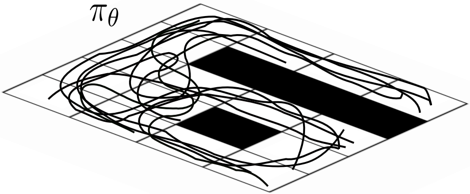
Direct Policy Search vs Reinforcement Learning

A closer look at policy gradient updates

Olivier Sigaud

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

Reminder: policy gradient updates

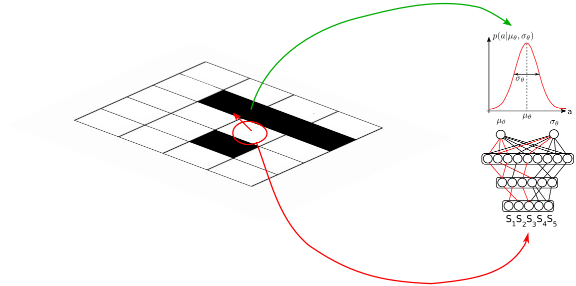


- Sample a set of m trajectories from $\pi_{\boldsymbol{\theta}}$
- Compute:

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

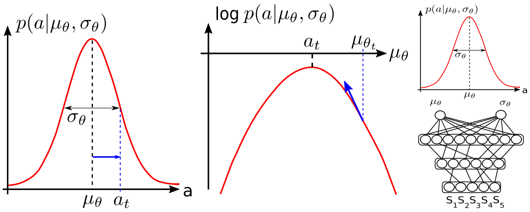
- Side note: if $R(\tau) = 0$, does nothing
- Update parameters $\boldsymbol{\theta}$ of $\pi_{\boldsymbol{\theta}}$ using $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \alpha_i \nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta})$ (or something more advanced)
- Iterate: sample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

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



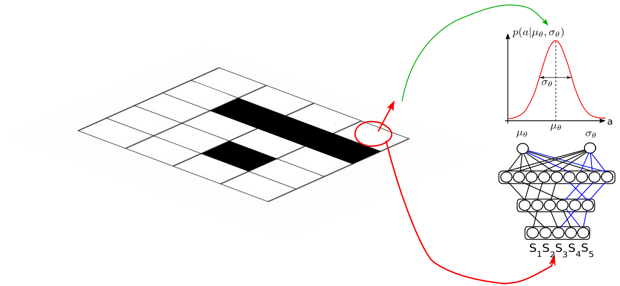
- We consider a Gaussian stochastic policy with state-dependent variance
- If a $s_t \times a_t$ pair was rewarded ($R(\tau) > 0$), we want to update $\boldsymbol{\theta}$ to increase its probability
- We feed s_t to the input, it provides $\mu_{\boldsymbol{\theta}}, \sigma_{\boldsymbol{\theta}}$
- We compute the log probability of a_t given $\mathcal{N}(\mu_{\boldsymbol{\theta}}, \sigma_{\boldsymbol{\theta}})$
- And we change $\boldsymbol{\theta}$ to increase it

Increasing $\log\pi_{\boldsymbol{\theta}}(a_t|s_t)$ at some (s_t, a_t)



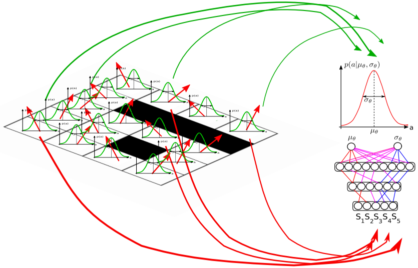
- We want to increase $\log\pi_{\boldsymbol{\theta}}(a_t|s_t)$ by changing $\boldsymbol{\theta}$ params
- We do so by minimizing the loss $L(\boldsymbol{\theta}) = -\log\pi_{\boldsymbol{\theta}}(a_t|s_t)$
- We compute the probability of a_t given $\mathcal{N}(\mu_\theta, \sigma_\theta)$
- $\pi_{\boldsymbol{\theta}}(a_t|s_t) = \exp(-\frac{(a_t - \mu_\theta)^2}{\sigma_\theta^2})$, 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 update $\boldsymbol{\theta}$
- Note, shown with μ_θ (fixed σ_θ)
- Other option: decrease σ_θ, get more deterministic
- Experimentally, decreasing σ_θ has more impact (need for entropy regularization)

Policy gradient updates: considering another sample



- The previous update moved $\boldsymbol{\theta}$ in some direction
- Then we do the same for another $s \times a$ pair
- The new update is added to the update performed for the previous pair
- It moves moved $\boldsymbol{\theta}$ in some other direction

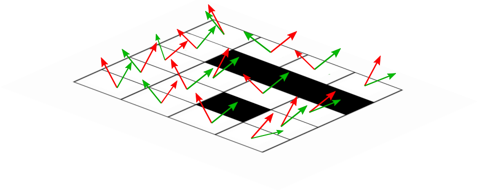
Policy gradient updates: all samples at a time

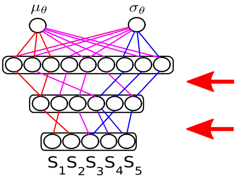


$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

- 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 further variance in the applied updates

Locality of updates

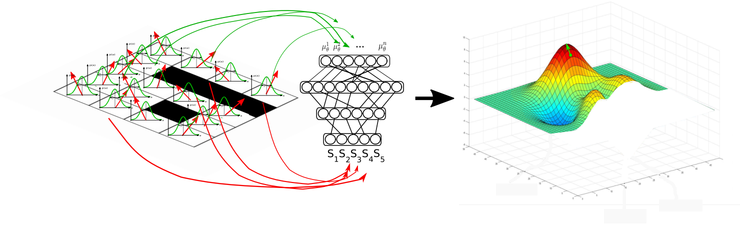




- If the set of impacted weights was different for each update, the global update would be perfect
- In practice, through the sum over weights, some updates contradict eachother
- It works "on average"
- If different intermediate neurons could represent different states, gradient for different state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

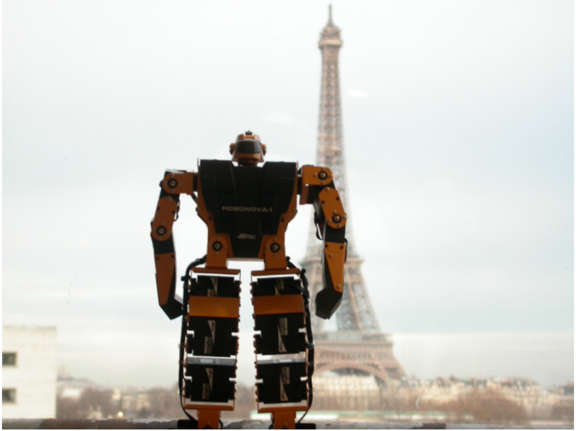
Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., & Filliat, D. (2018) State representation learning for control: An overview. *Neural Networks*, 108:379–392

Policy gradient updates: local conclusion



- Sample many trajectories from a single $\pi_{\boldsymbol{\theta}}$ (a point in the $J(\boldsymbol{\theta})$ landscape)
- The policy gradient changes the proba of actions locally in each state
- By updating each local action probability, $\boldsymbol{\theta}$ changes globally (we move to another point)
- This ends up in applying a gradient step in the $J(\boldsymbol{\theta})$ landscape
- But note that even with perfect sampling, the sum over local gradient steps does guarantee moving to a better policy

Any question?



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

ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE

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