POLICY SEARCH METHODS 3

Herke van Hoof

STAYING CLOSE TO PREVIOUS POLICIES

- Small policy update steps tends to be more ‘safe’
- E.g., value / return might be wrong/noisy, don’t fully trust it

STAYING CLOSE TO PREVIOUS POLICIES

- Small policy update steps tends to be more ‘safe’
- E.g., value / return might be wrong/noisy, don’t fully trust it
- We can think of a policy update as follows:
- Current parameters θ
- New parameters θ' obtained by adding adding an offset x
- $\theta' = \theta + x$
- Find x s.t. θ' is expected to have a high reward and difference between π_θ and π'_θ is ‘small’

STAYING CLOSE TO PREVIOUS POLICIES

- Find x s.t. θ' is expected to have a high reward and difference between θ and θ' is 'small'
- One possibility: limit l2 norm

$$\mathbf{x}^* = \max_{\mathbf{x}} J\left(\boldsymbol{\theta} + \mathbf{x}\right) \qquad \text{s.t. } \mathbf{x}^T\mathbf{x} = c$$

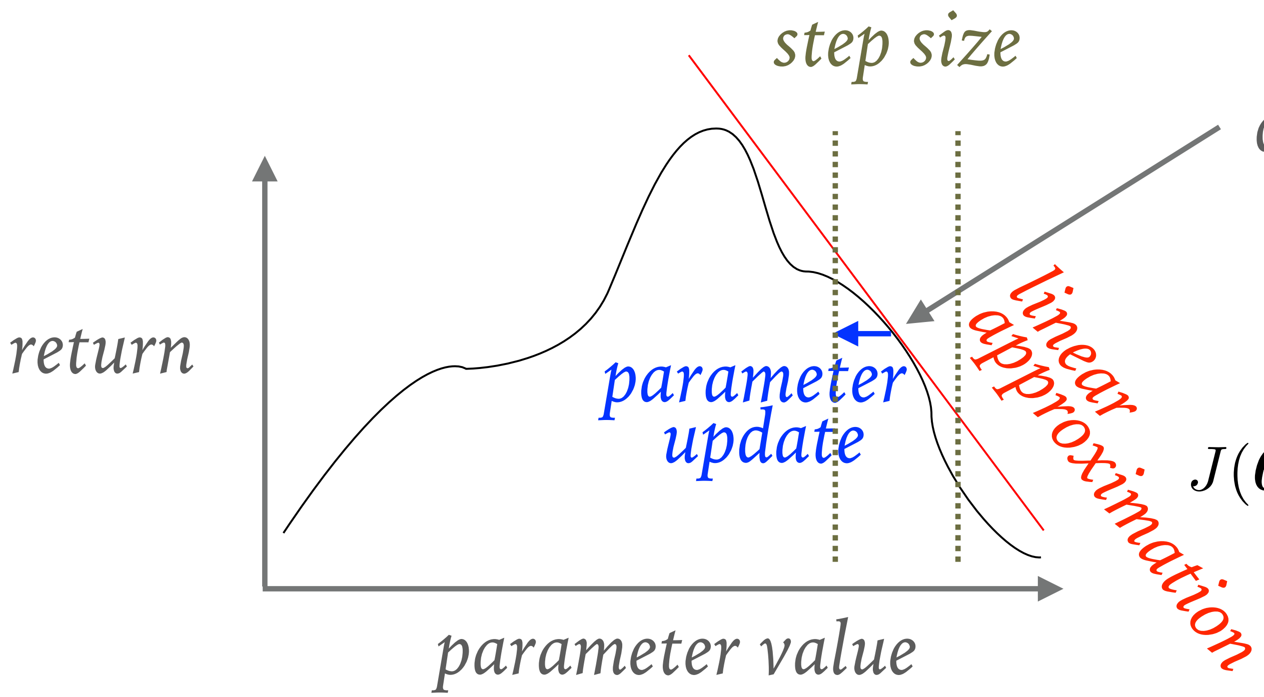
STAYING CLOSE TO PREVIOUS POLICIES

- Find x s.t. θ' is expected to have a high reward and difference between θ and θ' is 'small'
- One possibility: limit l2 norm

$$
\begin{aligned}
\mathbf{x}^* &= \max_{\mathbf{x}} J\left(\boldsymbol{\theta} + \mathbf{x}\right) && \text{s.t. } \mathbf{x}^T \mathbf{x} = c \\
&\approx \max_{\mathbf{x}} J\left(\boldsymbol{\theta}\right) + \left(\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right)\right)^T \mathbf{x} && \text{s.t. } \mathbf{x}^T \mathbf{x} = c \\
&\propto \nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right)
\end{aligned}
$$

- Standard ('vanilla') policy gradients finds direction of most improvement per unit l2 distance in parameters

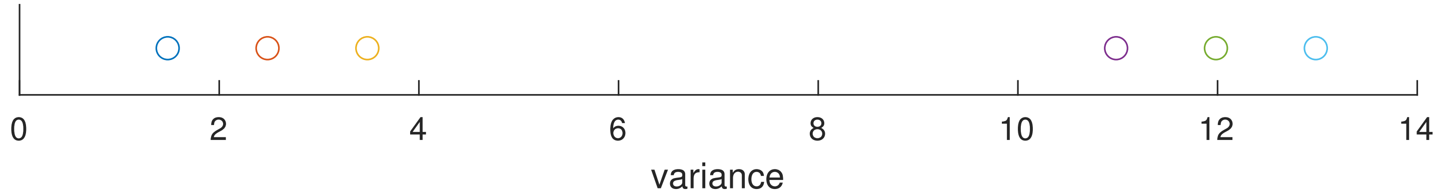
IN A PICTURE



$$J(\boldsymbol{\theta} + \mathbf{x}) \approx J(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))^T \mathbf{x}$$

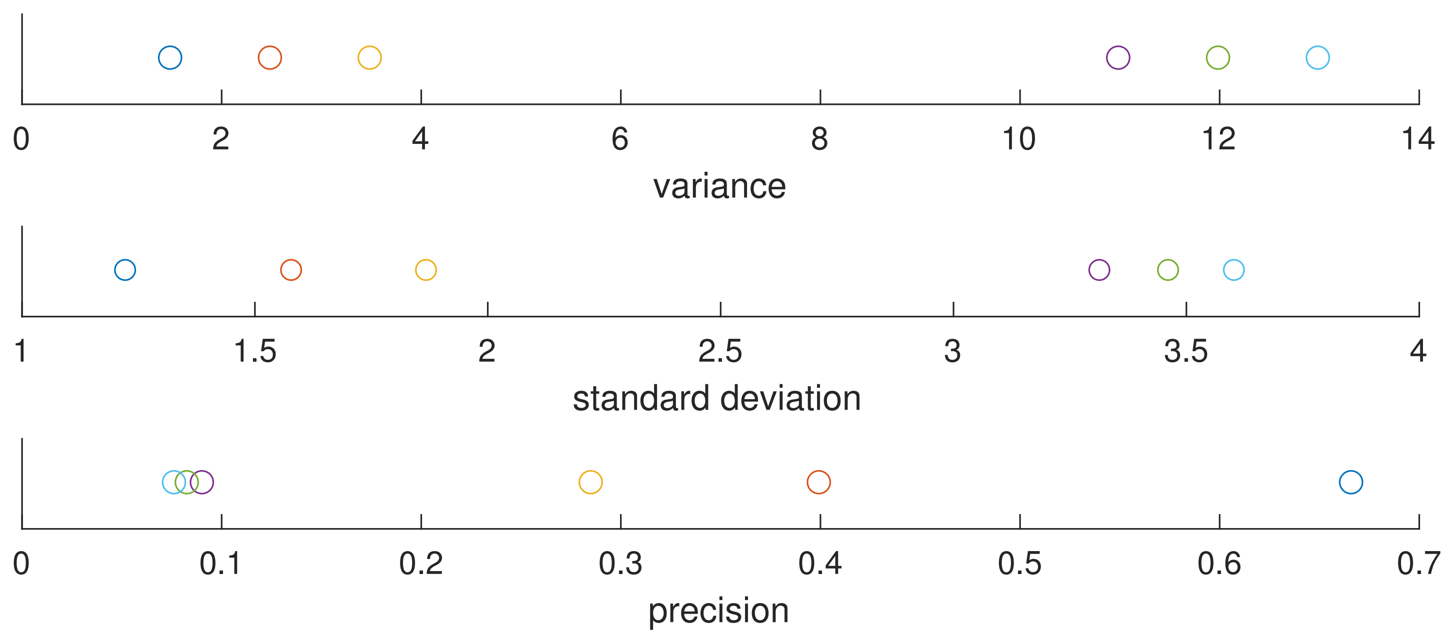
STAYING CLOSE TO PREVIOUS POLICIES

- Standard ('vanilla') policy gradients finds direction of most improvement per unit l2 distance in parameters
- This norm is sensitive to parametrisation:



STAYING CLOSE TO PREVIOUS POLICIES

- Standard ('vanilla') policy gradients finds direction of most improvement per unit l2 distance in parameters
- This norm is sensitive to parametrisation:



- How to express policy closeness covariantly?
- (covariant: independent of choice of parametrisation)

STAYING CLOSE TO PREVIOUS POLICIES

- Why do we want covariant norm (invariant to parametrisation)?
  - Don't waste time tuning parametrisation
  - Parameters with different 'meaning': mean and precision
    - does a norm in this space make sense?
    - step size never right on all parameters if scale different (have to take step small enough for most sensitive direction)
  - Correlations between parameters ignored (feature modulated by more parameters easier to change)

STAYING CLOSE TO PREVIOUS POLICIES

- Why do we want covariant norm (invariant to parametrisation)?
  - Don't waste time tuning parametrisation
  - Parameters with different 'meaning': mean and precision
    - does a norm in this space make sense?
    - step size never right on all parameters if scale different (have to take step small enough for most sensitive direction)
  - Correlations between parameters ignored (feature modulated by more parameters easier to change)
- Conceptually, it's not the change in parameters we care about!
  - Limit change in trajectories, states, and/or actions?

