Methods for Deep RL

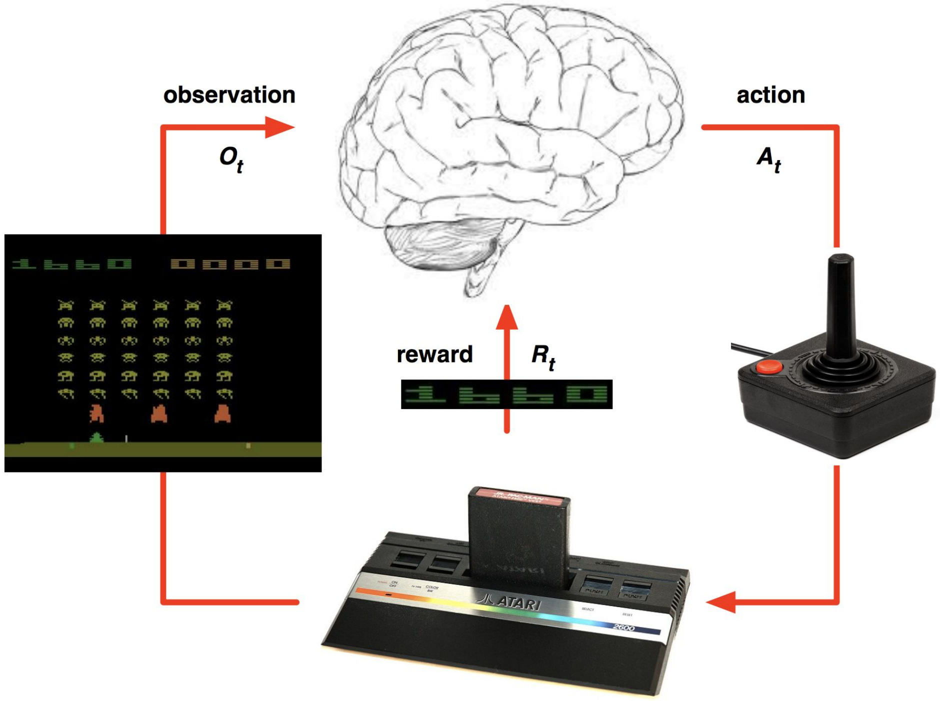
Gerben Meijer, Carl Beekhuizen
Ron van Bree, Maurice van Leeuwen

Agent observation raw pixels

Outdoor map overview

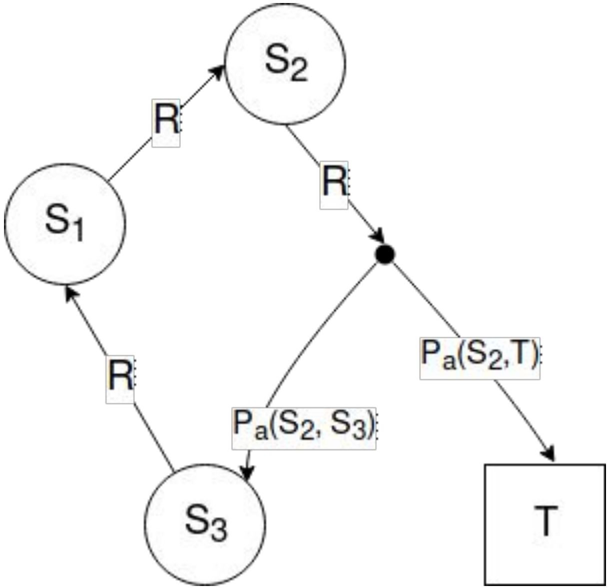
[1] Human-level performance in first-person multiplayer games with population-based deep reinforcement learning - DeepMind, 2018

Agent - Environment Interaction



[2] UCL Lectures on Reinforcement Learning - David Silver, 2015

MDP

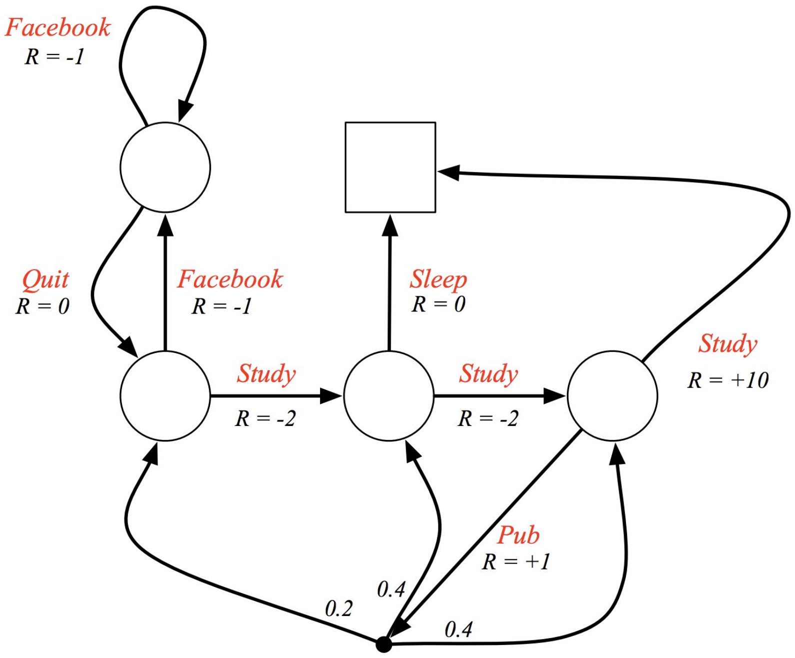


5-Tuple Containing:

- S - Set of states
- A - Set of actions
- $P_a(s,s')$ - Transition Probability
- R_s^a - Rewards for transition
- $\gamma \in [0,1]$ - discount factor

Markov assumption: The future is independent of the past, given the present

MDP



5-Tuple Containing:

- S - Set of states
- A - Set of actions
- $P_a(s,s')$ - Transition Probability
- R_s^a - Rewards for transition
- $\gamma \in [0,1]$ - discount factor

Markov assumption: The future is independent of the past, given the present

[2] UCL Lectures on Reinforcement Learning - David Silver, 2015

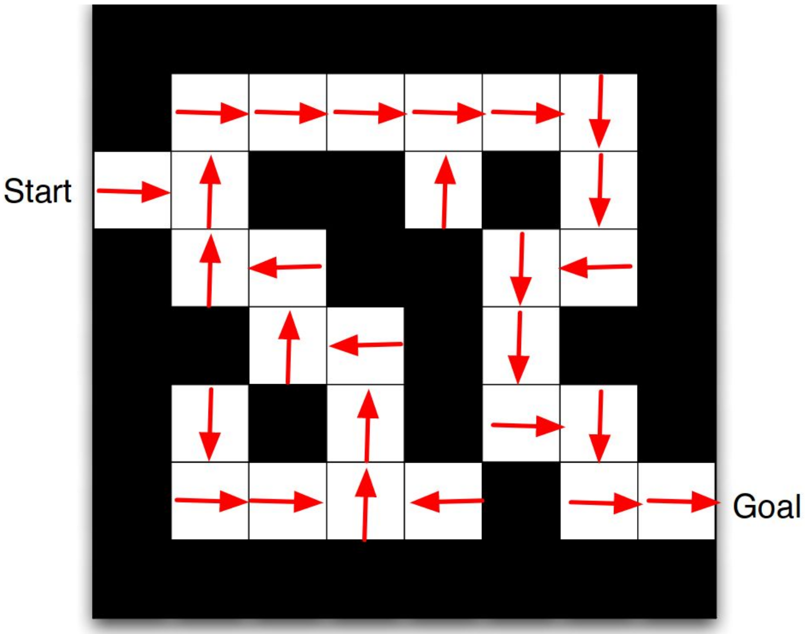
Goal of RL

$$\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\mid s_{t+1}\sim P(\cdot|s_t,a_t),\ a_t\sim\pi(\cdot|s_t),\ s_0\sim P(s)\right]$$

