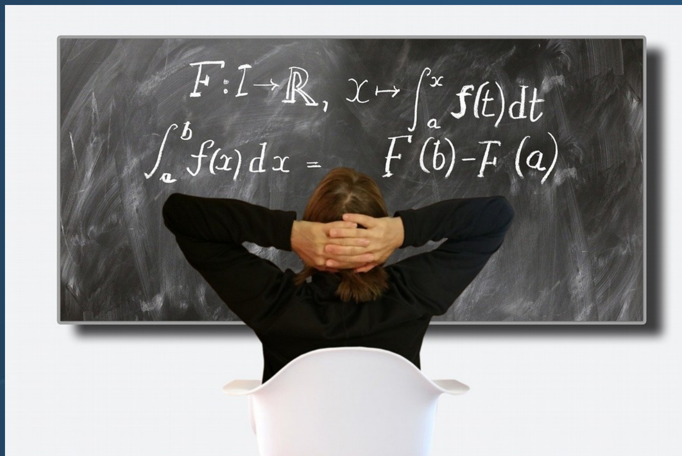
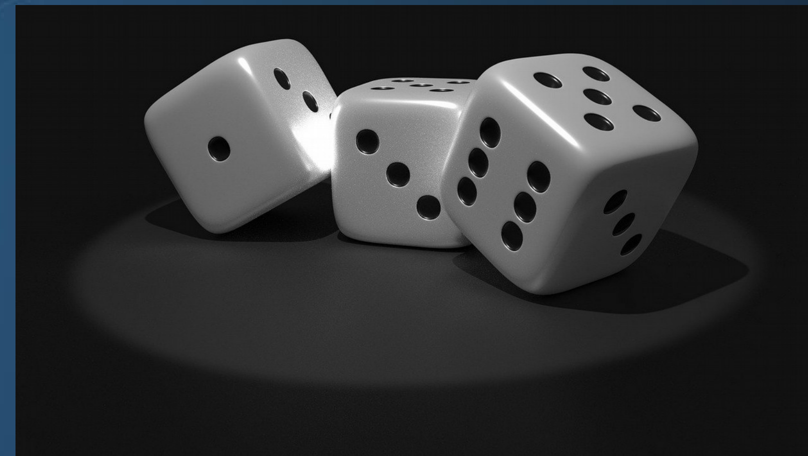
Actor Critic Methods: From Paper to Code

Policy Approximation and its Advantages

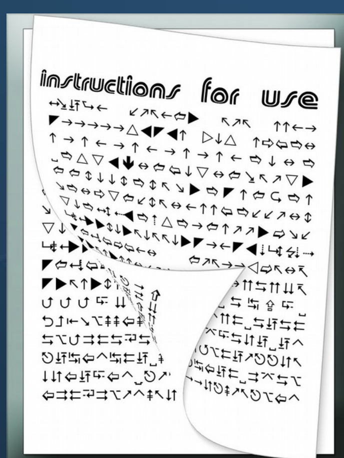
What's a Policy?



