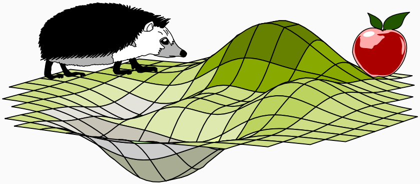
REINFORCEMENT LEARNING

ACTOR-CRITIC

Pavel Osinenko



Actor-critic

Consider an environment

$$X_{k+1} \sim \mathbb{P}_X(x_{k+1} \mid x_k, u_k)$$
$$U_k \sim \mathbb{P}_U^{\vartheta}(u_k \mid x_k)$$

along with an ∞-horizon optimal control problem

$$\max_{\vartheta} J(x_0 \mid \vartheta) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \rho(X_k, U_k) \mid X_0 = x_0\right]$$

We seek to optimize $J(\bullet \mid \vartheta)$ via the distribution parameters ϑ

Recalls :

- the optimal control problem is „split" into a head and tail parts that are related via the DPP
- DPP yields HJB
- HJB is a fixed-point equation
- For discounted problems, application of HJB „to itself" converges to the value function, whence VI, PI
- DP is a tabular method
- Policy gradient (PG) seeks to find optimal policy by gradient ascent/descent

A typical form of PG (REINFORCE)

$$\mathbb{E}\left[\sum_{k=0}^{\infty} \nabla_\vartheta \ln P_U^\vartheta(U_k \mid X_k)\, Q^\vartheta(X_k, U_k) \,\Big|\, X_0 = x_0\right],$$

where:

$$Q^\vartheta(x,u) = \mathbb{E}_{\{U_\bullet\}\sim\{P^\vartheta_{U_\bullet}\},\, \{X_\bullet\}\sim\{P_{X_\bullet}\}}\left[\rho(x,u) + \sum_{k=1}^{\infty} \gamma^k \rho(X_k, U_k)\right],$$

the Q-function of the P_U^ϑ policy at x, u

How do we find Q?

One direct way is to use Monte-Carlo methods.

This will work if episodes terminate.

Alternatively, we can exploit the HJB

Actor-critic

Let's recall it for the stated problem:

$$V(x) = \max_{u} \left\{ \rho(x, u) + \gamma \mathbb{E} \left[V(X_+^u) \right] \right\},$$

where:

$$V(x) := \max_{\vartheta} J(x \mid \vartheta), \text{ the value function;}$$

$$X_+^u \sim P_X(x_+ \mid x, u), \text{ the next state.}$$

Recall the verification principle: find a solution to HJB first, the show it's the value function. This is what we are doing in DP. We are explicitly constructing a solution

Actor-critic

In a nutshell, we are trying to „equalize" the left-hand and the right-hand side of HJB. This simple idea lies at the foundation of so-called temporal-difference* (TD) methods.

Let's rewrite HJB in terms of Q:

$$Q(x,u) = \rho(x,u) + \gamma \max_{\vartheta} \mathbb{E}\left[Q^{\vartheta}(X^{+}, U) \right]$$

In a form of Q-learning, we may have iterations like:

$$Q_{i+1}(x,u) := \rho(x,u) + \gamma \, \mathbb{E}_{U \sim \rho_U^{\vartheta_i}} \left[Q_i(X_{+}^{U}, U) \,\middle|\, x \right], \quad \forall x,u$$

* TD is also referred to as Bellman error

Actor-critic

(cont.)

This kind of iteration tries to „equalize" both sides of HJB as can be noticed.

The „error" is the TD:

$$e_i^{\vartheta_i}(x,u) := \rho(x,u) + \gamma \mathbb{E}_{U \sim \rho_U^{\vartheta_i}}\left[Q_i(X_+^U, U) \mid x \right] - Q_i(x,u)$$

It reflects how good HJB was „solved". Notice the temporal shift from x to X_+^U.

Ideally, we want TD $= 0$. One of the ways to solve such an equation is to iterate as:

$$Q_{i+1} := Q_i + \alpha e_i$$

with a learning rate α

(cont.)

Such kind of a method is also known as TD(0).

Now, we can write down the SARSA update rule:

$$Q_{i+1}(x,u) := Q_i(x,u) + \alpha\Big(\underbrace{\rho(x,u) + \gamma \mathbb{E}_{U \sim \rho_U^{\vartheta_i}}\left[Q_i(X_+^U, U) \mid x\right]} - Q_i(x,u)\Big), \forall x,u$$

This is also called the update target $O_i^{\vartheta_i}(x,u)$

SARSA is also known as on-policy Q-learning because in learning the Q-function, we follow the same (target) policy which we are also trying to learn, namely, the one given by the PDF $\rho_U^{\vartheta_i}$. Furthermore, the method is akin to bootstrapping because we are using estimates of the same variable, namely, Q_i

(cont.)