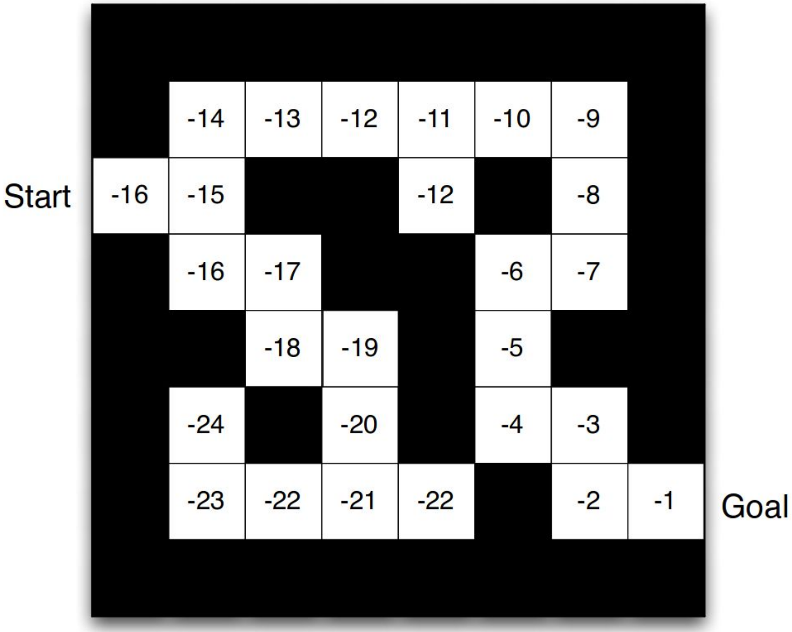
Perform the action that will give you the highest expected future reward.

Policy vs Value

Policy



Value



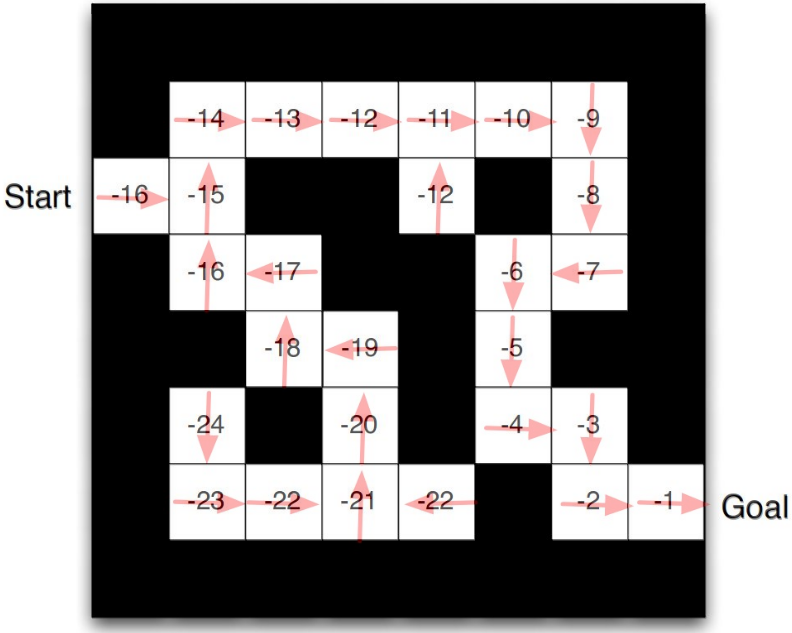
[2] UCL Lectures on Reinforcement Learning - David Silver, 2015

ε-Greedy Policy

Exploration/Exploitation tradeoff parameterised by ε

Decay ε over time

Value



[2] UCL Lectures on Reinforcement Learning - David Silver, 2015

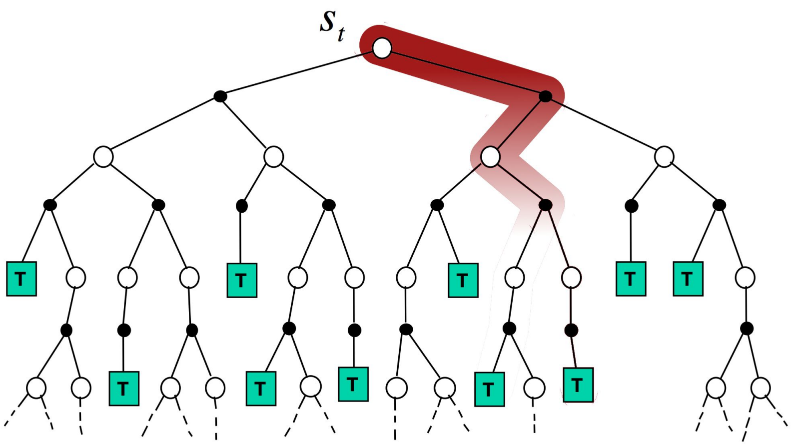
SARSA(λ)

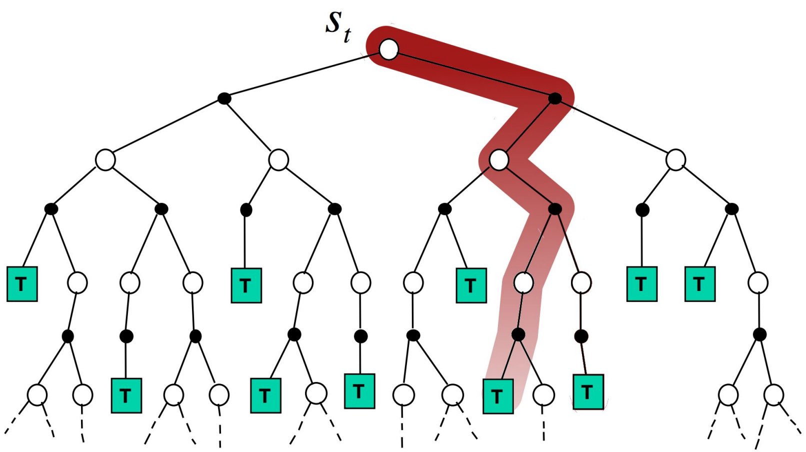
Smaller λ - Emphasis on recency

- Lower Variance
- Higher Bias
- λ = 0: TD(0)

Larger λ - Emphasis on future

- Higher Variance
- Lower Bias
- λ = 1: Monte Carlo





SARSA(λ)

