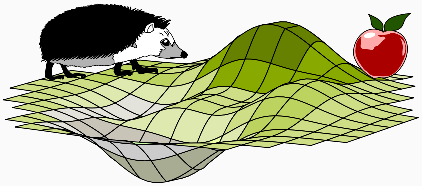
REINFORCEMENT LEARNING

DEEP REINFORCEMENT LEARNING

Pavel Osinenko



Deep reinforcement learning

Consider the following ∞-horizon optimal control problem:

$$\max_{\kappa} J(x_0 \mid \kappa) = \int_0^\infty \rho(x(t), \kappa(x(t)))\, dt$$

$$\text{s.t.} \quad \dot{x} = f(x, u)$$

Recall the HJB:

$$\max_{u \in \mathbb{U}} \left\{ \mathcal{L}_{f(x,u)} \mathcal{V}(x) + \rho(x, u) \right\} = 0, \ \forall x,$$

where the Lie derivative reads:

$$\mathcal{L}_{f(x,u)} \mathcal{V}(x) = \nabla \mathcal{V}(x)^T f(x, u) .$$

We consider here a deterministic environment only for the ease of exposition

Deep reinforcement learning

The corresponding Hamiltonian reads :

$$\mathcal{H}(x, u \mid h) := \mathcal{L}_{f(x,u)} h(x) + \rho(x,u)$$

for a generic function h of x.

If κ^* is the optimal policy , then the HJB can be rewritten as:

$$\mathcal{H}(x, \kappa^*(x) \mid V) = 0 \ , \ \forall x$$

We will learn the value function via a deep neural network assuming a suitable (stabilizing and exploring) behavior policy generating $u(t), t \geq 0$

Deep reinforcement learning

For the sake of brevity, let's take a two-layer network, whereas the argument will work for any topology in a similar manner.

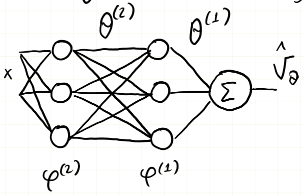
So,

$$\forall x \quad \hat{V}_\theta(x) := \theta^{(1)^T} \varphi^{(1)}\left(\theta^{(2)^T} \varphi^{(2)}(x) \right),$$

where $\theta^{(1)}$ is the weight vector of the output layer, $\theta^{(2)}$ is the weight matrix of the hidden layer, $\varphi^{(1)}, \varphi^{(2)}$ are the corresponding activation functions.

Details: $\theta^{(1)}$ is $N_{c1} \times 1$

$\theta^{(2)}$ is $N_{c1} \times N_{c2}$



Deep reinforcement learning

Let's write down the gradient:

$$\nabla_x \hat{V}_\theta(x) = \left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_j \right)_j^{N_{c1}} \left(\frac{\partial \varphi^{(2)}}{\partial o^{(2)}} \right)^T \theta^{(2)},$$

where $o^{(2)}$ is the output of the hidden layer:

$$o^{(2)} := \theta^{(2)^T} \varphi^{(2)}(x) .$$

The notation $\theta^{(2)}_j$ means the jth row of the matrix $\theta^{(2)}$ and $(\bullet_j)_j^{N_{c1}}$ means horizontal stacking over j, i.e.,

$$(\bullet_1, \bullet_2, \ldots, \bullet_{N_{c1}})$$

Deep reinforcement learning

The Hamiltonian for the critic reads:

$$\mathcal{H}(x,u \mid \hat{V}_\theta) = \mathcal{L}_{f(x,u)} \hat{V}_\theta(x) + \rho(x,u)$$
$$= \nabla_x \hat{V}_\theta(x)^T f(x,u) + \rho(x,u)$$

By a universal approximation theorem,

$$V(x) = \theta^{*(1)^T} \varphi^{(1)}\left(\theta^{*(2)^T} \varphi^{(2)}(x)\right) + \delta(x)$$

(on a compact).

