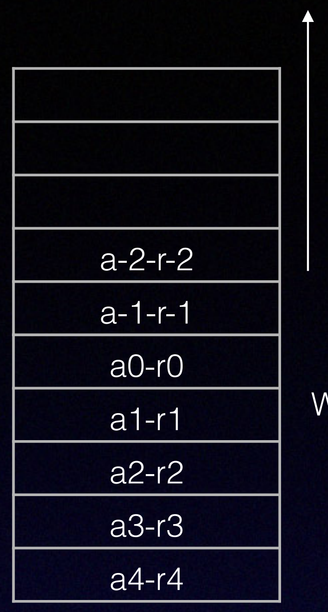
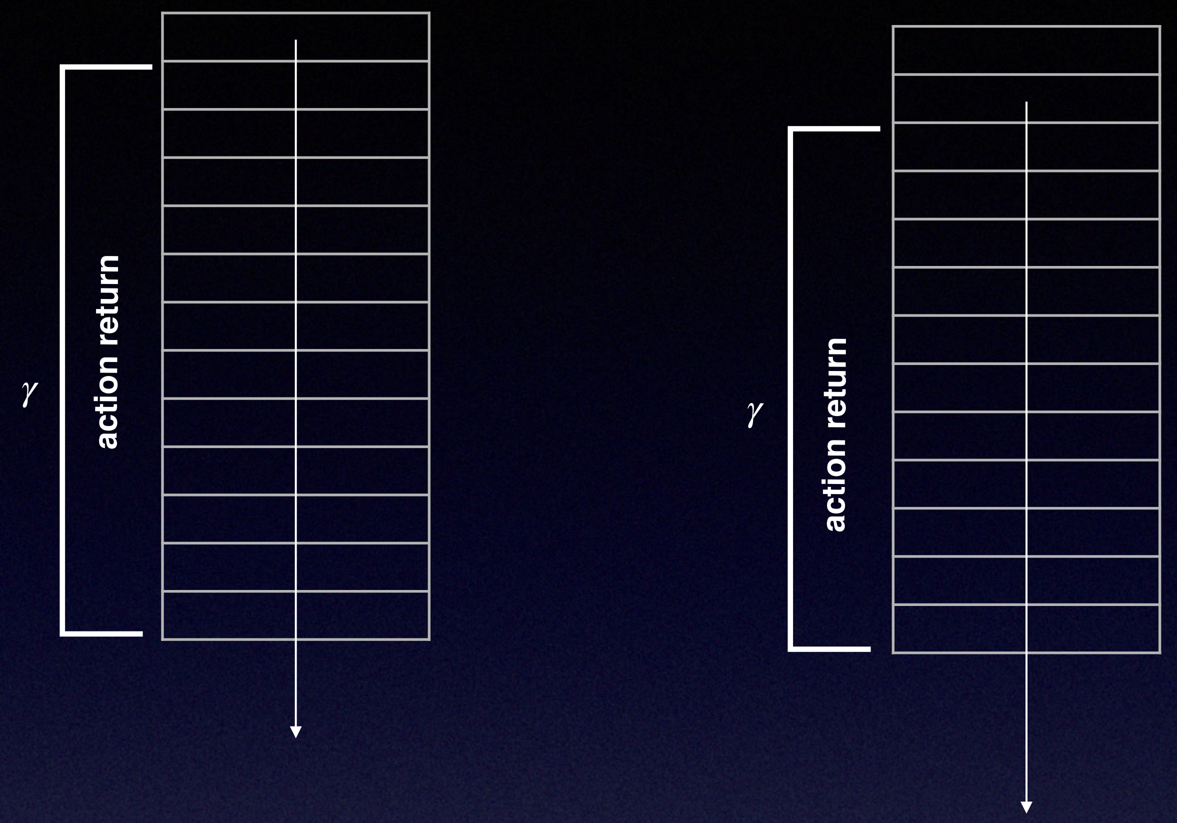
Reinforcement Learning

Credit Assignment Problem



Previous action/rewards

Who is responsible for reward r1. This action (a1) or previous actions



Sum of discounted rewards that come after a reward at a step helps evaluate action impact

γ **action return =** Σ $Discounted\ Rewards$

$\gamma \approx 0$ future rewards have little impact; how does this action affect immediate rewards

Cart-Pole actions have short term effects.

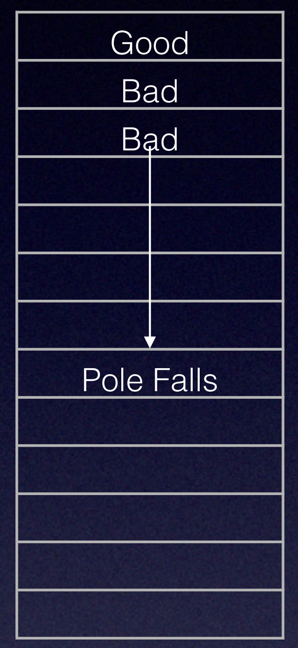
Why?

Bad or good actions may undone by immediate future actions

$\gamma \approx 1$ future rewards have large effect; how does this action affect future rewards

Credit Assignment Problem

- Good actor in bad movie
- Top runner losing race
- Secret drive route flooded with traffic



A **good** action may followed by **bad** actions, but this will be a low occurrence if many episodes are run.

Good actions are more likely to be followed by good actions.

Run many action episodes -> solve action returns for each step in each episode step -> Standardization on each action return is called action advantage

