Deep Reinforcement Learning with Memory

SERGEY BARTUNOV, HSE, MOSCOW

RL basics

Markov Decision Process

Markov Decision Process

Environment

Markov Decision Process

Agent

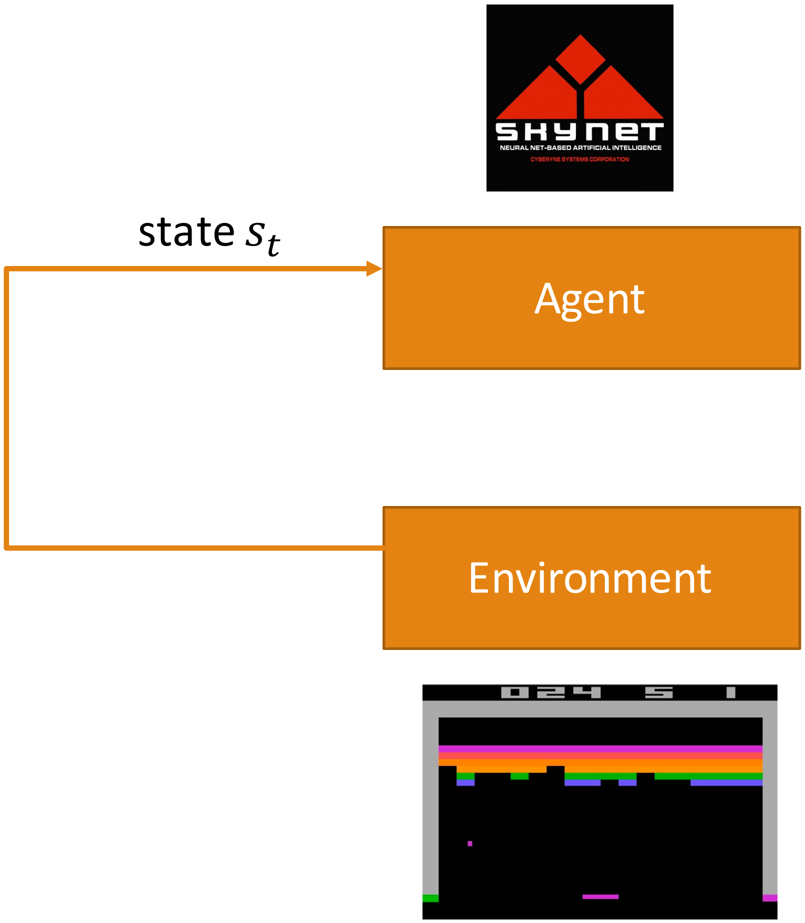
Environment

Markov Decision Process

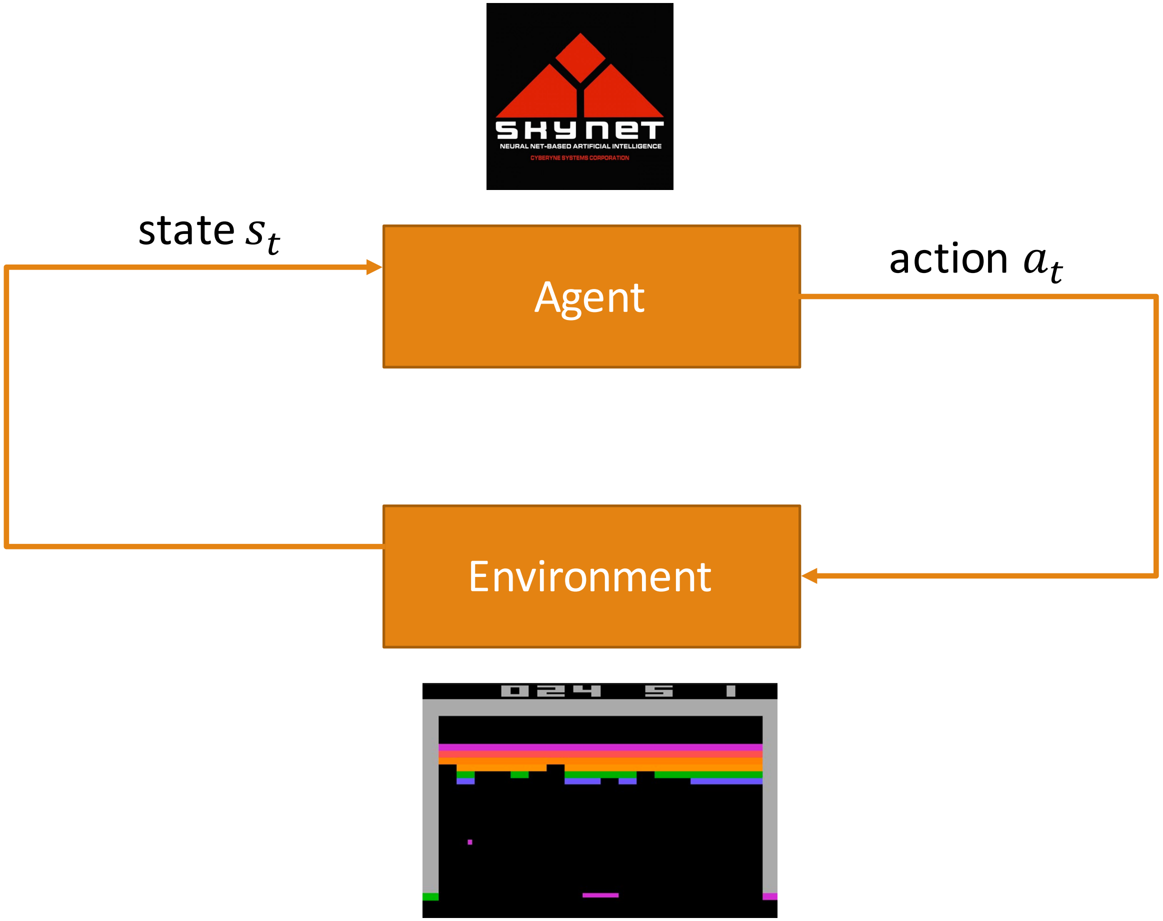
Agent

Environment

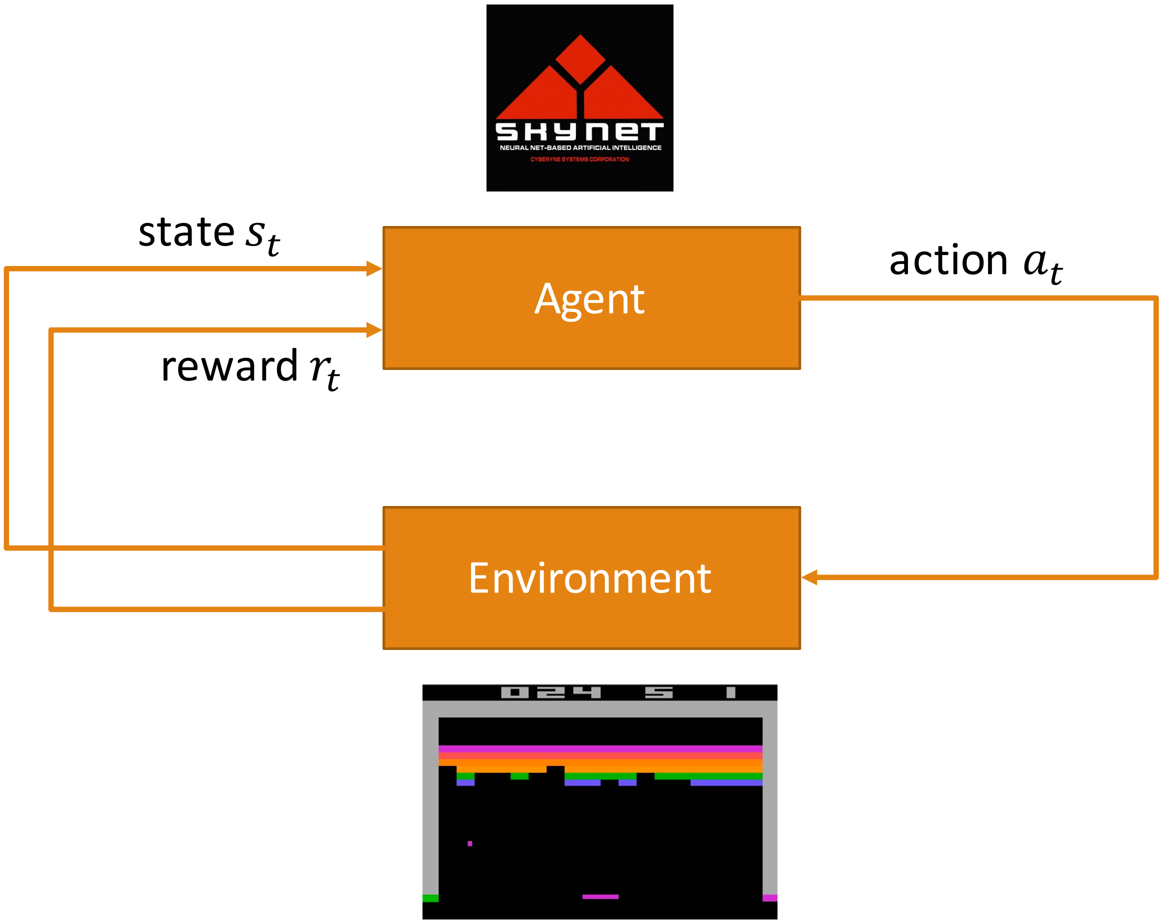
Markov Decision Process



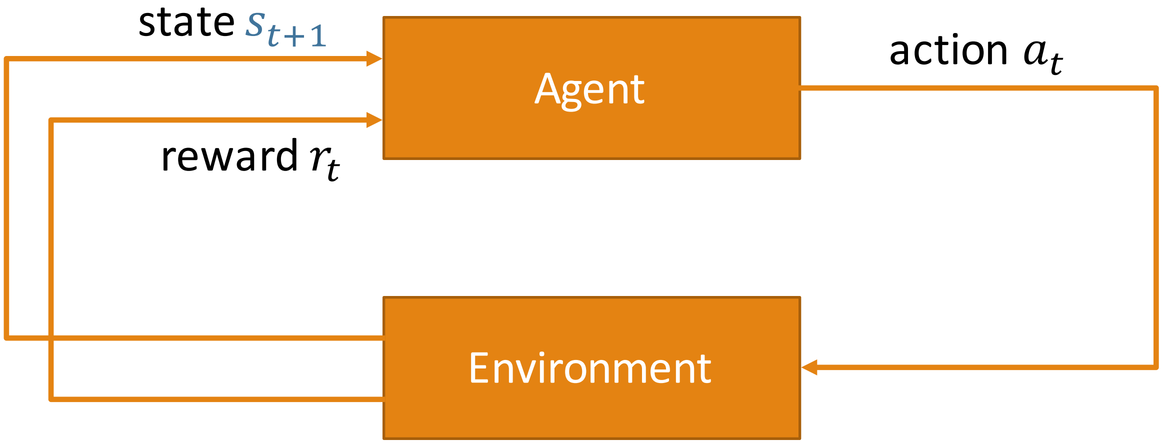
Markov Decision Process



Markov Decision Process

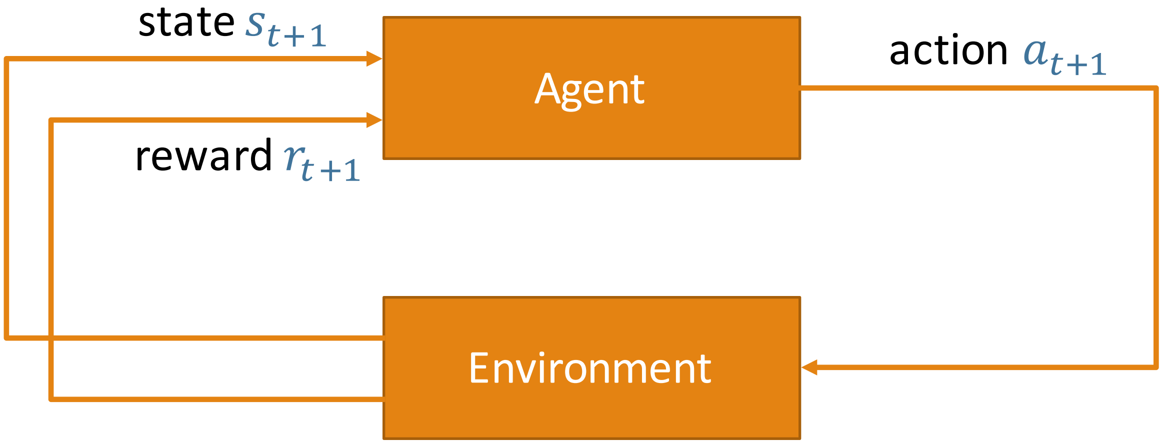


Markov Decision Process



Markov Decision Process

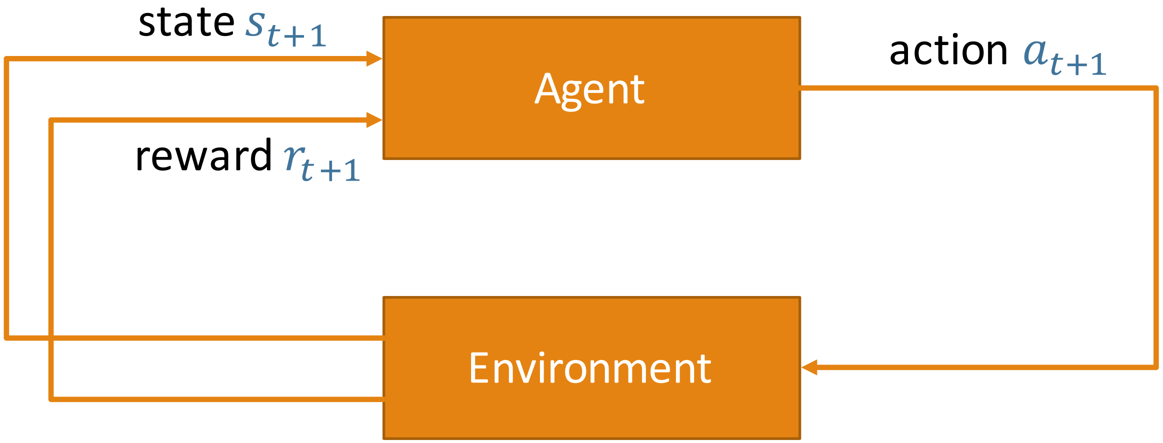
policy
$a_{t+1} \sim \pi(a_{t+1} | s_t)$



Markov Decision Process

state dynamics
$s_{t+1} \sim p(s_{t+1}|s_t)$

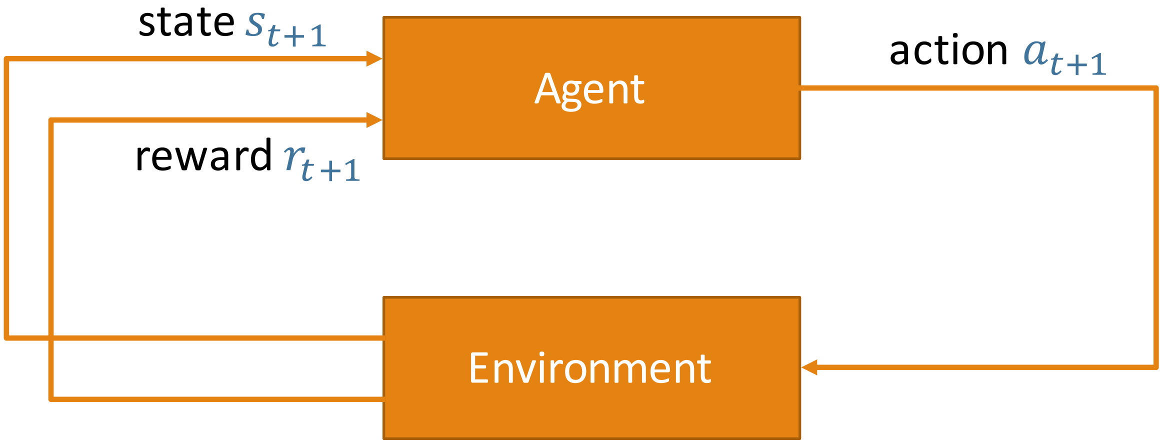
policy
$a_{t+1} \sim \pi(a_{t+1}|s_t)$



Markov Decision Process

state dynamics
$s_{t+1} \sim p(s_{t+1} | s_t)$

policy
$a_{t+1} \sim \pi(a_{t+1} | s_t)$



reward structure
$r_t = r(s_t, a_t, s_{t+1})$

Markov Decision Process

- Trajectory $s_0, a_0, s_1, r_0, a_1, s_2, r_1, \ldots$
  - $s_{t+1} \sim p(s_{t+1}|s_t)$
  - $a_t \sim \pi(a_t|s_t)$
  - $r_t = r(s_t, a_t, s_{t+1})$

Markov Decision Process

- Trajectory $s_0, a_0, s_1, r_0, a_1, s_2, r_1, \ldots$
  - $s_{t+1} \sim p(s_{t+1}|s_t)$
  - $a_t \sim \pi(a_t|s_t)$
  - $r_t = r(s_t, a_t, s_{t+1})$
- Expected return: $R^{\pi} = \mathbb{E}_{s_{1:T}, a_{1:T}}[\sum_{t=0}^{T} \gamma^t r_t]$

Markov Decision Process

- Trajectory $s_0, a_0, s_1, r_0, a_1, s_2, r_1, \ldots$
  - $s_{t+1} \sim p(s_{t+1}|s_t)$
  - $a_t \sim \pi(a_t|s_t)$
  - $r_t = r(s_t, a_t, s_{t+1})$
- Expected return: $R^\pi = \mathbb{E}_{s_{1:T}, a_{1:T}}[\sum_{t=0}^{T} \gamma^t r_t]$
- Value function: $V^\pi(s_t) = \mathbb{E}_{s_{t:T}, a_{t:T}}[\sum_{l=0}^{T} \gamma^{t+l} r_{t+l}]$

Markov Decision Process

- Trajectory $s_0, a_0, s_1, r_0, a_1, s_2, r_1, \ldots$
  - $s_{t+1} \sim p(s_{t+1}|s_t)$
  - $a_t \sim \pi(a_t|s_t)$
  - $r_t = r(s_t, a_t, s_{t+1})$
- Expected return: $R^\pi = \mathbb{E}_{s_{1:T}, a_{1:T}}[\sum_{t=0}^{T} \gamma^t r_t]$