We update our q-values:

$$q(s,a) \leftarrow q(s,a) + \alpha(q_t^\lambda - q(s,a))$$

where

$$q_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} q_t^{(n)}$$

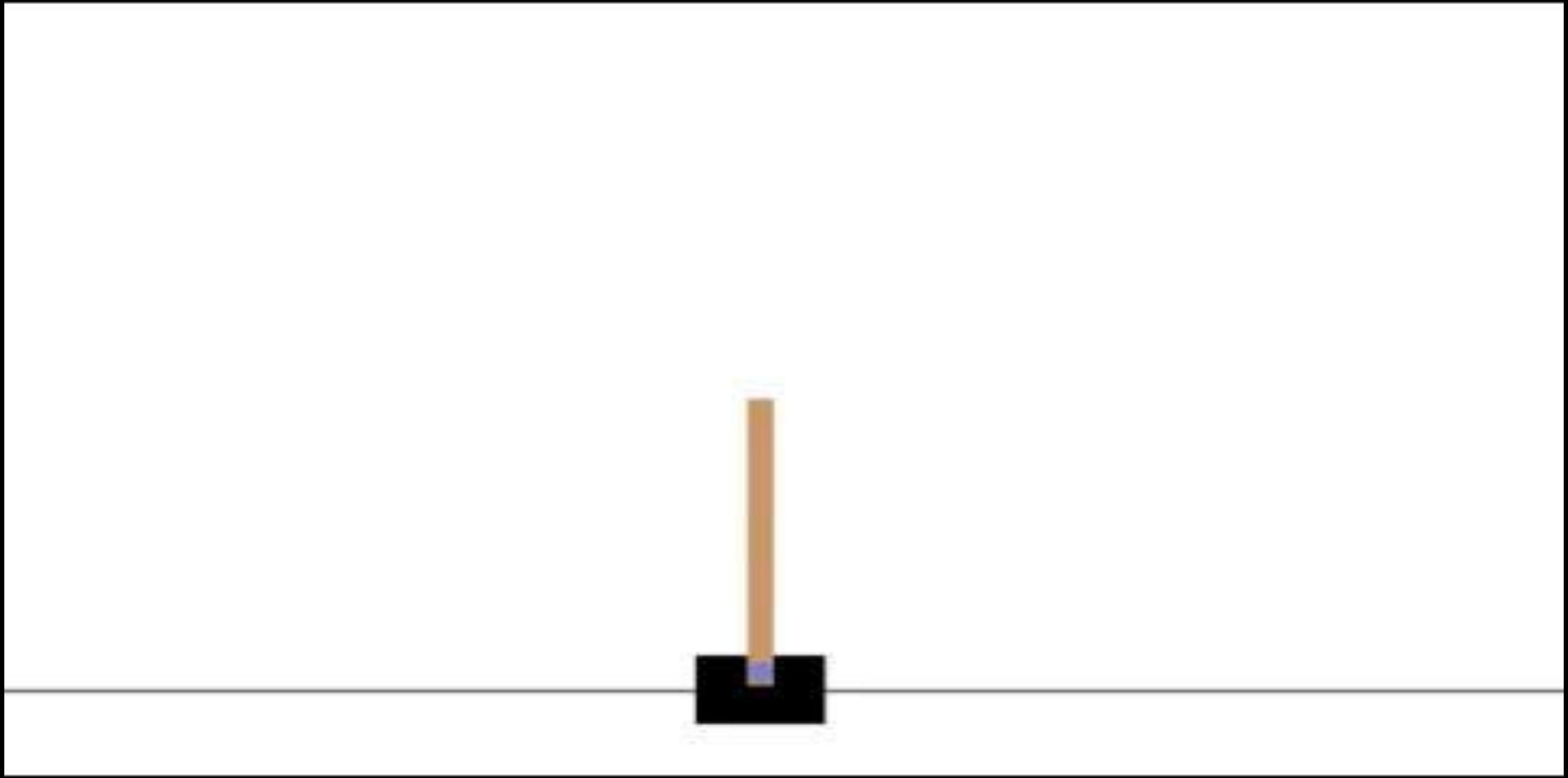
Cart Pole with Sarsa(λ)

Experiment Setup

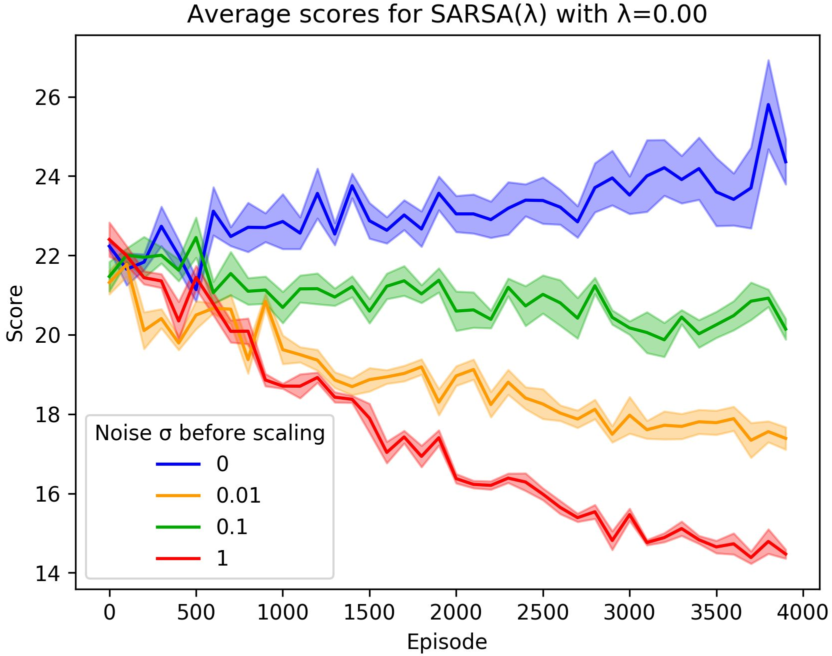
- Measurement noise σ
- Trace-decay λ
- ε-Greedy policy with decay

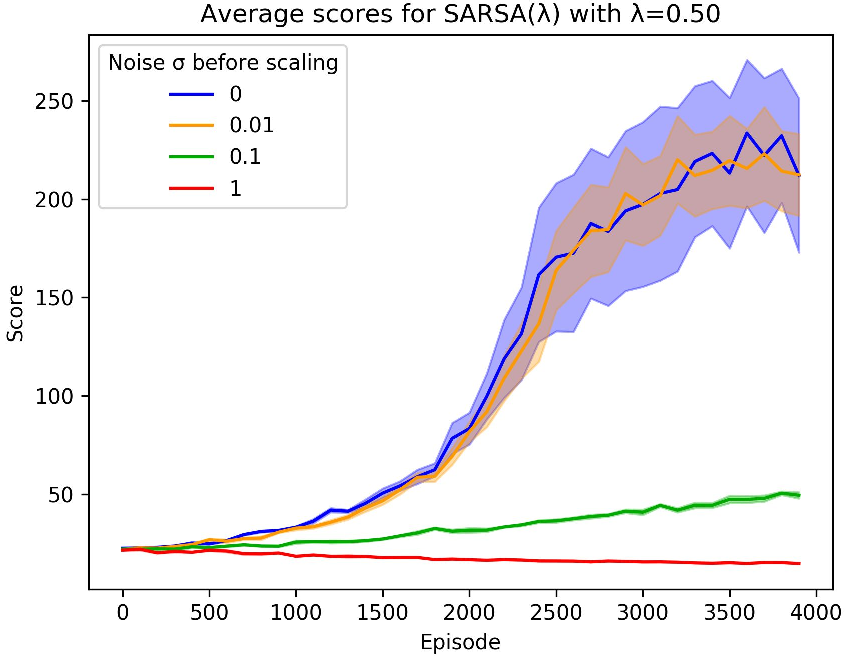
Expectations

- Slower learning and worse score at higher σ
- Quicker learning and greater stability at $\lambda \approx 0.9$