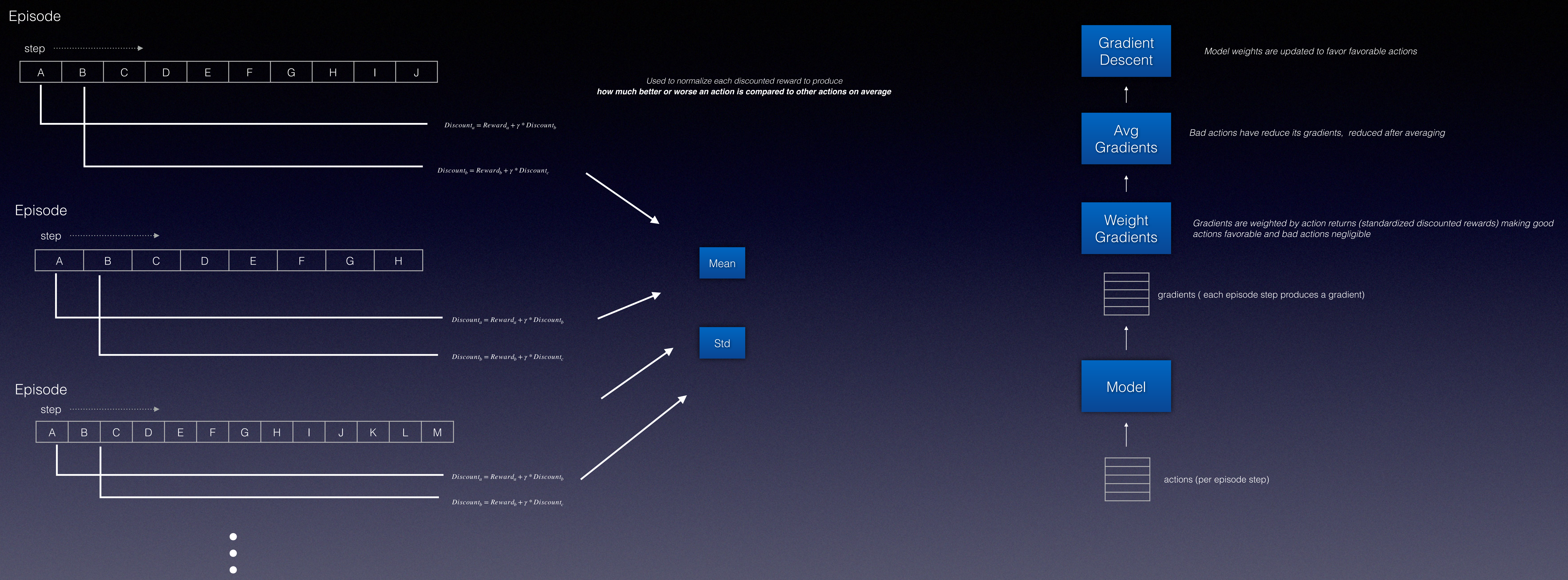
(-) Action Advantage (action will result in bad future)
(+) Action Advantage (action will result in good future)

Multiply gradient by action advantage

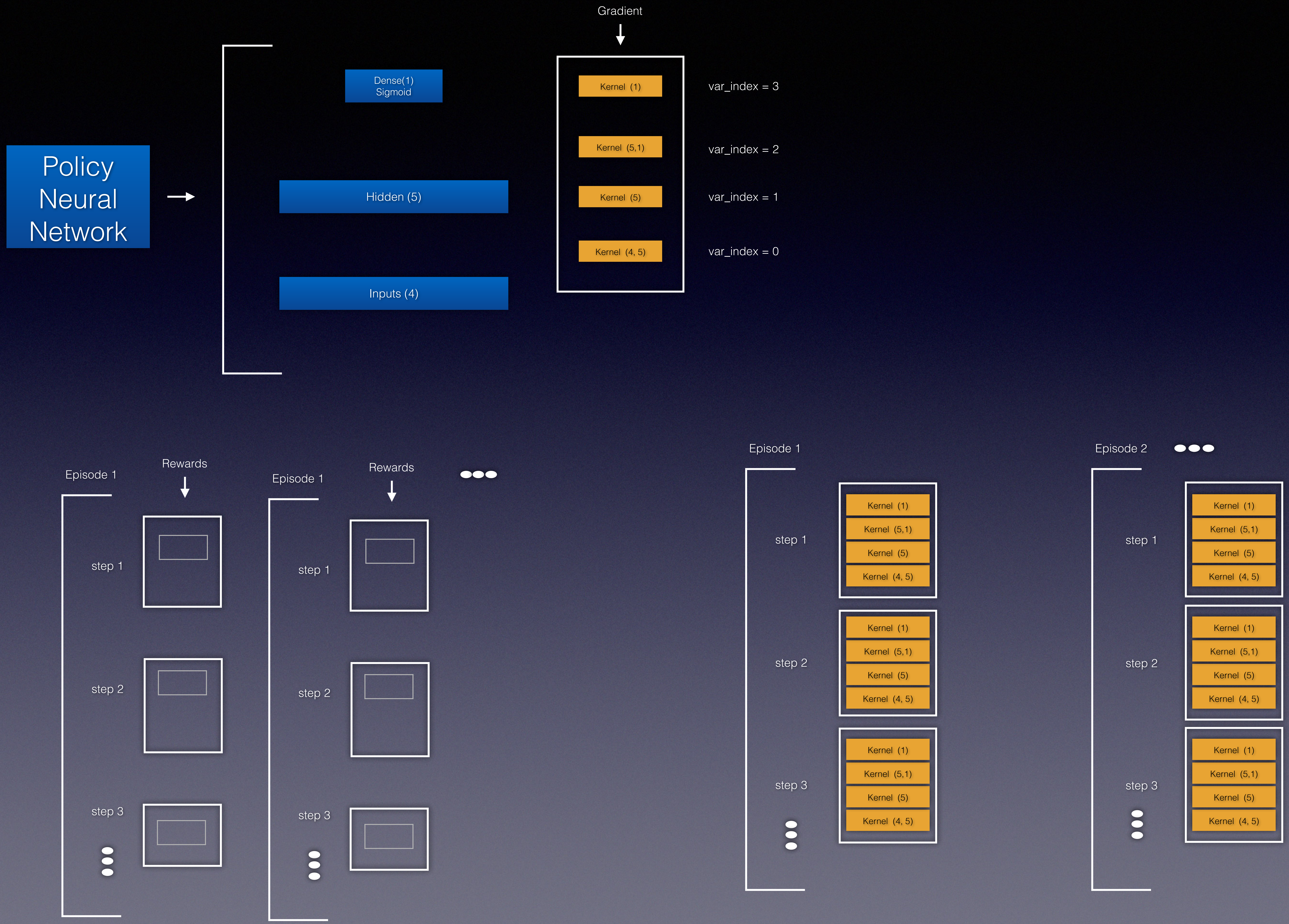
(+ Action Advantage Mult) The gradients will reduce loss. Makes action move likely in future.
(- Action Advantage Mult) The gradients will not reduce loss. Makes action less likely in future.

