Implementing Actor-Critic Architecture on Grid World

CAP6629 Reinforcement Learning Project 3

Fanchen Bao

Introduction

- Tabular method
  - Small state and action space
  - Curse of dimensionality
- Actor-Critic
  - Neural network to simulate state value function and policy function
  - Good for large or continuous state or action space

Formal Definition

- Parameterize total reward function (and policy function)

$$J(\theta) = \sum_{s \in S} d^\pi(s) V^\pi(s) = \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a|s) Q^\pi(s, a)$$

Formal Definition

- Policy gradient theorem [1]

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a|s) Q^\pi(s,a) \\
&\propto \sum_{s \in S} d^\pi(s) \sum_{a \in A} \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \\
&\propto \mathbb{E}_\pi [\underset{\text{Critic}}{\boxed{Q^\pi(s,a)}} \nabla_\theta \ln \underset{\text{Actor}}{\boxed{\pi_\theta(a|s)}}]
\end{aligned}
$$

Formal Definition

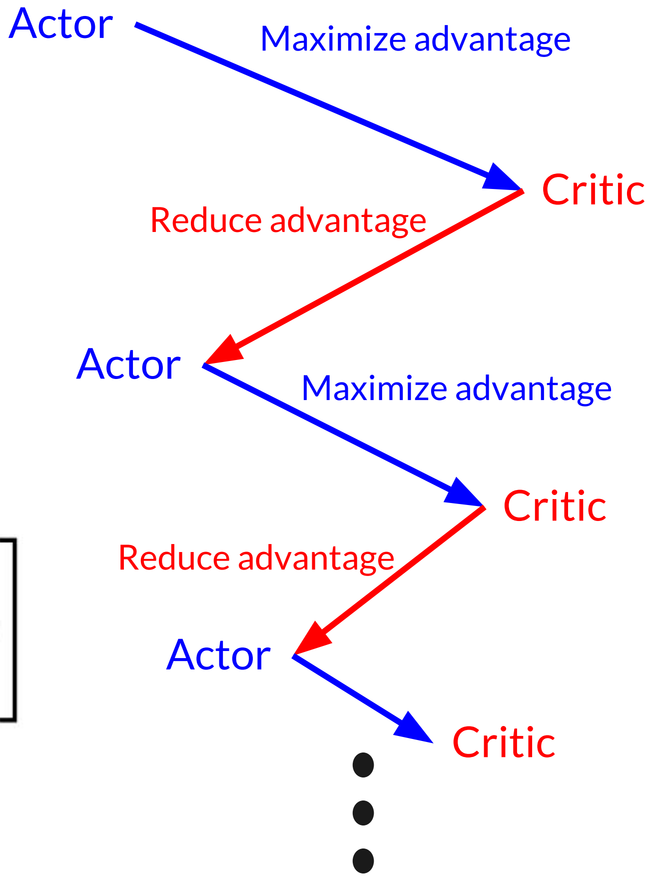
- Policy gradient theorem [2]

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \overbrace{\boxed{(G_t - \underbrace{\boxed{V_w(s_t)}}_{\text{Critic}})}}^{\text{Advantage}} \nabla_\theta \ln \underbrace{\boxed{\pi_\theta(a|s)}}_{\text{Actor}} \right]$$

Formal Definition

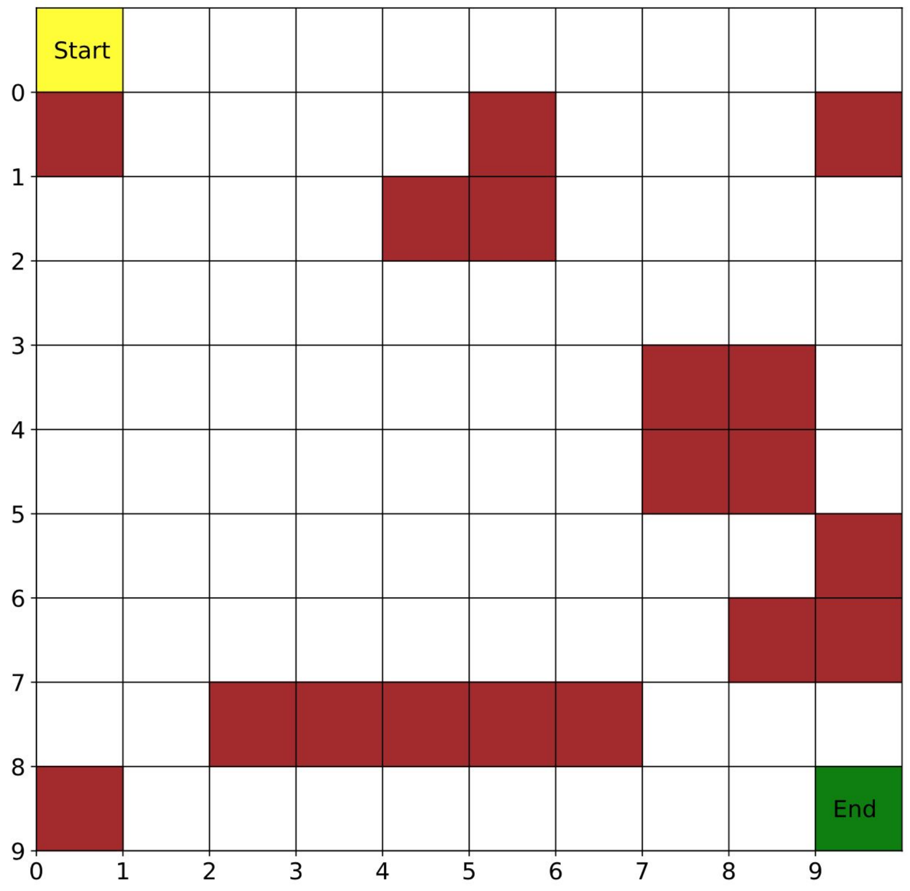
- Antagonistic interaction between actor and critic