- Value function: $V^\pi(s_t) = \mathbb{E}_{s_{t:T}, a_{t:T}}[\sum_{l=0}^{T} \gamma^l r_{t+l}]$
- Action-state function: $Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t:T}, a_{t+1:T}}[\sum_{l=0}^{T} \gamma^l r_{t+l}]$
  - $V^\pi(s_t) = \mathbb{E}_{a_t} Q^\pi(s_t, a_t)$

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

How to solve MDP

1. Start from some policy π

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

How to solve MDP

1. Start from some policy π
2. Evaluate it to obtain Q^{π}

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

How to solve MDP

$$\min_{Q} \mathbb{E} \left\| Q(s_t, a_t) - r_t - \gamma Q(s_{t+1}, a_{t+1}) \right\|^2$$

1. Start from some policy π
2. Evaluate it to obtain Q^{π}

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

How to solve MDP

$$\min_{Q} \mathbb{E} \left\| Q(s_t, a_t) - r_t - \gamma Q(s_{t+1}, a_{t+1}) \right\|^2$$

1. Start from some policy π
2. Evaluate it to obtain Q^{π}
3. Improve π to π'

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

How to solve MDP

$$\min_Q \mathbb{E} \left\| Q(s_t, a_t) - r_t - \gamma Q(s_{t+1}, a_{t+1}) \right\|^2$$

1. Start from some policy π
2. Evaluate it to obtain Q^{π}
3. Improve π to π' : $\pi'(a|s) = \arg\max_a Q^{\pi}(s, a)$

Bellman equation and Value iteration

- $V^{\pi}(s_t) = r_t + \gamma \mathbb{E}_{s_{t+1}} [V^{\pi}(s_{t+1})]$
- $Q^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q^{\pi}(s_{t+1}, a_{t+1})]$

How to solve MDP

$$\min_Q \mathbb{E} \left\| Q(s_t, a_t) - r_t - \gamma Q(s_{t+1}, a_{t+1}) \right\|^2$$

1. Start from some policy π
2. Evaluate it to obtain Q^{π}
3. Improve π to π' : $\pi'(a|s) = \arg\max_a Q^{\pi}(s, a)$
4. Repeat until convergence

Policy gradients

- Consider a parametric policy $\pi(a|s;\theta)$

Policy gradients

- Consider a parametric policy $\pi(a|s;\theta)$
- Our goal is to maximize the expected return:

$$R^\pi = \mathbb{E}_{s_{0:T}, a_{0:T}}[\sum_{t=0}^{T} \gamma^t r_t] \to \max_\theta$$

