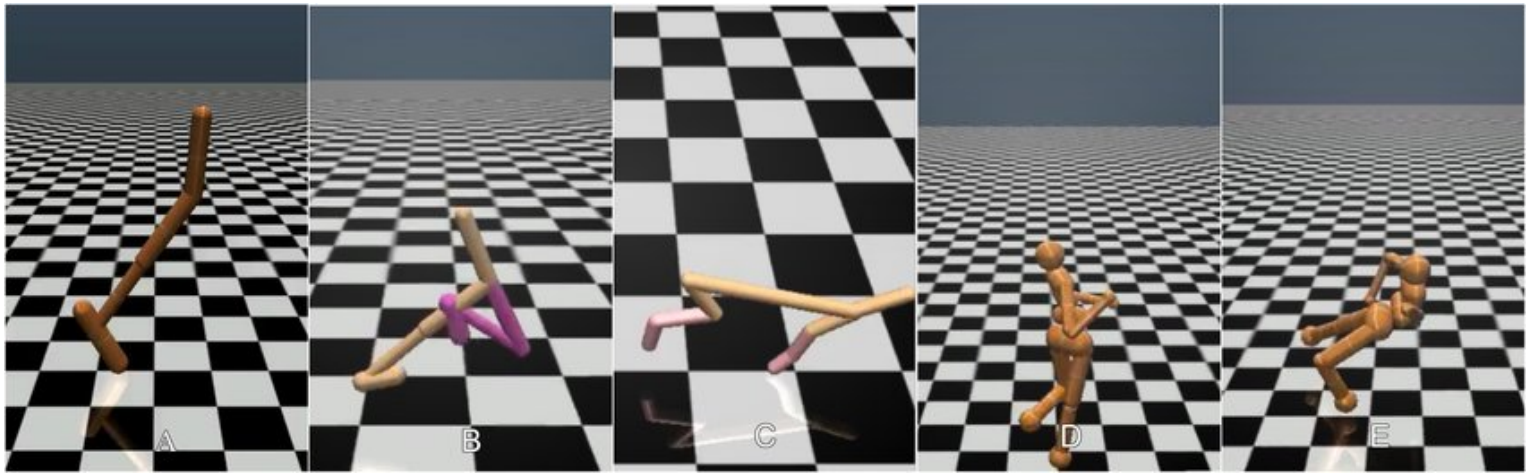
Variation-selection versus RL-based Policy Search

An investigation centred on sample efficiency

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

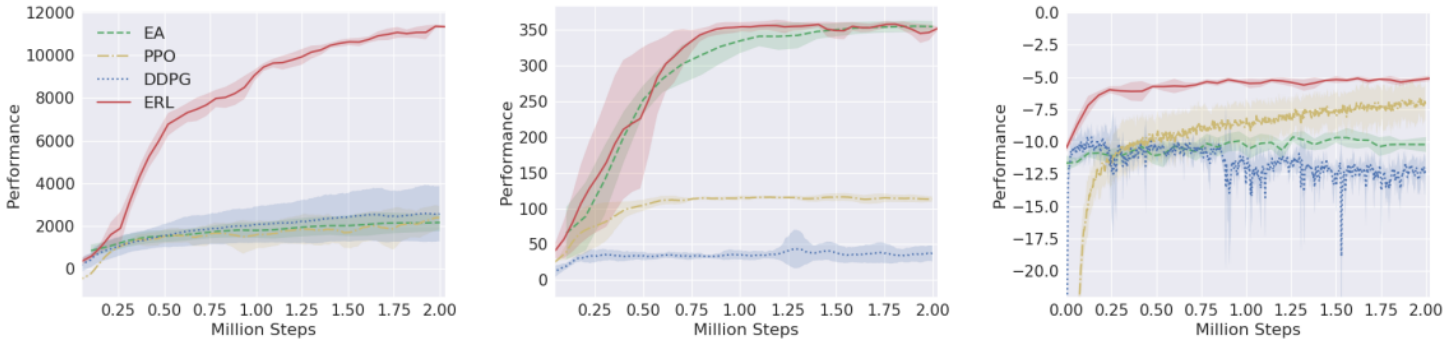
Goal of the class



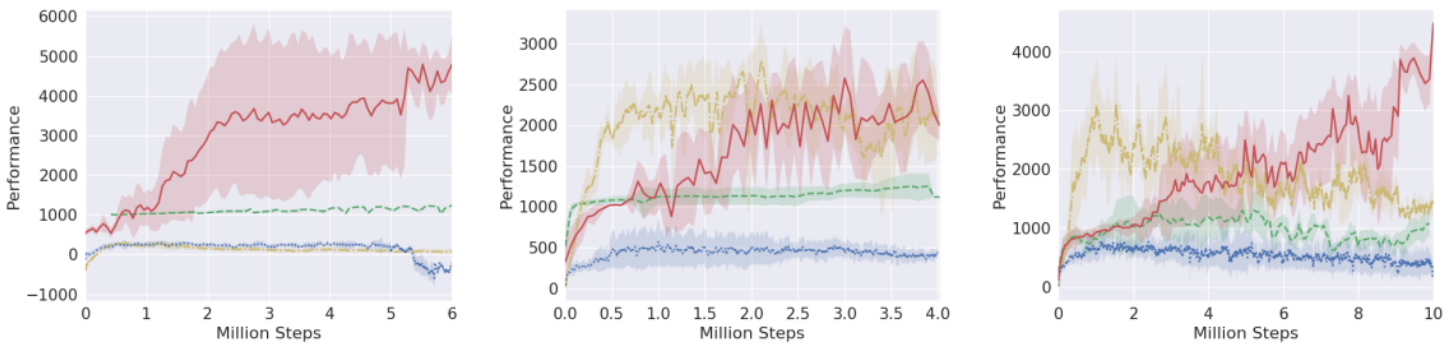
- Deep neuroevolution is competitive with deep RL in challenging benchmarks
- But deep RL methods seem to be an order of magnitude more sample efficient
- Why is this so?
- Opportunity for a deeper understanding of the inner mechanisms

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017

Experimental comparisons (1)



(a) HalfCheetah (b) Swimmer (c) Reacher

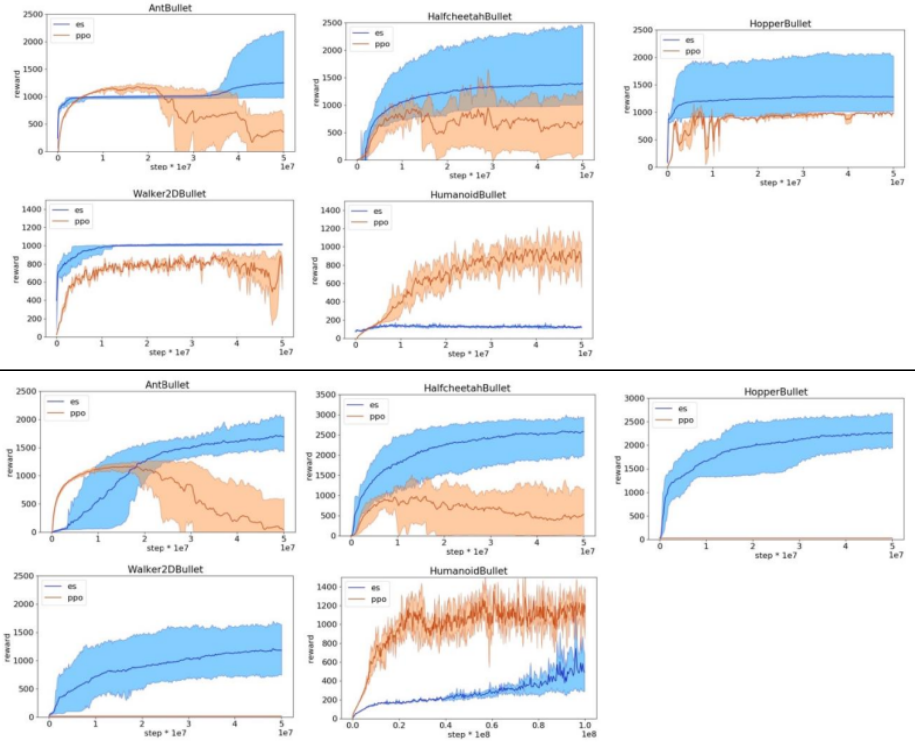


(d) Ant (e) Hopper (f) Walker2D

- A quick look at several papers shows that it is by no way obvious that PPO outperforms ESs

Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. In *Neural Information Processing Systems*, 2018

Experimental comparisons (2)



- The design of the reward function plays a big role in the relative performance of PG and ES
- Dubious validity of the results

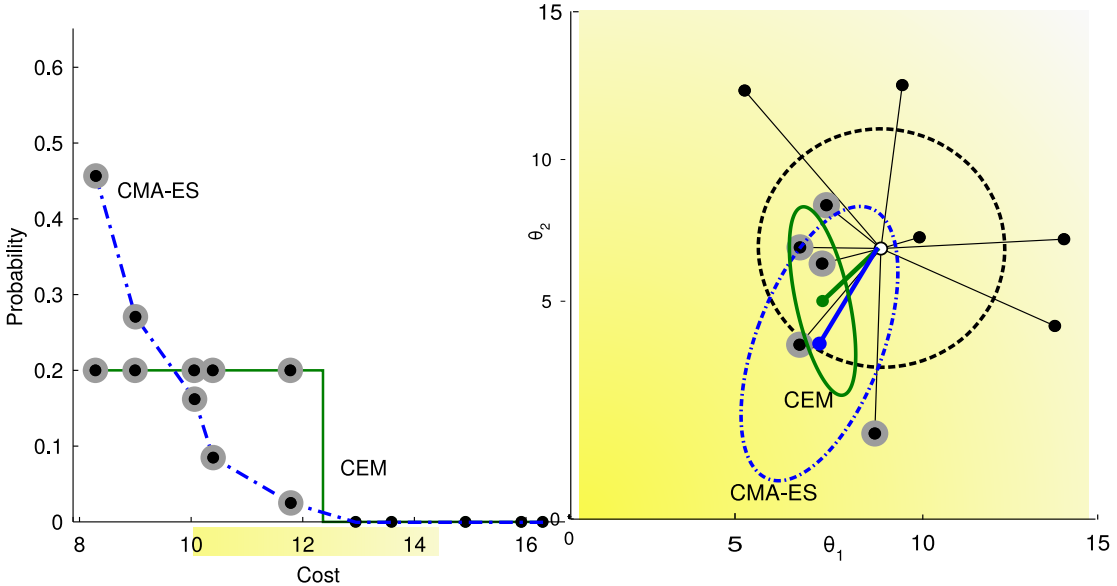
Paolo Pagliuca, Nicola Milano, and Stefano Nolfi. Efficacy of modern neuro-evolutionary strategies for continuous control optimization. *arXiv preprint arXiv:1912.05239*, 2019

Outline

- A quick presentation of EDAs (CEM, CMA-ES)
- Potential reasons why modern RL would be more sample efficient:
  - The gradient gives the direction of steepest ascent: it improves faster
  - Gradient ascent does not need sampling. It uses analytical knowledge of the function under optimization to improve it
  - RL searches a better space: it uses information at each state action pair, versus the whole episode for variation-selection methods
  - RL is better informed: it does not perform blind variations as variation-selection methods
  - RL methods can reuse samples, variation-selection methods cannot
- Approach: investigate each potential reason to see whether it holds in practice

Spoiler: sample reuse is the key reason

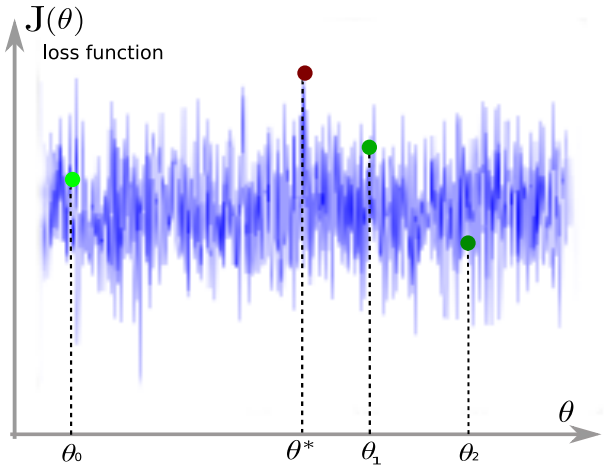
CEM and CMA-ES



- Genetic algorithms manage a population of solution
- They reproduce and mutate the most successful solutions
- EDAs replace the population with a Gaussian distribution of solutions
- The Gaussian distribution is coded as a covariance matrix
- The distribution is evolved to generate better and better individuals
- CEM and CMA-ES are among the most efficient, CEM is simpler
- They mostly differ in the solution selection method

De Boer, P.-T., Kroese, D., Mannor, S., and Rubinstein, R. (2005) A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67

(Truly) Random Search



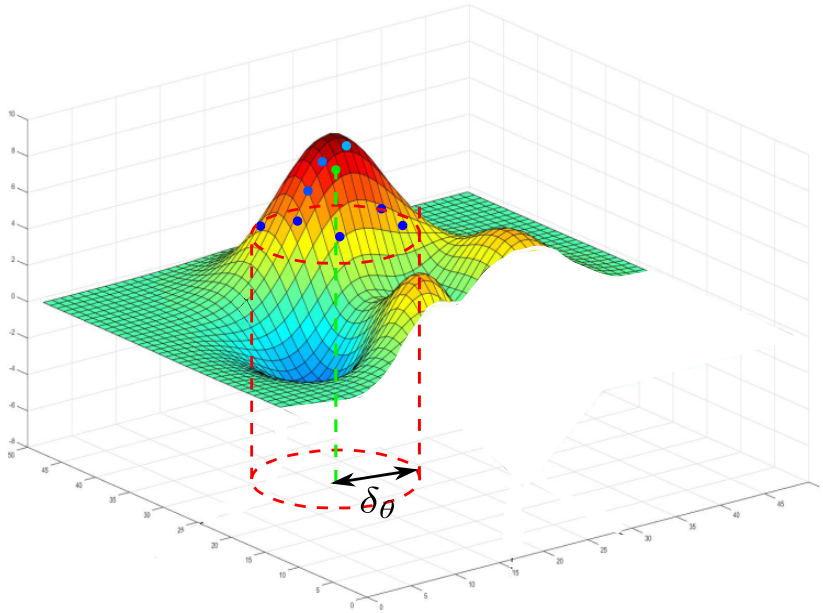
- Select $\boldsymbol{\theta}_i$ randomly
- Evaluate $J(\boldsymbol{\theta}_i)$
- If $J(\boldsymbol{\theta}_i)$ is the best so far, keep $\boldsymbol{\theta}_i$
- Loop until $J(\boldsymbol{\theta}_i) > target$ (maximize reward)

- Of course, this is not efficient if the space of $\boldsymbol{\theta}$ is large
- General "blind" algorithm, no assumption on $J(\boldsymbol{\theta})$
- We can do better if $J(\boldsymbol{\theta})$ shows some local regularity

