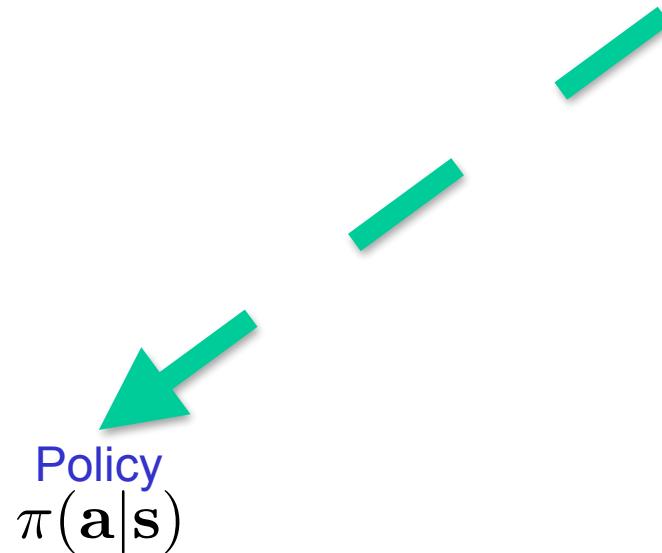

Policy-based methods

Herke van Hoof

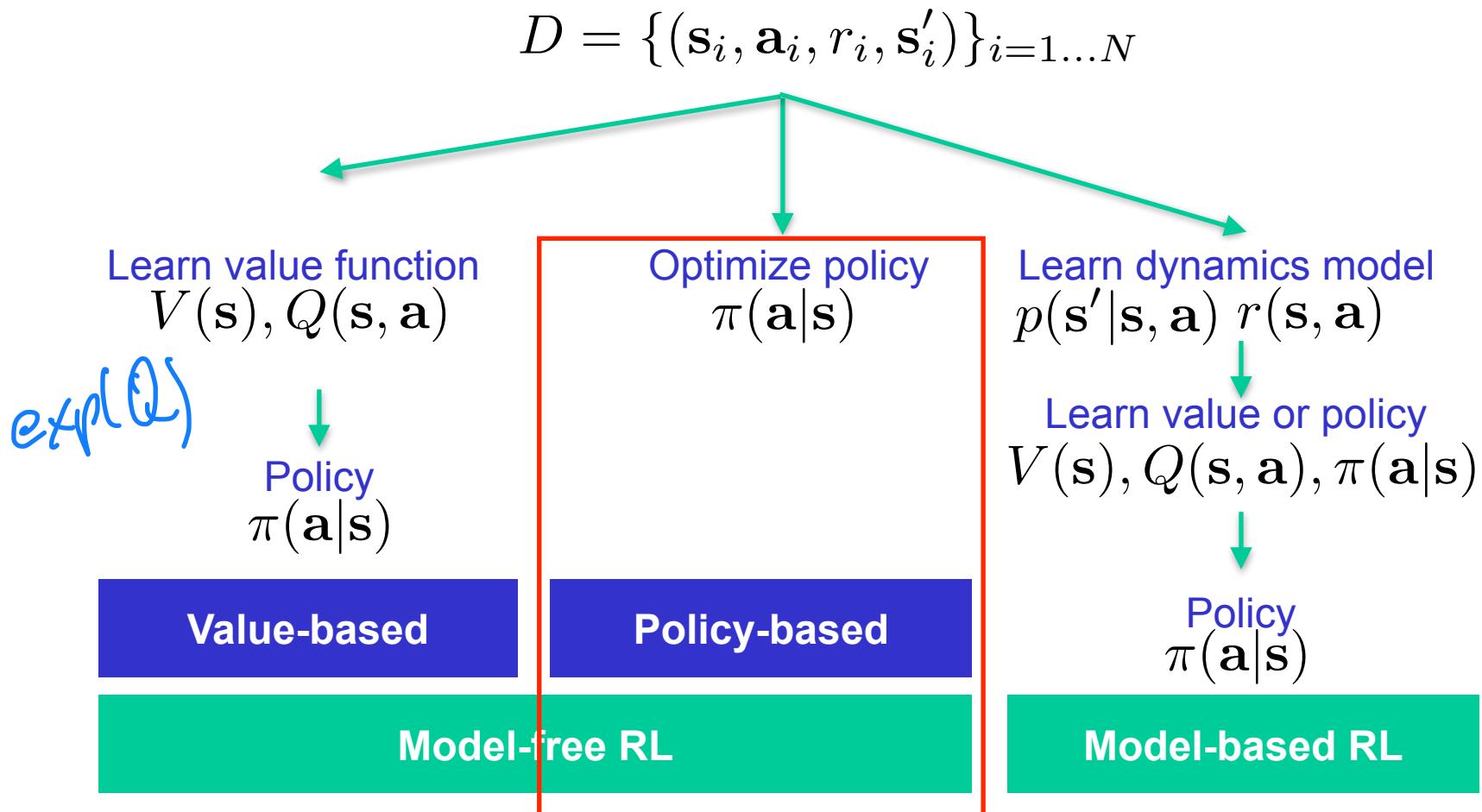
Big picture: How to learn policies

$$D = \{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{i=1\dots N}$$



Thanks to Jan Peters

Big picture: How to learn policies



Thanks to Jan Peters

Shortcomings of action-value methods

Handle continuous actions

- How can we apply the max operator efficiently?

Ensure smoothness in policies

- Sometimes, need to ensure policy does not change too much in 1 step
- We can make the stepsize for value small. But small change in value can lead to big change in policy

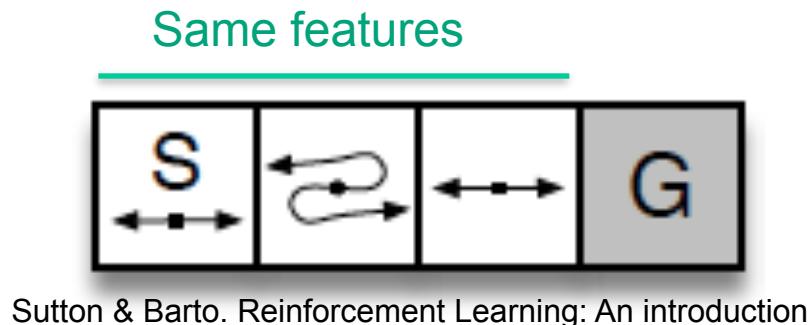
Hard to include prior knowledge about possible solutions

Can't learn stochastic policies

Shortcomings of action-value methods

Can't learn stochastic policies

- With function approximation, states might be aliased



- With ϵ greedy: will tend to get stuck at 'reversed' state
- Choosing either action with 50% probability would do better here
- No way to learn how much randomness is optimal
- We might want **stochastic** policies

Policy representation

Instead of optimising a value function that implicitly defines a policy; directly optimise the **parameters of a policy function**

Can choose policy how we want, we often need:

- A stochastic policy, that defines a distribution over all actions
- The (log)probabilities to be differentiable wrt parameters

Vlog($\pi(a|s)$)

Policy representation

Linear gaussian policy (cont. actions)