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \overbrace{\boxed{(G_t - \underbrace{\boxed{V_w(s_t)}}_{\text{Critic}})}}^{\text{Advantage}} \nabla_\theta \ln \underbrace{\boxed{\pi_\theta(a|s)}}_{\text{Actor}}\right]$$



Implementation

- Grid World
  - Move: ↑↓←→
  - Penalty per move: -1
  - Find optimal path



Implementation

- Pseudo-code
  - $\gamma = 0.9$
  - $\alpha_\theta = 0.01$
  - $\alpha_w = 0.01$

repeat

    for $t = 1...T$ **do**: ▷ Each step in an episode

        $a, P_a \leftarrow \pi_\theta(a|s)$ ▷ Sample action and its probability

        $\mathrm{c} \leftarrow \mathrm{V}_w(s)$ ▷ Obtain estimated critic value

        $r \leftarrow -1$ ▷ Obtain the reward, which is always -1

        Append c to C_{val}

        Append P_a to P_{act}

        Append r to R

        Update s by applying a to s

    end for

Monte Carlo

    $g \leftarrow 0$ ▷ Accumulation of discounted reward

    for r in reversed R **do**: ▷ The discounted total reward is computed in reverse

        $g \leftarrow \mathrm{r} + \gamma g$

        Append g to G

    end for

    Reverse G

    $\mathrm{A} \leftarrow \mathrm{G}$ - C_{val} ▷ Obtain advantage via pair-wise subtraction

    $L_c \leftarrow \mathrm{A}^2/2T$ ▷ Mean squared error, critic loss

    $L_a \leftarrow$-$\ln(P_{act}) \cdot A$ ▷ Dot product, actor loss

    Compute $\nabla_\theta \pi_\theta$ based on L_a ▷ Obtain actor gradient

    $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \pi_\theta$ ▷ Update actor parameters

    Compute $\nabla_w V_w$ based on L_c ▷ Obtain critic gradient

    $w \leftarrow \mathrm{w} + \alpha_w \nabla_w V_w$ ▷ Update critic parameters

    $\mathrm{ep} \leftarrow \mathrm{ep} + 1$

until $\mathrm{ep} = \mathrm{max\_eps}$

Implementation

- Pseudo-code
  - $\gamma = 0.9$
  - $\alpha_\theta = 0.01$
  - $\alpha_w = 0.01$

repeat
    for $t = 1...T$ **do**: ▷ Each step in an episode
        $a, P_a \leftarrow \pi_\theta(a|s)$ ▷ Sample action and its probability
        $\mathrm{c} \leftarrow \mathrm{V}_w(s)$ ▷ Obtain estimated critic value
        $r \leftarrow -1$ ▷ Obtain the reward, which is always -1
        Append c to C_{val}
        Append P_a to P_{act}
        Append r to R
        Update s by applying a to s
    end for

    $g \leftarrow 0$ ▷ Accumulation of discounted reward
    for r in reversed R **do**: ▷ The discounted total reward is computed in reverse
        $g \leftarrow \mathrm{r} + \gamma g$
        Append g to G
    end for
    Reverse G

Discounted rewards

    $\mathrm{A} \leftarrow \mathrm{G} - \mathrm{C}_{val}$ ▷ Obtain advantage via pair-wise subtraction
    $L_c \leftarrow \mathrm{A}^2/2T$ ▷ Mean squared error, critic loss
    $L_a \leftarrow \text{-}\ln(P_{act}) \cdot A$ ▷ Dot product, actor loss

    Compute $\nabla_\theta \pi_\theta$ based on L_a ▷ Obtain actor gradient
    $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \pi_\theta$ ▷ Update actor parameters
    Compute $\nabla_w V_w$ based on L_c ▷ Obtain critic gradient
    $w \leftarrow \mathrm{w} + \alpha_w \nabla_w V_w$ ▷ Update critic parameters