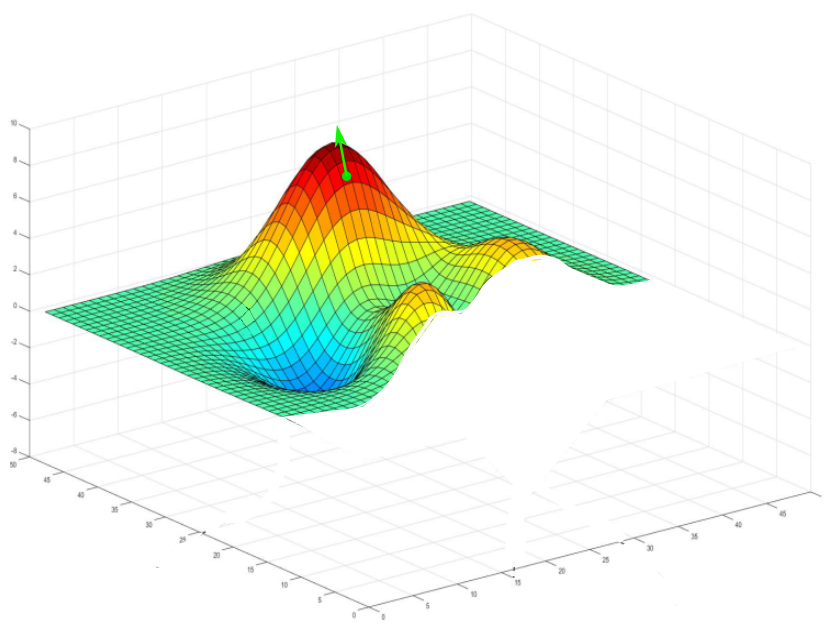
Sigaud, O. & Stulp, F. (2019) Policy search in continuous action domains: an overview. *Neural Networks*, 113:28-40

Two approaches to function optimization

- Assumption: The function is locally smooth, unknown good solutions are close to known good solutions



Variation - selection



Gradient descent

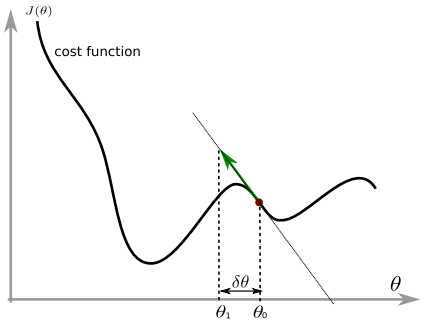
- Variation - selection: Perform variations and evaluate them
- Locality generally enforced using a multivariate Gaussian
- Gradient ascent: Following the gradient from analytical knowledge

Does policy gradient follow the steepest direction?

Outline of this part

- In a first step, let's ignore the policy search context and consider general function maximization
- Compare gradient ascent and variation-selection purely in terms of dynamics of improvement
- Standard gradient ascent performs steepest ascent: is it really faster?
- What about advanced gradient ascent techniques?

Strengths of the policy gradient approach



- The optimum of $\mathbf{J}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})$ over $\delta\boldsymbol{\theta}$ is reached when $\dfrac{\partial \mathbf{J}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})}{\partial \delta\boldsymbol{\theta}} = 0$
- First order approx: $\mathbf{J}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})^T \delta\boldsymbol{\theta} + \nu\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta} +$ higher order terms
- $\dfrac{\partial \mathbf{J}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})}{\partial \delta\boldsymbol{\theta}} \sim \nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})^T + 2\nu\delta\boldsymbol{\theta} = 0 \rightarrow \delta\boldsymbol{\theta}^* = -\alpha\nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})$

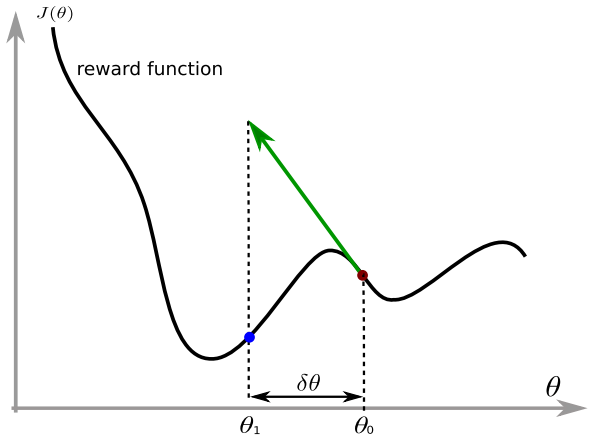
- In principle, the gradient ascent approach is superior:
  - The gradient provides the best direction of improvement
  - Gradient computation requires no sampling
  - Advanced and adaptive gradient descent techniques improve efficiency
  - The PG approach is step-based, it benefits from more information than the DPS approach, which is episode-based

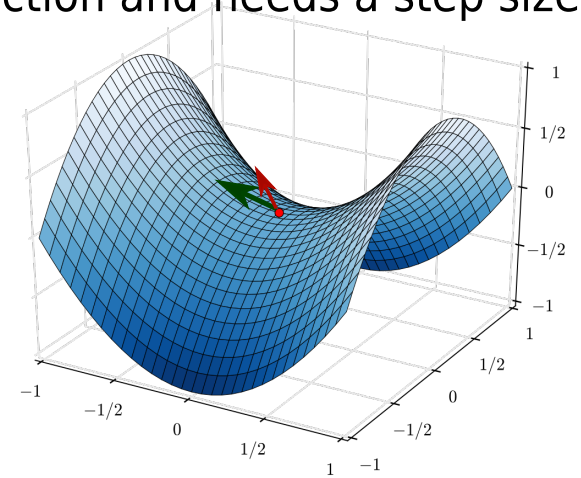
Ruder, S. (2016) An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*

Pierrot, T., Perrin, N., & Sigaud, O. (2018) First-order and second-order variants of the gradient descent: a unified framework. *arXiv preprint arXiv:1810.08102*

Limits of the PG approach

A gradient provides a local direction and needs a step size



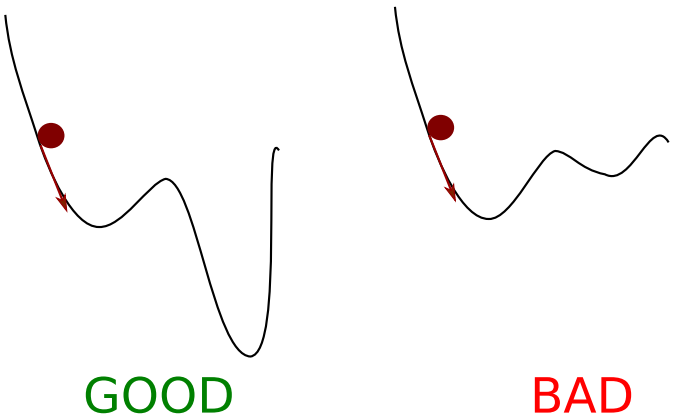


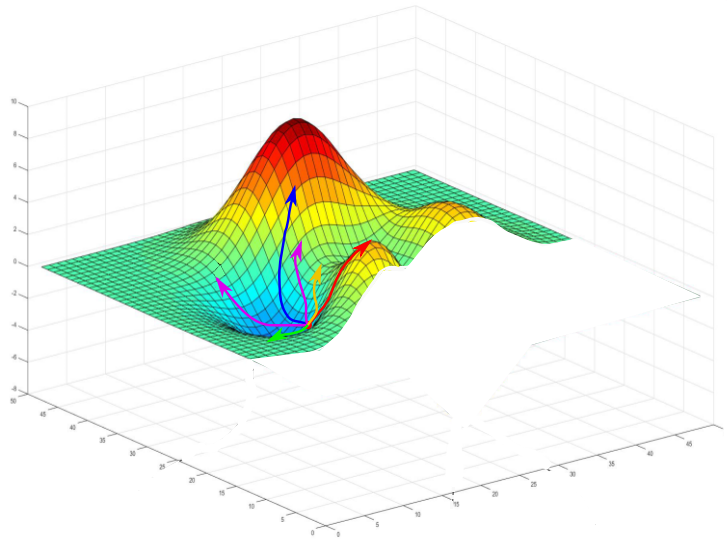
If the step size is too large, may result in loss