$$\mathbf{a} \sim \mathcal{N}(\boldsymbol{\theta}^T \mathbf{s}; \sigma)$$

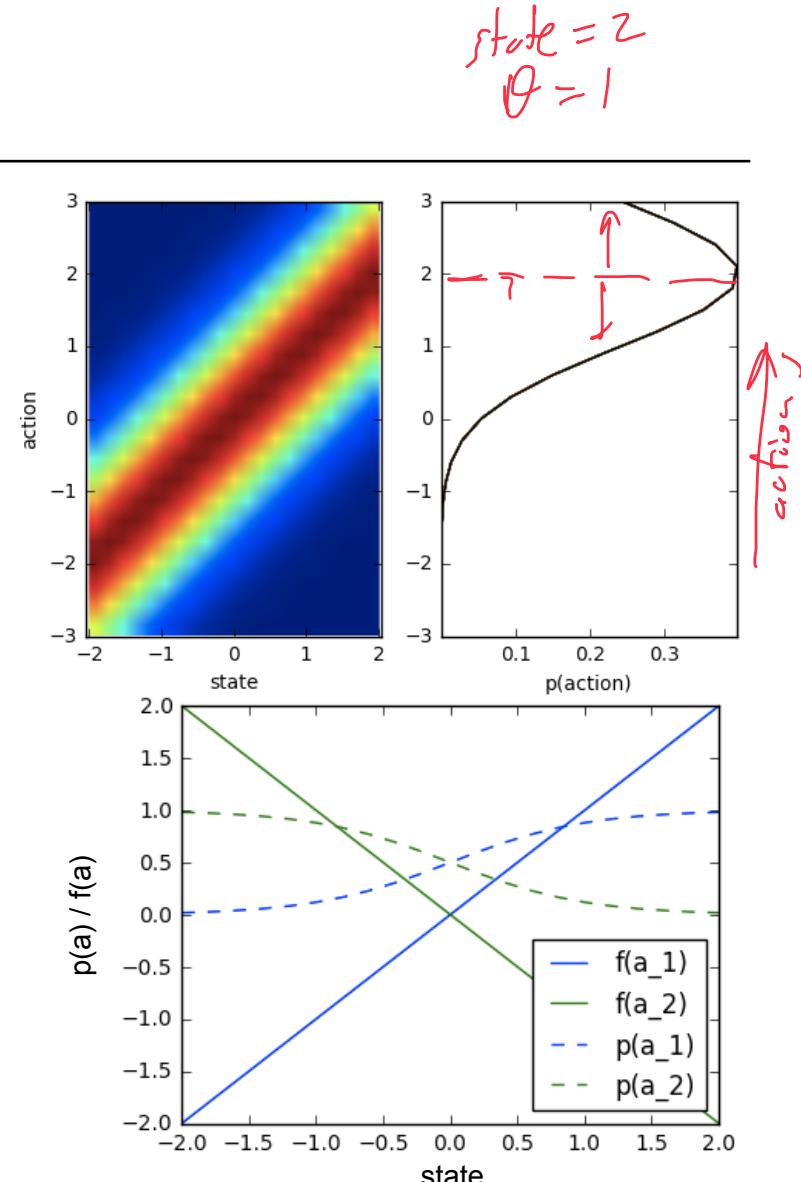
Neural network policy (cont. actions)

$$\mathbf{a} \sim \mathcal{N}(\text{NN}_{\theta_\mu}(\mathbf{s}); \text{NN}_{\theta_\sigma}(\mathbf{s}))$$

Softmax policy (discrete actions)

$$p(\mathbf{a}|\mathbf{s}) = \frac{\exp f_{\theta}(\mathbf{s}, \mathbf{a})}{\sum_{\mathbf{a}' \in \mathcal{A}} \exp f_{\theta}(\mathbf{s}, \mathbf{a}')}$$

f itself can be linear, NN, ...

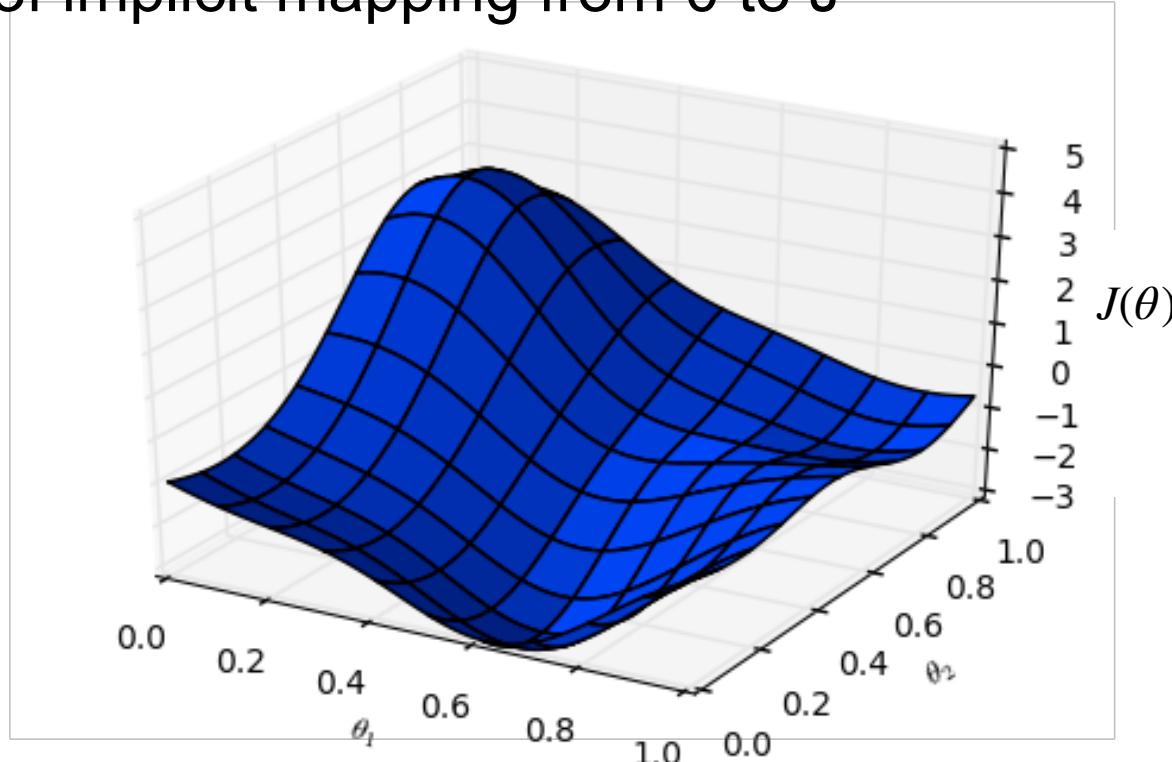


Objective

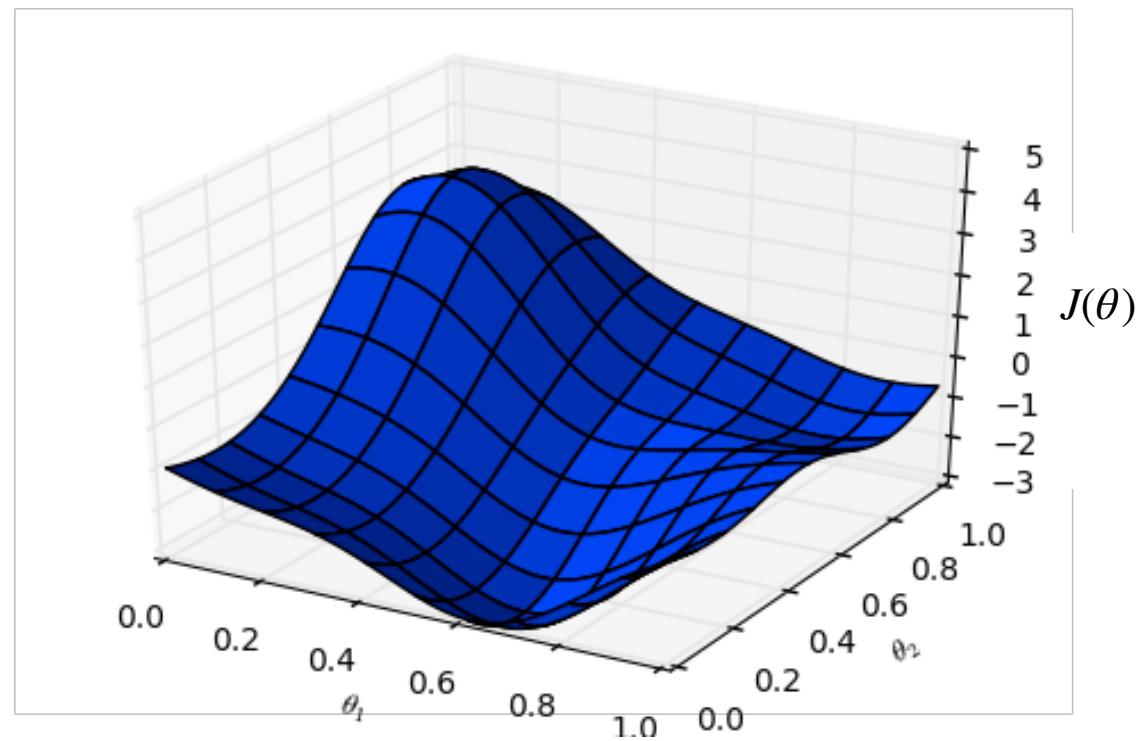
Every policy is defined by a parameter vector θ