STAYING CLOSE TO PREVIOUS POLICIES

- How to express policy closeness covariantly?
- Kullback-Leibler (KL) divergence is information-theoretic quantification of difference between distributions

$$D_{\mathrm{KL}}(\pi \| \pi') = \int_{-\infty}^{\infty} \pi(a) \log \frac{\pi(a)}{\pi'(a)} \mathrm{d}a$$

- Minimal value of 0 when $\pi = \pi'$
- KL is invariant under parameter transformations
- Idea: find direction of maximal improvement per unit of KL between policies

STAYING CLOSE TO PREVIOUS POLICIES

- Idea: find direction of maximal improvement per unit of KL between policies
- Several algorithms can be understood using this idea
  - Natural policy gradient
  - Trust region policy optimization (TRPO)

NATURAL POLICY GRADIENT

- Idea: make policy gradients covariant [Kakade 2002]
- this yields an algorithm that exploits structure of parameters
- Here, will look how it relates to KL [Bagnell 2003]

[Kakade 2002, Bagnell 2003]

NATURAL POLICY GRADIENT

- Recall vanilla policy gradients

$$
\begin{aligned}
\mathbf{x}^* &= \max_{\mathbf{x}} J(\boldsymbol{\theta} + \mathbf{x}) && \text{s.t. } \mathbf{x}^T\mathbf{x} = c \\
&\approx \max_{\mathbf{x}} J(\boldsymbol{\theta}) + \left(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\right)^T \mathbf{x} && \text{s.t. } \mathbf{x}^T\mathbf{x} = c \\
&\propto \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})
\end{aligned}
$$

- replace constraint by quadratic expansion of KL divergence

$$
\begin{aligned}
c &= \mathbb{E}_{\mathbf{s}}\left[D_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}) \| \pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}+\mathbf{x})\right] = \mathrm{EKL}(\mathbf{x}) \\
&\approx \mathrm{EKL}(\boldsymbol{\theta}) + \mathbf{x}^T \nabla_{\mathbf{x}} \mathrm{EKL}(\mathbf{x}) + \frac{1}{2}\mathbf{x}^T \left(\nabla_{\mathbf{x}}^2 \mathrm{EKL}(\mathbf{x})\right) \mathbf{x}
\end{aligned}
$$

(gradients and Hessians evaluated at $\boldsymbol{x} = 0$)

NATURAL POLICY GRADIENT

- Recall vanilla policy gradients

$$\begin{aligned} \mathbf{x}^* &= \max_{\mathbf{x}} J(\boldsymbol{\theta} + \mathbf{x}) && \text{s.t. } \mathbf{x}^T \mathbf{x} = c \\ &\approx \max_{\mathbf{x}} J(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))^T \mathbf{x} && \text{s.t. } \mathbf{x}^T \mathbf{x} = c \\ &\propto \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \end{aligned}$$

- replace constraint by quadratic expansion of KL divergence

$$\begin{aligned} c &= \mathbb{E}_{\mathbf{s}}\left[D_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})\|\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}+\mathbf{x})\right] = \mathrm{EKL}(\mathbf{x}) \\ &\approx \mathrm{EKL}(\boldsymbol{\theta}) + \mathbf{x}^T \nabla_{\mathbf{x}} \mathrm{EKL}(\mathbf{x}) + \frac{1}{2}\mathbf{x}^T \left(\nabla^2_{\mathbf{x}} \mathrm{EKL}(\mathbf{x})\right) \mathbf{x} \\ &= 0 \qquad\qquad\qquad\qquad\qquad\qquad + \frac{1}{2}\mathbf{x}^T \left(\nabla^2_{\mathbf{x}} \mathrm{EKL}(\mathbf{x})\right) \mathbf{x} \end{aligned}$$

- since minimal value of 0 is reached if parameter doesn’t change

NATURAL POLICY GRADIENT

$$c = \mathbb{E}_{\mathbf{s}}\left[D_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})\|\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}+\mathbf{x})\right] = \mathrm{EKL}(\mathbf{x})$$
$$= \frac{1}{2}\mathbf{x}^T\left(\nabla^2_{\mathbf{x}}\mathrm{EKL}(\mathbf{x})\right)\mathbf{x}$$

- This is the squared norm with respect to matrix

$$F = \nabla^2_{\mathbf{x}}\mathrm{EKL} = \mathbb{E}_s\left[\nabla^2_{\mathbf{x}}D_{\mathrm{KL}}\left(\pi\left(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}\right)\|\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}+\mathbf{x})\right)\right]$$

- F can be shown to be the expected Fisher information matrix of the policy
- F characterises information about parameters in action

[Kakade 2002, Bagnell 2003]

NATURAL POLICY GRADIENT

- Consider now the modified optimisation problem

$$\begin{aligned}\mathbf{x}^* &= \max_{\mathbf{x}} J(\boldsymbol{\theta} + \mathbf{x}) && \text{s.t. } 2\mathbf{x}^T F \mathbf{x} = c \\ &\approx \max_{\mathbf{x}} J(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))^T \mathbf{x} && \text{s.t. } 2\mathbf{x}^T F \mathbf{x} = c\end{aligned}$$

- solve constraint optimisation problem: Lagrangian

- At optimality, partial derivatives of L are 0

[Kakade 2002, Bagnell 2003]

NATURAL POLICY GRADIENT

- Consider now the modified optimisation problem

$$\begin{aligned} \mathbf{x}^* &= \max_{\mathbf{x}} J(\boldsymbol{\theta} + \mathbf{x}) && \text{s.t. } 2\mathbf{x}^T F \mathbf{x} = c \\ &\approx \max_{\mathbf{x}} J(\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))^T \mathbf{x} && \text{s.t. } 2\mathbf{x}^T F \mathbf{x} = c \end{aligned}$$

- solve constraint optimisation problem: Lagrangian

$$L(\mathbf{x}, \lambda) = J(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})^T \mathbf{x} + \lambda \left(2\mathbf{x}^T \mathbf{F} \mathbf{x} - c\right)$$

- At optimality, partial derivatives of L are 0

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = 0 \qquad \frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = 0$$

$$\Rightarrow \quad \begin{aligned} (\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})) + 4\lambda F \mathbf{x} &= 0 \\ \mathbf{x}^T F \mathbf{x} &= c \end{aligned}$$

[Kakade 2002, Bagnell 2003]

NATURAL POLICY GRADIENT

- So optimality conditions are

$$\begin{aligned}\left(\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right)\right) + 2\lambda F\mathbf{x} &= 0 \\ \mathbf{x}^T F\mathbf{x} &= c\end{aligned}$$

- From the first line, update direction

$$\mathbf{x}^* \propto F^{-1}\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right)$$

- This is the **natural gradient**
(natural gradients in ML used at least since [Amari, 1998], used in RL since [Kakade 2002])

NATURAL POLICY GRADIENT

- The policy is adapted using the **natural gradient** update rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha F^{-1} \nabla_{\boldsymbol{\theta}_t} J(\boldsymbol{\theta}_t)$$

- We can use any known approach for the vanilla gradient
- Will this always improve J?
- For small enough step size, objective improves if

$$\mathbf{x}^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) > 0$$

NATURAL POLICY GRADIENT

- The policy is adapted using the **natural gradient** update rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha F^{-1} \nabla_{\boldsymbol{\theta}_t} J(\boldsymbol{\theta}_t)$$