Note:
Negative action advantages shorten the number of steps in an episode(falling pole).
Thus, making negative action advantages smaller than positive action advantages,
This produces larger action-advantage and gradient products (training will learn to fit positive actions)

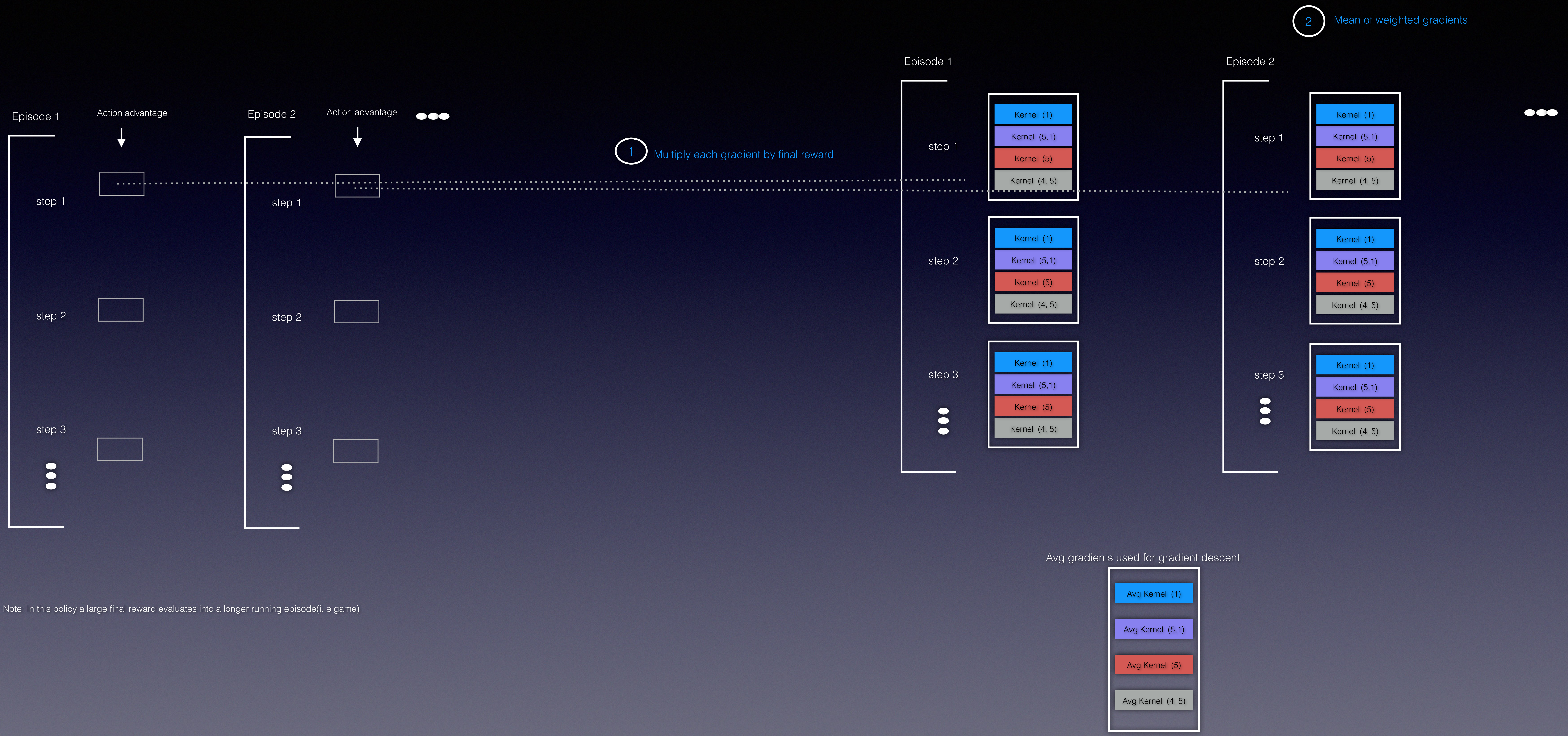
Policy Gradients



Cart Pole Gradient



Cart Pole Gradient



Note: In this policy a large final reward evaluates into a longer running episode(i..e game)

Cart Pole

ACTIONS

Some episode

$\gamma = 0.8$

→	→	→
reward	reward	reward
10	0	-50

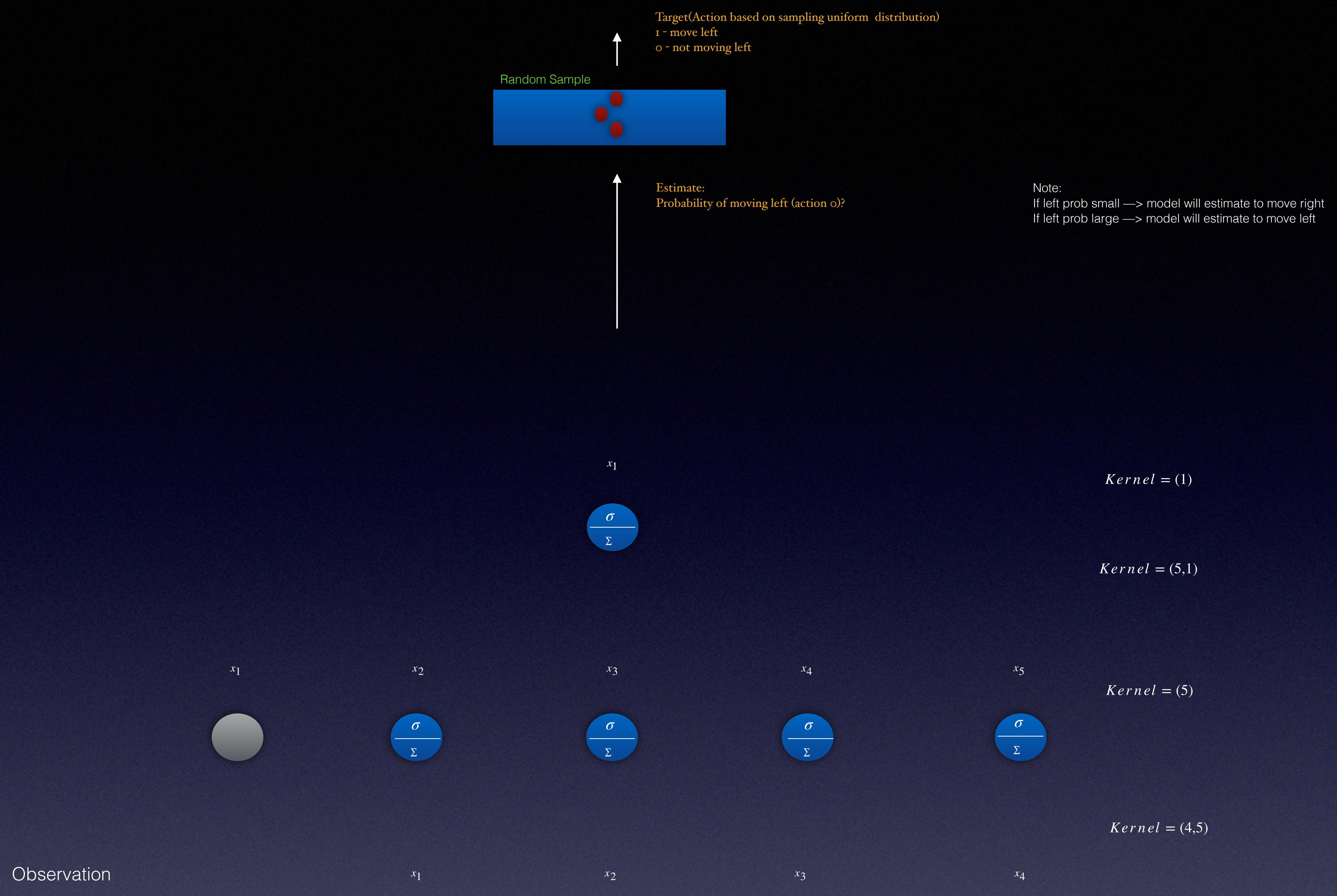
$10 + \gamma \times 0 + \gamma^2 \times (-50)$