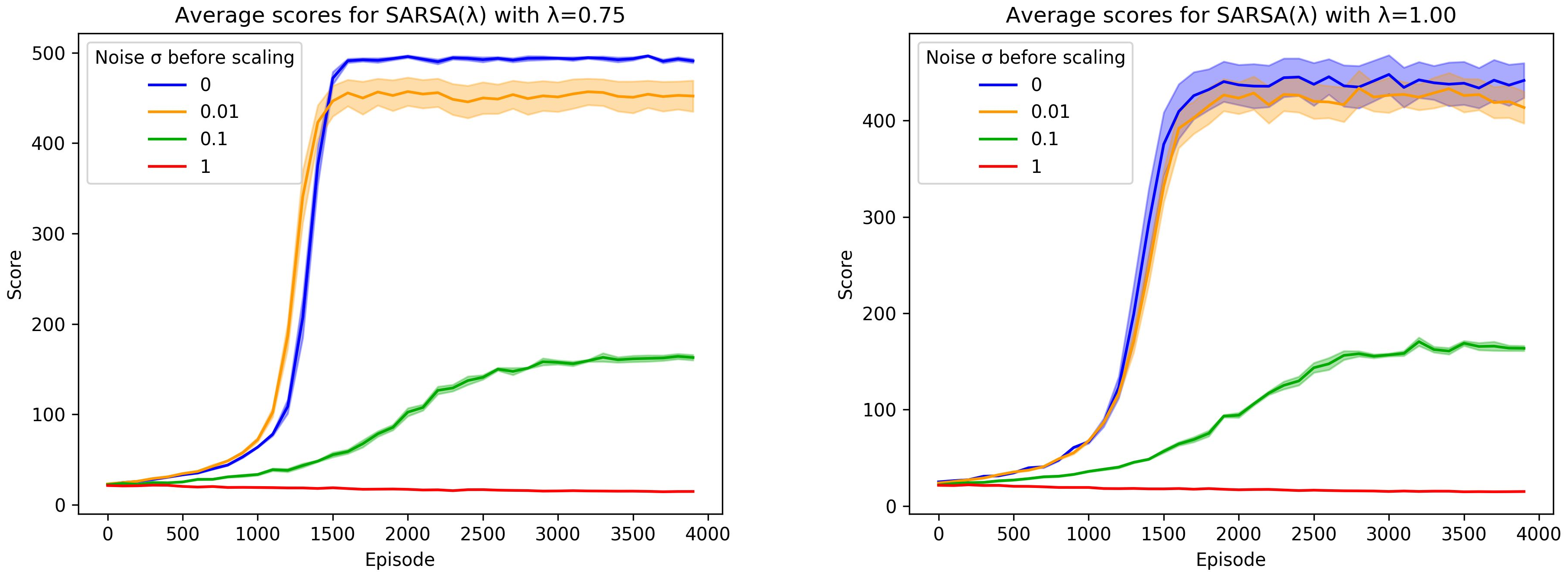


Limited learning when considering one step





Higher λ → greater variances



Deep Q-Learning (Mnih et. al. 2015)

Function Approximation with SGD

$$\hat{q}(s, a, \boldsymbol{w}) \approx q_\pi(s, a)$$

Replay Memory

Store (s a r s') $\rightarrow \mathscr{D}$, then sample $\mathscr{D}$ for learning

Fixed Q-Network

Cart Pole with Deep Sarsa(λ)

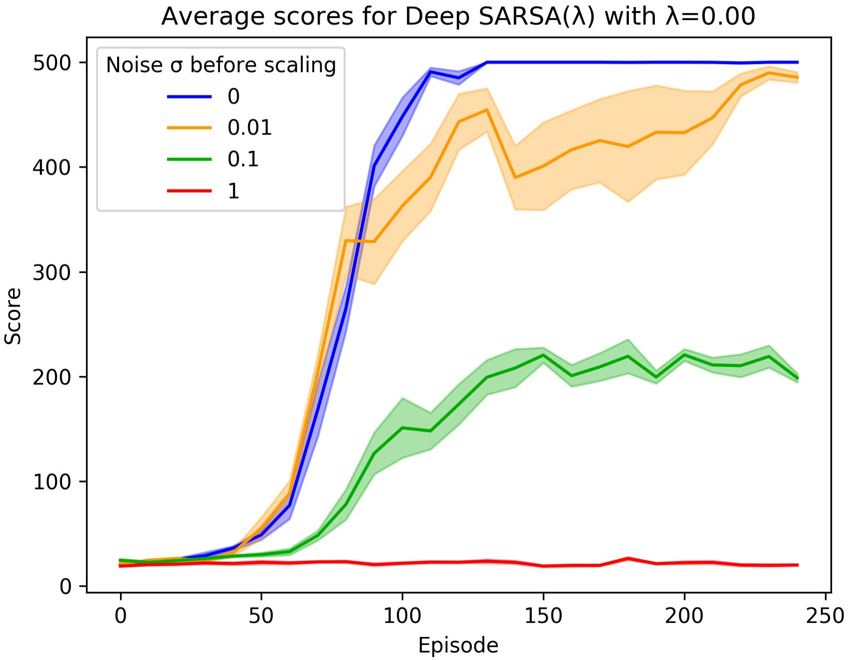
Experiment Setup

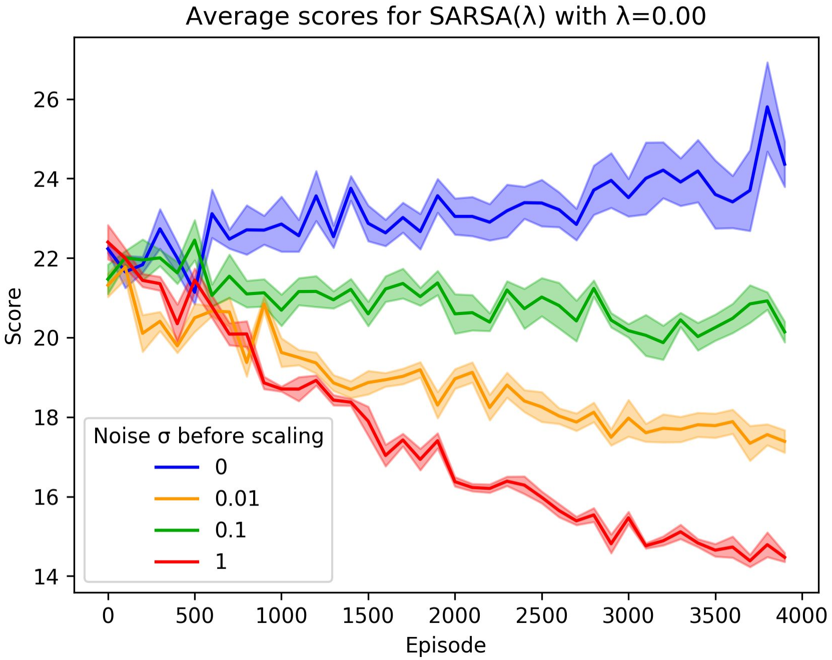
- Similar σ, λ and ε
- 150x50x2 FNN with ReLU
- Adam optimizer