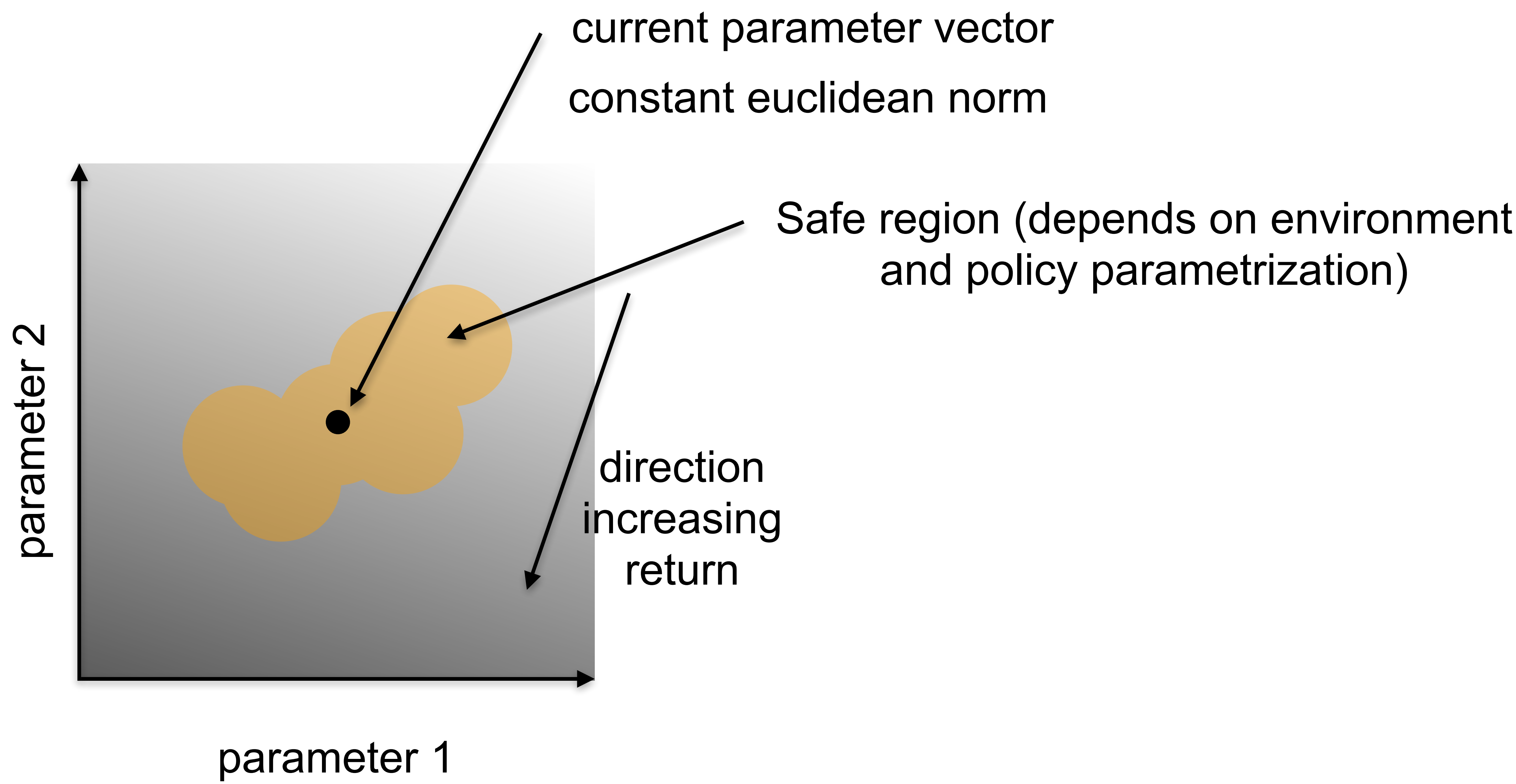
- We can use any known approach for the vanilla gradient
- Will this always improve J?
- For small enough step size, objective improves if

$$\mathbf{x}^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) > 0$$

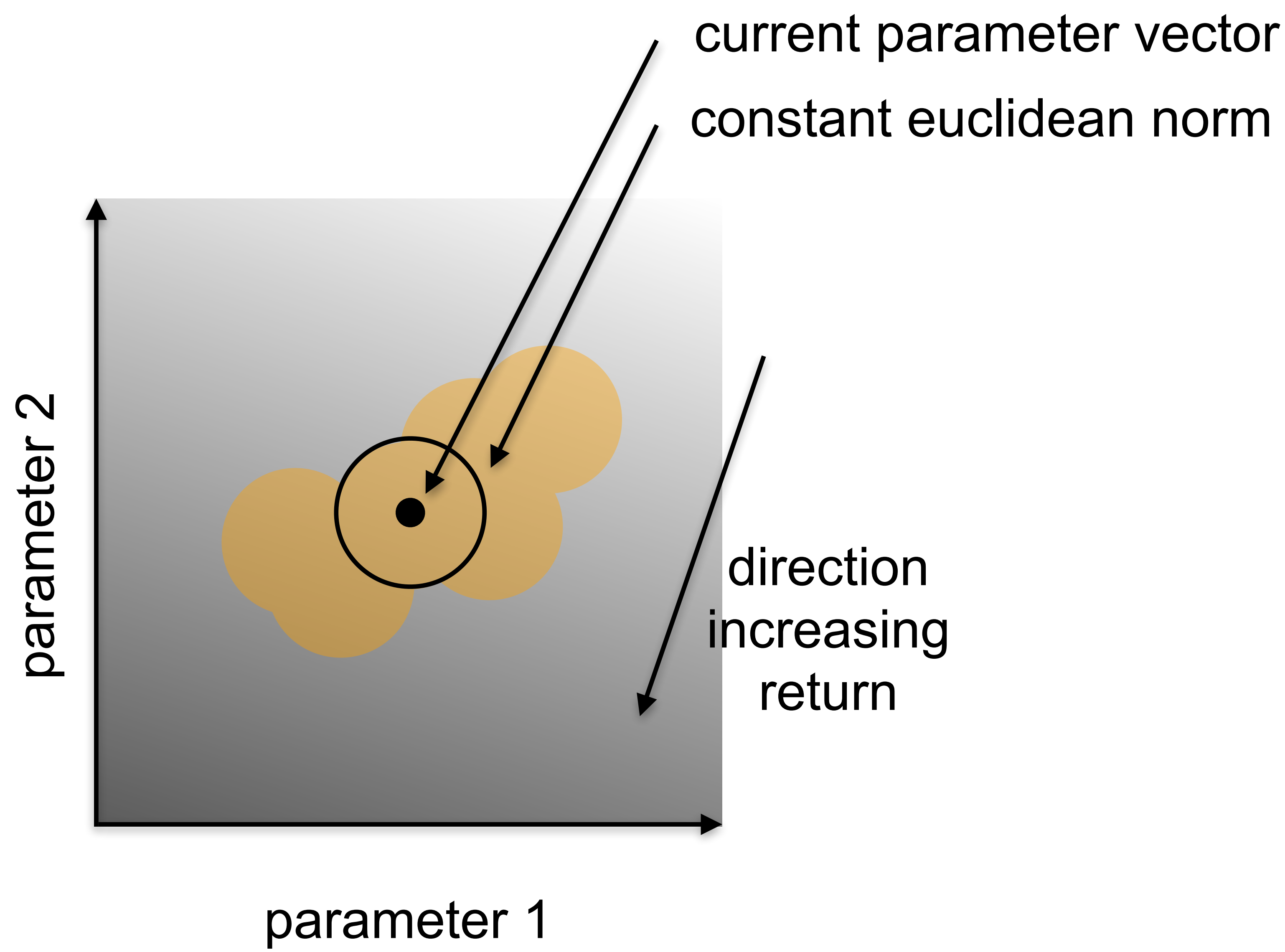
$$\left(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\right)^T F^{-1} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \overset{?}{>} 0$$