RETURN ACTION
(ADVANTAGE)

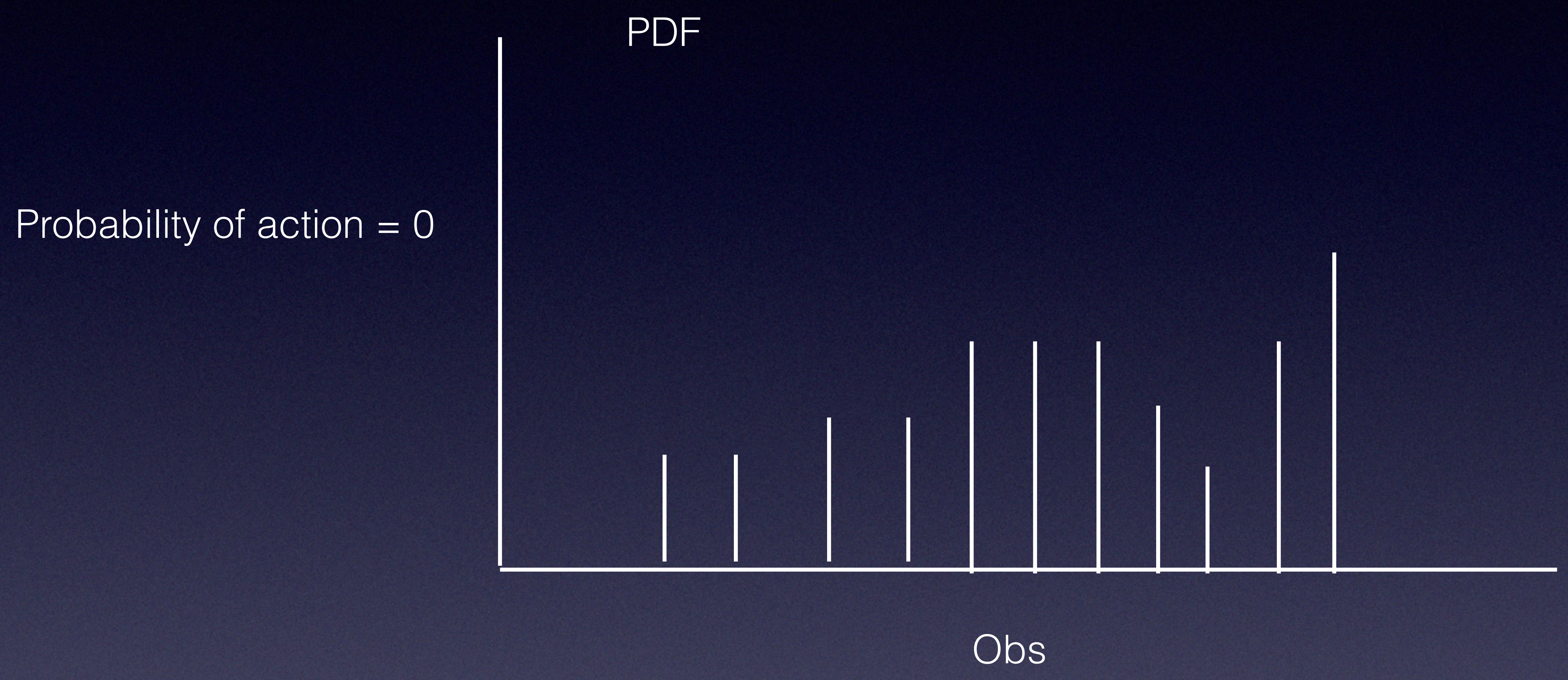
$0 + \gamma \times (-50)$

-50

Cart Pole Policy



Cart Pole Policy



The policy probability distribution is changed during training to increase rewards (keeping cart standing)

Discounted Reward

Discount allows us to evaluate actions. Good actions will get higher returns than bad ones on average.