    $\mathrm{ep} \leftarrow \mathrm{ep} + 1$
until $\mathrm{ep} = \mathrm{max\_eps}$

Implementation

- Pseudo-code
  - $\gamma = 0.9$
  - $\alpha_\theta = 0.01$
  - $\alpha_w = 0.01$

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} (G_t - V_w(s_t)) \nabla_\theta \ln \pi_\theta(a|s) \right]$$

repeat
    for $t = 1...T$ **do**: ▷ Each step in an episode
        $a, P_a \leftarrow \pi_\theta(a|s)$ ▷ Sample action and its probability
        c $\leftarrow \mathrm{V}_w(s)$ ▷ Obtain estimated critic value
        $r \leftarrow -1$ ▷ Obtain the reward, which is always -1
        Append c to C_{val}
        Append P_a to P_{act}
        Append r to R
        Update s by applying a to s
    end for

    $g \leftarrow 0$ ▷ Accumulation of discounted reward
    for r in reversed R **do**: ▷ The discounted total reward is computed in reverse
        $g \leftarrow \mathrm{r} + \gamma g$
        Append g to G
    end for
    Reverse G

    A $\leftarrow$ G - C_{val} ▷ Obtain advantage via pair-wise subtraction
    $L_c \leftarrow \mathrm{A}^2/2T$ ▷ Mean squared error, critic loss
    $L_a \leftarrow$-ln$(P_{act}) \cdot A$ ▷ Dot product, actor loss

Compute loss

    Compute $\nabla_\theta \pi_\theta$ based on L_a ▷ Obtain actor gradient
    $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \pi_\theta$ ▷ Update actor parameters
    Compute $\nabla_w V_w$ based on L_c ▷ Obtain critic gradient
    $w \leftarrow \mathrm{w} + \alpha_w \nabla_w V_w$ ▷ Update critic parameters

    ep $\leftarrow$ ep $+ 1$
until ep = max_eps

Implementation

- Pseudo-code
  - $\gamma = 0.9$
  - $\alpha_\theta = 0.01$
  - $\alpha_w = 0.01$

repeat
    for $t = 1 \ldots T$ **do**: ▷ Each step in an episode
        $a, P_a \leftarrow \pi_\theta(a|s)$ ▷ Sample action and its probability
        $\mathrm{c} \leftarrow \mathrm{V}_w(s)$ ▷ Obtain estimated critic value
        $r \leftarrow -1$ ▷ Obtain the reward, which is always -1
        Append c to C_{val}
        Append P_a to P_{act}
        Append r to R
        Update s by applying a to s
    end for

    $g \leftarrow 0$ ▷ Accumulation of discounted reward
    for r in reversed R **do**: ▷ The discounted total reward is computed in reverse
        $g \leftarrow \mathrm{r} + \gamma g$
        Append g to G
    end for
    Reverse G

    $\mathrm{A} \leftarrow \mathrm{G} - \mathrm{C}_{val}$ ▷ Obtain advantage via pair-wise subtraction
    $L_c \leftarrow \mathrm{A}^2/2T$ ▷ Mean squared error, critic loss
    $L_a \leftarrow -\ln(P_{act}) \cdot A$ ▷ Dot product, actor loss

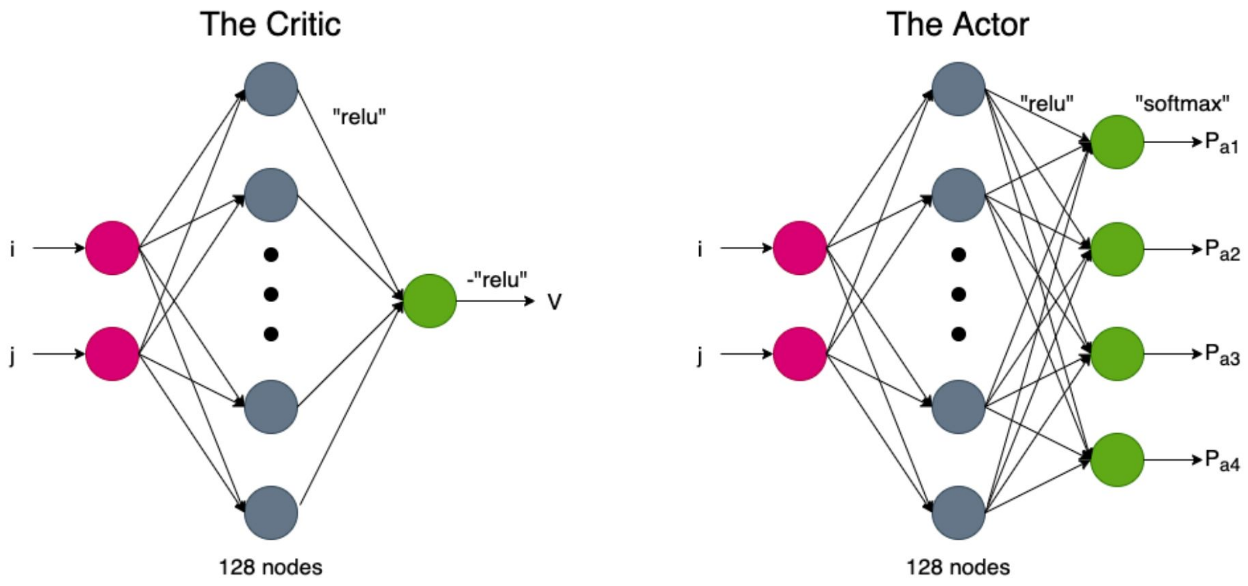
    Compute $\nabla_\theta \pi_\theta$ based on L_a ▷ Obtain actor gradient
    $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \pi_\theta$ ▷ Update actor parameters
    Compute $\nabla_w V_w$ based on L_c ▷ Obtain critic gradient
    $w \leftarrow \mathrm{w} + \alpha_w \nabla_w V_w$ ▷ Update critic parameters

