Deep Reinforcement Learning

Eligibility Traces

February 10, 2025

Vinay Kumar | Amberg

Table of Contents

Problem Statement

Ostbayerische Technische Hochschule Amberg-Weiden

How do we find good G_t?

- Should the agent rely on immediate reward ?

Problem Statement

Ostbayerische
Technische Hochschule
Amberg-Weiden

How do we find good G_t?

- Should the agent rely on immediate reward ?
- Or should agent consider long-term reward ?

Brief Recall

TD Methods

- After taking action a_t, observe reward r_{t+1} and next state s_{t+1}.

Brief Recall

Ostbayerische Technische Hochschule Amberg-Weiden

TD Methods

- After taking action a_t, observe reward r_{t+1} and next state s_{t+1}.
- One-step TD target (G_t) and TD error (δ_t):

$$G_t = r_{t+1} + \gamma V(s_{t+1})$$

$$\delta_t = G_t - V(s_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

where γ is the discount factor.

Ostbayerische Technische Hochschule Amberg-Weiden

TD Methods

- After taking action a_t, observe reward r_{t+1} and next state s_{t+1}.
- One-step TD target (G_t) and TD error (δ_t):

$$G_t = r_{t+1} + \gamma V(s_{t+1})$$

$$\delta_t = G_t - V(s_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

  where γ is the discount factor.
- Update the value of the current state s_t:

$$V(s_t) \leftarrow V(s_t) + \alpha \delta_t$$

  where α is the learning rate.

Brief Recall

Monte Carlo Methods

- Play an episode until the end of terminal state (e.g., game over or goal reached).

Brief Recall

Monte Carlo Methods

- Play an episode until the end of terminal state (e.g., game over or goal reached).
- Compute the total return G_t from time t to the end:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$$

where γ is the discount factor.

Monte Carlo Methods

- Play an episode until the end of terminal state (e.g., game over or goal reached).
- Compute the total return G_t from time t to the end:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$$

  where γ is the discount factor.
- For each state s_t visited in the episode, update its value:

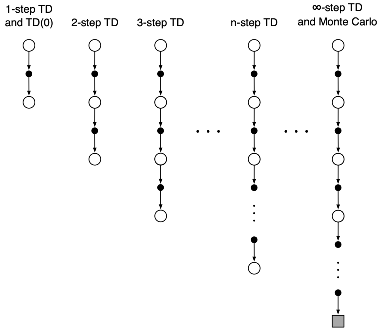
$$V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$$

  where α is the learning rate.

Brief Recall

n-step TD

- **Idea**: Look Farther into future when do TD backup (1, 2, 3,. . . , n steps)



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

Brief Recall

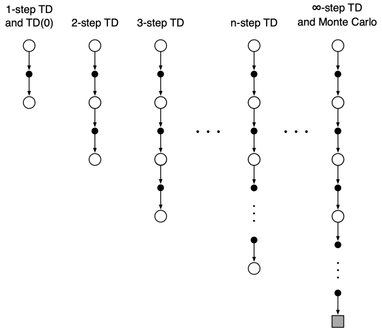
n-step TD

- **Idea**: Look Farther into future when do TD backup (1, 2, 3,. . . , n steps)
- The n-step return $G_t^{(n)}$ is given by:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

Where:

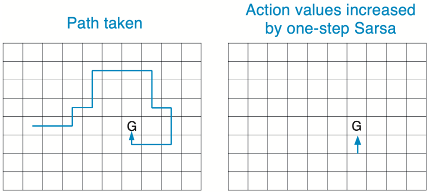
  - $R_{t+1}, R_{t+2}, \ldots, R_{t+n}$: rewards at each step
  - γ: discount factor
  - $V(S_{t+n})$: value of state at time $t + n$



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

Ostbayerische Technische Hochschule Amberg-Weiden

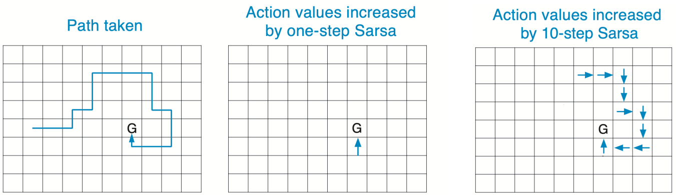
How can we learn most from the episode?