Discount Factor : $\gamma = 0.8$

R0	R1	R2

1st action last action

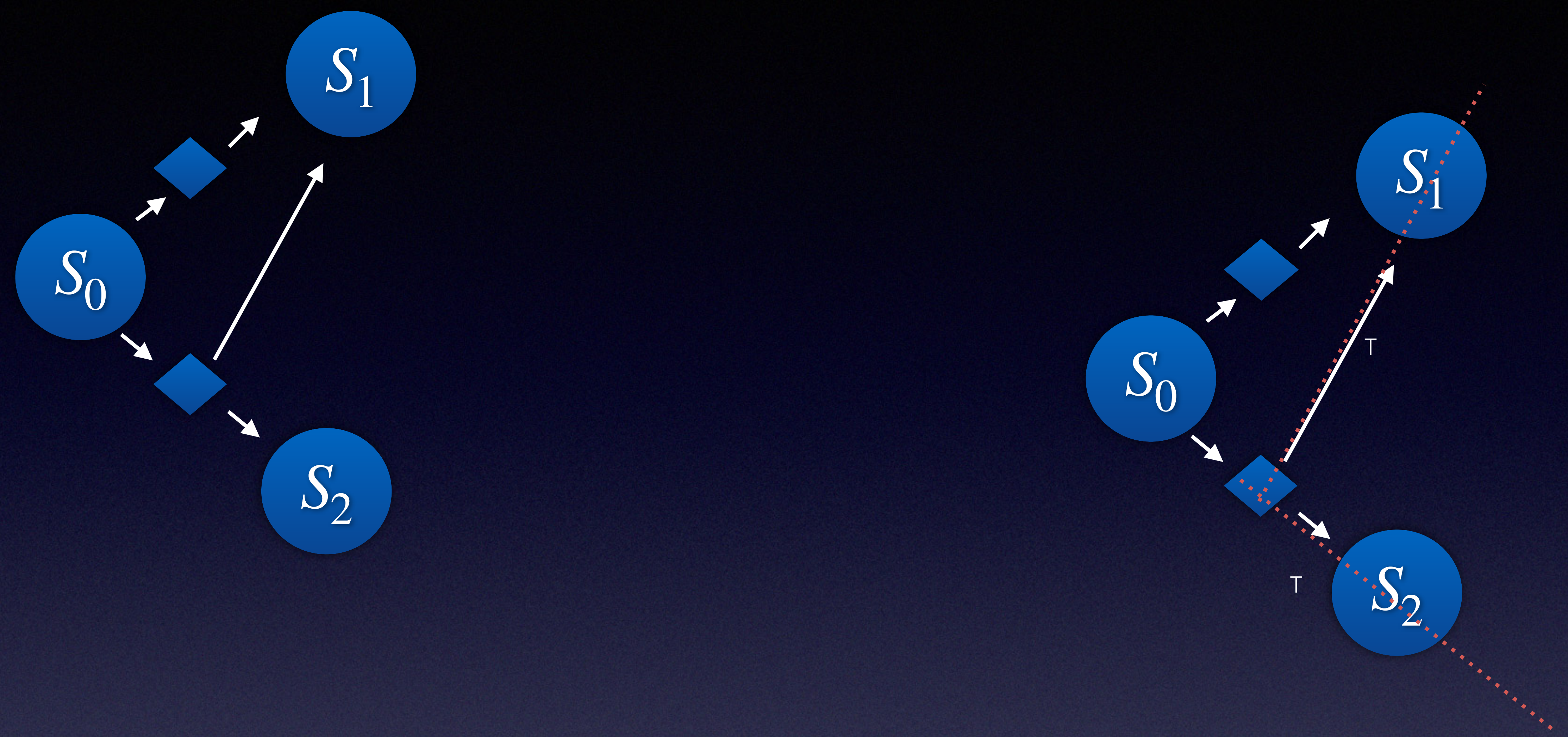
DISCOUNT0	DISCOUNT1	DISCOUNT2

$$R0 + R1 \times \gamma^1 + R2 \times \gamma^2 = R0 + \gamma(R1 + \gamma(R2))$$

$$\mathbf{R1} + \gamma(R2)$$

$$\mathbf{R2}$$

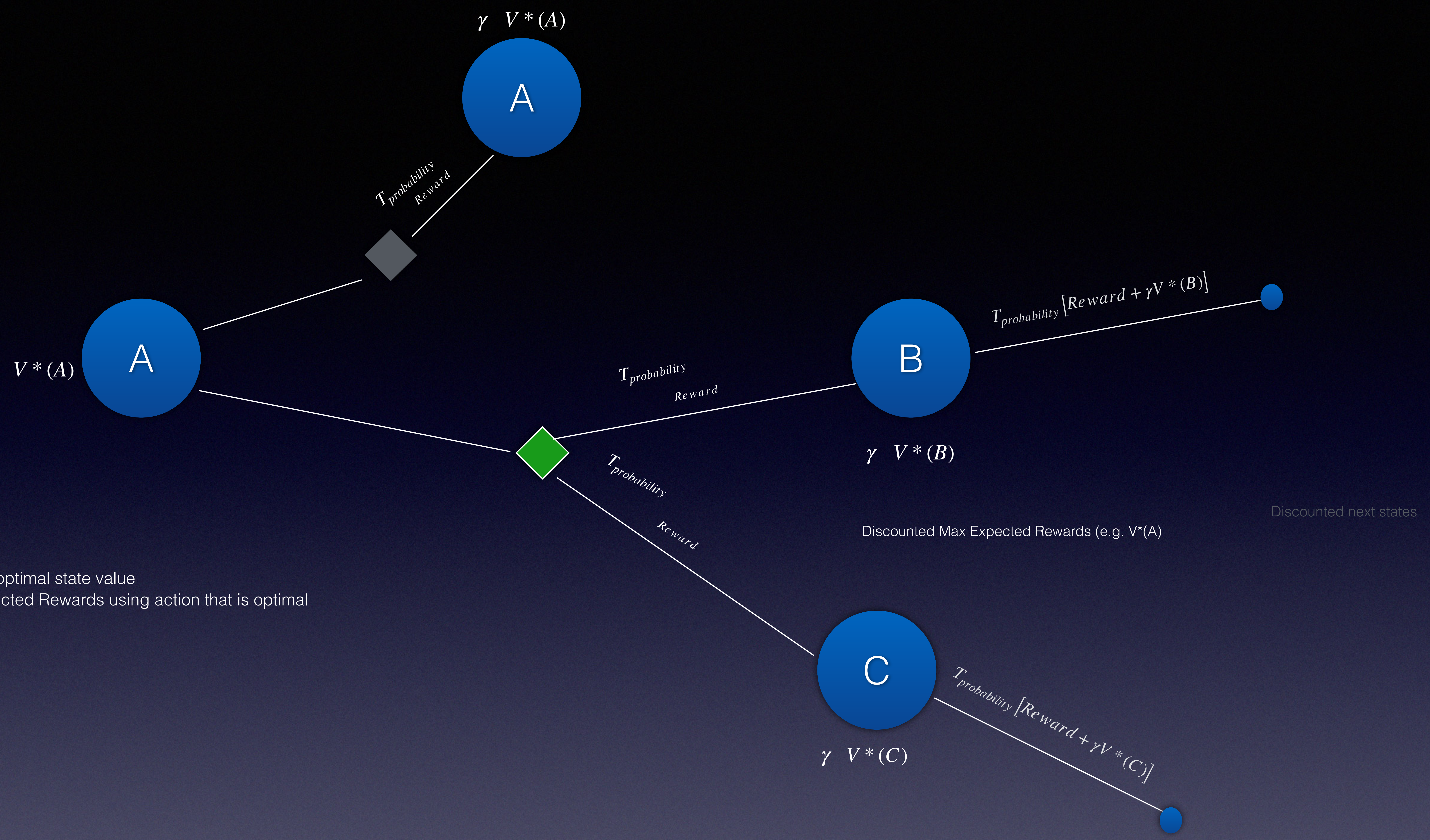
Bellman Optimal



Optimal State Value (Expectation)