Here, $\theta^* = \{ \theta^{(1)^T}, \theta^{(2)^T} \}$ is the set of the best weights

Deep reinforcement learning

Accordingly, the Hamiltonian approximation error reads:

$$\begin{aligned}\delta_{\mathcal{H}}(x,u) &:= \mathcal{H}(x,u \,|\, V) - \mathcal{H}(x,u \,|\, \hat{V}_{\theta^*}) \\ &= \mathcal{L}_{f(x,u)} V(x) - \mathcal{L}_{f(x,u)} \hat{V}_{\theta^*} \\ &= \mathcal{L}_{f(x,u)} \delta(x) \,.\end{aligned}$$

The Hamiltonian TD in turn is:

$$\begin{aligned}e_{\mathcal{H}}(\theta \,|\, x,u) &:= \mathcal{H}(x,u \,|\, \hat{V}_{\theta}) - \mathcal{H}(x, \kappa^*(x) \,|\, V) \\ &= \mathcal{H}(x,u \,|\, \hat{V}_{\theta}) \,,\end{aligned}$$

where κ^* is the optimal policy

Deep reinforcement learning

In more detail,

$$H(x,u/\hat{V}_\theta) = \theta^{(1)^T} \frac{\partial \varphi^{(1)}}{\partial \Theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_{:j} \right)_j^{N_{c1}} \right)^T f(x,u) + \rho(x,u).$$

Let us compute the gradients of the Hamiltonian TD:

$$\nabla_{\theta^{(1)}} \mathcal{H}(x,u/\hat{V}_\theta) = \frac{\partial \varphi^{(1)}}{\partial \Theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_{:j} \right)_j^{N_{c1}} \right)^T f(x,u)$$

(this part resembles the one in the case of a 1-layer network)

Deep reinforcement learning

For the second part, let's introduce the notation

$$G(\theta^{(2)}, x) := \frac{\partial \varphi^{(1)}}{\partial \theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_j \right)_j^{N_{c1}} \right)^T .$$

Then, $\mathcal{H}(x, u \mid \hat{v}_\theta) = \theta^{(1)^T} G(\theta^{(2)}, x)\, f(x,u) + \rho(x,u)$.

Let's do some unwrapping :

$$\mathcal{H}(x, u \mid \hat{v}_\theta) = \left(\left(\theta^{(1)^T} G_\ell^T (\theta^{(2)}, x) \right)_\ell^n \right)^T f(x,u) + \rho(x,u)$$

$$= \sum_{\ell=1}^{n} \theta^{(1)^T} G_\ell^T (\theta^{(2)}, x)\, f_\ell(x,u) + \rho(x,u)$$

Deep reinforcement learning

Let's abuse the notation and interpret $\theta^{(2)}$ as a vector while computing

$$\nabla_{\theta^{(2)}} \bullet$$

to mean, actually, $\nabla_{vec(\theta^{(2)})}$ where $vec(\cdot)$ is a transformation of a matrix into a vector. Alternatively, we could always interpret $\theta^{(2)}$ as a vector and reshape it into matrix to compute $\hat{\nabla}_\theta$.

Then,

$$\nabla_{\theta^{(2)}} \mathcal{H}(x, u \,|\, \hat{\nabla}_\theta) = \sum_{l=1}^{n} f_l(x,u) \nabla_{\theta^{(2)}} G_l^T(\theta^{(2)}, x) \, \theta^{(1)^T}$$

Deep reinforcement learning

So far, we have:

$$\nabla_\theta \mathcal{H}(x,u \,|\, \hat{V}_\theta) = \begin{pmatrix} \frac{\partial \varphi^{(1)}}{\partial \theta^{(2)}}\left(\left(\nabla_x \varphi^{(2)}(x)\,\theta_j^{(2)}\right)_j^{N_{c1}}\right)^T f(x,u) \\ \sum_{l=1}^{n} f_l(x,u)\, \nabla_{\theta^{(2)}}\left(\left(\nabla_x \varphi^{(2)}(x)\,\theta_j^{(2)}\right)_j^{N_{c1}} \frac{\partial \varphi^{(1)}}{\partial \theta^{(2)}}\right)_l \theta^{(1)^T} \end{pmatrix}.$$

Be aware that $\frac{\partial \varphi^{(1)}}{\partial \theta^{(2)}}$ in general depends nonlinearly on $\theta^{(2)}$!