https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Intuition behind

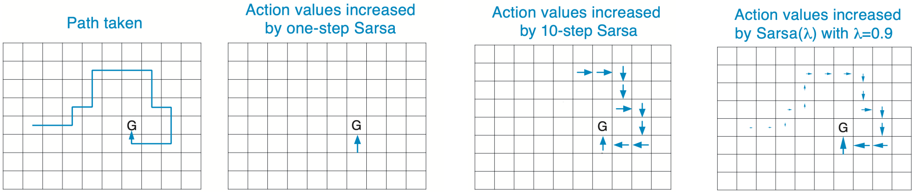
How can we learn most from the episode?



https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Intuition behind

How can we learn most from the episode?



https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Eligibility Tracing

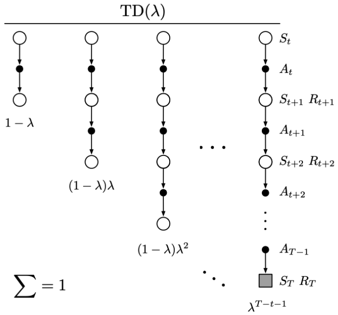
- A bridge from TD to Monte Carlo methods

Eligibility Tracing

- A bridge from TD to Monte Carlo methods
- A temporary record of which states/actions are eligible for learning updates

Key Concepts
The Forward View

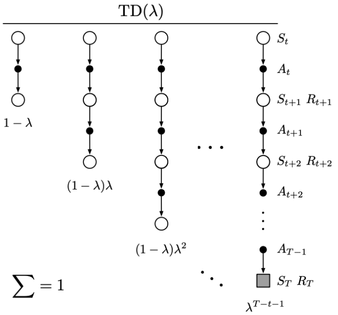
- The idea is to create an average of our n-step



http://incompleteideas.net/book/RLbook2020.pdf- Chapter 12

Key Concepts
The Forward View

- The idea is to create an average of our n-step
- Weighting of past experiences in eligibility traces is controlled by λ(lambda)

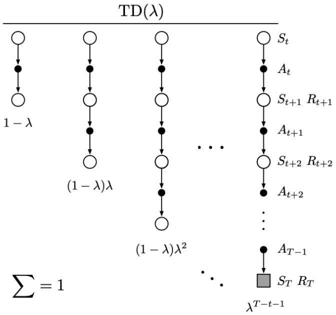


http://incompleteideas.net/book/RLbook2020.pdf- Chapter 12

Key Concepts

The Forward View

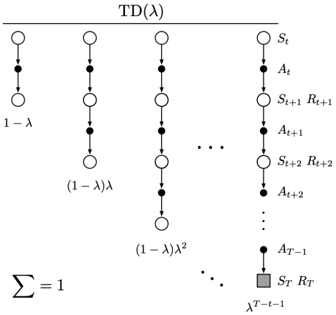
- The idea is to create an average of our n-step
- Weighting of past experiences in eligibility traces is controlled by λ(lambda)
- Weights on the component returns are positive and sum to 1



`http://incompleteideas.net/book/RLbook2020.pdf`- Chapter 12

Key Concepts
The Forward View

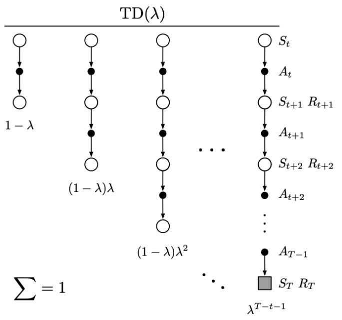
- The idea is to create an average of our n-step
- Weighting of past experiences in eligibility traces is controlled by λ(lambda)
- Weights on the component returns are positive and sum to 1
- This particular way of averaging n-step called as TD(λ)



http://incompleteideas.net/book/RLbook2020.pdf- Chapter 12

Key Concepts
The Forward View

- The idea is to create an average of our n-step
- Weighting of past experiences in eligibility traces is controlled by λ(lambda)
- Weights on the component returns are positive and sum to 1
- This particular way of averaging n-step called as TD(λ)
- Each state has a trace that keeps track how much credit that state should receive for future updates.