- Since Fisher information is positive definite, answer is **yes**

NATURAL POLICY GRADIENT: SOME INTUITION

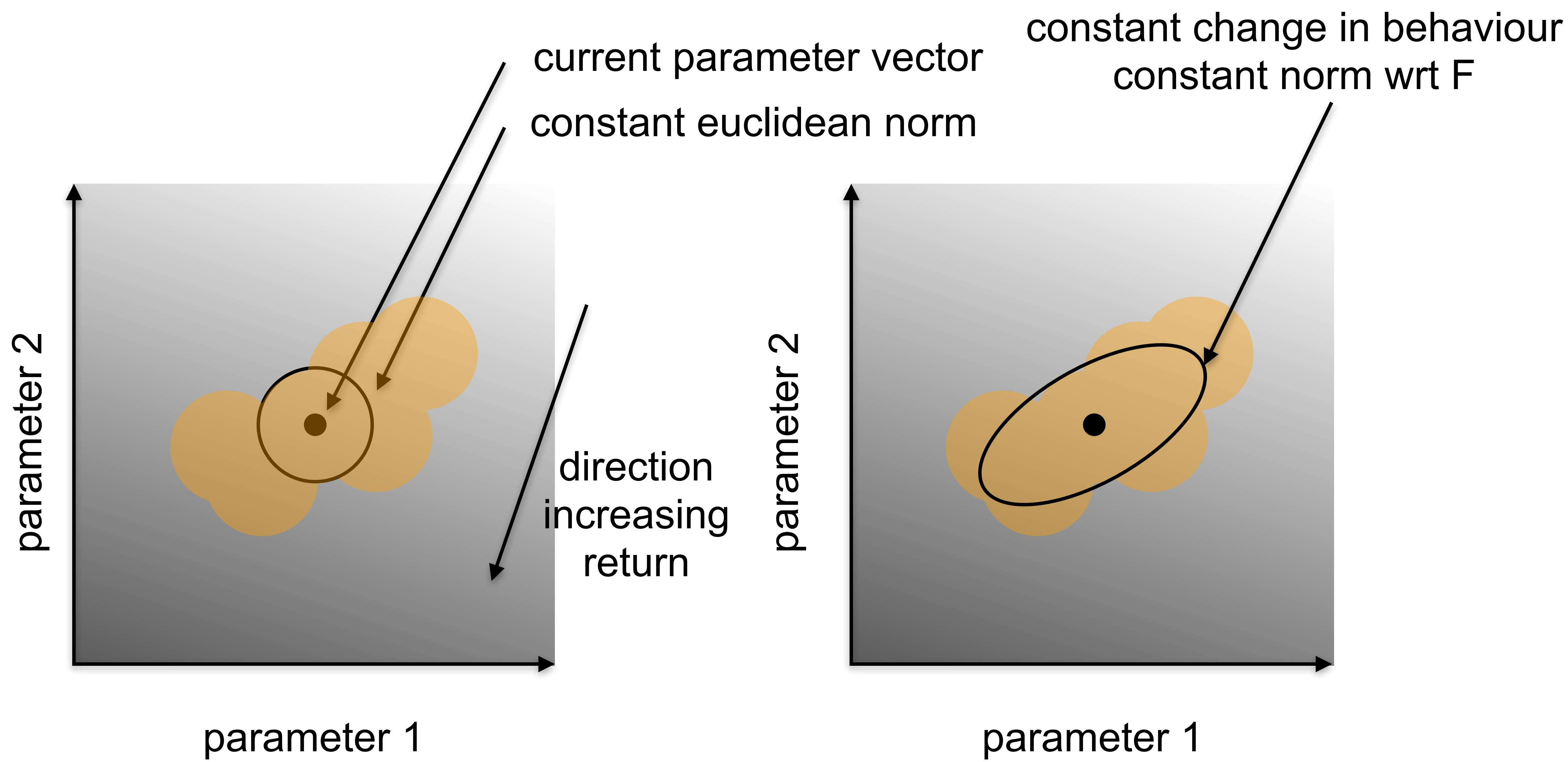


NATURAL POLICY GRADIENT: SOME INTUITION



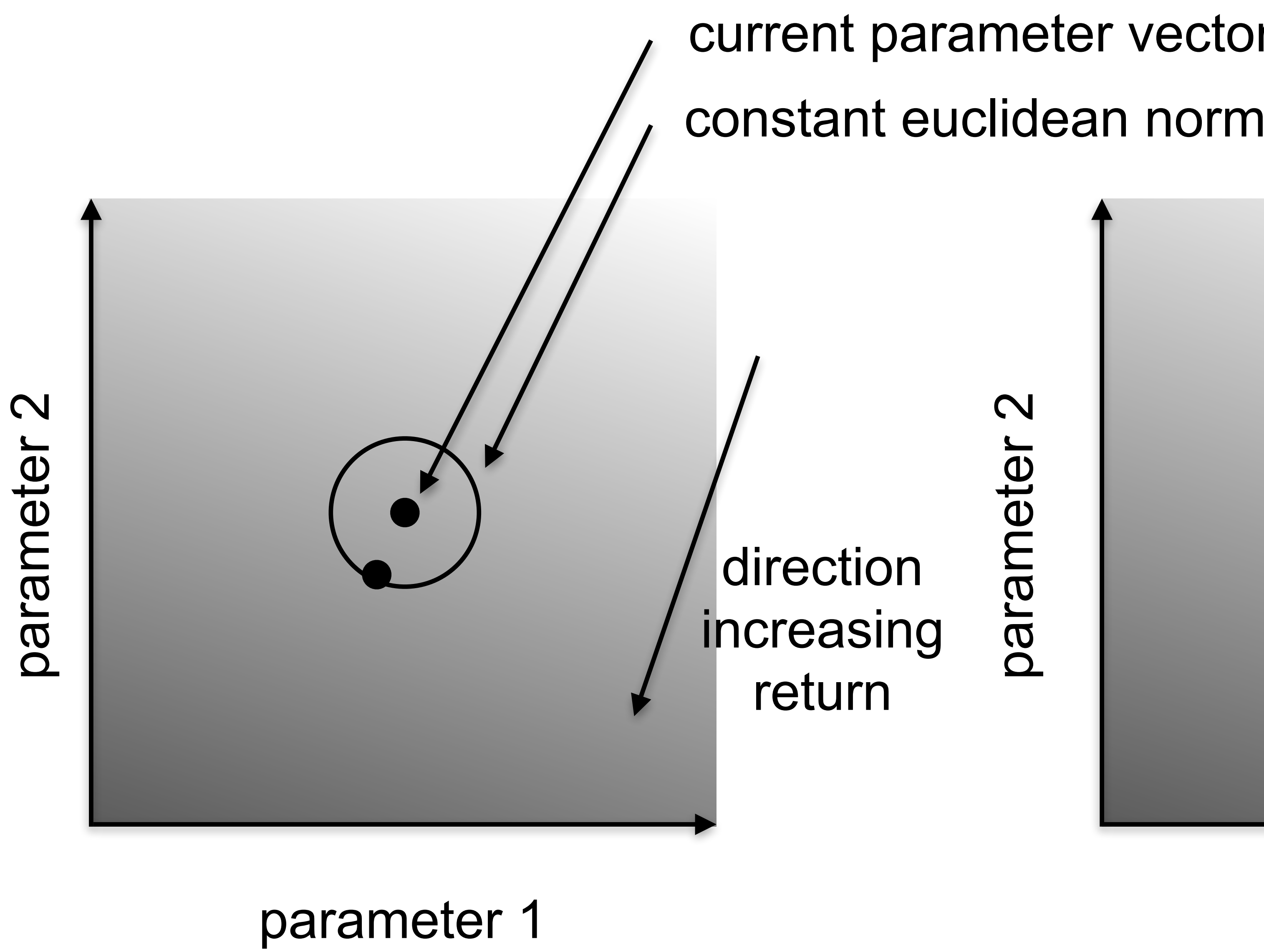
Regular gradient: increase step size till we notice we are outside of safe region