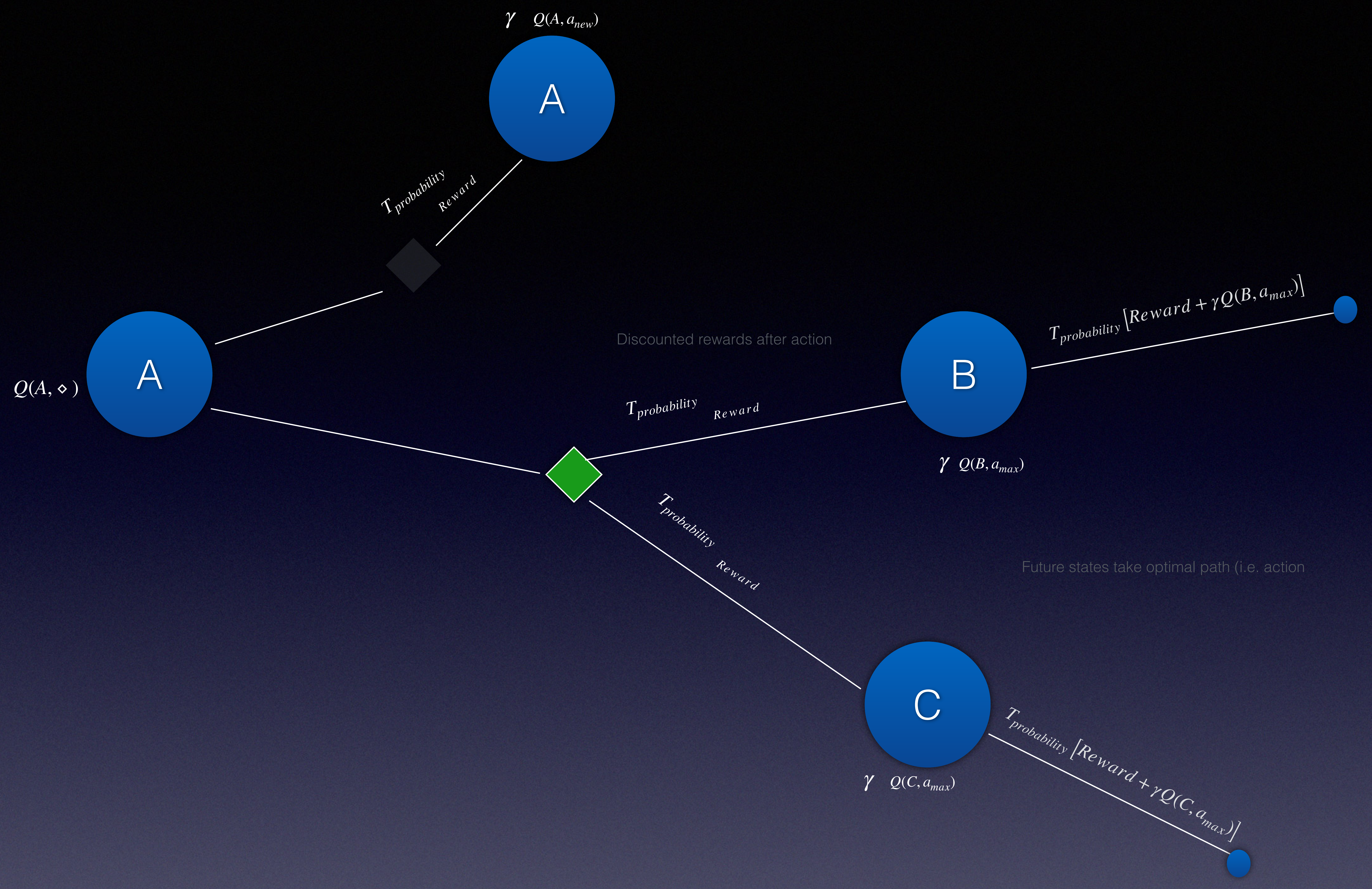
Note:
T= Transition Probability
Statistical Average of an agent moving to a state after an action

Bellman Optimal

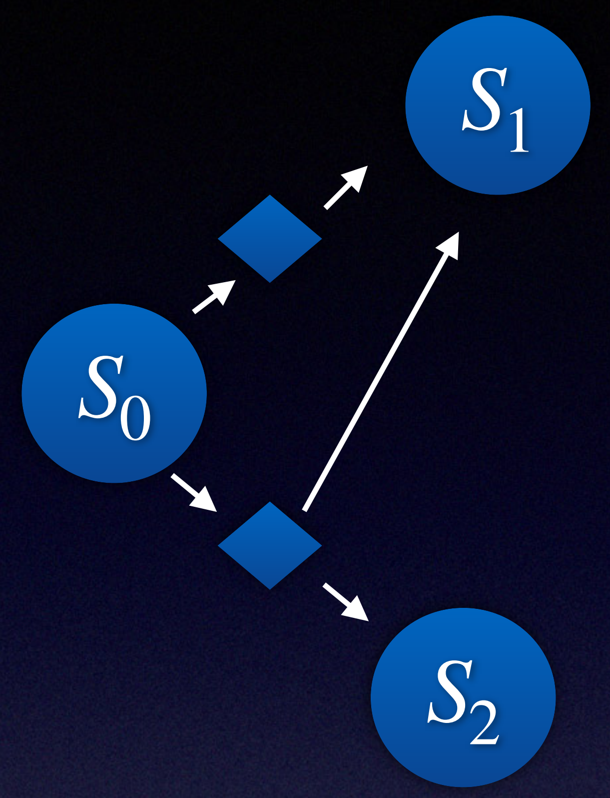


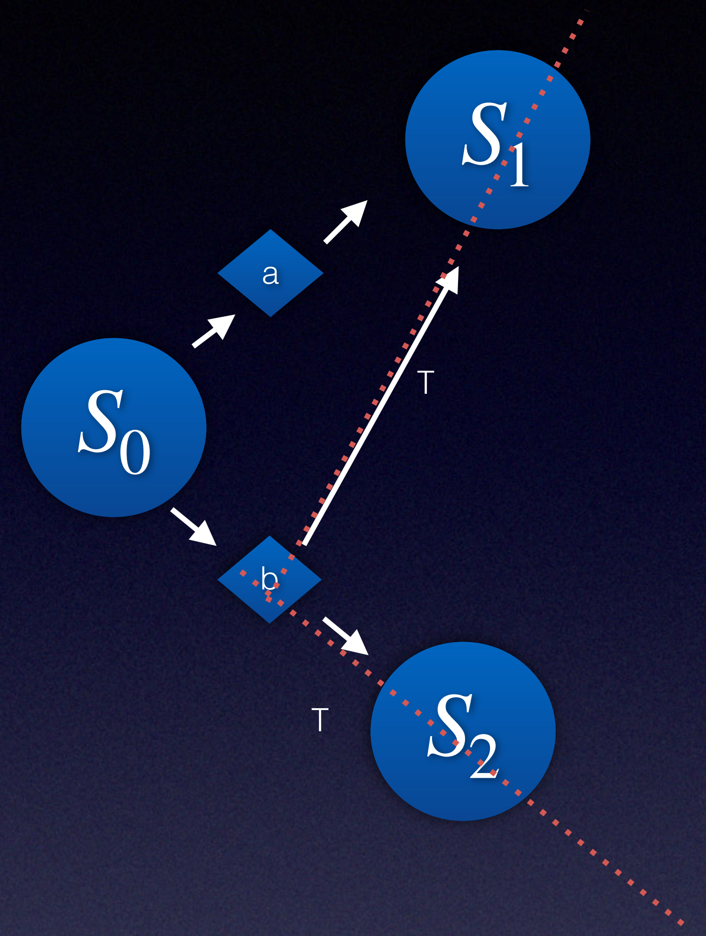
Estimate optimal state value
Max Expected Rewards using action that is optimal