Every policy has an expected return $J = \mathbb{E}[G]$

Can think of implicit mapping from θ to J

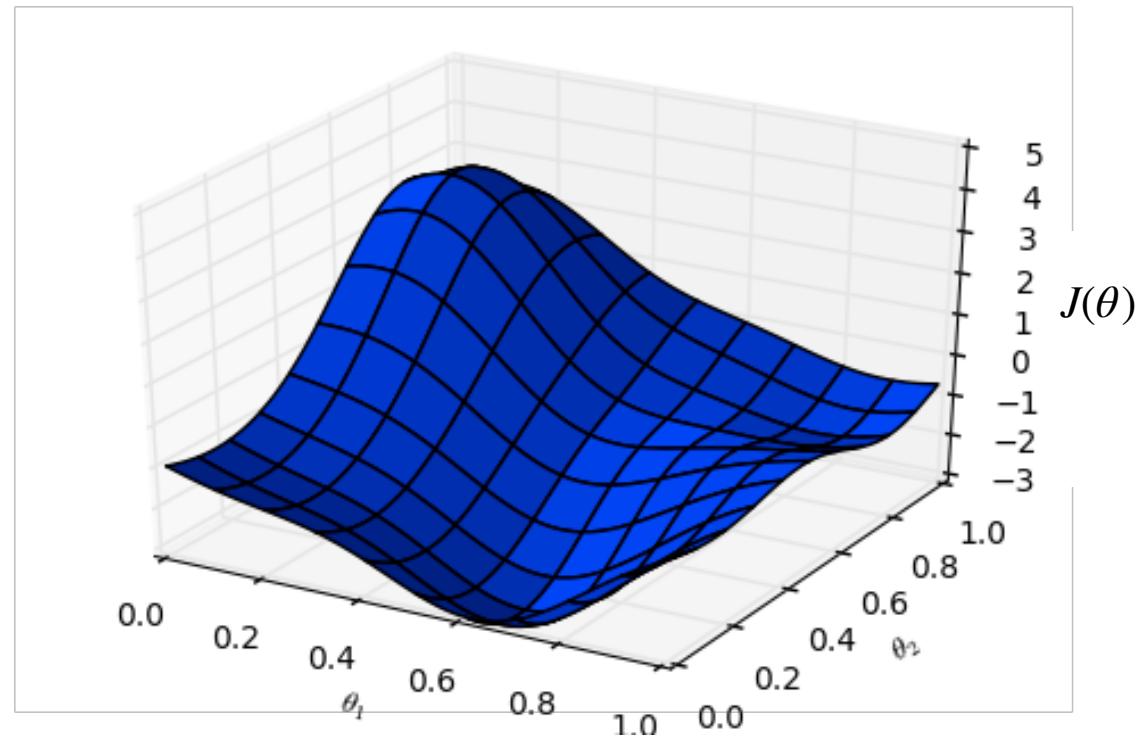


How do we find the best policy



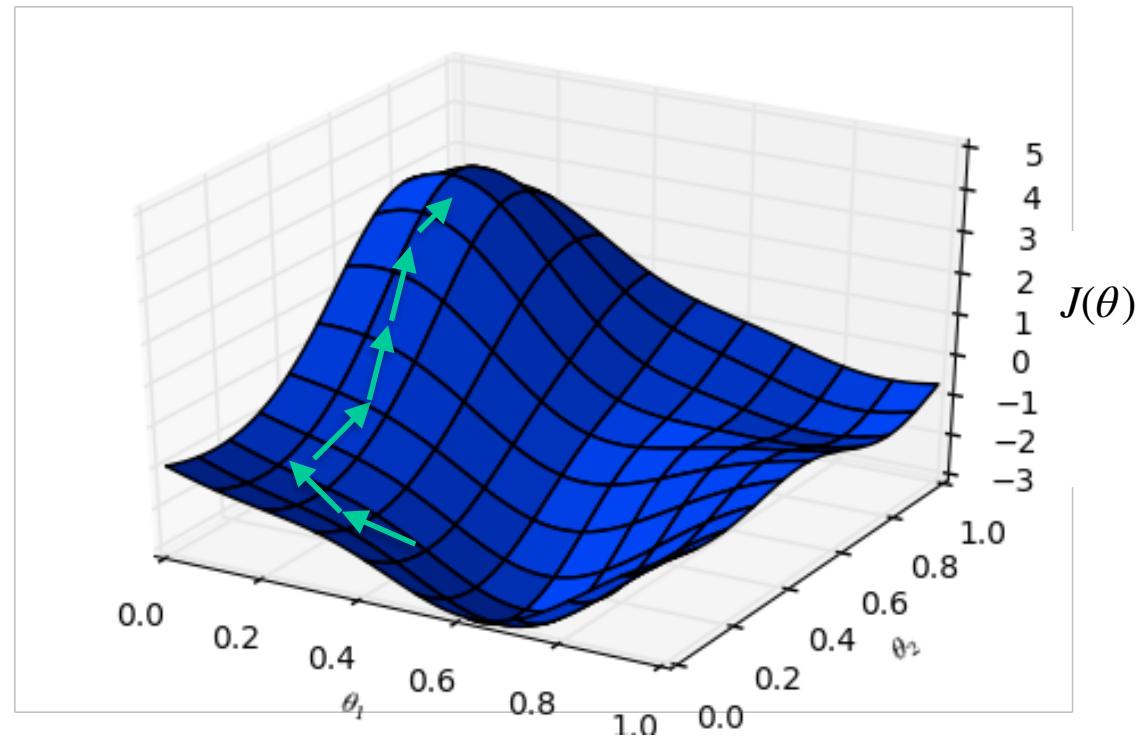
How do we find the best policy

- 0-order
 - Grid-search
 - Random search
 - Meta-heuristics



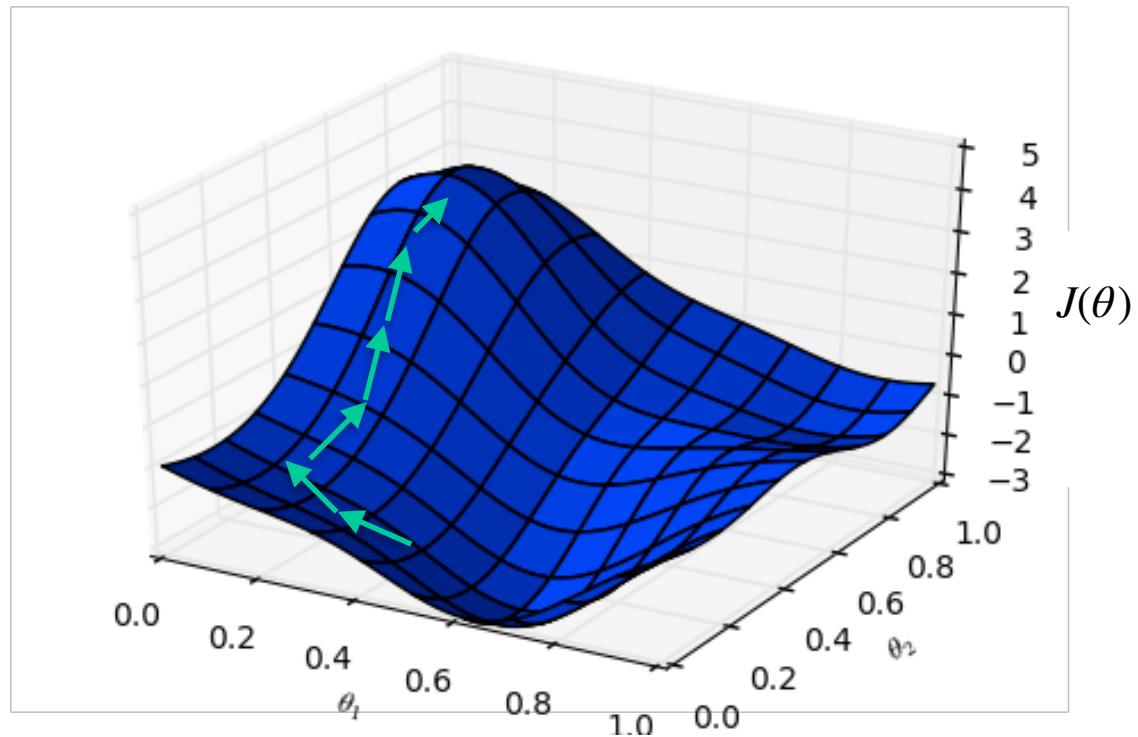
How do we find the best policy

- 0-order
 - Grid-search
 - Random search
 - Meta-heuristics
- 1st order
 - Policy Gradients
 - Today's topic