Expectations

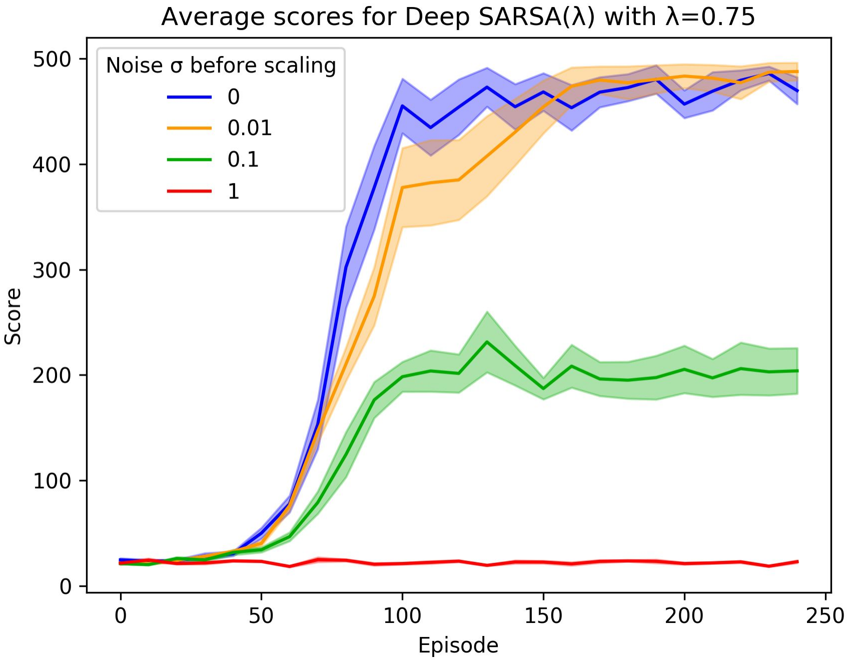
- Higher model complexity, require more samples
- But no aliasing due to discretization

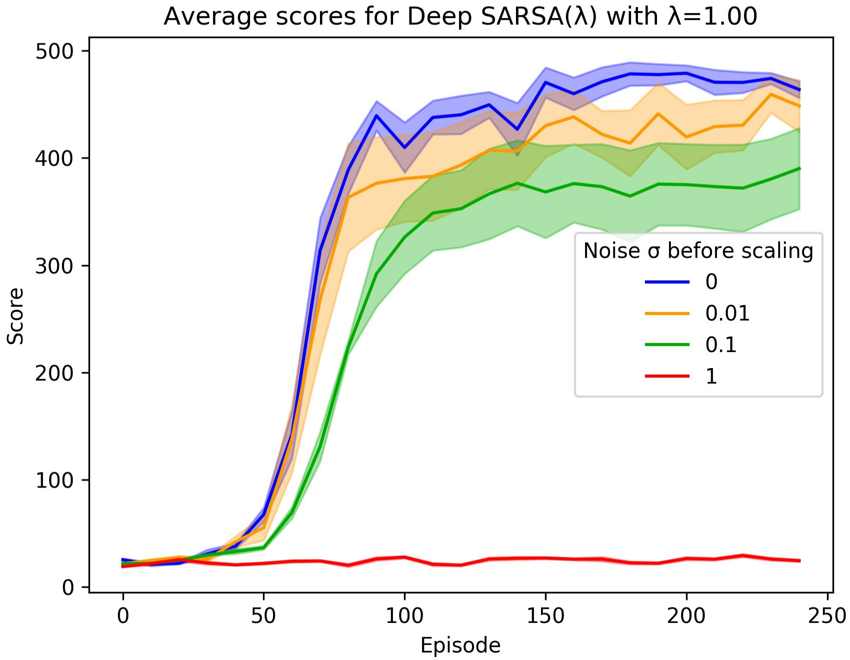
Deep Sarsa(λ=0) is a quick learner





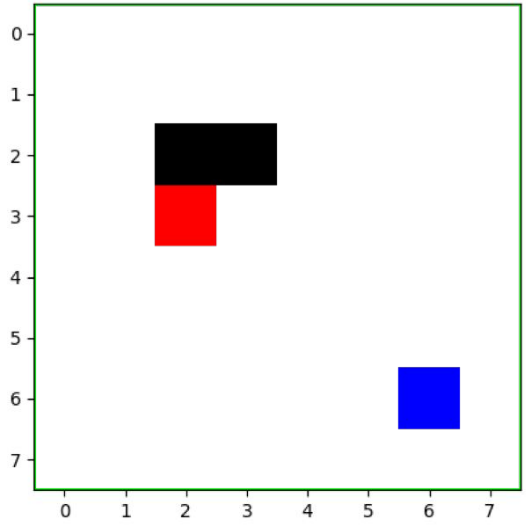
Larger λ help dampen effect of noise



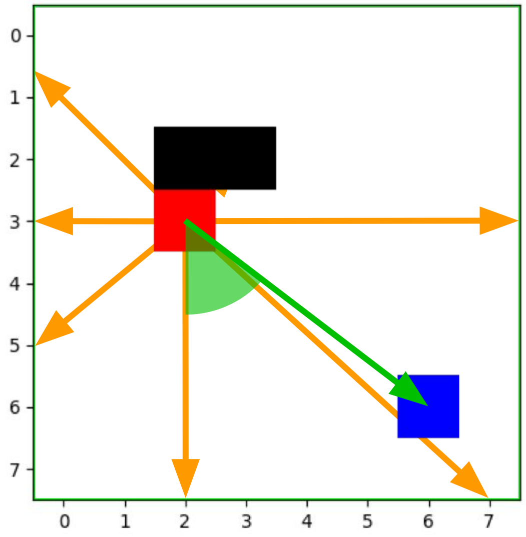


CNN vs NN: Environment

Raw Pixels - Convolutional Neural Network



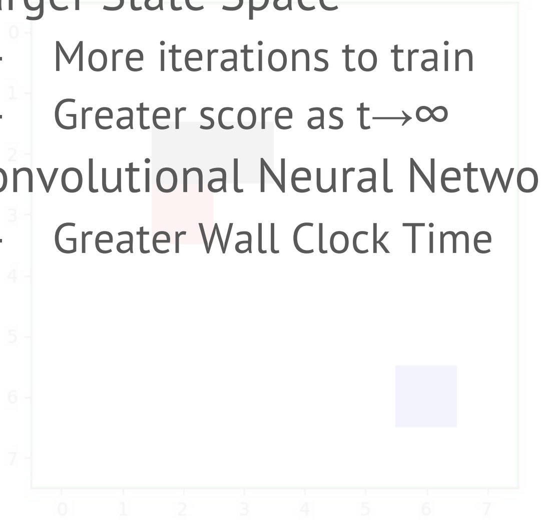
Feature Extracted - Deep Neural Network



CNN vs NN: Hypotheses

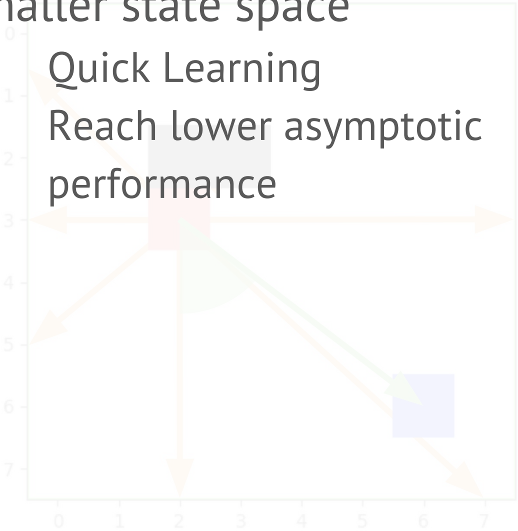
Raw Pixels - Convolutional NN

- Larger State Space
  - More iterations to train
  - Greater score as t→∞
- Convolutional Neural Network
  - Greater Wall Clock Time