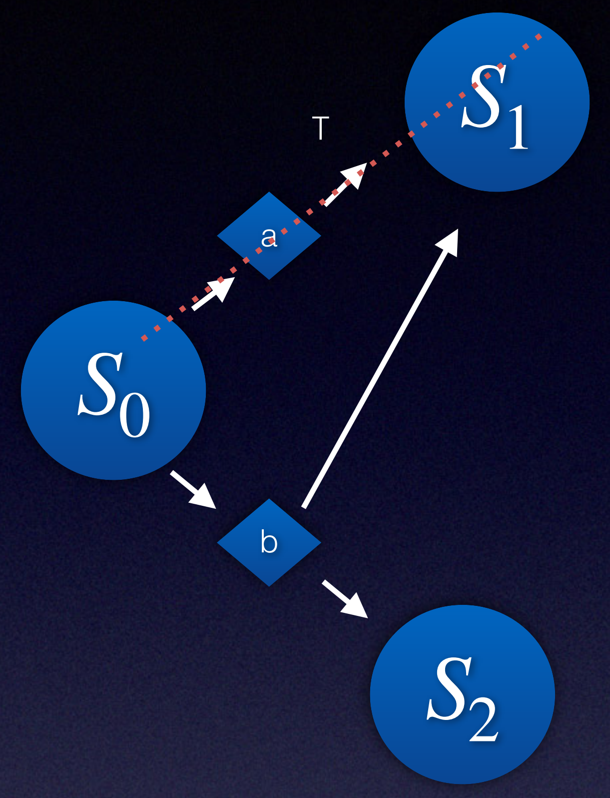
Quality Value



Quality Value







Optimal Policy for agent (Expectation)

$\boxed{Q(s_0, b)}$ $\boxed{T}$ $\boxed{R\_01}$ γ $\boxed{Q(s_1, max_{action})}$ +
$\boxed{T}$ $\boxed{R\_02}$ γ $\boxed{Q(s_2, max_{action})}$

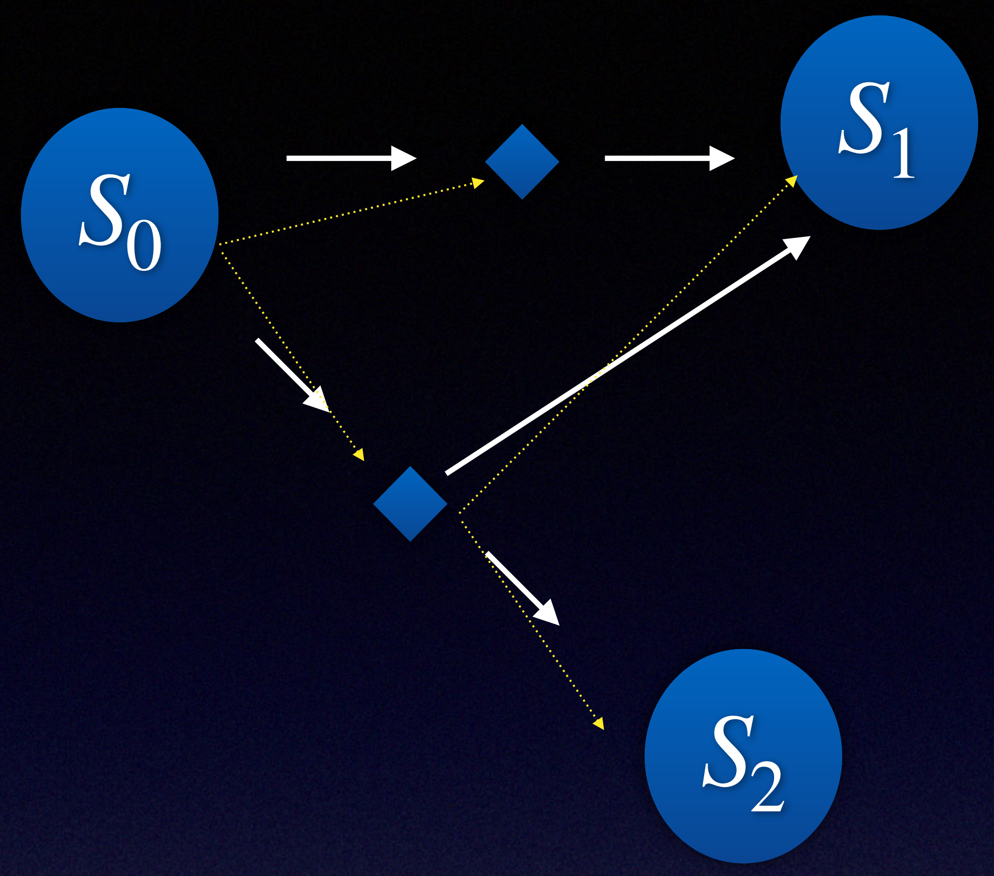
$\boxed{Q(s_0, a)}$ $\boxed{T}$ $\boxed{R\_01}$ γ $\boxed{Q(s_1, max_{action})}$

$Q_{init}(s, a)$

	a	b
S_0	0	0
S_1	0	0
S_2	0	0

$Q_{next}(s, a)$

	a	b
S_0	_	_
S_1		
S_2		



Random policy explores network, updates state values (i.e. estimations) based on rewards and transitions explored.