Gradient descent

    $\mathrm{ep} \leftarrow \mathrm{ep} + 1$
until $\mathrm{ep} = \mathrm{max\_eps}$

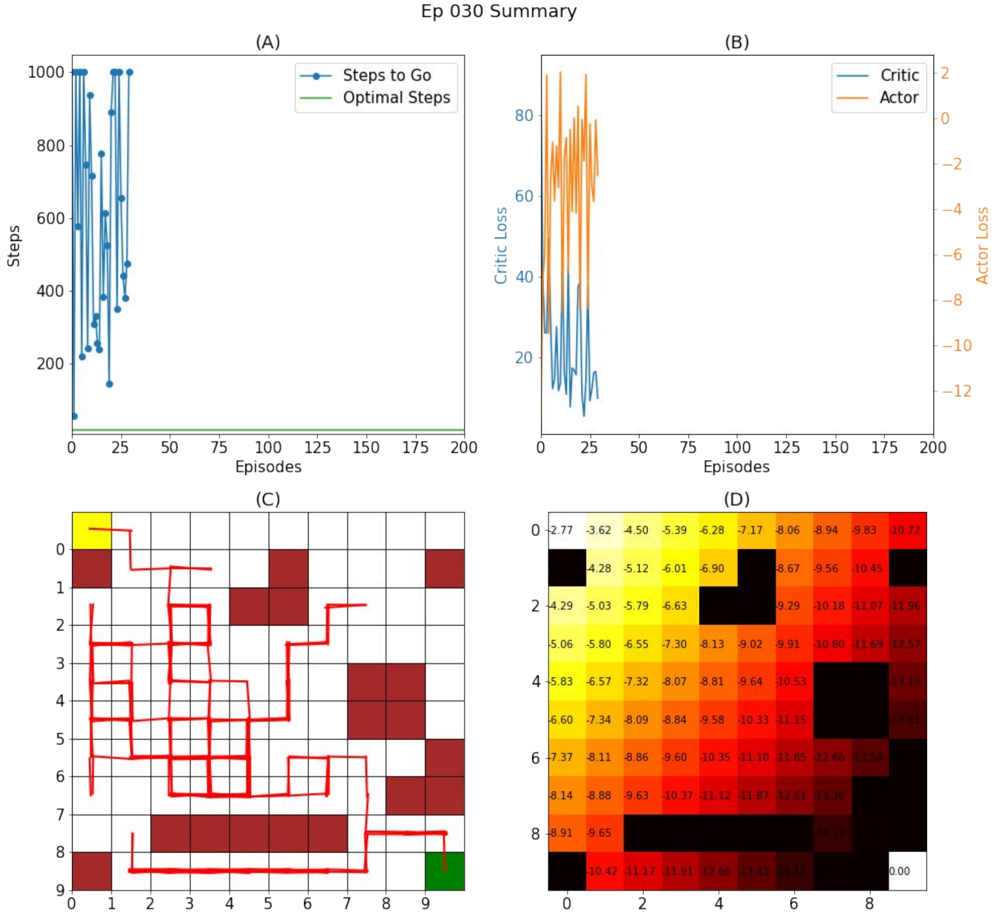
Implementation

- Neural network



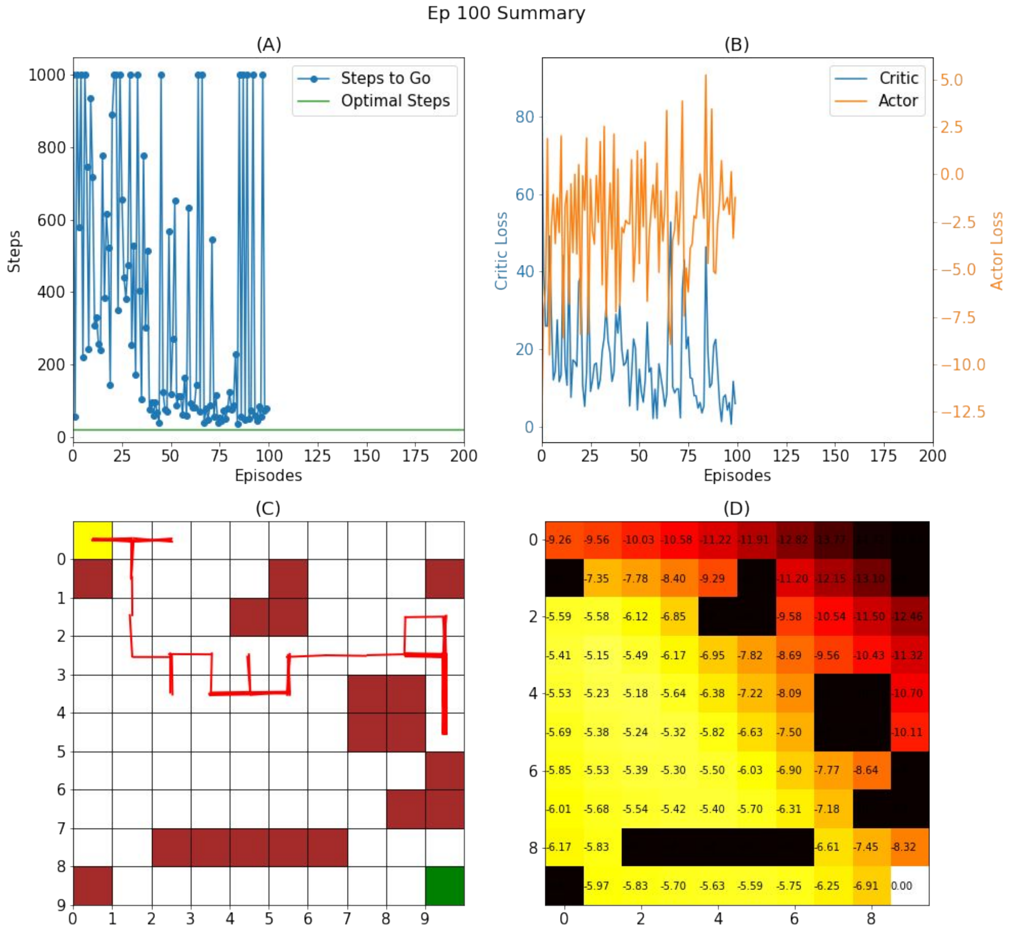
Evaluation

- Two openings



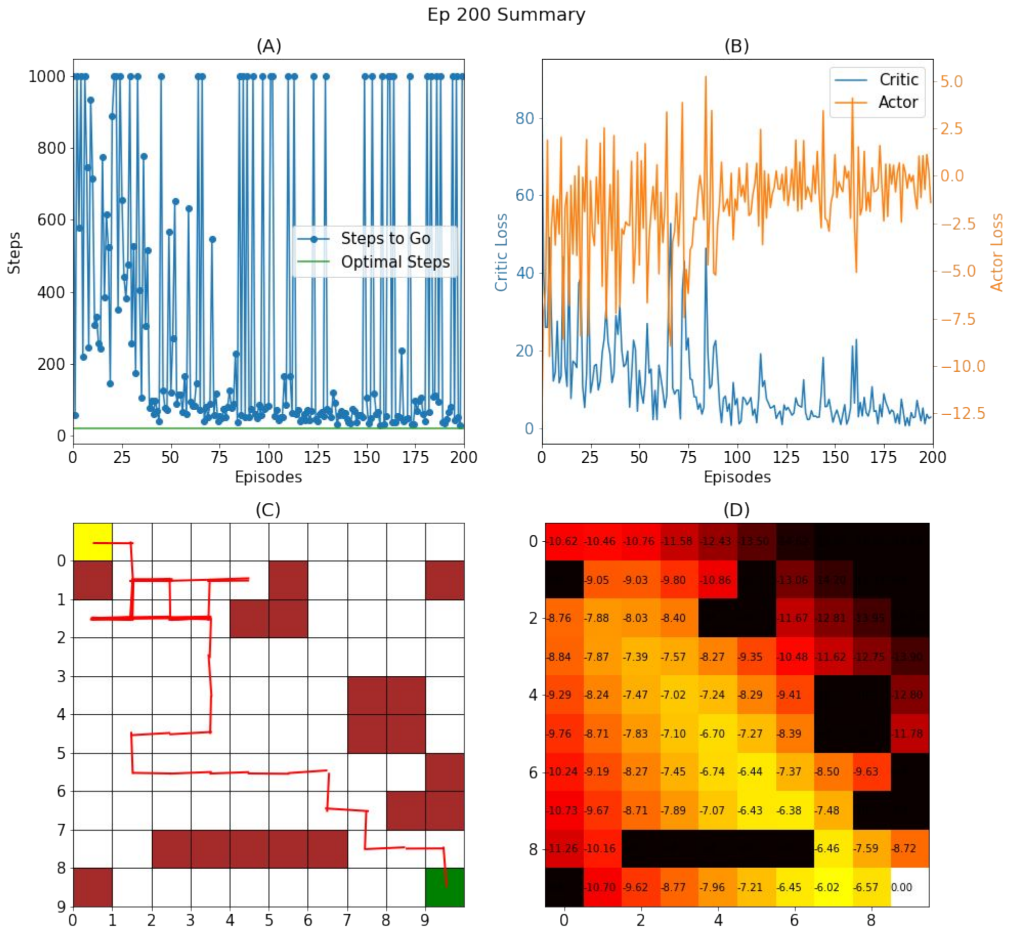