Deep reinforcement learning

Let us linearize $\mathcal{H}(x,u \mid \hat{V}_{\theta^*})$ around ϑ :

$$\mathcal{H}(x,u \mid \hat{V}_{\theta^*}) = \theta^{(1)^T} G(\theta^{(2)}, x) f(x,u) -$$

$$\left(\begin{array}{c} \frac{\partial \varphi^{(1)}}{\partial \Theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_j \right)_j^{N_{c1}} \right)^T f(x,u) \\ \sum_{l=1}^{n} f_l(x,u)\, \nabla_{\theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_j \right)_j^{N_{c1}} \frac{\partial \varphi^{(1)}}{\partial \Theta^{(2)}} \right)_l \theta^{(1)^T} \end{array} \right)^T \Bigg|_{\theta = \vartheta} \operatorname{vec}(\tilde{\theta}) +$$

$$\mathcal{O}\left(\| \operatorname{vec}(\tilde{\theta}) \|^2 \right) + \rho(x,u),$$

where $\operatorname{vec}(\tilde{\Theta}) := \operatorname{vec}(\Theta) - \operatorname{vec}(\Theta^*) = \left(\theta^{(1)^T}, \operatorname{vec}(\theta^{(2)})^T\right)^T - \left(\theta^{*(1)^T}, \operatorname{vec}(\theta^{*(2)})^T\right)^T$

Deep reinforcement learning

(cont.)

Expessing $\rho(x,u)$ from the last equality and plugging into $e_{\mathcal{H}}(\theta | x,u)$ yields:

$$e_{\mathcal{H}}(\theta|x,u) = \left.\begin{pmatrix} \frac{\partial \varphi^{(1)}}{\partial \Theta^{(2)}}\left(\left(\nabla_x \varphi^{(2)}(x)\,\theta^{(2)}_{:j}\right)_j^{N_{c1}}\right)^T f(x,u) \\ \sum_{l=1}^{n} f_l(x,u)\,\nabla_{\theta^{(2)}}\left(\left(\nabla_x \varphi^{(2)}(x)\,\theta^{(2)}_{:j}\right)_j^{N_{c1}} \frac{\partial \varphi^{(1)}}{\partial \Theta^{(2)}}\right)_l \theta^{(1)^T} \end{pmatrix}^T\right|_{\vartheta=\theta} \operatorname{vec}(\tilde{\theta}) +$$

$$\mathcal{H}(x,u \mid \hat{V}_{\theta^*}) + \mathcal{O}(\|\operatorname{vec}(\tilde{\theta})\|^2)$$

Deep reinforcement learning

Let the data vector be defined as:

$$\varpi(x,u,\theta) := \left(\begin{array}{c} \frac{\partial \varphi^{(1)}}{\partial \theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_j \right)_j^{N_{c1}} \right)^T f(x,u) \\ \\ \sum_{l=1}^{n} f_l(x,u)\, \nabla_{\theta^{(2)}} \left(\left(\nabla_x \varphi^{(2)}(x)\, \theta^{(2)}_j \right)_j^{N_{c1}} \frac{\partial \varphi^{(1)}}{\partial \theta^{(2)}} \right)_l \theta^{(1)^T} \end{array} \right) \Bigg|_{\theta = \theta} .$$

Notice it depends on θ, but we do not expect troubles with persistence of excitation because of that since the weights shouldn't normally converge to zero

Deep reinforcement learning

Summarizing, we have:

$$e_{\mathcal{H}}(\theta / x, u) = \operatorname{vec}(\tilde{\theta})^T \varpi(x, u, \theta) + \mathcal{H}(x, u / \hat{V}_{\theta^*}) + \mathcal{O}(\|\operatorname{vec}(\tilde{\theta})\|^2).$$

Relating this to the true Hamiltonian gives:

$$e_{\mathcal{H}}(\theta / x, u) = \operatorname{vec}(\tilde{\theta})^T \varpi(x, u, \theta) + \mathcal{H}(x, u / V) - \delta_{\mathcal{H}}(x, u) + \mathcal{O}(\|\operatorname{vec}(\tilde{\theta})\|^2)$$

Deep reinforcement learning

As in the case of a shallow critic, let's define the loss as:

$$J^c\left(\theta \mid \{x(t_k), u(t_k)\}_k^M\right) := \frac{1}{2} \sum_{k=1}^{M} \frac{e_{\mathcal{H}}^2(\theta \mid x(t_k), u(t_k))}{(\omega_k^T \omega_k + 1)^2},$$

where M is the size of the experience replay $\{x(t_k), u(t_k)\}_k^M$ with $t_M = t$. The data points may be equally distributed. Also, we used a short-hand notation

$$\omega_k := \omega(x(t_k), u(t_k), \theta)$$

Deep reinforcement learning

Accordingly, let's update the critic weights by gradient descent over mini-batches:

$$\frac{d}{dt}\mathrm{vec}(\theta) := -\alpha \nabla_\theta J^c\left(\theta \,\middle|\, \{x(t_k), u(t_k)\}_k^M\right):$$

$$= -\alpha \sum_{k=1}^{M} \frac{e_{\mathcal{H}}(\theta \,|\, x(t_k), u(t_k))\, \nabla_\theta e_{\mathcal{H}}(\theta \,|\, x(t_k), u(t_k))}{\left(\omega_k^T \omega_k + 1\right)^2},$$

where α is the learning rate.