If the direction is slightly wrong, may result in loss

- Even advanced gradient descent techniques are blind, the gradient step ensures improvement only very locally (need for a trust region).
- To get a policy gradient, an expectation over trajectories is replaced with sampling, thus trajectory sampling is also required
- Thus the gradient estimate suffers from (often large) variance, which may result of wrong direction of improvement

Advanced and adaptive gradient ascent



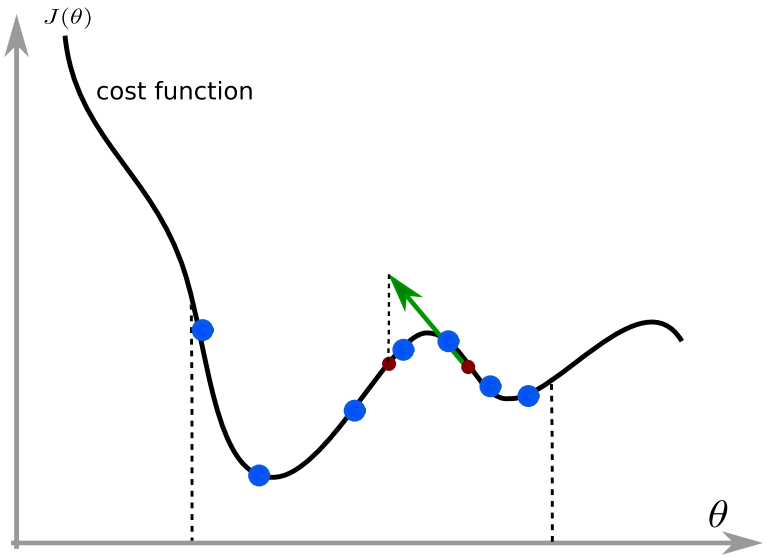


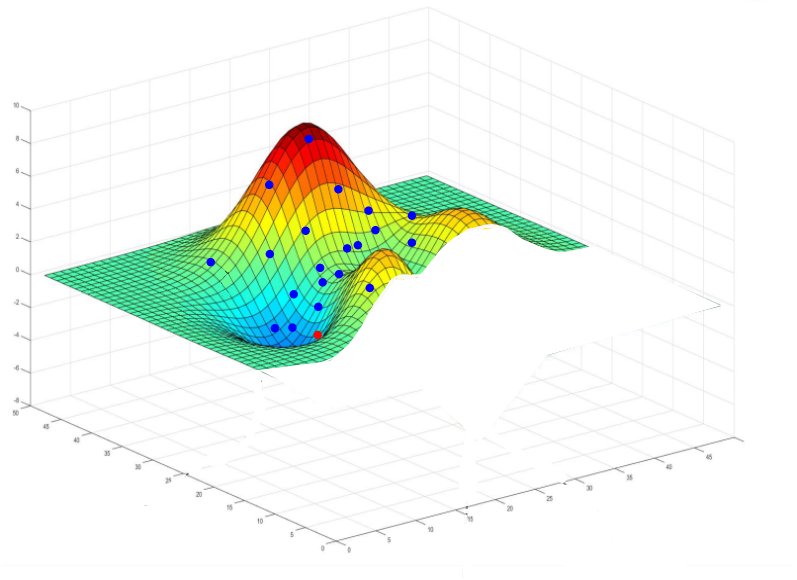
- Deep RL libraries and algorithms do more than plain gradient ascent
- Do advanced gradient ascent techniques improve sample efficiency?
- Adaptive gradient ascent methods: Adam, RMSProp, Momentum, Nesterov...
- Advanced gradient ascent methods: natural gradient, Gauss-Newton, ...
- General message: no free lunch, some technique improves in some context
- But it is probably not critical as improvement is problem-dependent

Ruder, S. (2016) An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*

Pierrot, T., Perrin, N., & Sigaud, O. (2018) First-order and second-order variants of the gradient descent: a unified framework. *arXiv preprint arXiv:1810.08102*

Advantage of variation selection





- Variation-selection over a range is less myopic than gradient ascent
- By considering more points, variation-selection investigates more potential solutions at each step

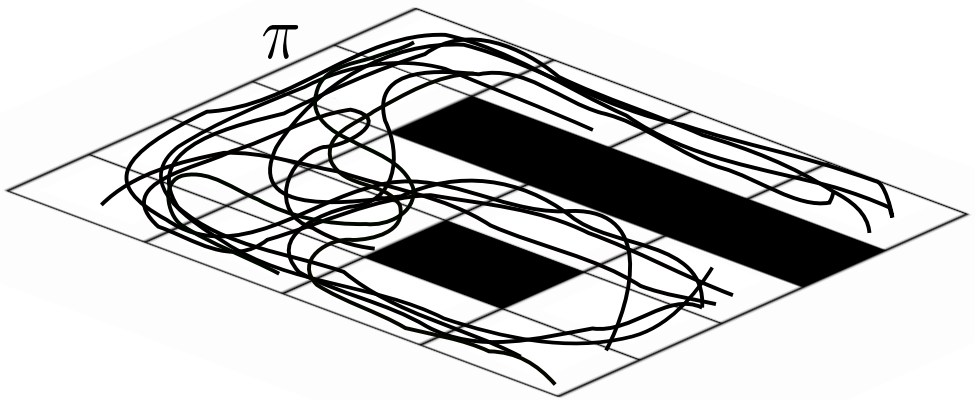
First messages

- It is not clear that gradient ascent finds better directions
- But variation-selection methods seem to consider more points
- What does it mean to "consider a point"?
- Gradient ascent uses the analytical gradient: what is the advantage?
- Move to the policy search context:
  - Policy gradient ascent **does** require sampling
  - Policy gradient ascent works in a different space

Policy search case

- So far, the function to maximize was assumed known
- Considering a point is free: no need for sampling
- But in policy search, the cost function $J(\boldsymbol{\theta})$ is the performance of a policy over policy parameters, it is known through sampling
- $$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)}) \qquad (1)$$
- But to apply (1), we also need many trajectories
- One point is that the gradient update makes profit of all state-action pairs, not just trajectories