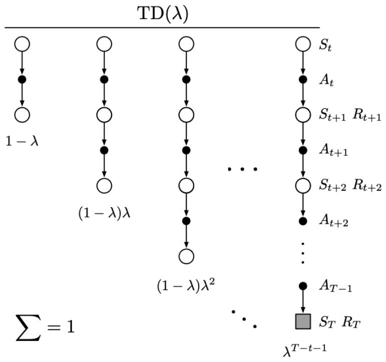


http://incompleteideas.net/book/RLbook2020.pdf- Chapter 12

How weighting n sequence works ?

TD(λ)

- The one-step return is the largest weight, given by 1-λ

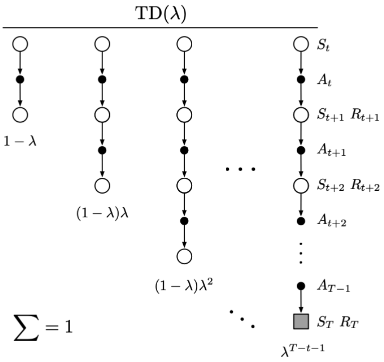


http://incompleteideas.net/book/RLbook2020.pdf- Chapter 7

How weighting n sequence works ?

TD(λ)

- The one-step return is the largest weight, given by 1-λ
- The weights fade away by λ with each additional steps

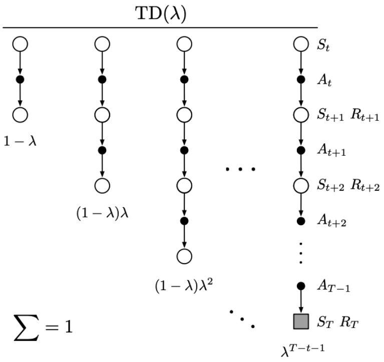


http://incompleteideas.net/book/RLbook2020.pdf- Chapter 7

How weighting n sequence works ?

TD(λ)

- The one-step return is the largest weight, given by 1-λ
- The weights fade away by λ with each additional steps
- If (λ =0) ?

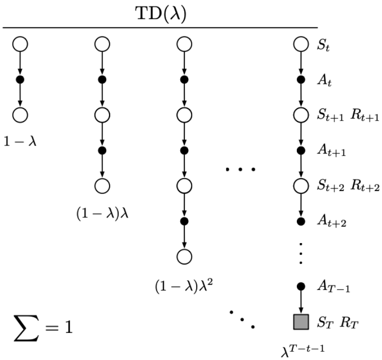


http://incompleteideas.net/book/RLbook2020.pdf- Chapter 7

How weighting n sequence works ?

TD(λ)

- The one-step return is the largest weight, given by 1-λ
- The weights fade away by λ with each additional steps
- If (λ =0) ?
- Overall update reduces to its first component, one-step TD

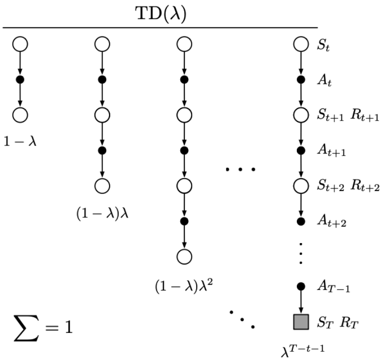


http://incompleteideas.net/book/RLbook2020.pdf- Chapter 7

How weighting n sequence works ?

TD(λ)

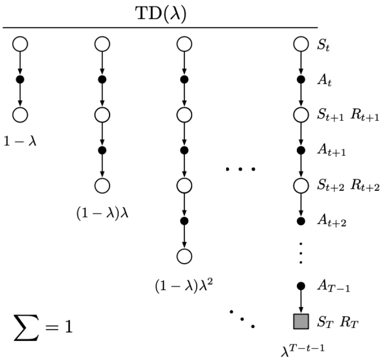
- The one-step return is the largest weight, given by 1-λ
- The weights fade away by λ with each additional steps
- If (λ =0) ?
- Overall update reduces to its first component, one-step TD
- If (λ =1) ?