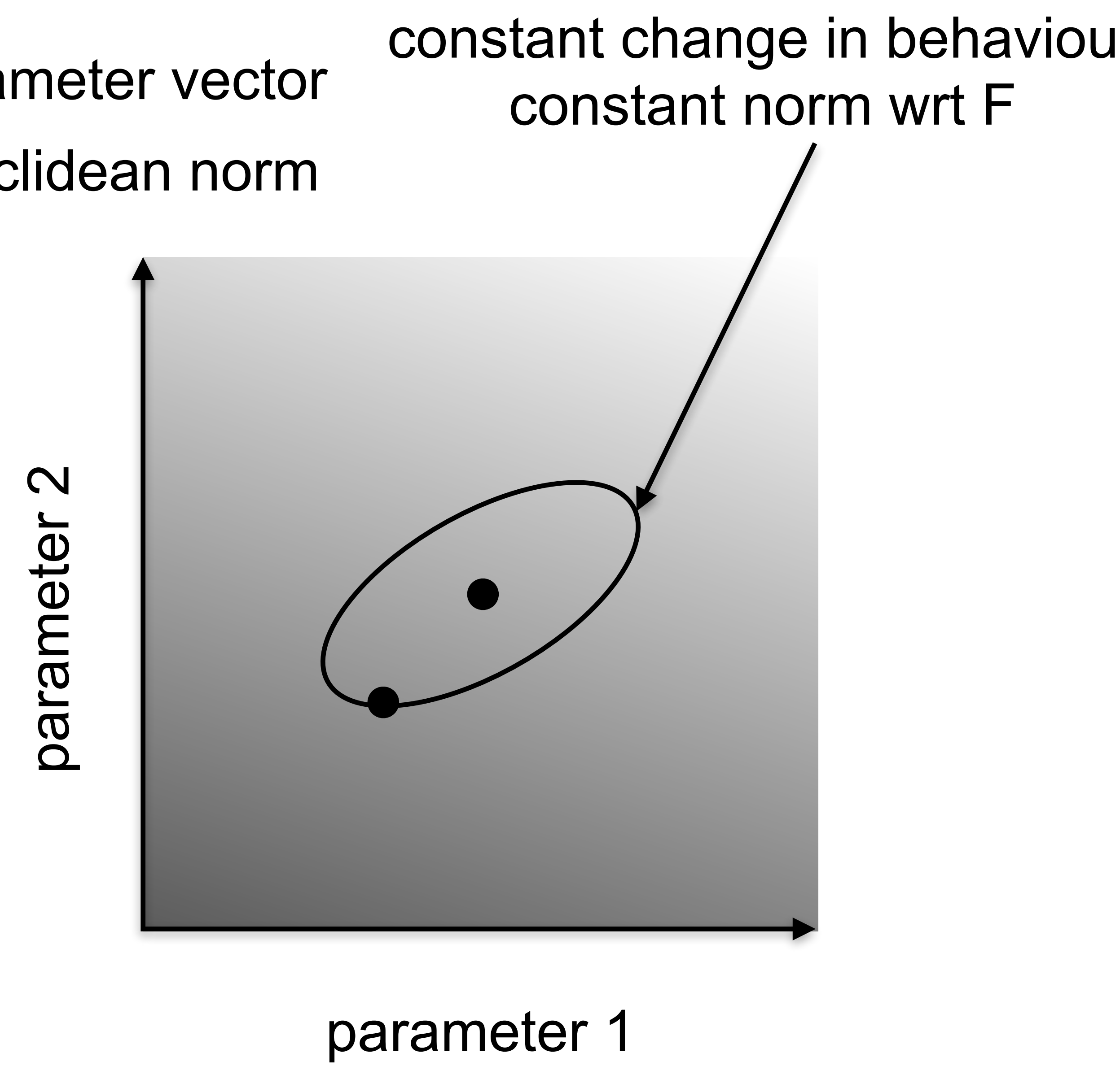
NATURAL POLICY GRADIENT: SOME INTUITION



Do the same for natural gradient…

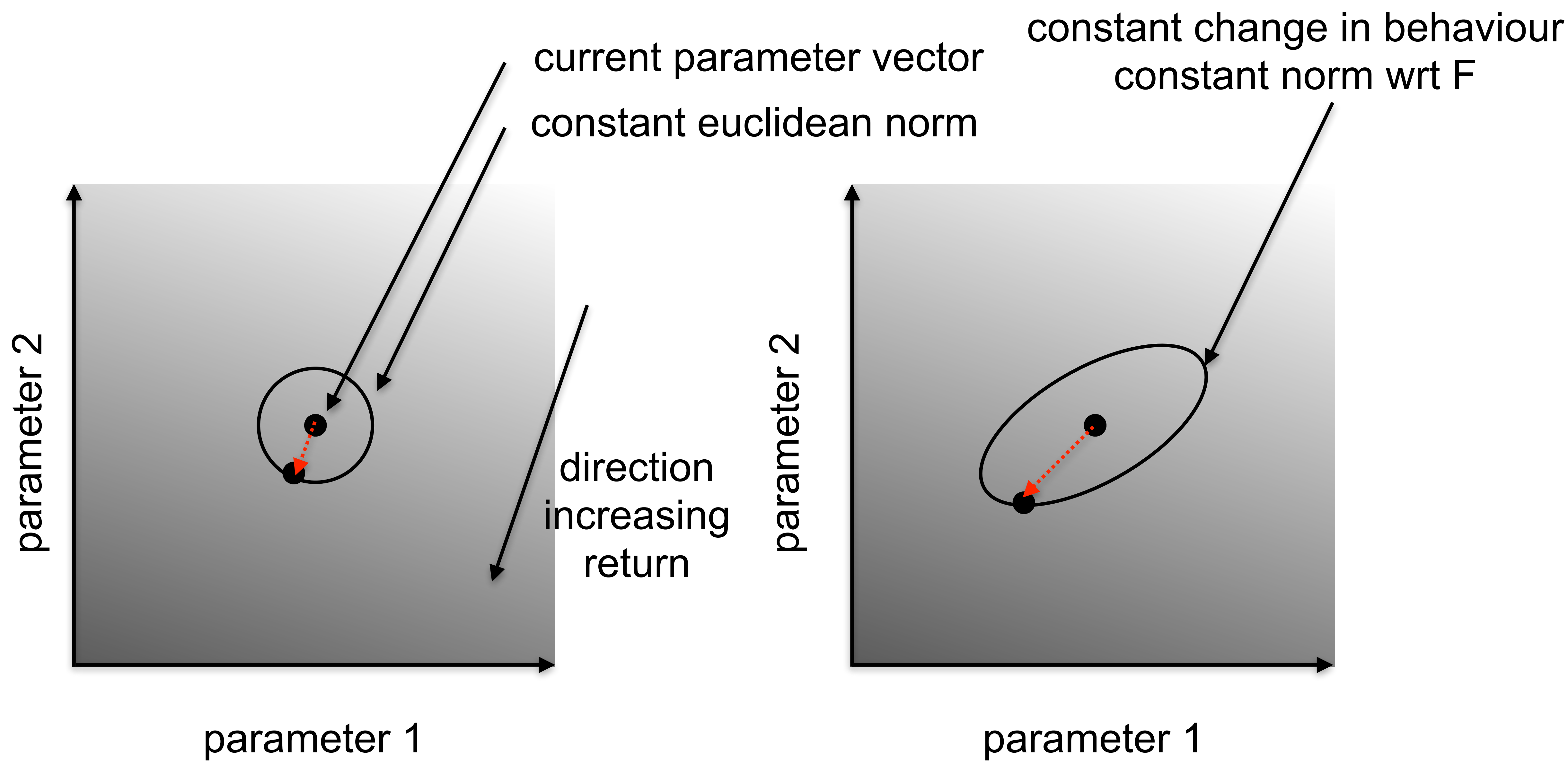
NATURAL POLICY GRADIENT: SOME INTUITION





As natural gradient can (hopefully!) better fit safe region, bigger steps possible

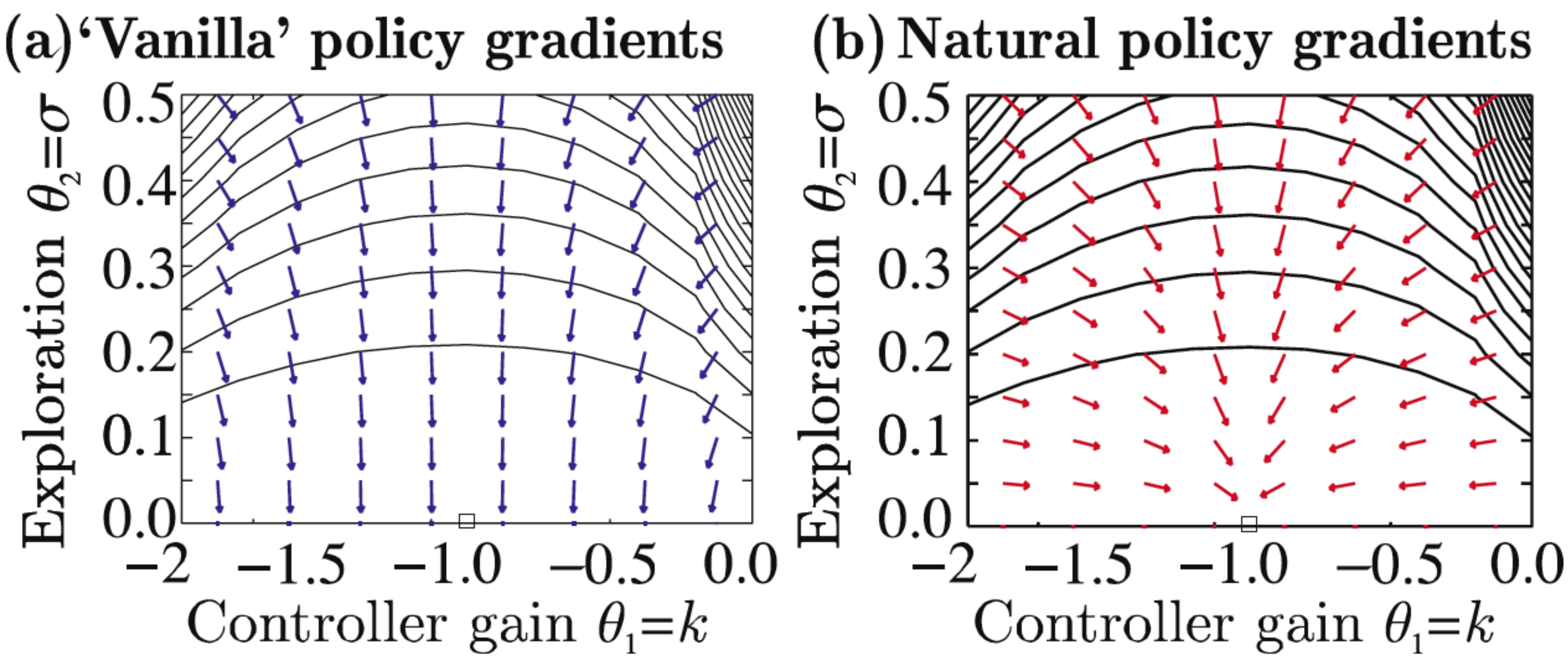
NATURAL POLICY GRADIENT: SOME INTUITION



Regular gradient: direction of steepest increase expected return
Natural gradient: within 90° of that direction: will improve objective
'steepest return per unit policy change'

NATURAL ACTOR CRITIC

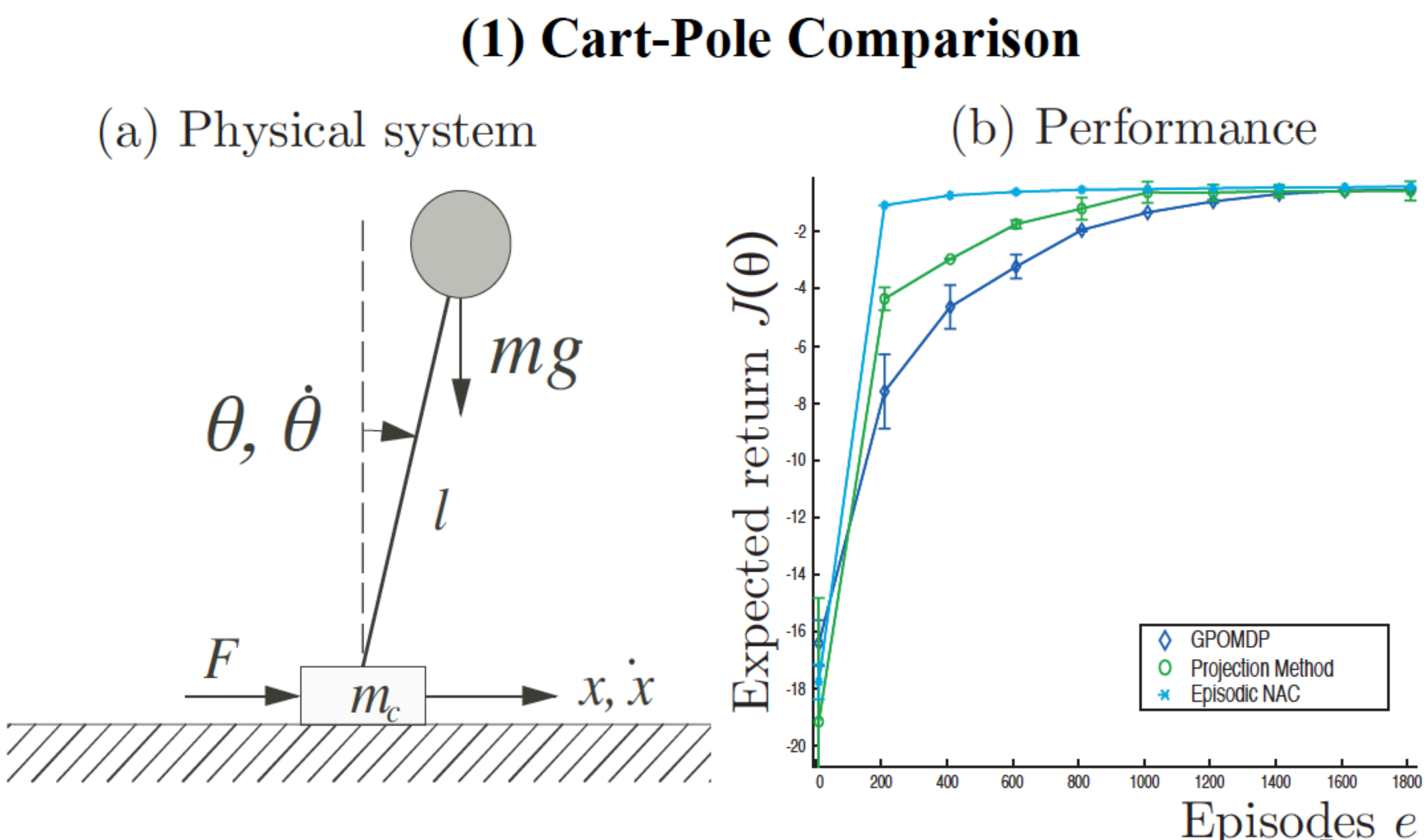
- Natural gradients can help where the likelihood is almost flat