Since $\nabla_\theta e_{\mathcal{H}}(\theta \,|\, x(t_k), u(t_k)) = \omega_k$, we have:

$$\frac{d}{dt}\mathrm{vec}(\theta) = -\alpha \sum_{k=1}^{M} \frac{e_{\mathcal{H}}(\theta \,|\, x(t_k), u(t_k))\, \omega_k}{\left(\omega_k^T \omega_k + 1\right)^2}$$

Deep reinforcement learning

Next, recalling that

$$e_{\mathcal{H}}(\theta|x,u) = \mathrm{vec}(\tilde{\theta})^T \omega(x,u,\theta) + \mathcal{H}(x,u|V) - \delta_{\mathcal{H}}(x,u) + \mathcal{O}(\|\mathrm{vec}(\tilde{\theta})\|^2),$$

rewrite the update rule as follows:

$$\frac{d}{dt}\mathrm{vec}(\theta) = -\alpha \sum_{k=1}^{M} \frac{\omega_k \omega_k^T}{(\omega_k^T \omega_k + 1)^2} \mathrm{vec}(\tilde{\theta}) +$$

$$\alpha \sum_{k=1}^{M} \frac{\omega_k \left(\mathcal{H}(x(t_k), u(t_k) | V) - \delta_{\mathcal{H}}(x(t_k), u(t_k)) + \mathcal{O}(\|\mathrm{vec}(\tilde{\theta})\|^2) \right)}{(\omega_k^T \omega_k + 1)^2}$$

Deep reinforcement learning

Define the following data matrix :

$$\mathcal{E}(t) := \sum_{k=1}^{M} \frac{\omega_k \omega_k^T}{(\omega_k^T \omega_k + 1)^2} .$$

Then, proceeding to the norm and noticing that $\frac{d}{dt} vec(\Theta) = \frac{d}{dt} vec(\tilde{\Theta})$, we have:

$$\frac{d}{dt} \| vec(\tilde{\Theta}) \|^2 = -\alpha \| vec(\tilde{\Theta}) \|^2_{\mathcal{E}(t)} +$$

$$\alpha \sum_{k=1}^{M} \frac{\omega_k \left(vec(\tilde{\Theta})^T \mathcal{H}(x(t_k), u(t_k) | V) - \delta_{\mathcal{H}}(x(t_k), u(t_k)) + \mathcal{O}(\| vec(\tilde{\Theta}) \|^3) \right)}{(\omega_k^T \omega_k + 1)^2} ,$$

where $\| vec(\tilde{\Theta}) \|^2_{\mathcal{E}(t)} = vec(\tilde{\Theta})^T \mathcal{E}(t) \, vec(\tilde{\Theta})$

Deep reinforcement learning

Since the behaviour policy is stabilizing, noticing that $\frac{\|\omega\|}{(\|\omega\|^2+1)^2} \leq \frac{1}{2}, \forall \omega$, and assuming $\mathcal{H}, V$ are continuous, there are constants $\bar{\mathcal{H}}, \bar{\delta}_{\mathcal{H}}, C > 0$ s.t.

$$\alpha \sum_{k=1}^{M} \frac{\omega_k \left(vec(\tilde{\theta})^T \mathcal{H}(x(t_k), u(t_k) | V) - \delta_{\mathcal{H}}(x(t_k), u(t_k)) + C \|vec(\tilde{\theta})\|^3 \right)}{\left(\omega_k^T \omega_k + 1 \right)^2} \leq$$

$$\frac{\alpha M}{2} \left(\|vec(\tilde{\theta})\| \cdot \left(\bar{\mathcal{H}} + \bar{\delta}_{\mathcal{H}} \right) + C \|vec(\tilde{\theta})\|^3 \right)$$