How do we find the best policy

- 0-order
 - Grid-search
 - Random search
 - Meta-heuristics
- 1st order
 - Policy Gradients
 - Today's topic
- 2nd order
 - Use information about how gradient changes
 - Next week



Finite difference gradients

Simplest update:

Consider finite difference
estimates of gradient

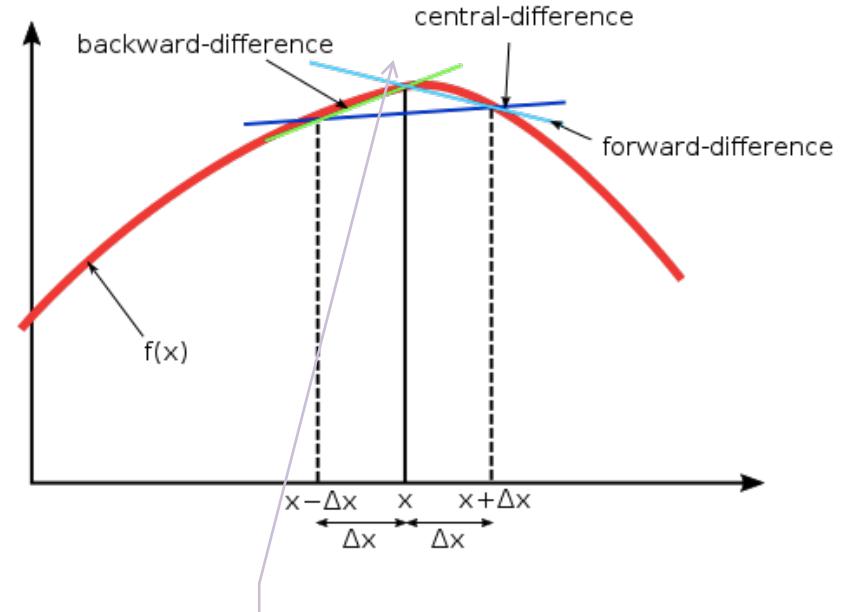
Evaluate two controllers, with
parameters $\theta + \epsilon$ and $\theta - \epsilon$

Estimate gradient as

$$\widehat{\nabla J} \approx \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

Update parameters

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J}$$



goal: find the best policy

approach: find the gradient of policy func (what policy gives the largest value. we have para in policy func)

question: how to compute the gradient? we can't find true gradient, but we can estimate it.

answer: finite difference gradient. we use a para theta, and a small step epsilo.

we compute two values: $J(\theta + \epsilon)$, $J(\theta - \epsilon)$.

两者相减得到estimated gradient.

we update theta

Finite difference gradients

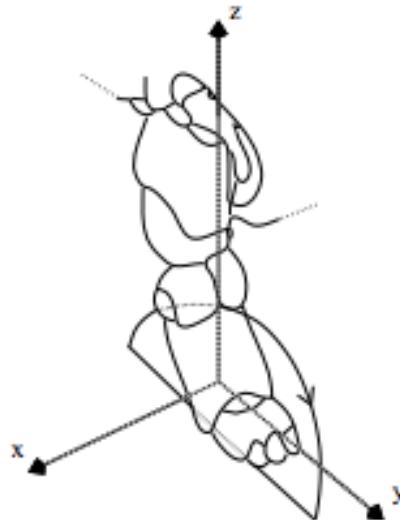


Fig. 2. The elliptical locus of the Aibo's foot. The half-ellipse is defined by length, height, and position in the x - y plane.



Kohl & Stone, Policy gradient reinforcement learning for fast quadrupedal locomotion, 2004

Finite difference gradients

General principle stays the same, always $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J}$

However: for 1 parameter, need 2 full roll-outs

For n parameters, will need $2n$ full roll-outs

If function evaluation is stochastic, gradients will be *incredibly* noisy - thus, inefficient

What can we do with policy gradients?

- handle continuous actions

Can use policy with continuous output (linear, neural net)

- ensure smoothness in policies

Small step size ensures small change in policy

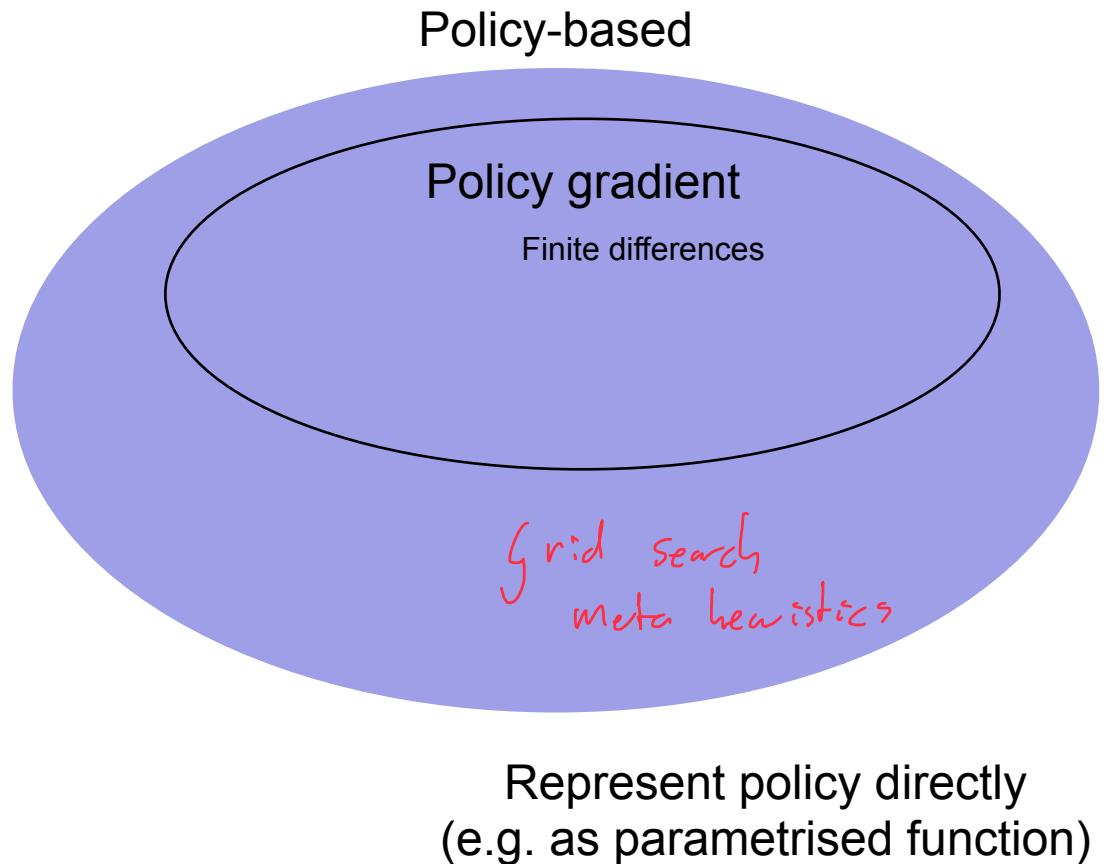
- hard to include prior knowledge about possible solutions

Include prior knowledge as policy form or initialisation

- can't learn stochastic policies

Can easily train stochastic policies

Bigger picture



Original REINFORCE

Can we calculate an **analytical** gradient for episodic problems

$$\nabla_{\theta} J = \nabla_{\theta} \mathbb{E}_{\tau} G(\tau)$$

$$= \int \nabla_{\theta} p_{\theta}(\tau) G(\tau) d\tau$$

$$1) \quad \nabla_{\theta} p_{\theta}(\tau) = \underbrace{\nabla_{\theta} p(\mathbf{s}_0)}_{?} \prod_{t=0}^T \underbrace{p(\mathbf{a}_t | \mathbf{s}_t)}_{\text{known}} \underbrace{p(\mathbf{s}_{t+1} | \mathbf{a}_t, \mathbf{s}_t)}_{?}$$

2) How to calculate the integral?