Policy gradients

- Consider a parametric policy $\pi(a|s;\theta)$
- Our goal is to maximize the expected return:

$$R^\pi = \mathbb{E}_{s_{0:T},a_{0:T}}[\sum_{t=0}^{T} \gamma^t r_t] \rightarrow \max_\theta$$

- Policy gradient can be written as follows:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T},a_{0:T}} \left[\sum_{t=0}^{T} \gamma^t \sum_{l=0}^{T-t} \gamma^l r_{t+l} \nabla_\theta \log \pi(a_t|s_t;\theta) \right]$$

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T},a_{0:T}}\left[\sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta)\right]$$

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T},a_{0:T}}\left[\sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta)\right]$$

- Stochastic estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta), \quad \mathbb{E}\left[\widetilde{\nabla}_\theta R^\pi\right] = \nabla_\theta R^\pi$$

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T},a_{0:T}}\left[\sum_{t=0}^{T}\gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta)\right]$$

- Stochastic estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T}\gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta), \quad \mathbb{E}\left[\widetilde{\nabla}_\theta R^\pi\right] = \nabla_\theta R^\pi$$

- Ψ_t may have very different form:

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T},a_{0:T}}\left[\sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta)\right]$$

- Stochastic estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|s_t;\theta), \quad \mathbb{E}\left[\widetilde{\nabla}_\theta R^\pi\right] = \nabla_\theta R^\pi$$

- Ψ_t may have very different form:
  - $\sum_{k=0}^{T} \gamma^{k-t} r_k$

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T}, a_{0:T}} \left[\sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | s_t; \theta) \right]$$

- Stochastic estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | s_t; \theta), \quad \mathbb{E}\left[\widetilde{\nabla}_\theta R^\pi \right] = \nabla_\theta R^\pi$$

- Ψ_t may have very different form:
  - $\sum_{k=0}^{T} \gamma^{k-t} r_k$
  - $\sum_{l=0}^{T-t} \gamma^l r_{t+l}$

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T}, a_{0:T}} \left[\sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | s_t; \theta) \right]$$

- Stochastic estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | s_t; \theta), \quad \mathbb{E}\left[\widetilde{\nabla}_\theta R^\pi \right] = \nabla_\theta R^\pi$$

- Ψ_t may have very different form:
  - $\sum_{k=0}^{T} \gamma^{k-t} r_k$
  - $\sum_{l=0}^{T-t} \gamma^l r_{t+l}$
  - $\sum_{l=0}^{T-t} \gamma^l r_{t+l} - b(s_t)$, where $b(s_t)$ is a baseline, often $b(s_t) = V^\pi(s_t)$

Stochastic approximation

- Full gradient:

$$\nabla_\theta R^\pi = \mathbb{E}_{s_{0:T}, a_{0:T}} \left[\sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | s_t; \theta) \right]$$

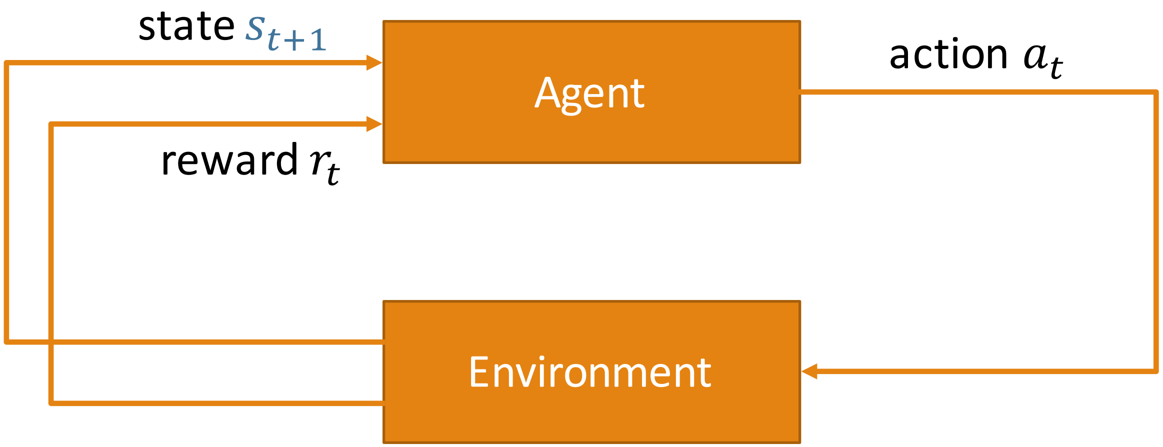
- Stochastic estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | s_t; \theta), \quad \mathbb{E}\left[\widetilde{\nabla}_\theta R^\pi \right] = \nabla_\theta R^\pi$$

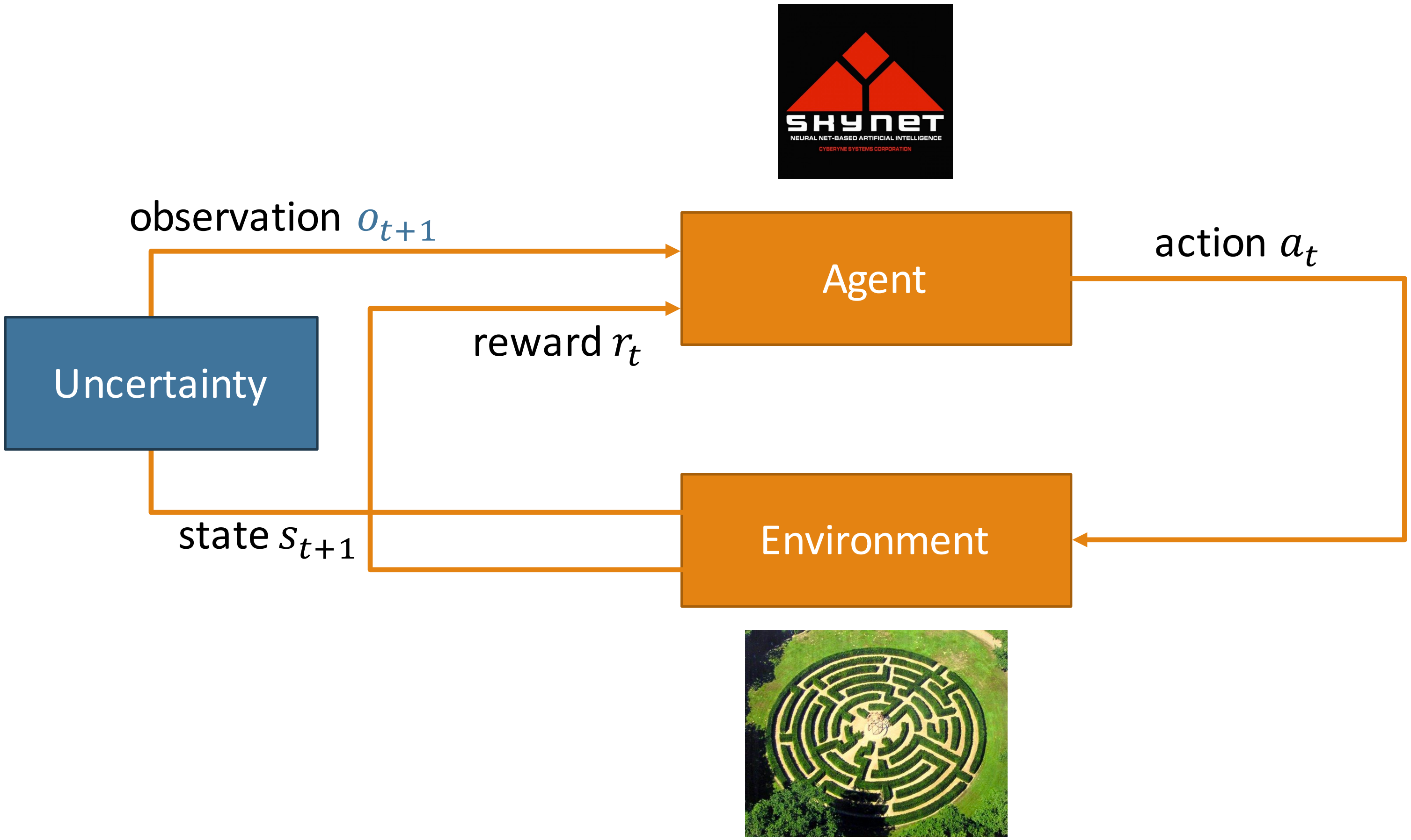
- Ψ_t may have very different form:
  - $\sum_{k=0}^{T} \gamma^{k-t} r_k$
  - $\sum_{l=0}^{T-t} \gamma^l r_{t+l}$
  - $\sum_{l=0}^{T-t} \gamma^l r_{t+l} - b(s_t)$, where $b(s_t)$ is a baseline, often $b(s_t) = V^\pi(s_t)$
  - $Q^\pi(s_t, a_t) - V^\pi(s_t)$ – advantage function, usually intractable