http://incompleteideas.net/book/RLbook2020.pdf- Chapter 7

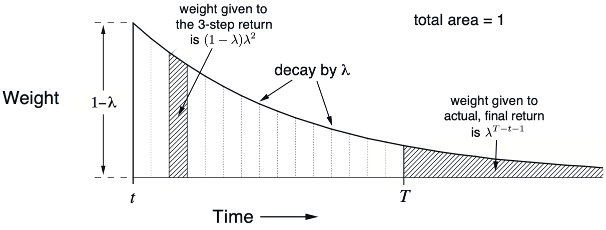
TD(λ)

- The one-step return is the largest weight, given by 1-λ
- The weights fade away by λ with each additional steps
- If (λ =0) ?
- Overall update reduces to its first component, one-step TD
- If (λ =1) ?
- Overall update reduces to last component, i.e Monte Carlo



http://incompleteideas.net/book/RLbook2020.pdf- Chapter 7

Working and showing

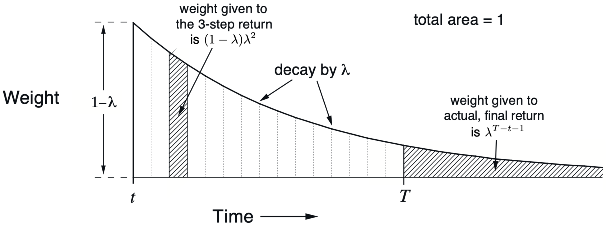


Weighting given in the λ-return to each of the n-step returns.

- λ characterizes how fast the exponential weighting falls off

https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Working and showing

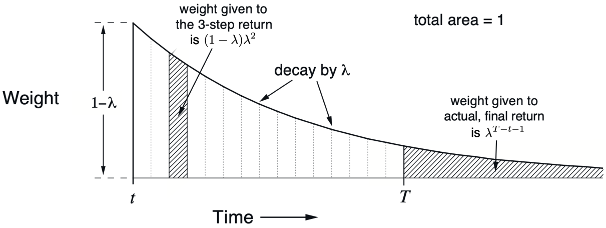


Weighting given in the λ-return to each of the n-step returns.

- λ characterizes how fast the exponential weighting falls off
- Thus how far into the future the λ look like determining the weights

https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Working and showing



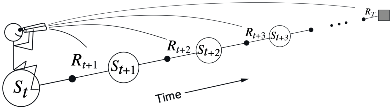
Weighting given in the λ-return to each of the n-step returns.

- λ characterizes how fast the exponential weighting falls off
- Thus how far into the future the λ look like determining the weights
- Known as Off-line λ-return algorithm

https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

The Forward View

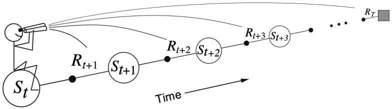
- Previously we discussed forward view (theoretical perspective)



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Forward View

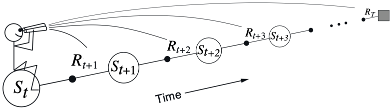
- Previously we discussed forward view (theoretical perspective)
- What we do is, we look ahead n steps for future rewards, and then we update our return



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Forward View

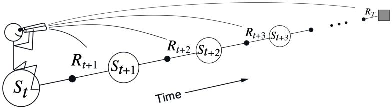
- Previously we discussed forward view (theoretical perspective)
- What we do is, we look ahead n steps for future rewards, and then we update our return
- **Problem arises when:**



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Forward View

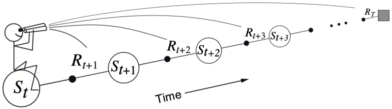
- Previously we discussed forward view (theoretical perspective)
- What we do is, we look ahead n steps for future rewards, and then we update our return
- **Problem arises when:**
- Cannot update until future rewards are known



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Forward View

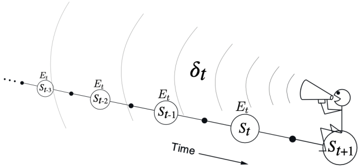
- Previously we discussed forward view (theoretical perspective)
- What we do is, we look ahead n steps for future rewards, and then we update our return
- **Problem arises when:**
- Cannot update until future rewards are known
- Somehow complex to implement because the update of each state depends on later events or rewards that are not available at current time