Original REINFORCE

Can we calculate an **analytical** gradient for episodic problems

$$\nabla_{\theta} J = \nabla_{\theta} \mathbb{E}_{\tau} G(\tau)$$

$$\begin{aligned} &= \int \nabla_{\theta} p_{\theta}(\tau) G(\tau) d\tau \\ &= \boxed{\int \frac{p_{\theta}(\tau)}{p_{\theta}(\tau)} \nabla_{\theta} p_{\theta}(\tau) \cdot g(\tau) d\tau} \rightarrow \nabla_{\theta} \log p_{\theta}(\tau) \\ &= \boxed{\mathbb{E}_{\tau} \left[\frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} \right] \cdot g(\tau)} \\ &= \mathbb{E} \nabla_{\theta} \log p_{\theta}(\tau) \cdot g(\tau) \\ &\quad \cancel{\nabla \log p(s_0)} + \cancel{\sum \nabla \log p(a_t | s_t)} + \cancel{\nabla \log p(s_{t+1} | s_t, a_t)} \end{aligned}$$

Original REINFORCE

$$E f(A) = \sum_{a \in D(A)} f(a) p(a)$$

$\int p(a) f(a)$

Can we calculate an **analytical** gradient for episodic problems

$$\nabla_{\theta} J = \nabla_{\theta} \mathbb{E}_{\tau} G(\tau)$$

$$= \int \nabla_{\theta} p_{\theta}(\tau) G(\tau) d\tau$$

Intuitively: make action in high-return trajectories more likely

$$= \int \frac{p_{\theta}(\tau)}{\overline{p_{\theta}(\tau)}} \nabla_{\theta} p_{\theta}(\tau) G(\tau) d\tau$$

$$= \mathbb{E}_{\tau} \left[\frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} G(\tau) \right]$$

$$= \mathbb{E}_{\tau} [\nabla \log p_{\theta}(\tau) G(\tau)] = \mathbb{E}_{\tau} \left[G(\tau) \sum_{t=0}^T \nabla_{\theta} \log p_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right]$$

$$\nabla_{\theta} \log p_{\theta}(\tau) = \underbrace{\nabla_{\theta} \log p(\mathbf{s}_0)}_{=0} + \sum_{t=0}^T \underbrace{\nabla_{\theta} \log p(\mathbf{a}_t | \mathbf{s}_t)}_{\text{known}} + \underbrace{\nabla_{\theta} \log p(\mathbf{s}_{t+1} | \mathbf{a}_t, \mathbf{s}_t)}_{=0}$$

Original REINFORCE

As before, we can replace the expectation operator with an average to get a practical algorithm:

Sample N trajectories (N=1 is possible)

$$\widehat{\nabla_{\theta} J} = \frac{1}{N} \sum_{i=1}^N \left[G(\tau_i) \sum_{t=0}^T \nabla_{\theta} \log p_{\theta}(\mathbf{a}_t^i | \mathbf{s}_t^i) \right]$$

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J}$$

Gradient estimates are **unbiased** and **consistent**
Easy to implement!

Original REINFORCE

The REINFORCE gradient uses knowledge of the policy.

- This makes it more efficient than finite differences

However

- Policy needs to be differentiable!
- Efficiency is still low!

Inefficient calculation of gradient

Consider a Bernoulli policy with parameter $\theta = \underline{0.5}$

Say, $r(a=0) = \cancel{1}$ and $r(a=1) = \cancel{0}$ *constant -2*

$$\begin{aligned} p(a=0) &= \cancel{\frac{1}{2}} \\ p(a=1) &= \theta \end{aligned}$$

$$\nabla_{\theta} \log(1-\theta)$$
$$\frac{1}{1-\theta} \cdot -1$$

$$\text{observe } \underline{a=0}, \underline{r=1}: \quad \widehat{\nabla J} = \frac{1}{1-\theta} \cdot -1 \cdot \cancel{0} = \cancel{-2} + 2$$

$$\nabla \log \theta$$

$$\text{observe } \underline{a=1}, \underline{r=3}: \quad \widehat{\nabla J} = \frac{1}{\theta} \cdot \cancel{3} \cdot 1 = \cancel{6} + 2$$

$$\frac{1}{\theta}$$

Inefficient calculation of gradient

Consider a Bernoulli policy with parameter $\theta=0.5$

Say, $r(a=0) = 1$ and $r(a=1) = 3$

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a})} r(\mathbf{a}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a})} [\nabla \log \pi_{\boldsymbol{\theta}}(\mathbf{a}) r(\mathbf{a})]$$

observe $a=0, r=1$: $\widehat{\nabla J} = \frac{1}{1-\theta} \cdot -1 \cdot 1 = -2$

observe $a=1, r=3$:

$$\widehat{\nabla J} = \frac{1}{\theta} \cdot 3 = 6$$

Expected value: +2, Variance: 16

Inefficient calculation of gradient

Consider a Bernoulli policy with parameter $\theta=0.5$

Say, $r(a=0) = \cancel{-1}$ and $r(a=1) = \cancel{+3}$ subtract a constant

$$\nabla_{\theta} \mathbb{E}_{\pi_{\theta}(\mathbf{a})} r(\mathbf{a}) = \mathbb{E}_{\pi_{\theta}(\mathbf{a})} [\nabla \log \pi_{\theta}(\mathbf{a}) r(\mathbf{a})]$$

observe $a=0, r=1$: $\widehat{\nabla J} = \frac{1}{1-\theta} \cdot -1 \cdot \cancel{-1} = \underline{-2}$

observe $a=1, r=3$:

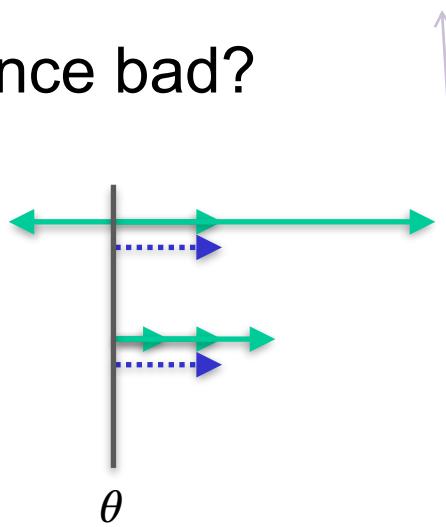
$$\widehat{\nabla J} = \frac{1}{\theta} \cdot \cancel{3} = \underline{6}$$

Expected value: +2, Variance: $\cancel{16}$
Same expected value, but now 0 variance!

Inefficient calculation of gradient

Why is variance bad?

estimator 1



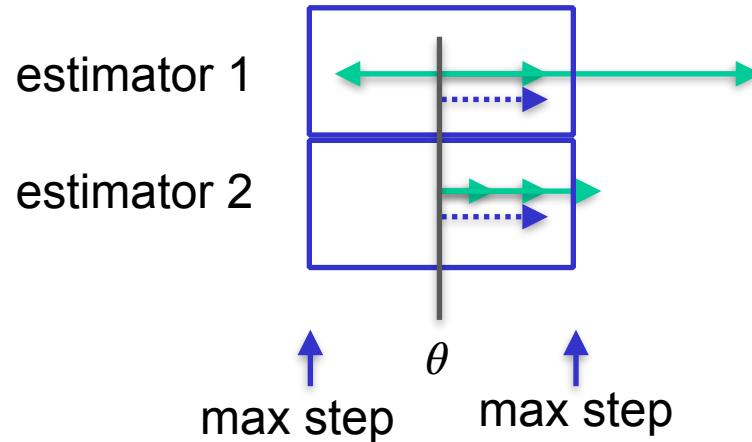
without using baseline, the variance of gradient is large.

with using baseline, the variance is low

these 2 methods, will have the same mean of gradient, but have different variance.