Memory problems

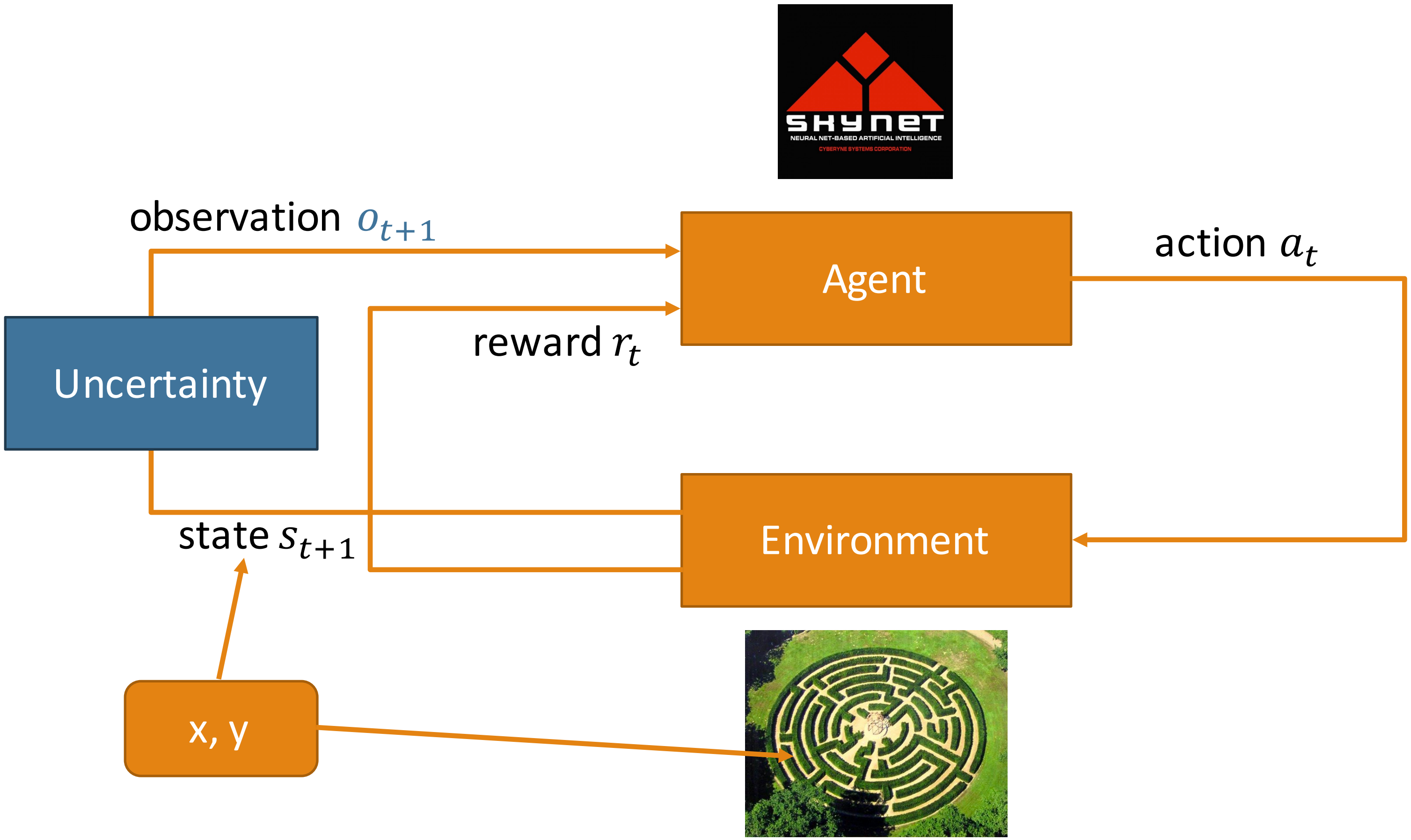
Markov Decision Process



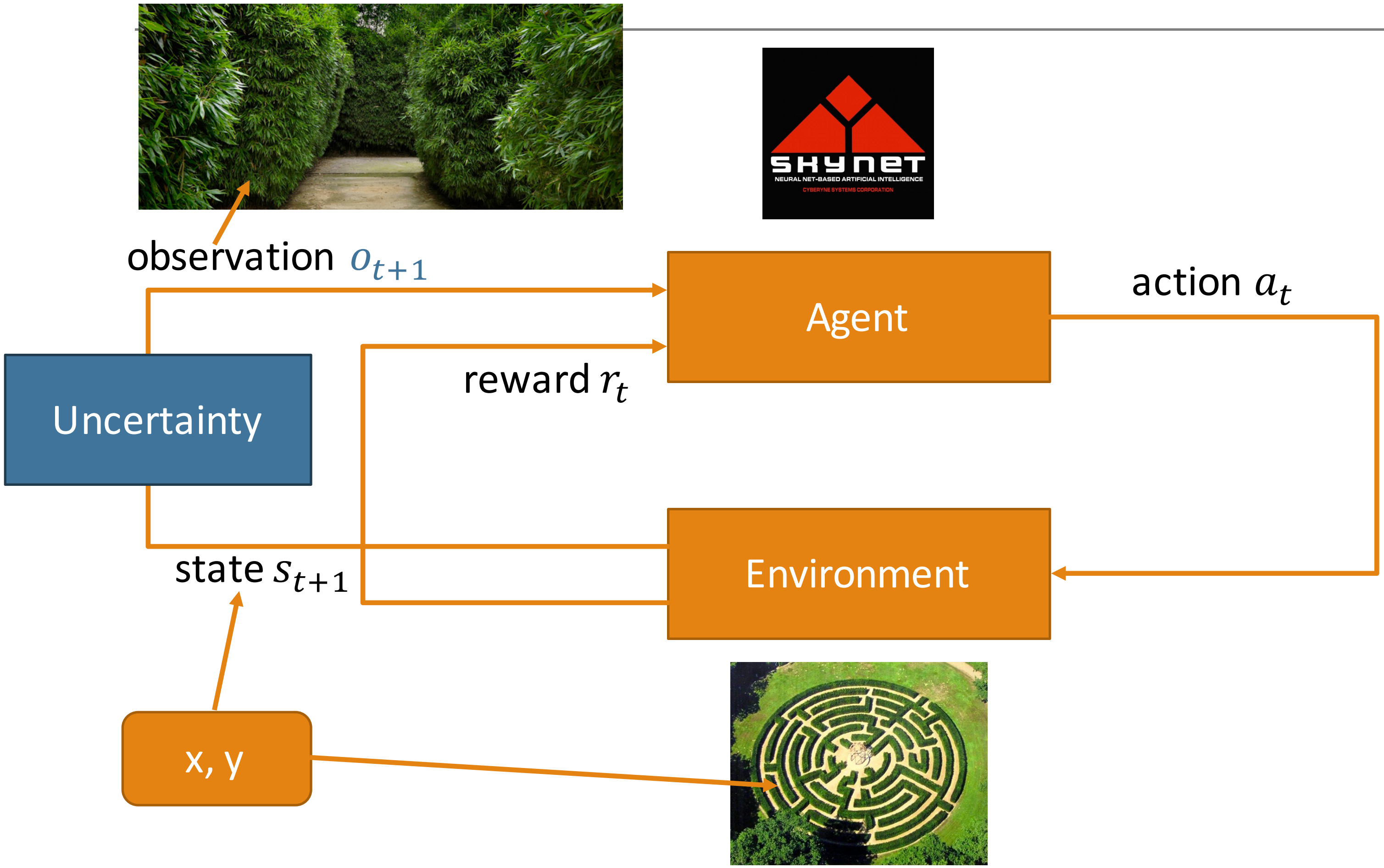
Partially-observable Markov Decision Process



Partially-observable Markov Decision Process



Partially-observable Markov Decision Process



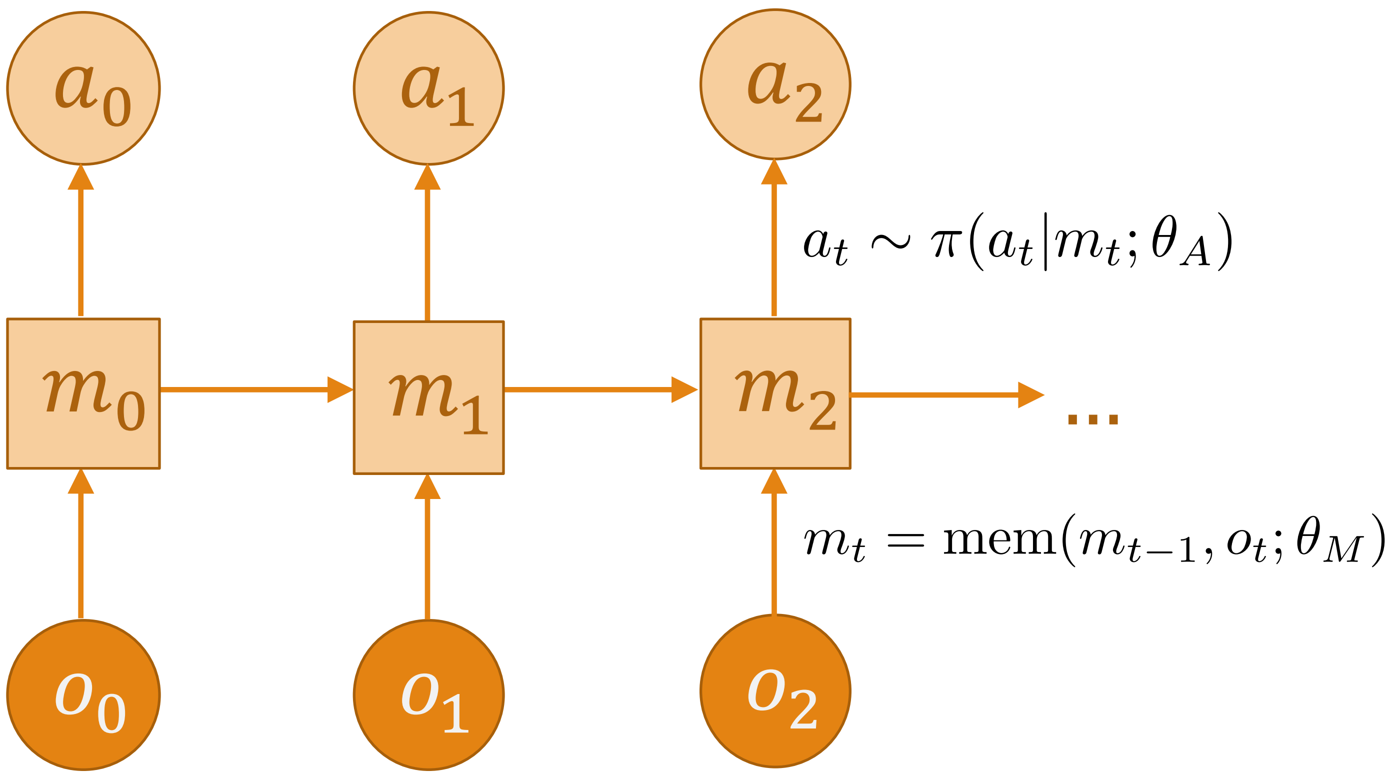
PO-MDP and Memory

- Trajectory $s_0, o_0, a_0, s_1, r_0, o_1, a_1, s_2, r_1, o_2 \ldots$
  - $s_{t+1} \sim p(s_{t+1}|s_t)$
  - $o_t \sim p(o_t|s_t)$
  - $a_t \sim \pi(a_t|o_t)$
  - $r_t = r(s_t, a_t, s_{t+1})$

PO-MDP and Memory

- Trajectory $s_0, o_0, a_0, s_1, r_0, o_1, a_1, s_2, r_1, o_2 \ldots$
  - $s_{t+1} \sim p(s_{t+1}|s_t)$
  - $o_t \sim p(o_t|s_t)$
  - $a_t \sim \pi(a_t|o_t)$
  - $r_t = r(s_t, a_t, s_{t+1})$
- Memory assumption:
  - there exists a memory $m_t = \mathrm{mem}(m_{t-1}, o_{t-1})$
  - such that $s_t \approx f(o_t, m_t)$