Algorithm

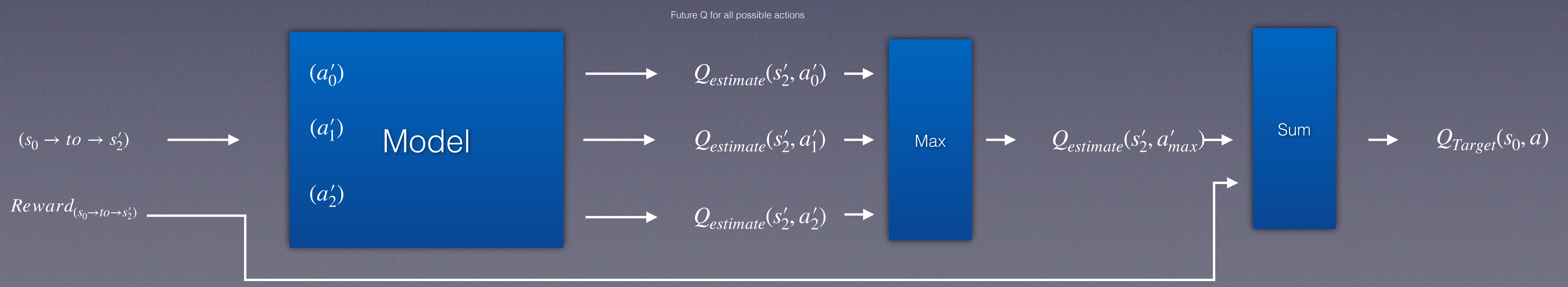
$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha[\underbrace{r + \gamma \overbrace{Q_k(s', a'_{max})}^{\text{sum of future discounted rewards}}}_{\text{Target Q Value (not scalable)}}]$$

Approximate $Q_k(s', a'_{max}) = Q_{k\approx}(s', a'_{max})$

Approx Algorithm

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha[r + \gamma \overbrace{Q_{\approx}(s'_2, a'_{max})}^{\text{Estimate sum of future discounted rewards}}]$$

$$Q_{k+1}(s,a) = Q_k(s,a) + \underbrace{\alpha}_{\text{learn\_rate}}(\overbrace{\underbrace{[r + \gamma Q_{\approx}(s', a'_{max})]}_{\text{Target Q Value}} - Q_k(s,a)}^{\text{Error (goal is to reduce error)}})$$



Algorithm

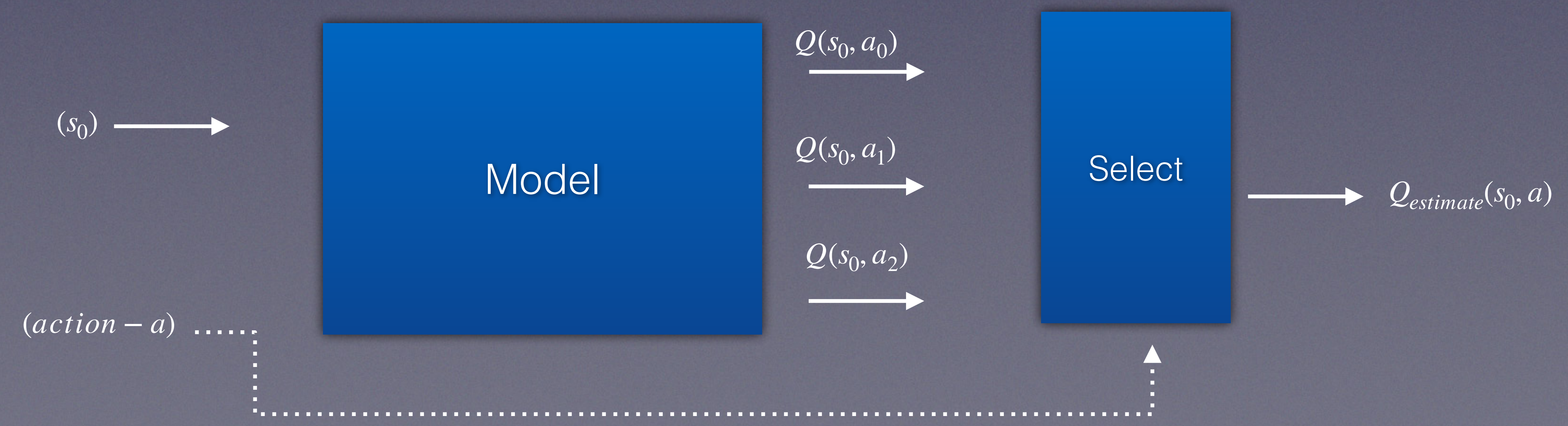
$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha[\underbrace{r + \gamma Q_k(s', a'_{max})}_{\text{Target Q Value (not scalable)}}]$$

Approximate $Q_k(s', a'_{max}) = Q_{k\approx}(s'_2, a'_{max})$

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha[r + \gamma Q_{\approx}(s'_2, a'_{max})]$$

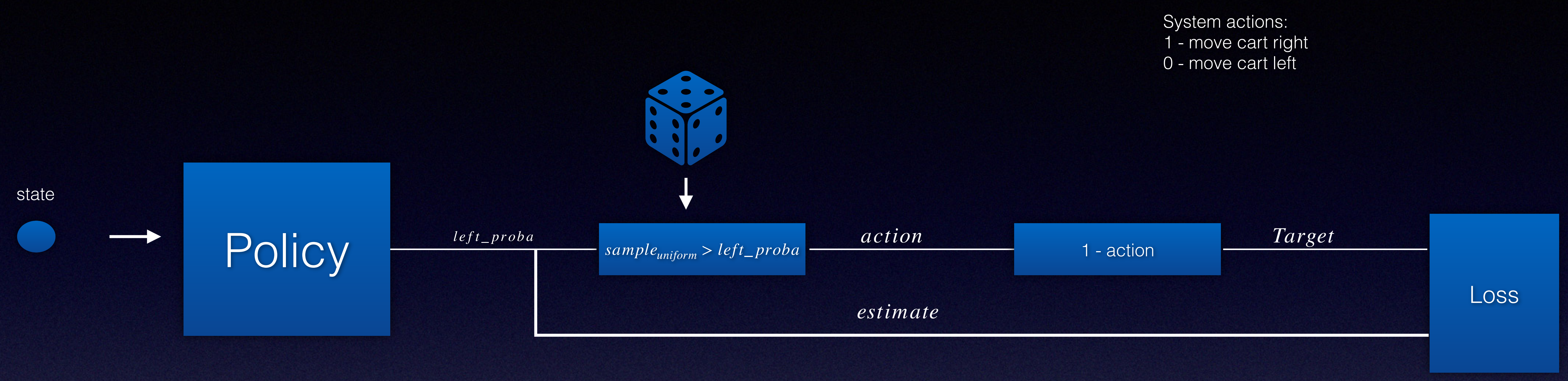
Error (algorithm will eventually converge when there is negligible error between Q(s,a) and QTarget(s,a). So model trains to lower error for all state-action combinations explored in network .

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha(\underbrace{[r + \gamma Q_{\approx}(s'_2, a'_{max})]}_{\text{Target Q Value}} - \underbrace{Q_k(s,a)}_{\text{Estimated using model}})$$



Lunar Lander v3

Cart Pole


System actions:
1 - move cart right
0 - move cart left
state
Policy
left_proba
sample_uniform > left_proba
action
1 - action
Target
Loss
estimate

Lunar Lander v3