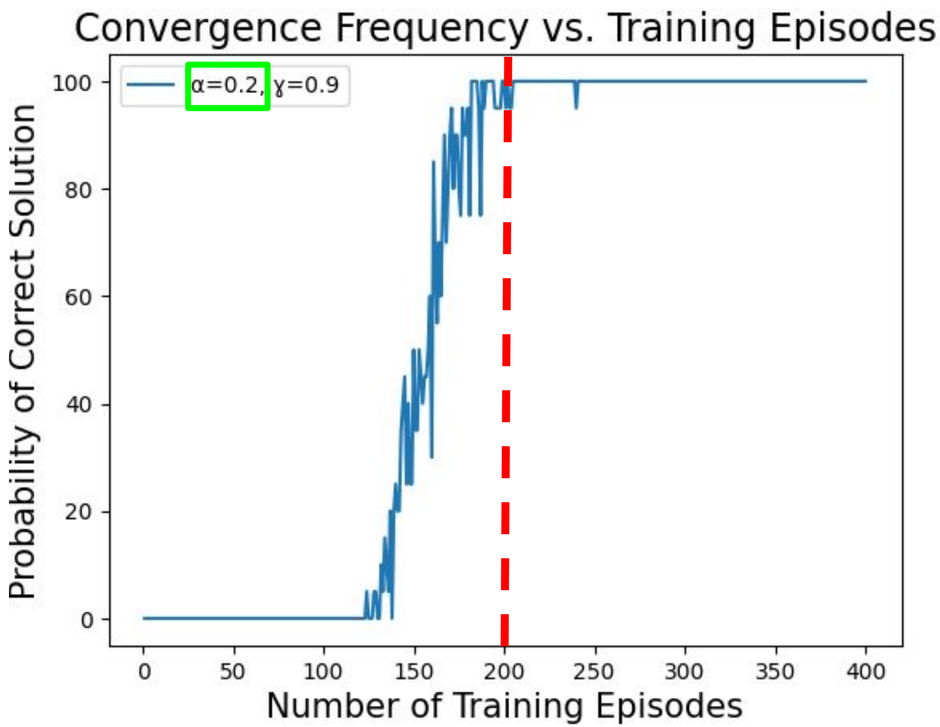


Q-Learning

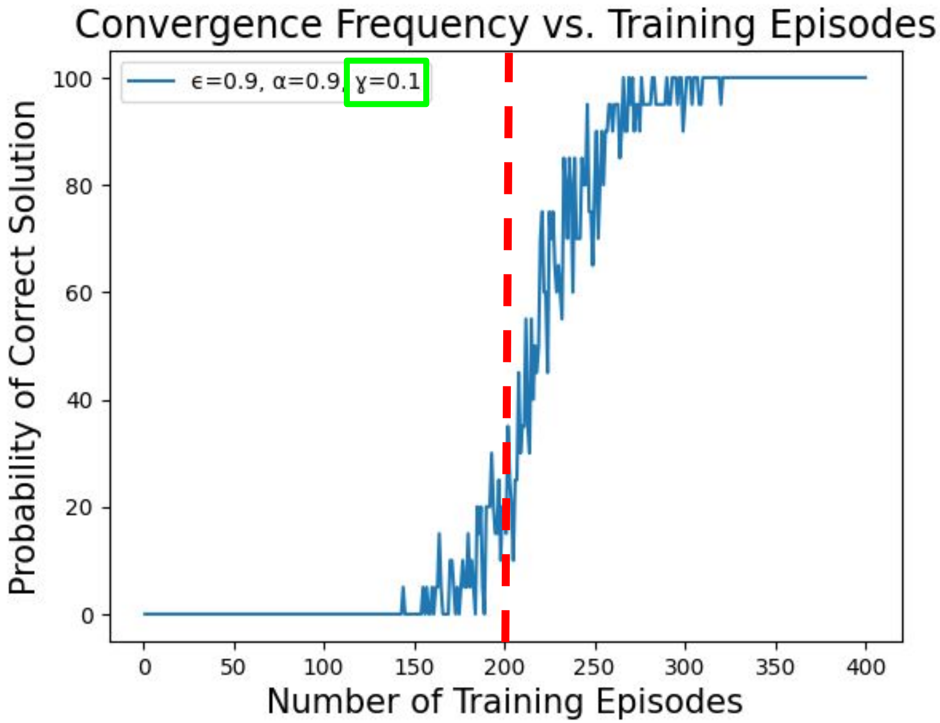
SARSA



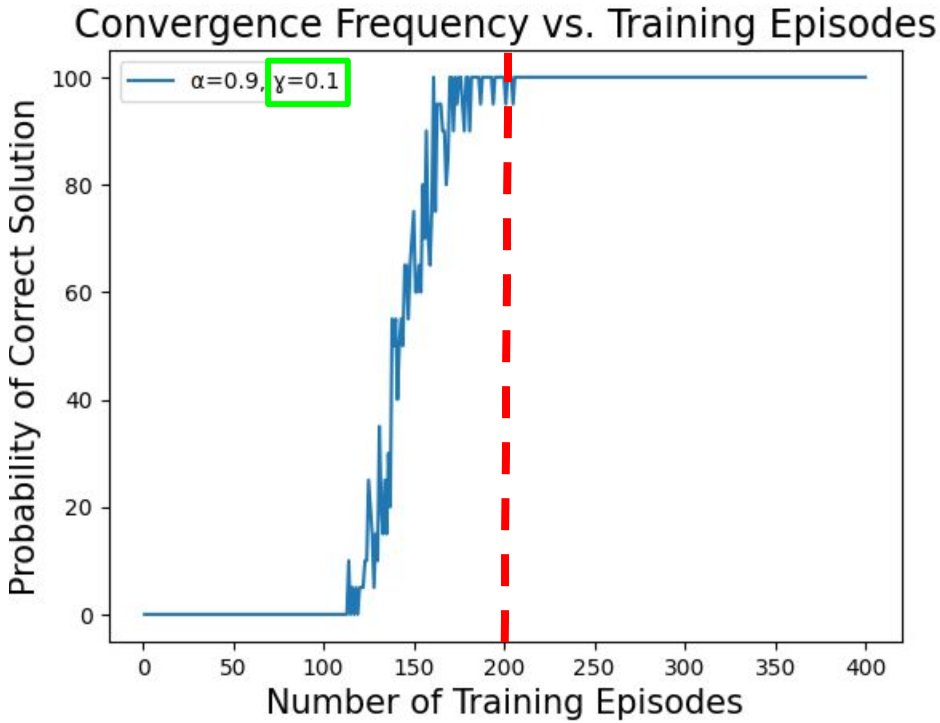
Q-Learning



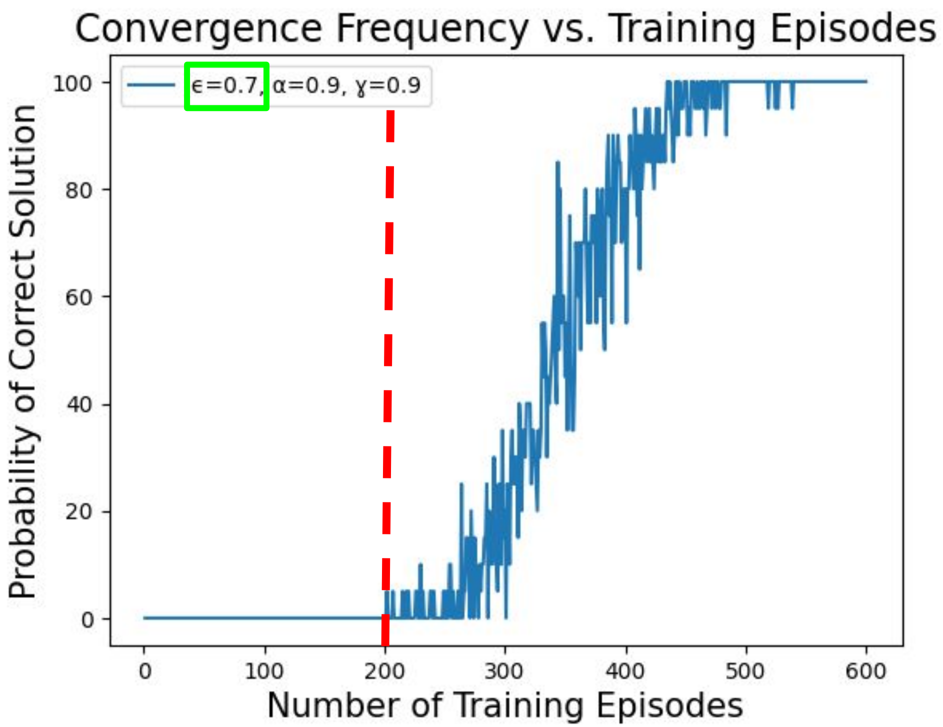
SARSA



Q-Learning



SARSA



Epsilon change not applicable

Q-Learning

SARSA

Notes on Model-Free Reinforcement Learning

- Q-learning is more widely applied than SARSA
- Deep reinforcement Q-learning
  - Gradient-based algorithms can be applied to continuous action spaces
  - Surpassed human expert control in video games
  - Frameworks are becoming more lightweight
- Some algorithms combine value-based and policy-based approaches
  - Actor-critic methods combine TD-learning and policy gradients