LSTM Agent



a_0
a_1
a_2
$a_t \sim \pi(a_t | m_t; \theta_A)$
m_0
m_1
m_2
...
$m_t = \mathrm{mem}(m_{t-1}, o_t; \theta_M)$
o_0
o_1
o_2

Recurrent policy gradients

- Stochastic gradient estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|m_t;\theta_A), \quad m_t = \mathrm{mem}(m_{t-1}, o_t; \theta_M)$$

  - $\Psi_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$

Recurrent policy gradients

- Stochastic gradient estimate:

$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|m_t;\theta_A), \quad m_t = \mathrm{mem}(m_{t-1}, o_t; \theta_M)$$

  - $\Psi_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$

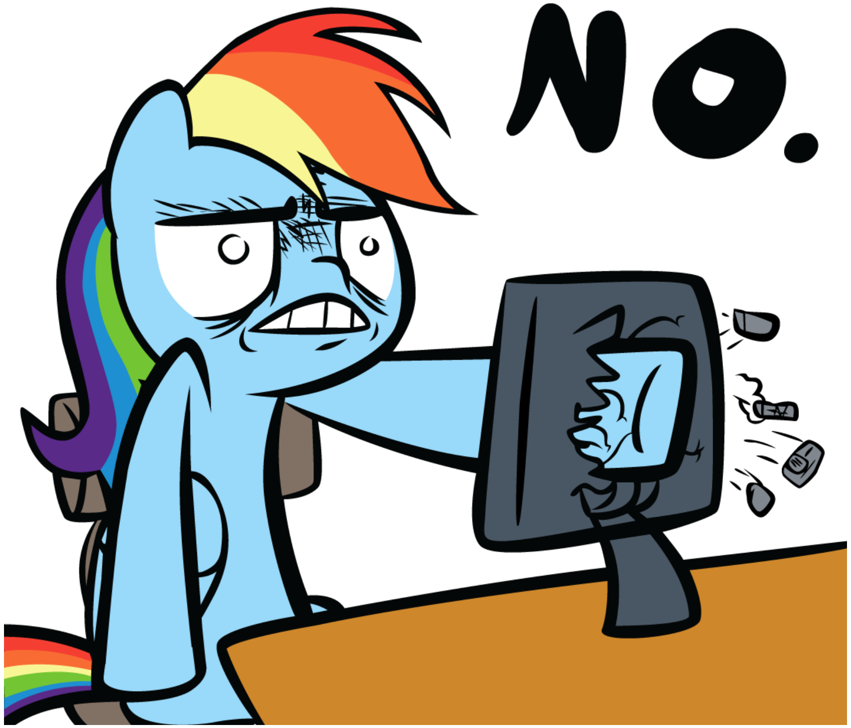
- Backpropagation through time:

$$\widetilde{\nabla}_{\theta_M} R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \frac{\partial \log \pi(a_t|m_t;\theta_A)}{\partial m_t} G_t$$

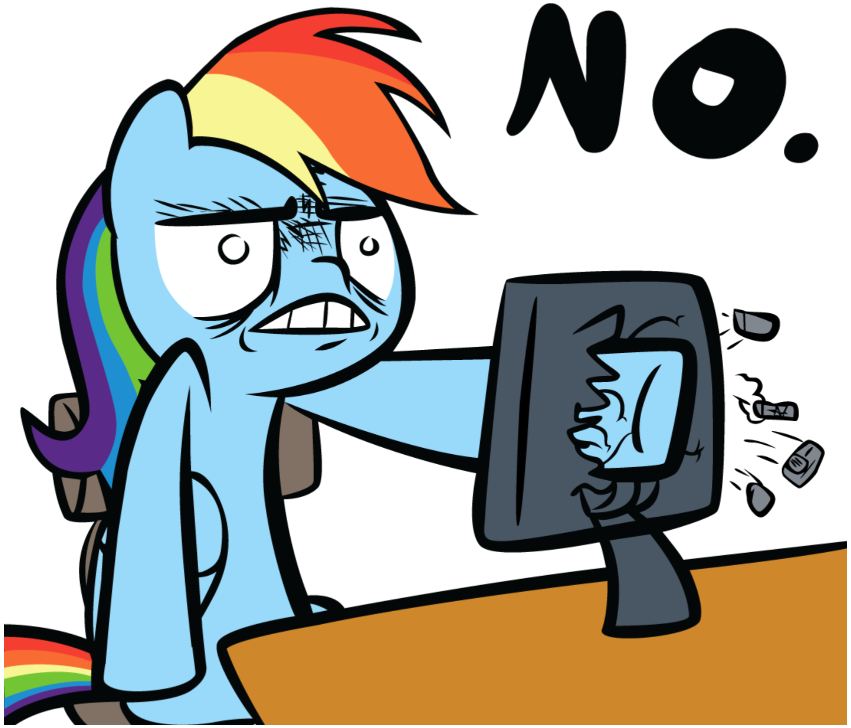
$$G_t = \frac{\partial m_t}{\partial \theta_M} + \frac{\partial m_t}{\partial m_{t+1}} G_{t+1}$$

Will this work out of box?

Will this work out of box?



Will this work out of box?



- High variance of gradients
- Usual problems with backpropagation through time
  - Exploding / vanishing gradients
  - Cannot work in a continuous settings

Variance reduction

- Stochastic gradient estimate:

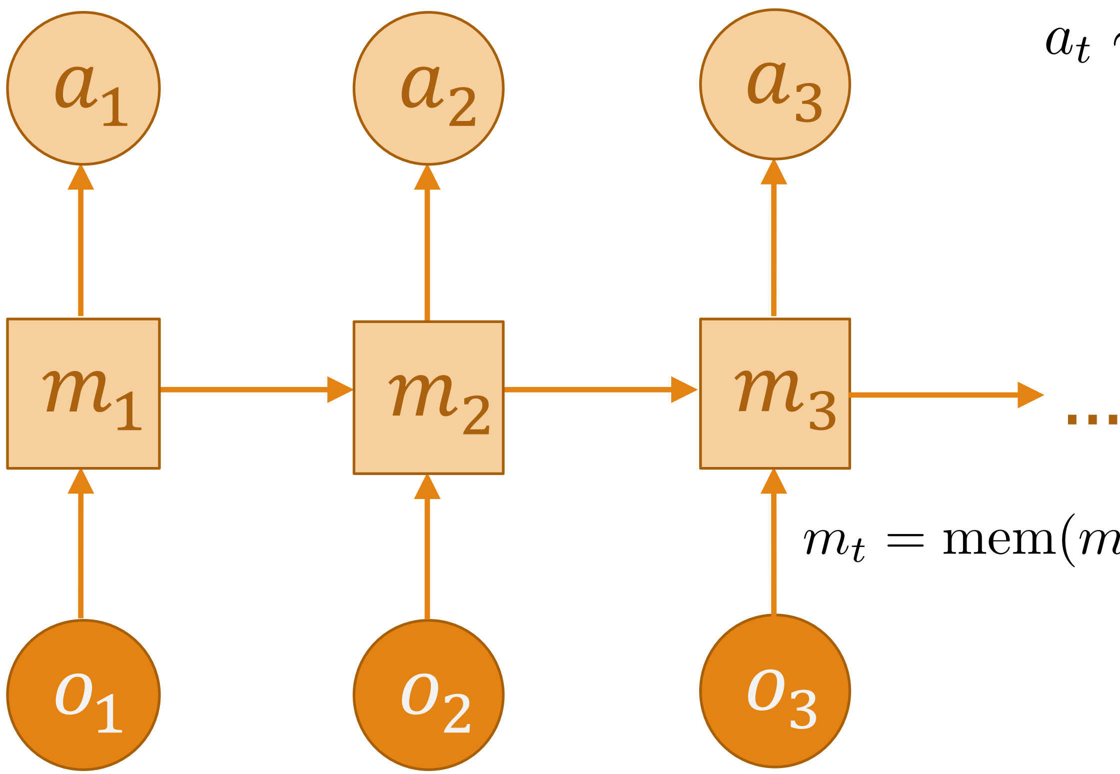
$$\widetilde{\nabla}_\theta R^\pi = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t|m_t;\theta), \quad m_t = \mathrm{mem}(m_{t-1}, o_t; \theta)$$

  - $\Psi_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$ - easy to compute, high variance
  - $\Psi_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l} - b(s_t)$ - baselined estimate

- The optimal baseline is $\dfrac{\mathbb{E}[(\sum_{l=0}^{T-t} \gamma^l r_{t+l})(\nabla_{\theta_j} \log \pi(a_t|m_t))^2]}{\mathbb{E}[(\nabla_{\theta_j} \log \pi(a_t|m_t))^2]}$

- Another important case: $b(s_t) = V^\pi(s_t)$

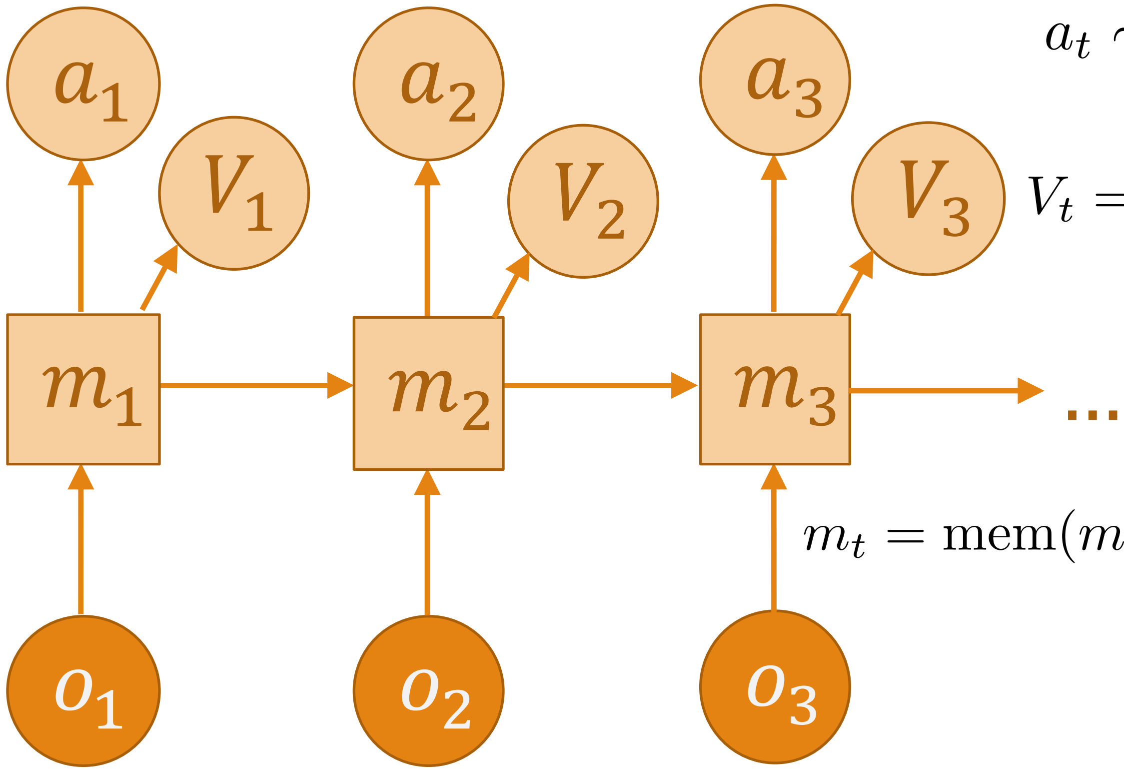
Learning the Value function



$$a_t \sim \pi(a_t | m_t; \theta_A)$$

$$m_t = \mathrm{mem}(m_{t-1}, o_t; \theta_M)$$

Learning the Value function



$$a_t \sim \pi(a_t | m_t; \theta_A)$$

$$V_t = V(m_t; \theta_V)$$

$$m_t = \text{mem}(m_{t-1}, o_t; \theta_M)$$

Final(?) learning algorithm

Repeat until convergence:

1. Collect trajectory $\{(o_t, a_t, r_t)\}_{t=0}^{T}$
2. Update policy parameters using $\widetilde{\nabla}_\theta R = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | m_t^\pi; \theta_A)$
3. Update recurrent parameters using BPTT
4. Update baseline parameters using $\nabla_{\theta_V} \sum_{t=0}^{T} (V(m_t^V; \theta_V) - \sum_{l=0}^{T} \gamma^l r_{t+l})^2$

Final(?) learning algorithm

Repeat until convergence:

1. Collect trajectory $\{(o_t, a_t, r_t)\}_{t=0}^{T}$
2. Update policy parameters using $\widetilde{\nabla}_\theta R = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | m_t^\pi; \theta_A)$
3. Update recurrent parameters using BPTT
4. Update baseline parameters using $\nabla_{\theta_V} \sum_{t=0}^{T} (V(m_t^V; \theta_V) - \sum_{l=0}^{T} \gamma^l r_{t+l})^2$

Actual (wrong) objective:

$$\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right] - \mathbb{E}\left[\sum_{t=0}^{T} \sum_{l=0}^{T-t} (r_{t+l} - V(m_t; \theta_V))^2\right]$$

Learning LSTM policies

- Gradients wrt recurrent parameters are bad after K steps
  - For LSTM K is larger than for RNN, but still a finite number
- Continous setting will require large amount of memory
- An obvious solution is to truncate BPTT after K steps
  - This limits the range of learned dependencies
- Gradient estimate:

$$\widetilde{\nabla}_\theta R = \sum_{t=0}^{T} \Psi_t \nabla_\theta \log \pi(a_t | m_t^\pi; \theta_A)$$

- Consider our advantage estimator:

$$\Psi_t = \sum_{l=0}^{T} \gamma^l r_{t+l} - V^\pi(s_t)$$

Eligibility traces

- Gradient estimate:

$$\widetilde{\nabla}_\theta R = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | m_t^\pi; \theta_A)$$

Eligibility traces

- Gradient estimate:

$$\widetilde{\nabla}_\theta R = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | m_t^\pi; \theta_A)$$

- Let's analyze our advantage estimator

$$\Psi_t = \sum_{l=0}^{T} \gamma^l r_{t+l} - V^\pi(s_t)$$

$$= \sum_{l=0}^{K} \gamma^l r_{t+l} + \sum_{l=K+1}^{T} \gamma^l r_{t+l} - V^\pi(s_t)$$

Eligibility traces

- Gradient estimate:

$$\widetilde{\nabla}_\theta R = \sum_{t=0}^{T} \gamma^t \Psi_t \nabla_\theta \log \pi(a_t | m_t^\pi; \theta_A)$$

- Let's analyze our advantage estimator

$$\Psi_t = \sum_{l=0}^{T} \gamma^l r_{t+l} - V^\pi(s_t)$$
$$= \sum_{l=0}^{K} \gamma^l r_{t+l} + \sum_{l=K+1}^{T} \gamma^l r_{t+l} - V^\pi(s_t)$$

- Without changing the expectation of gradient we can use

$$\Psi_t = \sum_{l=0}^{K} \gamma^l r_{t+l} + \gamma^K \underbrace{V^\pi(s_{t+K+1})}_{\approx V(m_{t+K+1}^V; \theta_V)} - V(m_t^V; \theta_V)$$

Bootstrapping the Baseline

- New error function for the Baseline network

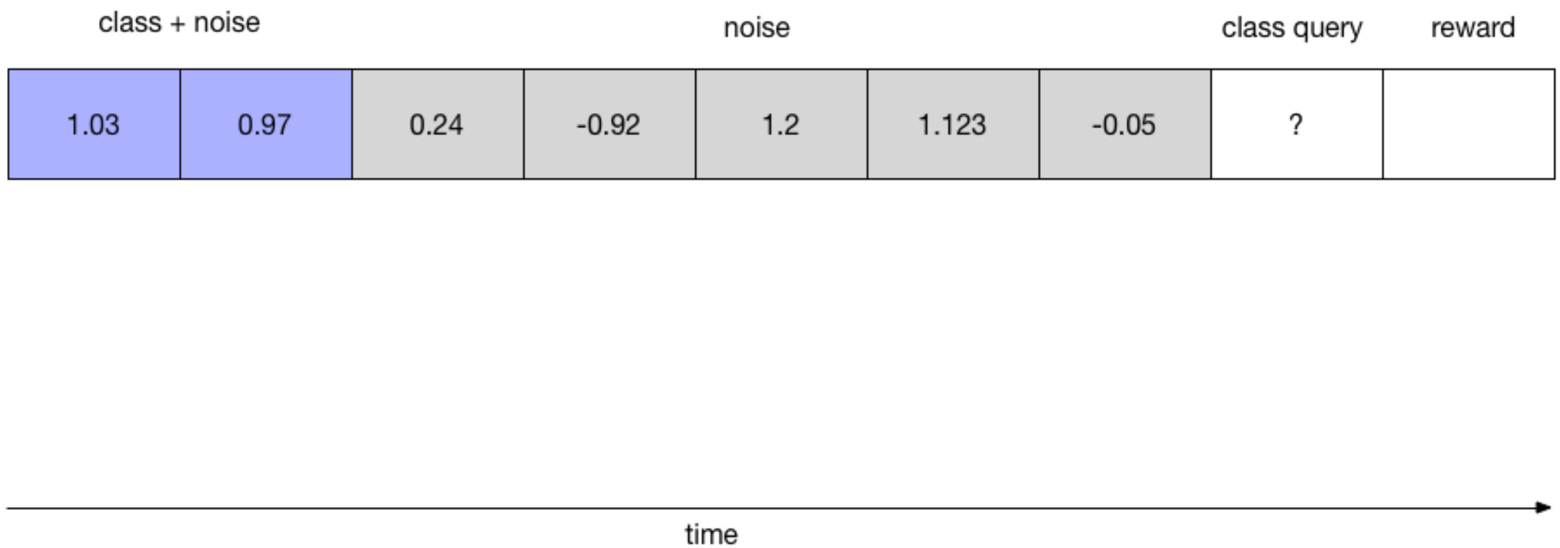
$$\sum_{t=0}^{T}\left(\sum_{l=0}^{K}\gamma^l r_{t+l} + \gamma^K V(m_{t+K+1}^V;\theta_M) - V(m_t^V;\theta_M)\right)^2 \rightarrow \min_{\theta_V}$$

- Memory dynamics is controlled by a second LSTM:

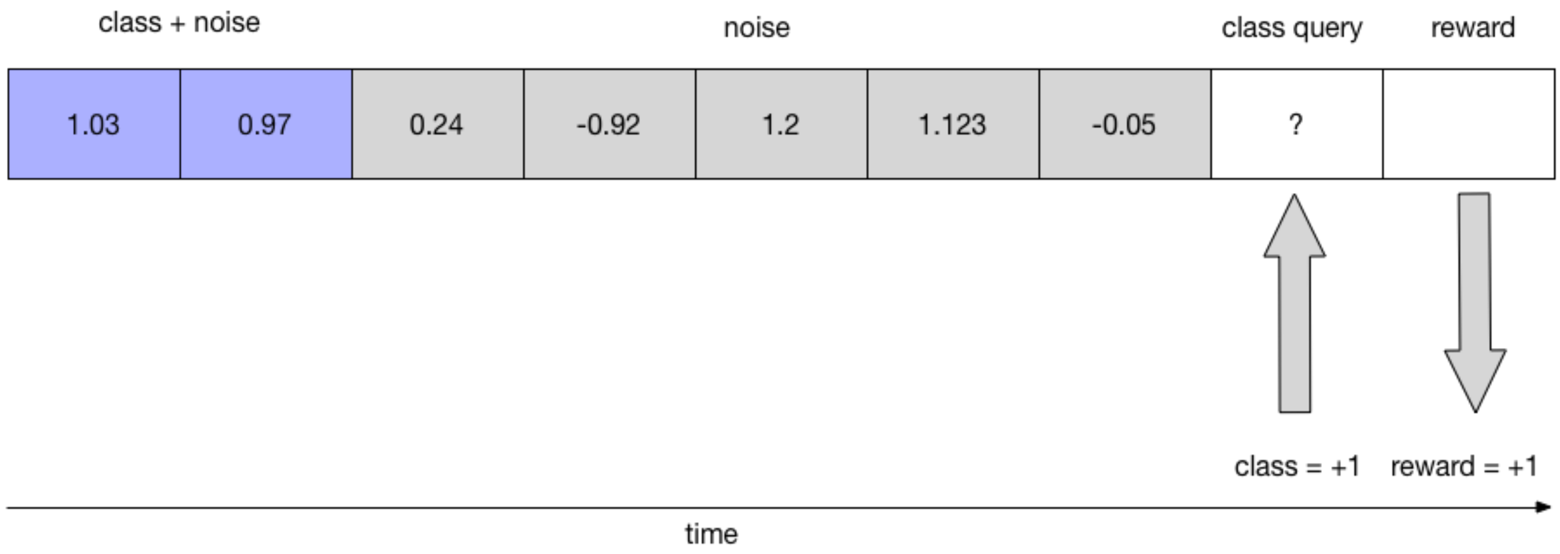
$$m_t^V = \mathrm{mem}(m_{t-1}^V, o_t; \theta_V)$$