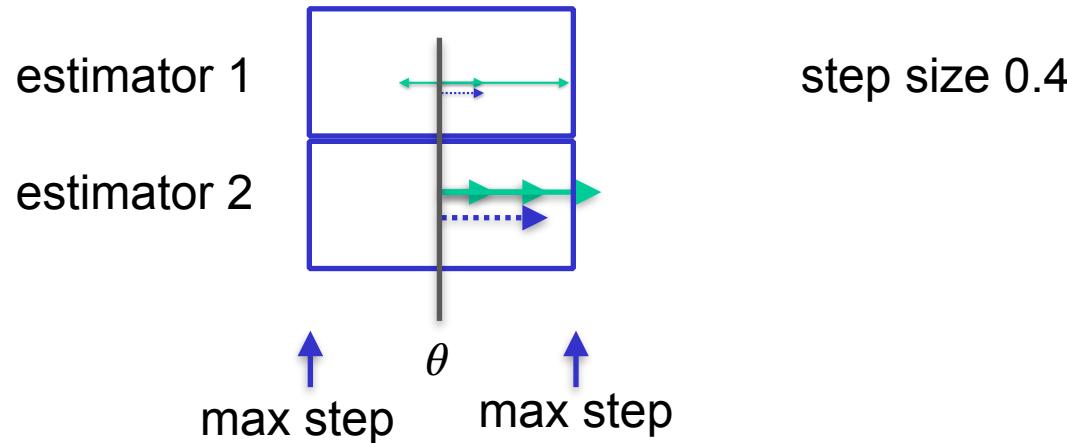
Inefficient calculation of gradient

Why is variance bad?



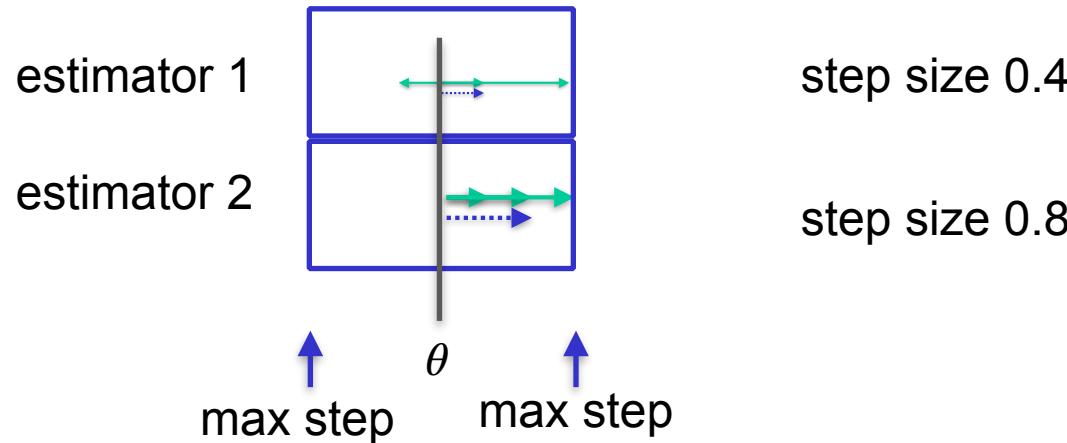
Inefficient calculation of gradient

Why is variance bad?



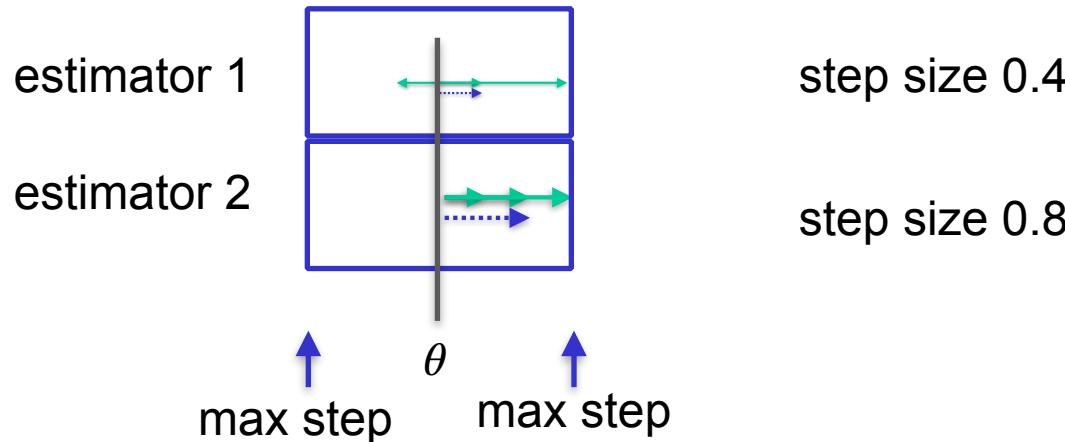
Inefficient calculation of gradient

Why is variance bad?



Inefficient calculation of gradient

Why is variance bad?



- With lower variance, can probably use a higher learning rate

Inefficient calculation of gradient

Does average indeed not change?

$$\begin{aligned}\nabla J &= \mathbb{E}_\tau [(G(\tau) - b) \nabla \log p(\tau)] \\ &= \underbrace{\mathbb{E}_\tau [G(\tau) \nabla \log p(\tau)]}_{=\nabla J} - \mathbb{E}_\tau [b \nabla \log p(\tau)]\end{aligned}$$

$$\begin{aligned}\mathbb{E}_\tau [b \nabla \log p(\tau)] &= b \int p(t) \nabla \log p(t) dt \\ &\quad b \int p(t) \frac{\nabla p(t)}{p(t)} dt = b \cdot \nabla \int p(t) dt\end{aligned}$$

$$\begin{aligned}&\quad \boxed{1} \\ &= 0.\end{aligned}$$

考试考了 prove it is still unbiased, after adding baseline. prove this part = 0

sum of all prob of each trajectory must = 1

1h25min

Inefficient calculation of gradient

Does average indeed not change?

$$\mathbb{E}_\tau [(G(\tau) - b)\nabla \log p(\tau)]$$

$$= \underbrace{\mathbb{E}_\tau [G(\tau)\nabla \log p(\tau)]}_{=\nabla J} - \mathbb{E}_\tau [b\nabla \log p(\tau)]$$

$$\mathbb{E}_\tau [b\nabla \log p(\tau)] = b \int p(\tau) \nabla \log p(\tau) d\tau$$

$$= b \int p(\tau) \frac{\nabla p(\tau)}{p(\tau)} d\tau = b \nabla \underbrace{\int p(\tau) d\tau}_{=1} = 0$$

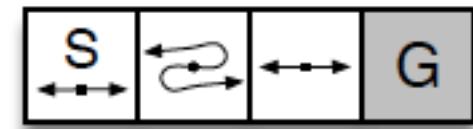
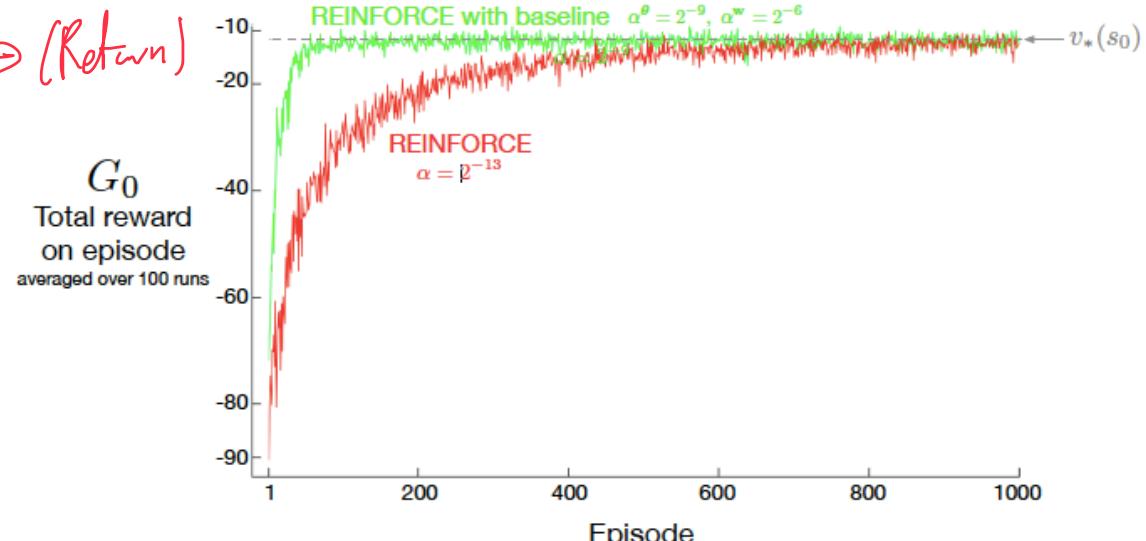
Thus, the gradients are still **unbiased**

We'll look at the **variance** in the exercises

Inefficient calculation of gradient

A good baseline is the expected reward (e.g. observed average)

Baselines can depend on state. Then, a value function estimate is a popular choice.