Feature Extracted - Deep NN

- Smaller state space
  - Quick Learning
  - Reach lower asymptotic performance



CNN vs NN: Results & Discussion

Results

✓ CNN trains more slowly

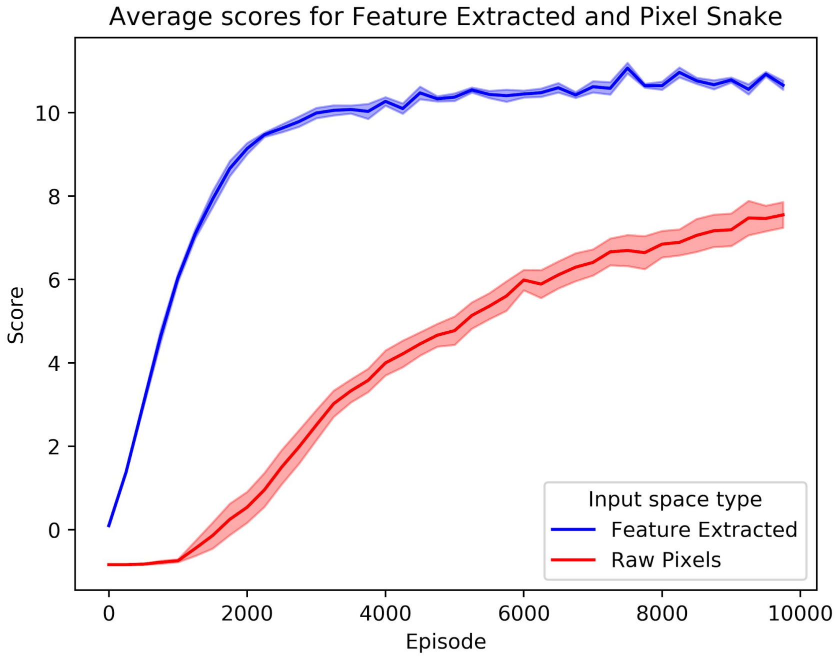
✓ NN reaches asymptotic performance

○ CNN achieves better performance

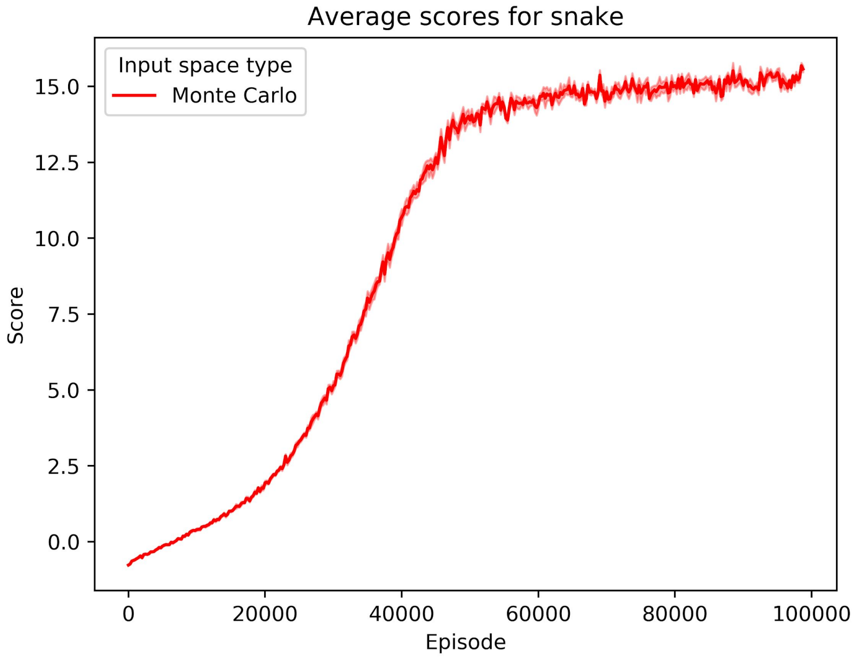
Discussion

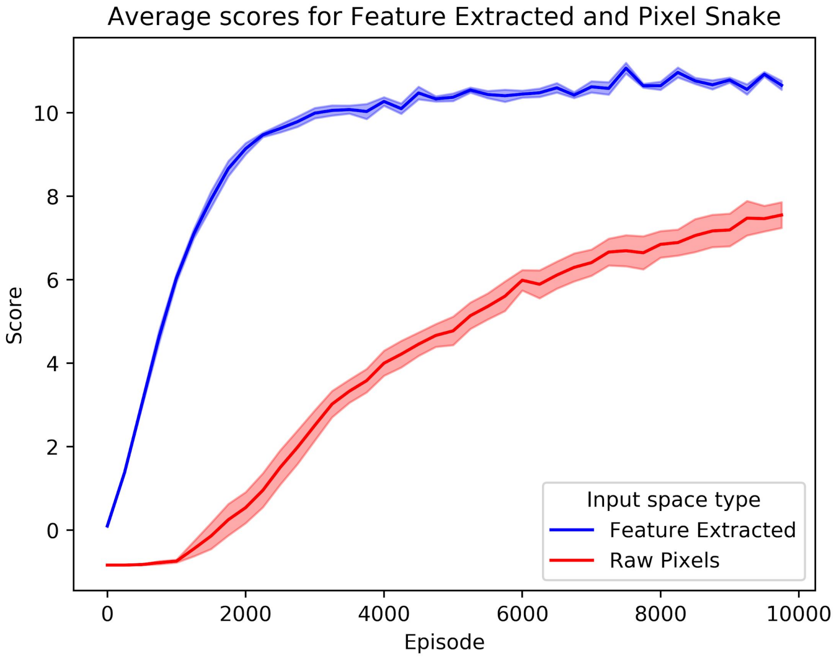
- ε should probably be increased

- Experiment needs to be run longer



CNN vs NN: Results & Discussion





Actor Critic

Up until now policy has only been derived from Q(s,a)

Actor Critic agents have 2 parts:

- The Critic learns the action-value function
- The Actor learns a policy based on “feedback” from the Critic

This allows for continuous action spaces

Actor Critic: implementation

Critic uses Sarsa(λ)

Actor (policy) updates by maximizing

$$L(\theta) = \mathbb{E}_{\pi(\cdot|s,\theta)}[A(s, a) - \alpha \log(\pi(a|s, \theta))]$$

Where A(s, a) is the advantage function Q(s, a) - V(s)

Entropy regularization

$$L(\theta) = \mathbb{E}_{\pi(\cdot|s,\theta)}[A(s,a) - \alpha \log(\pi(a|s,\theta))]$$

Where α is the entropy regularization factor.