Empirical results

Latch task

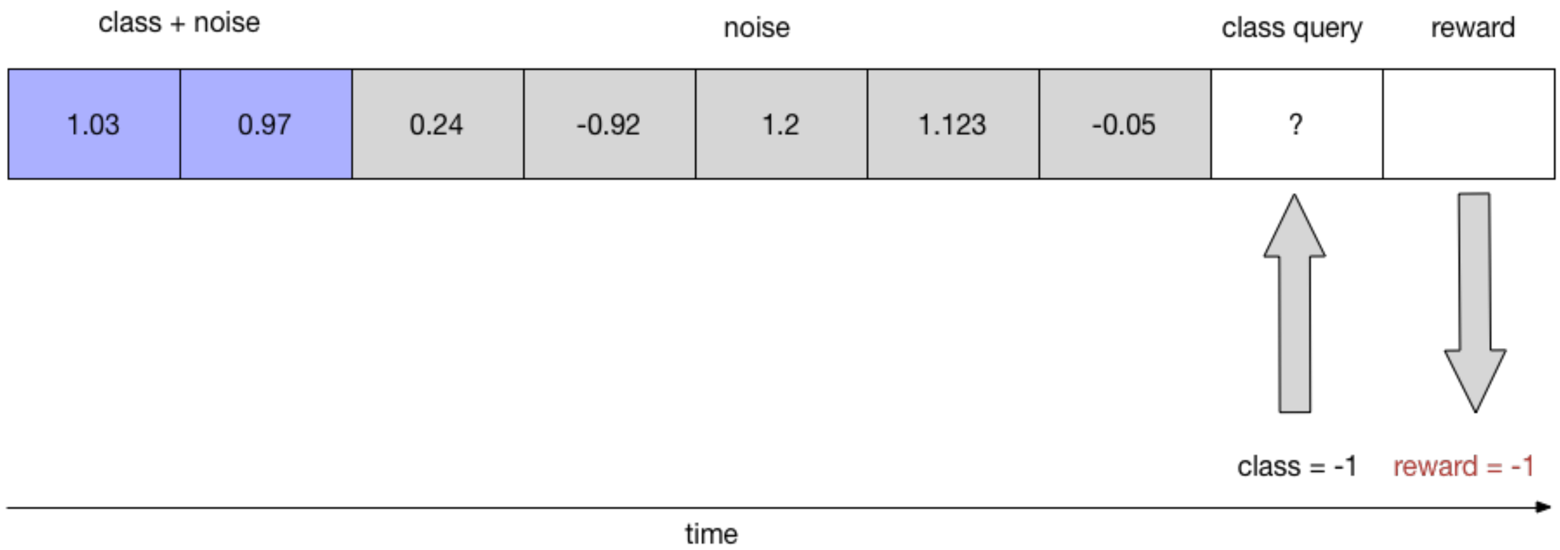

class + noise
noise
class query
reward
1.03
0.97
0.24
-0.92
1.2
1.123
-0.05
?
time

Latch task



class + noise
noise
class query
reward
1.03
0.97
0.24
-0.92
1.2
1.123
-0.05
?
class = +1
reward = +1
time

Latch task



class + noise
noise
class query
reward
1.03
0.97
0.24
-0.92
1.2
1.123
-0.05
?
class = -1
reward = -1
time

Latch task

- Sequences of length 100 are already hard to learn

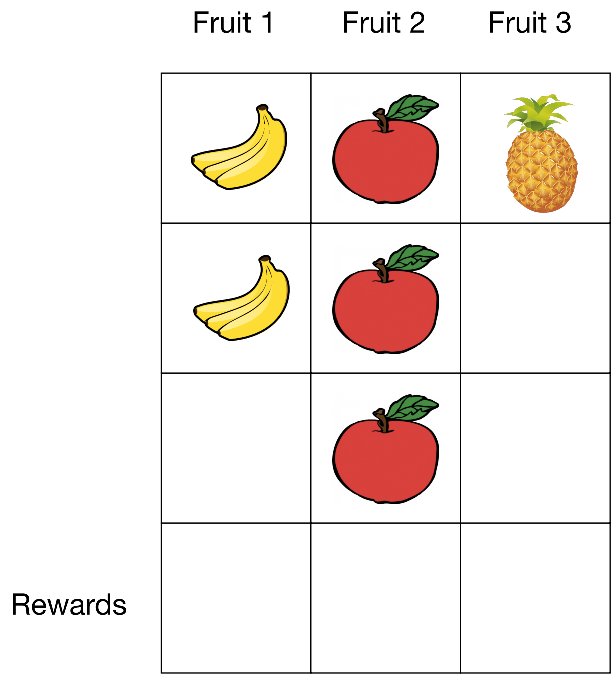
Latch task

- Sequences of length 100 are already hard to learn
- Eliglibility traces work sometimes, but not very stable

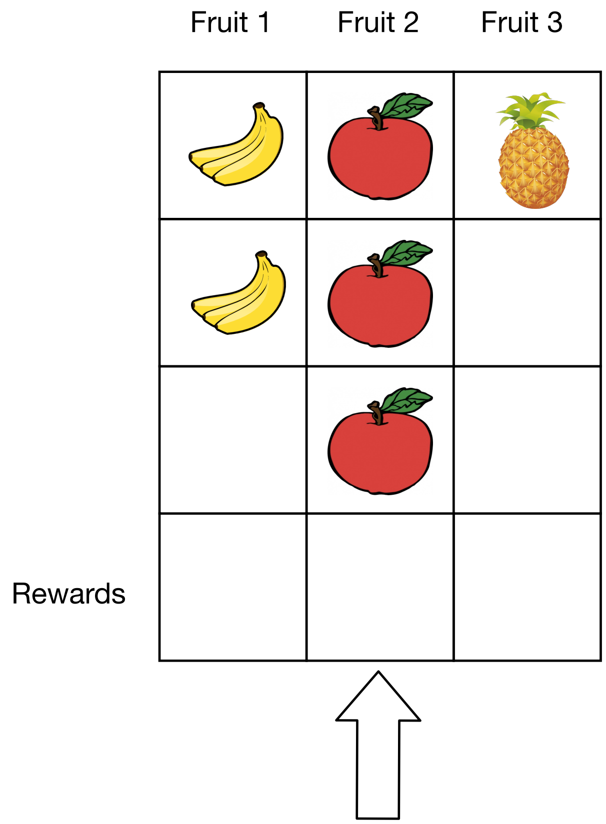
Latch task

- Sequences of length 100 are already hard to learn
- Eliglibility traces work sometimes, but not very stable
- Curriculum learning:
  - Train on shorter sequences
  - Increase sequence length over time
  - Works well even with truncated BPTT, but no guarantees