Sampling

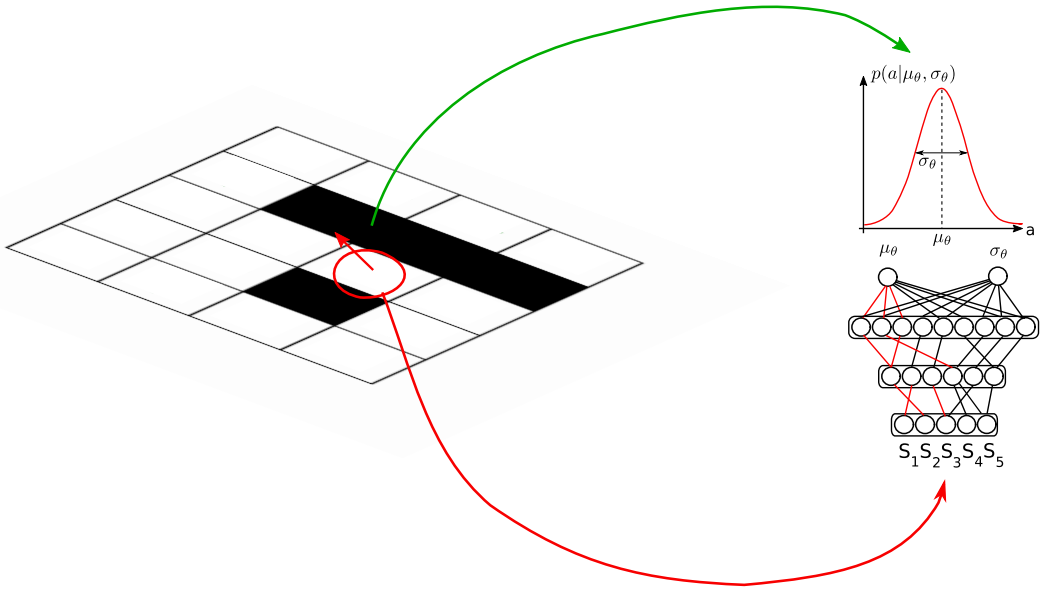


- Sample a set of trajectories from $\pi_{\boldsymbol{\theta}}$
- Compute:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

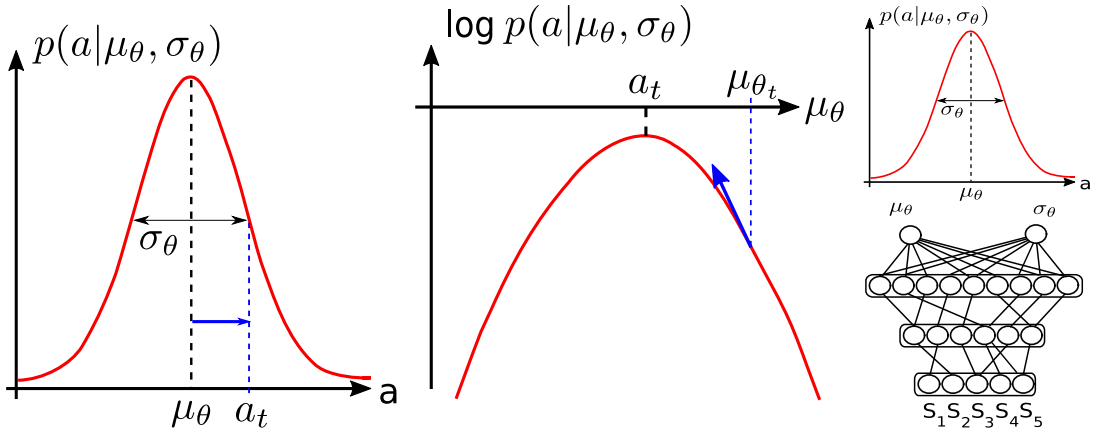
- Side note: if $R(\tau) = 0$, does nothing
- Update parameters $\boldsymbol{\theta}$ of $\pi_{\boldsymbol{\theta}}$ using $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \alpha_i \nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta})$ (or something more advanced)
- Iterate: sample again

Computing $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) R(\tau)$



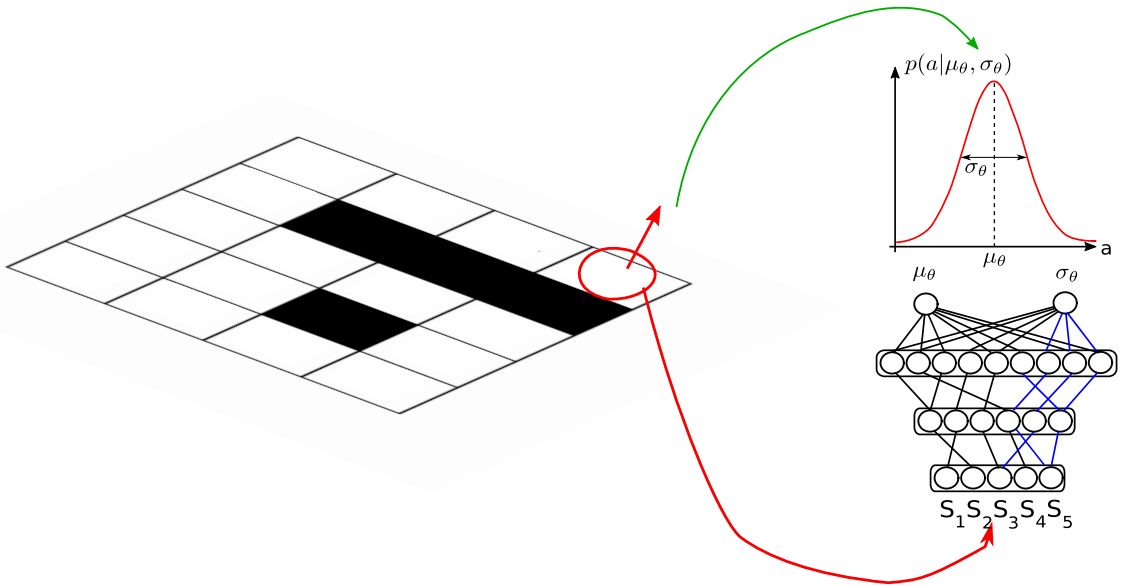
- Consider Gaussian stochastic policies
- If a $s_t \times a_t$ pair was rewarded, we want to increase its probability
- We feed s_t to the input, it provides μ, σ
- We compute the log probability of a_t given $\mathcal{N}(\mu, \sigma)$
- And we change $\boldsymbol{\theta}$ to increase it

Computing $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) R(\tau)$



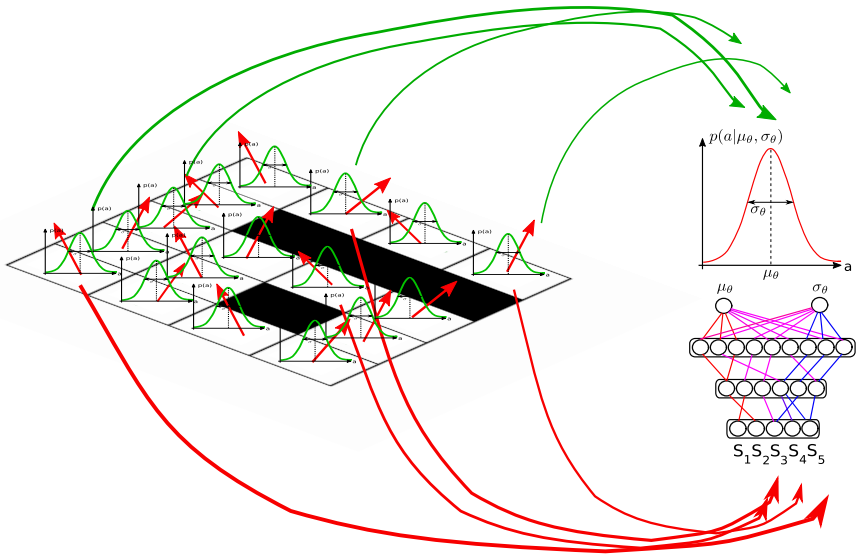
- We want to increase $\log \pi_{\boldsymbol{\theta}}(a_t|s_t) R(\tau)$ by changing $\boldsymbol{\theta}$ params
- We do so by minimizing the loss $L(\boldsymbol{\theta}) = -\log \pi_{\boldsymbol{\theta}}(a_t|s_t) R(\tau)$
- We compute the probability of a_t given $\mathcal{N}(\mu_\theta, \sigma_\theta)$
- $\pi_{\boldsymbol{\theta}}(a_t|s_t) = \exp(-\frac{(a_t - \mu_\theta)^2}{\sigma_\theta^2})$, thus $log \pi_{\boldsymbol{\theta}}(a_t|s_t) = -\frac{(a_t - \mu_\theta)^2}{\sigma_\theta^2}$
- The gradient is wrt $\mu_\theta, \sigma_\theta$, then backproped to $\boldsymbol{\theta}$
- Note, shown with μ_θ (fixed σ_θ)
- Other option: decrease σ_θ, get more deterministic