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Backward View

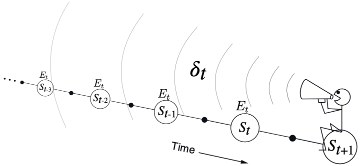
- Additional short term memory variable associated with each state



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Backward View

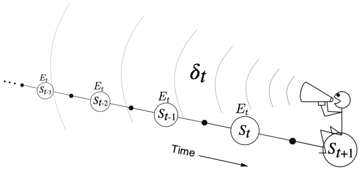
- Additional short term memory variable associated with each state
- Updates the value function continuously



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Backward View

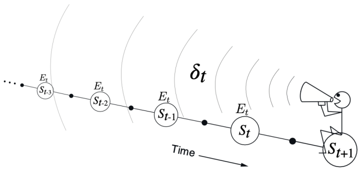
- Additional short term memory variable associated with each state
- Updates the value function continuously
- When a reward is received, not just the current state, but also the past states get some credit



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

The Backward View

- Additional short term memory variable associated with each state
- Updates the value function continuously
- When a reward is received, not just the current state, but also the past states get some credit
- Recently visited states get more credit, while older states get less credit over time.



https://bjpcjp.github.io/pdfs/math/nstep-bootstrap-RL.pdf

Pseudo Code

TD(λ)

Semi-gradient TD(λ) for estimating $\hat{v} \approx v_\pi$

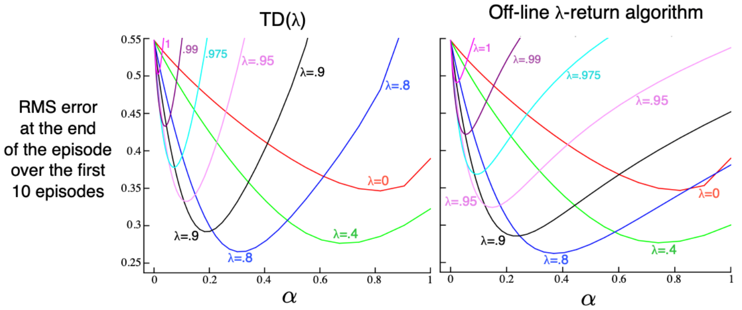
Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    Initialize S
    $\mathbf{z} \leftarrow \mathbf{0}$ (a d-dimensional vector)
    Loop for each step of episode:
    | Choose $A \sim \pi(\cdot|S)$
    | Take action A, observe R, S'
    | $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \nabla\hat{v}(S,\mathbf{w})$
    | $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
    | $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
    | $S \leftarrow S'$
    until S' is terminal

https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Comparison

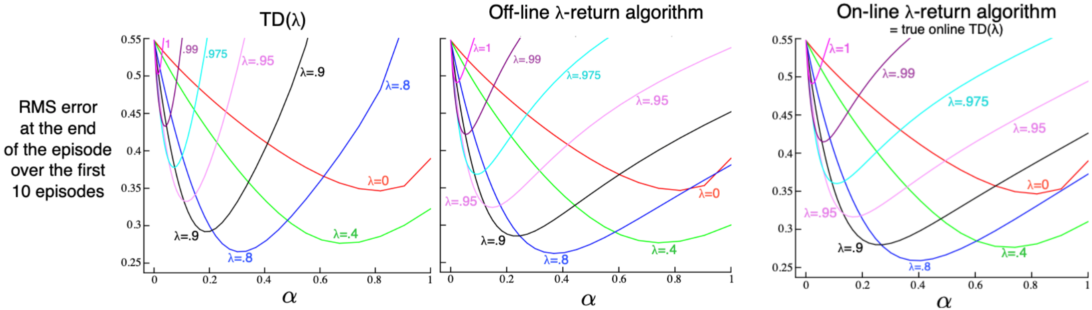
Performance Evaluation



https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Comparison

Performance Evaluation



https://bjpcjp.github.io/pdfs/math/eligibility-traces-RL.pdf

Improvement

Generalized Advantage Estimation (GAE)

- The advantage function measures how much better (or worse) taking a specific action is compared to the average following the current policy in a given state

Improvement

Generalized Advantage Estimation (GAE)