Higher α → more randomness and therefore more exploration

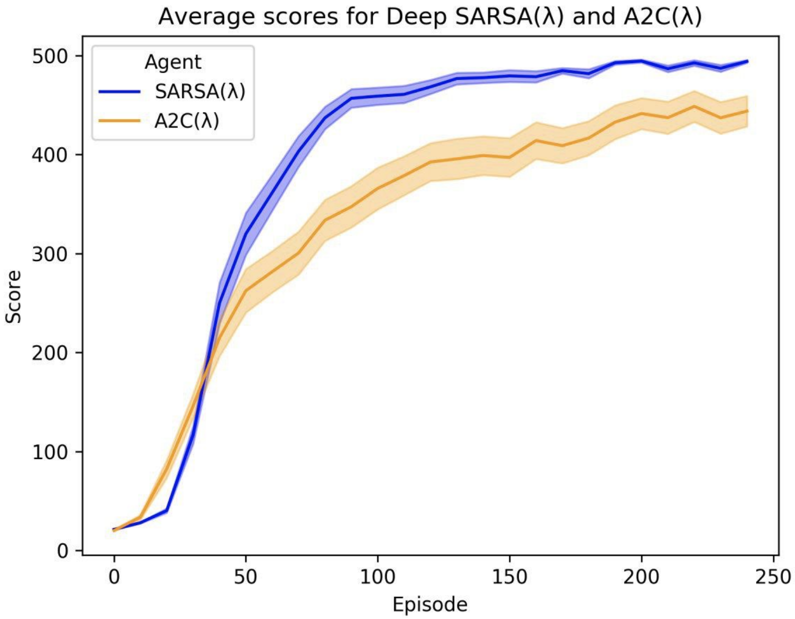
Lower α → more greedy policy

Actor Critic: Experiment

AC compared to Sarsa(λ), 25 runs each, on the cartpole environment

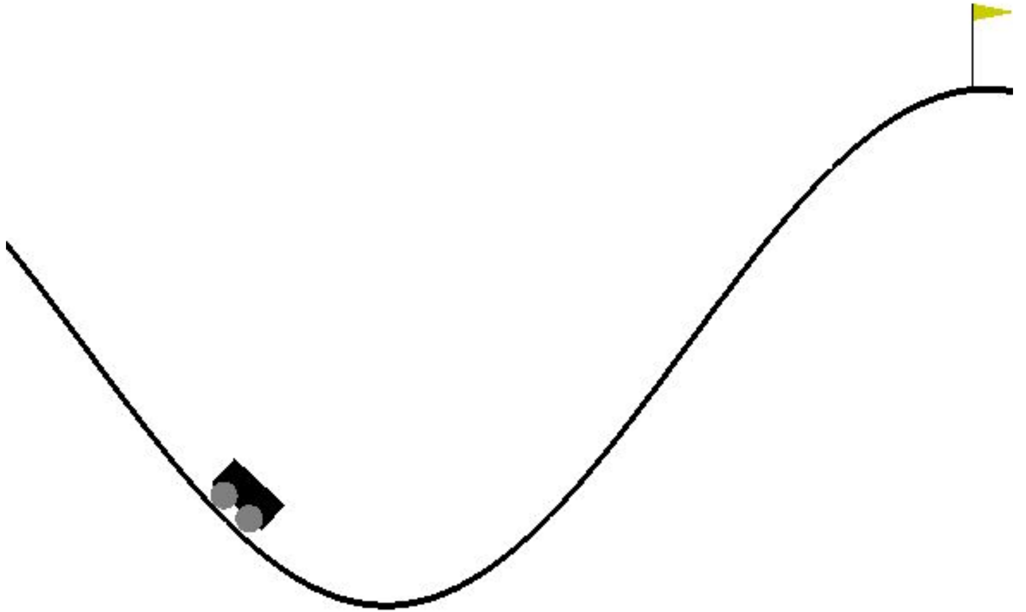
AC was slower:

- Many runs constant 500 score
- Some runs learn slow



A problematic environment: MountainCar

- The car needs to reach the flag
- Agent only sees position and velocity
- There is only a reward for reaching the flag
- Essentially, a random search is required to reach the flag



Sparse reward environments

Many environments are like MountainCar and only provide very sparse rewards

- Using random exploration, learning takes very long
- The usual solution is reward shaping, but reward shaping often changes the optimal policy and value functions
- How can we fix this problem?

Scheduled Auxiliary Control

SAC-X (Scheduled Auxiliary Control):

- Paper by DeepMind, published February 2018
- Uses predefined auxiliary tasks to aid learning
- A scheduler picks a new task multiple times per episode

Tasks

The main task and all auxiliary tasks together form the set of tasks called T

- All tasks have a separate MDP
- But state, observation and action space as well as transition dynamics are shared with the main task
- Reward functions are different for each task

Formal definition:

$$\mathcal{T} \in T = \mathcal{A} \cup \{\mathcal{M}\}$$

Intentions

Actor-Critic approach:

For each task $\mathcal{T}$, the agent learns a policy and action-value function.

$$Q_{\mathcal{T}}(s_t, a_t) = R_{\mathcal{T}}(s_t, a_t) + \gamma \mathbb{E}_{\pi_{\mathcal{T}}} \left[G_{\mathcal{T}}(\tau_{t+1:\infty}) \right]$$

$$\pi_{\mathcal{T}} = \pi(a_t | s_t, \mathcal{T})$$

The action-value function is trained using the RETRACE algorithm,

this enables off-policy learning on other tasks

Scheduling

SAC-X involves a scheduler to switch between tasks

- Tasks are switched every ξ steps
- The SAC-X paper proposes two scheduling strategies:
    - SAC- U, which schedules tasks uniformly at random
    - SAC-Q, which schedules based on the expected value of the return $G_{\mathcal{M}}(\mathcal{T}_{h:H})$

SAC-Q

Scheduler decides scheduled tasks based on past experience.

The probability of scheduling a task is approximated by the Boltzmann equation:

$$P_{\mathcal{S}}(\mathcal{T}|\mathcal{T}_{0:h-1};\mu) = \frac{\exp(\mathbb{E}_{P_{\mathcal{S}}}[G_{\mathcal{M}}(\mathcal{T}_{h:H})]/\mu)}{\sum_{\hat{\mathcal{T}}_{h:H}} \exp(\mathbb{E}_{P_{\mathcal{S}}}[G_{\mathcal{M}}(\hat{\mathcal{T}}_{h:H})]/\mu)}$$

Where μ controls the greediness.