Policy gradient updates: considering another sample



- Then do the same for another $s \times a$ pair
- The new update is added to the update performed for the previous pair

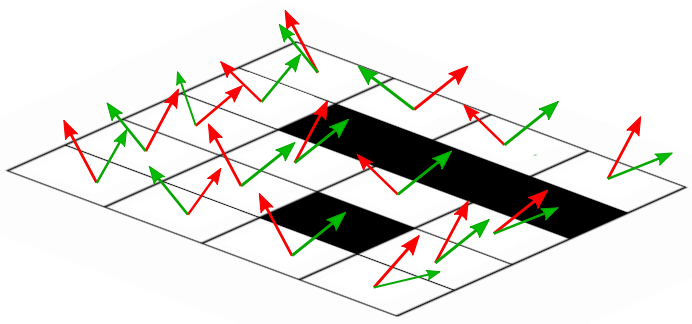
Policy gradient updates: all samples at a time

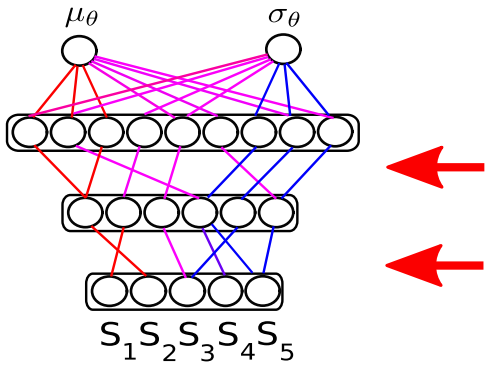


$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

- Thus the gradient of the reward is the mean sum over all the small gradients
- Since this is a sum, the order does not matter
- To speed up, one may rather take several minibatches
- Introduces variance in the applied updates

Locality of updates

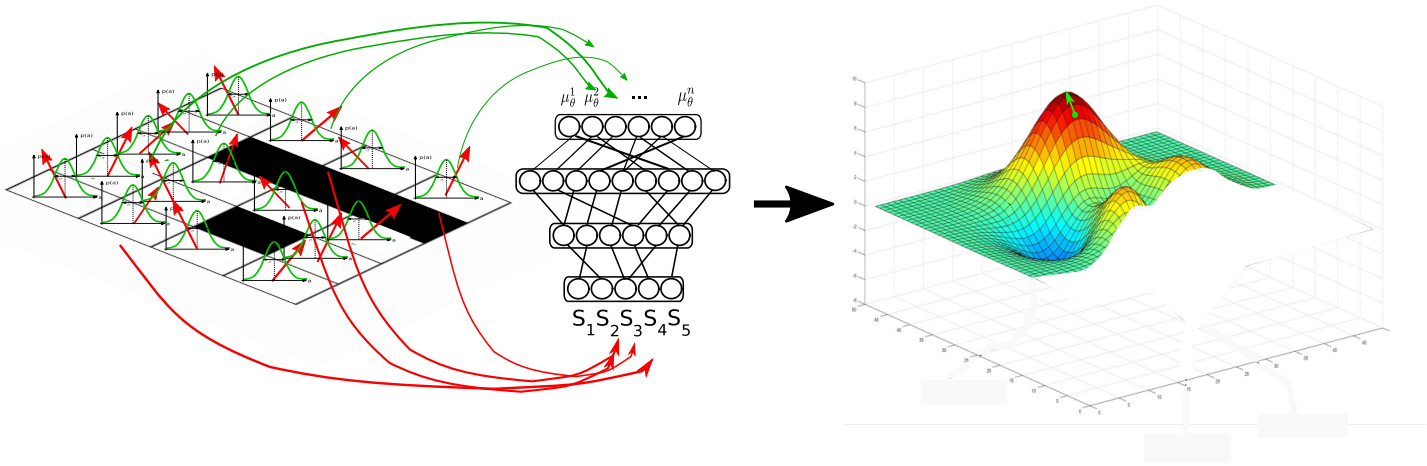




- If the set of impacted weights was different for each update, the global update would be perfect
- In practice, through the sum over weights, some updates contradict eachother
- It works "on average"
- If different intermediate neurons could represent different states, gradient for different state action pairs would not sum up
- The updates would be more independent
- To be studied: Is there a natural drive in deep RL for this disentangling?
- Topic: state representation learning

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., & Filliat, D. (2018) State representation learning for control: An overview. *Neural Networks*, 108:379–392

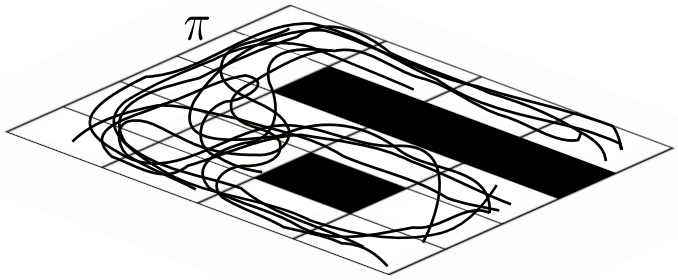
Policy gradient updates: local conclusion

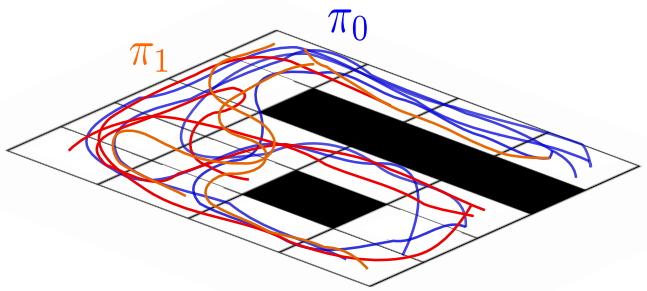


- Sample many trajectories *from $\pi_{\boldsymbol{\theta}}$*
- The policy gradient changes the proba of action locally in each state
- Changing each local action probability changes $\boldsymbol{\theta}$ globally
- Thus we apply a gradient step in the $J(\boldsymbol{\theta})$ landscape
- But many approximations make it so that the steepest ascent view does not hold:
- Locality vs step size, sum of gradients, large variance (limited sampling, mini-batches)...

Do policy gradient updates require less samples?

Sampling comparison





- In policy gradient, sample from $\pi_{\boldsymbol{\theta}}$
- Then derive a direction of gradient, but there is a large variance
- Need to sample many trajectories
- In variation-selection, sample from variations of $\pi_{\boldsymbol{\theta}}$
- Then select policies with a higher chance of being good

Lower variance in variation-selection?