[Peters 2008]

NATURAL POLICY GRADIENT

- Natural policy gradients can be used in actor-critic set-up



[Peters 2008]

NATURAL POLICY GRADIENT

- Advantages
  - Usually needs less training than regular policy gradients
  - Inherits advantageous properties from vanilla gradients
- Limitations
  - Need Fisher information matrix
    - Known for some standard distributions, e.g. Gaussian
    - Computationally costly to invert
  - Inherits disadvantages from PG (e.g., high variance)

STAYING CLOSE TO PREVIOUS POLICIES

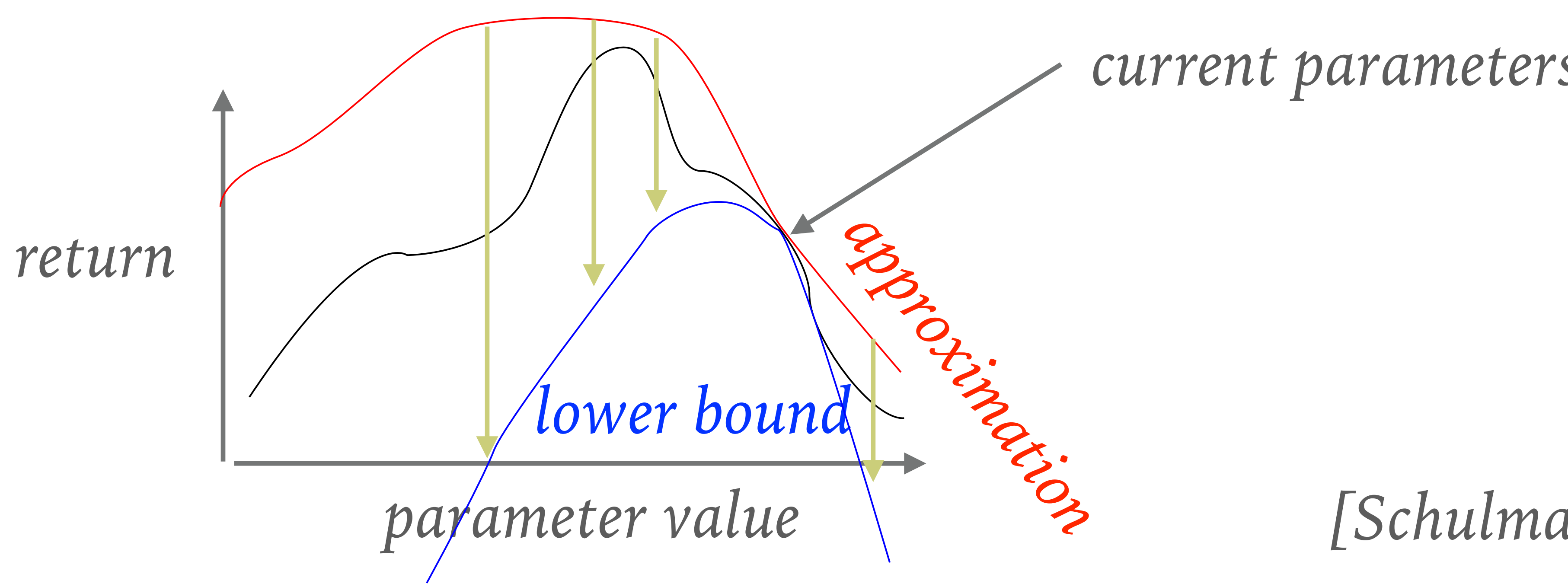
- Idea: use KL to specify how the policy can change in one step
  - Natural policy gradient
  - Trust region policy optimization (TRPO)

TRUST REGION POLICY OPTIMISATION

- Trust region: region where approximation is valid
- Schulman's "Trust region policy optimisation" defines a trust region based RL algorithm inspired by a theoretical lower bound
- Theoretical starting point quite different than policy gradients, but practical implementation is quite similar

TRPO: THEORY

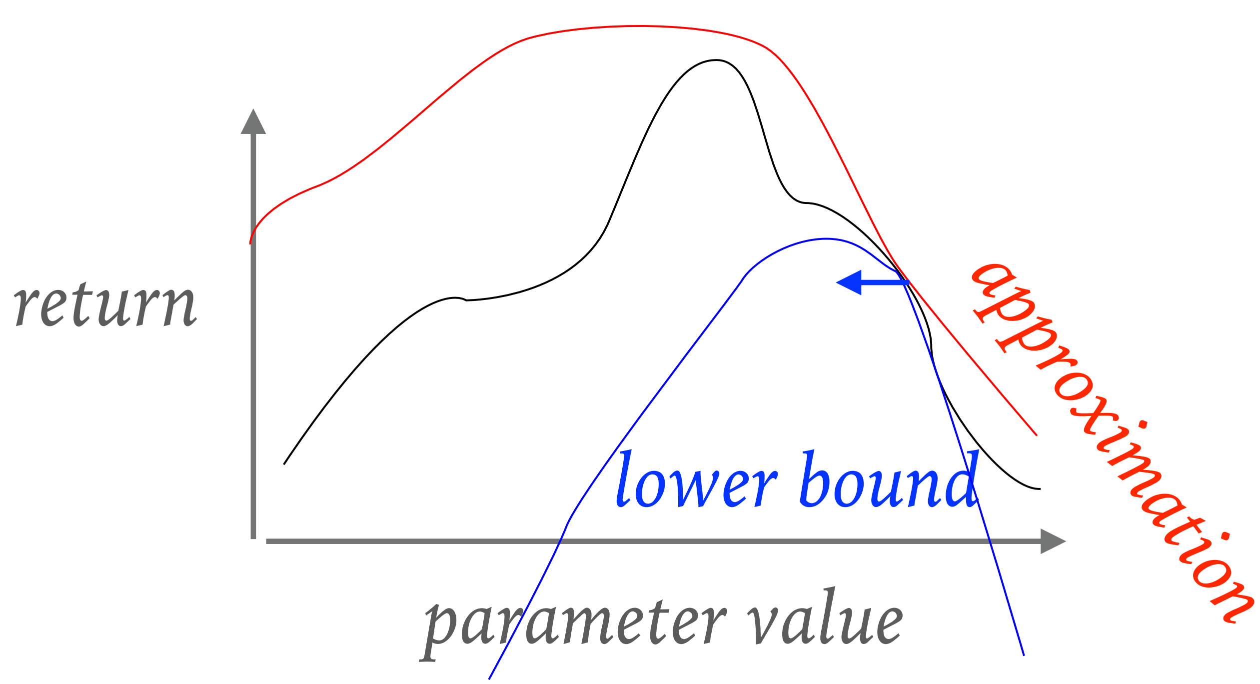
- Idea: take big steps while guaranteeing improvement
  1. approximate the return function
     (ignoring the difference between μ_b and μ_π)
  2. apply KL-based penalty term to yield lower bound
  3. maximize this lower bound (no need to specify step size!)