In an off-policy Q-learning, we may use an update like:

$$Q_{i+1}(x,u) := Q_i(x,u) + \alpha\left(\rho(x,u) + \gamma \max_{\vartheta}\left\{\mathbb{E}_{U \sim p_U^{\vartheta}}\left[Q_i(X_+^U, U) \mid x\right]\right\} - Q_i(x,u)\right), \ \forall x,u$$

and so ϑ is a „blind" variable, hence no dependence on ϑ^i (the learning is over a greedy, maximizing policy).

Another way is to use a policy different to the target one, also called behavior policy.

A particular option is an ε-greedy policy that picks a random action with probability $1-\varepsilon$, but otherwise is greedy.

However, a behavior policy may introduce a bias in the sampled estimate of the expected value.

Importance sampling may remedy this in turn.

Actor-critic

Of course, TD methods can be formulated in terms of approximations of the value and advantage functions. Also, TD(0) is just a particular way of updating the value. Say, in TD(N), the update target may read:

$$O^{\vartheta,N}(x,u) := \rho(x,u) + \mathbb{E}_{\{U_\bullet\}\sim\{\mathbb{P}^{\vartheta}_{U_\bullet}\}}\left[\sum_{k=1}^{N-1} \gamma^k \rho(X_{+k}, U_{+k}) + \gamma^N F(\bullet)\right],$$

where „+k" means kth rollout from the current step, F may be taken as Q, V, A with according arguments

Actor-critic

(cont.)

Using this update target, we can formulate one for TD(λ):

$$O^{\vartheta,\lambda}(x,u) := (1-\lambda)\sum_{k=1}^{N-1} \lambda^{k-1} O^{\vartheta,k}(x,u) + \lambda^{N-1}\sum_{j=1}^{N} \gamma^{j-1} \mathbb{E}_{\{U_\circ\}\sim\{P^{\vartheta}_{U_\circ}\}}\left[\rho(X_{+j}, U_{+j})\right]$$

As always, those targets may be taken in on- and off-policy fashion.

One immediate difference in Monte-Carlo approach to the value updates we have just considered is that, in the former, we sample trajectories, whereas in the latter, we perform iterations over the whole (mesh in) state space.

Such approaches are offline, i.e., they are not performed time step to time step along a trajectory. But we can make TD methods online using function approximators, such as artificial neural networks (NN)

Instead of traking a Q-function approximation as a table of the respective values, we can track parameters of an approximator.
The key idea of this approach comes from the learning theory in the hope that there be a model of lower dimension than there are data points.
Thus, we can design $\hat{V}_\theta(x)$ or $\hat{Q}_\theta(x,u)$ as NNs with weights θ as approximators of the value and Q-function, respectively.
Such an approximator is also referred to as critic, whereas the (parametrized) policy plays as actor

This would transform the value update into the following:

$$\theta_k := \arg\min_{\theta} J_k^c(\theta),$$

where k is the current time step and J_k^c is the critic loss at k.

Now, a great variety of such loss functions is possible. For instance, TD(0)-like for $\hat{Q}$:

$$e_{\theta_{k-1}}^{\vartheta_k}(\theta \mid x_k, u_{k-1}) := \rho(x_k, u_{k-1}) + \gamma \mathbb{E}_{U \sim \rho_U^{\vartheta_k}}\left[\hat{Q}_{\theta_{k-1}}(X_{k+1}^U, U) \mid x_k\right] - \hat{Q}_{\theta_{k-1}}(x_k, u_{k-1}),$$

and we may take $J_k^c(\theta) := \frac{1}{2}\left(e_{\theta_{k-1}}^{\vartheta_k}(\theta \mid x_k, u_{k-1}\right)^2$

(cont.)

... or for $\hat{V}$:

$$e^{\vartheta_k}_{\theta_{k-1}}(\theta \mid x_k, u_k) := \rho(x_k, u_k) + \gamma \mathbb{E}_{\upsilon \sim P^{\vartheta_k}_{\upsilon}}\left[\hat{V}_{\theta_{k-1}}(X^{\upsilon}_{k+1}) \mid x_k\right] - \hat{V}_{\theta_{k-1}}(x_k).$$

A few remarks:
we assumed a VI style of update, i.e., we are defining the critic weights for next step in which the actor weights will be updated first. We also used one rollout, but instead we could use a buffer also known as experience replay

(cont.)

In this case, the critic loss would read, for instance:

$$J_k^c(\theta) := \frac{1}{2} \sum_{(x,u) \in \mathcal{X} \times \mathcal{U}} \left(e_{\theta_{k-1}}^{\vartheta_k}(\theta \mid x, u) \right)^2 ,$$

where $\mathcal{X} \times \mathcal{U}$ is an experience replay of past states and actions.