- From Salimans: in RL, variance accumulates along each individual action
- In RL, the longer the trajectories, the more variance (a lower γ helps)
- In variation-selection, variance does not grow with the length of trajectories
- So variation-selection is more advantageous in problems with longer trajectories

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017

Exploration in sample collection

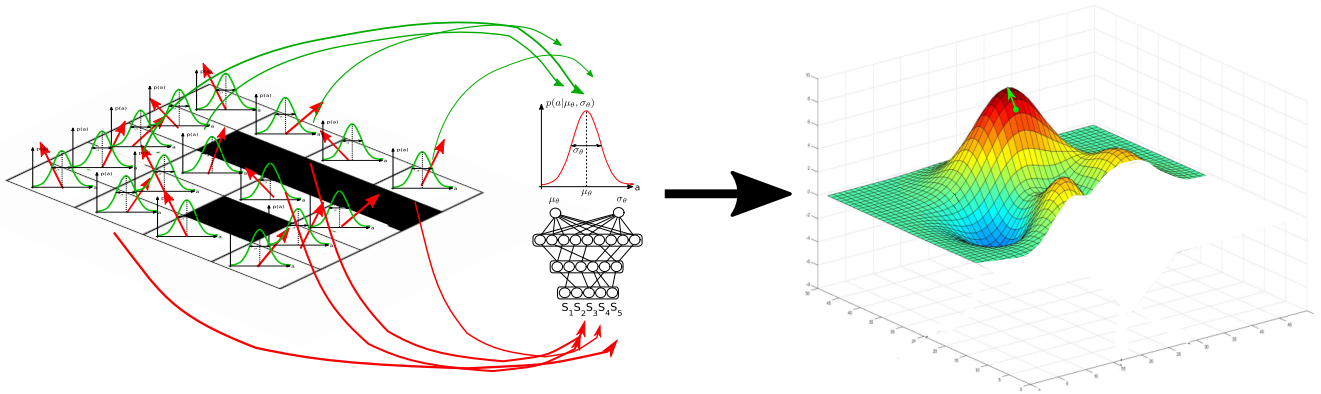
- When policies are stochastic, sampling from $\pi_{\boldsymbol{\theta}}$ may result in sampling the same $s \times a$ pairs as sampling from its variations $\pi_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}$
- This may be a source of inefficiency
- When converging to deterministic, no more gradient estimation, the networks degenerate
- Mechanisms to keep stochastic: natural gradient, entropy regularization
- See my TRPO, PPO and SAC videos
- In variation-selection, add noise to Gaussian covariance matrix to prevent collapse

Sampling: local conclusions

- Variation-selection may not require more sampling
- Higher sample efficiency of PG depends on the number of samples needed to get an accurate gradient estimate
- This is very environment-dependent

Does deep RL search in a better space?

Where does search take place?



- Search comes from the exploration part
- In variation-selection, exploration acts directly in the $\boldsymbol{\theta}$ space
- The $s \times a$ space is generally much smaller than the $\boldsymbol{\theta}$ space
- Searching the $s \times a$ space might be faster than searching the $\boldsymbol{\theta}$ space
- The policy gradient defines a change in action probabilities in the $s \times a$ space
- Then this change is implemented (without search) in the $\boldsymbol{\theta}$ space
- This may be an advantage, in small enough $s \times a$ spaces

Policy perturbation

- Most deep RL algos explore in the $s \times a$ space: adding noise to actions
- But a few add noise to policy parameters
- Policy parameter noise is often better than action noise in deep RL
- Better performance does not imply higher sample efficiency
- Using policy parameter noise is closer to variation-selection (acts over $\boldsymbol{\theta}$)
- Better investigate the difference

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017

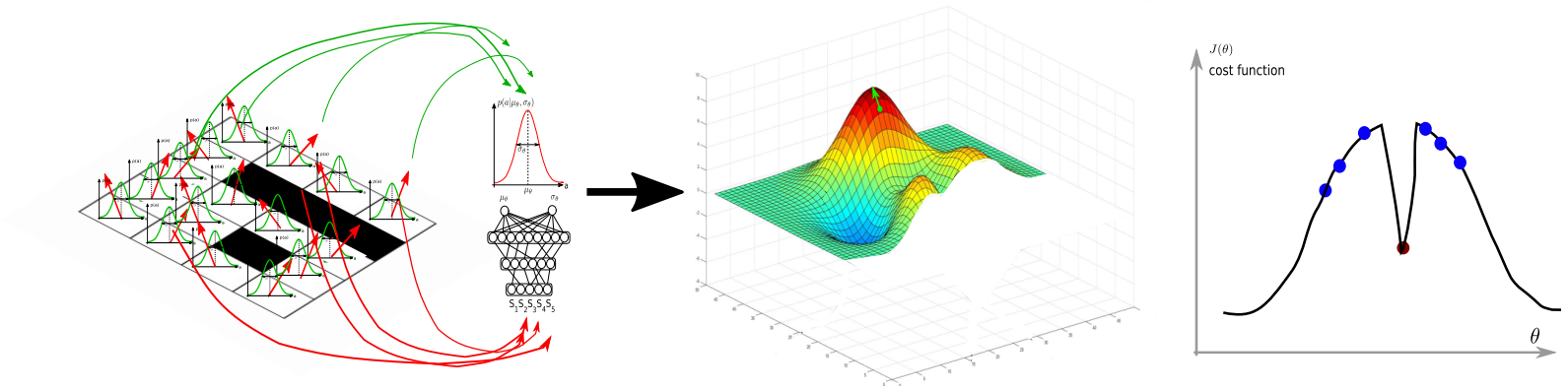
Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017

Impact of Markov assumption

- Local action improvement is more informed that pure black-box $J(\boldsymbol{\theta})$ improvement
- Improving each action locally improves the global performance
- True under Markov assumption, but not general
- Variation-selection methods do not rely on Markov assumption
- They are more general
- It is hard to conclude whether searching in $s \times a$ is better than in $\boldsymbol{\theta}$

Does deep RL search in the right direction, vs blind variations?

Deep RL is guessing, but variation-selection does so too

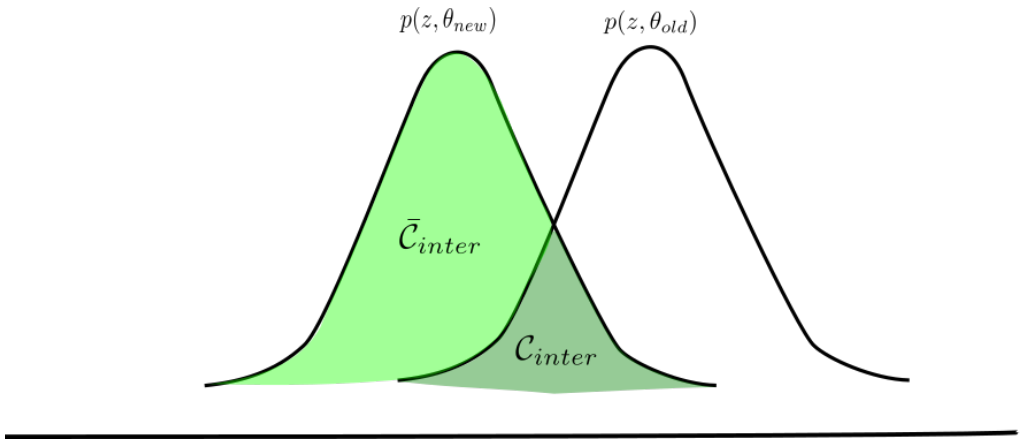


- In PG, take samples *from π_θ* and determine a local direction of change
- This is a strong guess
- In variation-selection, directly evaluate different policies over several trajectories
- Provides an accurate estimate of the value of policies it samples from
- Advocates in favor of greater robustness of variation-selection
- But this is not what CEM and CMA-ES do: they rather take a barycenter
- The iCEM algorithm proposes to restart from the best sample instead

Pinneri, C., Sawant, S., Blaes, S., Achterhold, J., Stueckler, J., Rolinek, M., & Martius, G. (2020) Sample-efficient cross-entropy method for real-time planning. *arXiv preprint arXiv:2008.06389*

Can RL reuse samples better than variation-selection?