Sutton & Barto. Reinforcement Learning: An introduction.

Inefficient credit assignment

		1	
	→	→	-1
	↑		
	↑	←	←
			-1

		1	
	→	↑	-1
	↑		
	↑	←	↓
		↑	← -1

Reinforce with baseline

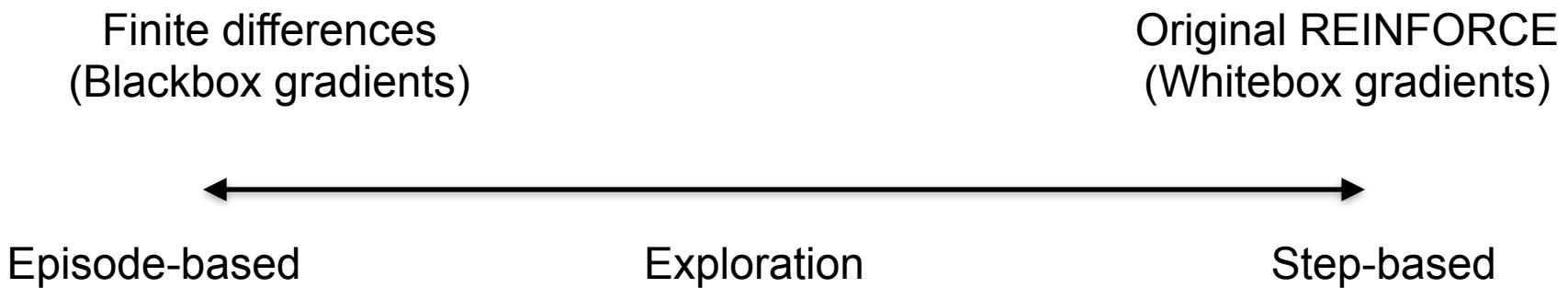
The REINFORCE gradient uses knowledge of the policy.

- This makes it more efficient than finite differences

However

- Policy needs to be differentiable!
- Efficiency is still low!
- Baseline improves variance a bit
- All actions get credit for all rewards in a trajectory
 - Good action in end gets punished for bad action in beginning
- Variance from stochastic transitions increases with T

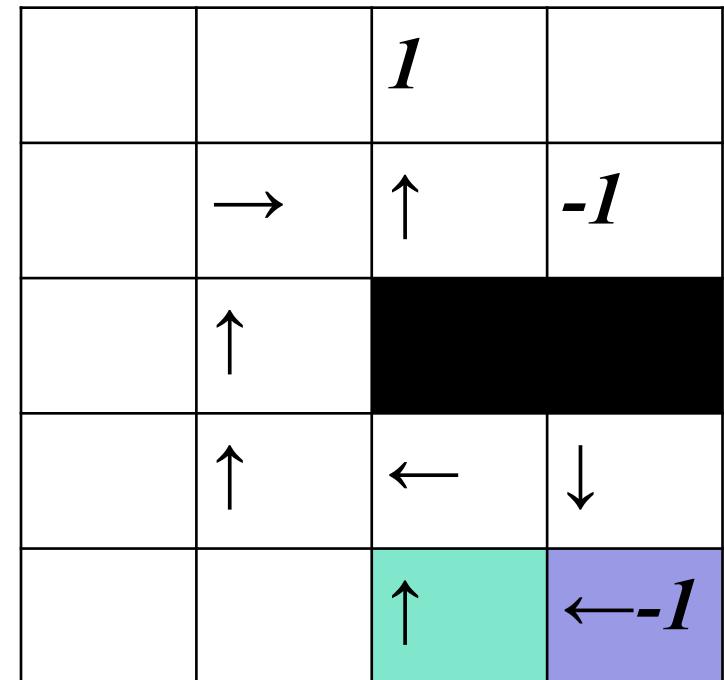
Comparison



Reinforce / G(PO)MDP

Intuitively, we can't conclude whether the **green** action was good or bad based on the **blue** reward.

Can we use this to come up with a better update?



Reinforce / G(PO)MPD

$$\begin{aligned}
 \nabla J &= \mathbb{E}_\tau \left[\sum_{t=1}^T r_t \sum_{t'=1}^T \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right] \\
 &= \sum_{t=1}^T \mathbb{E}_{\tau_{1:t}} \mathbb{E}_{\tau_{t+1:T}} \left[r_t \left(\sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) + \sum_{t'=t+1}^T \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right) \middle| \tau_{1:t} \right] \\
 &= \sum_{t=1}^T \mathbb{E}_{\tau_{1:t}} \left[r_t \left(\sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) + \mathbb{E}_{\tau_{t+1:T}} \left[\sum_{t'=t+1}^T \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \middle| \tau_{1:t} \right] \right) \right] \\
 &\quad - \sum_{t'} \mathbb{E}_{s_{t'}} \left[\int p(a|s) \frac{\nabla p(a'|s')}{p(a'|s')} da \right] \\
 &\quad \sum_{t'} \mathbb{E}_{s_{t'}} \nabla \int p(a|s') da \\
 &\quad \sum_{t'} \mathbb{E}_{s_{t'}} I
 \end{aligned}$$

Reinforce / G(PO)MPD

$$\begin{aligned}
\nabla J &= \mathbb{E}_\tau \left[\sum_{t=1}^T r_t \sum_{t'=1}^T \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right] \\
&= \sum_{t=1}^T \mathbb{E}_{\tau_{1:t}} \mathbb{E}_{\tau_{t+1:T}} \left[r_t \left(\sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) + \sum_{t'=t+1}^T \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right) \middle| \tau_{1:t} \right] \\
&= \sum_{t=1}^T \mathbb{E}_{\tau_{1:t}} \left[r_t \left(\sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) + \underbrace{\mathbb{E}_{\tau_{t+1:T}} \left[\sum_{t'=t+1}^T \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \middle| \tau_{1:t} \right]}_{= \sum_{t'} \mathbb{E}_{s_{t'}} \mathbb{E}_{a_{t'}} \left[\frac{\nabla p(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{p(\mathbf{a}_{t'} | \mathbf{s}_{t'})} | \mathbf{s}_{t'} \right]} \right) \right] \\
&= \mathbb{E}_\tau \left[\sum_{t=1}^T r_t \sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right]
\end{aligned}$$

Reinforce / G(PO)MPD

$$\mathbb{E}_\tau \left[\sum_{t=1}^T r_t \sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right] \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T r_t \sum_{t'=1}^t \nabla \log p(\mathbf{a}_{t'} | \mathbf{s}_{t'}) \right)$$

Replacing the expected value by an average again creates an implementable algorithm

Reinforce/GPOMDP with baseline

REINFORCE with Baseline (episodic), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Input: a differentiable state-value function parameterization $\hat{v}(s, w)$

Algorithm parameters: step sizes $\alpha^{\theta} > 0$, $\alpha^w > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to 0)

Loop forever (for each episode):