Evaluation

- Two openings



Evaluation

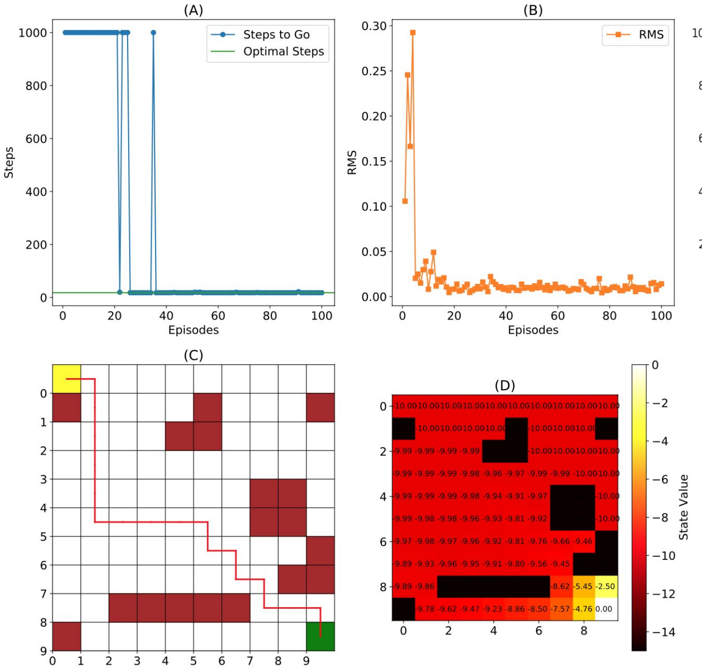
- Two openings



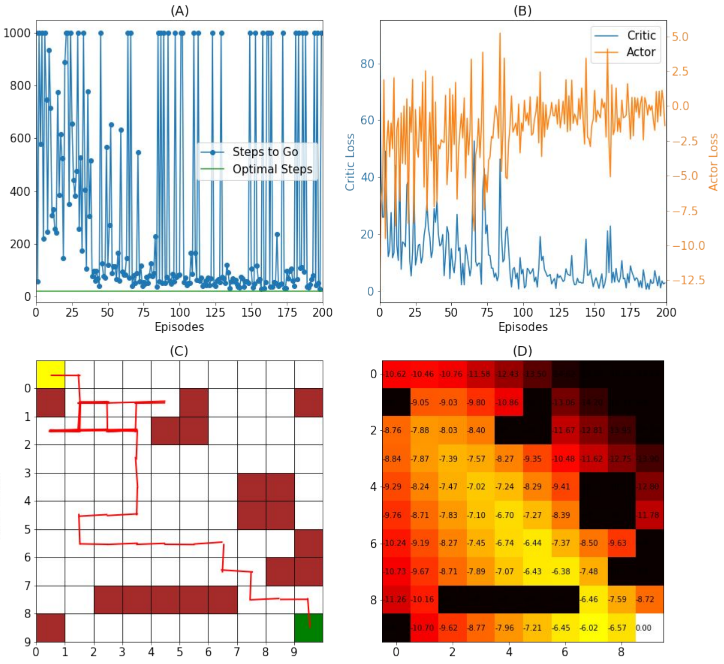
Evaluation

- Comparison with TD-λ

TD-λ

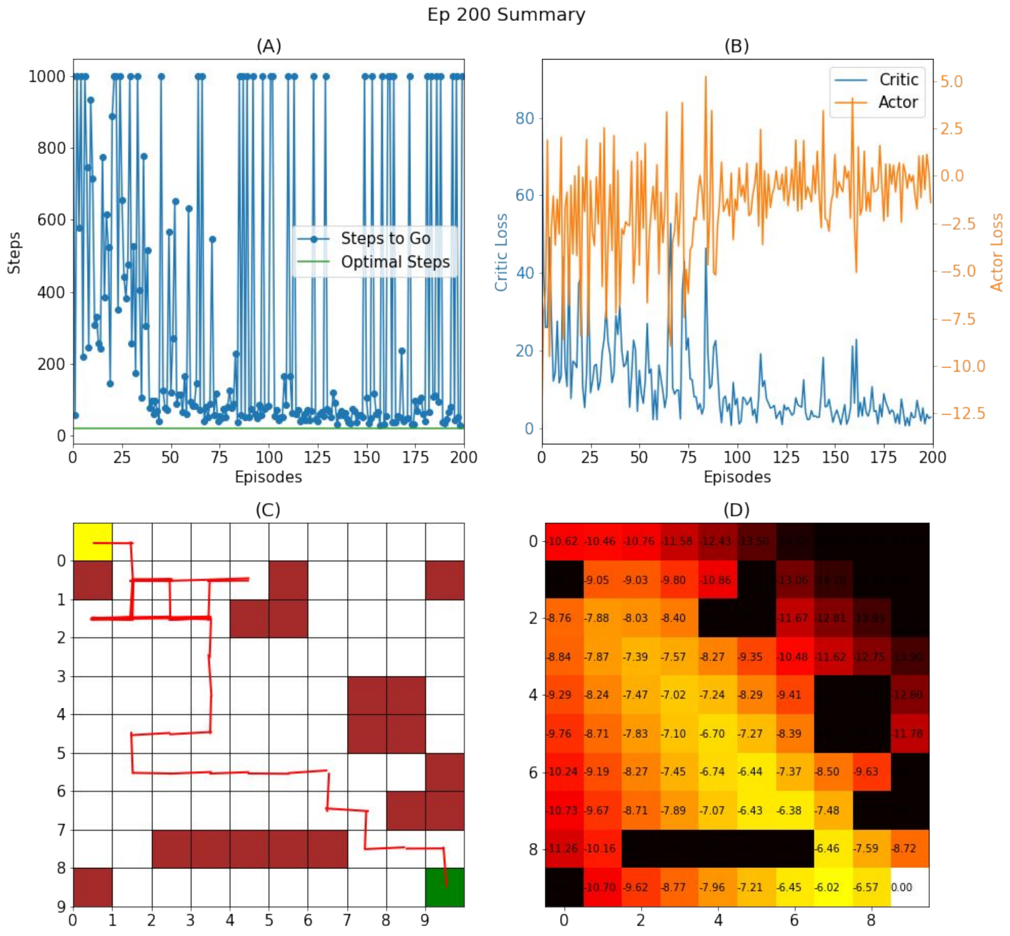


Actor-Critic



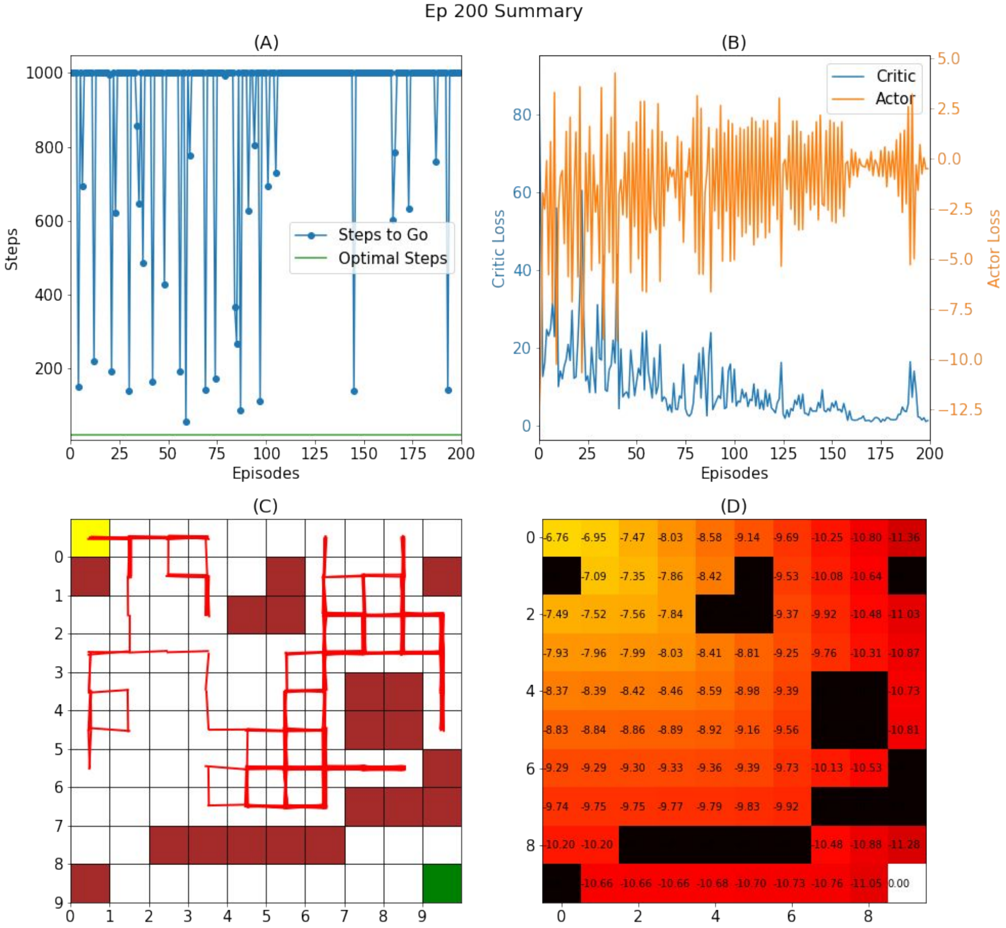