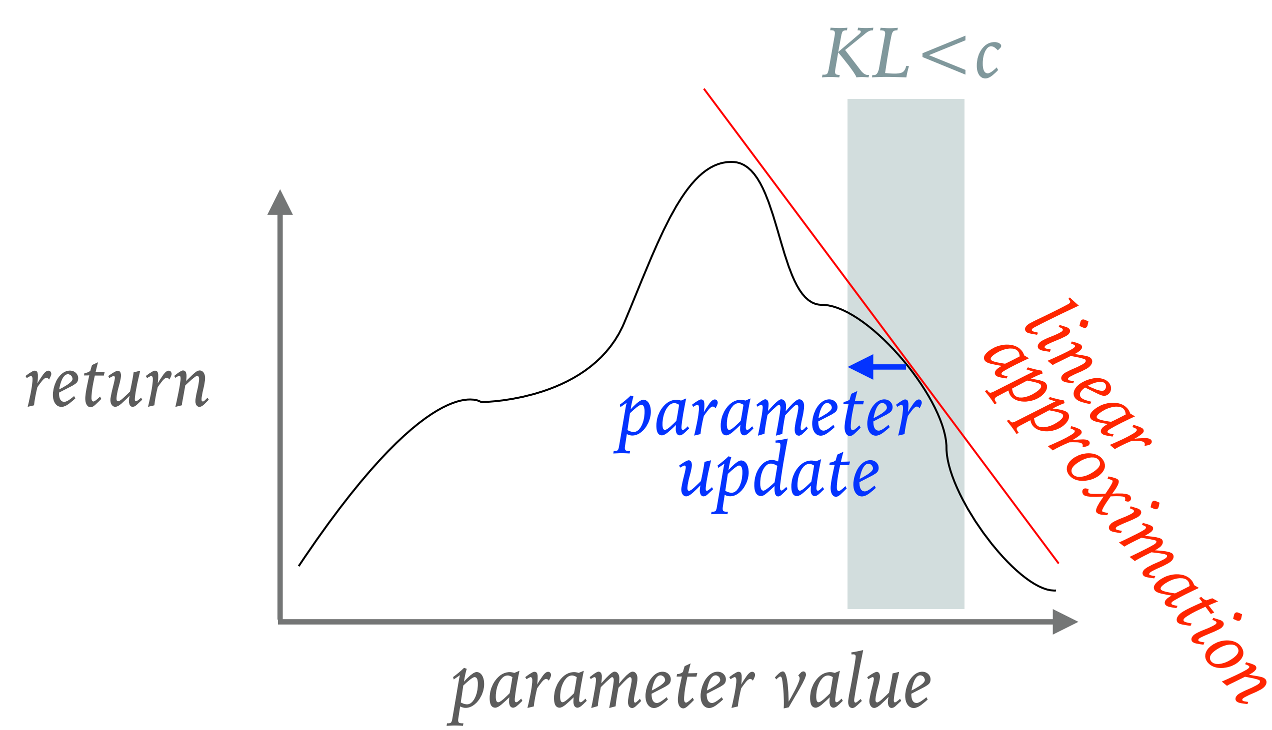


[Schulman 2016]

TRPO: PRACTICE

➤ Approximate theoretical update by maximising a linearised approximation under a KL *constraint* inspired by the KL-based *penalty* term





CONNECTION TO NATURAL GRADIENTS

- Maximising linearised performance under KL constraint is very reminiscent of our perspective on natural gradients!
- In natural gradients, we found the update direction with maximal improvement per unit KL
  $\mathbf{x} \propto \tilde{\nabla} J = F^{-1} \nabla_{\mathbf{x}} J\left(\boldsymbol{\theta} + \mathbf{x}\right)$
  taking a fixed-size step β.
- The optimal step size might not be same everywhere in parameter space
- Instead: derive step-size from KL constraint.

CONNECTION TO NATURAL GRADIENTS

- Instead: derive step-size from KL constraint. Write $\mathbf{x} = \beta \tilde{\nabla} J$ and look again at the Taylor expansion of the expected KL

$$c = \mathbb{E}_{\mathbf{s}}\left[D_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})\|\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}+\mathbf{x})\right] = \mathrm{EKL}(\mathbf{x})$$

$$= \frac{1}{2}\mathbf{x}^T\left(\nabla^2_{\mathbf{x}}\mathrm{EKL}(\mathbf{x})\right)\mathbf{x}$$

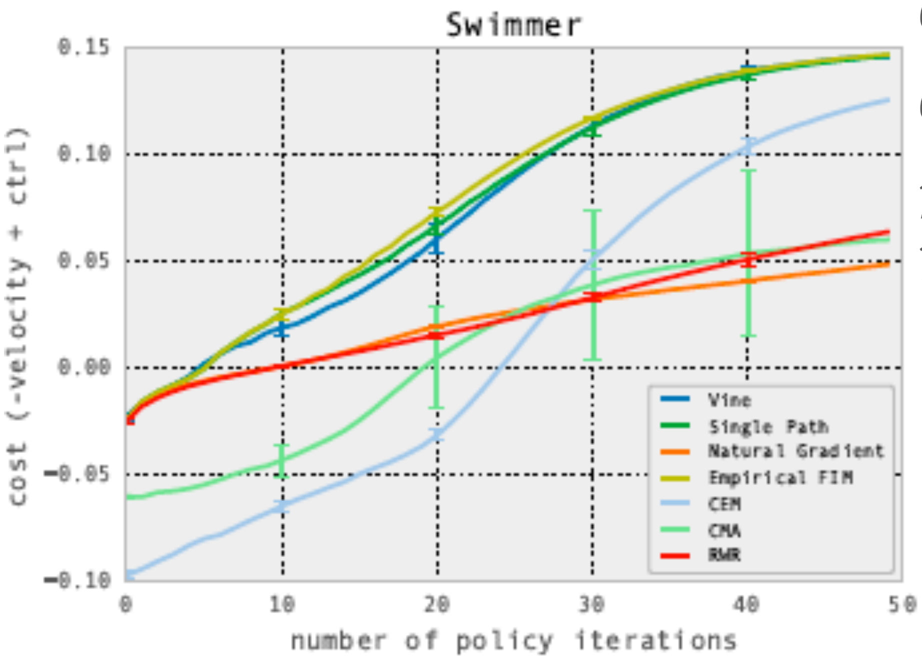
CONNECTION TO NATURAL GRADIENTS

- Instead: derive step-size from KL constraint. Write $\mathbf{d}\boldsymbol{\theta} = \beta\tilde{\nabla}J$ and look again at the Taylor expansion of the expected KL

$$
\begin{aligned}
c &= \mathbb{E}_{\mathbf{s}}\left[D_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})\|\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}+\mathbf{x})\right] = \mathrm{EKL}(\mathbf{x}) \\
&= \frac{1}{2}\mathbf{x}^T\left(\nabla^2_{\mathbf{x}}\mathrm{EKL}(\mathbf{x})\right)\mathbf{x} \\
&= \frac{1}{2}\beta^2(\tilde{\nabla}J)^T\left(\underbrace{\nabla^2_{\mathbf{x}}\mathrm{EKL}(\mathbf{x})}_{F}\right)\tilde{\nabla})
\end{aligned}
$$

- $c \approx \beta^2(\tilde{\nabla}J)^T F(\tilde{\nabla}J)/2$ can now be solved to get β
- Based on approximation of objective and KL
- Make sure constraint is met using analytic objective & KL
- Additional tricks to work with large number of parameters (NN)