With such a critic loss, the learning is not only online, but also model-free.

Surely, other kinds of losses are possible, e.g., Huber loss

Actor-critic

Let us summarize the described online, model-free actor-critic Q-learning:

1. Initialize $\vartheta_0, x_0, u_0, M = |\mathcal{X}| = |\mathcal{U}|$
2. Optionally perform an initial exploration stage and fill the experience replay $\mathcal{X} \times \mathcal{U}$
3. For the subsequent steps indexed k do:
   3.1. Push (x_k, u_{k-1}) into $\mathcal{X} \times \mathcal{U}$
   3.2. Policy update
   $$\vartheta_k := \arg\max_{\vartheta} \mathbb{E}_{U \sim P_U^{\vartheta}} \left[\hat{Q}_{\theta_{k-1}}(x_k, U) \right]$$
   3.3. Value update
   $$\theta_k := \arg\min_{\theta} \frac{1}{2} \sum_{(x,u) \in \mathcal{X} \times \mathcal{U}} \left(e_{\theta_{k-1}}^{\vartheta_k}(\theta \mid x, u) \right)^2$$
   3.4 Sample u_k from $P_U^{\vartheta_k}$ and apply to the system

Actor-critic

Notice that using squared TD as the loss is handy to update the critic weights by gradient descent:

$$\theta_k^{i+1} := \theta_k^i - \alpha \sum_{(x,u) \in \mathcal{X} \times \mathcal{U}} e_{\theta_{k-1}}^{\mathcal{Q}_k}(\theta | x, u) \nabla_\theta e_{\theta_{k-1}}^{\mathcal{Q}_k}(\theta | x, u).$$

Of course, other optimization methods are possible, including least squares.

The gradient $\nabla_\theta e_{\theta_{k-1}}^{\mathcal{Q}_k}(\theta | x, u)$ can systematically be computed via back propagation in case of a deep NN critic

Actor-critic

Evidently, the described method of critic learning can be employed in policy gradient methods. The weights can be trained on the sampled trajectories required in policy gradient methods. In fact, past samples also go (cf. experience replay) which gives another variant of off-policy learning, i.e., we don't need rollouts using the target policy. A further option is to take samples from the experience replay and thereby perform mini-batch critic updates

Actor-critic

As we said, sky is the limit when designing actor-critics. For instance, learning the V or Q directly is just one option. We can learn ∇V (called the co-state), ∇Q or both etc.
Here is so-called classical menu due to Werbos, Miller and Sutton:

Heuristic DP (HDP) : learn V
Action-dependent HDP (ADHDP) : learn Q
Dual heuristic programming (DHP) : learn ∇V
Action-dependent DHP (ADDHP) : learn ∇Q
Globalized (AD) DHP : learn value and co-state

Extras:

- multiple actors:
  A3C — asynchronous actor updates
  A2C — synchronous actor updates
- deep Q-learning (or deep Q-network, DQN):
  pretty much the same as what we described above, on-line, off-policy, experience-replay-based (mini-batch mode), but stresses $\hat{Q}$ as a deep NN

(cont.)

- Double Q-learning:

  use two Q-function approximators and mix them up via the update targets:

$$Q^A_{i+1}(x,u) := Q^A_i(x,u) + \alpha\left(\rho(x,u) + \gamma\, \mathbb{E}_{U\sim P_U^{\vartheta_i^A}}\left[Q^B_i(X^U_+, U) \,\middle|\, x\right] - Q^A_i(x,u)\right),$$

$$Q^B_{i+1}(x,u) := Q^B_i(x,u) + \alpha\left(\rho(x,u) + \gamma\, \mathbb{E}_{U\sim P_U^{\vartheta_i^B}}\left[Q^A_i(X^U_+, U) \,\middle|\, x\right] - Q^B_i(x,u)\right),$$

where

$$\vartheta^A_i := \arg\max_{\vartheta} \mathbb{E}_{U\sim P_U^{\vartheta}}\left[\hat{Q}^A_i(X^U_+, U)\right],$$

$$\vartheta^B_i := \arg\max_{\vartheta} \mathbb{E}_{U\sim P_U^{\vartheta}}\left[\hat{Q}^B_i(X^U_+, U)\right]$$

Actor-critic

(cont.)

- Deep deterministic policy gradient (DDPG):
  „DPG + DQN"

- Soft Q-learning:
  add an entropy term into the reward to stimulate exploration ⇒ „soft" Q-function

- Soft actor-critic (SAC):
  a modification of soft Q-learning with additional training of a soft value function