SAC-Q, our implementation

The paper discussed 2 approaches of approximating $G_{\mathcal{M}}(\mathcal{T}_{h:H})$:

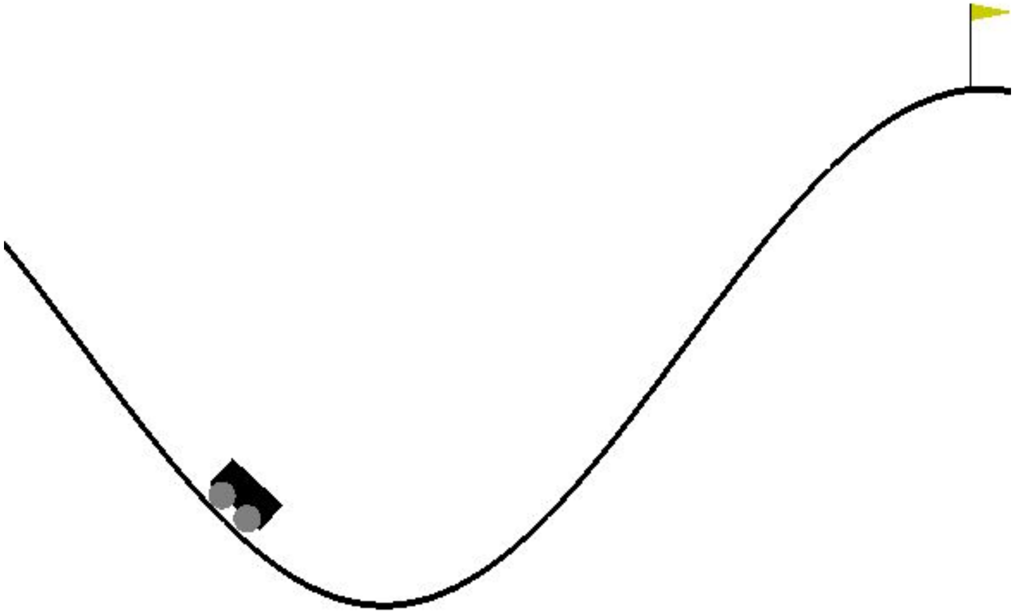
- Using a Q-table, where $\mathcal{T}_{0:h-1}$ is the state
- Calculating the MC return for the last M trajectories directly

In our implementation we use a Q-table with a constant learning rate

Evaluating SAC-Q : Environment

We evaluated SAC-Q on modified MountainCar:

- Terminates with 0 reward after 1000 steps
- Or 1 reward if the flag is reached



Evaluating SAC-Q : Tasks

The following auxiliary tasks were used:

GO_LEFT

$$R_{\texttt{GO\_LEFT}}(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 1 & \text{iff } \boldsymbol{a} = \texttt{ACTION\_LEFT} \\ 0 & \text{else} \end{cases}$$

GO_RIGHT

$$R_{\texttt{GO\_RIGHT}}(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 1 & \text{iff } \boldsymbol{a} = \texttt{ACTION\_RIGHT} \\ 0 & \text{else} \end{cases}$$

GO_FAST

$$R_{\texttt{GO\_FAST}}(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 1 & \text{iff } \boldsymbol{s}_1 \geq 0.03 \\ 0 & \text{else} \end{cases}$$

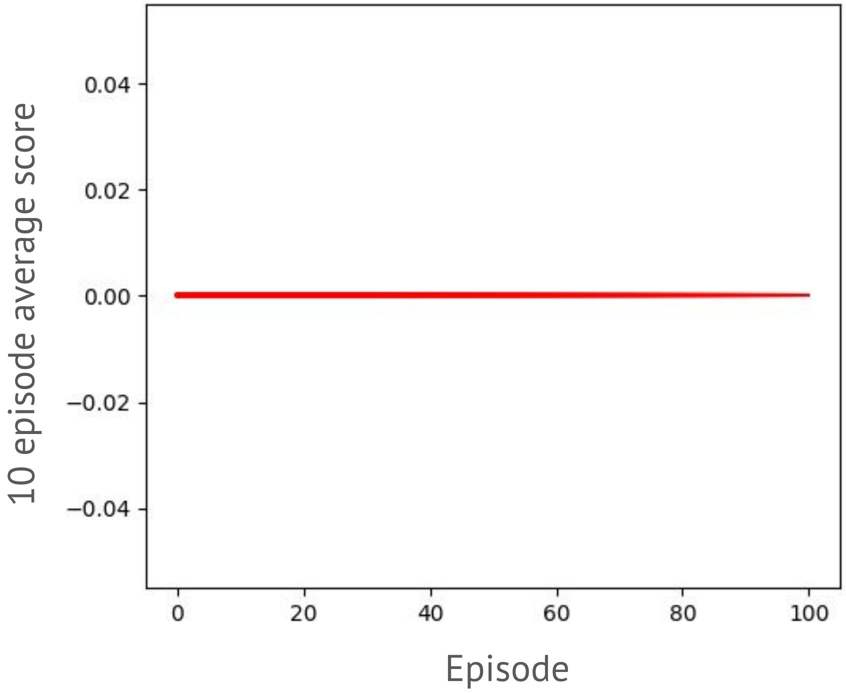
Evaluating SAC-Q: Setup

2 configurations were run on MountainCar for 100 episodes:

- Actor Critic (SAC-Q without auxiliary tasks)
- SAC-Q with auxiliary tasks

Actor Critic: Results

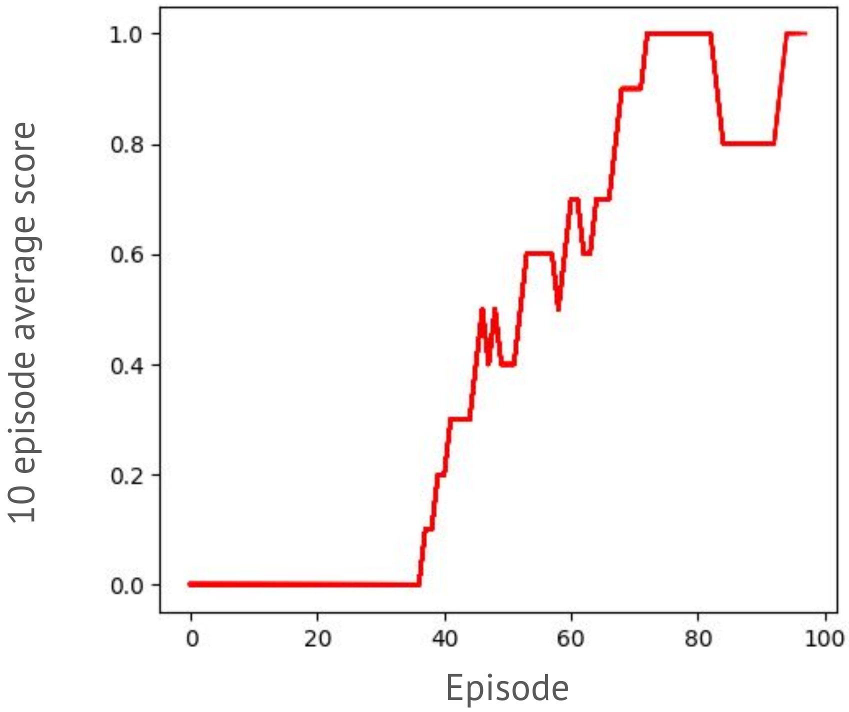
100 episode random search did not reach the goal and the agent did not learn at all



SAC-Q: Results

Agent managed to reach the flag reliably in only 100 episodes

But we found that main and GO_FAST policy were not used to reach the flag



Questions