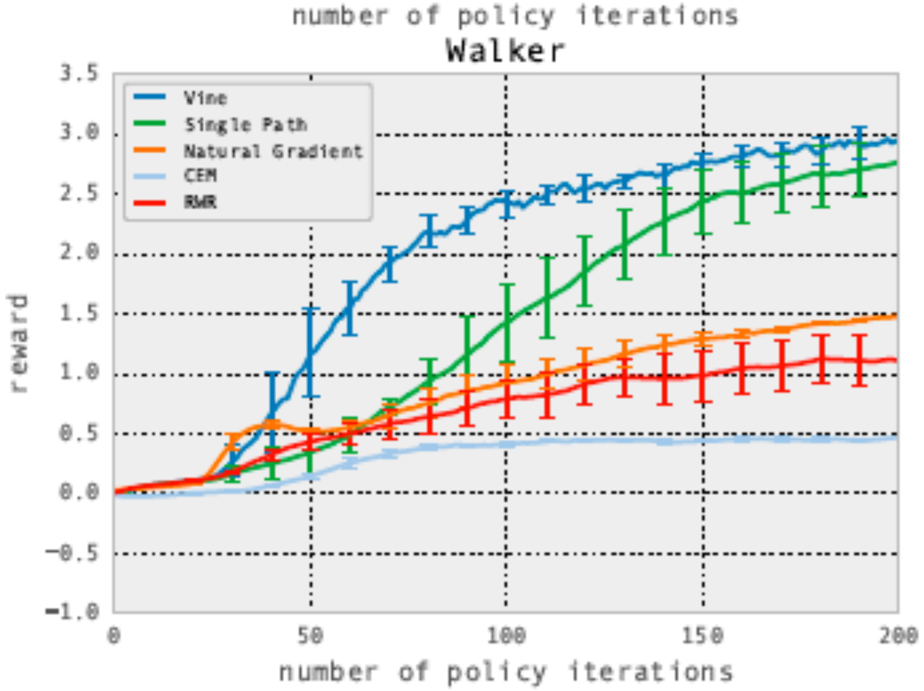
TRPO EVALUATION



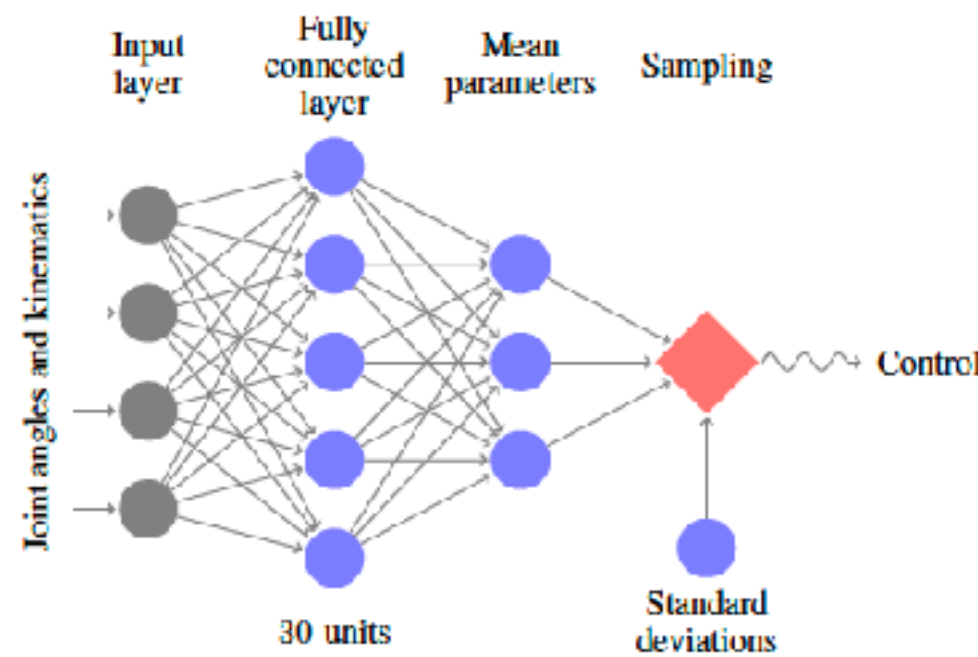
different TRPO variants

direct policy search

natural gradients,
reward-weighted regression



neural network policy used



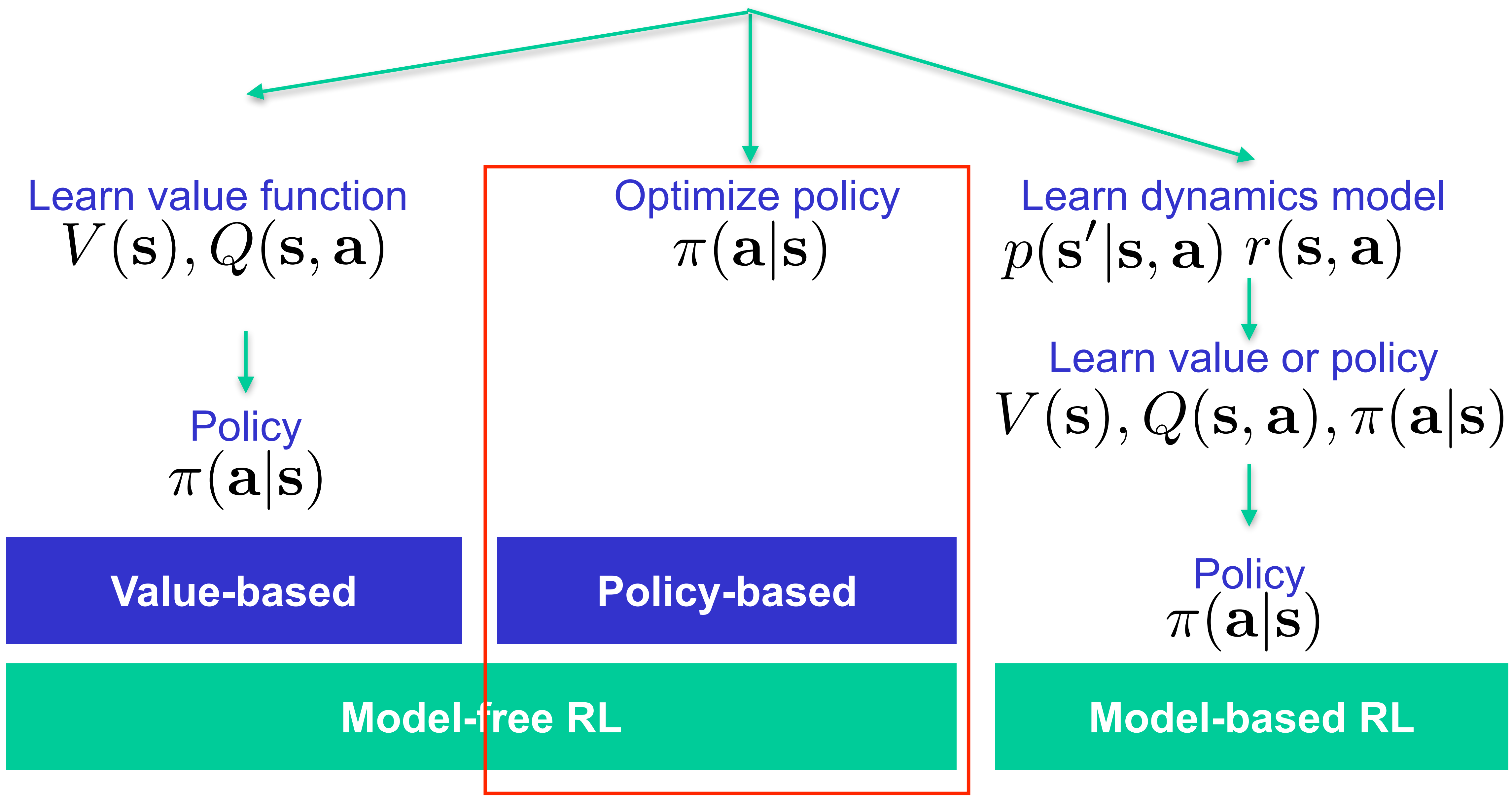
[Schulman, 2015]

TRPO EXAMPLE

TRPO with generalised advantage estimate, [Schulman 2016]

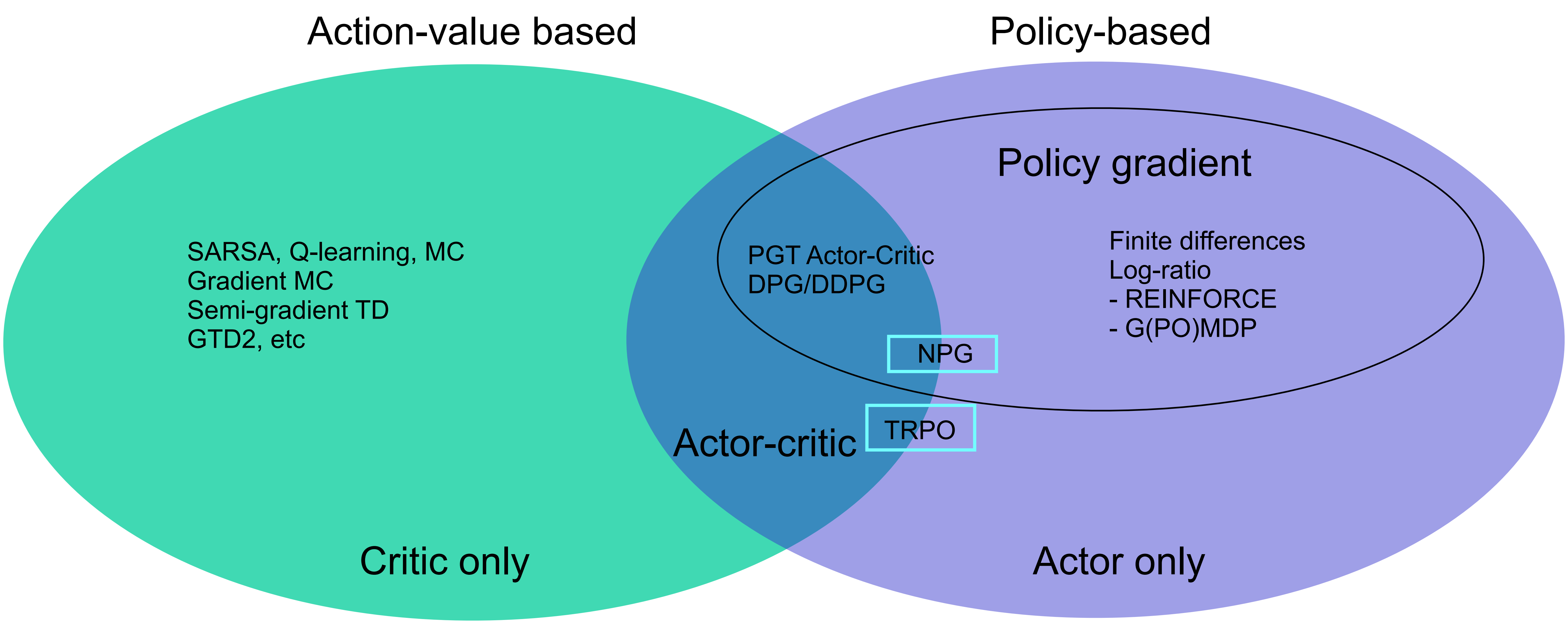
Big picture: How to learn policies

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



Thanks to Jan Peters

Policies and action-values



CONCLUSIONS

- NPG, TRPO: all use improved metric for policy updates, exploit structure of policy
- Both have a computational cost in inverting F
- Generally allows taking larger steps in policy space than 'vanilla' methods
- NPG: easier to implement, still manual step size
- TRPO: even larger steps (faster), step size from KL constraint

WHAT YOU SHOULD REMEMBER

- Advantage of covariant representation of distances?
- Advantage of specifying constraint instead of stepsize?
- Why do we need a constraint / penalty / stepsize?

THANKS FOR YOUR ATTENTION

- Feedback?
- h.c.vanhoof@uva.nl

REFERENCES

- [Bagnell 2003] Bagnell, J.A. and Schneider, J. Covariant policy search. IJCAI.
- [Kakade 2002] Kakade, S., 2002. A natural policy gradient. In: NeurIPS
- [Peters 2008] Peters, J. and Schaal, S., 2008. Natural actor-critic. Neurocomputing, 71(7), pp.1180-1190
- [Sutton 2000] Sutton, R.S., McAllester, D., Singh, S. & Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: NeurIPS.
- [Schulman 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M.I. and Moritz, P. Trust Region Policy Optimization. In: ICML
- [Schulman 2016] Schulman, J., Moritz, P., Levine, S., Jordan, M.I. and Abbeel, P. High-dimensional continuous control using generalised advantage estimates. In ICLR.