Learning together with v-function

Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot | \cdot, \theta)$

Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad (G_t)$$

$$\delta \leftarrow G - \hat{v}(S_t, w)$$

$$w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S_t, w)$$

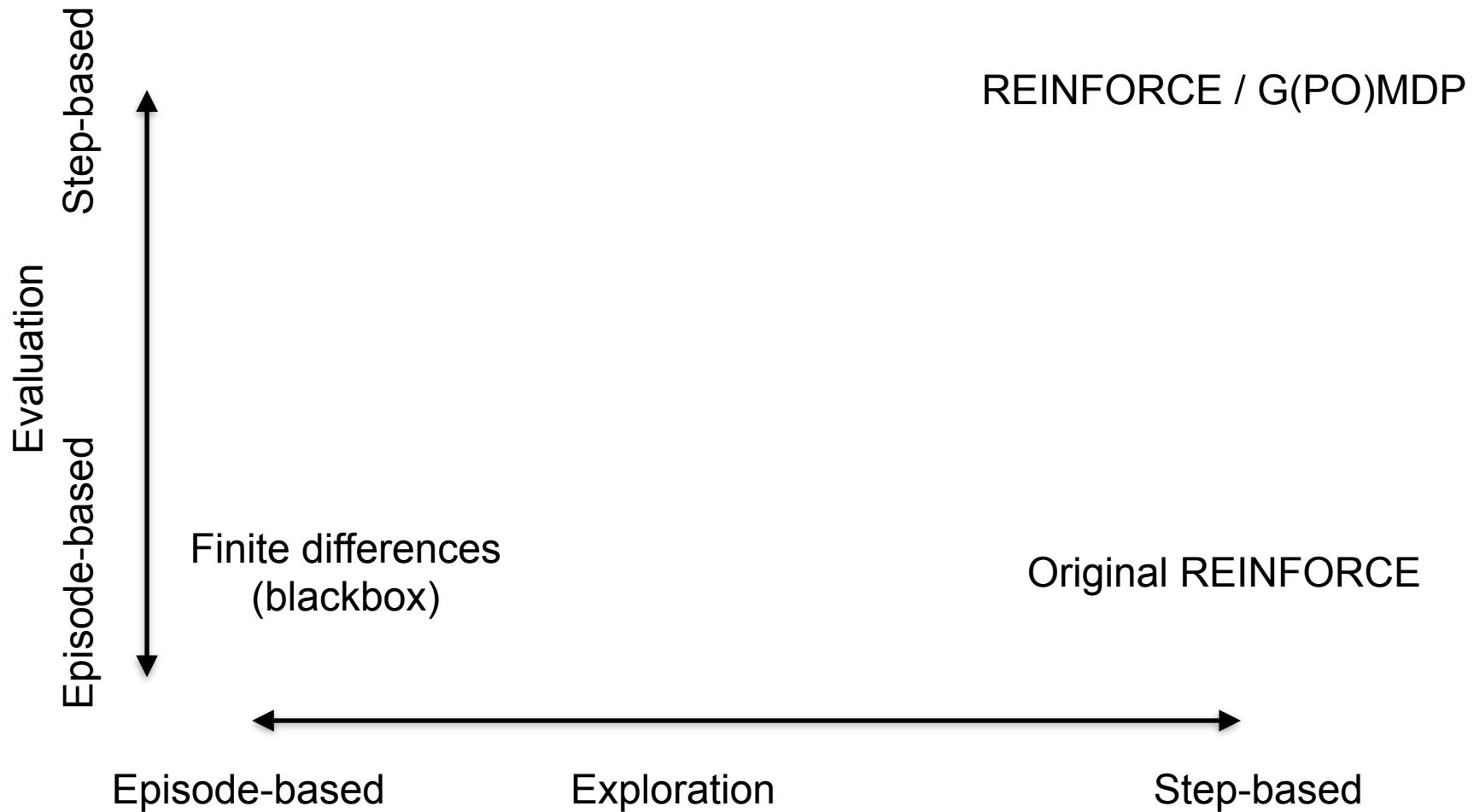
$$\theta \leftarrow \theta + \alpha^{\theta} \gamma^t \delta \nabla \ln \pi(A_t | S_t, \theta) \quad \text{Policy update}$$

V-fc update

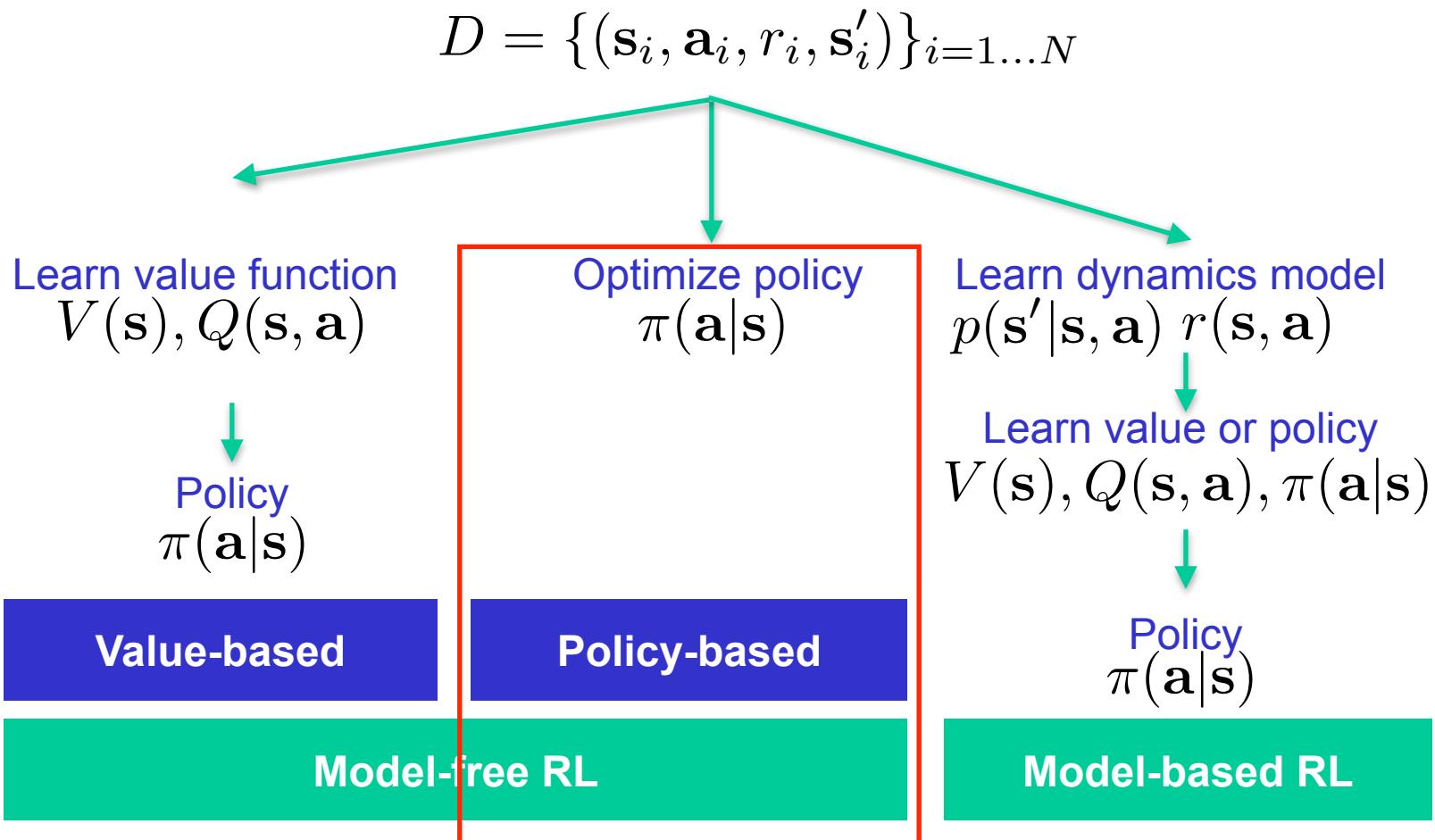
Reinforce/GPOMDP with baseline

- Still easy to implement
- Cut one source of variance compared to REINFORCE
- Not changing any expected value: still **consistent, unbiased**
- Can be combined with baseline (e.g., learned value f_c)
- Still: variance from stochastic transitions increases with T

Comparison



Big picture: How to learn policies



Thanks to Jan Peters

Advantages of policy-based methods

- handle continuous actions

Can use policy with continuous output (linear, neural net)

- ensure smoothness in policies

- Setting small step size will ensure change are small in each step
- We'll see better methods next week

- hard to include prior knowledge about possible solutions

Include prior knowledge as policy form or initialisation

- can't learn stochastic policies

Can easily train stochastic policies

Weaknesses of policy-based methods

actor only methods: we dont use value func

Actor-only methods have high variance from Monte-Carlo

A lot of the methods we discussed are specific to episodic setting

Requires stochastic policies, what if deterministic is optimal?

- If amount of randomness is learned, can get close to deterministic
- We will also see a policy gradient method to learn deterministic policies

Thanks for your attention!

Feedback?

h.c.vanhoof@uva.nl