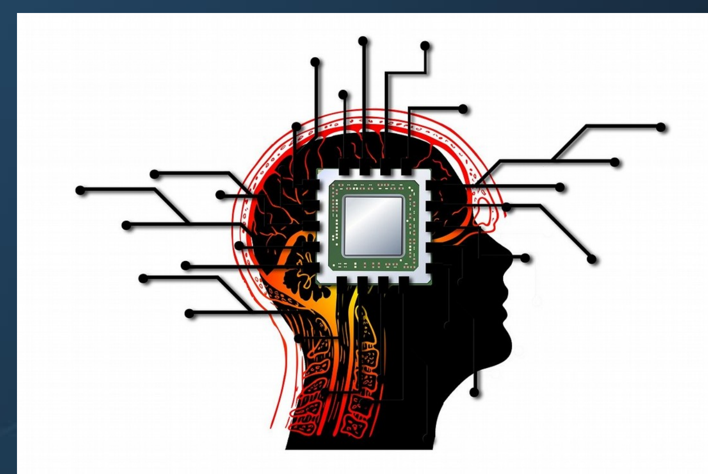
Policy → probability distribution



M.C. → deterministic / epsilon soft

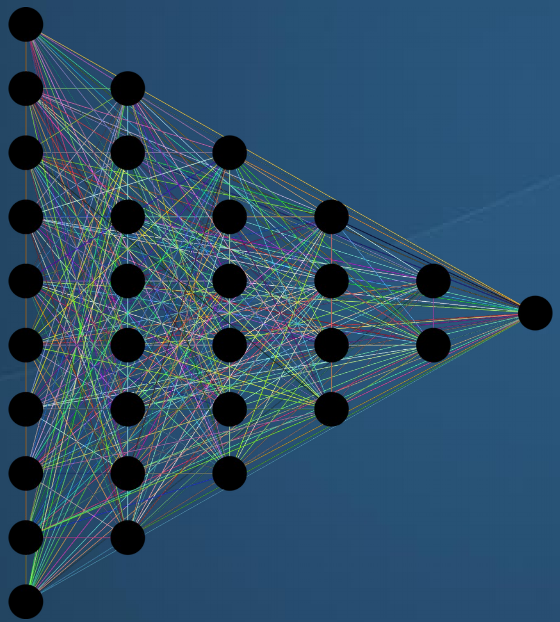


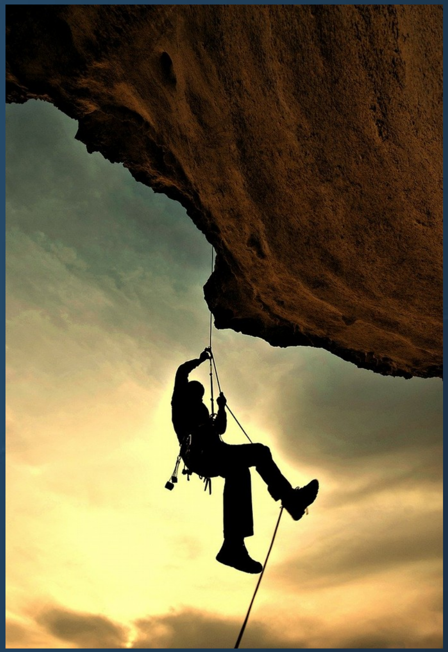
Policy is a function of V or Q



Approximate policy directly

Policy Approximation





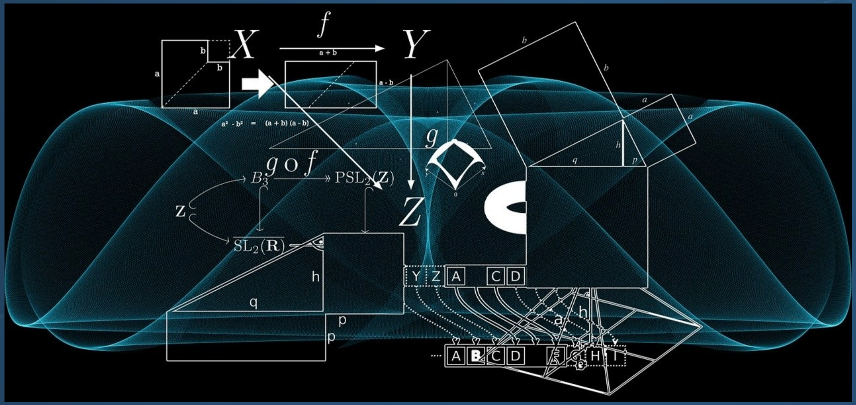
Parameterize with N.N. weights θ

Gradient ascent in performance $J\left(\theta\right)$

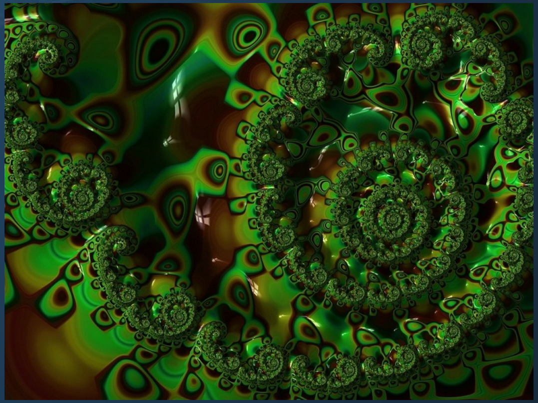
$$\boxed{\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} J\left(\theta_t\right)}$$

Policy Gradient Methods

Advantages of Policy Approximation



Policy is continuous and finite



Works for continuous action spaces

All actions sampled so no dilemma

Action Selection in Policy Gradients

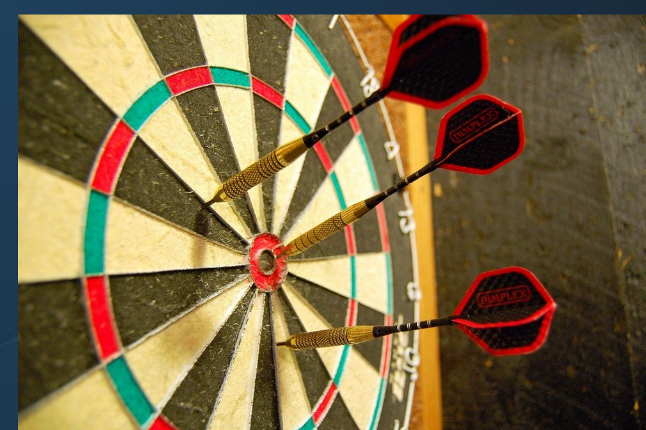
$h(s,a,\theta) \rightarrow$ Numerical preference for state action pair

$$\pi(a|s,\theta)=\frac{\exp\left(\boxed{h(s,a,\theta)}\right)}{\boxed{\sum_b \exp\left(h(s,b,\theta)\right)}}$$

Computed by N.N.

Add up to 1

Preference for profitable actions



Accurate estimates

Limitations of (Some) Policy Gradient Methods

Sample inefficiency

Credit assignment problem

Unstable solutions

May still need to learn V, Q

Policy Gradient Theorem

$J(\theta) \stackrel{\text{def}}{=} v_{\pi_\theta}(s_0)$ → value of the episode start state

Are the updates helping?

$\nabla J(\theta)$ Depends on distribution of states encountered

Policy Gradient Theorem

$$\nabla J(\theta) \propto \boxed{\sum_s \mu(s)} \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta)$$

Turns into expectation value

Can learn from experience to improve performance

Conclusion

- Approximate policy with deep N.N.
- Stochastic policy solves explore exploit dilemma
- Handles continuous action spaces
- Policy gradient theorem lets us use experience