Evaluation

- Only the right opening is used



Evaluation

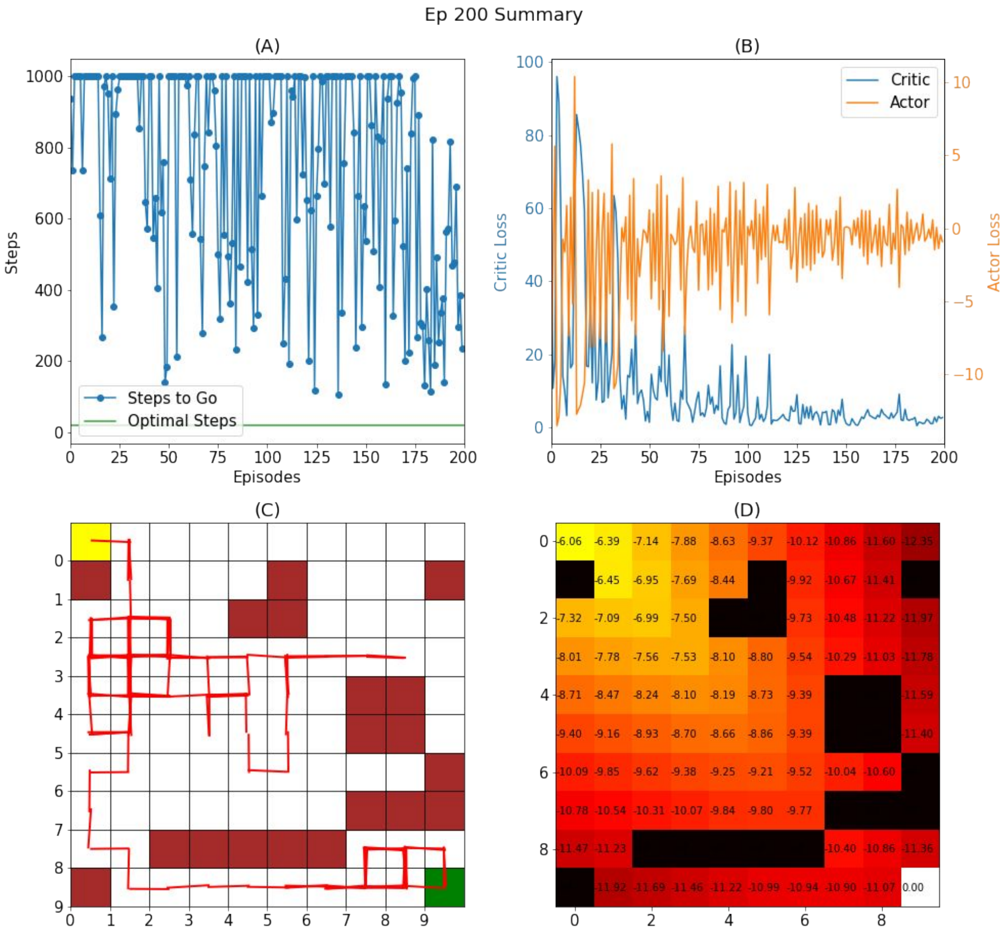
- One opening



Evaluation

- One opening
- Bottom right to top left

Domain knowledge is important!



Conclusions

- Actor-Critic depends on random walk initially.
- Actor-Critic only approximate optimal solution.
- Actor-Critic is slower and less stable in training than TD-λ.
- Domain knowledge crucial in solving reinforcement learning problem.

References

- [1] L. Weng, “Policy Gradient Algorithms,” Apr. 2018. [Online]. Available: https://lilianweng.github.io/2018/04/08/policy-gradient-algorithms.html
- [2] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-Dimensional Continuous Control Using Generalized Advantage Estimation,” arXiv:1506.02438 [cs], Oct. 2018, arXiv: 1506.02438. [Online]. Available: http://arxiv.org/abs/1506.02438