Deep reinforcement learning

Assuming persistence of excitation, i.e., $\exists \varepsilon > 0$ s.t. $\forall t \geq 0 \quad \mathcal{E}(t) \succcurlyeq \varepsilon I$, we may claim

$$\frac{d}{dt} \|vec(\tilde{\theta})\|^2 \leqslant -\alpha \varepsilon \|vec(\tilde{\theta})\|^2 + \frac{\alpha M}{2} \left(\|vec(\tilde{\theta})\| \cdot \left(M + \bar{\delta}_{\mathcal{H}}\right) + C \|vec(\tilde{\theta})\|^3 \right).$$

In absense of the linearization error $C\|vec(\tilde{\theta})\|^3$, the argumentation about the critic weight convergence would have been the same as in the case of a shallow network: the weights converge from everywhere into a vicinity of the best weights

Deep reinforcement learning

$$\frac{d}{dt}\|vec(\tilde{\Theta})\|^2 \leqslant -\alpha\varepsilon\|vec(\tilde{\Theta})\|^2 + \frac{\alpha M}{2}\left(\|vec(\tilde{\Theta})\|\cdot\left(\underline{M} + \bar{\delta}_{\mathcal{H}}\right) + C\|vec(\tilde{\Theta})\|^3\right)$$

This vicinity depends primarily on two factors:

1. how good the chosen network is capable of approximating $\sqrt{}$ and thus $\mathcal{H}$ (affects $\bar{\delta}_{\mathcal{H}}$);
2. how close the behavior policy is to the optimal one (affects $\underline{M}$)

Deep reinforcement learning

$$\frac{d}{dt}\|vec(\tilde{\theta})\|^2 \leqslant -\alpha\varepsilon\|vec(\tilde{\theta})\|^2 +$$

$$\frac{\alpha M}{2}\left(\|vec(\tilde{\theta})\|\cdot\left(M + \bar{\delta}_{\mathcal{H}}\right) + C\|vec(\tilde{\theta})\|^3\right)$$

(cont.) The point 2. could in theory be tackled by combination of exploration/exploitation phases thus, hopefully, approaching the optimal policy.

Question: is such a method on- or off-policy?

Important remark: there is no guarantee of stabilizability during the exploitation following this recipe

Deep reinforcement learning

$$\frac{d}{dt}\|vec(\tilde{\theta})\|^2 \leqslant -\alpha\varepsilon\|vec(\tilde{\theta})\|^2 + \frac{\alpha M}{2}\left(\|vec(\tilde{\theta})\|\cdot\left(M + \bar{\delta}_{\mathcal{H}}\right) + C\|vec(\tilde{\theta})\|^3\right)$$

(cont.) The key principle utilized in the convergence analysis is that the first, quadratic term dominates the linear term outside a vicinity of zero. But in the case of a deep neural network critic, there is a higher-order term which makes an essential difference

Deep reinforcement learning

$$\frac{d}{dt}\|vec(\tilde{\theta})\|^2 \leqslant -\alpha\varepsilon\|vec(\tilde{\theta})\|^2 +$$

$$\frac{\alpha M}{2}\left(\|vec(\tilde{\theta})\|\cdot\left(M + \bar{\delta}_{\mathcal{H}}\right) + C\|vec(\tilde{\theta})\|^3\right)$$

(cont.) The principle here is exactly the opposite: if the weight error is not too large initially, it converges. Now, we are faced by the problem that the convergence domain is limited by this factor and also by what we discussed earlier — the effect of the Hamiltonian approximation error and goodness of the behavior policy

Deep reinforcement learning

Convergence of the actor weights is performed in a similar manner. Again, linearization errors affect the convergence.

But local character of convergence is a general problem of deep learning.

A further remark on the derivation of the update rule: in general, back propagation is the systematic routine to analytically derive the gradients

$$\nabla_\theta \mathcal{H}(x, u | \hat{V}_\theta)$$

Deep reinforcement learning

(cont.)

Furthermore, modifications of the gradient descent usual in deep learning may be applied, such as momentum techniques, Nesterov accelerated gradient descent; gradient clipping to avoid fading/explosion (at deep layers) may also be of use, as well as such techniques as dropout (switching off the network partly). Next, ReLu activation functions can be employed whose gradients are either one or zero

Deep reinforcement learning

An overview of deep RL methods:

- DQN („deep QL") and double DQN
- DDPG („DPG + DQN")
- TD3 (twin delayed deep deterministic): a modification of DDPG
- αGo Zero: uses a single deep network to fit the policy and value, updates via Monte-Carlo tree search