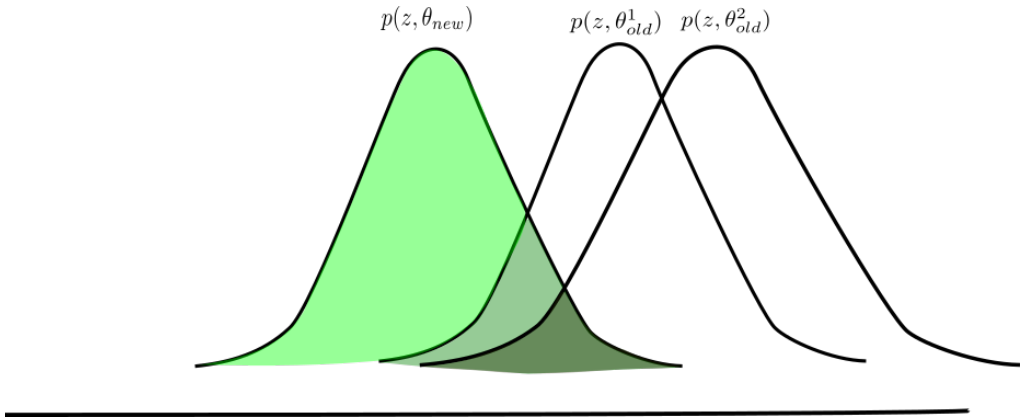
Sample reuse in ES: Importance Mixing



- A sample is a $< \boldsymbol{\theta}, J(\boldsymbol{\theta}) >$ pair
- No idea of recombining several "pieces of trajectories"
- To build a new generation, we can reuse samples from the previous one
- Importance Mixing does so: up to 90% gain in sample efficiency

Sun, Y., Wierstra, D., Schaul, T., & Schmidhuber, J. (2009) Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (pp. 539–546).: ACM.

Extendend Importance Mixing



- Importance Mixing can be further improved over several generations
- The gain is marginal

Pourchot, A., Perrin, N., & Sigaud, O. (2018) Importance mixing: Improving sample reuse in evolutionary policy search methods. *arXiv preprint arXiv:1808.05832*

Policy gradient is on-policy

- The policy gradient calculation assumes that the training trajectories are obtained from the current policy
- Reminder: we want to find $\text{argmax}_{\boldsymbol{\theta}} \sum_{\tau} P(\tau, \boldsymbol{\theta}) R(\tau)$
- We use

$$P(\tau^{(i)}, \boldsymbol{\theta}_{samp}) = \prod_{t=1}^{H} p(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) . \pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)} | s_t^{(i)})$$

- Here, by definition, $\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)} | s_t^{(i)})$ is the policy which generated the trajectories
- Then we take the gradient and get the policy gradient formula
- If we want to optimize another policy $\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)} | s_t^{(i)})$, the derivation is wrong
- If on-policy methods reuse samples, they are doing wrong

Importance sampling

- If we want to optimize another policy $\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})$, we can rewrite

$$P(\tau^{(i)}, \boldsymbol{\theta}_{other}) = \prod_{t=1}^{H} p(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}).\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)}).\frac{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}$$

- Or

$$P(\tau^{(i)}, \boldsymbol{\theta}_{other}) = \prod_{t=1}^{H} p(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}).\frac{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}.\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})$$

- And we can get

$$\nabla_{\boldsymbol{\theta}_{other}} J(\boldsymbol{\theta}_{other}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}\frac{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}\nabla_{\boldsymbol{\theta}_{other}}\log\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})R(\tau^{(i)})$$

- The term $\frac{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}$ is the importance sampling term
- Doing this is called "applying off-policy correction" (used in TRPO)
- To do this, you need to know $\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})$
- This is not the case e.g. when using human expert data

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015) Trust Region Policy Optimization. *CoRR, abs/1502.05477*

Being off-policy: using a replay buffer

- If $\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})$ is close enough to $\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})$ and the return landscape is smooth enough, performing policy gradient ascent without off-policy correction may work "more or less"
- Using a critic $V^*(s)$ or $Q^*(s,a)$ and perform gradient ascent on the policy with respect to the critic can make you more off-policy
- Off-policy RL algorithms can use a replay buffer
- They keep data from olders (or expert) policies and train from it
- Drastically improves sample efficiency
- But makes the algorithm more unstable
- No deep RL algorithm is "truly off-policy"
- Policies should not evolve too quickly (trust region, ...)
- Thus off-policy methods are more sample efficient than variation-selection methods, but they are more unstable

Conclusions

Intermediate messages

- The gradient gives the direction of steepest ascent: it improves faster
- $\Rightarrow$ Wrong: gradient estimation suffers from step size, averaging and large variance
- Gradient ascent does not need sampling:
- It uses analytical knowledge of the optimized function to improve it
- $\Rightarrow$ Wrong: it knows the policy, not the cost function
- RL uses info from each $s \times a$ pair, vs the whole episode for ES
- $\Rightarrow$ Yes, but performs averaging over $s \times a$ pairs
- $\Rightarrow$ Dependent on Markov assumption
- RL guesses a direction with data collected from the current policy, whereas variation selection uses data from other policies
- $\Rightarrow$ Yes, but ES performs averaging

Take home messages

- Variation-selection methods can be better parallelized
- Variation-selection provides poor sample reuse
- Yes, apart from importance mixing
- With RL we can reuse samples, if we are off-policy
- This dramatically improves sample efficiency
- Yes, but the algorithms run unstable
- Variation-selection may not perform so differently from on-policy RL
- The most sample efficient case is "offline RL"
- But it is hard to provide performance guarantees

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020

Any question?

Send mail to: `Olivier.Sigaud@upmc.fr`


De Boer, P.-T., Kroese, D., Mannor, S., and Rubinstein, R. (2005).
A tutorial on the cross-entropy method.
Annals of Operations Research, 134(1):19–67.


Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017).
Noisy networks for exploration.
arXiv preprint arXiv:1706.10295.


Khadka, S. and Tumer, K. (2018).
Evolution-guided policy gradient in reinforcement learning.
In *Neural Information Processing Systems*.


Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. (2018).
State representation learning for control: An overview.
Neural Networks, 108:379–392.


Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020).
Offline reinforcement learning: Tutorial, review, and perspectives on open problems.
arXiv preprint arXiv:2005.01643.


Pagliuca, P., Milano, N., and Nolfi, S. (2019).
Efficacy of modern neuro-evolutionary strategies for continuous control optimization.
arXiv preprint arXiv:1912.05239.


Pierrot, T., Perrin, N., and Sigaud, O. (2018).
First-order and second-order variants of the gradient descent: a unified framework.
arXiv preprint arXiv:1810.08102.


Pinneri, C., Sawant, S., Blaes, S., Achterhold, J., Stueckler, J., Rolinek, M., and Martius, G. (2020).
Sample-efficient cross-entropy method for real-time planning.
arXiv preprint arXiv:2008.06389.


Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. (2017).

Parameter space noise for exploration.
arXiv preprint arXiv:1706.01905.

Pourchot, A., Perrin, N., and Sigaud, O. (2018).

Importance mixing: Improving sample reuse in evolutionary policy search methods.
arXiv preprint arXiv:1808.05832.

Ruder, S. (2016).

An overview of gradient descent optimization algorithms.
arXiv preprint arXiv:1609.04747.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. (2017).

Evolution strategies as a scalable alternative to reinforcement learning.
arXiv preprint arXiv:1703.03864.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015).

Trust region policy optimization.
In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org.

Sigaud, O. and Stulp, F. (2019).

Policy search in continuous action domains: an overview.
Neural Networks, 113:28–40.

Sun, Y., Wierstra, D., Schaul, T., and Schmidhuber, J. (2009).

Efficient natural evolution strategies.
In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546. ACM.