- The advantage function measures how much better (or worse) taking a specific action is compared to the average following the current policy in a given state
- The temporal difference (TD) error at each timestep is:

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Ostbayerische Technische Hochschule Amberg-Weiden

Generalized Advantage Estimation (GAE)

- The advantage function measures how much better (or worse) taking a specific action is compared to the average following the current policy in a given state
- The temporal difference (TD) error at each timestep is:

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- The total return G_t used in advantage calculation:

$$G_t = \delta_t + \gamma \delta_{t+1} + \cdots + \gamma^{n-1} \delta_{t+n-1}$$

which represents discounted sum of future rewards.

Improvement

The idea of eligibility traces

Generalized Advantage Estimation (GAE)

- Helps to estimate balance between immediate vs future rewards

Improvement

The idea of eligibility traces

Generalized Advantage Estimation (GAE)

- Helps to estimate balance between immediate vs future rewards
- Better decisions by looking at both short and long-term outcomes

Improvement

The idea of eligibility traces

Generalized Advantage Estimation (GAE)

- Helps to estimate balance between immediate vs future rewards
- Better decisions by looking at both short and long-term outcomes
- GAE sums over multiple TD errors from multiple time steps using eligibility traces

Improvement

The idea of eligibility traces

Generalized Advantage Estimation (GAE)

- Helps to estimate balance between immediate vs future rewards
- Better decisions by looking at both short and long-term outcomes
- GAE sums over multiple TD errors from multiple time steps using eligibility traces
- Estimate the advantage function, which measures how much better an action is compared to the average action in a given state

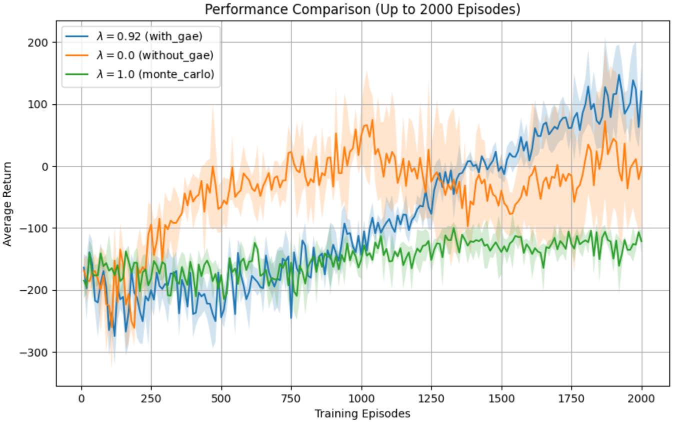
Improvement

The idea of eligibility traces

Generalized Advantage Estimation (GAE)

- Helps to estimate balance between immediate vs future rewards
- Better decisions by looking at both short and long-term outcomes
- GAE sums over multiple TD errors from multiple time steps using eligibility traces
- Estimate the advantage function, which measures how much better an action is compared to the average action in a given state
- Works particularly well with deep neural networks and replay buffers.

Comparison on Lunar-Lander



Conclusion

Key Takeaway

- GAE act as a modern solution for complex environments

Conclusion

Key Takeaway

- GAE act as a modern solution for complex environments
- GAE inherits eligibility traces, that provide better stability

Conclusion
Key Takeaway

- GAE act as a modern solution for complex environments
- GAE inherits eligibility traces, that provide better stability
- Improved performance in delayed reward scenarios

Conclusion

Key Takeaway

- GAE act as a modern solution for complex environments
- GAE inherits eligibility traces, that provide better stability
- Improved performance in delayed reward scenarios
- Usage in real world scenarios like Robotics control, Game AI which helps deciding what to do in situations where each choice affects the next

Conclusion
Key Takeaway

- GAE act as a modern solution for complex environments
- GAE inherits eligibility traces, that provide better stability
- Improved performance in delayed reward scenarios
- Usage in real world scenarios like Robotics control, Game AI which helps deciding what to do in situations where each choice affects the next
- Combining GAE with advance models like PPO, make stable learning in many environments particularly real-world robotics applications.

End !

Thanks for paying attention !

https://makeameme.org/meme/haben-sie-noch#google_vignette