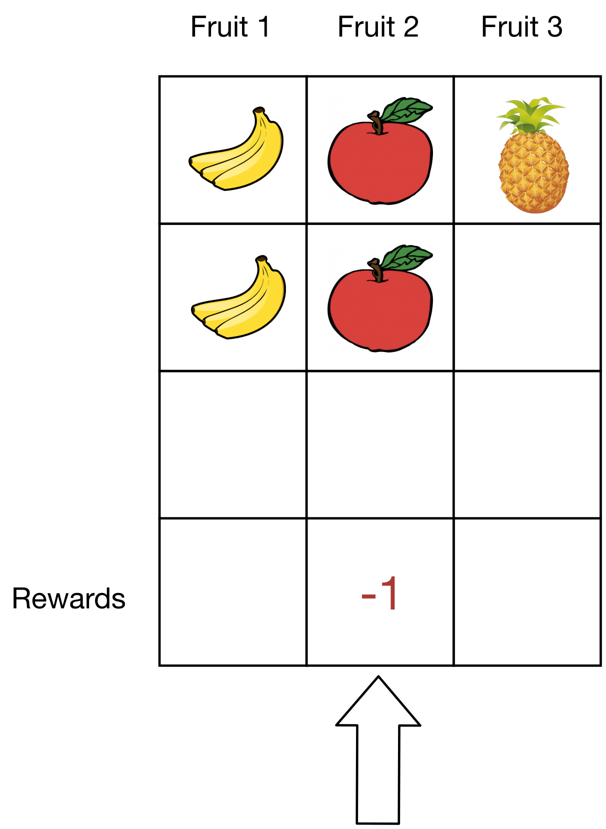
EAT game



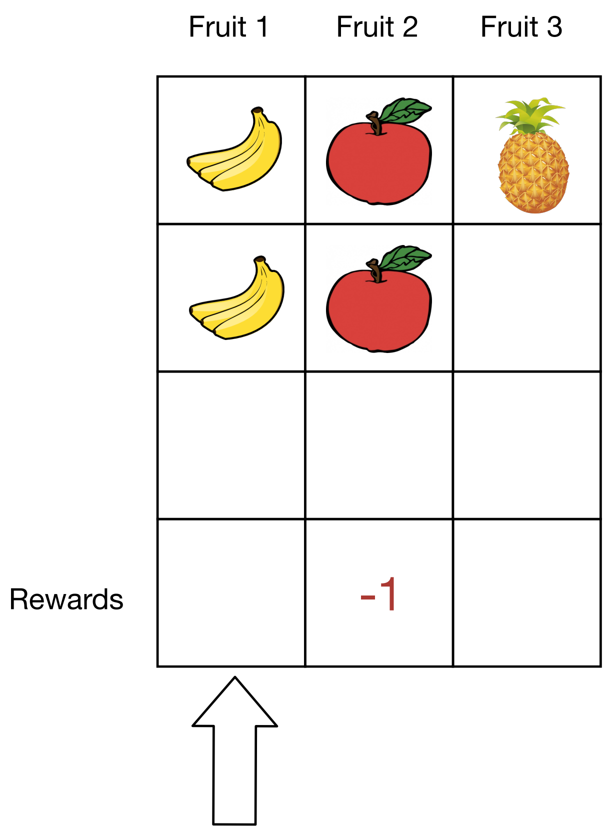
EAT game



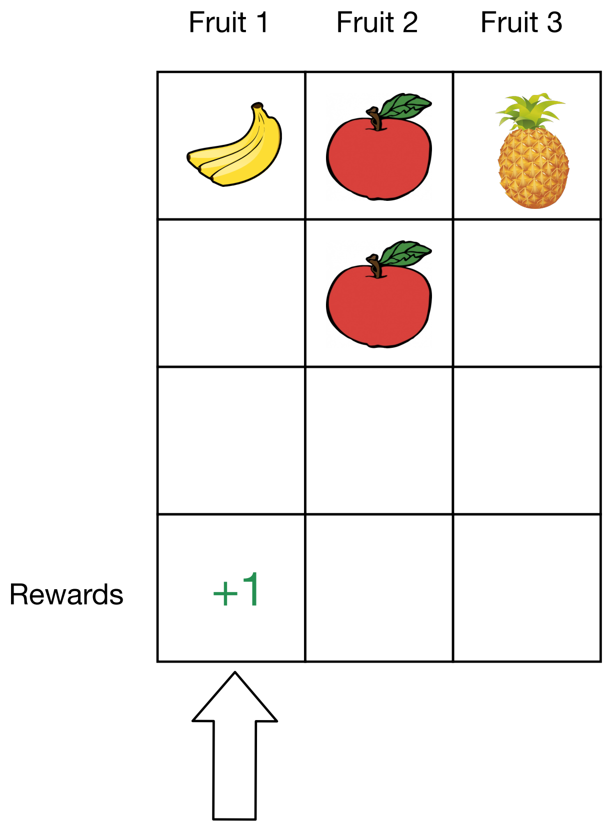
EAT game



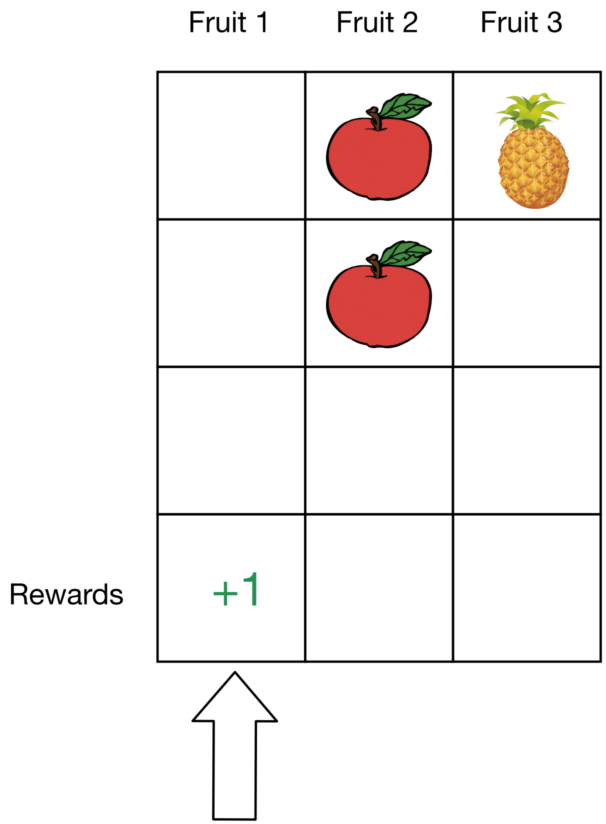
EAT game



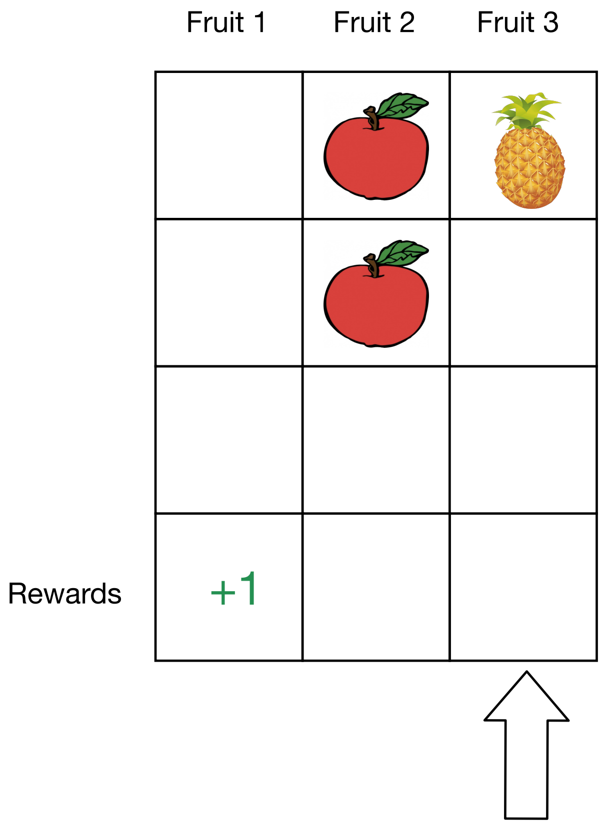
EAT game



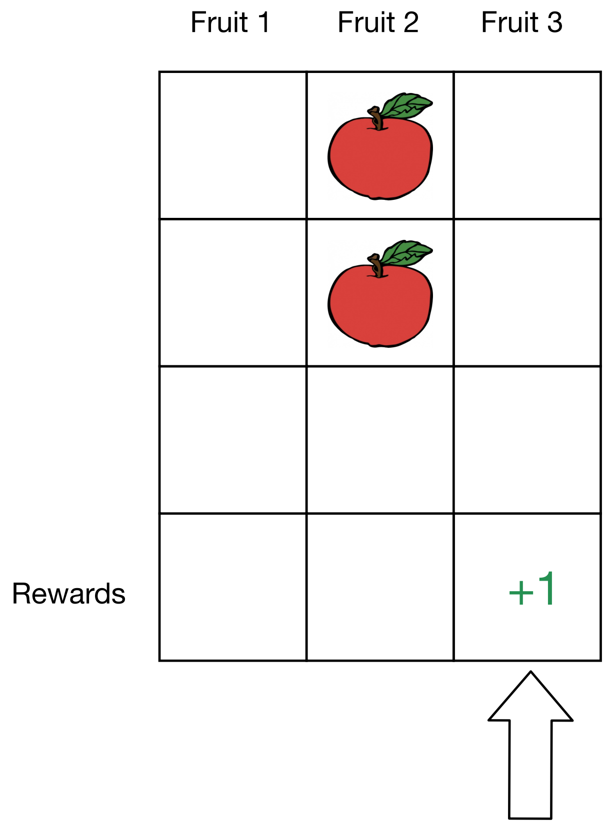
EAT game



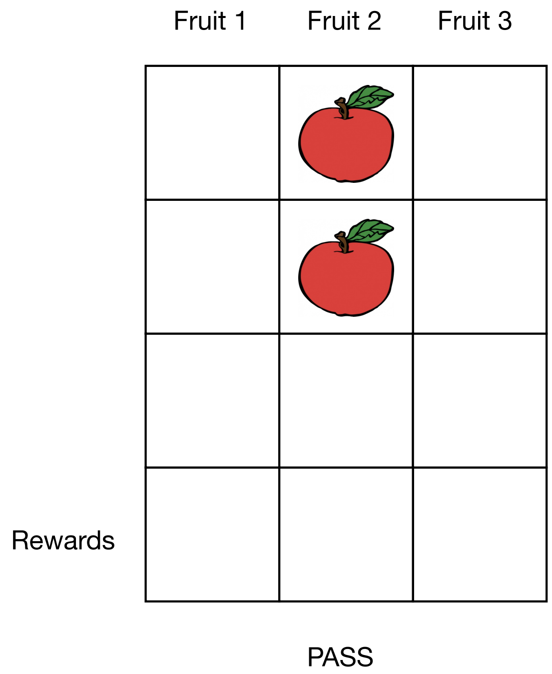
EAT game



EAT game



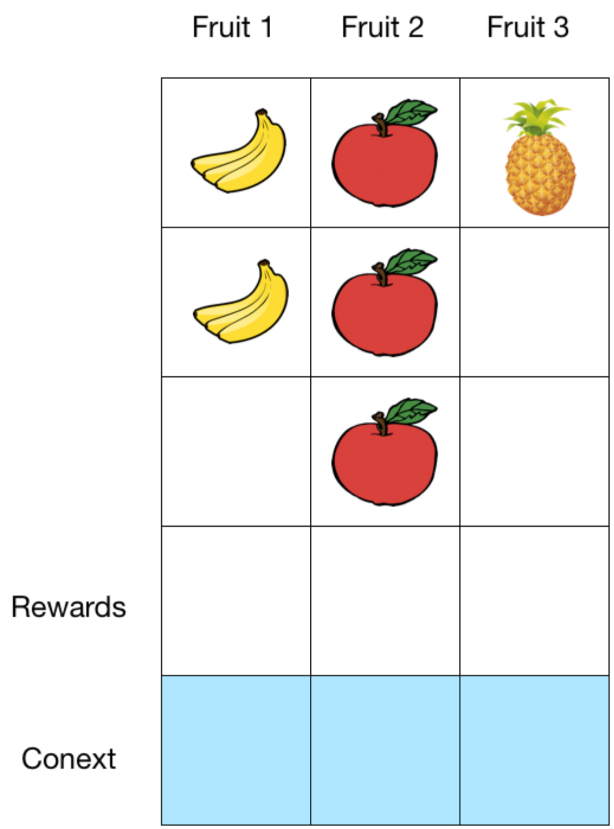
EAT game



Eat game

- Given enough time LSTM can learn the optimal strategy
- Variance reduction techniques and advances optimization methods dramatically improve convergence

Big episode EAT



Big episode EAT

	Fruit 1	Fruit 2	Fruit 3
			
			
			
Rewards	+1	-1	+1
Conext	1	0	0

Big episode EAT

	Fruit 1	Fruit 2	Fruit 3
			
			
Rewards	-1	-1	+1
Conext	0	1	0

Big episode EAT

- This environment appeared to be very tough for the LSTM agent

Big episode EAT

- This environment appeared to be very tough for the LSTM agent
- Eligibility traces do work, but achieve 80-90% of the optimal score
  - Couldn't approximate the Value function well

Big episode EAT

- This environment appeared to be very tough for the LSTM agent
- Eligibility traces do work, but achieve 80-90% of the optimal score
  - Couldn't approximate the Value function well
- The number of contexts is the main bottleneck
  - Cannot be handled by curriculum learning directly

Big episode EAT

- This environment appeared to be very tough for the LSTM agent
- Eligibility traces do work, but achieve 80-90% of the optimal score
  - Couldn't approximate the Value function well
- The number of contexts is the main bottleneck
  - Cannot be handled by curriculum learning directly
- Dirty trick with setting $\Psi_t = r_t$ worked
  - Since our strategy is recurrent future rewards influence gradients at time t
  - Prone to bad value function