Dual heuristic programming

D.D.Kuznedelev (Skoltech)

September 24, 2021

A simple optimal control problem can be stated as follows:

$$\max J = \int_0^T \rho(t, x, u)dt \tag{1}$$

$$\text{subject to } \dot{x} = f(t, x, u) \tag{2}$$

$$x(0) = x_0, x(T) - \text{free} \tag{3}$$

$$u(t) \in \mathbb{U} \ \forall t \in [0, T] \tag{4}$$

The maximun principle

First order (necessary condition) is known as *Pontryagin's maximum principle*. One introduces the Hamiltonian function $H(t, x, u, \lambda) = \rho(t, x, u) + \lambda(t) f(t, x, u)$
Here $\lambda(t)$ is the *costate* variable.
The necessary conditions for maximum read as follows:

- $\max_u H(t, x, u, \lambda)\ \forall t \in [0, T]$
- $\dot{x} = \frac{\partial H}{\partial \lambda}$
- $\dot{\lambda} = -\frac{\partial H}{\partial x}$
- $\lambda(T) = 0$ - transversality condition

Costate equation

$$\frac{d\lambda}{dt} = -\frac{\partial H}{\partial x} \tag{5}$$

Costate variables $\lambda(t)$ have the interpretation of Lagrange multipliers, associated with state equations. Costate variables represent the cost of violating the constraints in optimization problem.

There are approaches to adapt Critic networks, based on the gradient with respect to state of the value function

Agent and environment interact as follows at each step k:

1. Estimate of state x_k becomes available
2. Agent invokes the action network to calculate the action $u_k = \mathcal{A}(x_k)$ and the reward $\rho(x_k, u_k)$ is calculated.
3. Action u_k is transmitted to the enviroment

There is also a state network for estimating the state:

$$\hat{x}_{k+1} = \mathcal{S}(x_k, u_k)$$

DHP

DHP is based on differentiating the Bellman equation:

$$V(x_k) = V(x_k, u_k(x_k)) + \mathbb{E}[V(x_{k+1})]$$

Taking the derivative with respect to x_k and applying the chain rule, one gets:

$$\lambda_i(x_k) \triangleq \frac{dV(x_k)}{dx_k^i} = \frac{\partial}{\partial x_k^i}\rho(x_k, u_k) + \mathbb{E}\left[\frac{\partial V(x_{k+1})}{\partial x_k^i}\right] = \frac{\partial \rho(x_k, u_k)}{\partial x_k^i} + \sum_j \frac{\partial \rho(x_k, u_k)}{\partial u_k^j}\frac{\partial u_k^j(x_k)}{\partial x_k^i} +$$
$$+ \sum_j \mathbb{E}\left[\frac{\partial V(x_{k+1})}{\partial x_{k+1}^j}\frac{\partial x_{k+1}^j}{\partial x_k^i}\right] + \sum_{j,l} \mathbb{E}\left[\frac{\partial V(x_{k+1})}{\partial x_{k+1}^j}\frac{\partial x_{k+1}^j}{\partial u_k^l}\frac{\partial u_k^l}{\partial x_k^i}\right] \quad (6)$$

These chain rule calculations are carried out via backpropagation through the actor and critic network.

Implementation of DHP

DHP is a procedure for adapting a critic (network or function) $\hat{\lambda}(x_k)$ to approximate the function $\lambda(x_k)$ defined in (6). DHP can use any real-time supervised learning method for the learning itself. Backpropagation is only needed to calculate the target λ^*. The algorithm reads as follows:

1. Get x_k, u_k, x_{k+1}. x_{k+1} can be predicted from $\mathcal{S}(x_k, u_k)$ or taken from the next time step.
2. Calculate $\hat{\lambda}(x_k) = \hat{\lambda}^\theta(x_k)$ and the target $\lambda^*(x_k)$ by (6).
3. Update θ based on inputs x_k and target vector $\lambda^*(x_k)$

Implementation of DHP

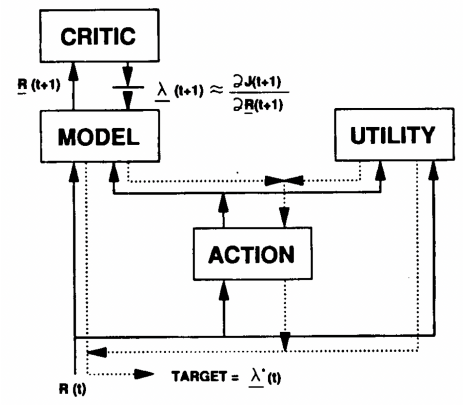

Figure: Calculation of targets λ^* in DHP

References

- Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches
  https://www.werbos.com/HICChapter13.pdf
- ECON 402 : Optimal control theory
  https://people.stfx.ca/tleo/advmacrolec3.pdf